GE VERNOVA

PROFICY® SOFTWARE & SERVICES

# PROFICY HISTORIAN

## User Guide

GE VERNOVA

**Proprietary Notice**
The information contained in this publication is believed to be accurate and reliable. However, GE Vernova assumes no responsibilities for any errors, omissions or inaccuracies. Information contained in the publication is subject to change without notice.

No part of this publication may be reproduced in any form, or stored in a database or retrieval system, or transmitted or distributed in any form by any means, electronic, mechanical photocopying, recording or otherwise, without the prior written permission of GE Vernova. Information contained herein is subject to change without notice.

© 2024 GE Vernova and/or its affiliates. All rights reserved.

**Trademark Notices**
"GE VERNOVA" is a registered trademark of GE Vernova. The terms "GE" and the GE Monogram are trademarks of the General Electric Company, and are used with permission.

Microsoft® is a registered trademark of Microsoft Corporation, in the United States and/or other countries.

All other trademarks are the property of their respective owners.

We want to hear from you. If you have any comments, questions, or suggestions about our documentation, send them to the following email address:
doc@ge.com

# Contents

**Chapter 1. Release Notes** ..... **47**

What's New ..... 47

Resolved Issues ..... 55

Known Issues and Limitations ..... 60

**Chapter 2. Getting Started Guide** ..... **65**

Overview ..... 65

System Architecture ..... 69

System Components ..... 73

About the Historian Server ..... 80

About Tags ..... 81

Prerequisites ..... 82

Setting Up the Historian Environment ..... 82

Activate the Historian License ..... 82

Hardware Requirements ..... 87

Software Requirements ..... 92

Compatibility with Other Products ..... 94

Supported Formats ..... 94

Optimize Performance ..... 96

Enable Trust ..... 97

VMWare Support ..... 98

Installation ..... 100

Installation Workflow ..... 100

The Historian Server ..... 104

Alarms and Events ..... 140

Collectors ..... 142

Client Tools ..... 150

Web-based Clients ..... 154

Remote Management Agents....187
Install the OPC UA HDA Server....191
The Excel Add-In for Historian....195
The Excel Add-In for Operations Hub....198
ETL....202
Stand-Alone Help....205
Configuring Certificate-based Security to use MTLS....206
Overview of the Certificate-based Security....206
Certificate-based Security Specifc Configurations....214
Upgrade Scenarios when Working with Historian Server Certificates....227
Troubleshooting Historian Server Certificates....228
Using Proficy Authentication or LDAP Groups....229
About Proficy Authentication....229
About Proficy Authentication Groups....231
Using Server Certificates....234
Change the Log Levels of Proficy Authentication....234
Migrating Historian Data....235
Migrating the Alarms and Events Data....235
Using the Migration Tool....239
Data Migration Scenarios....242
Migration Tool Command-Line Syntax....246
Interoperability of Historian Versions....247
Migrate User Authentication Data from Historian to Common Proficy Authentication Service....248
Implementing Historian Security....251
Implementing Historian Security....251
Uninstalling Historian....294
Uninstalling Historian....294
Troubleshooting....295

Managing Historian Log Files....295
Troubleshooting the Historian Server....298
Troubleshooting Web-based Clients....300
**Chapter 3. Configuration Hub....303**
Overview....303
About Configuration Hub....303
Workflow....304
Setting up Configuration Hub....305
About Setting up Configuration Hub....305
Install the Historian Server....305
Install Web-based Clients....314
Install Collectors....331
Perform Post-Installation Tasks....335
Upgrade....336
Access Configuration Hub....336
Historian Plugin Management in Configuration Hub....341
Common Tasks....353
Setting up a Stand-Alone System....356
About Setting up....356
Add a Collector....357
Add Tags....357
Setting up a Horizontally Scalable System....359
About Setting up a Horizontally Scalable System....359
Add a Server....361
Add a Collector....362
Add Tags....363
Browse Tags using Distributed or Mirror Node Servers when Primary Server is Inactive....365
Setting up High Availability....367
About Data Mirroring....367

Create....368
Create a Data Store....370
Creating a Model....371
About a Historian Model....371
About Object Templates....376
Workflow for Creating a Historian Model....379
Create an Object Type....380
Include a Contained Type....385
Create an Object Instance....388
Provide Data for a Static Variable....390
Collect Data for a Direct Variable....392
Collect Data for an Indirect Variable....395
Export an Object Type/Instance....400
Import an Object Type/Instance....402
Copy an Object Type....403
Delete a Template....407
Delete an Object Instance....409
Delete an Object Type....410
Managing Historian Systems....411
Access a System....411
Access the Collectors in a System....420
Access Offline Configuration Collectors....422
Access the Tags in a System....425
Add a System....427
Add a Server....428
Set Up a Mirror of Mirror....429
Remove a Server....439
Set a Default Location....439
Modify a System....440

Configure Advanced Settings..... 441
Configure Labels of Spare Fields..... 444
Set a Default System..... 445
Delete a System..... 445
Managing Mirror Locations..... 446
Create..... 446
Rename..... 447
Add a Machine..... 448
Remove a Machine..... 449
Delete..... 450
Managing Data Stores..... 451
About Data Stores ..... 451
Create a Data Store..... 452
Access a Data Store..... 453
Rename a Data Store..... 458
Set as Default..... 462
Access Archives..... 463
Apply Configuration Template..... 463
Multiple Archive Paths..... 464
Access Activity Logs..... 470
Access Tags..... 470
Specify Tags for Data Collection..... 471
Add a Tag Manually..... 473
View Performance..... 476
Delete..... 478
Adding a Collector Instance..... 479
The Calculation Collector..... 479
CygNet Collector..... 482
The File Collector..... 486

The HAB Collector ... 489
About Adding an iFIX Collector Instance ... 501
The iFIX Collector ... 505
The MQTT Collector ... 510
The MQTT Sparkplug B Collector ... 519
The ODBC Collector ... 527
The OPC Classic Alarms and Events Collector ... 532
The OPC Classic DA Collector ... 534
The OPC Classic HDA Collector ... 540
The OPC UA DA Collector ... 544
The OSI PI Collector ... 549
The OSI PI Distributor ... 553
The Python Collector ... 556
The Server-to-Server Collector ... 559
The Server-to-Server Distributor ... 563
The Simulation Collector ... 567
The Windows Performance Collector ... 571
The Wonderware Collector ... 575
Collector Configuration - Common Fields ... 579
Sending Data to Cloud ... 587
Alibaba Cloud ... 587
AWS Cloud ... 594
Azure Cloud (Key-Value Format) ... 601
Azure Cloud (KairosDB Format) ... 608
Google Cloud ... 614
Predix Cloud ... 621
Protocols and Port Numbers ... 626
Managing Collector Instances ... 627
Managing Collectors Using Configuration Hub ... 627

Access a Collector....628
Access Tags....631
Add a Collector....632
Enable MTLS Security for Collectors....632
Modify a Collector....634
Add a Comment....635
Access Comments....636
Start a Collector....637
Stop a Collector....638
Restart a Collector....639
Pause Data Collection....641
Resume Data Collection....642
Clear Buffer....643
Move Buffer....644
Change Destination....645
Reset Performance Counters....646
Reset Overruns....647
Update Collector Credentials....648
Apply Configuration Template to a Collector....650
Configure Collector Redundancy....651
Delete a Collector....654
Managing Offline Configuration Collector Instances....655
Access Offline Configuration Collectors....655
Manage Offline Configuration Collectors....657
Access Tags....658
Managing Tags....659
About Tags....659
About Array Tags....660
About Collector and Archive Compression....660

About Scaling....666
About Condition-Based Collection....666
Specify Tags for Data Collection....667
Add a Tag Manually....669
Access a Tag....672
Configure Multiple Tags....687
Access Trend Chart....689
Access Last 10 Values....690
Access a Tag Alias....694
Export Tags as a CSV File....695
Import Tags from a CSV File....697
Rename a Tag....698
Copy a Tag....700
Stop Data Collection....702
Resume Data Collection....703
Remove a Tag....704
Delete a Tag....705
Managing Enumerated Sets....707
About Enumerated Sets....707
Create an Enumerated Set....708
Assign an Enumerated Set to a Tag....710
Export an Enumerated Set....710
Import an Enumerated Set....711
Rename Enumerated Set....711
Delete Enumerated Set....712
Managing Data Attribute Enumerated Set....712
About Data Attribute Enumerated Set....712
Create a Data Attribute Enumerated Set....713
Assign a Data Attribute Enumerated Set to a Tag....716

Export Data Attribute Enumerated Sets....717
Import Data Attribute Enumerated Sets....717
Rename a Data Attribute Enumerated Set....718
Delete a Data Attribute Enumerated Set....718
Managing User-Defined Data Types....719
About UDTs....719
Create UDT....719
Assign to Tag....720
Export User-defined Types....721
Import User-defined Types....721
Rename User-defined Types....722
Delete User-defined Types....722
Managing Archives....723
About Archives....723
Guidelines for Archive Sizing....724
Access an Archive....725
Create Archives Automatically....727
Create Archives Manually....728
Back up an Archive....730
Back up Archives with Volume Shadow Copy Service....731
Restore an Archive....733
Close an Archive....734
Remove an Archive....734
Reading/Writing Data....734
Query Data....734
Write Data....738
About Saved Query....741
Managing Alarms and Events....750
About Alarms and Events....750

Requirements ... 750
Create an Alarm ... 751
Access/Filter Alarms ... 752
Back up Alarms ... 753
Restore Alarms ... 754
About Purging Alarms ... 754
Managing Configuration Templates ... 757
About Configuration Templates ... 757
Create a Configuration Template for Collectors ... 757
Apply Configuration Template to a Collector ... 765
Create a Configuration Template for Data Stores ... 767
Apply the Configuration Template to a Data Store ... 772
Managing Reports ... 773
About Reports ... 773
Generate Reports ... 774
Export the Generated Report as a CSV File ... 781
Save the Generated Report as a PDF File ... 781
Accessing Activity Logs ... 782
Troubleshooting Configuration Hub ... 784
**Chapter 4. Remote Collector Management ... 789**
Overview of Remote Collector Management ... 789
Installing Remote Management Agents ... 790
Install Using the Installer ... 790
Install at a Command Prompt ... 793
About Managing Collector Instances Using the RemoteCollectorConfigurator Utility ... 794
Create a Sample JSON File ... 796
Add a Collector Instance ... 797
Modify a Collector Instance ... 799
Collector Instance Parameters ... 801

General Parameters of a Collector....814
Delete a Collector Instance....817
Add an Offline Collector Instance....819
Delete an Offline Collector Instance....819
Manage a Collector Remotely....820
Troubleshooting Remote Collector Management Issues....820
**Chapter 5. Using Historian Administrator....822**
Historian Administrator....822
Overview....822
Access Historian Administrator....823
Historian in a Regulated Environment....824
High Availability....825
About High Availability....825
Enable High Availability....827
Register Historian with a Cluster....827
Historian Administrator - Pages....828
The Main Page....828
The Data Store Page....835
Searching in Message Panel....850
Managing Data Stores....851
About Data Stores....851
Create....852
Rename....855
Move Tags....857
Delete....860
Managing Archives....863
About Archives....863
About Remote Storage....864
Guidelines for Archive Sizing....866

Create Automatically.... 868
Create Manually.... 870
Back up Automatically....874
Back up Manually.... 876
Restore....881
Configure System File Cache Memory....883
Configure DA for Remote Storage....884
Reuse a Config File....884
Managing Tags.... 885
About Tags.... 885
About Array Tags.... 886
About Collector and Archive Compression....886
About Scaling.... 893
About Condition-Based Collection....893
Access/Modify.... 894
Add Tags from Source.... 906
Create Manually.... 909
Copy.... 912
Rename....915
View Trend Chart.... 918
View Last 10 Values....921
Stop Data Collection....924
Resume Data Collection....929
Get Tag Fields.... 934
Remove.... 934
Delete.... 938
About UDTs.... 941
Manage UDTs....942
Assign/Remove Tags from UDTs.... 947

Manage Enumerated Sets....951
Assign/Remove Tags from Enumerated Sets....955
Managing Collectors....959
About Collectors....959
Access/Modify a Collector....960
Delete a Collector....968
Enable Spike Logic....970
About Collector Redundancy....975
Configure Redundant Collectors....978
Maintaining, Operating, and Monitoring Historian....983
Maintain, Operate, and Monitor Historian....983
Data Types....983
Plan for Data Recovery....988
Develop a Maintenance Plan....989
Evaluate and Control Data Compression....991
Handle Value Step Changes with Collector Data Compression....992
Reviewing System Alerts and Messages....995
Monitor Historian Health and Status....995
Monitor Historian Performance....999
Troubleshooting....1009
Solve Minor Operating Problems....1009
FAQ: Run a Collector as a Service....1010
Changing the Base Name of Automatically Created Archives....1011
Configuring the Inactive Timeout Value....1012
Configuring Deep Data Tree Warnings....1012
Control Data Flow Speeds....1013
Configure Inactive Server Reset Timeout....1016
Historian Errors and Message Codes....1016
Determining the Version of the Historian Server....1019

Return a List of Valid Field Options ..... 1020
Scheduled Software Performance Impact ..... 1020
Intellution 7.x Drivers as OPC Servers ..... 1020
Troubleshooting Failed Logins ..... 1021
Troubleshoot Data Collector Configuration ..... 1022
Troubleshoot Tags ..... 1023
Troubleshoot Historian Performance ..... 1024
Troubleshoot the Archive Service ..... 1025
**Chapter 6. Historian Advanced Topics ..... 1027**
Historian Advanced Topics Overview ..... 1027
About Historian Advanced Topics ..... 1027
Storage ..... 1027
Archive Compression ..... 1027
Determining Whether Held Values are Written During Archive Compression ..... 1032
Determining Expected Value ..... 1033
How Archive Compression Timeout Works ..... 1034
Archive De-fragmentation - An Overview ..... 1035
About Storing Future Data ..... 1037
Retrieval ..... 1044
Retrieval ..... 1044
Sampling Modes ..... 1044
About Retrieving Data from Historian ..... 1071
Sampling Modes ..... 1072
Calculation Modes ..... 1073
Query Modifiers ..... 1075
Filtered Data Queries ..... 1076
Filter Parameters for Data Queries ..... 1076
Filtered Queries in the Excel Add-in Example ..... 1078
Filtering Data Queries in the Excel Add-in ..... 1079

Hybrid Modes....1079
Calculation Modes....1085
StepValue Tag Property....1152
Comment Retrieval Mode....1158
Query Modifiers....1159
Work with Data Stores from the Command Line....1186
Using the Command Line to Work with Data Stores....1186
Creating a Data Store....1186
Deleting a Data Store....1186
Examples....1187
Measuring Historian Performance....1187
About Measuring Performance of Proficy Historian....1187
About the Proficy Historian Overview Objects....1188
About Proficy Historian Message Queue Object....1192
About Proficy Historian Cache Object....1196
**Chapter 7. Historian Alarms and Events....1200**
Overview....1200
Requirements....1200
Installation....1201
Upgrade....1203
Changing the SQL Server....1203
Backing up Data....1204
Using Configuration Hub....1204
Using the Utility....1205
Using Historian Administrator....1206
At a Command Prompt....1208
Restoring Data....1209
Using Configuration Hub....1209
Using Historian Administrator....1209

Migrating Data....1210
Using the Utility....1211
At a Command Prompt....1212
Querying Data....1215
Importing Data....1216
Exporting Data....1217
Purging Data....1218
Using Configuration Hub....1220
Using the Utility....1221
At a Command Prompt....1223
Closing Alarms....1225
Using the OPC AE Collector with FIX32 SCADA Systems....1225
**Chapter 8. Historian REST APIs....1227**
Introduction to Historian REST APIs....1227
Historian APIs....1227
About Security and Authentication....1227
Standards....1229
API Methods....1230
API Status Messages....1230
Common API Parameters....1231
Overview of Commonly Used API Parameters....1231
TagNames Parameter....1231
Start and End Timestamps Parameter....1232
TagSamples Parameter....1232
DataSample Parameter....1234
SamplingModeType Parameter....1235
Direction Parameter....1237
CalculationModeType Parameter....1237
FilterModeType Parameter....1243

ReturnDataFields Parameter....1244
Payload Parameter....1245
Error Code Definitions....1251
Historian REST APIs....1254
Overview of the Historian REST APIs....1254
Managing Systems....1254
Managing Historian Model....1287
Managing Collector Instances....1329
Collector Type and Subtype....1339
Managing Collectors....1340
Managing Data Stores....1365
Managing Tags....1381
Managing Alarms and Events....1417
Swagger Documentation....1427
**Chapter 9. Historian System API....1429**
Overview of the Historian System API....1429
Overview of the Historian System API....1429
ihapi.h File Overview....1430
ihapi.h File Overview....1430
ihConfiguration Functions....1433
System API Programming....1435
System API Programming....1435
System API Functions....1436
System API Connect Functions....1436
System API Tag Functions....1443
Read and Write Functions....1455
Archiver Configuration Functions....1469
Archiver Backup/Restore Functions....1479
User Defined Type Functions....1488

Utility Functions.... 1495
Sample Programs.... 1512
Sample Programs.... 1512
**Chapter 10. Historian User API.... 1513**
Historian User API Overview.... 1513
About the Historian User API.... 1513
Prerequisites.... 1514
Connect Functions.... 1514
Connect Functions Overview.... 1514
ihuConnect.... 1514
ihuConnectEx.... 1516
ihuDisconnect.... 1517
ihuSetConnectionParameters.... 1517
ihuRestoreDefaultConnectionParameters.... 1519
ihuServerRegisterCallbacks.... 1519
ihuBrowseCollectors.... 1520
Archiver Functions.... 1521
Archiver Functions Overview.... 1521
ihuSetArchiverProperty.... 1521
ihuGetArchiverProperty.... 1523
Tag Functions.... 1523
Tag Functions Overview.... 1523
Tag Property Value Types.... 1525
ihuCreateTagCacheContext.... 1527
ihuFetchTagCache.... 1527
ihuFetchTagCacheEx.... 1528
ihuFetchTagCacheEx2.... 1528
ihuFetchTagCacheEx3.... 1529
ihuGetTagNameCacheIndex.... 1530

ihuGetTagNameCacheIndexEx2 1530
ihuGetNumericTagPropertyByTagname 1531
ihuGetNumericTagPropertyByIndex 1531
ihuGetNumericTagPropertyByIndexEx2 1532
ihuGetStringTagPropertyByTagName 1533
ihuGetStringTagPropertyByTagNameEx2 1533
ihuGetStringTagPropertyByIndex 1534
ihuGetStringTagPropertyByIndexEx2 1535
ihuTagAdd 1535
ihuTagDelete 1536
ihuTagDeleteEx 1537
ihuTagRename 1537
ihuTagRenameEx 1538
ihuTagCacheCriteriaClear 1539
ihuTagCacheCriteriaClearEx2 1539
ihuTagCacheCriteriaSetStringProperty 1540
ihuTagCacheCriteriaSetStringPropertyEx2 1540
ihuTagCacheCriteriaSetNumericProperty 1541
ihuTagCacheCriteriaSetNumericPropertyEx2 1541
ihuTagClearProperties 1542
ihuTagSetStringProperty 1542
ihuTagSetNumericProperty 1542
ihuCloseTagCache 1542
ihuCloseTagCacheEx2 1543
Write Functions 1543
Write Functions Overview 1543
ihuWriteData 1543
ihuWriteComment 1546
Query Modifiers Functions 1547

Query Modifiers Functions Overview.... 1547
ihuBrowseQueryModifiers.... 1547
ihuClearQueryModifiers.... 1548
ihuRetrieveCalculatedDataEx2.... 1548
ihuSetQueryModifiers.... 1549
Read Functions.... 1549
Read Functions Overview.... 1549
ihuReadCurrentValue.... 1550
ihuReadInterpolatedValue.... 1551
ihuReadInterpolatedValueEx.... 1552
ihuReadRawDataByTime.... 1553
ihuReadRawDataByTimeEx.... 1554
ihuReadRawDataByCount.... 1555
ihuReadRawDataByCountEx.... 1556
ihuReadMultiTagRawDataByCount.... 1557
ihuReadMultiTagRawDataByCountEx.... 1557
ihuRetrieveSampledData.... 1558
ihuRetrieveSampledDataEx.... 1559
ihuRetrieveSampledDataEx2.... 1560
ihuRetrieveCalculatedData.... 1561
ihuRetrieveCalculatedDataEx.... 1562
ihuRetrieveCalculatedDataEx3.... 1563
Utility Functions.... 1564
Utility Functions Overview.... 1564
IHU_timestamp_FromParts.... 1564
IHU_timestamp_ToParts.... 1565
ihuServerGetTime.... 1566
Enumerated Sets Functions.... 1566
Enumerated Sets Functions Overview.... 1566

ihuGetEnumeratedSets....1567
ihuEnumeratedSetAdd....1567
ihuEnumeratedSetRawValue....1568
ihuEnumeratedSetsFree....1569
ihuEnumeratedSetRename....1570
ihuEnumeratedSetDelete....1570
ihuEnumeratedStateAdd....1571
ihuEnumeratedStateModify....1572
ihuEnumeratedStateDelete....1572
User-Defined Type Functions....1573
User-Defined Type Functions Overview....1573
ihuUserDefinedTypeAdd....1574
ihuUserDefinedTypeDelete....1574
ihuUserDefinedTypeRename....1575
ihuUserDefinedTypeExists....1576
ihuGetUserDefinedTypes....1576
ihuUserDefinedTypeSetProperties....1577
ihuUserDefinedTypeFreeProperties....1577
Publish Functions....1578
Publish Functions Overview....1578
Historian User API Error Codes....1586
Error Codes....1586
Historian User API Sample Programs....1588
Sample Programs Overview....1588
**Chapter 11. Historian SDK....1591**
Object Model Overview....1591
Historian SDK Overview....1591
Connect the SDK to the Server....1593
Working with Blob Data....1594

Working with Archives....1598
SDK Reference....1598
Object Summary....1598
Alarms Object....1598
Archive Object....1604
Archives Object....1604
Alarms.PurgeAlarmsById....1604
Collector Object....1605
Collectors Object....1605
Data Objects....1605
Message Objects....1606
OPC Objects....1607
Server Objects....1607
Tag Objects....1608
UserCalcFunction Object....1609
Property Reference A-B....1609
Method Reference A-B....1803
Event Reference A-Z....1889
**Chapter 12. Historian Utility Suite....1894**
Alerting System....1894
Version Validator....1895
DiagnostiX Tool....1895
**Chapter 13. Collector Tool Kit....1897**
Collector Toolkit Overview....1897
Overview....1897
Prerequisites....1897
Prerequisites....1897
Installing the Collector Toolkit with Historian....1899
About Installation....1899

Creating the Custom Collector Using the Wizard..... 1899
Installing and Configuring the Collector Toolkit for Linux.....1900
Installing and Configuring the Collector Toolkit for Linux..... 1900
Configuring Custom Collector Wizard..... 1901
Configuring a Custom Collector using the Wizard.....1901
Creating a Custom Collector..... 1901
Changing Historian Server Name..... 1902
Changing the Historian Server Name Using Registry..... 1902
Working with collector Interfaces..... 1903
About Interfaces..... 1903
Custom Collector Design.....1905
Design topics for Creating Custom Collectors.....1905
Backward Compatibility of the Collector Toolkit..... 1906
Backward Compatibility of the Collector Toolkit.....1906
Custom Collector Toolkit Interface Technical Reference.....1906
Custom Collector Toolkit Interface Technical Reference..... 1906
Custom Structure Technical Reference.....1925
What is a Structure?.....1925
Custom Collector Toolkit Structure Reference.....1926
Hierarchical Custom Controller Browsing..... 1976
Browsing Custom Controller in a Hierarchy..... 1976
Developing Hierarchical Browsing using Collector Toolkit .....1978
Collector Initialization Callbacks..... 1980
Polled Tag Callbacks..... 1982
Unsolicited Tags Callbacks.....1984
**Chapter 14. Data Collectors - General..... 1990**
Data Collectors Overview..... 1990
About Historian Data Collectors.....1990
Bi-Modal Cloud Data Collectors..... 1993

Data Collector Software Components.... 1995
Supported Windows versions for Data Collectors.... 1995
Data Collector Functions.... 1996
Supported Acquisition Interfaces.... 1998
Best Practices for Working with Data Collectors.... 1998
About Installing Historian Data Collectors.... 1999
Install Collectors.... 1999
Installing a Collector at a Command Prompt.... 2003
Upgrade Collectors.... 2005
Sending Data to Cloud.... 2006
Alibaba Cloud.... 2006
AWS Cloud.... 2012
Azure Cloud.... 2019
Google Cloud.... 2026
Predix Cloud.... 2033
Protocols and Port Numbers.... 2038
Offline Collector Configuration.... 2039
Offline Configuration for Collectors.... 2039
Creating Offline Configuration XML file.... 2040
Collector Interface Properties.... 2041
Tag List and Tag Properties.... 2043
About Updating Tag Properties Dynamically.... 2048
Cloud Collector Specific Registry Configuration.... 2049
Working with Tags.... 2050
Understanding Tag names.... 2050
Adding Tags from a Collector.... 2052
Manually Adding Tags.... 2056
Copy a Tag.... 2060
Search for Tags.... 2061

Remove Tags....2063
Browse a Data Source for New Tags....2064
About Configuring Collector Options....2066
About Collector Redundancy....2067
Collect Vendor Attributes....2067
Collector Spare Configuration....2069
Data Collector Operation and Troubleshooting....2069
Data Collector File Locations....2069
Pause or Resume Data Collection for All Tags....2070
Pause Data Collection for a Subset of Tags....2071
Modify User Privileges for Starting a Collector....2072
About Monitoring Data Collector Performance Statistics....2072
Disabling Rebroadcasting for Historian Data Archiver....2073
Troubleshooting Tag Configuration....2074
Reviewing the Active Collector Configuration....2074
Collector and Archive Compression....2076
Data Buffering....2076
Editing the Registry to Change the Buffer Size....2077
Setting Up Services Recovery Actions in Windows....2077
Working with Python Expression Tags....2078
Python Expression Tags in Historian....2078
Constructing and Adding Python Expression Tags....2080
Python Expression Tag Examples....2084
Uninstall Collectors....2094
Troubleshooting....2094
**Chapter 15. The Calculation Collector....2098**
Overview....2098
About the Tags Used by the Calculation Collector....2099
Workflow for Using the Calculation Collector....2100

Configuration....2101
Recalculate Tag Values....2103
Using Configuration Hub....2103
Using Historian Administrator....2104
Using the Calculation Collector....2106
Write Data to an Arbitrary Tag....2106
Creating Triggers....2111
Types of Triggers....2111
Create a Polled Trigger....2112
Examples of Scheduling Polled Triggers....2113
Create an Unsolicited Trigger....2114
Examples of Scheduling Unsolicited Triggers....2117
Calculation Formulas....2118
About Calculation Formulas....2118
General Guidelines for Defining a Calculation Formula....2119
Create a Calculation Formula Using a VBScript Code....2121
Create a Calculation Formula Using the Pre-built Functions....2123
Create a User-Defined Function....2125
Built-in Functions....2127
Types of Functions Supported....2135
User-defined Functions....2136
Date/Time Shortcuts....2138
Converting a Collected Value....2138
Calculations Inside Formulas....2138
Conditional Calculation....2139
Combining Tag Values and Assigning a Trigger....2139
Using CreateObject in a Formula....2139
Using a File....2140
Converting a Number to a String....2140

Detecting Recovery Mode Inside a Formula....2141
Looping Through Data Using the SDK....2141
Using an ADO Query....2143
Windows Performance Statistics Physical Memory Usage....2144
Windows Performance Statistics Virtual Memory Usage....2144
Determining Collector Downtime....2144
Analyzing the Collected Data....2145
Simulating Demand Polling....2146
Native Alarms and Events Functions....2147
About Native Alarms and Events Functions....2147
Retrieving and Setting Alarm Properties Manually....2147
Insert Calculation Functions Manually....2148
Data Input....2152
Calculation and Server-to-Server Collectors....2152
Recovery....2152
Manual Recalculation....2153
Troubleshoot Calculation Collector....2164
Troubleshooting Calculation collector....2164
Unsupported Data Types for Calculation Tags....2164
Unsupported Calculations in Calculation collector....2164
Writing Messages to the Collector Log File for Debugging Purposes....2165
Importing Calculations with Line Breaks into Historian....2165
Recovery Mode....2166
**Chapter 16. The CygNet Collector....2167**
Overview....2167
Configuration....2168
Using Configuration Hub....2168
Using Historian Administrator....2172
Specifying Tags for Data Collection....2174

Using Configuration Hub....2174
Using Historian Administrator....2176
Disabling Bad Offline Values....2179
Disabling Deleting Values....2179
Troubleshooting....2179
**Chapter 17. The File Collector....2181**
Overivew....2181
Configuration....2184
Using Configuration Hub....2184
Using Historian Administrator....2186
Importing Files....2188
CSV File Format....2188
XML File Format....2195
Troubleshooting....2205
**Chapter 18. The HAB Collector....2208**
Overview....2208
High Availability....2210
Configuration....2213
The HAB Collector....2213
Configure the Tags....2219
Configure the Alarms....2230
Start the Collector....2242
Approve Tag Changes....2243
Delete the Collector....2245
FAQs....2246
**Chapter 19. iFIX Collector....2251**
Overview....2251
About Adding an iFIX Collector Instance....2251
Specify the Tags for Data Collection....2255

Editing FixTag.dat File ..... 2257
Example: Restarting the iFIX Collector Using a Heartbeat..... 2259
Using an STK with the iFIX collector..... 2261
Setting Up..... 2263
Upgrading the iFIX Collectors..... 2263
The Configuration Section for iFIX collectors..... 2264
Collector-Specific Configuration (iFIX)..... 2264
Configuration of iFIX Data Collector-Specific Fields..... 2265
Starting an iFIX Collector Instance..... 2266
Troubleshooting..... 2268
**Chapter 20. Migrating iFix Data..... 2270**
Migrating iFix Data to Historian..... 2270
About Migrating iFix to Historian..... 2270
Historian Migration Utilities..... 2271
Adding the Historian Toolbar..... 2276
Migration Checklist..... 2276
Migrating Classic Historical Data..... 2277
About Migrating Classic Historical Data..... 2277
Migrating Classic Historian Data to Your Historian Database..... 2278
Configuring Classic Migration Options..... 2280
Migrating Advanced Historian Data..... 2284
Migrating Advance Historian Data..... 2284
Migrating iFIX Alarms and Events Collector..... 2291
Migrating iFIX Alarms and Events collector..... 2291
iFIX Alarms and Events collector Migration Options Configuration..... 2293
Alarm Source Options ..... 2293
Alarm Destination Options..... 2295
Troubleshoot iFIX Alarms and Events collector..... 2297
**Chapter 21. The MQTT Collector..... 2298**

Overview....2298
Configuration....2300
The MQTT Collector....2300
Using Historian Administrator....2309
**Chapter 22. The MQTT Sparkplug B Collector....2312**
Overview....2312
Adding an MQTT Sparkplug B Collector Instance....2317
Add an MQTT Sparkplug B Collector Instance using Configuration Hub....2317
Add an MQTT Sparkplug B Collector Instance using RemoteCollector Configurator....2325
Configuring an MQTT Sparkplug B Collector Instance....2331
Configure an MQTT Sparkplug B Collector Instance using Configuration Hub....2331
Configure an MQTT Sparkplug B Collector Instance using Historian Administrator....2336
**Chapter 23. The ODBC Collector....2341**
Overview....2341
Configuration....2343
Using Configuration Hub....2343
Using Historian Administrator....2347
Mapping the Data Format....2349
Data Recovery....2353
Reconnecting Automatically....2354
Troubleshooting....2355
**Chapter 24. The OPC Classic DA Collector....2356**
Overview....2356
Configuration....2357
Using Configuration Hub....2357
Using Historian Administrator....2363
Configuring Drivers and Deadbands....2366
Specifying Tags for Data Collection....2367
Using Configuration Hub....2367

Using Historian Administrator ... 2369
Creating OPC Groups ... 2371
Troubleshooting ... 2371
**Chapter 25. The OPC Classic HDA Collector ... 2374**
Overview ... 2374
Configuration ... 2376
Using Configuration Hub ... 2376
Using Historian Administrator ... 2380
Specifying Tags for Data Collection ... 2382
Using Configuration Hub ... 2382
Using Historian Administrator ... 2384
Data Recovery ... 2387
Reconnecting Automatically ... 2388
Troubleshooting ... 2388
**Chapter 26. The OPC Classic Alarms and Events Collector ... 2390**
About the OPC Classic Alarms and Events Collector ... 2390
About Event Types, Categories, and Conditions ... 2390
About Event Attributes ... 2391
Workflow for Using the OPC Alarms and Events Collector ... 2391
Configure the OPC Alarms and Events Collector ... 2392
Filter Alarms and Events Data ... 2393
**Chapter 27. OPC Classic HDA Server ... 2396**
Overview ... 2396
About the Historian OPC Classic HDA Server ... 2397
Setting Up ... 2398
Set Up the Historian OPC Classic HDA Server ... 2398
Enable Tag-Level Security ... 2398
Turn On Debug Mode for Trace Log Files ... 2399
Browse Large Number of Collectors and Tags ... 2399

Reference ... 2400
Supported Attributes ... 2400
Supported Data Types ... 2400
Supported Quality Values ... 2401
Supported Filter Attributes ... 2402
Example Trace Log File ... 2403
OPC Classic HDA Aggregates ... 2403
Average Aggregate ... 2405
Maximum Aggregate ... 2406
Minimum Aggregate ... 2407
Before Aggregate ... 2407
After Aggregate ... 2408
Nearest Aggregate ... 2408
Proficy Historian Interpolative Aggregate ... 2409
**Chapter 28. OPC UA HDA Server ... 2410**
Overview ... 2410
About the Historian OPC UA HDA Server ... 2411
Configuration ... 2412
Install the OPC UA HDA Server ... 2412
The OPC UA HDA Server Workflow ... 2416
Configure the OPC UA HDA Server Settings ... 2416
Connect the OPC UA HDA Server and the OPC UA HDA Client ... 2420
Authenticate a User to Connect to the OPC UA HDA Server ... 2421
Supported Attributes ... 2422
Supported Data Types ... 2423
Supported Quality Values ... 2424
Troubleshooting ... 2424
**Chapter 29. The OPC UA DA Collector ... 2426**
Overview ... 2426

Configuration....2428
Using Configuration Hub....2428
Using Historian Administrator....2432
Trusting a Client Certificate....2434
Connecting with Server....2434
Working with the Collector....2438
Using Configuration Hub....2438
Using Historian Administrator....2440
About OPC UA DA Collector Groups....2442
Troubleshooting....2442
**Chapter 30. OSI PI Collector....2443**
Overview....2443
Before You Begin....2444
OSI PI Collector Configuration....2446
Configuring the OSI PI Collector....2446
OSI PI Collector-specific Field Descriptions....2448
Tag Attributes Available in Browse....2450
Configuring Recovery Mode....2450
OSI PI Collector and Distributor Supported Data Types....2451
OSI PI Collector - Notes....2451
Starting and Stopping the OSI PI Collector....2452
Configuring Auto-synchronization of Digital States....2453
Renaming Digital States....2453
Deleting Digital States....2454
OSI PI Collector Troubleshooting....2455
**Chapter 31. OSI PI Distributor....2457**
OSI PI Distributor....2457
Overview....2457
Getting Started....2458

Configuring Multiple OSI PI Distributors to use Registry Keys....2459
OSI PI Distributor Configuration....2459
Configuring an OSI PI Distributor....2459
Tag Attributes Available in Browse....2461
OSI PI Collector and Distributor Supported Data Types....2461
Starting and Stopping the OSI PI Distributor Service....2462
**Chapter 32. The Python Collector....2463**
Overview....2463
Supported Data Types....2464
Install the Python Collector....2464
Adding a Python Collector Instance....2465
Add a Python Collector Instance using Configuration Hub....2465
Add a Python Collector Instance using RemoteCollectorConfigurator....2467
Configuring the Python Collector....2470
Configure the Python Collector using Configuration Hub....2470
Configure the Python Collector using Historian Administrator....2471
Configure Python Library Path using Configuration Hub....2472
Using the Python Collector....2473
Write Data to an Arbitrary Tag....2473
Create Triggers....2475
About Calculation Triggers....2475
Create a Polled Trigger....2476
Create an Unsolicited Trigger....2477
About Calculations....2478
About Calculations using Python Collector....2478
Create Python Script using Built-in Functions....2479
Create Python Script by Importing Third-party or Custom Python Modules or Functions....2481
Available Functions....2483
Examples of using Calculation Functions....2491

Examples: Using the Built-in Functions.... 2491
Examples: Custom or Third-party Python Modules.... 2491
Example: Storing Current Values of Arrays.... 2495
Example: Storing Dictionary Data as Multifield Data.... 2496
Example: Storing Python Integer List in Historian.... 2497
Example: Storing Python String List in Historian.... 2498
Example: Use Historian Data as Input to a Python Script.... 2498
**Chapter 33. Server-to-Server Collector....2499**
Overview....2499
Overview....2499
About Recovery Mode....2503
About Collection of Raw Samples....2504
Using the Collector....2504
Workflow for Using the Server-to-Server Collector....2504
Configure the Server-to-Server Collector Instance....2506
Tag Properties that are Copied....2508
Tag Properties that are Copied....2508
Examples of Data Collection....2509
Raw Samples Collection Example....2509
Advanced Collection Example....2510
Creating Calculation Formulas....2512
About Calculation Formulas....2512
General Guidelines for Defining a Calculation Formula....2513
Create a Calculation Formula Using a VBScript Code....2515
Built-in Functions....2517
User-defined Functions....2525
Create a User-Defined Function....2527
Date/Time Shortcuts....2529
Create a Calculation Formula Using the Pre-built Functions....2529

Types of Functions Supported ..... 2531
Data Input..... 2532
Calculation and Server-to-Server Collectors..... 2532
Recovery..... 2533
Manual Recalculation..... 2533
Examples of Calculation Formulas..... 2545
Converting a Collected Value..... 2545
Calculations Inside Formulas..... 2545
Conditional Calculation..... 2545
Combining Tag Values and Assigning a Trigger..... 2545
Using CreateObject in a Formula..... 2545
Using a File..... 2546
Converting a Number to a String..... 2547
Detecting Recovery Mode Inside a Formula..... 2547
Looping Through Data Using the SDK..... 2548
Using an ADO Query..... 2549
Windows Performance Statistics Physical Memory Usage..... 2550
Windows Performance Statistics Virtual Memory Usage..... 2551
Determining Collector Downtime..... 2551
Analyzing the Collected Data..... 2552
Simulating Demand Polling..... 2552
**Chapter 34. The Server-to-Server Distributor ..... 2554**
Overview..... 2554
Workflow for Using the Server-to-Server Distributor..... 2555
Configure the Server-to-Server Distributor..... 2555
**Chapter 35. The Simulation Collector..... 2558**
Overview..... 2558
Configuration..... 2558
Using Configuration Hub..... 2558

Using Historian Administrator ... 2559
Tags with Sequential Values ... 2560
**Chapter 36. Windows Performance Collector ... 2562**
Windows Performance Collector ... 2562
Overview ... 2562
Windows Performance Collector Feature Summary ... 2562
Windows Performance Collector Configuration ... 2563
Understanding Windows Performance Collector Tag Hierarchy ... 2563
The Configuration Section for Windows Performance Collector ... 2564
**Chapter 37. The Wonderware Collector ... 2566**
Overview ... 2566
Installation Prerequisites ... 2567
Wonderware Collector Features ... 2567
Hierarchical Tags Available in Browse ... 2568
Supported Data Types for Wonderware Collector ... 2569
Configuring Wonderware Collector ... 2570
Data Recovery ... 2571
Initiating Manual Recovery ... 2573
Reconnecting to the Wonderware Server ... 2574
Troubleshooting the Wonderware Collector ... 2575
**Chapter 38. OLE DB Provider ... 2576**
Overview ... 2576
Setting Up ... 2577
Install the OLE DB Provider ... 2577
Connect to a Historian Server ... 2580
Working with Clients ... 2580
Power BI Desktop ... 2580
VisiconX ... 2585
Oracle ... 2587

Crystal Reports ... 2587
Microsoft Excel ... 2592
Visual Basic and ADO ... 2596
Proficy Real-Time Information Portal ... 2600
Linked Servers ... 2600
Working with Queries ... 2606
Access the Historian Interactive SQL Application ... 2607
Run a Query ... 2608
Connect to a Server ... 2609
Save a Query ... 2610
Export Results ... 2611
Optimize the Query Performance ... 2611
Supported SQL Syntax ... 2612
SELECT Statements ... 2613
SET Statements ... 2629
Parameterized SQL Queries ... 2636
Optimize the Query Performance ... 2637
Troubleshooting and Frequently Asked Questions ... 2638
Troubleshooting ... 2638
Frequently Asked Questions ... 2642
Historian Database Tables ... 2647
The Historian Database Tables ... 2647
ihTags Table ... 2651
ihArchives Table ... 2659
ihCollectors Table ... 2661
ihMessages Table ... 2667
ihRawData Table ... 2670
ihHabAlarms Table ... 2680
ihComments Table ... 2682

ihTrend Table....2693
ihQuerySettings Table....2709
ihCalculationDependencies Table....2715
ihAlarms Table....2716
ihEnumeratedSets Table....2720
ihEnumeratedStates Table....2721
ihUserDefinedTypes Table....2722
ihFields Table....2723
**Chapter 39. The Excel Add-In for Historian....2725**
Overview....2725
Setting Up....2726
Install Using the Installer....2726
Install at a Command Prompt....2727
Activate Excel Add-In....2728
Querying Data....2730
Query Current Values....2730
Query Filtered Data....2732
Querying Calculated Data....2734
Querying Alarms and Events Data....2736
Modify a Query....2737
Query Modifiers....2738
Access Archive Statistics....2741
Access Collector Statistics....2742
Managing Tags....2743
Search for a Tag (Basic)....2743
Search for a Tag (Advanced)....2744
Export Tags....2746
Add/Modify Tags....2747
Import Tags....2748

Rename Tags.... 2748
Working with Array Tags.... 2749
Importing and Exporting Data.... 2750
Import Tag Data.... 2750
Importing Alarms and Events Data.... 2750
Export Tag Data .... 2751
Exporting Alarns and Events Data.... 2752
Working with Messages.... 2754
Search for Messages.... 2754
Import Messages.... 2755
Export Messages.... 2755
Managing Enumerated Sets.... 2756
Export Enumerated Sets.... 2758
Import Enumerated Sets.... 2759
Rename Enumerated Sets.... 2759
Managing User-Defined Types.... 2760
Export User-Defined Types .... 2761
Import User-Defined Types.... 2762
Reference.... 2762
Excel Add-In Options.... 2762
Reports.... 2764
Relative Time Entries.... 2777
Filter Parameters for Data Queries.... 2778
Batch IDs.... 2780
Sampling Types.... 2781
Calculation Algorithm Types.... 2782
Tag Criteria List.... 2784
Troubleshooting.... 2785
**Chapter 40. The Excel Addin for Operations Hub.... 2788**

Overview....2788
About Operations Hub....2789
Setting Up....2789
Software Requirements....2789
Install Excel Add-In for Operations Hub....2790
Copy or Export the Issuer Certificate on Server....2791
Install/Import the Issuer Certificate....2792
Connect to Operations Hub....2793
Querying Data....2793
Query Operations Hub Model....2794
Troubleshooting....2797
**Chapter 41. Trend Client....2799**
Overview....2799
Access Trend Client....2799
Access Help....2799
Managing Tags....2799
Add Tags for Analysis....2801
Creating a Display....2802
Add a Trend Chart....2802
Add a Current Value Table....2803
Add a Value Card....2803
Add a Text Box....2804
Access a Display....2804
Provide a Title to a Display....2805
Filter Data....2805
Change the Sampling Mode....2806
Change the Time Zone....2806
Export Data....2807
Set the Refresh Interval....2808

Working with a Trend Chart.... 2808
Add a Trend Chart.... 2808
Switch the Y-Axis.... 2810
Change the Format.... 2810
Change the Duration.... 2812
Access the Statistics.... 2813
Change the Sampling Mode.... 2813
Change the Scale of a Trend Chart.... 2814
Managing Favorites.... 2815
Access a Favorite.... 2815
Export a Favorite.... 2815
Import a Favorite.... 2816
Delete a Favorite.... 2816
**Chapter 42. Historian Web Admin Console.... 2817**
Overview.... 2817
Overview.... 2817
Difference Between the Web Admin Console and Historian Administrator.... 2817
Actions You Can Perform Using the Web Admin Console.... 2818
Access the Web Admin Console.... 2818
Understanding the Interface.... 2819
Understand the Historian Interface.... 2819
Client Panel.... 2820
Collector Panel.... 2822
Data Node Panel.... 2826
Configuration Panel.... 2830
Configure General Collector Options.... 2848
Configure General Collector Options.... 2848
Maintain, Operate, and Monitor Historian.... 2854
Plan For Data Recovery.... 2854

Develop a Maintenance Plan ..... 2855
Monitor Historian Performance ..... 2855
Evaluate Data Compression Performance ..... 2865
Historian Data Types ..... 2867
Managing Tags ..... 2870
Access a Tag ..... 2870
Add a Tag to a Data Source ..... 2870
Add a Tag Manually ..... 2871
Add a Source Address to a Tag ..... 2872
Adding OPC Tags from a Collector ..... 2872
Adding Simulation Tags from a Collector ..... 2872
Filter and Search Tags ..... 2873
Access the Trend Chart of Tag Values ..... 2875
Displaying Raw Data Samples ..... 2875
Dynamic Collector Updates ..... 2876
Reload Tag Parameters ..... 2877
Rename Tags ..... 2879
Stale Tag Management ..... 2881
Delete Tags ..... 2881
Managing Data Stores ..... 2882
About Data Stores ..... 2882
Moving Tags Between Data Stores ..... 2883
Adding a Data Store ..... 2884
Deleting a Data Store ..... 2884
Editing a Data Store ..... 2885
Managing Data Archives ..... 2885
Configure Data Archives ..... 2885
**Chapter 43. Extract, Transform, and Load (ETL) ..... 2901**
Overview ..... 2901

Workflow for Transferring Data from eDNA ..... 2902
Workflow for an ODBC Data Source..... 2903
Workflow for Proficy Historian..... 2905
Workflow for PI Historian..... 2906
Installation..... 2908
Upgrade..... 2910
Extracting Data from an eDNA Server..... 2910
Specify Tags Using the Utility..... 2911
Specify Tags Manually..... 2912
Specify Tags Using a Template..... 2913
Specify Tags Using a Blank Spreadsheet..... 2914
Tag Properties..... 2916
Configure..... 2916
Extract Historical Data..... 2922
Start Data Extraction..... 2923
Extracting Data from an ODBC Data Source..... 2924
Specify Tags and Tables Manually..... 2924
Specify Tags and Tables Using a Template..... 2926
Specify Tags and Tables Using a Blank Spreadsheet..... 2928
..... 2931
Table Properties..... 2932
Configure..... 2933
Start Data Extraction..... 2937
Extracting Data from Proficy Historian..... 2938
Specify Tags Manually..... 2938
Specify Tags Using a Template..... 2939
Specify Tags Using a Blank Spreadsheet..... 2940
Tag Properties..... 2942
Configure..... 2942

Extract Historical Data..... 2946
Start Data Extraction..... 2947
Extracting Data from PI Historian..... 2948
Specify Tags Manually..... 2948
Specify Tags Using a Template..... 2949
Specify Tags Using a Blank Spreadsheet..... 2950
Tag Properties..... 2951
Configure..... 2952
Extract Historical Data..... 2956
Start Data Extraction..... 2957
Transferring Data Using BITS..... 2957
Configure BITS..... 2957
Verify Settings..... 2958
Transfer Data..... 2958
Transferring Data Using FTP..... 2959
Configure FTP..... 2959
Transfer Data..... 2960
Loading Data..... 2961
Configure..... 2961
Load Data..... 2965
Reference..... 2965
Example of a Regeneration File..... 2966
Example of a State File..... 2968
Troubleshooting..... 2969

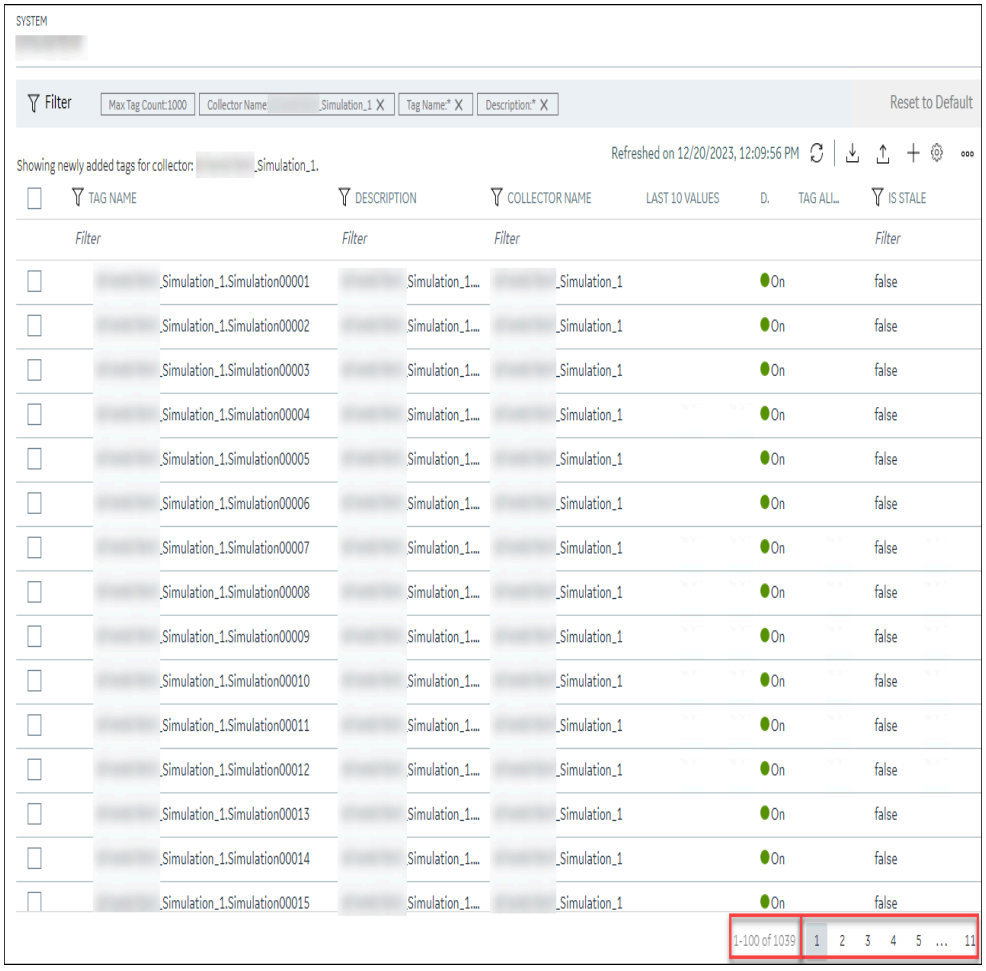
# Chapter 1. Release Notes

## What's New in Historian 2024

| Description | Tracking ID |
|---|---|
| **Configuration Hub Enhancements**<br><br>Configuration Hub provides a better user experience using an IDE model to manage systems, bringing in consistency across the portfolio to manage the configuration and administration of multiple products.<br><br>The following Historian specific enhancements are made in Configuration Hub: | |
| **Ability to register and manage Configuration Hub plugin for Historian:** You can manage the Configuration Hub plugin for Historian using Configuration Hub. You can register a plugin, unregister a plugin, modify a plugin, and view Historian node and plugin-specific details using the Node Manager. For more information, refer to Historian plugin management. | F69440 |
| • **Ability to browse tag details when primary node is inactive**: In a horizontally scaled system, when a primary server's central configuration manager is inactive, you can still browse tags by adding distributed or mirror nodes as a backup for the primary server. When you add a node as the backup, the Data Archiver of that node functions as the Configuration Manager, and sends the relevant tag information to the client manager.<br><br>To facilitate this enhancement, a new **Support for Browse Tags, if primary server down** property has been added in the **Systems > GENERAL** section of the **DETAILS** section.<br><br>For more information, refer to browse tags using distributed or mirror node servers when primary server is inactive. | F69877 |
| • **Create custom templates for common configurations**: You can create configuration templates for both data stores and collectors separately.<br><br>To facilitate this enhancement, a new **Configuration Templates** section has been added in the Configuration Hub plugin for Historian. Within the **Configuration Templates**, you are provided with two different tabs, one tab to configure collector-specific template, and the other tab to configure data store-specific template. | |

| Description | Tracking ID |
|---|---|
| **Note:**<br>You can apply a data store configuration template to a user-created data store, provided they are not set as the default data store.<br><br>For more information, refer to Configuration templates. | |
| **Ability to reset collector overruns**: You can reset overrun count of a collector instance to zero without restarting the collector. To facilitate this enhancement, a new collector action **Reset Overruns** has been added. For more information, refer to Reset Overruns. | F70190 |
| **Ability to be notified when a restart of collector instance is required**: When you modify certain configurations of a collector instance, you will be prompted to restart the collector instance. To facilitate this enhancement, a new confirmation dialog has been added. Whenever you modify certain configuration and save, you will be prompted with an option to restart the collector instance. If you decide not to restart, a notification will be displayed in the **DETAILS** section, stating that the selected collector instance needs a restart.<br><br>**Note:**<br>The modification(s) will take effect only after you restart the collector instance.<br><br>For more information, refer to Modify collector instance. | |
| **Ability to validate the destination Historian connection while adding a collector instance**: You can test a destination Historian server's connection status while adding a collector instance. This will enable you to connect to a valid, and an active destination Historian server. To facilitate this enhancement, a new **Test Connection** button has been added in the **Collector Initiation** section, and also in the **Update Collector Credentials: <Collector name>** window for all the collectors. | |
| **Ability to validate the source Historian connection while adding a Server-to-Server Collector** or **Server-to-Server Distributor** collector instance: You can test a source Historian server's connection status while adding a collector instance. This will enable you to connect from a valid, and an active source Historian server. To | |

| Description | Tracking ID |
|---|---|
| facilitate this enhancement, a new **Test Connection** button has been added in the **Source Configuration** section. | |
| **Ability to configure multiple tags at once**: You can select multiple tags and configure their common configurations at once. To facilitate this enhancement, you are now enabled to select multiple tags using their corresponding check boxes, or you can select all the available tags using the check box available in the grid's header.<br><br>For more information, refer to Configure multiple tags. | |
| **Ability to add and view the actual unit of measure of a tag's value**: You can add the actual unit of measure of a tag's value. This will help you to view and add more context to a tag's value that you pull. To facilitate this enhancement, a new **EGU Description** property has been added in the **SCALING** section of tag's **DETAILS** section.<br><br>The engineering unit that you provide will be displayed under the **Engineering Units** column in the table view, and as a tool tip in the trend view, when you view the view the last 10 values of the tag, or when you generate a query/write report in the **Data** page.<br><br>For more information, refer to the **SCALING** section in Access Tags. | |
| **Ability to view Tag Creation Audit Trail**: You can view the Tag Creation Audit Trail details for a tag. To facilitate this enhancement, new **Created By**, and **Created Time** properties have been added in the **Tags > DETAILS > GENERAL** section. | |
| **Ability to add a stale period check for data store level**: You can view the number of days or the frequency for checking data store validity. To facilitate this enhancement, a new **Stale Period (Days)** property has been added in the **MAINTENANCE** section in the Data Stores **DETAILS** section.<br><br>For more information, refer to the **MAINTENANCE** section in Access a Data Store. | |
| **Ability to Import/Export**: You can import/export tags and their details, user defined-defined types (UDT), and Enumerated sets in a CSV format. This will enable you to export the data, modify the data, and then import them back using Configuration Hub. To facilitate this enhancement, the ⬇ and ⬆ buttons have been added in the corresponding sections.<br><br>For more information, refer to the following sections: | |

| Description | Tracking ID |
|---|---|
| • Import, and Export tag details.<br>• Import, and Export UDT.<br>• Import, and Export Data Attribute Enumerated sets.<br>• Import, and Export Enumerated Sets. | |
| **Ability to view the grid count and navigate through grids**: You can view a grid's total and selected count, and when there are more than 100 rows in a grid, you can use the page numbers to navigate through the grid details.<br><br>To facilitate this enhancement, grid count (total and selected) and page numbers have been added below a grid.<br><br>  <br> | |
| **Implement Data Store-level security**: Similar to tag-level security, you can also define data store-level security to protect sensitive tags. | F70957 |

| Description | Tracking ID |
|---|---|
| **Note:**<br>However, the security settings defined at the tag level, if any, take the precedence over those at the data store-level.<br><br>To facilitate this enhancement, the following properties have been added in the **SECURITY** section, in the data store **DETAILS** section. For more information, refer to Implementing data store-level security. | |
| **Ability to configure multiple data archive paths**: Multiple archive paths feature extends the ability to configure a default archive path for each data store.<br><br>This feature enables you to automatically store your archives in different locations based on the age of the archive. It can be useful when splitting the data for a data store across multiple hard drives, primarily for performance and cost optimization. Additionally, it could help optimize disk space constraints.<br><br>To facilitate this enhancement, a **Multiple Archive Paths** property has been added in the **ARCHIVE CREATION** section in the data store's **DETAILS** section. For more information, refer to Multiple archive paths. | F70956 |
| **Ability to configure security groups to use Proficy Authentication**: You can configure **Security Group** at global (system) level to use Proficy Authentication. To facilitate this enhancement, a new **Use Proficy Authentication** option has been added to **Security Group** in the System's **GLOBAL SECURITY** section. If you select the **Use Proficy Authentication** option, the system will use the groups specific to Proficy Authentication (UAA) for authorization. Only the users or groups that belong to the Proficy Authentication will have specific permissions and access rights. For more information, refer to the **GLOBAL SECURITY** section in Access a System.<br><br>**Note:**<br>To use **Proficy Authentication** security group, you must perform the configurations listed in the Configurations to use Proficy Authentication Security Groups section. | F69870 |
| **Ability to generate configuration reports**: You can generate report to get a concise summary of tag-specific data of a specific collector. You can select a report from the available predefined templates and generate a categorized report. | F66520 |

| Description | Tracking ID |
|---|---|
| **Note:**<br>The reports are applicable only to the Historian destination collectors.<br><br>To facilitate this enhancement, a new **Reports** section has been added in the Configuration Hub plugin for Historian. Within the **Reports** section, you are provided with a list of available predefined templates. You can select a template and generate its corresponding report and export it. For more information, refer to About reports. | |
| **Ability to save data query and access it**: You can save queries that you use to read tags data. You can use this option to save queries that you may frequently use to read specific tags data. This will help you in the following:<br><br>• Quickly gather insights of specific tags.<br>• Create and save a predefined query only once and use it as needed.<br>• You can use the **Shared** option to make the query visible to the other users in the same network.<br><br>**Note:**<br>Once you share a query, all the other users will also be able to edit, and delete it. Be mindful about making a query as a shared query.<br><br>To facilitate this enhancement, a new **Save Query** button has been added in the **New Query** tab. The button will appear after generating a **Read Data** report. Additionally, a new **Saved Queries** tab has been added, which contains the list of all saved queries. For more information, refer to Save a Query.<br><br>You can also update a saved query. For more information, refer to Update a saved query. | |
| **Other Enhancements** | |
| **Term Licensing**: In parity with other products within the Proficy portfolio, Historian licensing also supports a term license model. This will give you the flexibility to subscribe to Historian for a subscription fee based on your requirement. | F69872 |
| **Enhanced data type support for MQTT Collector**: The support of Array tags has been added for the MQTT Collector. | |

| Description | Tracking ID |
|---|---|
| **ETL Tools Enhancement**: Performance improvements are made for the eDNA extractor. | |
| **New MQTT Sparkplug B Collector**: You can connect as a Primary Host application and collect data based on the MQTT Sparkplug B specification (Sparkplug 3.0.0). Specifically, you can subscribe to a topic that was published using the Sparkplug B payload and collect time-series data in a structured and understandable format. To facilitate this enhancement, a new **MQTT Sparkplug B Collector** has been added. You can install this collector and configure it to collect data based on the MQTT Sparkplug B specification (Sparkplug 3.0.0).<br><br>For more information, refer to MQTT Sparkplug B Collector. | F73170 |
| **Ability to extensively use the Python Collector**: You can now extensively use Python collector. This collector is available for use in both the Configuration Hub and the Historian Administrator. However, for a seamless user experience, it is recommended to use the collector in the Configuration Hub. To facilitate this enhancement, the following enhancements have been added in the Configuration Hub:<br><br>• Dedicated Python script editor.<br>• Extensive list of built-in functions that you can use in your script. For more information, refer to Available functions.<br>• Extensive list of supported data types. For more information, refer to Supported data types.<br>• Capability to import third-party, custom Python modules and use them in the script. For more information, refer to Import third-party or custom Python Modules.<br>• Capability to configure path for the newly installed Python modules using Configuration Hub. For more information, refer to Configure Python Library Path. | F63933 |
| **Support for Mutual TLS (MTLS) Protocol for Enhanced Security**: You can use MTLS protocol for Historian core services and also collectors. To facilitate this enhancement, the following enhancements have been added: | F70082 |

| Description | Tracking ID |
|---|---|
| • Generating the required root certificate for Historian and core services can be done using the installer. You do not have to manually generate and trust the root certificates. For more information, refer to Certificate-based security for MTLS.<br>• An option to Enable MTLS Security for collector specific instances has been added in Configuration Hub. For more information, refer to Enable MTLS Security for Collectors.<br>• An option to encrypt the MTLS messages sent to Historian has been added. For more information, refer to Enable MTLS Security for Collectors. | |
| **Ability to create data attribute enumerated set**: You can create and define a custom text for the quality status and view it in the **Attributes** column instead of the value when you retrieve the tag data. For example, if your tag is receiving bit positions indicating the quality status (that is, 1 for Old data, 2 for Bad data, and 30 for Good data), you can map the bit position to a custom text and assign it to a set. So, when you view the actual values of the tag, the **Attributes** column will display the mapped custom text instead of the integer (the bit positions). To facilitate this enhancement, in Configuration Hub, a new **DATA ATRRIBUTE ENUMERATED SET** option has been added in the **Manage Enumerated Sets** section. To enable data attribute enumerated set, you must set the **DATA ATRRIBUTE ENUMERATED SET** option to **True**. For more information, refer to about data attribute enumerated set.<br><br>**Note:**<br>Once you created a data attribute enumerated set in Configuration Hub, you can use it in Historian Administrator. However, it is not recommended to modify the created Data Attribute Enumerated set in the Historian Administrator. If you modify it using the Historian Administrator, the **DATA ATTRIBUTE ENUMERATED SET** field will be changed to **False** in Configuration Hub. | F73356 |
| **Ability to visualize and interact with Historian specific REST APIs using Swagger UI**: You can now access Historian REST APIs using Swagger UI. This tool enables you to visualize and interact with the API resources without having any of the implementation logic in place, establishing a fully interactive documentation experience using Swagger. For more information, refer to Swagger Documentation. | F69878 |
| **Enhancements in Historian REST APIs for Tags** | F70197 |

| Description | Tracking ID |
|---|---|
| **Taglist API**: To avoid pagination related issue, you can set the pagesize parameter to 0. This will provide all tags at once without any pagination.<br><br>**Engineering Units**: Tag Data response now includes Engineering Units. For example,<br><br>{<br>"ErrorCode": 0,<br>"ErrorMessage": null,<br>"Data": [<br>{<br>"TagName": "Test2020_Simulation.Simulation00001",<br>"ErrorCode": 0,<br>"DataType": "Float",<br>"EngineeringUnits": null,<br>"HiEngineeringUnits": 200000,<br>"LoEngineeringUnits": 0,<br>"Samples": [<br>{<br>"TimeStamp": "2023-12-04T13:04:51.000Z",<br>"Value": "93319.5",<br>"Quality": 3,<br>"DataAttributes": []<br>}, | |

# Resolved Issues

| Description | Tracking ID |
|---|---|
| Previously, when you viewed the Last 10 Values of the tags of Sever-to-Server collector, the trend was not visible, however, if you right-click on the tags, it was visible. This issue has been resolved in Historian 2024. | 01092678 (DE205966) |
| Previously, there were Apache Tomcat vulnerabilities observed in Historian server. This issue has been resolved in Historian 2024. | 01088103, 01086760, 01115035, 01114659, 01113071, |

| Description | Tracking ID |
|---|---|
| | 01107496, 01130189, 01118147 (DE204880) |
| Previously, using Web Administrator, when you viewed the Last 10 Values of a tag, the trend data was not displayed. This issue has been resolved in Historian 2024. | 01086847 (DE205687) |
| Previously, on Configuration Hub, when you searched tags by their collection interval, the tags were not returned. This issue has been resolved in Historian 2024. | 01083356 (DE205085) |
| Previously, there were OpenSSL 1.1.1k vulnerabilities observed in Historian. This issue has been resolved in Historian 2024. | 01070308, 01050129 (DE204882) |
| Previously, when you attempted to extract 71 and more good quality codes using the Proficy Historian eDNA ETL tool, and then used them in configuration, all the codes were not being considered. This issue has been resolved in Historian 2024. | 01093884 (DE212931) |
| Previously, when you attempted to start the Historian ETL PI Extract Settings, the settings did not start as the IHELPIExtract services did not start or respond. This issue has been resolved in Historian 2024. | 01052974 (DE199961) |
| Previously, files with the wrong time format were not moved to the Error folder. Instead, they kept executing forever. This issue has been resolved in Historian 2024. | 01093946 (DE212928) |
| Previously, there was an issue in creating a LAX file for unacknowledged and/or not closed alarm. This issues has been resolved in Historian 2024. | 01098478 (DE212929) |
| Previously, when you attempted to load the LAX file containing custom attributes, it did not work. This issue has been resolved in Historian 2024. | 01082173 (DE212930) |
| Previously, using the Proficy Historian Excel Add-in, when you attempted to export tags from a Historian with more than 82,000 tags, it appended an underscore at the beginning of the tag name. This issue has been resolved in Historian 2024. | 01093704, 01095104 (DE204980) |
| Previously, when you attempted to browse, the Show ALL Children was disabled when you had collector with a large hierarchy. This | 01106682 (DE208000) |

| Description | Tracking ID |
|---|---|
| issue has been resolved in Historian 2024 by providing recursive browsing support for the OPC UA Collector. | |
| Previously, STA model was not supported for OPC Collector. This issue has been resolved in Historian 2024 by adding support of the STA model. | 01097084 (DE208654) |
| Previously, when you attempted to query string tags in trend mode, there was an increase in the DataArchiver memory until it stopped or became unresponsive. This issue has been resolved in Historian 2024 through memory optimization. | 01112199 (DE210018) |
| Previously, after Historian alarms and events backup, purge, and restore procedures, not all alarms were available. This issue has been resolved in Historian 2024. | 01109547 (DE213006) |
| Previously, when you used tag type Array of string and multifield, memory spike was observed. This issues has been resolved in Historian 2024. | DE204209 |
| Previously, due to memory leak issue, the Historian 2022 Server-Server Collector did not function. This issue has been resolved in Historian 2024 through memory optimization. | DE203744 |
| Previously, in operating system of specific languages, the iFIX collector stopped working when tags were browsed in Historian Administrator. This issue has been resolved in Historian 2024 by updating the latest APIs. | 01092706, 01093358 (DE206183) |
| Previously, when the Historian local security was enabled, and the LDAP user was added to 'ih_security_admins' group, the LDAP user was not able to query data from Historian and did not load Historian plugin on Configuration Hub. This issues has been resolved in Historian 2024. | 01081620 (DE208053) |
| Previously, in a redundant configuration, if you set fail over triggers, the OPC UA collector stopped working on both the collector machines. This issue has been resolved in Historian 2024. | 01090819 (DE208653) |

| Description | Tracking ID |
|---|---|
| Previously, in a redundant configuration, there were bad values observed in OPC UA Collector. This issue has been resolved in Historian 2024. | 01066966 (DE199964) |
| Previously, in a redundant configuration, there were bad values observed in OPC UA Collector and lot of logs were created. This issue has been resolved in Historian 2024. | 01080450 (DE201897) |
| Previously, the MQTT Collector had memory leak. This issue has been resolved in Historian 2024. | 01100809 (DE206909) |
| Previously, the OPC client ReadRaw call caused Server to fragment memory. This issue has been resolved in Historian 2024. | 01112205 (DE210166) |
| Previously, the collector HDA became unresponsive after adding new tags. This issue has been resolved in Historian 2024. | 01094822 (DE205288) |
| Previously, the OPC Classic HDA Collector restart took more time than expected. This issue has been resolved in Historian 2024. | 01090894 (DE205285) |
| Previously, the OPC Classic HDA Collector was failing with Geo SCADA, bot being on the same machine. This issue has been resolved in Historian 2024. | 01047010 (DE199907) |
| Previously, when you viewed the Last 10 Values of a tag using the OPC UA Collector in Historian Administrator, the data values were reported to be of bad quality, although, the values came from another good tag. This issue has been resolved in Historian 2024. | 01111077 (DE210392) |
| Previously, you were able to enter a Scada Buffer Duration value beyond the actual supported maximum value in the Scada Buffer Duration(Days) field. This issue has been resolved in Historian 2024 by adding a conditional check on the maximum value that is being entered. | 01100502 (DE208287) |
| Previously, when using the OPC UA HDA Collector, tags with bad quality in the Archive were incorrectly shown as having good quality. This issue has been resolved in Historian 2024. | 01074939, 01082827 (DE202695) |
| Previously, the Filtered Data window screen in Excel Add-in was very small. This issue has been resolved in Historian 2024 by increasing the dimension (Height, Width, Top, and Left). | 01071028, 01071216, 01084176, 01096784, 01098978 (DE173096, DE199256) |

| Description | Tracking ID |
|---|---|
| Previously, when you query Alarms and Events, the Perform_Comment field was blank, although it existed in the database. This issue has been resolved in Historian 2024. | 01095954, 01085812 (DE205472, DE205095) |
| Previously, restoring .zip archives from Historian 7.1 to a Historian 2022 server resulted in an empty archive. This issue has been resolved in Historian 2024. | 01048625 (DE199965) |
| Previously, the OPC UA collector stored the data quality as 'GOOD' when the actual quality was BAD. This issue has been resolved in Historian 2024. | 01019528 (DE181224) |
| Previously, there were some differences in data retrieval using the interpolated or calculated mode between Historian versions 3.1 and 9.1. This issue has been resolved in Historian 2024. | 01048854 (DE201650) |
| Previously, both tags, one of type 'Byte' and the other of type 'Quad Integer', were incorrectly returning same value for the datatype when queried through the User API for properties. This issue has been resolved in Historian 2024. | 01055391 (DE199962) |
| Previously, the configuration manager became unresponsive on primary node unexpectedly, preventing the users from adding tags. This issue has been resolved in Historian 2024. | 01072354 (DE212949) |
| Previously, although the OPC UA Data Access Collector was able to connect and browse some folders but could not see the items that are located under sub-folders. This issue has been resolved in Historian 2024. | 01114602 (DE210148) |
| Previously, the Data Archiver stopped working frequently and there was no information in the log. This issue has been resolved in Historian 2024. | 01107015 (DE212951) |
| Previously, when Historian server stopped working, restarting the services caused the Configuration manager to become unresponsive. This issue has been resolved in Historian 2024. | 01107511 (DE209451) |
| Previously, when using the Excel Add-in for Current Value and Raw Data, if the tag quality was 'BAD', then **BAD** was displayed instead of | 01058496 (DE199960) |

| Description | Tracking ID |
|---|---|
| the actual value in the **Value** column. This issue has been resolved in Historian 2024. | |
| Previously, when using Condition Based Collection, the Historian OPC Collector stopped working. Also, excessive logs were created. This issue has been resolved in Historian 2024. | 1043533 (DE196097) |
| Previously, when you ran an Interpolated Query in Excel Add-in by setting a specific time interval, the query returned incorrect value. This issue has been resolved in Historian 2024. | 1065803 (DE199958) |
| Previously, in Excel Add-in, the Calculated Data with Query Criteria as String returned incorrect values. This issue has been resolved in Historian 2024. | 01031714 (DE198804) |
| Previously, when the connection to the OPC UA Server was lost, the OPC UA Collector was displaying the tag quality as GOOD, even though the actual quality was BAD. This issue has been resolved in Historian 2024. | 1052808 (DE200995) |
| Previously, on Non-web Admin, when viewing the Default Collector Compression Timeout setting, the value was not visible or updated correctly if the Deadband type was set to Absolute. This issue has been resolved in Historian 2024. | 01083360, 1009963 (DE203189) |

# Known Issues and Limitations

**Table 1. Upgrade Related Known Issues**

| Description | Tracking ID |
|---|---|
| Upgrading only the Client Tools to 2024 from an older version upgrades the Historian Server, but does not include the MTLS files.<br><br>**Workaround**: When upgrading Client Tools to 2024, ensure that you also upgrade the Historian Server to 2024. | DE208482 |
| In a Historian server 2023, AlarmArchiver 2023, and Collectors 2023 setup, if you upgrade only the Alarm archive to 2024, and start the collectors, the collectors do not start. | DE211329 |

**Table 1. Upgrade Related Known Issues (continued)**

| Description | Tracking ID |
|---|---|
| **Note:**<br>This is also applicable to an upgrade of Historian 2022, AlarmArchiver 2022, and Collectors 2022 setup to 2024.<br><br>**Workaround**:<br>• Along with the AlarmArchiver, you must upgrade the Historian server to 2024.<br>-OR-<br>• If you do not use MTLS, disable it. | |
| If you update AlarmArchiver from 9.1 Alarms & Events to 2024 Alarms & Events, without updating Historian server to 2024, the Data Archiver will not start.<br><br>**Workaround**: Along with the AlarmArchiver, you must upgrade the Historian server to 2024. | DE206514 |
| In a 9.0 or 9.1 standard Historian, Alarm Archiver, and AEDB SIM (if any) setup, when you upgrade the Alarm Archiver to 2024, the Data Archiver will also get upgraded to 2024. | DE210847 |
| When you upgrade Web-based clients to Historian 2024 on a non-English operating system, you will see port conflicts that do not allow you to proceed.<br><br>**Workaround**: Use the silent installation method to install the Historian Web-based clients. | DE210880 |
| Port conflict occurs when you upgrade Web-based Clients from 2023 to 2024, on a non-English operating system. | DE212901 |

**Table 2. Path Related Issues After Install or Upgrade**

| Description | Tracking ID |
|---|---|
| Even if the Historian server is not installed on the C: drive, **RemoteCollectorConfigurator.exe** is installed in `C:\Program Files\GE Digital\NonWebCollectorInstantiationTool`. | DE212800 |
| The File collector cannot be processed after upgrading from version 2022 to version 2024. | DE210946 |

**Table 2. Path Related Issues After Install or Upgrade (continued)**

| Description | Tracking ID |
|---|---|
| If you installed Calculation collector or Windows Performance collector using either iFIX Proficy installer or CIMPLCITY Proficy installer, they fail to start after you upgrade them to 2024.<br><br>**Workaround**: After you upgrade the collectors, copy the collector files to the initial install directory. That is, `C:\Program Files (x86)\Proficy\GE Digital\`. | 516236 |

**Table 3. Security Related Known Issues**

| Description | Tracking ID |
|---|---|
| In a setup where you installed Historian collectors along with client tools using the Proficy installer, such as iFIX Proficy installer, if you uninstall the client tools alone, the created MTLS registry gets removed, leaving the MTLS folder intact. This prevents you from using MTLS security. | DE212965 |
| The POSTGRES and Proficy Authentication (UAA) passwords are admissible in the Historian Web Components log. | DE212644 |

**Table 4. Other Known Issues**

| Description | Tracking ID |
|---|---|
| If you import tags, along with their CreationTime and CreationUser, that were exported from Configuration hub, the Excel Add-in displays the "Proficy Historian Import Tags Error".<br><br>**Workaround**: You must manually remove the CreationTime and CreationUser columns in the exported file before importing it. | DE207159 |
| After you create a Data Attribute Enumerated Set in Configuration Hub, and then edit it using Historian Administrator, the **DATA ATTRIBUTE ENUMERATED SET** field is changed to **False** in Configuration Hub. | DE209931 |
| When you use the Python collector to convert a value to boolean type, the results are provided in string format.<br><br>**Workaround**: When you convert a value to boolean type, you must store the results as `Result = bool(Historian.CurrentValue('TP.Simulation00001'))` | |

**Table 4. Other Known Issues (continued)**

| Description | Tracking ID |
|---|---|
| Configuration Hub is not able to discover and add a system with Strict client and collector authentication enabled. | DE211901 |
| In Configuration Hub with LDAP and UAA setup, where the security groups are stored in the Domain Controller, the Tags and Data store Security Domain groups are not available for selection when assigning security group to a data store.<br><br>**Workaround**: You can create the required security groups locally in Windows. This will ensure that the group becomes available for selection in Configuration Hub. | DE211554 |
| The trend decimal values in Configuration Hub are different from those in Historian Administrator. | DE212903 |
| In Historian Interactive SQL, the sampling mode "rawbyfiltertoggle" is not returning the expected output. | DE212904 |
| Collector Compression is not working as expected for the Calculation collector. | DE212897 |
| In an enterprise setup, when you remove an archive from the system using Configuration Hub or Historian Administrator, the archive is being removed, but it is not being zipped and placed in the `Archives\Offline` directory. | DE208001 |
| When you add a Server-to-Server Distributor collector instance and try to access it using Configuration Hub in the Source Historian Server machine, it fails to fetch the instance details, and an error is being displayed. | DE213348 |
| Adding tags to the Server-to-Server Distributor Collectors using Configuration Hub is not functioning correctly.<br><br>**Workaround**: Add tags to the Server-to-Server Distributor Collector using Historian Administrator connected to Source Historian Server. | DE213362/ DE213694 |
| When using older Historian collectors to connect to Configuration Hub 2024, an error is displayed, and the Interface Details section of the collectors is not loaded.<br><br>**Workaround**: Upgrade all collectors to 2024, or install the 2024 Remote Collector Management Agent from the 2024 ISO on the Collector machine. | DE213981 |

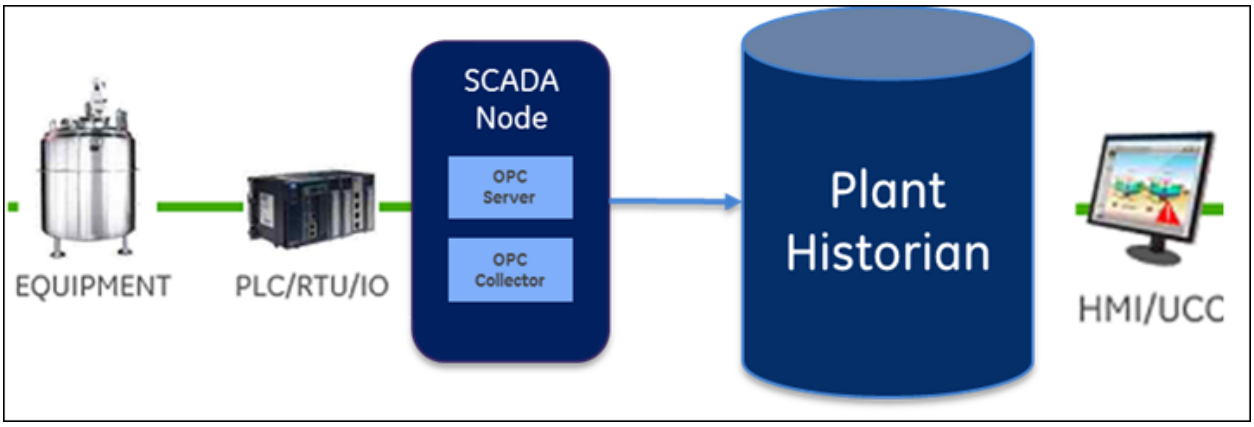
**Table 5. Limitations**

| Description | Tracking ID |
|---|---|
| Collector compression is not automatically applied to tags created for an MQTT Sparkplug B collector instance that was added with collector compression enabled. | DE212048 |
| Hierarchical browsing of tags is not supported for an OPC collector connected to an iFIX source. | DE204135 |
| Unable to import more than 50000 tags to Historian server for the OPC UA collector using Excel Add-in.<br><br>**Workaround**: It is recommended to add less than 30000 tags for a collection interval of 1 second or more. | DE212906 |

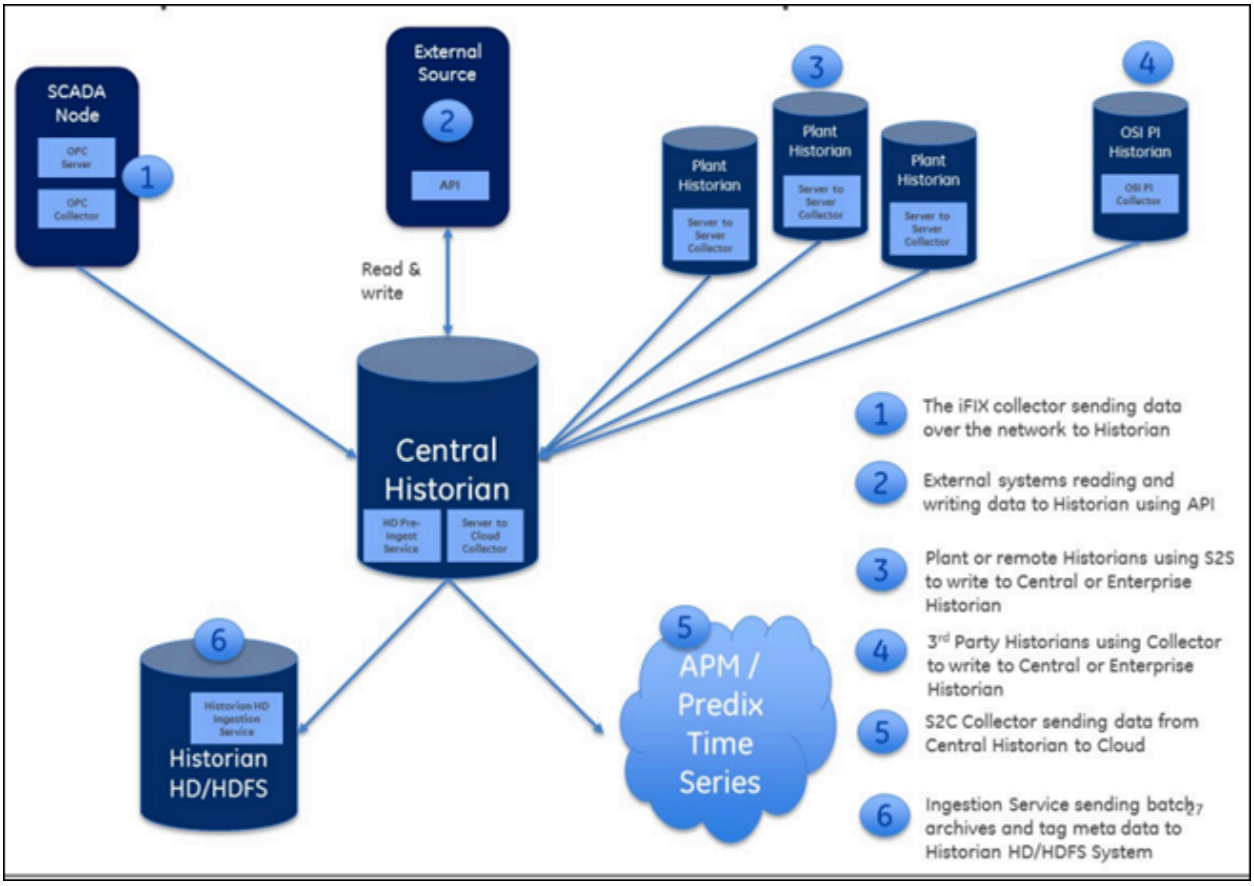
# Chapter 2. Getting Started Guide

## Overview of Historian

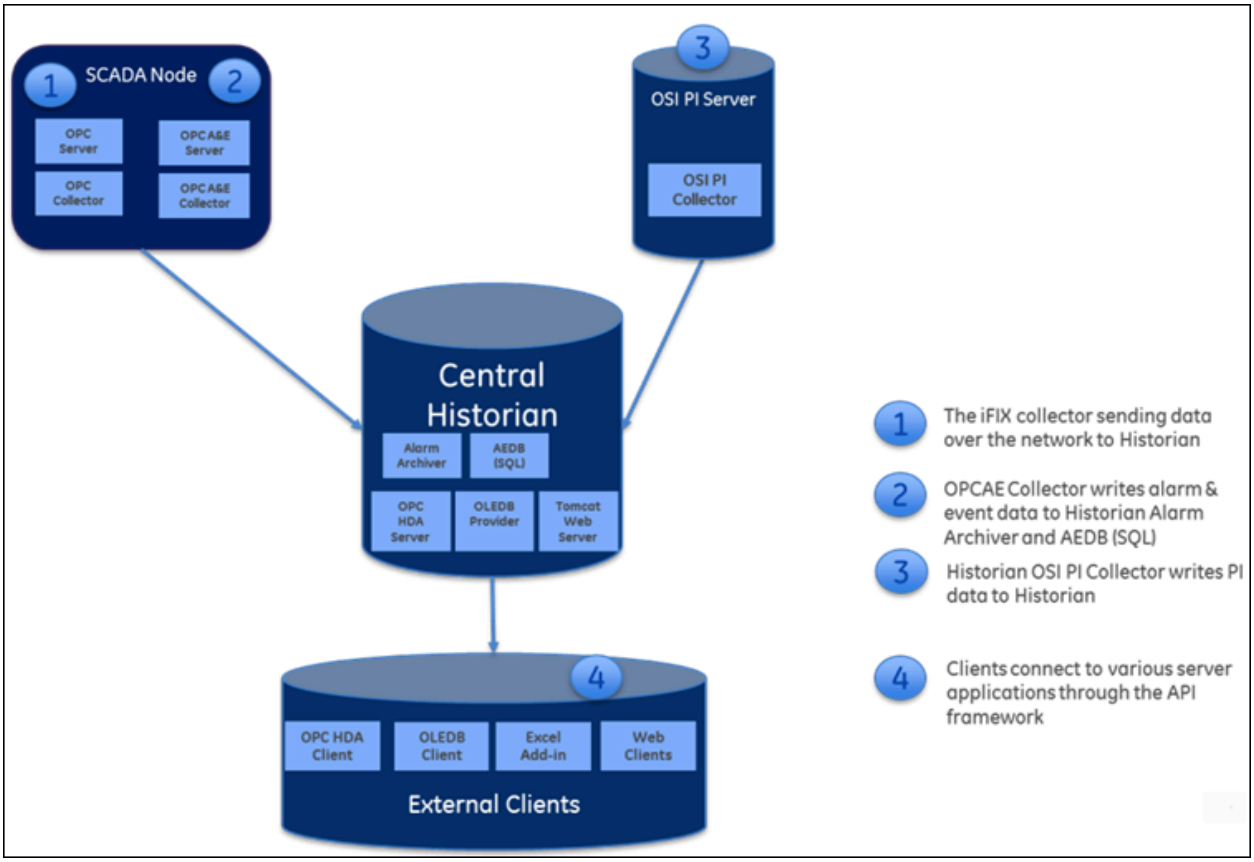
Proficy Historian is a high-performance data archiving system designed to collect, store, and retrieve time-based information at an extremely high speed. The following diagram shows an HMI or an OPC server from which data is collected and stored in Historian.




You can collect data from multiple SCADA systems and various applications, and store them in a central Proficy Historian server.




You can then use various clients to fetch and analyze this data.




Historian contains the following main components:

- **Data collectors:** Collect and analyze the tag data.
- **The Historian server:** Stores tag data.
- **Clients:** Retrieve tag data from the Historian server using APIs.

For information on how these components work in a Historian system, refer to System Architecture *(on page 69)*.

### Data Collectors

Collectors are applications that collect data from a wide variety of applications such as iFIX, CIMPLICITY, OPC servers, OSI PI, and text files (.csv or .xml). This data is then stored in the Historian server.

In addition, Historian contains the Calculation and the Server-to-Server collectors. The Calculation collector performs calculations and analyses on Historian data and stores the results in tags on the server. The Server-to-Server collector has the same calculation capabilities as the Calculation collector, but it stores the results in tags on a remote server.

Most collectors can perform first-order deadband compression, a browse-and-add configuration, and store and forward buffering.

> **Note:**
> Standard collectors that are included as part of the product will not consume a client-access license (CAL). Other interfaces developed by customers or system integrators using the Collector Toolkit or APIs will consume a CAL for each instance or connection.

**Bi-Modal Collectors:**

The Historian data collectors can send data to an on-premises Historian server as well as cloud destinations such as Google Cloud, Azure IoT Hub, AWS Cloud, and Predix Cloud. Therefore, these collectors are called bi-modal collectors. The following collectors, however, are not bi-modal collectors; they can send data only to an on-premises Historian server:

- The Calculation collector
- The File collector
- The HAB collector
- The iFIX Alarms and Events collector
- The OPC Classic Alarms and Events collector
- The OSI PI Distributor
- The Python collector
- The Server-to-Server distributor

### The Historian Server

The Historian server is the central point for managing all of the client and collector interfaces, storing data and (optionally) compressing and retrieving data.

In the Historian server, data is stored in files called data archives. These files contain all the tag data gathered during a specific period of time (for example, time-based archives such as daily archives). They have the .iha extension.

You can store data of various data types such as Float, Integer, String, Byte, Boolean, Scaled, and binary large object data type (BLOB). The source of the data defines the ability of Historian to collect specific data types. If you have the license to store the alarms and events data, the server also manages the storage and retrieval of OPC Alarms and Events in a SQL Server Express.

You can further segregate your tags and archives into data stores. A data store is a logical collection of tags used to store, organize, and manage tags according to the data source and storage requirements. A data store can have multiple data archives, and includes logical and physical storage definitions.

The primary use of data stores is segregating tags by data collection intervals. For example, you can put name plate or static tags where the value rarely changes in one data store, and put process tags in another data store. This can improve the query performance.

The Historian Data Archiver is a service that indexes all the data by tag name and timestamp and stores the result in an .iha file. The tag name is a unique identifier for a tag (which is a specific measurement attribute). For iFIX users, a Historian tag name normally represents a Node.Tag.Field (NTF). Searching by the tag name and time range is a common and convenient way to retrieve data from Historian. If you use this technique to retrieve data from the archive files, you need not know which archive file contains the data. You can also retrieve data using a filter tag.

**Clients**

Clients are applications that retrieve data from the archive files using the Historian API. The Historian API is a client/server programming interface that maintains connectivity to the Historian Server and provides functions for data storage and retrieval in a distributed network environment.

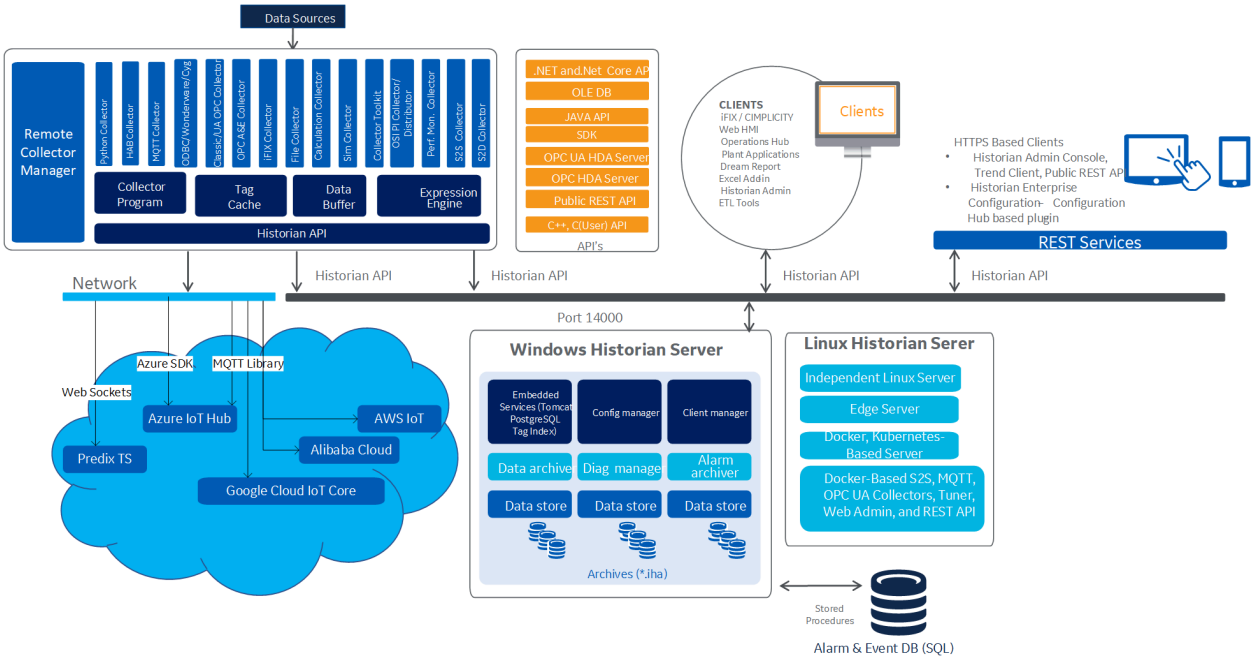
## System Architecture

**Standard or Stand-Alone Historian Architecture:**

In this type of system, there is a single Historian server. It offers the following unique capabilities and benefits for a sustainable competitive advantage:

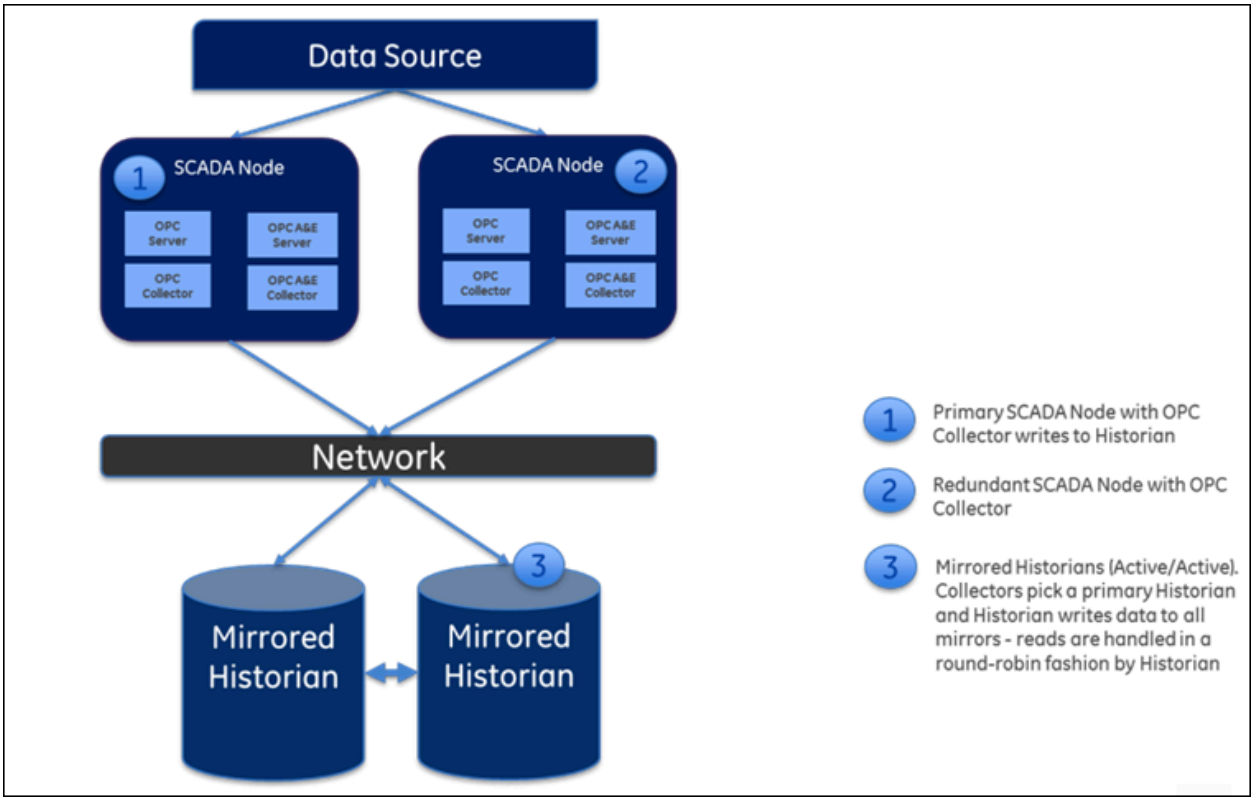
- Built-in data collection
- Good read/write performance speed
- Enhanced data security

The following image shows the architecture of a stand-alone Historian system (single server):




**Horizontally Scalable Historian system:**

In addition to the capabilities of a stand-alone Historian system, a horizontally scalable one offers data redundancy and high availability. You can have mirroring of stored data on multiple nodes to provide high levels of data reliability. Data mirroring also involves the simultaneous action of every insert, update, and delete operations that occur on any node. You can spread the data collection, server, administration, and client data retrieval functions across various nodes.



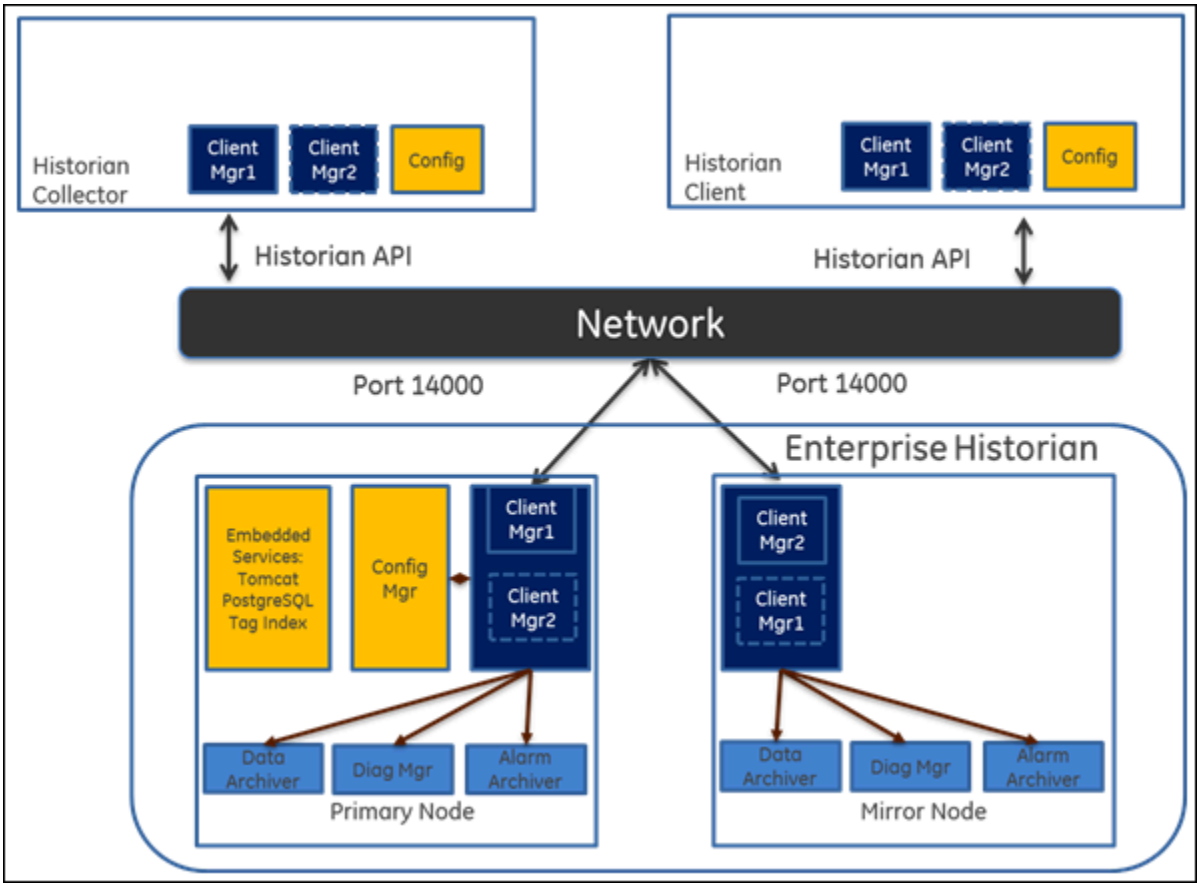

In a typical data mirroring scenario, one server acts as a primary server to which the clients connect. To create a mirror, you must add mirror nodes and establish a data mirroring session relationship between the server instances. All communication goes through Client Manager, and each Client Manager knows about the others.

When a client (either a writing collector or reading client) connects to the Client Manager, it gathers information about each Client Manager, along with all archive, tag, and collector configuration information, from the Configuration Manager, and stores this information locally in its Windows Registry.

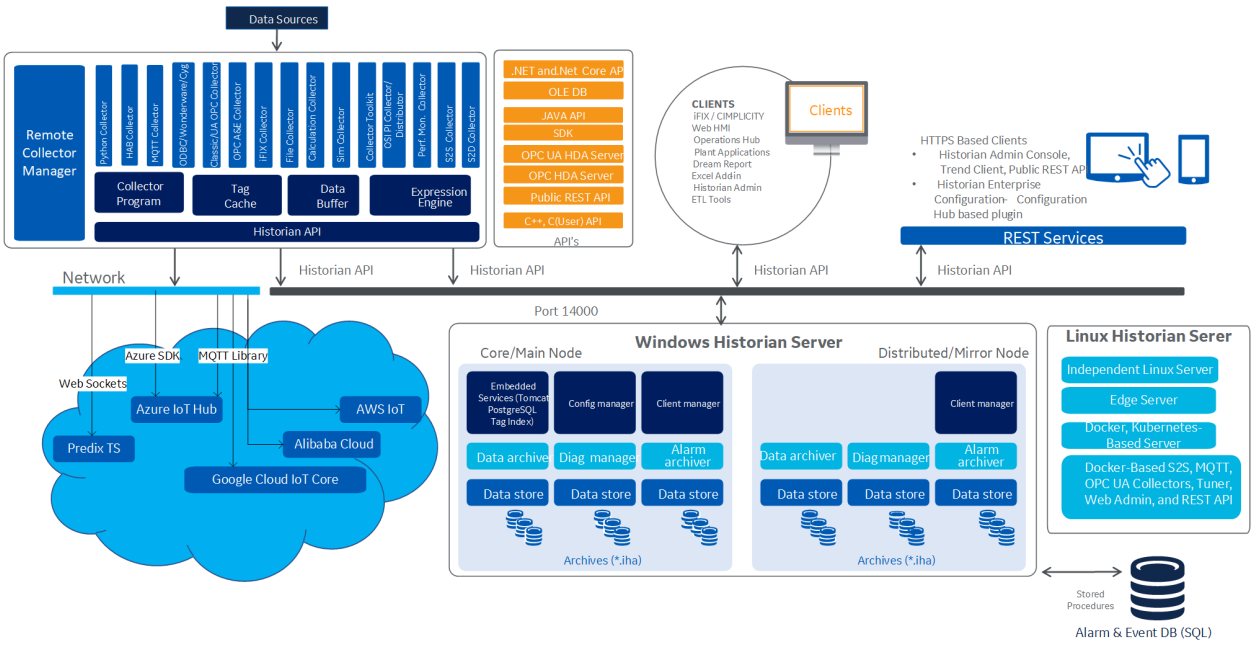
A relationship is then established between each remote client and a single Client Manager, which directs read and write requests across the other mirrors. If that relationship is broken, it will establish a new relationship with the next available Client Manager, which assumes the same responsibilities. This bond is maintained until that Client Manager is unavailable, and then the process of establishing a relationship with another Client Manager is repeated.

When more than one node is running, the Client Manager uses a "round robin" method between the good nodes to balance read loads. Each read request is handled by a node as a complete request.

Writes are sent independently but nearly simultaneously to any available data archiver so that the same tag shares a common GUID, name, timestamp, value, and quality as passed to it by the collector.




The following diagram shows the architecture of a horizontally scalable Historian system.



# System Components

A typical Historian system contains components for the following functions:

- Data collection/migration
- Data storage
- Data management, analysis, and monitoring
- Data retrieval

All these components communicate with the Historian server through the Historian API. This topic describes the functions performed by each component.

## Data Collection/Migration Components

Data collection/migration components are used to collect data from various sources and ingest the data into a Historian server (or to cloud). The following data collection/migration tools are used in Historian:

**Data Collectors**

Data collectors gather data from a data source based on a schedule or an event, process it, and forward it to the Historian server or a web socket for archiving. The following collector functions are common across all types of collectors (except the File collector):

- Automatically discovering available tags from a data source and presenting them to Historian Administrator and Configuration Hub.
- Providing options to send data to an on-premises Historian server or to cloud through a web socket connection
- Performing a first-level of data compression (collector compression).
- Responding to control requests, such as requests to pause or resume data collection.
- Maintaining a local cache of tag information to sustain collection while the server connection is down.
- Buffering data during loss of connection to the server and forwarding it to the server when the connection is restored.
- Optionally, automatically adjusting timestamps for synchronizing collector and archiver timestamps.
- Supporting the timestamp of both the collector and the device, as applicable.
- Scheduling data polling for a polled collection.

For mission-critical data collection, you can set up redundant collectors. Historian includes a mirroring option for high availability and load balancing, so the data is available all the time.

**The File Collector**

A File collector imports .csv and .xml files into the Historian server. These files can contain data, alarms, tag names, or other configuration information, and messages that you can import with a File collector.

**The Extract, Transform, and Load Tools**

Transferring data from one Historian server to another is typically performed by the data collectors. These tools provide a connected streaming data transfer mechanism (except the calculation and file transfer collectors). In a system where a steady network connection is not possible or not cost-effective, a periodic file-oriented data transfer is preferred. The Historian ETL tools consist of a comprehensive set of file-oriented data extraction, transfer, and loading tools.

**Migration Tools**

Migration tools are used to migrate existing Historian configuration and data and iFIX Alarms and Events collector data to the Historian server. Tags, collection rates, and deadbands for tags configured in Historian can be transferred by the migration tools.

**The Collector Toolkit**

Collector Toolkit is used to develop a customized collector. You can use Collector Toolkit to write programs that integrate with Historian and leverage the same configuration tools, redundancy schemes, and health monitoring as the collectors. It collects data and messages from a data source and writes them to a data archiver. Each deployment of a collector developed on the Collector Toolkit consumes a client access license (CAL).

## Data Storage, Analysis, and Maintenance Components

Data collected by the collection/migration components is stored in the Historian server (or cloud). You can then analyze and maintain the data using the following components:

**Historian Alarms and Events**

Historian Alarms and Events provides tools to collect, archive, and retrieve alarms and events data in Historian.

**Historian Administrator** *(on page 822)*

Historian Administrator provides a graphical user interface for performing Historian maintenance functions in a Windows environment including:

- Tag addition, deletion, and configuration.
- Maintaining and backing up archive files.

- Data collector configuration.
- Security configuration.
- Searching and analyzing system alerts and messages.
- Configuring the Calculation collector to create a new tag based on calculations, and storing the result as time series data.
- Setting up your OPC Classic HDA server and OPC UA HDA server.

**Historian Administrator**

The Web Admin console provides a dashboard, which displays the health of the system in one convenient location. You can view the following diagnostics details:

- **Data node diagnostics:** Displays the Historian servers connected to the system.
- **Collector diagnostics:** Displays the details of the faulty collectors.
- **Client diagnostics:** Displays the top five busiest clients connected to the system.

The dashboard provides interactive configuration management, which helps you configure mirror nodes, tags, collectors, data stores, and archives. However, the functionality of the Calculation collector and the ability to configure OPC HDA servers are not included in the Web Admin console.

The Web Admin console uses a CAL.

**The Historian Server**

The Historian server performs the following tasks:

- Manages all system configuration information.
- Manages system security, audit trails, and messaging.
- Services write and read requests from distributed clients.
- Performs final data compression.
- Manages archive files.

**Historian Diagnostics Manager**

The Historian Diagnostics Manager monitors the health of the Historian system and executes a few rules on the nodes, collectors, and clients, and generates the appropriate fault record. The details of these faults are displayed in the Web Admin console.

The following are the faults and their severity level:

| Fault Type | Fault Description | Fault Level |
|---|---|---|
| Collector Status Fault | Generated when the collector goes to the Unknown or Stopped state. | Error |
| Collector Overrun Fault | Generated when at least one overrun occurs on a collector in last 24 hours. | Warning |
| Collector OutOfOrder Fault | Generated when at least one OutOfOrder occurs on a collector in last 24 hours. | Information |
| Collector StoreForward Fault | Generated when the collector Last Data Sample Time Stamp is delayed by more than an hour. | Information |
| Collector ConnectDisconnect Fault | Generated when the collector is Disconnected and connected at least once in last 24 hours. | Information |
| Service DiskSpace Fault | Generated when a node disk space is about to reach its free space limit. | Warning |
| Client InActive Fault | Generated when a client is not active for the last one hour. | Information |
| Client BusyRead Fault | Generated when the client makes relatively more number of reads per minute. | Information |
| Client BusyWrite Fault | Generated when the client makes relatively more number of writes per minute. | Information |
| Client TimedOutRead Fault | Generated when the client makes a timed out read query. | Warning |

**Client Manager**

Client Manager acts as the client connection manager and message router for the system. It examines messages and forwards them to the correct data archiver or to Configuration Manager. This service is deployed only for mirrored systems.

**Configuration Manager**

Configuration Manager maintains and distributes the entire system configuration. There can be multiple Historian nodes but only one Configuration Manager. This Configuration Manager node is used to store system configuration, such as tag names, collector names, and Historian node names. This service is deployed only for mirrored systems.

**Configuration Hub**

Configuration Hub allows you to manage the Historian systems and its components, including:

- Creating a Historian system, and adding its components
- Creating mirror groups
- Creating and managing data stores
- Installing and managing collector instances

Using Configuration Hub, you can achieve high availability of servers in a Historian system.

**Remote Collector Manager**

Typically, collectors are distributed geographically, and so, accessing them can be challenging and not cost-effective. To overcome this challenge, the Remote Collector Management agent provides the ability to manage collectors remotely.

**Historian Tomcat Container**

An instance of Tomcat is used exclusively by Historian as an open source Java-based web server to support the Web Admin console and Trend Client. It supports SSL and the use of certificates for enhanced security.

**Proficy Authentication Tomcat Container**

An instance of Tomcat is used exclusively as an open source Java-based web server to support external Proficy Authentication.

**Proficy Historian PostgreSQL Database**

An instance of PostgreSQL is used exclusively to store tag names to improve searching for tags in the Trend tool and Web Admin console.

**Proficy Authentication PostgreSQL Database**

An instance of PostgreSQL is used exclusively to store Proficy Authentication details.

**Reverse Proxy Service**

Provides secure connection by supporting https protocol.

**Indexing Service**

The indexing service speeds up search results. It periodically queries the database, creates a tag index, and stores the information in the PostgreSQL database instance.

**Excel Add-in for Historian**

Excel Add-in is a very useful tool for presenting and analyzing data stored in archive files. Using this tool, you can design custom reports of selected data, automatically process the information, and analyze the results. You can also use it for performing tag maintenance functions in Historian, such as adding tags, importing or exporting tags, or editing tag parameters.

**Excel Add-in for Operations Hub**

The Excel Add-in for Operations Hub enables you to query historical data of objects and object types defined in Operations Hub.

Customers purchasing Historian Standard or Enterprise licenses now receive a no-cost license for the Operations Hub server and the Historian Analysis run-time application. The Operations Hub server enables customers to define an asset model including tag mapping. The Historian Analysis application is a pre-built Operations Hub HTML5 application that enables users to do advanced trend analyses, including the ability to make annotations.

**The OPC Classic HDA Server**

The OPC Classic HDA server reads the raw data stored in Historian and sends it to the connected OPC clients. The Historian OPC Classic HDA server complies with OPC Server HDA 1.20 standards.

**Historian SDK**

The Historian Software Development Kit (SDK) is designed for writing Visual Basic (VB) or Visual Basic for Applications (VBA) scripts. Using the SDK, you can develop your own scripts to perform selected repetitive or complex tasks or to make your own custom user interface. To use the SDK, create a VB/VBA project with the SDK as a project reference.

## Data Retrieval Components

Data retrieval components are used to retrieve data that is stored in the Historian server. Historian contains the following data retrieval components:

**Historian Alarms and Events**

Historian Alarms and Events provides tools to collect, archive, and retrieve alarms and events data in Historian.

**Client Manager**

Client Manager acts as the client connection manager and message router for the system. The Client Manager will examine messages and forward them to the correct Data Archiver or to the Configuration Manager. This service is deployed only for mirrored systems.

**Configuration Manager**

Configuration Manager maintains and distributes the entire System configuration. There can be multiple Historian nodes but only one Configuration Manager. This Configuration Manager node is used to store system configuration, such as tag names, collector names and Historian Node names. This service is deployed only for mirrored systems.

**Proficy Authentication Tomcat Container**

An instance of Tomcat is used exclusively by Historian as an open source Java-based Web server to support an external Proficy Authentication.

**Proficy Historian PostgreSQL Database**

An instance of PostgreSQL is used exclusively by Historian to store tag names to improve searching for tags in the Trend tool and Web Admin console.

**The OPC Classic HDA Server**

The Historian OPC Classic HDA server reads the raw data stored in Historian and sends it to the connected OPC Classic HDA collectors. The Historian OPC Classic HDA server is in compliance with OPC Server HDA 1.20 standards.

**The OPC UA HDA Server**

The Historian OPC UA HDA server retrieves historical process data from Proficy Historian, and sends it to OPC UA HDA clients. It dynamically updates the clients when tags are added and/or deleted in Historian. Clients that comply with this specification can connect to the OPC UA HDA server to retrieve data from Historian.

For information, refer to the OPC UA HDA Server section of the online documentation.

**User API**

The Historian User API is intended to provide high speed read/write access to Historian data and read access to Historian tags. There is no access to alarms, events, or messages.

Since the Historian User API is the basic building block for connectivity, all Historian functions, including data collection, administration, and data retrieval, use the Historian API. You can use the Historian User API to connect to a local Historian server or a remote one (by just providing the IP address or host name of the server).

Use the API to develop applications in C or C++, which read and write data to the Historian server when the Historian SDK and Historian OLE DB do not meet your project requirements for performance or programming language.

Historian allows you to develop both 32-bit and 64-bit User API programs.

> **Note:**
> If you want to build a 32-bit User API program on a 64-bit operating system, then you need to rename the `ihuapi32.lib` to `ihuapi.lib` and include it in your program.

**REST APIs**

Historian includes a REST API to connect your Java Web-based Clients with Historian data.

**Historian SDK**

The Software Development Kit (SDK) is designed for writing Visual Basic (VB) or Visual Basic for Applications (VBA) Scripts. Using the SDK, you can develop your own scripts to perform selected repetitive or complex tasks or to make your own custom user interface. To use the SDK, create a VB/VBA project with the SDK as a project reference. Refer to the *SDK Help* system for more information.

**Historian Client Access API**

The Historian Client Access API is a .NET Core assembly that interacts with Historian from any .NET Core applications. Since it works with .NET Core, it is platform-independent - you can use it on any operating system, such as Windows, Linux, and Mac OS.

> **Note:**
> You can still use the old Client Access API, which is a .NET assembly. It is installed when you install Client Tools.

**JAVA APIs**

Most open source, quick development applications rely on JAVA as their programing language. To enable easier integration with Historian, JAVA APIs are provided. The JAVA APIs support 64-bit Windows Operating Systems.

## About the Historian Server

The Historian server is the central point for managing all of the client and collector interfaces, storing data and (optionally) compressing and retrieving data.

In the Historian server, data is stored in files called data archives. These files contain all the tag data gathered during a specific period of time (for example, time-based archives such as daily archives). They have the .iha extension.

You can store data of various data types such as Float, Integer, String, Byte, Boolean, Scaled, and binary large object data type (BLOB). The source of the data defines the ability of Historian to collect specific data types. If you have the license to store the alarms and events data, the server also manages the storage and retrieval of OPC Alarms and Events in a SQL Server Express.

You can further segregate your tags and archives into data stores. A data store is a logical collection of tags used to store, organize, and manage tags according to the data source and storage requirements. A data store can have multiple data archives, and includes logical and physical storage definitions.

The primary use of data stores is segregating tags by data collection intervals. For example, you can put name plate or static tags where the value rarely changes in one data store, and put process tags in another data store. This can improve the query performance.

The Historian Data Archiver is a service that indexes all the data by tag name and timestamp and stores the result in an .iha file. The tag name is a unique identifier for a tag (which is a specific measurement attribute). For iFIX users, a Historian tag name normally represents a Node.Tag.Field (NTF). Searching by the tag name and time range is a common and convenient way to retrieve data from Historian. If you use this technique to retrieve data from the archive files, you need not know which archive file contains the data. You can also retrieve data using a filter tag.

The Historian server performs the following tasks:

- Manages all system configuration information.
- Manages system security, audit trails, and messaging.
- Services write and read requests from distributed clients.
- Performs final data compression.
- Manages archive files.

## About Tags

A Historian tag is used to store data related to a property.

For example, if you want to store the pressure, temperature, and other operating conditions of a boiler, a tag will be created for each one in Historian.

When you collect data using a collector, tags are created automatically in Historian to store these values. These tags are mapped with the corresponding properties in the source.

For example, suppose you want to store OSI PI data in Historian. You will specify the OSI PI tags for which you want to collect data. The OSI PI collector creates the corresponding tags in Historian, and it stores the values in those tags.

You can also choose to create tags manually (for example, to store the result of a calculation performed by the Calculation collector).

# Prerequisites

## Setting Up the Historian Environment

Identify the computers that will function as your clients, data collectors, administration workstations, and archiver.

1. Set up each computer.
   See Hardware Requirements *(on page 87)*, and refer to the user manual that accompanies each component for the setup information.
2. Use a login account with administrator rights so that you can install Historian later.
   See Software Requirements *(on page 92)*, and refer to the user manual that accompanies each software product for the setup information.
3. Activate the license key on your Historian server node. Additional licenses may be required on other nodes (such as mirroring and collector nodes) depending on your configuration requirements. See Activate the Historian License *(on page 82)*.
4. Disable the guest account in Windows security if you want to limit authentication to known Windows users only.
5. Ensure that the protocols and ciphers (TLS 1.0, 1.1, and 1.2) required to install Historian are available.

## Activate the Historian License

Advantage Licensing is the software system for activating and managing product licenses. Using the tools in licensing and our Customer Center website, you can view, activate, and manage licenses at your site.

Using Advantage Licensing, you can:

- View current licenses for the products residing on a computer.
- Choose a licensing method (Internet, local intranet, or file-based).
- Change licenses (Activate, Return, Refresh).

Historian is available in three license types:

- Essentials
- Standard
- Enterprise

The Essentials edition is included as the on-board Historian with the purchase of some iFIX and CIMPLICITY licenses, and cannot be licensed or sold outside of those packages. Essentials edition customers who require options available in the Standard or Enterprise editions or require more than a 1000 tags must purchase either a Standard or Enterprise License with the appropriate tag count.

Historian HD is sold and licensed separately from Historian. Historian HD provides the Historian user a standard method to move Historian tag configuration and historical archive data from a Windows environment to a Hadoop Distributed File System (HDFS). HDFS is the primary distribution storage used by Hadoop applications.

A component that is used only by the Historian HD license is installed with your Historian installation: the Historian Archive Ingestion service. This service is reserved for use only with the Historian HD big data analytics platform and is listed as Manual under Startup Type. Stopping this service does not impact the Historian functionality. Unless you are licensed to use Historian HD, do not attempt to start or monitor this service, as it may impact the ability to run the Historian Data Archiver service.

The following table provides information on the availability of each Historian component for each license type. Optional indicates that the component is not available by default, but can be purchased separately.

**Note:**
For a Calculation collector and a Server-to-Server collector, you can either opt for standalone use of bi-modal collectors or add Enterprise collectors to the Standard Historian license. Using these options, you can quickly and easily send data from one Historian server to another or directly to Predix Timeseries. For information on the pricing, contact the Support team.

| Component | Essentials | Standard | Enterprise | Distributed |
|---|---|---|---|---|
| **Server Functionality** | | | | |
| Data modification | Yes | Yes | Yes | Yes |
| Client Access Licenses (CALs) | 2 | 2500 | 2500 | 2500 |
| Cluster support | No | Yes | Yes | Yes |

| Component | Essentials | Standard | Enterprise | Distributed |
|---|---|---|---|---|
| Collector redundancy | Optional | Yes | Yes | Yes |
| Horizontal scalability (data mirroring) | No | No | Yes | Yes |
| Data stores | 5 | 10 | 20 | 20 |
| Data stores expansion (200) | No | No | Optional | Optional |
| Digital / Enumerated / Array Tags | Yes | Yes | Yes | Yes |
| Distributed Historian | No | No | No | Yes |
| Electronic signatures | No | Optional | Optional | Optional |
| The Extract, Transform, and Load (ETL) tools | No | No | Yes | Yes |
| Fault-tolerant computer support | Yes | Yes | Yes | Yes |
| Maximum historical tags | 1,000 | 50,000 | 20,000,000 | 20,000,000 |
| Microsecond support | No | Yes | Yes | Yes |
| The OLE DB provider | Yes | Yes | Yes | Yes |
| The OPC Alarms and Events server | No | Optional | Yes | Yes |
| The OPC Classic HDA server | Yes | Yes | Yes | Yes |
| The OPC UA HDA server | No | Yes | Yes | Yes |
| The Historian server | Yes | Yes | Yes | Yes |
| Remote Collector Management | No | No | Yes | Yes |
| SCADA buffer (10000 tags, 200 days) | Yes | Yes | Yes | Yes |
| User-Defined multi-field tags | No | Yes | Yes | Yes |
| **Client Functionality** | | | | |
| The Historian Model | No | Yes | Yes | Yes |
| The Historian Excel Add-in | Yes | Yes | Yes | Yes |
| Historian Administrator | Yes | Yes | Yes | Yes |
| Operations Hub Freemium | No | Yes | Yes | Yes |
| The Web Admin console | No | Yes | Yes | Yes |

| Component | Essentials | Standard | Enterprise | Distributed |
|---|---|---|---|---|
| Trend Client | No | Yes | Yes | Yes |
| Configuration Hub | Yes | Yes | Yes | Yes |
| **Collector Functionality** | | | | |
| Wonderware Collector with the cloud option | No | Yes | Yes | Yes |
| The Calculation collector | No | Available as a part of the Enterprise Collectors option | Yes | Yes |
| Collector Toolkit SDK | No | Yes | Yes | Yes |
| The CygNet collector with the cloud option | No | Yes | Yes | Yes |
| Expressions | No | No | Yes | Yes |
| The File collector | No | Yes | Yes | Yes |
| The HAB collector | No | Yes | Yes | Yes |
| The iFIX collector | Yes | Yes | Yes | Yes |
| The MQTT collector | No | Yes | Yes | Yes |
| The MQTT Sparkplug B collector | No | Yes | Yes | Yes |
| The ODBC collector with the cloud option | No | Yes | Yes | Yes |
| The OPC Classic Alarms and Events collector | No | Optional | Yes | Yes |
| The OPC DA collector with the cloud option | Yes | Yes | Yes | Yes |
| The OPC Classic HDA collector with the cloud option | No | Yes | Yes | Yes |
| The OPC UA Data Access (DA) collector with the cloud option | No | Yes | Yes | Yes |
| The OSI PI collector with the cloud option | No | Yes | Yes | Yes |
| The OSI PI distributor | No | Yes | Yes | Yes |

| Component | Essentials | Standard | Enterprise | Distributed |
|---|---|---|---|---|
| The Python collector | No | Available as a part of the Enterprise Collectors option | Yes | Yes |
| The Server-to-Server collector with the cloud option | No | Available as a part of the Enterprise Collectors option | Yes | Yes |
| The Simulation collector | Yes | Yes | Yes | Yes |
| The Windows Performance collector | No | Yes | Yes | Yes |

**Note:**
***** Starting Historian 7.2, SCADA buffer count is increased from 2500 to 10000. Historian Essentials license includes 2500 buffered tags and a 200-day circular buffer if permanent storage is less than 1000 tags (that is, if CIMPLICITY points are 1500 and below). It includes 10000 buffered tags and a 200-day circular buffer if permanent storage is 1000 tags (that is, if CIMPLICITY points are 5000 and above).

### To activate the Historian license:

1. If the license is already activated on your system, access License Client, select **Advanced > Clear license information on this computer**.
2. If you received an email containing an activation code, you must migrate to Advantage Licensing. Get the latest licensing software at http://digitalsupport.ge.com.

   If you did not receive an activation code, follow the instructions about M4 keys at http://digitalsupport.ge.com.
3. Download the Historian software.
4. Install the licensing software.
5. Activate the Historian license.

## Hardware Requirements

**Table 6. Historian Server**

| Hardware Component | Standard Historian | Enterprise Historian, Data Mirroring |
|---|---|---|
| RAM | 8 GB | 16 GB or 32 GB (recommended) |
| Disk size | 80 GB free hard-drive space | 250 GB (minimum) |
| Processor type | Intel Core i3 or i5 or i7 CPU or an equivalent AMD Phenom CPU | Intel Core-i5, i7 family, or equivalent |
| CPU | | Dual/Quad cores |
| CPU speed | 2.4 GHz | 2.8 GHz |
| Recommended CPU clock | 2.4 GHz | 2.8 GHz |
| Storage type | | SAS SSD with RAID Level 0 configured |
| Operating system | | • Microsoft® Windows® Server 2022 (64-bit).<br>• Microsoft® Windows® Server 2019 (64-bit).<br>• Microsoft® Windows® Server 2016 (64-bit).<br>• Microsoft® Windows® 11 (64-bit).<br>• Microsoft® Windows® 10 (64-bit), Professional ,or Enterprise Edition. |
| Tags | | Up to 50,000 |
| Years of data online | | 1 year |

**Table 6. Historian Server (continued)**

| Hardware Component | Standard Historian | Enterprise Historian, Data Mirroring |
|---|---|---|
| Other requirements | • A DVD-ROM drive.<br>• 100 Mbps TCP/IP-compatible network interface adapter for network communication and certain I/O drivers. | |

The size of a Historian server is determined by:

- The number of tags from which data is collected. The number of tags is an indicator of the number of concurrent users likely to access the system. The primary factor is server memory requirements; CPU load is a secondary factor. If the number of concurrent users is significantly different from the suggested guidelines, adjust server memory size accordingly.
- The rate of alarms and events collection.
- The frequency of data collection.
- The amount of data you want to keep online.

The following table provides the recommended hardware components for a Historian server with the Standard license based on the number of tags that you want to use. These recommendations may vary based on years of data online, update rate, data compression setting, and other tag configuration parameters.

| Hardware Component | Less Than 10,000 Tags | 10,000 to 50,000 Tags | 100,000 to One Million Tags | One Million to Two Million Tags | Two Million to Five Million Tags |
|---|---|---|---|---|---|
| RAM (in GB) | 8 GB/16 GB (recommended for a single node setup) | 16 or 32 | 16 or 32 | 16 or 32 | 32 or 64 |
| Disk Size (in GB) | 100 or 250 | 250 | 250 | 500 | 500 |
| Processor Type | Intel Core-i5, i7 family, or equivalent | Intel Core-i5, i7 family, or equivalent | Intel Xeon (56xx, E5 family or AMD Opteron 42xx/62xx family) | | |
| CPU | Dual/Quad core | Dual/Quad core | Dual/Quad core | 2-socket | 2-socket or 4-socket |

| Hardware Component | Less Than 10,000 Tags | 10,000 to 50,000 Tags | 100,000 to One Million Tags | One Million to Two Million Tags | Two Million to Five Million Tags |
|---|---|---|---|---|---|
| CPU Speed (in GHz) | 2.8 | 2.8 | 2.8 | 2.6 | 2.6 |
| CPU clock speed (in GHz) | 2.8 | 2.8 | 2.8 | 2.6 | 2.6 |
| Storage Type | SAS SSD with RAID level 0 configured | SAS SSD with RAID level 0 configured | Direct-attached or shared storage with SAS enterprise class drives. Hardware RAID controller with cache memory. SAN recommended over NAS | | High speed shared storage with SAS or SSD drive types. Hardware RAID controller with cache memory. SAN recommended over NAS. |
| Years of data online | 1 | 1 | 1 | 1 | 1 |

**Note:**

- The Historian server runs only on 64-bit versions of Windows.
- When possible, for performance reasons, consider using computers with multiple disk drives so that archives and buffers can be given their own drive. Or, multiple data stores can each have their own drive.
- Sustained event rate is 60-100 million samples per minute.
- Historian supports Intel Core i3, i5, i7 Duo based processors as long as they are compatible with the operating system.
- Historian does not support Titanium processors.

System performance may vary depending on the hardware specifications, operating system, and tuning parameters. The following table provides sample hardware specifications for medium-sized and large-sized servers.

| Hardware Component | For a Medium-Sized Server | For a Large-Sized Server |
|---|---|---|
| Processor type | Intel Xeon 5540 | Intel Xeon E5-2670 or E5-4650 |
| CPU | Dual socket | Dual socket or quad-socket |
| CPU speed (in GHz) | 2.5 | 2.7 |
| RAM (in GB) | 64 | 256 |

**Table 7. Collectors**

| Hardware Component | Recommendation |
|---|---|
| RAM | 8 GB |
| Disk size | 80 GB |
| Historian collectors | 32-bit or 64-bit<br>**Note:**<br>Wonderware Collector supports 64-bit only |

**Note:**

- Historian Collectors work as 32-bit applications on a 64-bit Windows operating systems using WoW64 mode (Windows-on-Windows 64-bit). However, you can read and write data from a 64-bit Historian Server.
- RAM and Disk Size required may vary based on the collectors available on the system.
- Recommended number of tags per collector is 20 to 30K.
- For iFIX systems, count each Node.Tag.Field (NTF) as a separate tag when you determine the size of the system. For example, FIX.FIC101.F_CV and FIX.FIC101.B_CUALM (current alarm) both count as tags, even though they are derived from the same iFIX tag.

**Microsoft Windows Server**

Many desktop-class computers are not certified to run Windows. Check the Microsoft website and your computer hardware vendor website for possible conflicts between your hardware and Windows server. These specifications are sufficient to meet the needs of a small pilot application. However, production system requirements may be significantly different depending on many application-specific factors. Please contact your product manager to review the requirements of your application.

**Table 8. Microsoft Cluster Service**

| Hardware Component | Requirement |
|---|---|
| CPU speed, processor type, and RAM | A 2.6 GHz clock-speed Intel Core i3 or i5 or i7 or Xeon or equivalent AMD Opteron CPU with minimum 8 GB RAM. |
| Disk size | A 80 GB free hard-drive space and a 40 GB shared SCSI hard-drive (RAID preferred). |
| Other requirements | Two 100Mbit TCP/IP-compatible network interface adapters for network communication and certain I/O drivers (One for public network, another for private network). |

**Note:**
The configuration of each server added to the cluster must be identical to the other servers in the cluster.

**Table 9. Data Mirroring and Redundancy Service**

| Hardware Component | Requirement |
|---|---|
| Operating system | A 64-bit Windows operation system. |
| RAM | Minimum 8 GB RAM.<br>**Note:**<br>If you are using single node setup, it is recommended to use 32 GB RAM. |
| Processor type | A dual core processor. |

Ensure that you are using the same hardware requirement for the mirror node as well.

**Network Speed**

For a large Configuration Hub setup, it is recommended that the network speed is 1 GBPS.

**Note:**

- If you are using a single node setup, then it is recommended to use 32 GB RAM.
- Ensure that you are using the same hardware requirement for the mirror node as well.
- You must have a minimum of 10 GB free space available for the data archiver to start.
- Many desktop-class computers are not certified to run Windows server. Check the Microsoft website and your computer hardware vendor website for possible conflicts between your hardware and Windows server. These specifications are sufficient to meet the needs of a small pilot application. However, production system requirements may be significantly different depending on many application-specific factors. Please contact your product manager to review the requirements of your application.

## Software Requirements

This topic describes the minimum software requirements for Historian.

- **Operating System:** Historian requires one of the following operating systems, with latest service packs or revisions:
  - Microsoft® Windows® Server 2022 (64-bit)
  - Microsoft® Windows® Server 2019 (64-bit)
  - Microsoft® Windows® Server 2016 (64-bit)
  - Microsoft® Windows® 11 (64-bit)
  - Microsoft® Windows® 10 (64-bit), Professional ,or Enterprise Edition.

**Note:**
The Historian server runs on a 64-bit Windows operating system only.

Historian 7.2 32-bit components such as Collectors, Excel Add-in 32-bit, Interactive SQL 32-bit, APIs, and Non-Web Administrator work as 32-bit application on 64-bit Windows operating systems using WoW64 mode (Windows-on-Windows 64-bit). However, you can read and write data from a 64-bit Historian Server.

If you use Historian 6.0 or later on Windows Server 2008 R2, you must go for a Full Installation and not Core Installation of Windows.

- **Network Interface Software:** The TCP/IP network protocol is required.
- **Microsoft®.NET Core or .NET Framework:** Historian requires the following .NET frameworks:To install the framework, you must configure your proxy server for internet access.
    - **Microsoft®.NET Core Framework 3.1:** This is required if you want to use the new Client Access API.
    - **Microsoft®.NET Framework 4.8:** This is required on the machine on which you will install the Excel Add-in for Operations Hub. In addition, this is required for Historian collectors.
    - **Microsoft®.NET Framework 4.6.1:** This is required for all the other Historian components. You can install it manually, or you will be prompted to download and install it while installing Historian.

> **Note:**
> If your machine is Firewall/proxy-enabled, Microsoft .NET Framework may not be installed automatically. In that case, before installing Historian, you must install Microsoft .NET Framework manually (if it is not available).

- **Microsoft® SQL Server®:** Historian requires one of the following 32-bit or 64-bit Microsoft® SQL Server® SQL server systems to configure archiving for alarms and events or to use Historian as a linked server:
    - Microsoft® SQL Server® 2022
    - Microsoft® SQL Server® 2019
    - Microsoft® SQL Server® 2017

> **Note:**
> The collation for your alarms and events database must match the collation of your SQL Server. This happens automatically by default unless the alarms and events database is moved to another SQL server.

- **Browser:** You can access the Web Admin console and Trend Client using the following browsers:
    - Firefox version 46 or later
    - Google Chrome version 92 or later

  To access Configuration Hub, use Google Chrome only.
- **Screen Resolution:** You can access Configuration Hub, the Web Admin console, and Trend Client using the following screen resolutions:
    - 1280 x 1024
    - 1366 x 768

- **Microsoft Excel:** The Excel Add-in for Historian or the Excel Add-in for Operations Hub requires any of the following versions of Excel:
  - Microsoft® Excel® 2021 (32 & 64 bit)
  - Microsoft® Excel® 2019 (32 & 64 bit)
  - Microsoft® Excel® 2016 (32 & 64 bit)
  - Microsoft® Excel® 2013 (32 & 64 bit)

- **Web Server:** The web server requires the following applications:
  - Microsoft®.NET Framework 4.6.1
  - Historian Client Tools 7.0 or later
  - OLE DB, User API, and Historian Client Access Assembly

## Compatibility with Other Products

Several other products work with Historian. The following is a general set of versions tested to work with the Historian 2024 product.

| Product | Supported Version |
|---|---|
| CIMPLICITY | 2022, 2023 |
| iFIX | 2022, 2023, 2024 beta |
| Proficy Plant Applications | 2023, 2024 beta |
| Workflow | 2.6 SP2 |
| Proficy Operations Hub | 2023.1, 2024 beta |
| Configuration Hub | 2024 |
| Proficy Authentication | 2024 |
| Dream Reports | 2023 Patch 1 |
| Habitat | 5.12 |

Historian REST APIs are required to integrate between Historian and Operations Hub. Historian REST APIs are installed automatically when you install Historian Web-based Clients *(on page 155)*.

## Supported Regional Settings, Data Types, and Date/Time Formats

### Supported Regional Settings

Historian supports the following regional settings available in Control Panel:

- Decimal symbol - one character
- Digit grouping symbol
- List separator - one character
- Time style
- Time separator
- Short date style
- Date separator

### Supported Data Types

| Data Type | Size |
|---|---|
| Single Float | 4 bytes |
| Double Float | 8 bytes |
| Single Integer | 2 bytes |
| Double Integer | 4 bytes |
| Quad Integer | 8 bytes |
| Unsigned Quad Integer | 8 bytes |
| Unsigned Single Integer | 2 bytes |
| Unsigned Double Integer | 4 bytes |
| Byte | 1 byte |
| Boolean | 1 byte |
| Fixed String | Configured by user. |

| Data Type | Size |
|---|---|
| Variable String | No fixed size. |
| Binary Object | No fixed size.<br><br>Historian does not support the use of the Binary Object data type with the Data Collectors. Refer to the SDK online Help for more information on working with BLOB data types. |
| Scaled | 2 bytes |

### Supported Date Formats

Historian supports the following short date formats, some of which may not be available in certain language versions of Windows:

- dd/mm/yy
- dd/yy/mm
- mm/dd/yy
- mm/yy/dd
- yy/dd/mm
- yy/mm/dd

Avoid changing the time style or short date style in regional settings to values that are outside of the standard styles provided.

## Optimize Performance

You can optimize performance in the following ways:

- **Optimize virtual memory:** Through the use of paging files, Windows allocates space on your hard drive for use as if it were actually memory. This space is known as virtual memory. This topic describes how to optimize the virtual memory on the Historian archiver computer.

  **Note:**
  If the paging file is set to grow dynamically, your system may experience severe performance problems during runtime. To ensure optimal performance, the Initial Size and Maximum Size values for the paging file must be the same so that the paging file does not grow dynamically. For more information on creation and sizing of Windows paging files, refer to Microsoft Windows Help.

- **Optimize the server performance:** If the file sharing and printer sharing options on the computer on which you want to install Historian is set to maximize data throughput, it can lead to excessive paging when dealing with large files, which can interfere with applications like Historian. You can change these settings to optimize the performance.

1. To optimize virtual memory:
    a. Access Control Panel, and then select **System > Advanced system settings > Advanced**.
    b. Under **Performance**, select **Settings > Advanced**.
    c. Under **Virtual Memory**, select **Change**.
    d. In the **Initial size** and **Maximum size** fields, enter a value equal to three times your physical memory.
    e. Select **Set**, and then select **OK**.
2. To optimize server performance:
    a. Access Control Panel.
    b. Select **Network and Sharing Center**.
    c. Select the network that you use.
    The **<network name> Status** window appears.
    d. Select **File and Printer Sharing for Microsoft Networks**, and select **Properties**.
    e. Select **Properties**.
    f. Ensure that the **Maximize Data Throughput for Network Applications** option is selected.
    g. Select **OK**.

## Enable Trust for Proficy Historian for a Self-signed Certificate

During Historian installation, a self-signed certificate is generated that you use with Historian web applications. A self-signed certificate is a certificate that is signed by itself rather than signed by a trusted authority. Therefore, a warning appears in the browser when connecting to a server that uses a self-signed certificate until it is permanently stored in your certificate store. This topic describes how to ensure that Google Chrome trusts the self-signed certificate.

1. Using Google Chrome, access the site to which you want to connect.
    A message appears to inform you that the certificate is not trusted by the computer or browser.
2. Select **Not Secure** in the URL, and then select **Certificate**.
    The **Certificate** window appears.

3. Select **Certification Path**, select the root certificate, and then select **View Certificate**.
   The **Certificate** window appears, displaying the **General**, **Details**, and **Certification Path** sections.
4. Select **Details**, and then select **Copy to File**.
5. Follow the on-screen instructions to save the certificate to a local file. Use the default format: **DER encoded binary X.509 (.CER)**.
6. Right-click the .CER file that you have exported, and select **Install Certificate**.
   The **Certificate Import Wizard** window appears.
7. Select **Trusted Root Certificate Authorities**, and then select **OK**.

   **Note:**
   Do not let the wizard select the store for you.

   A **Security Warning** window may appear. If it does, ignore the message by selecting **Yes**. The certificate is installed.
8. Restart the browser, and connect to the server.
9. Open the URL authenticated by the certificate.
   If error messages do not appear, the certificate is successfully imported.

## VMWare Support

Historian provides support for VMware ESXi server version 5.0 and later. The virtualization capability provided by VMware lets you run multiple virtual machines on a single physical machine, with each virtual machine sharing the resources of that one physical computer. Please be aware that while we have tested VMware ESXi 5.0 and above, issues with the VMware software or the virtualized environment are outside the scope of GE Vernova's responsibility. You must use VMWare Compatibility Hardware and Software before installing Historian 7.0 or greater Data Archiver on a Virtual Machine. For the current release, the only supported type of Proficy licensing for use with VMware is keyless (software) licensing.

**Note:**
VMware Player is not supported.

**Important:**
Advanced features of the ESXi server (such as VMotion, High Availability, and Clustering support) have not been tested with Historian.

For information regarding VMware compatibility and its supported software and hardware environments, refer to http://www.vmware.com/resources/guides.html.

## VMWare Best Practices and Limitations

### Disk Growth

To prevent disk growth during run time, make sure you pre-allocate the hard disk in your VMware image.

> **Important:**
> If the VMware disk needs to grow at runtime because of IHA growth or creation, the Data Archiver will be slowed. If there is not enough disk space on the host machine to grow the VMware disk, the archiver may lose data.

### Suspended Images/Power Metered Images

ESXi servers have power meter functions and options as well as the ability to suspend images to conserve power. We do not recommend or support these functions due to the potential effects on the Guest operating system, specifically in regards to polling I/O and timely updates.

### I/O Devices and Connections and VMware

There are a multitude of devices and methods of communications on the market. These devices may be used if you can successfully connect them from the virtual machine through the physical HOST, but we do not support the setup of that connection. Be aware that device drivers used to write to proprietary cards for the ESXi HOSTS as part of virtual device setup can cause issues.

### USB Controller Limitations

The USB controller has these limitations when using Historian and VMware:

- Minimum virtual hardware version 7 is required.
- Only one USB controller of each type can be added to a virtual machine.
- The USB arbitrator can monitor a maximum of 15 USB controllers. If your system includes an additional number of controllers and you connect USB devices to these controllers, the devices are not available to be passed through to a virtual machine.
- You must add a USB controller to a virtual machine before you can add a USB device.
- You must remove all USB devices from a virtual machine before you can remove the controller

### USB Device Limitations

USB devices have these limitations when using Historian and VMware:

- A virtual machine may have up to 20 USB devices attached to it; however, each unique USB device can only be attached to one virtual machine at a time.
- Unsupported USB devices may not interact as expected with other ESXi features.

**Additional VMware Notes**

GE Vernova cannot guarantee the performance of the Historian software in a virtualized environment due to the wide range of parameters associated with the hardware, configuration, memory settings, third-party software installations, and the number of virtual machines running; all of which can affect performance. Therefore, GE Vernova cannot provide support related to the performance of the Historian software running on a virtual machine if it is determined that the issue is related to the virtual environment. Also, GE Vernova does not provide support or troubleshoot a customer's virtual machine infrastructure.

It is the responsibility of you, the customer, to ensure that the performance of the Historian software and any third-party applications (especially those not recommended by GE Vernova) are adequate to meet the needs of your run mode environment. GE Vernova does not support issues related to functionality that is not available as a result of running in a virtual machine infrastructure. Examples include the functionality of card level drivers such as those for the Genius® family of drivers, the Allen-Bradley® DH/DH+ drivers, the Cyberlogic's MBX® Driver for the SA85 card, as well as functions requiring direct video access. Check with the vendor of your third-party application for support statements regarding that third-party product's ability to run in a virtualized environment.

For more detailed information regarding VMware specifications and requirements, visit the VMware web site: http://www.vmware.com/resources/compatibility/search.php.

# Installation

## Historian Installation Workflow

1. Design your system architecture.

   Decide what collectors to instantiate on which nodes, which computers to designate as the Historian server and Historian Administrator, whether or not they will be web-based, and how much memory and disk space you can assign to buffers and archives. Record the computer names of each node.

2. Ensure that data sources are installed.
3. Set up your Historian environment *(on page 82)*.

4. On the server node, launch the installer, select **Install Historian**, and follow the on-page instructions to install Historian *(on page 105)* on a single server on in a distributed environment.
5. Activate your product license *(on page 82)*.
6. Install the collectors *(on page 142)*.
7. Restart your computer if prompted to do so.
8. As needed, install Web-based Clients *(on page 154)*.
9. Perform post install setup for MTLS features, by installing certificates. For more information, refer to Overview of certificate-based security.
10. For the Windows-based Historian Administrator clients, start the Administrator from **Historian Startup Group**.

    When the home page for Historian Administrator appears, you are ready to set up archives, collectors, and tags in the **Data Store Maintenance**, **Collector Maintenance**, and **Tag Maintenance** pages.

The following table provides a list of installation options available in Historian, along with purpose of installing each one.

| Installation Option | When to Install |
|---|---|
| Historian Server *(on page 104)* | Installing the Historian server is mandatory to work with Historian. If you want to use Web-based Clients, you must provide the Proficy Authentication server details while installing the Historian server.<br><br>When you install the Historian server, the following components are installed as well:<br><br>• **The RemoteCollectorConfigurator utility:** A command-line tool, which allows you to manage collectors remotely.<br>• **The Proficy Authentication Configuration tool:** A utility that allows you to specify the Proficy Authentication server details to match with the Proficy Authentication server used by Web-based Clients. |
| Alarms and Events *(on page 140)* | Install Alarms and Events if you want to retrieve and store alarms and events data from any OPC- |

| Installation Option | When to Install |
|---|---|
| | compliant alarms and events server using the OPC Classic Alarms and Events collector. |
| Collectors *(on page 142)* | Installing collectors is mandatory to collect and store data in Historian.<br><br>When you install collectors, all the collectors and the Remote Management Agents are installed. You must then create instances of each collector and manage them using Configuration Hub.<br><br>When you install collectors, if iFIX/CIMPLICITY are installed on the same machine, instances of the following collectors are created automatically:<br><br>• The iFIX collector<br>• The iFIX Alarms & Events collector<br>• The OPC Classic Data Access collector for CIMPLICITY<br>• The OPC Classic Alarms and Events collector for CIMPLICITY |
| Client Tools *(on page 150)* | Install Client Tools if you want the following components:<br><br>• Client Tools<br>• Historian Administrator<br>• OLE DB driver and samples<br>• The OPC Classic HDA server<br>• User API and SDK<br>• Historian Client Access API<br>• Collector Toolkit |
| Web-based Clients *(on page 154)* | Install Web-based Clients if you want to manage Historian administrative tasks and analyze the data using components such as Configuration Hub, the Web Admin console, Trend Client, and REST APIs. |

| Installation Option | When to Install |
|---|---|
| | To use Web-based Clients, you need Proficy Authentication (UAA server) to handle user authentication. It provides identity-based security for applications and APIs. It supports open standards for authentication and authorization, including Oauth2.<br><br>During the Web-based Clients installation, you can choose to install a Proficy Authentication instance and Configuration Hub, or you can use existing ones. |
| Historian Remote Management Agents *(on page 187)* | Remote Management Agents (RMA) include Remote Collector Manager, which is used to manage collectors remotely.<br><br>RMA is installed automatically when you install collectors. If, however, you are using RMA version 8.1, and you want to upgrade only RMA (and not the collectors), use this option. |
| Excel Add-in for Historian *(on page 195)* | Install Excel Add-in for Historian make bulk changes to tag parameters using Excel, and then import it to Historian. You can also perform mathematical, retrieve selected data, generate reports and charts, and so on. |
| Excel Add-in for Operations Hub *(on page 198)* | Install Excel Add-in for Operations Hub if you want to query historical data of objects and object types defined in Operations Hub. |
| ETL Tools *(on page 202)* | Install the Historian Extract, Transform, and Load (ETL) tools if you want to transfer data where there is limited internet connectivity. |
| The OPC UA HDA Server *(on page 191)* | Install the OPC UA HDA server if you want to use Historian as an OPC UA HDA server. You can then connect any OPC UA HDA clients with the server. |
| Standalone Help *(on page 205)* | Install Standalone Help to access the Historian product documentation offline. |

## About Installing the Historian Server for the First Time

You can choose one of the following types of installation:

- **Single server:** This is for a stand-alone Historian system, which contains only one Historian server. This type of system is suitable for a small-scale Historian setup.
- **Mirror primary server and distributed/mirror node:** This is for a horizontally scalable Historian system, which contains multiple Historian servers, all of which are connected to one another. This type of system is used to scale out the system horizontally. For example, if you have 5,00,000 tags in your Historian system, you can distribute them among the various servers to improve performance.

  In this setup, one of the nodes acts the primary server, whereas the others are the distributed/mirror servers. Configuration Manager and the embedded web services are installed only on the primary server, which are used by the distributed/mirror servers as well.

  When the Archive Duration property is changed in a distributed environment, the changes will take effect after 15 minutes.

  The distributed environment works only for tag data; it does not work for alarms and events data. Therefore, do not install alarm archiver in a distributed environment.

  In this setup, one of the nodes acts the primary server, whereas the others are the distributed/mirror nodes.

For all these types of installation, you can use the GUI-based installer *(on page 105)* or the Command Prompt window *(on page 114)*. You can also install the Historian server in a cluster environment *(on page 119)*.

This section provides the high-level steps in installing the Historian server for the first time. You can perform the same steps to upgrade the server. If you are upgrading from either Historian 6.0 Enterprise or previous releases of Historian 7.2 (including any of the service packs), both Client Manager and Configuration Manager services will be removed. However, this will have no impact on your data or use of Historian unless you intend to use a distributed system.

**Note:**
The number of alarms in the Historian Alarms and Eventss database, and the frequency of new events being added during the installation, impact the time it takes to install Historian. For example, an installation for a system with 1.5 million alarms can take up to three hours to complete.

**Important:**
You cannot use size-based archives with a Historian Mirror Primary Server and Historian Mirror Node installation. This is because having archives of different sizes introduces archive synchronization risks in a mirrored environment. The restriction is enforced on all Historians, even those not using mirroring.

**Important:**
Be aware that you cannot close your current archive with a Historian Mirror Primary Server and Historian Mirror Node installation. This is because closing the current archive introduces archive synchronization risks in a mirrored environment. The restriction is enforced on all Historians, even those not using mirroring.

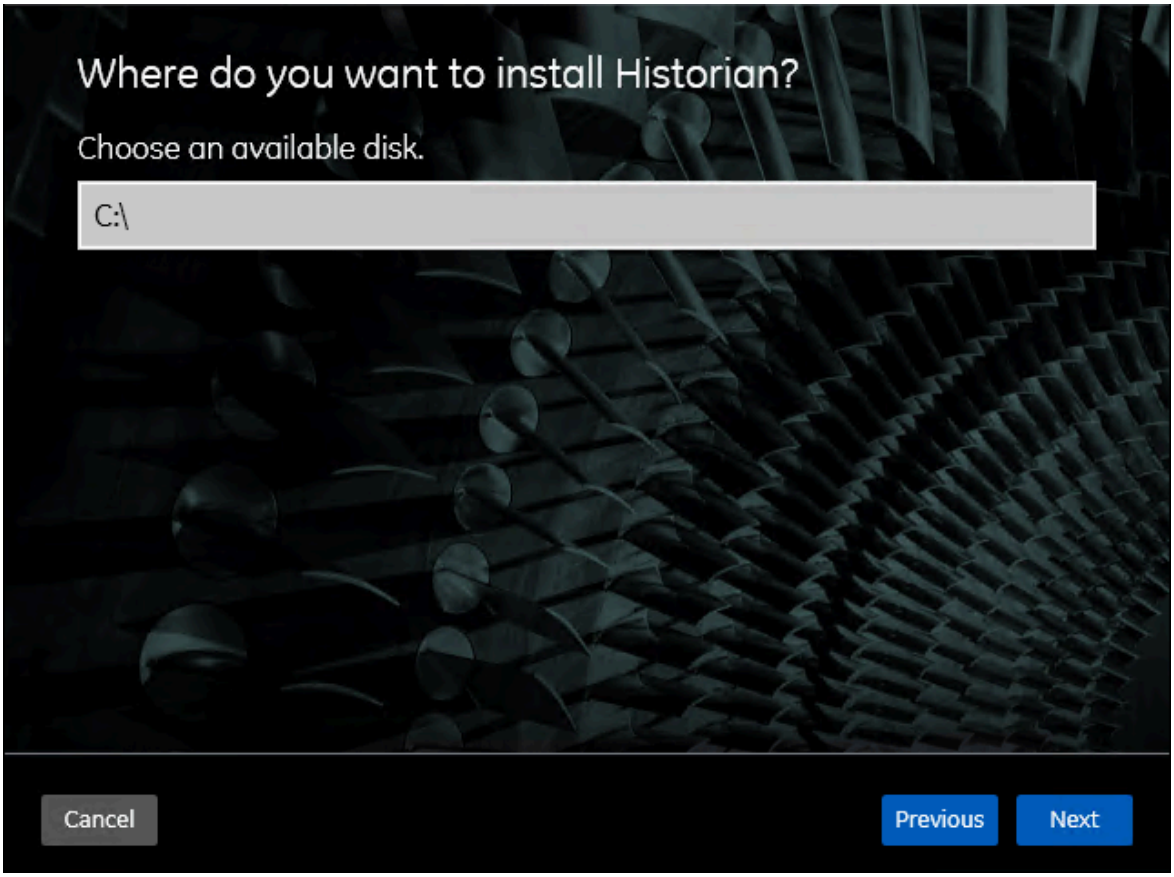
## Install the Historian Server Using the Installer

- set up the Historian environment.
- If you want to install web-based clients and view Historian license information on Configuration Hub, you must provide the Configuration Hub and Proficy Authentication server details during installation. Therefore, ensure that you have already set up Proficy Authentication in Configuration Hub. For more information on setting up Proficy Authentication in Configuration Hub, refer to https://www.ge.com/digital/documentation/confighub/version2023/t_authentication_setup.html.
- If you are changing the role of a Historian server that was previously a distributed/mirror server to any other configuration (single-server or mirror primary server), you must first uninstall Historian.
- If you are installing a distributed/mirror server, use the same configuration, license key, installation drive, Proficy Authentication instance, and domain as the primary server.

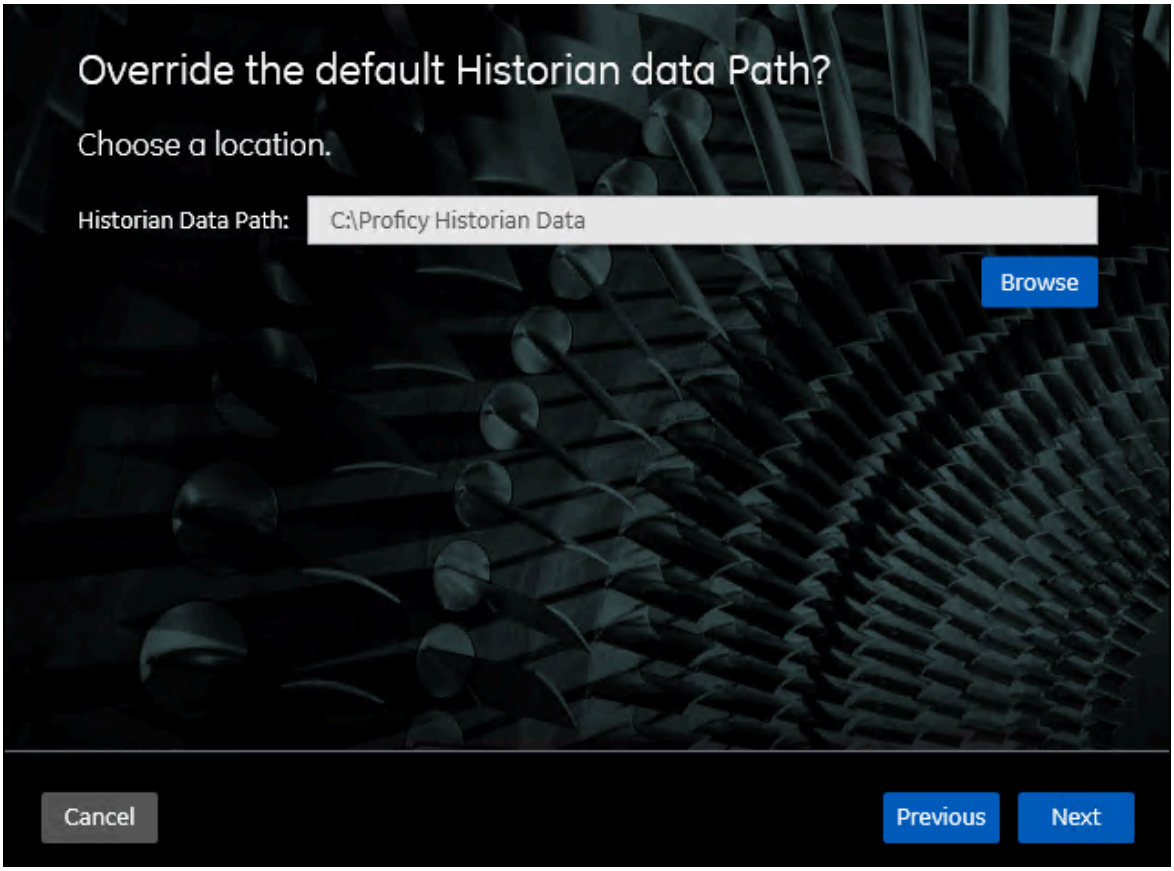
This topic describes how to install the Historian server using the installer.

You can also install it at a command prompt.

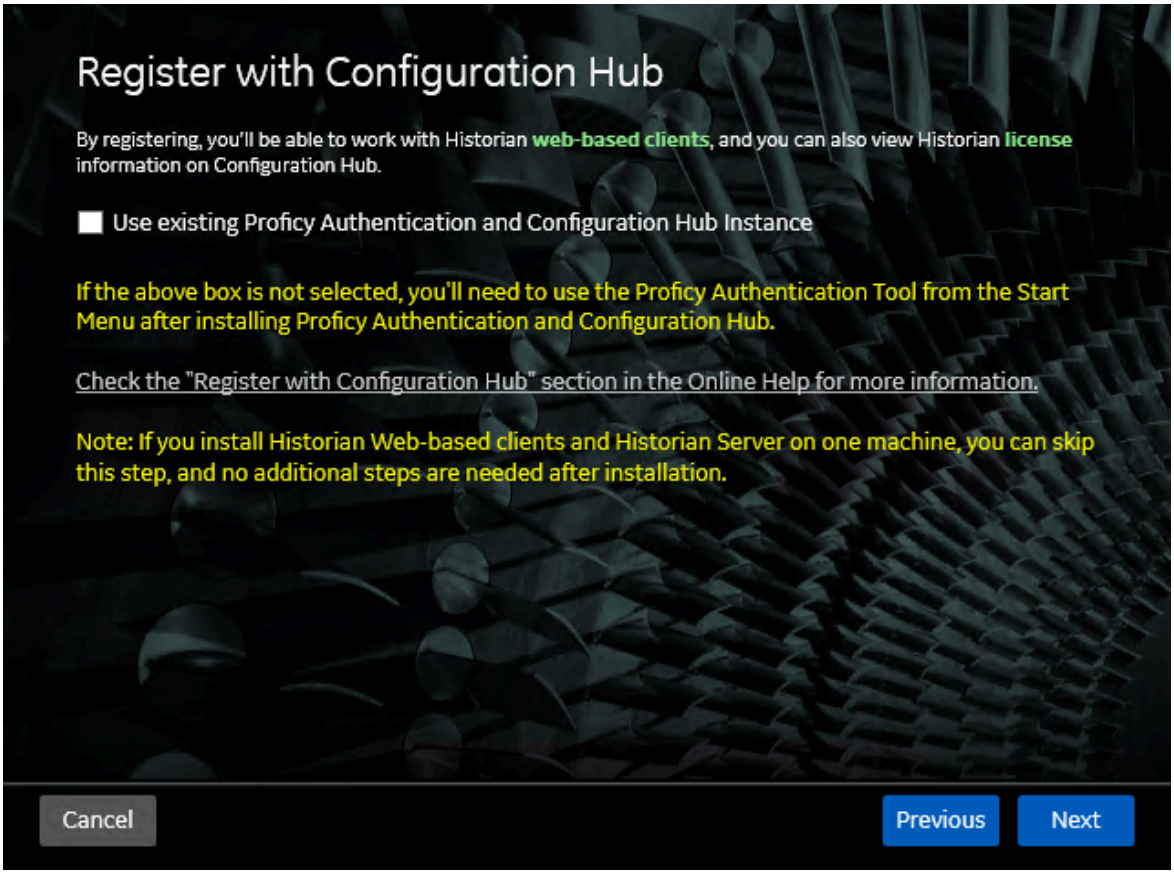
1. Log in as an administrator to the machine on which you want to install the Historian server.
2. Run the `InstallLauncher.exe` file.
3. Select **Install Historian Server**.
   The welcome page appears.
4. Select **Next**.
   The license agreement appears.
5. Select the **Accept** check box, and then select **Next**.
   The **Where do you want to install Historian?** page appears.




6. If needed, change the default installation drive of the Historian server, and then select **Next**.
   The **Override the default Historian data Path?** page appears.




7. If needed, change the default folder of the log files, and then select **Next**. If you want to include the Historian server in a cluster, enter the path to the shared folder of the cluster.
   The **Register with Configuration Hub** page appears.



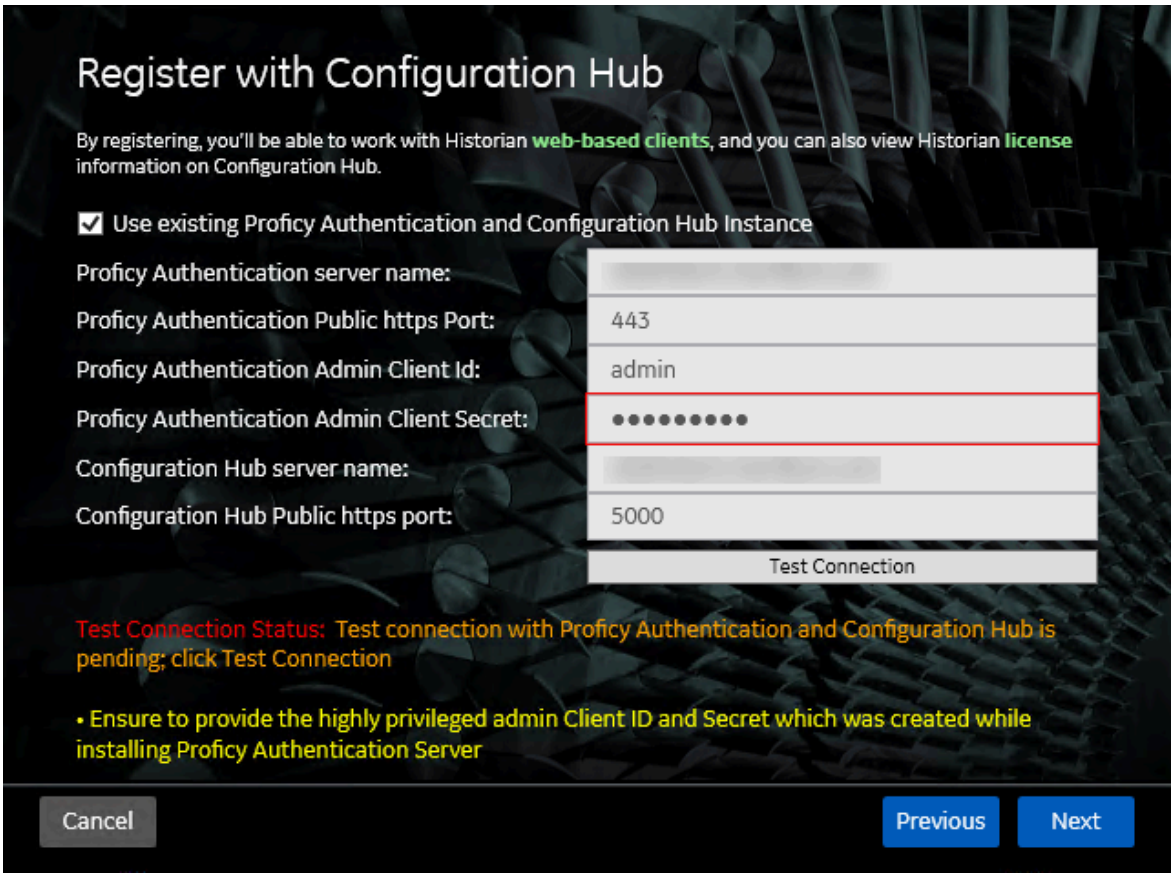

8. Select the **Use existing Proficy Authentication and Configuration Hub Instance** check box, and provide values as described in the following table.

   This step is needed only in the following cases:
   - The Historian Server and the Web-based clients will be on two different machines.
   - You already installed Proficy Authentication and Configuration Hub, and you want to use the Historian Web-based clients and view the Historian license information on Configuration Hub.

     If you do not select the **Use existing Proficy Authentication and Configuration Hub Instance** during the installation, to use the Historian Web-based clients and view the Historian license information on Configuration Hub, you must use the Proficy Authentication Tool from the Start menu to register with Proficy Authentication and Configuration Hub servers.

     If you do not have Proficy Authentication and Configuration Hub installed, and you intend to use the Historian Web-based clients and view the Historian license information on Configuration Hub, you can install them while installing the Web-based clients.

**Note:**
Proficy Authentication is required for user authentication. It provides identity-based security for applications and APIs. It supports open standards for authentication and authorization, including Oauth2.




| Field | Description |
|---|---|
| **Proficy Authentication server name** | Enter the name of the machine on which the Proficy Authentication server is installed. If the machine uses a fully qualified domain name (FQDN), provide the FQDN. By default, the local hostname is considered. |
| **Proficy Authentication Public https port** | Enter the port number used by the Proficy Authentication service. The default value is 443. Ensure that this port number matches the one |

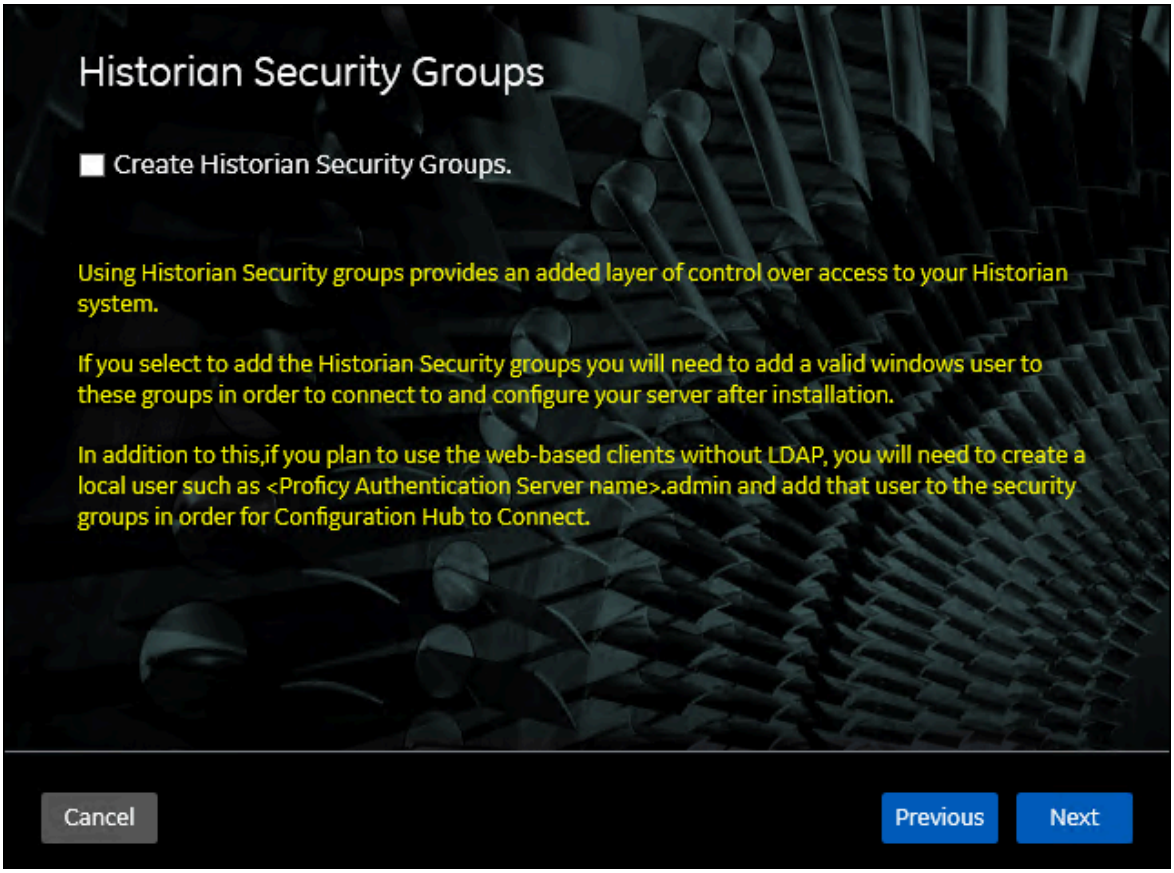
| Field | Description |
|---|---|
| | on the **TCP Port Assignments** page during Web-based Clients installation. |
| **Proficy Authentication Admin Client Id** | The client ID to connect to the Proficy Authentication service. |
| **Proficy Authentication Admin Client Secret** | The password to connect to the Proficy Authentication service. |
| **Configuration Hub server name** | The server name or the FQDN of the existing Configuration Hub server, as displayed in the address bar of the browser when you access Configuration Hub from the machine where Configuration Hub is installed. |
| **Configuration Hub Public https port** | The web server (https) port that you want to use for Configuration Hub. By default, it is 5000. |
| **Test Connection** | Option to test the status of the connection with external Proficy Authentication server. |

**Note:**
If you change the Proficy Authentication server for Web-based Clients later, you must also change the Proficy Authentication server for the Historian server. This can be done using the Proficy Authentication Configuration Tool located at `<Install Drive>:\Program Files\Proficy\Proficy Historian\x64\Server` without the need to install the Historian server again. Alternatively, you can search for the Proficy Authentication Tool in the Windows search bar and open it.

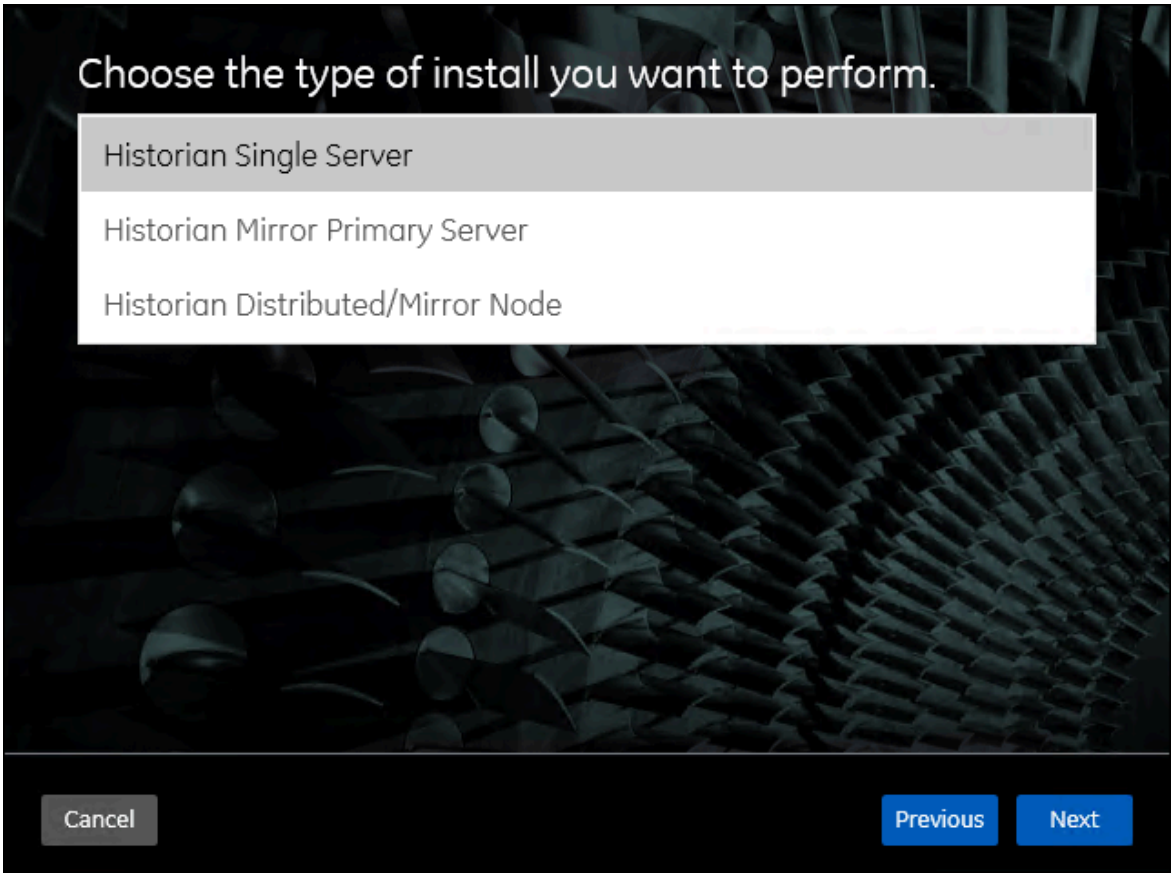
9. Select **Next**.
   The **Historian Security Groups** page appears.

   Using Historian security groups provides an added layer of control over access to your Historian system.

   By default, the option to create Historian security groups is not selected.

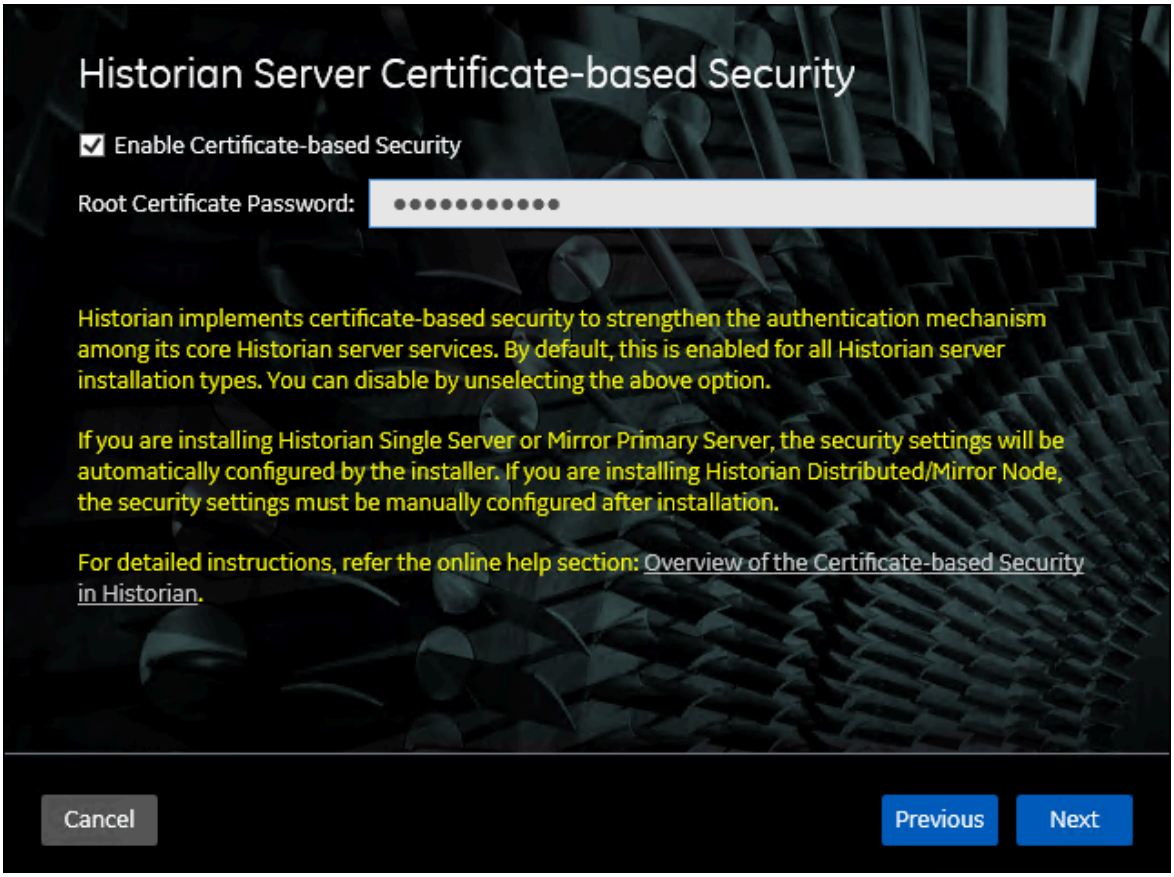



10. If you want the installer to create Historian security groups *(on page 256)*, select the corresponding check box, and then select **Next**.
    The **Choose the type of install you want to perform** page appears.




11. Select the type of the Historian server that you want to install, and then select **Next**.
    - **Historian Single Server**: This is for a stand-alone Historian system, which contains only one Historian server. This type of system is suitable for a small-scale Historian setup.
    - **Historian Mirror Primary Server**: This is for a horizontally scalable Historian system, which contains multiple Historian servers, all of which are connected to one another. Installing this server will allow you to add machines and distributed/mirror servers to this system.
    - **Historian Distributed/Mirror Node**: This is for a horizontally scalable Historian system. Installing this server will allow you to add this node to a primary server.

    The **Historian Server Certificate-based Security** page appears.




12. If you want to enable certificate-based security (MTLS-based security), leave the **Enable Certificate-based Security** check box selected, and then enter **Root Certificate Password**.

> **Note:**
> Ensure to use the same password to create MTLS (client) certificates. For more information on creating MTLS (client) certificates, refer to Generate MTLS certificate.

For more information on certificate-based security, refer to overview of the Certificate-based Security in Historian *(on page 206)*.

> **Warning:**
> If you do not select the **Enable Certificate-based Security** check box during the installation, you must generate the root certificates manually, as described in the Manually Install Certificates for Historian section. However, this is not recommended.

13. Select **Next**.
    The **You are ready to install** page appears.

14. Select **Install**.
    The installation begins.
15. After the installation, when you are asked to reboot your system, select **Yes**.
    The Historian server is installed on your machine in the following folder: `<installation drive>:\Program Files\Proficy\Proficy Historian\x64\Server`, and the following registry path is created: `HKEY_LOCAL_MACHINE\SOFTWARE\Intellution, Inc.\iHistorian\Services`.

    In addition, the following components are installed:
    - **The RemoteCollectorConfigurator utility:** A command-line tool, which allows you to manage collectors remotely. By default, it is located in the `C:\Program Files\GE Digital\NonWebCollectorInstantiationTool` folder. For instructions on using this utility, refer to About Installing and Managing Collectors Remotely.
    - **The Proficy Authentication Configuration tool:** A utility that allows you to specify the Proficy Authentication server details to match with the Proficy Authentication server used by Web-based Clients. By default, it is located in the `C:\Program Files\Proficy\Proficy Historian\x64\Server` folder. For instructions on using this tool, refer to Register with Configuration Hub *(on page 137)*.

## Install the Historian Server at a Command Prompt

- Set up the Historian environment *(on page 82)*.
- If you want to install web-based clients and manage the Historian node in Configuration Hub, you must provide the Configuration Hub and Proficy Authentication server details during installation. Therefore, ensure that you have already set up Proficy Authentication in Configuration Hub. For more information on setting up Proficy Authentication in Configuration Hub, refer to https://www.ge.com/digital/documentation/confighub/version2023/t_authentication_setup.html.
- If you are changing the role of a Historian server that was previously a distributed/mirror server to any other configuration (single-server or mirror primary server), you must first uninstall Historian *(on page 294)*.
- If you are installing a distributed/mirror server, use the same configuration, license key, installation drive, Proficy Authentication instance, and domain as the primary server.

This topic describes how to install single-server Historian at a command prompt. You can also install the Historian server using the installer *(on page 105)*.

After you install Historian at a command prompt, you can choose to generate a template XML file, which contains the installation parameters and the values that you have provided. You can use this XML file for

subsequent installations. Similarly, you can use a template XML file instead of providing command-line arguments.

1. Open Command Prompt, and navigate to the `<DVD drive>:\Historian` folder (for example, `E:\Historian`).
2. Run the following command:

   ```
   install.exe <argument>=<value> <flag> HistorianCmd=<installation type>
   ```

   For example,

   To install Historian Server without configuring web-based clients, run the following command:

   ```
   install.exe RootDrive=E:\ EnableCertificateSecurity=1 ServerCertPassPhrase="Password"
    CreateHistorianSecurityGroups=false -silent HistorianCmd=HistorianCore
   ```

   To install Historian Server and configure web-based clients, run the following command:

   ```
   install.exe RootDrive=E:\ ActiveUaaBaseUrl=https://histmachine123.test.com:443/uaa AdminClientId="admin"
    AdminClientSecret="secret" ConfighubServerName="histmachine123.test.com" ConfighubhttpsPort="5000"
    EnableCertificateSecurity=1 ServerCertPassPhrase="Password" CreateHistorianSecurityGroups=false -silent
    HistorianCmd=HistorianCore
   ```

   The following table provides a list of installation types that you can enter.

| Installation Type | Description |
|---|---|
| `StandAlone` | Enter this value if you want to install a stand-alone Historian system. |
| `HistorianCore` | Enter this value if you want to install a primary server in a horizontally scalable system. |
| `mirror` | Enter this value if you want to install a distributed/mirror server in a horizontally scalable system. |

   The following table provides a list of arguments that you must enter.

| Argument | Description |
|---|---|
| `RootDrive` | The drive letter where the Historian server binary files will be installed. |
| `DataPath` | The disk path where the Historian data files will be stored. |

| Argument | Description |
|---|---|
| `HistAdministrator-Password` | The password for the built-in administrator account. |
| `ActiveUaaBaseUrl` | The URL to connect to Proficy Authentication to allow Web-based Clients to access Historian. Only if you want to use Web-based Clients, this parameter is required for user authentication. Proficy Authentication provides identity-based security for applications and APIs. It supports open standards for authentication and authorization, including Oauth2.<br><br>Proficy Authentication details are required if you want to use Web-based Clients.<br><br>By default, the local hostname and the port number of 443 are considered. If the Proficy Authentication service is on the same machine on which you are installing the Historian server, you can accept the default value. If, however, the Proficy Authentication service is on a different machine or uses a different port number, replace those values in the URL as follows:<br>`https://<local host name>:<port number>/uaa`<br><br>where:<br>◦ *<Proficy Authentication server>* is the name of the machine on which Proficy Authentication is installed. If the machine uses a fully qualified domain name (FQDN), provide the FQDN.<br>◦ *<port number>* is the one that you have specified for the public https port in the **TCP Port Assignments** page during Web-based Clients installation.<br><br>**Note:**<br>You can install a Proficy Authentication service using Operations Hub or Historian Web-based Clients. You can provide the URL of an existing Proficy Authentication instance. Or, if a Proficy Authentication service is not available, you can install it during Web-based Clients installation. In that case, provide the server name where Proficy Authentication is installed. |
| `AdminClientId` | The client ID to connect to the Proficy Authentication service. |

| Argument | Description |
|---|---|
| `AdminClientSecret` | The password to connect to the Proficy Authentication service. |
| `ConfighubServerName` | The server name or the FQDN of the existing Configuration Hub server, as displayed in the address bar of the browser when you access Configuration Hub from the machine where Configuration Hub is installed. |
| `ConfighubhttpsPort` | The web server (https) port that you want to use for Configuration Hub. By default, it is 5000. |
| `CreateHistorianSecurityGroups` | Indicates whether you want the installer to create Historian security groups.<br><br>Using Historian security groups provides an added layer of control over access to your Historian system.<br><br>Enter true or false. If you enter true:<br>◦ You must add a Windows user to the appropriate group *(on page 278)* (for example, add an administrative user to the iH Security Admins group). Only then you can configure this server.<br>◦ If the Historian server and collectors are installed on the same machine, you can skip this step; by default, the machine name of the local Historian server is considered. If, however, they are installed on different machines, you must provide the credentials of the Windows user who can access the Historian server machine. In addition, if security groups are available, add the user to the appropriate group *(on page 278)* (for example, add an administrative user to the iH Security Admins group). Only then you can access Web-based Clients without LDAP.<br>For more information, refer to Implementing Historian Security *(on page 251)*. |
| `EnableCertificateSecurity` | Indicates whether you want the installer to enable certificate-based security (MTLS-based security).<br><br>◦ To enable certificate-based security, set this value to **1**.<br>◦ To disable certificate-based security, set this value to **0**.<br><br>For more information on the certificate-based security, refer to overview of the Certificate-based Security in Historian *(on page 206)*. |

| Argument | Description |
|---|---|
| `ServerCertPassPhrase` | If you enabled certificate-based security, enter the passphrase for the server certificate.<br><br>**Note:**<br>Ensure to use this passphrase as the password to create MTLS (client) certificates. For more information on creating MTLS (client) certificates, refer to Generate MTLS certificate. |

The following table provides a list of flags that you can use.

| Flag | Description |
|---|---|
| `[-q], [-quiet], [-s], [-silent]` | Use any of these flags for a silent installation. The installation will then happen in the background (without a UI). |
| `[-passive]` | Use this flag for a passive installation. The progress of the installation then appears on your screen. |
| `/t` | Use this flag to generate the template file, which will contain all the installation arguments and the values that you have provided for each of them. You can then use this file for subsequent installations.<br><br>By default, this file is named `Template_Historian.xml`, and it is placed in the `temp` folder, defined by the `%temp%` environment variable. If, however, you want to save the file in another folder as well, enter: `/t TemplateOutputDirectory=<path>` |
| `/c TemplateInputFile=<path>` | Use this flag to use a template file (instead of providing command-line arguments). However, if you do provide command-line arguments as well, they take precedence over the values in the template. |

The Historian server is installed on your machine in the following folder: <*installation drive*>:\Program Files\Proficy\Proficy Historian\x64\Server, and the following registry path is created: HKEY_LOCAL_MACHINE\SOFTWARE\Intellution, Inc.\iHistorian\Services.

In addition, the following components are installed:

- **The RemoteCollectorConfigurator utility:** A command-line tool, which allows you to manage collectors remotely. By default, it is located in the C:\Program Files\GE Digital\NonWebCollectorInstantiationTool folder. For instructions on using this utility, refer to Remote Collector Management.
- **The Proficy Authentication Configuration tool:** A utility that allows you to specify the Proficy Authentication server and Configuration Hub server details to match with the Proficy Authentication server used by Web-based Clients, and also register the Historian node with Proficy Authentication server and Configuration Hub server. By default, it is located in the C:\Program Files\Proficy\Proficy Historian\x64\Server folder. Alternatively, you can search for the Proficy Authentication Tool in the Windows search bar and open it. For instructions on using this tool, refer to Register with Configuration Hub *(on page 137)*.

While installing the Historian server, if you have allowed the installer to create Historian security groups, create a local Windows user with the format <Web-based Clients server name>.admin, and add the user to the ihSecurityAdmins group *(on page 278)*.

This user will log in to Web-based Clients.

Alternatively, you can create Proficy Authentication users in an external Proficy Authentication and map their security groups. For information, refer to About Proficy Authentication Groups *(on page 231)*.

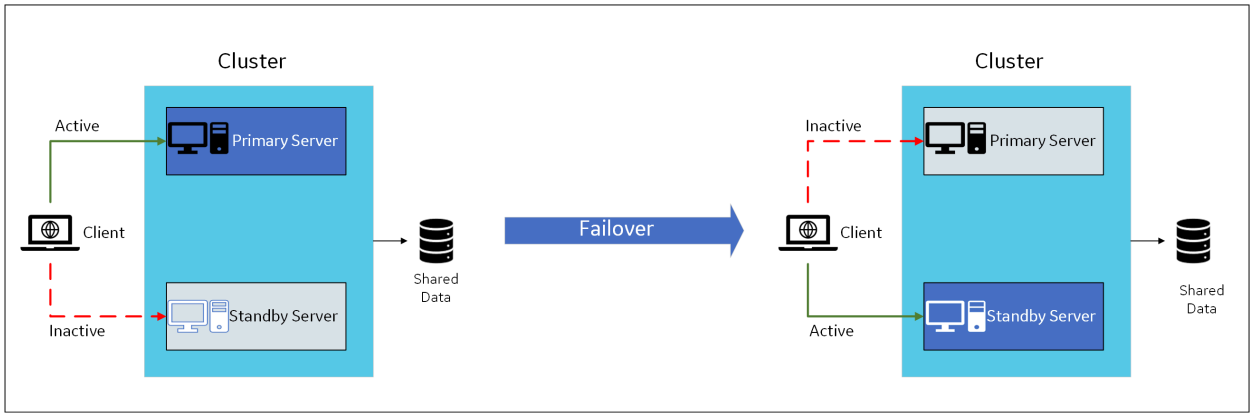
Depending on whether the Historian server will use local or domain security groups, select the appropriate option in Historian Administrator.

## Set Up High Availability of the Historian Server

Historian works with Microsoft Failover Cluster Manager to ensure high availability of the server.

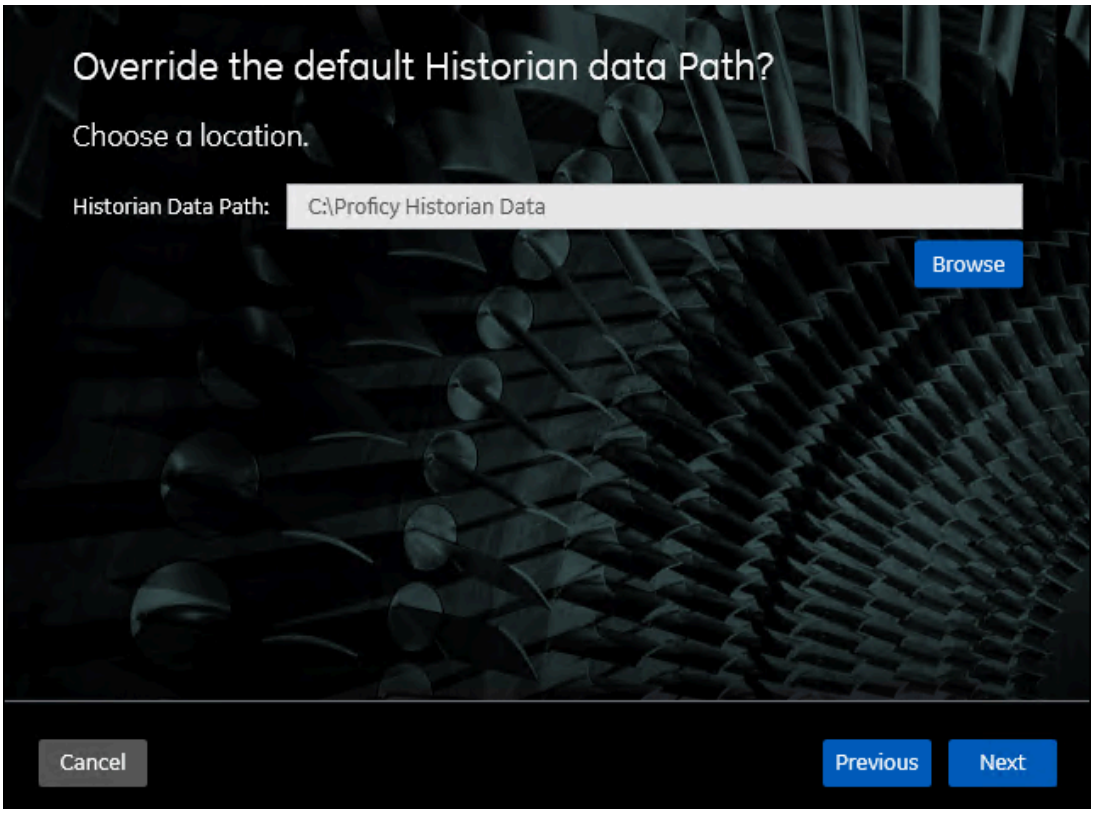
In a cluster environment, multiple servers are installed, which share the same data. Each of these servers is called a node. One of them acts as the primary server, while the others are standby servers. If the primary server is down, one of the standby servers is used. See the following figure for an example.

**Note:**
Cluster supports only domain security.




1. Ensure that your network is enabled for multicast traffic.
2. Create a shared drive on your network that all the servers in the cluster can access, and create a database folder in that drive.
3. On each node that you want to add to the cluster:

a. Install the Failover Clustering feature.
b. Install the Historian server *(on page 105)*. During the installation, in the **Historian Data Path** field, enter the path to the folder on the shared drive that you have created.




4. If you are upgrading the Historian server on a passive node, an error message may appear behind the installer screen, stating that the Archives directory is not created. You can ignore this message, or you can make the node active before upgrading the Historian server.

1. If you have upgraded the Historian server, on all the cluster nodes:
   a. Right-click the cluster, and then select **Properties > Resource Types**.
   b. If the user-defined resource types are not available, select **Add**.
   c. Select `<Installation folder of the Historian server>/x64/Server/Historian.dll` as the resource DLL path with Historian and AlarmArchiver as both the resource type names and display names.

   The Historian servers are now part of the cluster, thus achieving high availability. The remaining steps are required only for first-time installation of the Historian server (not upgrade).
2. Access the primary node of the cluster.
3. Create a failover cluster.
4. Add a storage to the failover cluster.

5. Add user-defined resource types, Historian and AlarmArchiver, to the cluster. These resources are created when you install the Historian server.
    a. Right-click the cluster, and then select **Properties > Resource Types**.
    b. If the user-defined resource types are not available, select **Add**.
    c. Select `<Installation folder of the Historian server>/x64/Server/Historian.dll` as the resource DLL path with Historian and AlarmArchiver as both the resource type names and display names.
6. Select **Roles > Create Empty Role**.
   A role is created.
7. Add the Historian application to the cluster:
    a. Select the role.
    b. In the **Actions** section, select **Add Resource > Historian**.
    c. Follow the on-screen instructions to add the new Historian resource to the role.
8. Add a client access point to the role:
    a. Select the role.
    b. In the **Actions** section, select **Add Resource > Client Access Point**.
    c. Follow the on-screen instructions to add a client access point to the role.
9. Add a storage to the role:
    a. Select the role.
    b. In the **Actions** section, select **Add Storage**.
    c. Follow the on-screen instructions to add the storage that you have created in step 3. You can use a storage only once.
10. Add the following dependencies to the Historian resource:

    a. Double-click the Historian resource.
       The **Historian Properties** window appears.

    b. Select **Dependencies**.

    c. Select **Insert**, and add the following dependencies using the AND operation.
        - Client access point
        - IP address
        - Storage

11. Add the Alarm Archiver resource to the cluster:

    a. Select the role.

    b. In the **Actions** section, select **Add Resource > Alarm Archiver**.

c. Follow the on-screen instructions to add the alarm archiver resource to the role.

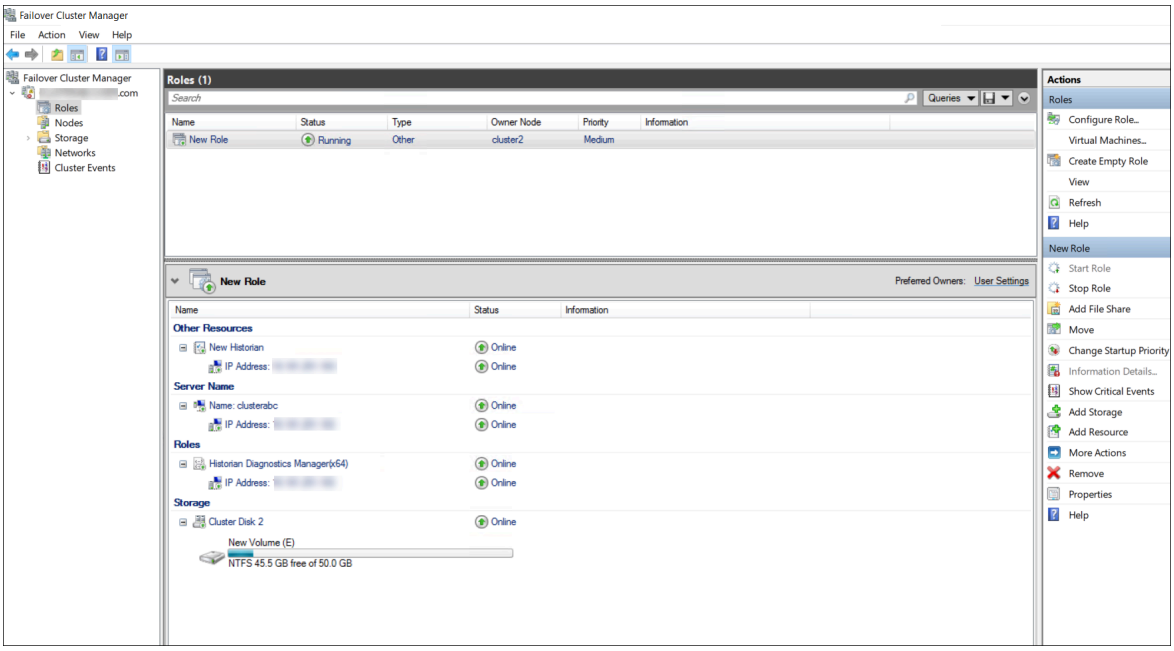
d. Double-click the Alarm Archiver resource.
The **Alarm Archiver Properties** window appears.

e. Select **Dependencies**.

f. Select **Insert**, and then add the Historian service as a dependency. You do not need to add the cluster disk and IP address as dependencies.

12. Add generic services.
    a. In the **Summary** section of the Historian role, right-click the Client Manager resource, and then select **Properties > Dependencies**.
    b. Add the IP address as a dependency, and then select **Apply**.
    c. Select **General**, select the **Use Network Name for Computer Name** check box, and then select **OK**.
13. Repeat the previous step for the following services:
    - Configuration Manager
    - Diagnostics Manager
    - Historian Embedded PostgreSQL Database
    - Historian Embedded Tomcat Container
    - Historian Indexing Service

The services appear under the Historian role.



14. Select the Historian role, and then in the **Actions** section, select **Start Role**.
    The Historian servers are now part of the cluster, thus achieving high availability.

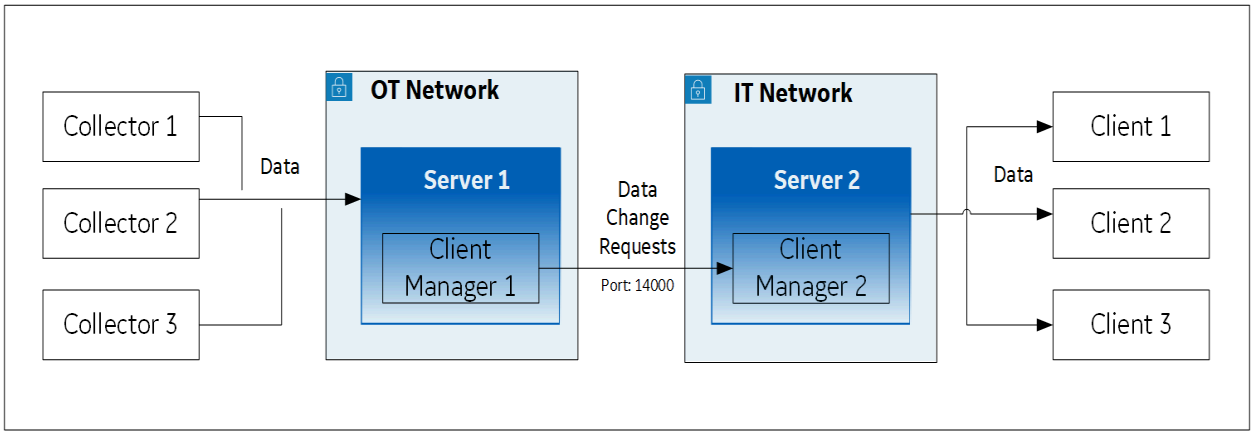
## Set Up a Mirror of Mirror

1. Install Historian server *(on page 105)* on each machine that you want to use in the mirror of mirror setup.
2. Set up Configuration Hub *(on page 305)* on each machine that you want to use in the mirror of mirror setup.
3. Add a system *(on page 427)*. The server that you specify while adding the system serves as the primary server for the system.
4. Create data stores *(on page 370)* in the primary server in the public/IT network with the same name as the data stores in the primary server in your organization network.

You can set up a mirror of the Historian server in a network different from that of your organization. When you do so, any tag/data update requests to the Historian server can be routed to the public and IT network instead of your organization's network.

> **Note:**
> For the Historian Enterprise Mirror Architecture, only "Time based" archives are supported.

**Single-Node Setup:** The following image shows two networks - OT and IT - with a Historian server installed in each network. These networks communicate using port 14000.




In this setup:

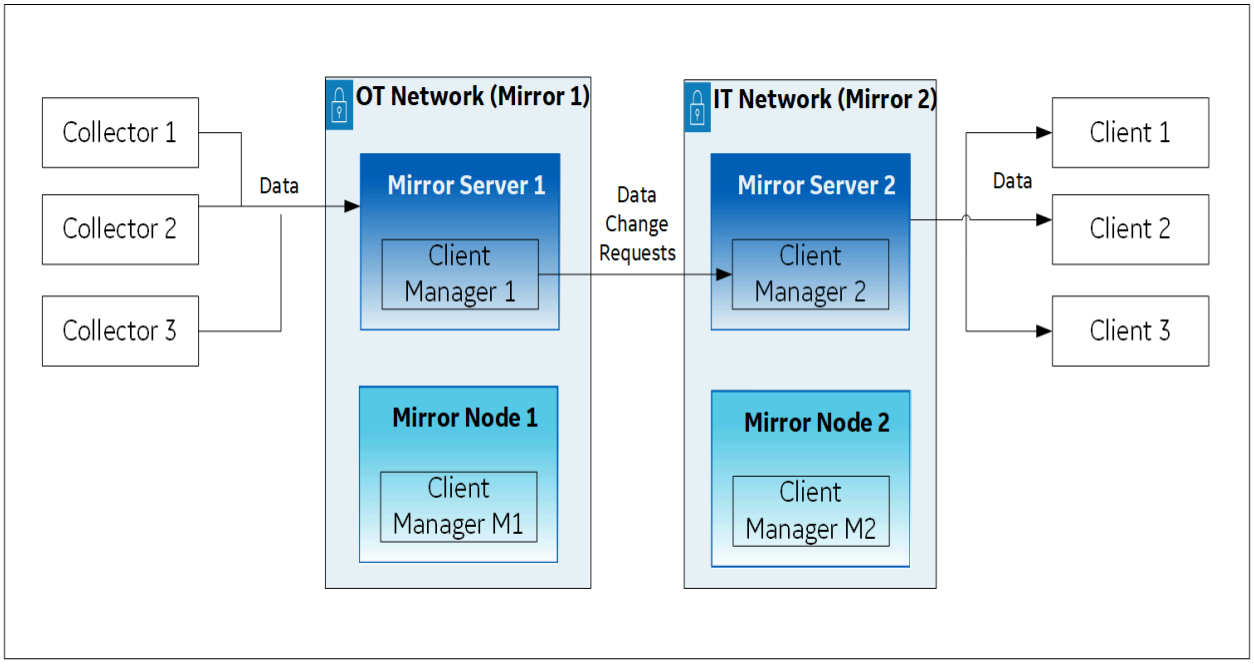
1. Server 1 is the primary server in the OT network; it stores data from collectors.
2. Server 2 is the primary server in the IT network; it is connected to clients.

3. When a tag/data is created, updated, or deleted, Client Manager 1 communicates the same with Client Manager 2 (installed with Server 2 in the IT network).
4. The change in the tag/data is replicated in Server 2 (that is, data is created, updated, or deleted accordingly).
5. The latest data is retrieved from Server 2 using the clients.

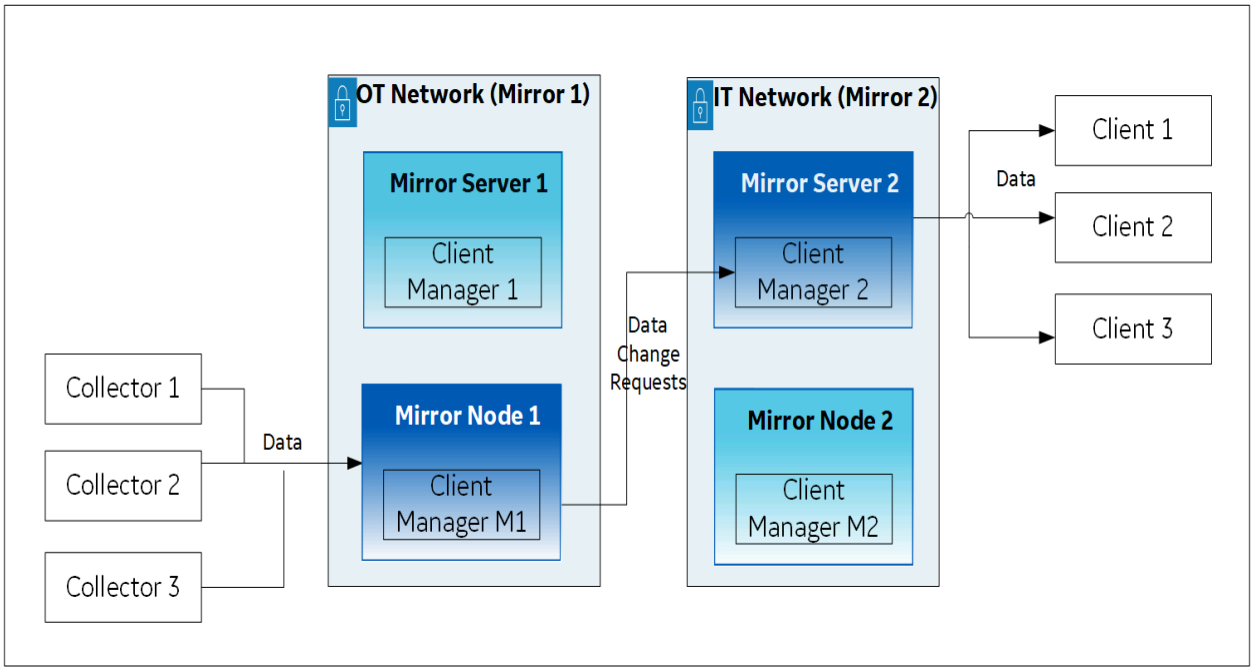
**Mirror Setup:** The following image shows two mirrors:

- Mirror 1 includes Mirror Server 1 and Mirror Node 1, which is a backup/standby node for Mirror Server 1; both these machines are in the OT network.
- Mirror 2 includes Mirror Server 2 and Mirror Node 2, which is a backup/standby node for Mirror Server 2; both these machines are in the IT network.

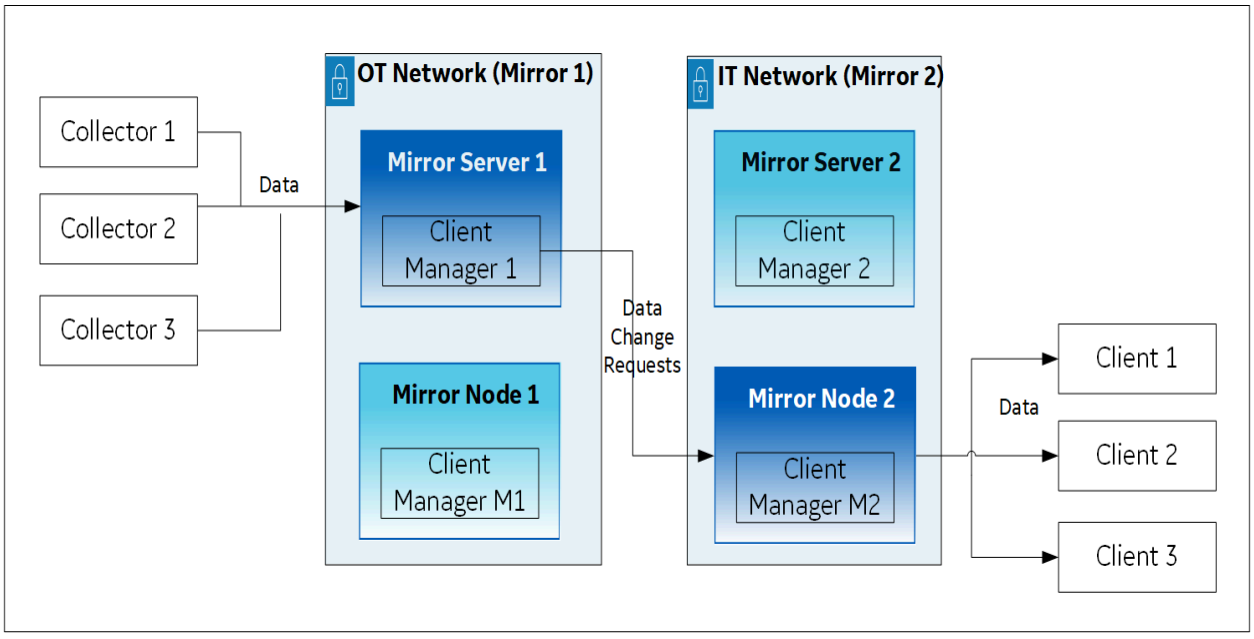
Client Manager 1 in Mirror Server 1 communicates with Client Manager 2 in Mirror Server 2.



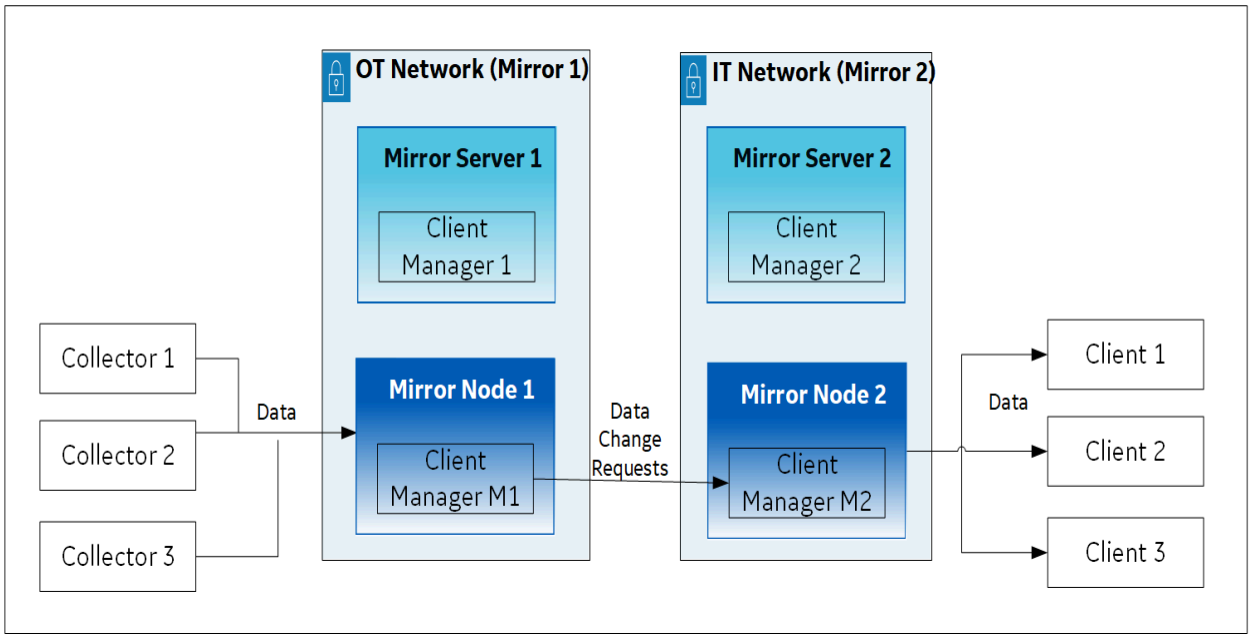

If Mirror Server 1 goes down, Client Manager M1 in Mirror Node 1 communicates with Client Manager 2 in Mirror Server 2.



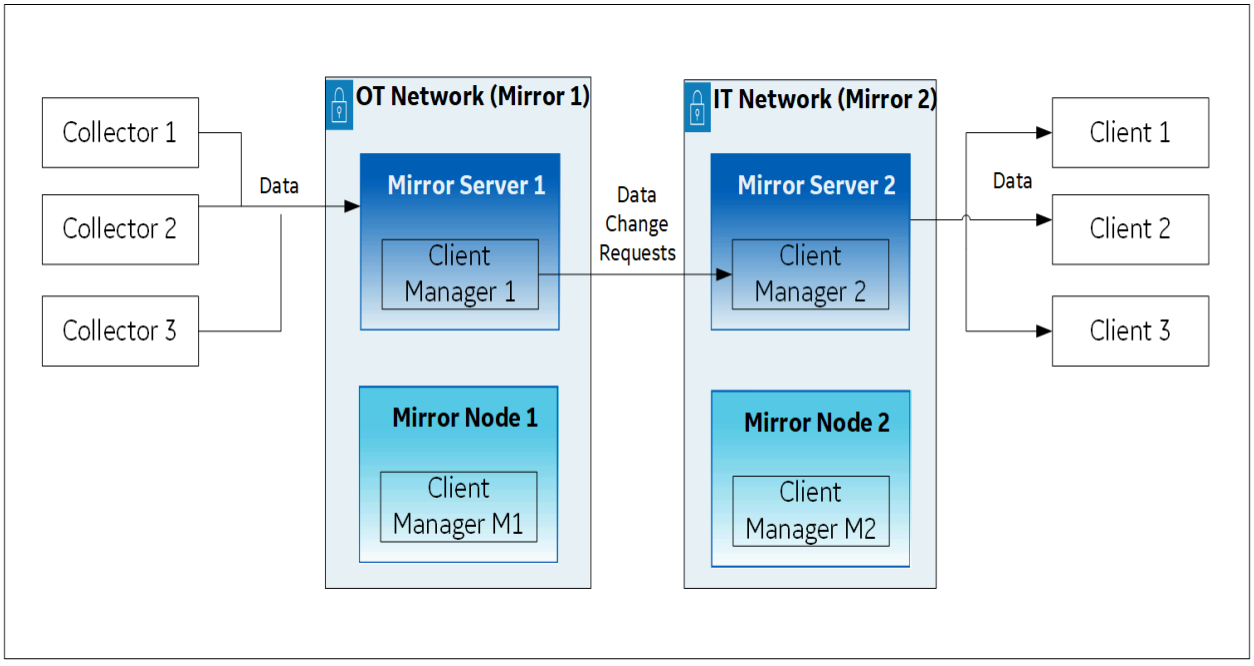

Similarly, if Mirror Server 2 goes down, Client Manager 1 in Mirror Server 1 communicates with Client Manager M2 in Mirror Node 2.




If both Mirror Server 1 and Mirror Server 2 are down, Client Managers M1 and M2 communicate with each other.




If Mirror Server 1 and/or Mirror Server 2 are available, the connection is re-established using these primary servers.



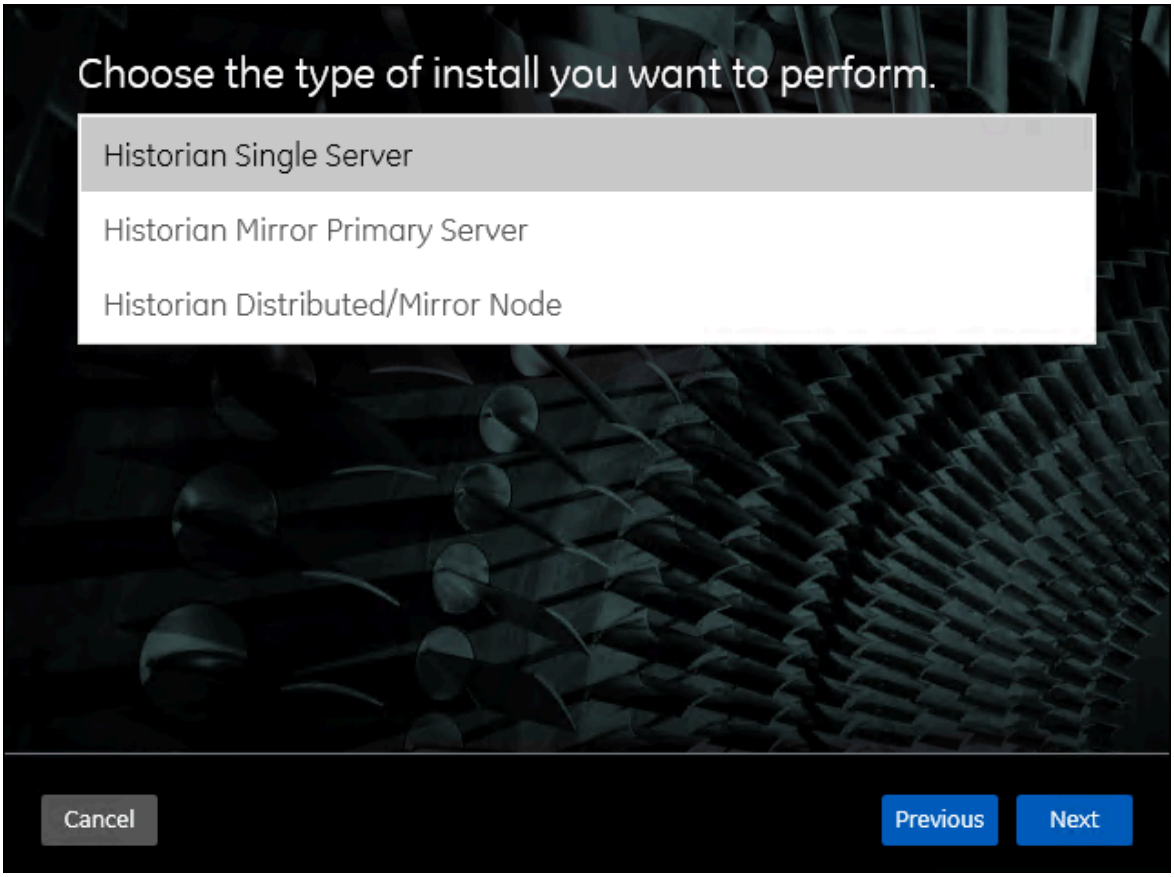

Thus, you can choose to always retrieve data from either Mirror Server 2 or Mirror Node 2. In addition, the store-and-forward functionality is available (in case Client Managers are not yet connected).

This topic describes how to set up a mirror of mirror for the configuration described in the preceding example. It includes the following high-level steps:

1. Installing the Historian server on all the machines
2. Setting up mirror 1
3. Setting up mirror 2
4. Setting up a mirror of mirror

**Installing the Historian server**

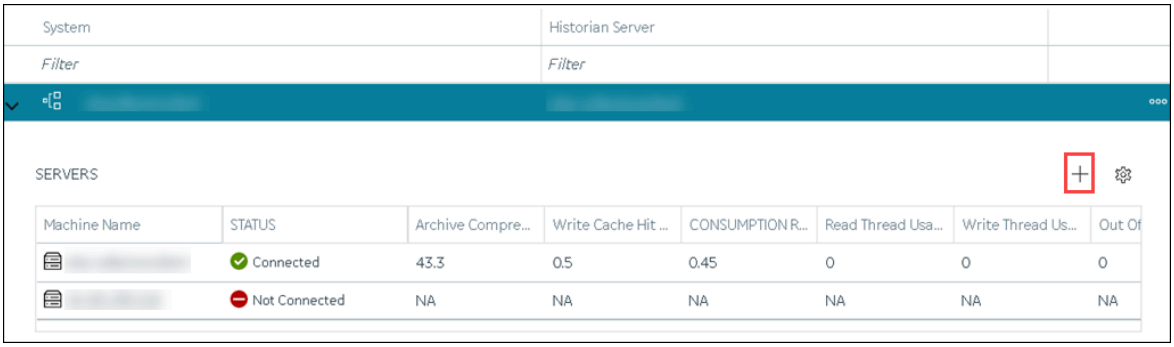
1. On the machines designated as the mirror primary servers (Mirror Server 1 and Mirror Server 2 in the example), install the Historian server *(on page 105)*. During the installation, select **Historian Mirror Primary Server** on the **Choose the type of install you want to perform** page.



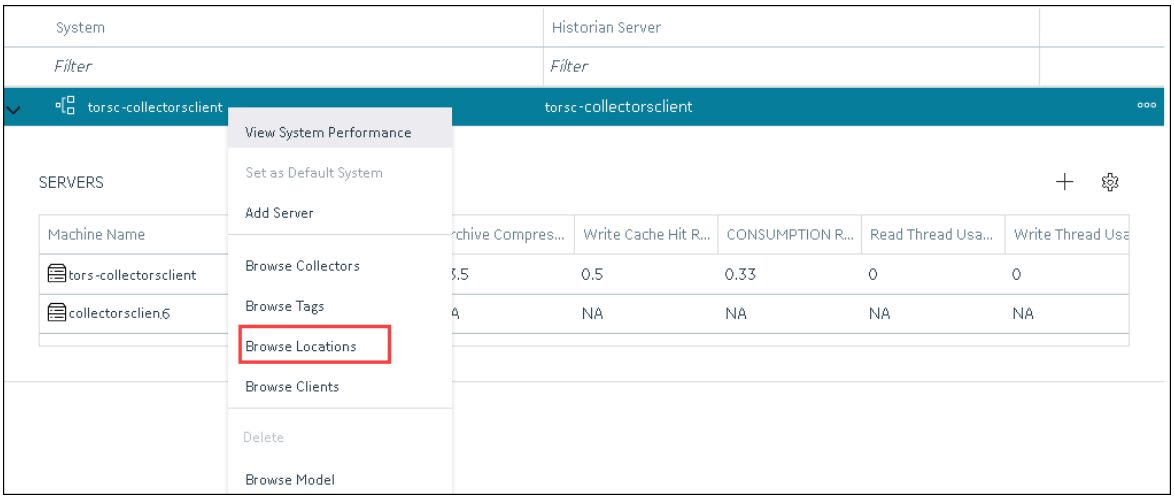

2. On the machines designated as mirror nodes (Mirror Node 1 and Mirror Node 2 in the example), install the Historian server *(on page 105)*. During the installation, select **Historian Distributed/ Mirror Node** on the **Choose the type of install you want to perform** page.

**Set up Mirror 1:**

3. On the mirror primary server in your organization's network (Mirror Server 1 in the example), access Configuration Hub *(on page 336)*.
4. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Systems**.
   A list of systems appears in the main section.
5. Expand Mirror Server 1.
   A list of servers in the system appears.
6. Select +.




   The **Add Server Machine: <system name>** window appears.
7. Enter the host name or IP address of the mirror node in your organization's network (Mirror Node 1 in the example), and then select **Add**.
   The distributed server is added to the system. A distributed location is added in the server.
8. Right-click Mirror Node 1, and then select **Browse Locations**.




   A list of distributed locations in the system appears.
9. Select **Mirror Locations**.
   A list of mirror locations in the system appears.
10. In the upper-right corner of the main section, select +.

The **Add Mirror Location** window appears.

11. Provide values as described in the following table.

| Field | Description |
|---|---|
| **MIRROR LOCATION NAME** | Enter a name for the mirror location. A value is required and must be unique for the system. |
| **SERVER MACHINES** | Select the servers that you want to add to the mirror group (Mirror Server 1 and Mirror Node 1 in this example). This box contains a list of all the servers in the system. You must add minimum two servers to a mirror location. |

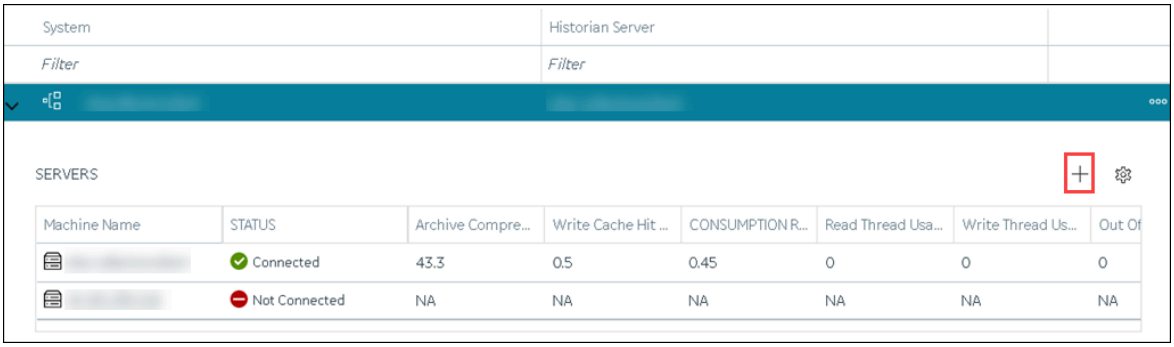
12. Select **Add**.
    Mirror Node 1 is created.
13. Right-click the system name, and then select **Add Data Store**.
    The **Add Data Store: Mirror Node 1** window appears.
14. Enter values as described in the following table.

| Field | Description |
|---|---|
| **DATA STORE NAME** | Enter a unique name for the data store. A value is required. You can use all alphanumeric characters and special characters except / \ * ? < > \|<br><br>You must provide the same name for the mirror setup in the IT network (mirror 2 in the example). |
| **DESCRIPTION** | Enter a description for the data store. |
| **Set as default data store for the system** | Select this check box if you want to set this data store as the default one. A default data store is the one that is considered if you do not specify a data store while adding a tag. You can set only one data store as default. |

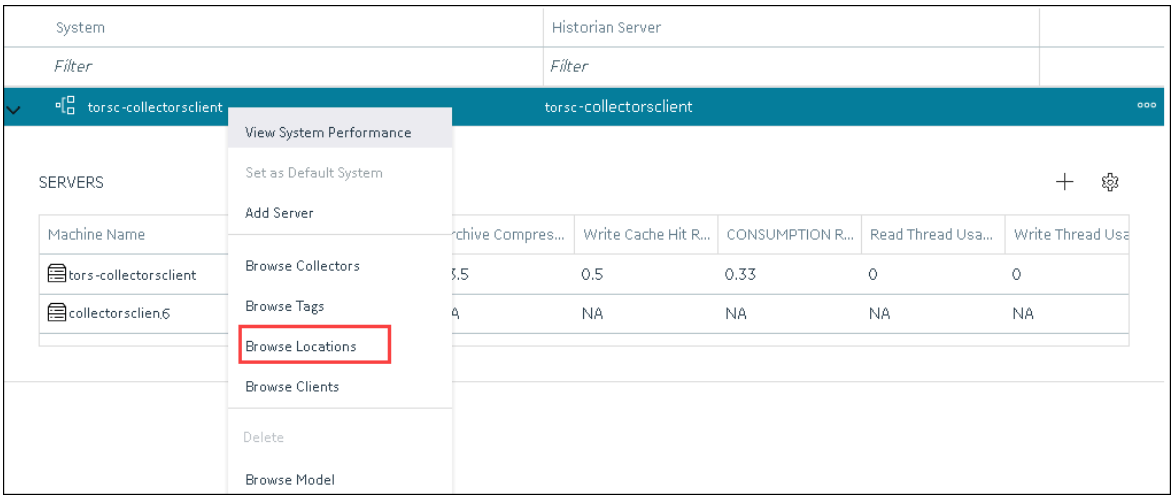
15. Select **Add**.
    Mirror 1 is configured.

**Set up Mirror 2:**

16. On the mirror primary server in the IT network (Mirror Server 2 in the example), access Configuration Hub *(on page 336)*.
17. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Systems**.
    A list of systems appears in the main section.
18. Expand Mirror Server 2.
    A list of servers in the system appears.
19. Select +.




    The **Add Server Machine: <system name>** window appears.
20. Enter the host name or IP address of the mirror node in your organization's network (Mirror Node 2 in the example), and then select **Add**.
    The distributed server is added to the system. A distributed location is added in the server.
21. Right-click the system name, and then select **Browse Locations**.




    A list of distributed locations in the system appears.
22. Select **Mirror Locations**.
    A list of mirror locations in the system appears.
23. In the upper-right corner of the main section, select +.

The **Add Mirror Location** window appears.

24. Provide values as described in the following table.

| Field | Description |
|---|---|
| **MIRROR LOCATION NAME** | Enter a name for the mirror location. A value is required and must be unique for the system. |
| **SERVER MACHINES** | Select the servers that you want to add to the mirror group (Mirror Server 2 and Mirror Node 2 in this example). This box contains a list of all the servers in the system. You must add minimum two servers to a mirror location. |

25. Select **Add**.
    Mirror Node 2 is created.
26. Right-click Mirror Node 2, and then select **Add Data Store**.
    The **Add Data Store: Mirror Node 2** window appears.
27. Enter values as described in the following table.

| Field | Description |
|---|---|
| **DATA STORE NAME** | Provide the same name that you provided while setting up mirror 1. |
| **DESCRIPTION** | Enter a description for the data store. |
| **Set as default data store for the system** | Select this check box if you want to set this data store as the default one. A default data store is the one that is considered if you do not specify a data store while adding a tag. You can set only one data store as default. |

28. Select **Add**.
    Mirror 2 is configured.

**Set up Mirror of Mirror:**

29. Access Configuration Hub in the primary server in the OT network (Mirror Server 2).
30. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Systems**.
    A list of systems appears in the main section.
31. Expand Mirror Server 1.
    A list of servers in the system appears.
32. Select +.

The **Add Server Machine: <system name>** window appears.

33. Enter the host name or IP address of the mirror server in the IT network (Mirror Server 2 in the example), select the **Set as Mirror of Mirror** check box, and then select **Add**.
    The distributed server is added to the system. A distributed location is added in the server.
34. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Systems**.
    A list of systems appears in the main section.
35. Expand Mirror Server 1.
    A list of servers in the system appears. In the example, Mirror Server 1, Mirror Node 1, and Mirror Server 2 appear.

A mirror of mirror is configured with one primary node and one mirror node each in the OT and IT networks. As needed, you can add more mirror nodes in each network.

## Upgrade the Historian Server

If any Historian applications or components are open, close them before upgrading the Historian server.

If you are upgrading from either Historian 6.0 Enterprise or previous releases of Historian 7.2 (including any of the service packs), both Client Manager and Configuration Manager services will be removed. However, this will have no impact on your data or use of Historian unless you intend to use a distributed system.

Install the Historian server *(on page 104)*.
The Historian server is upgraded to the latest version.

## About Historian Log Files

Historian creates the following types of log files:

- **The .IHA files:** These files contain data about archives. They are created by the Historian server after data collection begins. By default, these files are located in the `C:\Historian Data \Archives` folder.
- **The .IHC files:** These files contain data about Historian configuration. They are created by the Historian server. By default, these files are located in the `C:\Historian Data\Archives` folder.

  There are two types of .IHC files:
  - `*CentralConfig.ihc`: This is the master configuration file used by Configuration Manager.
  - `*config.ihc`: This is used by the data archiver and is generated from `*CentralConfig.ihc`. This is to maintain consistency between Historian versions.

- **The .LOG files:** These files contain logging data (such as events, warnings, and errors). They are created by the archiver and the collectors. By default, they are located in the `C:\Historian Data\LogFiles` folder.
- **The .SHW files:** These files contain configuration data. They are created by the archiver and the collectors. By default, they are located in the `C:\Historian Data\LogFiles` folder.

## FAQs on Installing Historian in a Distributed Environment

- What happens when a node that was down is up and running? Is the data written to one node synchronized with another?

  There is no automatic synchronization. If a node is down, the information to be written is buffered by Client Manager, or if Client Manager is down, it is buffered by the collector. When the node is up and running, data is written to the data archiver.

- There is only one Configuration Manager on the primary node. Can I still configure if the primary node is down?

  No. If the Configuration Manager is not available, you can read the configuration (because this information is stored in the collectors), but you cannot edit or modify the configuration.

- Is Configuration Manager a single point of failure?

  Yes. If the primary node is down, you cannot edit the configuration. However, since information about the configuration is stored in the registry of each client, the information is still available as read-and-write-only when the primary node is down.

  If the Configuration Manager service is down, you cannot query tags and data in a horizontally scalable system. However, you can query tags and data in the following scenarios:
  - The Historian system contains only one node, which is installed as the primary mirror Historian server.
  - The Historian system contains only one mirror location, and there are no data stores in the distributed locations.

- What happens if a node crashes in the middle of a read/write request?

  The operation continues to function in the same way as in prior releases. Client Manager holds a copy of the message request; therefore, once the node is up and running, the write operation is resumed. However, read requests will fail.

- The server where my primary node is installed is down. What is the expected behavior?

The Web Admin console and Trend Client will not be available; you can access tag configuration using Historian Administrator, but you will not be able to edit tag configuration. All other existing clients continue to work as expected, with the ability to collect and store data, search for tags, trend and report on tag information. A new user connection with default Historian server set to primary must connect to the primary node to get information about all the nodes before it gains the ability to automatically failover when the primary node is down.

- Client Manager on the primary node is down, but the server is running. What is the expected behavior?

  The Web Admin console and Trend Client, along with all other existing clients, will work as expected with the ability to do configuration changes, collect and store data, search for tags, trend and report on tag information. A new user connection with default Historian server set to primary must connect to the primary node to get information about all the mirrors before it gains the ability to automatically failover when the primary node is down.

- One of the data archivers is down, but at least one is active. What is the expected behavior?

  The system should continue to function as designed. The Web Admin console, Trend Client, Historian Administrator, as well as other clients continue to work as expected, with the ability to collect and store data, search for tags, trend and report on tag information.

- If there are calculated tags in a distributed environment, are the calculations done on all nodes?

  Yes.

- Are Historian tag statistics created independently? Can they be different between different nodes?

  Yes. These are queries, not tags, to a specific data archiver. As writes are independent, one data archiver may be ahead of another, so the statistics may vary slightly.

- How do we ensure that the data is consistent across data archivers?

  Tag information is consistent; there is only one tag. The time stamp and value are sent to all the nodes.

- Are there specific log files that I should be looking for to help diagnose issues with failure modes?

  No changes were made to the logs for data archiver; however, there are new log files for Client Manager and Configuration Manager.

- There are now two `*.ihc` files: `*config.ihc` and `*CentralConfig.ihc`. What is the difference between the two?

`*CentralConfig.ihc` is the master configuration file used by Configuration Manager. The `*config.ihc` file is used by the data archiver and is generated from `*CentralConfig.ihc`. This is to maintain consistency between Historian versions.

- With mirroring, is Microsoft Cluster Server still supported? What is the recommended approach?

  Mirroring is offered as a substitute to Microsoft Cluster Server. Mirroring provides high availability for locations. Microsoft Cluster Server has not been tested or validated to date with Historian systems.

- Should I install SQL server in a distributed environment?

  No. SQL server is only required for the Historian Alarms and Events database.

- How does mirroring work with Historian Alarms and Events SQL logging?

  There is still an alarm archiver; it does not go through Client Manager, so it connects with SQL as earlier.

- How does Historian Alarms and Events fit with their synching?

  There is one database, so everyone talks to the same SQL database. You can cluster the database, but that is separate from mirroring.

- How does mirroring work in a workgroup environment or non-domain?

  Mirroring is not supported in Workgroups.

- Are there any issues when making changes in Historian Administrator and a mirrored system?

  You must establish a mirror using the Historian Configuration Hub, but compatibility with all APIs has been maintained. Therefore, you can make tag changes in either the Web Admin or the VB Windows Admin, and those changes will show up in both Admins.

- Are there any plans to add more than three mirrors?

  No performance benefits have been seen beyond three mirrors.

- Do redundant collectors behave differently in a distributed environment?

  No.

- Are there any conflicts when using port 14000 for Historian to Historian communications (for example, Site to Corporate)?

  No. Client Manager is now on port 14000, data archiver is on port 14001, and Configuration Manager is on port 14002.

- If load balancing uses round robin reads, does the cache need to be loaded separately on both machines, and will it decrease performance?

  It does require more memory. Client Manager decides where to send the messages, and it knows about the configuration. There is some overhead, but it is overcome by having multiple data archivers to service multiple requests. That is why there is a 1.5X improvement with two mirrors, instead of 2X.

- Are there any additional considerations if a distributed system is used with other GE applications such as Workflow or Plant Applications?

  No. It still looks like one Historian to other applications.

- Is the store-and-forward feature also used in a distributed environment?

  Yes. This is a feature of the collector. Once the message is sent to Client Manager, it is done. If the Client Manager cannot reach one of the data archivers, it buffers the request until the archiver is available.

- In a distributed environment, do existing queries and reports work the same?

  Yes. Everything works the same as it did before. It sees it as a single Historian and communicates over the same ports through the same API.

- Does the Historian OPC Classic HDA server still work in a distributed environment?

  Yes.

- If data is being written to two data archivers, does this double the traffic from the collector?

  No. It does not double traffic from the collector; it sends a single message to Client Manager. The traffic is doubled between the Client Manager and the two data archivers.

## Register with Configuration Hub

Ensure the following:

- Installed Web-based Clients *(on page 154)*, specifying the details of the new Proficy Authentication and Configuration Hub servers.

The Historian server and Web-based Clients must always point to the same Proficy Authentication server and Configuration Hub server. Only then you can access Web-based Clients (such as Configuration Hub, Trend Client, the Web Admin console, and REST APIs), and also view Historian license information on Configuration Hub. Therefore, you can use the Proficy Authentication Tool in the following cases:

- If you changed the Proficy Authentication server used by Web-based Clients, the Historian server must point to the same Proficy Authentication and Configuration Hub servers.
- If you did not select the **Use existing Proficy Authentication and Configuration Hub Instance** during the Historian Server, and now you want to register with the Proficy Authentication and Configuration Hub servers to use the Historian Web-based clients and view Historian license information on Configuration Hub.

This topic describes how to register the Historian Server with the Proficy Authentication server and Configuration Hub server without the need to reinstall it.

1. Access the `UAAConfiguration.exe` file. By default, it is located at `<Install Drive>:\Program Files\Proficy\Proficy Historian\x64\Server`.
   Alternatively, you can search for the Proficy Authentication Tool in the Windows search bar and open it.
   The **Proficy Authentication Configuration tool** window appears, displaying the Proficy Authentication server details that you specified while installing the Historian server. Also, it allows you to specify the Configuration Hub server details.
2. In **Proficy Authentication Server Details**, provide the following details as described in the table below.

| Field | Description |
|---|---|
| **Proficy Authentication server name** | Enter the name of the machine on which the Proficy Authentication server is installed. If the machine uses a fully qualified domain name (FQDN), provide the FQDN. By default, the local hostname is considered. |
| **Public https port** | Enter the port number used by the Proficy Authentication service. The default value is 443. Ensure that this port number matches the one on the **TCP Port Assignments** page during Web-based Clients installation. |
| **Admin Client Id** | The client ID to connect to the Proficy Authentication service. |
| **Admin Client Secret** | The password to connect to the Proficy Authentication service. |
| **Proficy Authentication Test Connection** | Option to test the status of the connection with external Proficy Authentication server. |

3. In **Configuration Hub Server Details**, provide the following details as described in the table below.

**Table 10.**

| Field | Description |
|---|---|
| **Configuration Hub server name** | The server name or the FQDN of the existing Configuration Hub server, as displayed in the address bar of the browser when you access Configuration Hub from the machine where Configuration Hub is installed. |
| **Configuration Hub Public https port** | The web server (https) port that you want to use for Configuration Hub. By default, it is 5000. |
| **Configuration Hub Test Connection** | Option to test the status of the connection with an existing Configuration Hub server. |

4. Select **Configure**.
   The Proficy Authentication server and Configuration Hub details are updated for the Historian server. The changes are reflected as soon as you refresh the browser in which you have opened any of the Web-based Clients components (such as Configuration Hub, the Web Admin console, Trend Client).

If testing the connection fails, try these steps:

- Verify that you can ping the Proficy Authentication server. If you cannot ping the Proficy Authentication server, add the IP address and the server name in the `hosts` file located in `C:\Windows\System32\drivers\etc`.
- Ensure that the following services are running on the Proficy Authentication server machine:
  - Proficy Operations Hub Httpd Reverse Proxy
  - Proficy Authentication Tomcat Web Server
  - All the GE Proficy Configuration Hub services
- Verify that the Proficy Authentication server details provided during Web-based Clients installation match the ones you have specified in the Proficy Authentication Configuration tool.

## Historian Installation Limitations

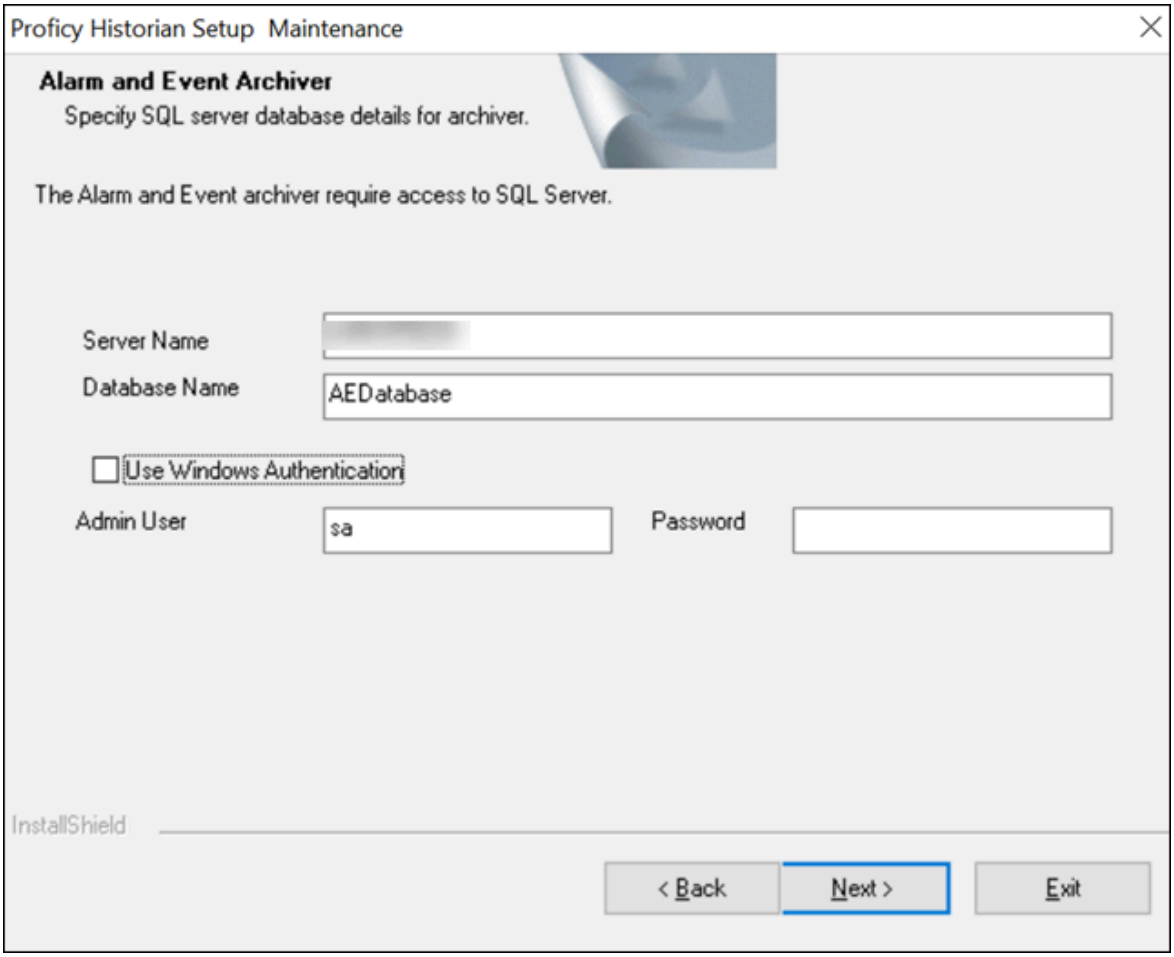
- You cannot close your current archive with a Historian Mirror Primary Server and Historian Mirror Node installation. This is because closing the current archive introduces archive synchronization risks in a mirrored environment. The restriction is enforced on all Historians, even those not using mirroring.
- You cannot use size-based archives with a Historian Mirror Primary Server and Historian Mirror Node installation. This is because having archives of different sizes introduces archive synchronization risks in a mirrored environment. The restriction is enforced on all Historians, even those not using mirroring.

## Alarms and Events

## Install Alarms and Events

- You must install Historian Alarms and Events on the same machine as the data archiver.
- If you have chosen to connect Historian to a remote SQL server, the following conditions must be satisfied:
    - The Historian Alarm Archiver service must be run on a user account that has privileges to log in to the SQL server using Windows authentication.
    - The default backup path, which you can set on the Archive page, must be a shared directory that is accessible to both the Historian Data Archiver and the remote SQL server. It is recommended that this shared directory be placed on the same computer as the Historian Data Archiver service.

1. Run the `InstallLauncher.exe` file.
2. Select **Install Alarms and Events**.
   The **Alarms and Events Archiver** page appears.




3. If needed, change the values in the **Server Name** and **Database Name** fields to provide the name of the SQL server and the name of the database where the alarms and events data is archived.
4. If you want to use the SQL server credentials, clear the **Use Windows Authentication** check box, and then enter the SQL server login credentials in the **Admin User** and **Password** fields. If you want to use Windows authentication, select the **Use Windows Authentication** check box. When you do so, the **Admin User** and **Password** fields are disabled.
5. Select **Next**.
6. When prompted to restart your system, select **Yes**.
   Historian Alarms and Events is installed in the following folder: `<installation drive>:\Program Files\Proficy\Proficy Historian\x86\Server`, and the following registry path is created: `HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\ Intellution, Inc.\iHistorian\Services\AlarmArchiver`

7. To verify that the Alarms service has started, access the **Services** window, and check the status of the Historian Alarm Archiver service.
   If the **Startup Type** field is set to **Automatic**, the service is started automatically when the system is started or restarted.

## Upgrade Alarms and Events

- If Alarms and Events were installed prior to Historian 7.0, you must install them separately.
- If you want to upgrade from Historian 4.5, since the database schema are different, if you select the same database name that is pre-populated by default, you will get an error message: `Later or Higher version of Alarms and Events database is already installed. Hence, you cannot proceed further.` You need to enter a different database name and then proceed with the upgrade.

Install Alarms and Events *(on page 140)*.
Alarms and Events are upgraded to the latest version.

## About Installing Historian Data Collectors

When you install collectors, the required binaries are downloaded. In addition, if iFIX/CIMPLICITY are installed on the same machine as the collectors, instances of the following collectors are created:

- The iFIX collector
- The iFIX Alarms & Events collector
- The OPC Classic Data Access collector for CIMPLICITY
- The OPC Classic Alarms and Events collector for CIMPLICITY

If an iFIX collector instance created in version 9.0 exists, after you upgrade collectors, another instance of the iFIX collector is created. Because of this, the Remote Collector Manager (RCM) will not work correctly. Therefore, if you want to use RCM, you must delete one of the instances. If needed, you can manually create another instance of the iFIX collector using Configuration Hub or the RemoteCollectorConfigurator utility. This is applicable to the iFIX Alarms and Events collector as well.

**Note:**
If you want to upgrade collectors earlier than version 7.1, additional registries that you create manually are deleted. Therefore, we recommend that you back them up, uninstall the collectors, and then install the latest version.

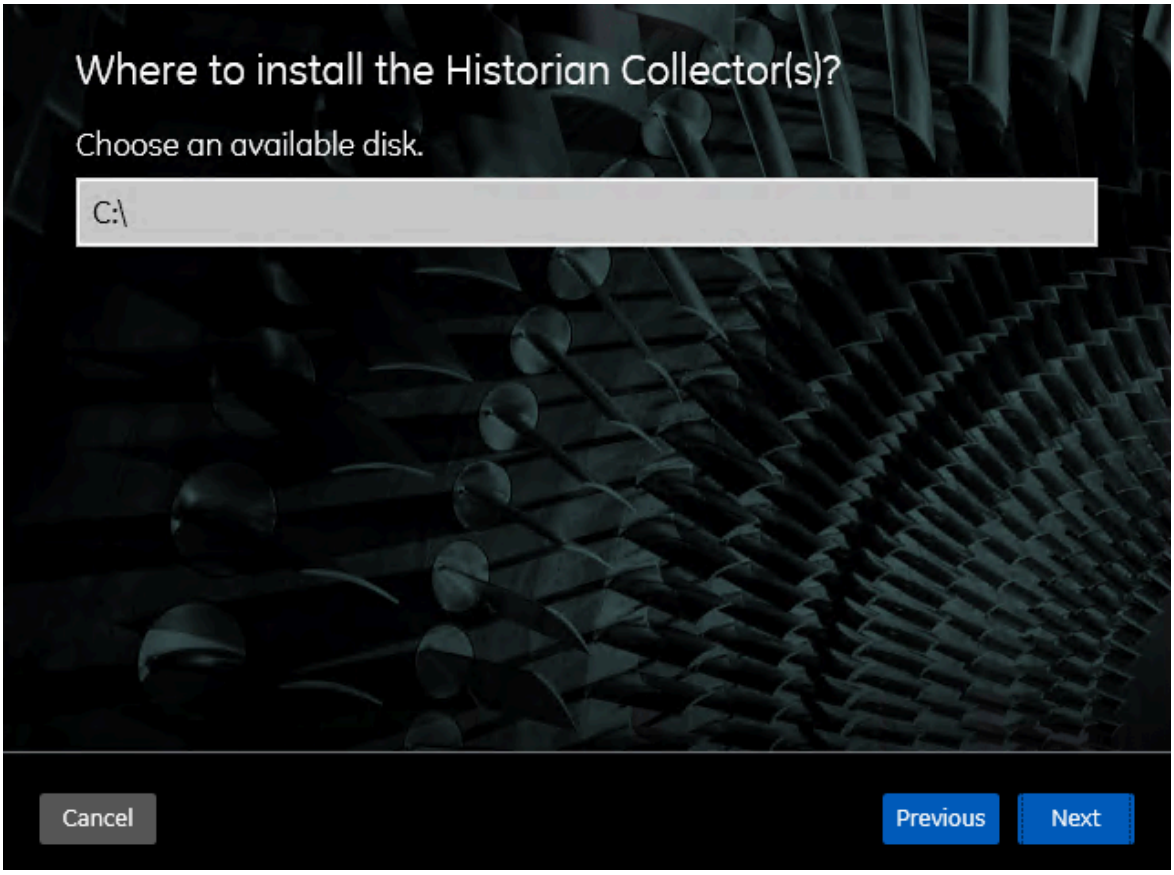
## Install Collectors Using the Installer

After you install collectors, the following artefacts will be available:

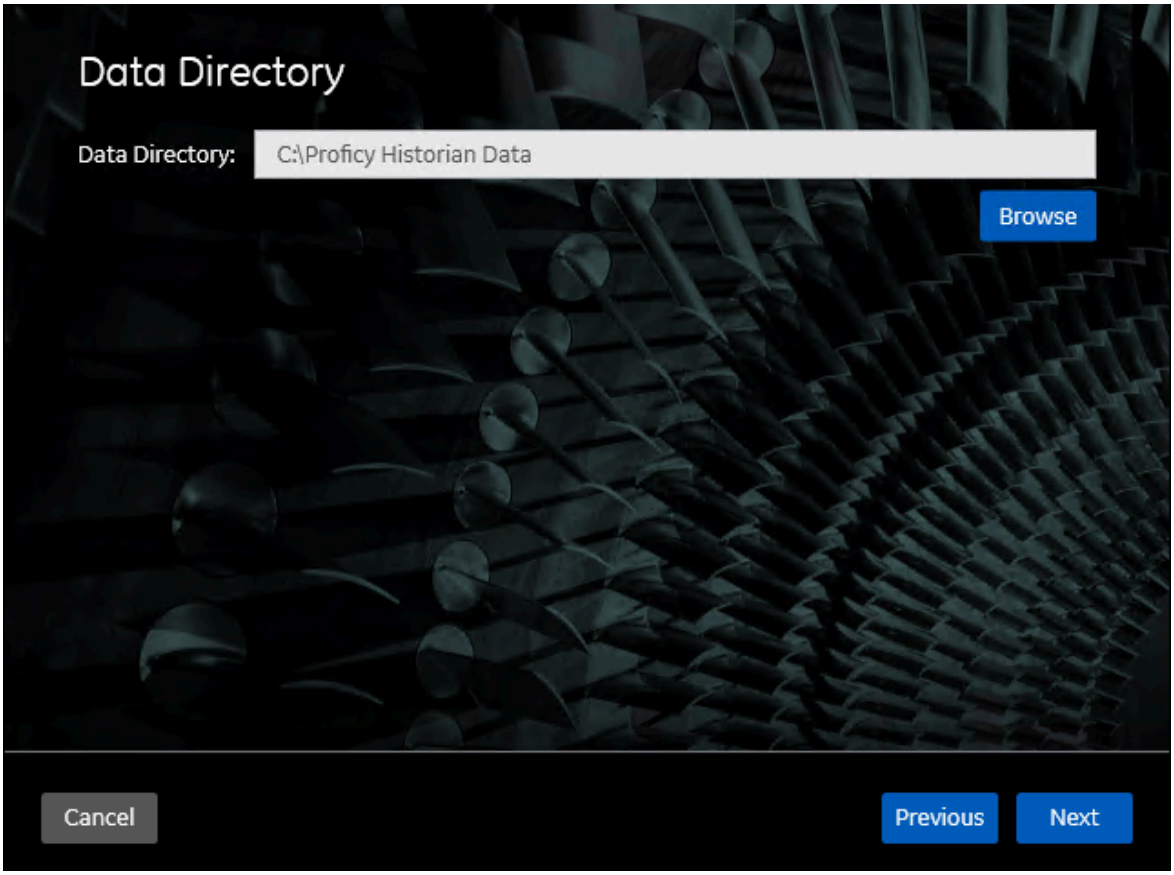
- **Executable files:** These files are required to add a collector instance.
- **Instances of the following collectors:**
  - The iFIX collector
  - The iFIX Alarms & Events collector
  - The OPC Classic Data Access collector for CIMPLICITY
  - The OPC Classic Alarms and Events collector for CIMPLICITY

  These instances will be created only if iFIX and/or CIMPLICITY are installed on the same machine as the collectors.
- **The Remote Collector Management agent:** Provides the ability to manage collectors remotely.

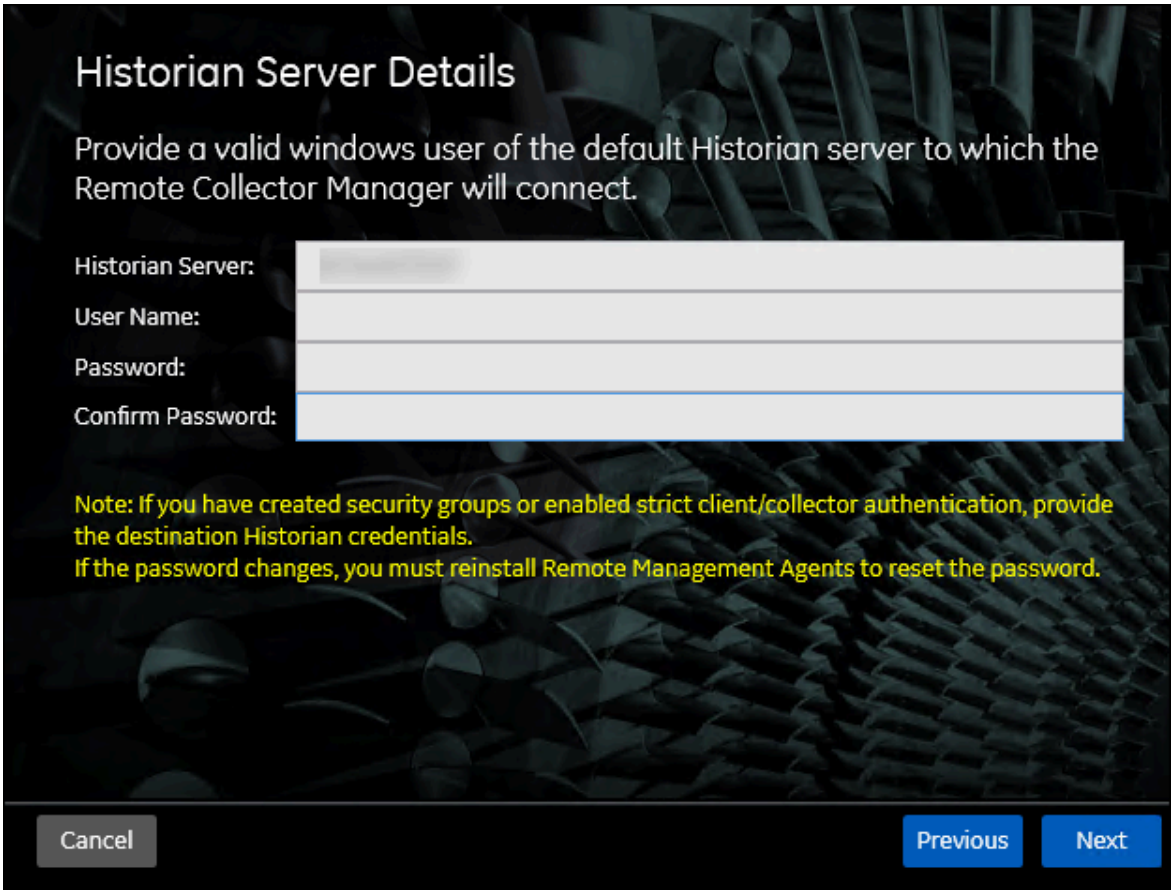
1. Run the `InstallLauncher.exe` file.
2. Select **Install Collectors**.
   The welcome page appears.
3. Select **Next**.
   The license agreement appears.
4. Select the **Accept** check box, and then select **Next**.
   The default installation drive appears.




5. If needed, modify the installation drive, and then select **Next**.
   The data directory page appears.




6. If needed, change the folder for storing the collector log files, and then select **Next**.
   The destination Historian server page appears.




7. Provide the credentials of the Windows user account of the destination Historian server to which you want Remote Management Agent to connect.

   These details are required for Remote Collector Manager to connect to Historian to manage the collectors remotely. If you are installing collectors on same machine as the Historian server, and if strict collector authentication is disabled, you need not provide these details; by default, the machine name of the local Historian server is considered. If, however, they are installed on different machines, or if strict collector authentication is enabled, you must provide the credentials of the Historian server user.

8. Select **Next**.
   The **You are ready to install** page appears.
9. Select **Install**.
   The installation begins.
10. When you are prompted to reboot your system, select **Yes**.

The collector executable files are installed in the following folder: `<installation drive>:\Program Files (x86)\GE Digital\<collector name>`. The iFIX collectors are installed in the following folder: `C:\Program Files\GE\iFIX`. The following registry paths are created:

- `HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\ GE Digital\iHistorian\Services\<collector type>`
- `HKEY_LOCAL_MACHINE\SOFTWARE\GE Digital\iHistorian\Services\<collector type>`

In addition, if iFIX and/or CIMPLICITY are installed on the same machine as the collectors, instances of the following collectors are created:

- The iFIX collector
- The iFIX Alarms & Events collector
- The OPC Classic Data Access collector for CIMPLICITY
- The OPC Classic Alarms and Events collector for CIMPLICITY

## Installing a Collector at a Command Prompt

After you install collectors and Remote Management Agent, the following artefacts will be available:

- **Executable files:** These files are required to add a collector instance.
- **Instances of the following collectors:**
  - The iFIX collector
  - The iFIX Alarms & Events collector
  - The OPC Classic Data Access collector for CIMPLICITY
  - The OPC Classic Alarms and Events collector for CIMPLICITY

  These instances will be created only if iFIX and/or CIMPLICITY are installed on the same machine as the collectors.
- **The Remote Collector Management agent:** Provides the ability to manage collectors remotely.

Using Configuration Hub, you will then add a collector instance and begin using the collector.
This topic describes how to install collectors at a command prompt. You can also install them using the installer *(on page 143)*.

1. Navigate to the `Collectors` folder in the installation folder.
2. At a command prompt, enter:

```
Collectors_Install.exe -s RootDrive=<value> DestinationServerName=<value>
DataPath="<value>" UserName1=<value> Password=<value>
```

| Parameter | Description | Default Value |
|---|---|---|
| RootDrive | The installation drive for the collectors. | `C:\` |
| DataPath | The folder for storing the collector log files. | `C:\Proficy Historian Data` |
| DestinationServerName | The host name of the destination Historian server to which you want collectors to send data.<br><br>This is required for Remote Collector Manager to connect to Historian to manage the collectors remotely. If you are installing collectors on the same machine as the Historian server, and if strict collector authentication is disabled, you need not provide the server name; by default, the machine name of the local Historian server is considered. If, however, they are installed on different machines, or if strict collector authentication is enabled, you must provide the credentials of the Historian server user. | local host name |
| UserName1 | The username of the Windows user of the destination Historian server. A value is required only if the destination Historian server and collectors are on different machines. | |

| Parameter | Description | Default Value |
|---|---|---|
| Password | The password of the Windows user of the destination Historian server. A value is required only if the destination Historian server and collectors are on different machines. | |

For example: `Collectors_Install.exe -s RootDrive=C:\ DestinationServerName=myservername DataPath="C:\Proficy Historian Data" UserName1=user123 Password=xyz123`

3. Restart the machine. If you uninstall a collector or install another one before restarting the machine, an error may occur.

The collector executable files are installed. In addition, if iFIX and/or CIMPLICITY are installed on the same machine as the collectors, instances of the following collectors are created:

- The iFIX collector
- The iFIX Alarms & Events collector
- The OPC Classic Data Access collector for CIMPLICITY
- The OPC Classic Alarms and Events collector for CIMPLICITY

1. Ensure that the Windows user that you have specified while installing collectors is added to the iH Security Admins and iH Collector Admins groups.
2. Enable trust for a client certificate for Configuration Hub.
3. Enable trust for a self-signed certificate on Chrome *(on page 97)*.
4. Import an issuer certificate.

You are now ready to use Configuration Hub. To add and manage collector instances, you can use Configuration Hub or Remote Collector Management. For instructions specific to setting up the iFIX collector and the iFIX Alarms and Events collector, refer to Working with iFIX Collectors.

## Upgrade Collectors

- Ensure that you are attempting to upgrade the collectors that were installed using the Historian Installation Package. If you previously installed the collectors using the Proficy Installer, such as the iFIX Proficy Installer, you must upgrade the collectors using the Proficy Installer only.
- If an iFIX collector instance created in version 9.0 exists, after you upgrade collectors, another instance of the iFIX collector is created. Because of this, the Remote Collector Manager (RCM)

will not work correctly. Therefore, if you want to use RCM, you must delete one of the instances. If needed, you can manually create another instance of the iFIX collector using Configuration Hub or the RemoteCollectorConfigurator utility. This is applicable to the iFIX Alarms and Events collector as well.

- For collectors earlier than version 7.1, additional registries that you create manually are deleted. Therefore, we recommend that you back up them, uninstall the collectors, and then install the latest version.

Install the collectors *(on page 142)*.
The collectors are upgraded to the latest version.

# Client Tools

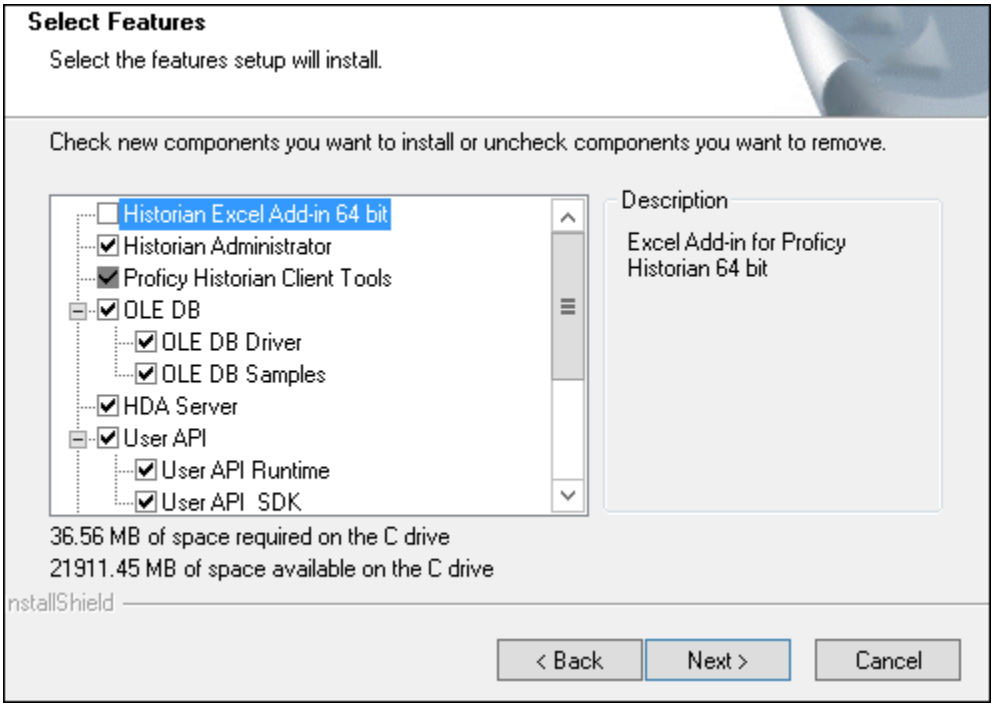
## Install Client Tools

When you install Client Tools, the following components are installed by default:

- Client Tools
- Historian Administrator
- OLE DB provider (driver and samples)
- The OPC Classic HDA server
- User API and SDK
- Historian Client Access API
- Collector Toolkit

This topic describes how to install Client Tools using the installer. You can also install it at a command prompt *(on page 153)*.

1. Run the `InstallLauncher.exe` file.
2. Select **Install Client Tools**.
   The **Select Features** page appears, displaying a list of components that you can install with Client Tools.



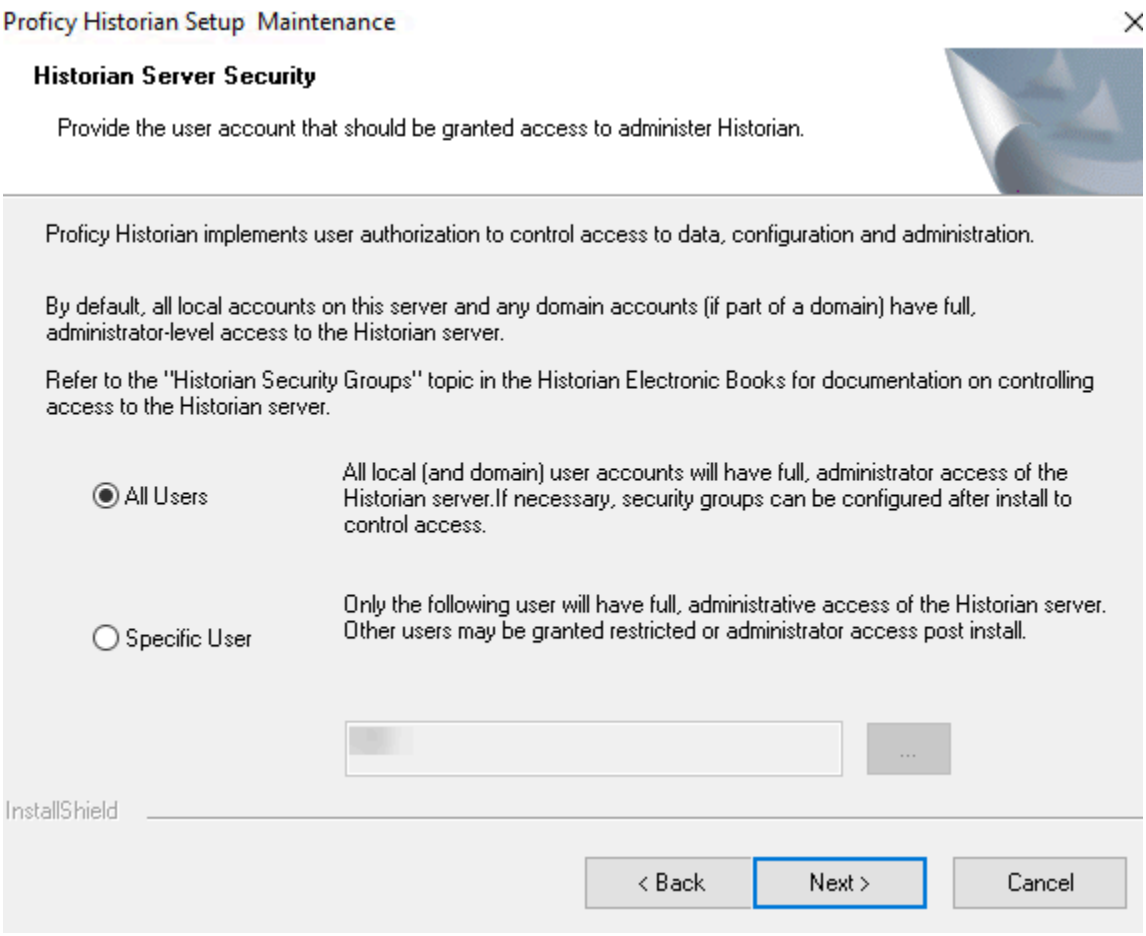

By default, the check boxes for components such as **Historian Administrator**, **HDA Server**, **OLE DB**, and **User API and SDK** are selected. If you do not want to install them at this time, clear the check boxes. You cannot, however, clear the **Proficy Historian Client Tools** check box.

**Important:**
If you are reinstalling, you must select all of the previously installed components. If you do not do so, the component will be uninstalled.

By default, the **Historian Excel Add-in 64-bit** check box is cleared. If you want to install Excel Add-In along with Client Tools installation, select the check box.

**Note:**
If using certain versions of Windows (like Windows 10 or Windows 2019), you may receive an error message while installing Excel Add-In, stating that some of the DLL files are not registered. You can ignore these messages.

The **Historian Server Security** page appears.




3. Select the user account that should be granted the access to administer Historian.
   You are provided with the following options:

| Option | Description |
|---|---|
| **All Users** | If you select this option, all local, including domain user accounts will have the administrator access of the Historian server. If needed, to control access, you can configure security groups after installation. |
| **Specific User** | If you select this option, the user that you select will have the administrator access of the Historian server. If needed, you can grant a restricted or administrator access to other users after installation. |

4. Select **Next**.
5. When you are asked to reboot your system, select **Yes**.

Client Tools, along with the selected components, are installed in the following folder: `<installation drive>:\Program Files\Proficy\Proficy Historian\x86\<tool name>`. If you have selected HDA Server, Microsoft .NET Framework 4.5 and the OPC Core Components 3.00 redistributable are installed as well.

## Install Client Tools at a Command Prompt

1. If you want to install Excel Add-In for Historian, install one of the following 32-bit or 64-bit Microsoft® Excel® applications:
    - Microsoft® Excel® 2019
    - Microsoft® Excel® 2016
    - Microsoft® Excel® 2013
    - Microsoft® Excel® 2010
2. Install Client Tools using the installer *(on page 150)* on a machine. When you do so, a template file named `setup.iss` is created at `C:\Windows`. This file stores the installation options that you have provided. You can then use this template to install Client Tools at a command prompt on other machines.

When you install Client Tools, the following components are installed by default:

- Client Tools
- Historian Administrator
- OLE DB driver and samples
- The OPC Classic HDA server
- User API and SDK
- Historian Client Access API
- Collector Toolkit

1. Copy the `setup.iss` file to the machine on which you want to install Client Tools at a command prompt.
2. In the folder in which you have copied the file, run the following command: `setup.exe /s /sms`
   The installer runs through the installation steps.

> **Note:**
> If using certain versions of Windows (like Windows 10 or Windows 2019), you may receive an error message, stating that some of the DLL files are not registered. You can ignore these messages.

3. When prompted to reboot your system, select **Yes**.

Client Tools are installed.

If you have installed Excel Add-in, activate it *(on page 197)*.

## About Installing Web-based Clients

Using Web-based Clients, you can configure and manage Historian systems and their components using a browser.

When you install Web-based Clients, you can install the following components:

- Configuration Hub: Allows you to manage Historian systems and its components. You can set up stand-alone as well as horizontally scalable systems. You can also add collector instances and manage them.
- Trend Client: Provides access to process and equipment data, allowing you to quickly troubleshoot and make improvements, leading to time and cost savings through the use of trend charts and current value tables.
- The Web Admin console: Allows you to monitor, supervise, archive, retrieve, and control data gathered from Historian systems.
- The Proficy Authentication service (optional): Provides identity-based security for applications and APIs. It supports open standards for authentication and authorization, including Oauth2. You can install Proficy Authentication and Configuration Hub, or you can point to existing Proficy Authentication and Configuration Hub instances.
- Rest APIs: Allow you to query data from a Historian server.

You can install Web-based Clients using a GUI-based installer *(on page 155)* or at a command prompt *(on page 171)*.

> **Important:**
> When you install Web-based Clients:

- If you want to reinstall the same version of Web-based Clients, you must uninstall and then install Web-based Clients.
- If, even after installing Web-based Clients, you cannot access a web component, start the Proficy Operations Hub Httpd Reverse Proxy and the Data Archiver services.

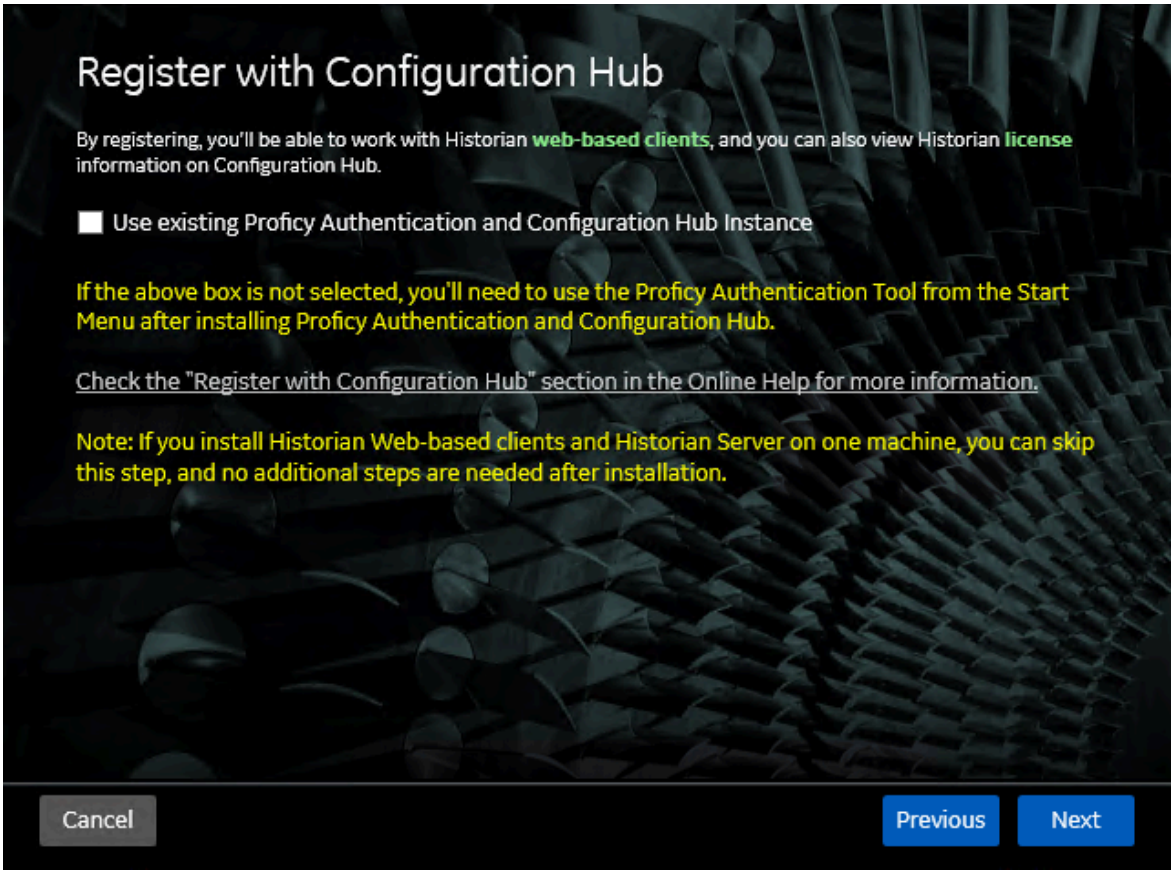
## Install Web-based Clients Using the Installer

1. Install the Historian server *(on page 105)*. During the installation, in the **Register with Configuration Hub** page, select the **Use existing Proficy Authentication and Configuration Hub Instance** check box, and the provide the Proficy Authentication server and Configuration Hub server details.

   **Note:**
   This step is needed only in the following cases:
   - The Historian Server and the Web-based clients will be on two different machines.
   - You already installed Proficy Authentication and Configuration Hub, and you want use the Historian Web-based clients and view the Historian license information on Configuration Hub.

     If you do not select the **Use existing Proficy Authentication and Configuration Hub Instance** during the installation, to use the Historian Web-based clients and view the Historian license information on Configuration Hub, you must use the Proficy Authentication Tool *(on page 137)* to register with Proficy Authentication and Configuration Hub servers.

     If you do not have Proficy Authentication and Configuration Hub installed, and you intend to use the Historian Web-based clients and view the Historian license information on Configuration Hub, you can install them while installing the Web-based clients.



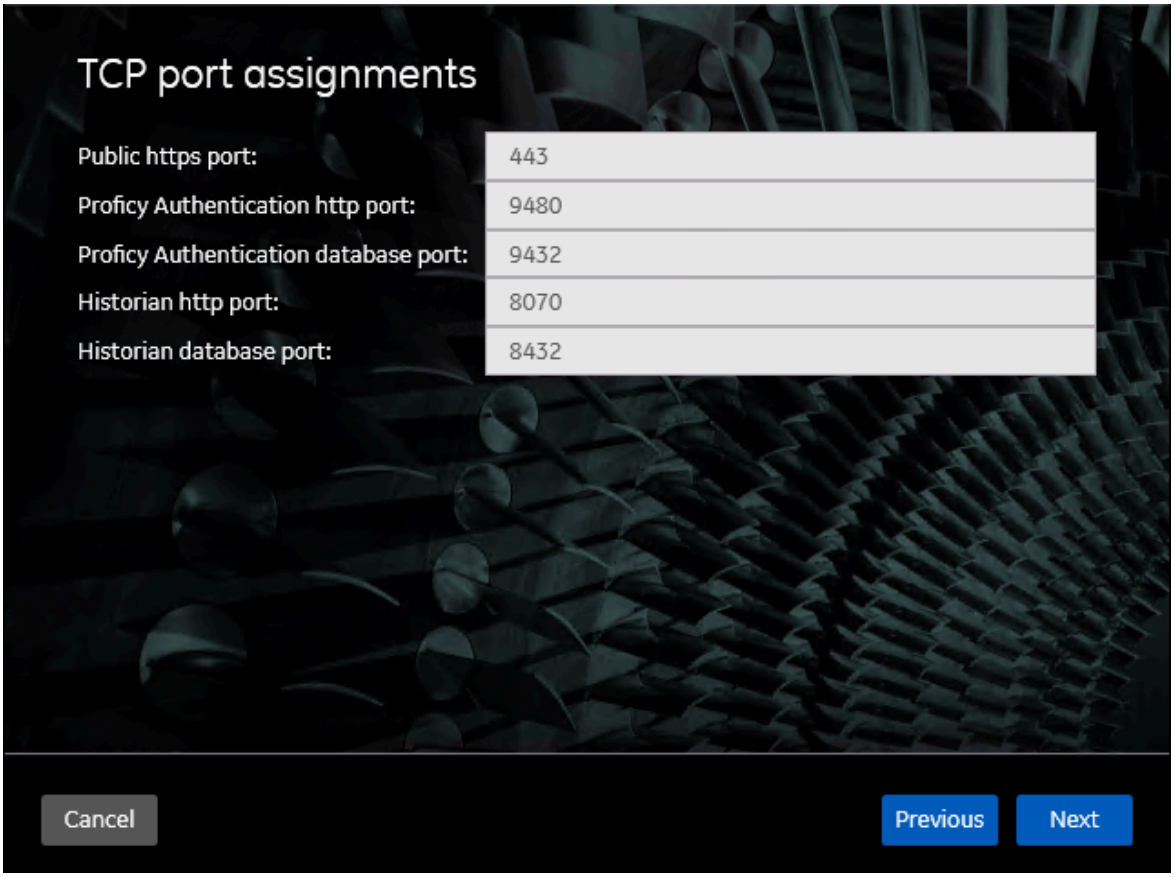

2. If you want to use Web-based Clients in a cluster environment, ensure that your network is enabled for multicast traffic, and set up high availability on each node in the cluster.

This topic describes how to install Web-based Clients using a GUI-based installer.

You can also install Web-based Clients using the command line.

During the installation, you can choose to use Web-based Clients in a cluster environment, thus ensuring high availability of connection to the Historian server using the client applications.

1. Run the `InstallLauncher.exe` file.
2. Select **Install Web-based Clients**.
   The welcome page appears.
3. Select **Next**.
   The license agreement appears.
4. Select the **Accept** check box, and then select **Next**.
   The **TCP port assignments** page appears.



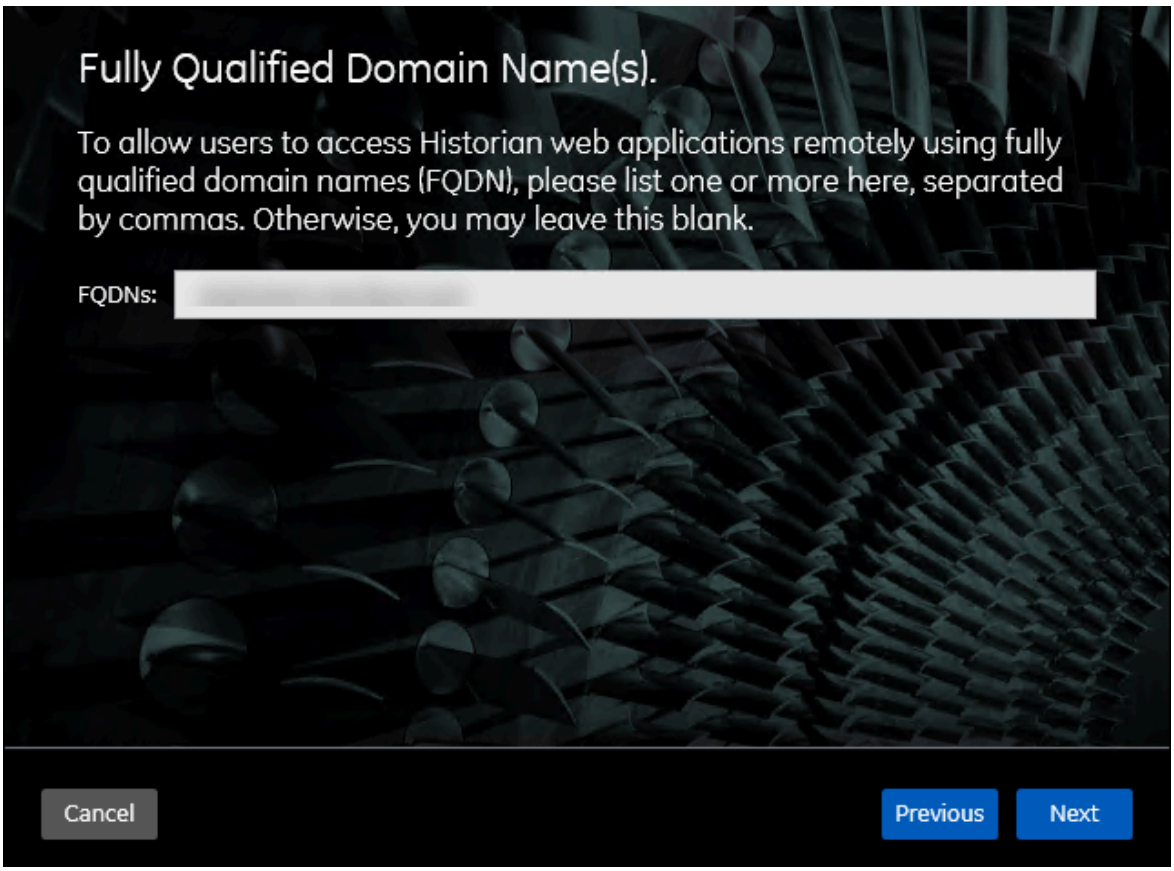

5. As needed, change the values for TCP port assignments as described in the following table, and then select **Next**.

| Field | Description |
|---|---|
| **Public https port** | Port for https protocol communication used by Web-based Clients (through a firewall). The default value is 443. Ensure that this port number matches the one you specify while installing the Historian server. In addition:<br>◦ If you will install Operations Hub later on the same machine, the value that you provide in this field is populated while installing Operations Hub.<br>◦ If you have already installed Operations Hub on the same machine, this field is disabled and populated with the value you have provided while installing Operations Hub. |
| **Proficy Authentication http port** | Port for http protocol communication used by the Proficy Authentication service. The default value is 9480. |

| Field | Description |
|---|---|
| **Proficy Authentication database port** | Port for the Proficy Authentication database. The default value is 9432. |
| **Historian http port** | Port for the http protocol communication used by Web-based Clients. The default value is 8070. |
| **Historian database port** | Port for the PostgreSQL Historian database. The default value is 8432. |

The **Fully Qualified Domain Name(s)** page appears.

- If you will install Operations Hub later on the same machine, the value that you provide in the **FQDNs** field is populated while installing Operations Hub.
- If you have already installed Operations Hub on the same machine, the **FQDNs** field is disabled and populated with the value you have provided while installing Operations Hub.



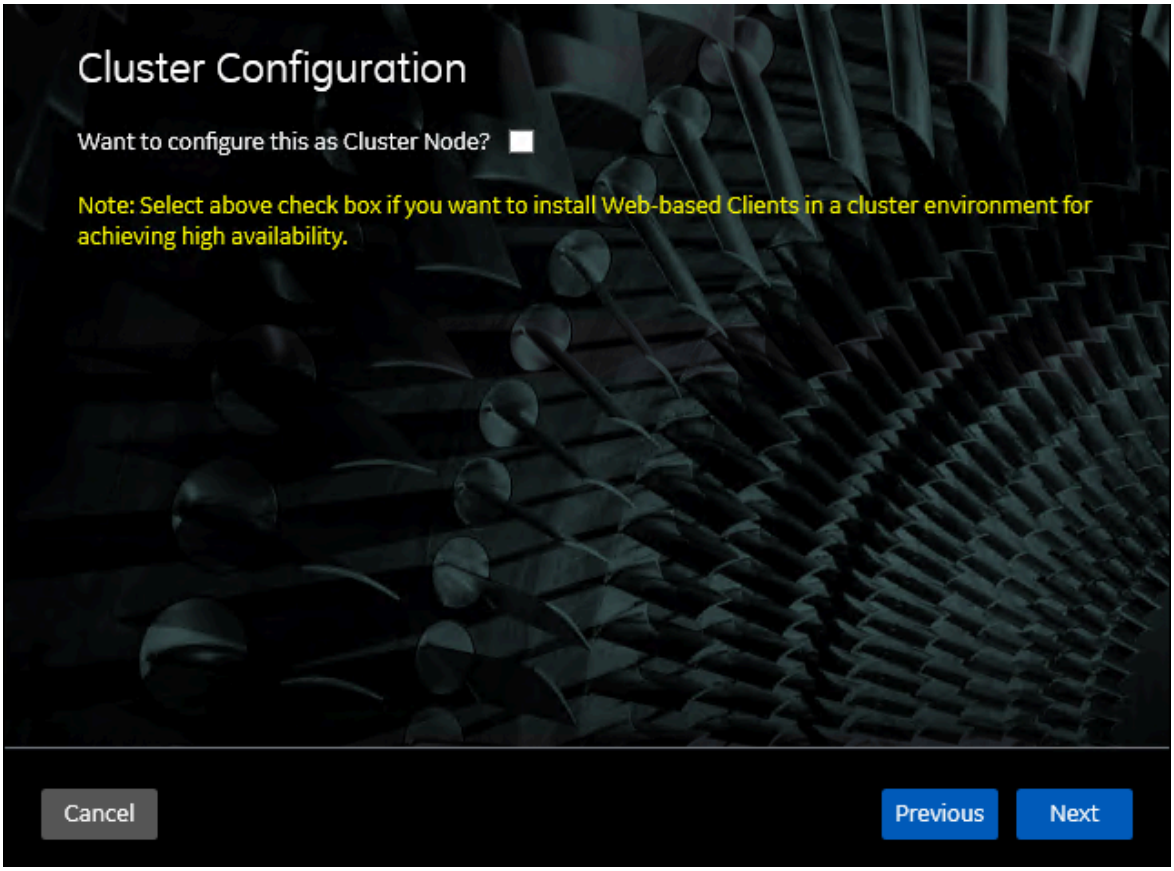
6. In the **FDQNs** field, enter the fully qualified domain names, and then select **Next**.
   This enables you to access Historian web applications remotely. You can use it to access the Web Admin console using alias names. Enter the values separated by commas.
   To access the Web Admin console using any of the following URLs, enter `Test.abc.ge.com,localhost,127.0.0.1,aliasName`
   - https:// Test.abc.ge.com /historian-visualization/hwa
   - https:// 127.0.0.1 /historian-visualization/hwa
   - https:// aliasName /historian-visualization/hwa
   - https:// localhost /historian-visualization/hwa

   **Important:**
   - Do not enter a space between the values.
   - You must add the IP address and alias name in the `hosts` file located at `C:\Windows\System32\drivers\etc`. The IP address that you add must be a static or fixed IP address.

     **Format:** `<IP address> <alias name>`

     **Example:** `1.2.3.4 myservername`
   - FQDN is not supported for Configuration Hub.

   The **Cluster Configuration** page appears.




If, however, you are upgrading Web-based Clients, this page does not appear. In that case, skip the next step.

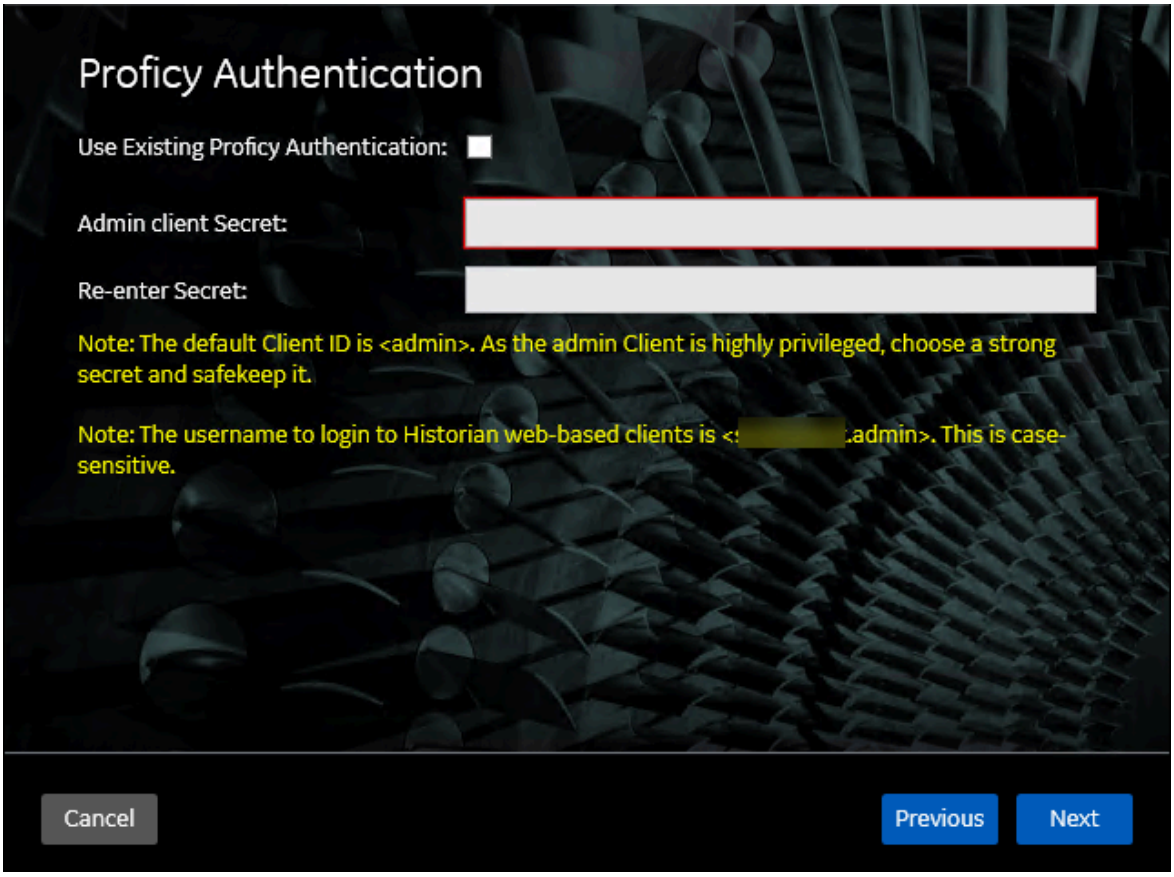
7. If you want high availability of Web-based Clients, select the **Cluster Node** check box, and enter values as described in the following table.

| Field | Description |
|---|---|
| **Historian Database Folder** | Provide the database folder in the shared drive that you have created. The default value is `C:\ProgramData\GE\OperationsHub`. You *must* change this value. |
| **Cluster FQDN** | Enter the client access point of the role for which you have added the resources while setting up high availability *(on page 182)*. |
| **Multicast Address** | If needed, modify the common IP address that all the nodes in the cluster can use. Enter a value between 224.0.0.0 and 239.255.255.255 |

| Field | Description |
|---|---|
| | (or a hostname whose IP address falls in this range).The default value is 228.0.0.4. |
| **Historian Cluster Membership Port** | If needed, modify the common port number that all the nodes in the cluster can use. The default value is 45564. This port number, in conjunction with the multicast address, is used to create the cluster. |
| **Historian Cluster Receiver Port** | If needed, modify the multicast port number that you want to use for incoming Historian data. The default value is 4000. |

8. Select **Next**.

   The **Proficy Authentication** page appears, allowing you to choose whether you want to install Proficy Authentication along with Web-based Clients installation or use an existing Proficy Authentication.



- If you want to install Proficy Authentication, clear the **Use Existing Proficy Authentication** check box. If you want to include Proficy Authentication in the cluster, you must install Proficy Authentication locally on each cluster node.
- If you want to use an existing Proficy Authentication server, select the **Use Existing Proficy Authentication** check box. Proficy Authentication is detected if you installed it using a unified installer or Operations Hub, or if Historian uses Proficy Authentication installed remotely from an earlier version.

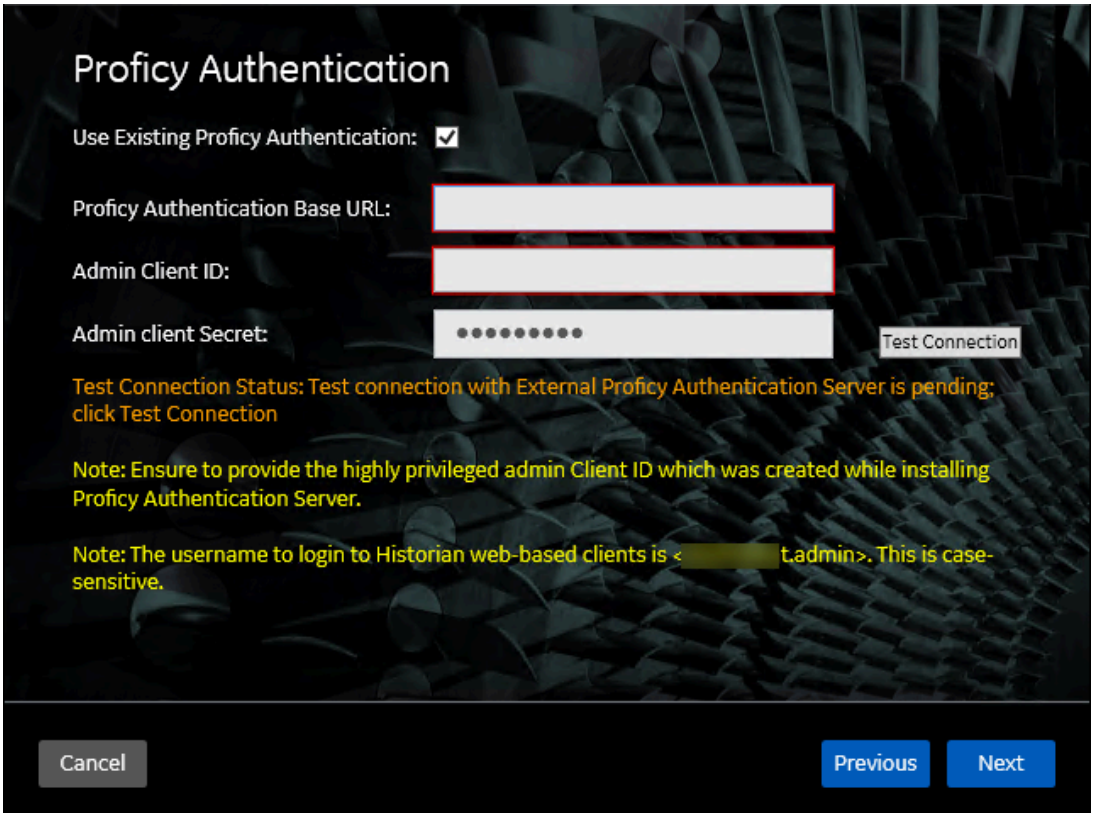
9. If you want to install Proficy Authentication, enter the **Admin client secret**, re-enter the secret, and then select **Next**.
   The admin client secret must satisfy the following conditions:
   - Must not contain only numbers.
   - Must not begin or end with a special character.
   - Must not contain curly braces.

   **Note:**
   The format of username for Historian Web-based Clients is <host name>.admin, where <host name> is the machine on which Web-based Clients are installed. And, the default client ID is admin. Both the host name and client ID are case-sensitive.

   If, however, the Proficy Authentication server hostname is long, resulting in a username longer than 20 characters, Windows does not allow you to create the user. In that case, you can create a Proficy Authentication user, and then create the corresponding Windows user, using the uaa_config_tool utility.

10. Alternatively, if you want to use an existing Proficy Authentication service (that is, a Proficy Authentication instance already installed by an external application such as Operations Hub):

    a. Select the **Use Existing Proficy Authentication** check box.
       The fields for the existing Proficy Authentication service appear.

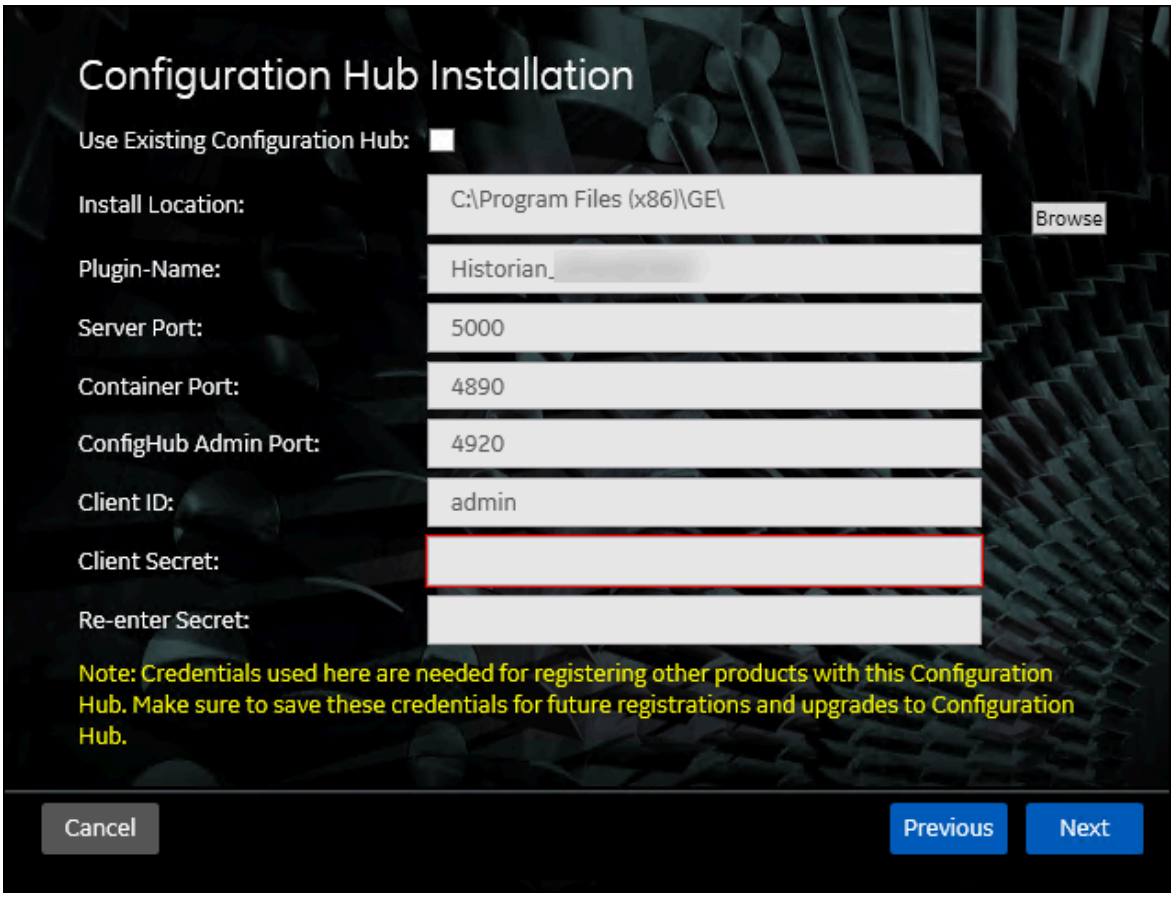



b. Enter values as described in the following table.

| Field | Description |
|---|---|
| **Proficy Authentication Base URL** | Enter the URL of the external Proficy Authentication server in the following format: https://*<Proficy Authentication server name>*:*<port number>*, where *<Proficy Authentication server name>* is the FQDN or hostname of the machine on which Proficy Authentication is installed. By default, the port number is 443.<br>**Note:**<br>Do not enter a trailing slash character. |
| **Admin Client ID** | Enter the client name that you provided while installing the external Proficy Authentication. The default value is admin. |
| **Admin Client Secret** | Enter the client secret that you provided while installing the external Proficy Authentication. |

c. Select **Test Connection**.
   The results of the connection test appear. You cannot proceed until the connection is successful.

11. Select **Next**.
    The **Configuration Hub Installation** page appears, allowing you to choose whether you want to install Configuration Hub along with Web-based Clients or use an existing Configuration Hub.




Configuration Hub allows you to add and manage a collector instance remotely. For more information, refer to About Configuration Hub.

If, however, an earlier version of Configuration Hub is available on the same machine, you will be prompted to enter the details of the existing Configuration Hub, and it will be upgraded to the latest version. If that happens, skip the next step.

**Important:**
By default, Configuration Hub points to the same Proficy Authentication server as the one you provided during the Historian server installation. If you want to install Web-based Clients in a cluster environment, ensure that:

- Configuration Hub does not use the same Proficy Authentication server as that used by the cluster.
- The Proficy Authentication and Configuration Hub details must be the same for all cluster nodes.

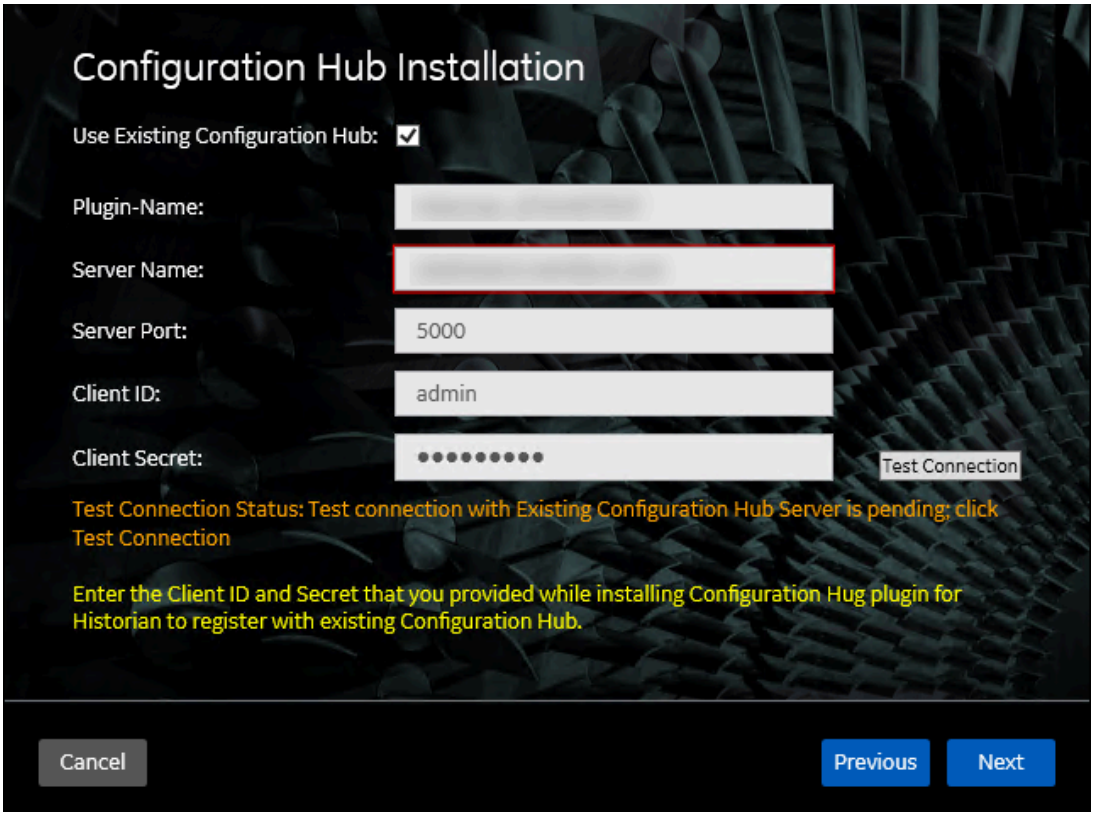
12. If you want to install Configuration Hub, ensure that the **Use Existing Configuration Hub** check box is cleared, and then provide values as described in the following table.

| Field | Description |
|---|---|
| **Install Location** | If needed, modify the installation folder for Configuration Hub. |
| **Plugin Name** | If needed, modify the name of the Configuration Hub plugin for Historian. The default value is in the following format: Historian_<*host name*>. If, however, you are installing Web-based Clients in a cluster environment, the default value is Historian_<*cluster name*>. You can modify this value, but provide the same value for all the nodes in the cluster. |
| **Server Port** | If needed, modify the port number that you want to use for the web server. The default value is 5000. If you want to install Web-based Clients in a cluster environment, provide the same value for all the nodes in the cluster. |
| **Container Port** | If needed, modify the port number for the Configuration Hub container. The default value is 4890. |
| **ConfigHub Admin Port** | This is the port number of the Configuration Hub admin. The default value is 4890. If needed, you can change the port number. |

| Field | Description |
|---|---|
| **Client ID** | Enter the username to connect to Configuration Hub. The default value is admin. The value that you enter can contain:<br>◦ All English alphanumeric characters (ABCDEFGHIJKLMNOPQRSTUVXYZ abcdefghijklmnopqrstuvwxyz_-0123456789)<br>◦ The following special characters: ><:~!@#$%^&*?\| |
| **Client Secret** | Enter the password to connect to Configuration Hub. The value that you enter can contain:<br>◦ Must contain at least eight characters.<br>◦ All English alphanumeric characters (ABCDEFGHIJKLMNOPQRSTUVXYZ abcdefghijklmnopqrstuvwxyz_-0123456789)<br>◦ The following special characters: ><:~!@#$%^&*?\| |
| **Re-enter Secret** | Re-enter the password to connect to Configuration Hub. |

13. Alternatively, if you want to use an existing Configuration Hub:

    a. Select the **Use Existing Configuration Hub** check box. This check box is disabled if an existing Configuration Hub is detected.
    The fields for the existing Configuration Hub appear.



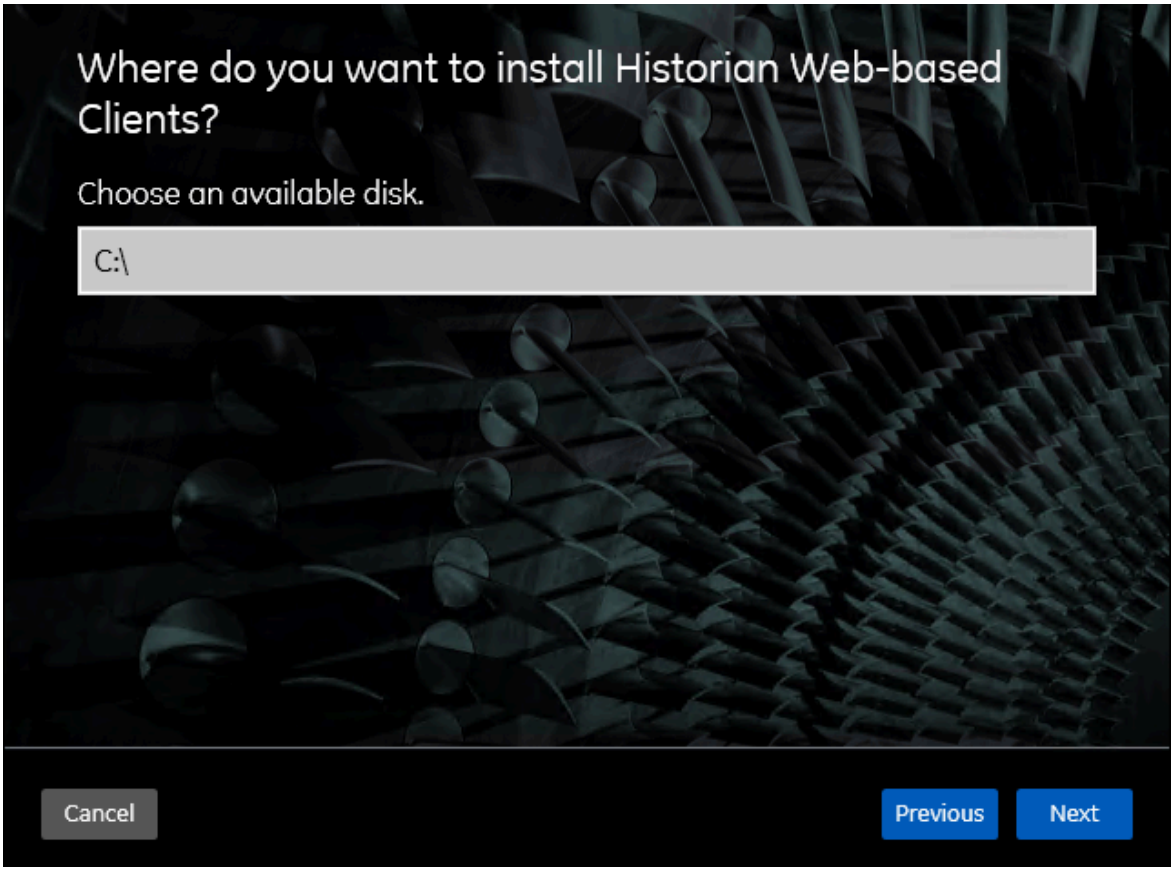

b. Provide values as described in the following table.

| Field | Description |
|---|---|
| **Plugin Name** | If needed, modify the name of the Configuration Hub plugin for Historian. The default value is in the following format: Historian_-*<host name>* |
| **Server Name** | Enter the server name or the FQDN of the existing Configuration Hub server, as displayed in the address bar of the browser when you access Configuration Hub from the machine where Configuration Hub is installed. |
| **Server Port** | If needed, modify the port number that you want to use for the web server. The default value is 5000. |

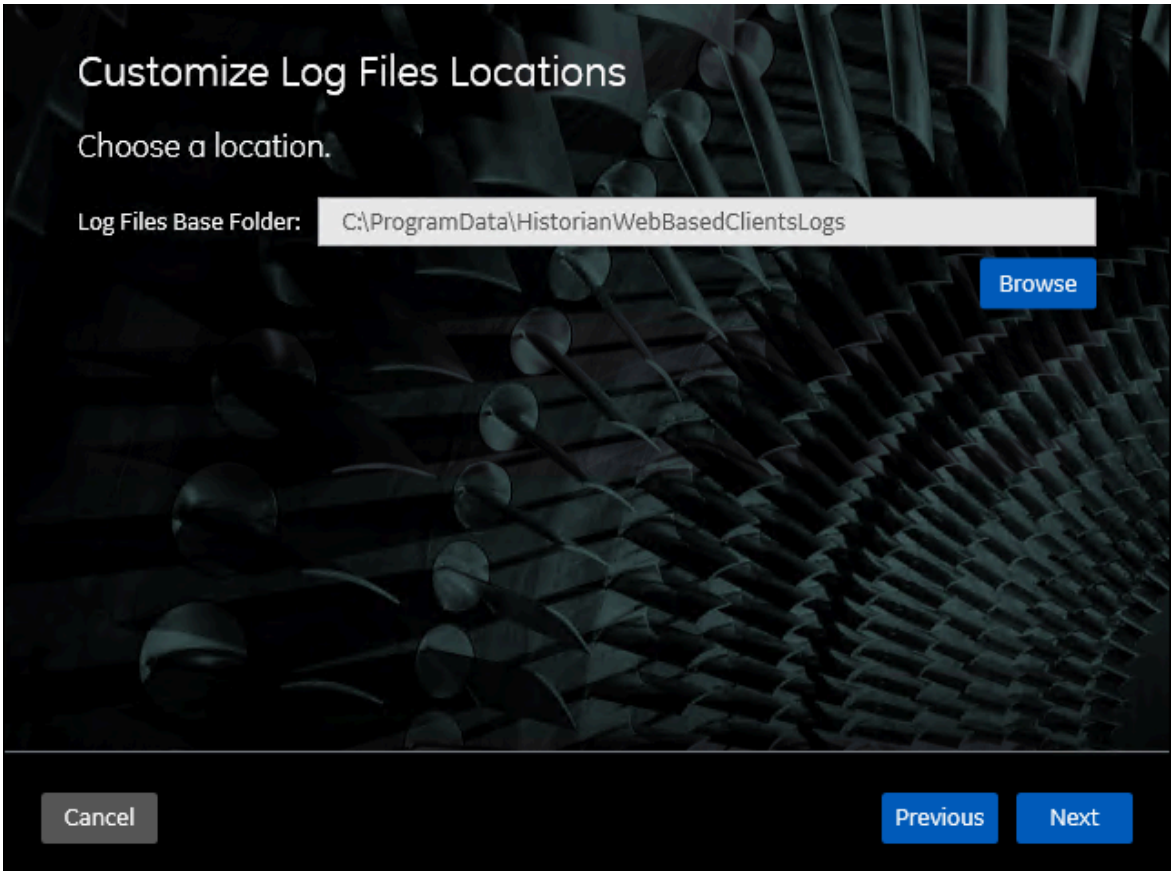
| Field | Description |
|---|---|
| **Client ID** | If needed, modify the username to connect to Configuration Hub. The default value is admin. |
| **Client Secret** | Enter the password to connect to Configuration Hub. |

c. Select **Test Connection**.
The results of the connection test appear. You cannot proceed until the connection is successful.

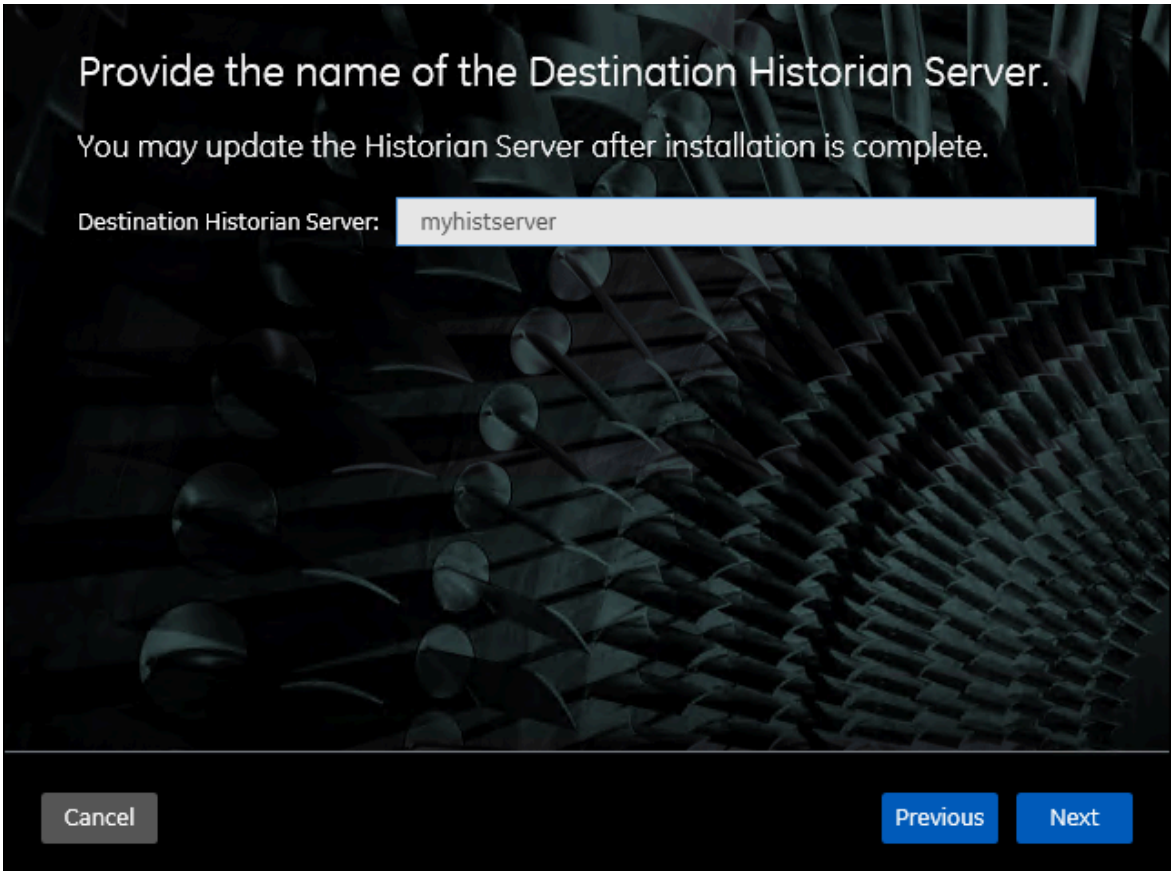
14. Select **Next**.
The default installation drive appears.




15. If needed, change the installation drive for Web-based Clients, and then select **Next**.
The log files location page appears.




16. If needed, change the location for log files, and then select **Next**.
    The destination Historian server page appears.




17. Provide the name of the destination Historian server to which Web-based Clients are connected by default. When you login to Configuration Hub, the default system will point to this server.

> **Note:**
> - Provide the name of either Historian single-server or mirror primary server because the systems in Configuration Hub will be either a stand-alone system or a horizontally scalable system.
> - If you want to connect to a remote Historian server, you must disable the **Enforce Strict Client Authentication** and **Enforce Strict Collector Authentication** options using Historian Administrator in the remote server.

18. Select **Next**.
    The **You are ready to install** page appears.
19. Select **Install**.
    The Web-based Clients installation begins.
20. When you are prompted to reboot your machine, select **Yes**.

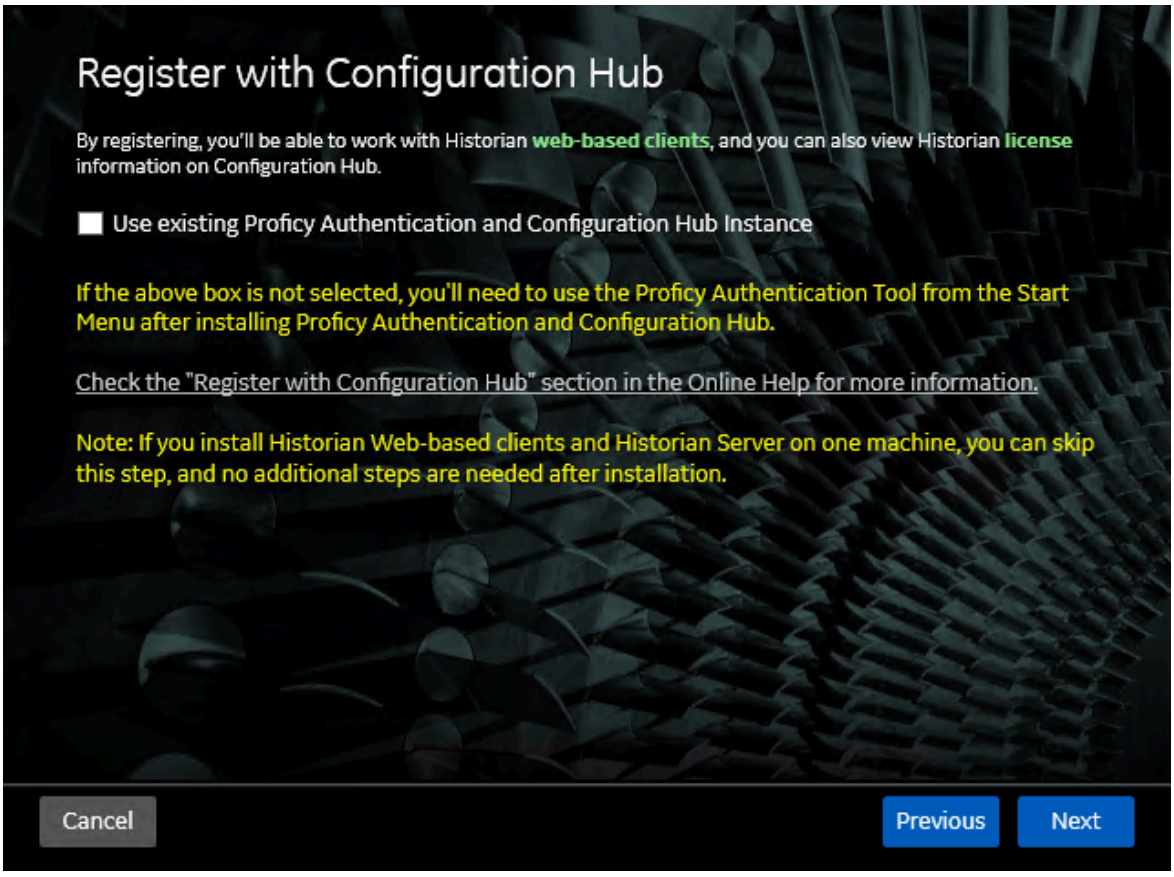
Historian Web-based Clients are installed in the following folder: `<installation drive>:\Program Files\GE`, and the following registry paths are created:

- `HKEY_LOCAL_MACHINE\SOFTWARE\GE Digital`
- `HKEY_LOCAL_MACHINE\SOFTWARE\GE`

If you want to use Configuration Hub installed using other products such as iFIX, Plant Applications, and so on, set up authentication to point to the Proficy Authentication instance.

## Install Web-based Clients at a Command Prompt

1. Install the Historian server *(on page 105)*. During the installation, in the **Configure Web-based Clients** page, select the **Do you wish to continue configuring web-based clients** check box, and provide the Proficy Authentication server and Configuration Hub server details.




2. If you want to use Web-based Clients in a cluster environment, ensure that your network is enabled for multicast traffic, and set up high availability on each node in the cluster.

This topic describes how to install Web-based Clients at a command prompt. You can also install Web-based Clients using an installer *(on page 154)*.

During the installation, you can choose to use Web-based Clients in a cluster environment, thus ensuring high availability of connection to the Historian server using the client applications.

1. If you want to install Web-based Clients with the default values, run the following command:

   ```
   install.exe /quiet AdminClientSecret=<password> ConfigHubClientSecret=<password>
   ```

2. If you want to modify the default values, run the following command:

   ```
   install.exe /quiet AdminClientSecret=<password> ConfigHubClientSecret=<password> <parameter>=<value>
   ```

   The following table describes the installation parameters.

| Parameter | Description | Default Value |
|---|---|---|
| `PublicPort` | Port for https protocol communication used by Web-based Clients. Ensure that the value for this parameter matches the one you specify while installing the Historian server. In addition:<br>◦ If you will install Operations Hub later on the same machine, the value that you provide for this parameter is populated while installing Operations Hub.<br>◦ If you have already installed Operations Hub on the same machine, provide the same value that you provided while installing Operations Hub. | 443 |

| Parameter | Description | Default Value |
|---|---|---|
| `UAAHttpPort` | Port for http protocol communication used by the Proficy Authentication service. | 9480 |
| `UAADatabasePort` | Port for the Proficy Authentication database. | 9432 |
| `HistorianHttpPort` | Port for the http protocol communication used by Web-based Clients. | 8070 |
| `HistorianDatabasePort` | Port for the PostgreSQL Historian database. | 8432 |
| `EmbeddedWebServerAlternativeNames` | The fully qualified domain names to access Historian web applications remotely. You can use it to access the Web Admin console using alias names. Enter the values separated by commas.<br><br>For example, to access the Web Admin console using any of the following URLs, enter `Test.abc.ge.com,localhost,127.0.0.1,aliasName`<br>◦ https:// Test.abc.ge-.com /historian-visualization/hwa<br>◦ https:// 127.0.0.1 /historian-visualization/hwa<br>◦ https:// aliasName /historian-visualization/hwa<br>◦ https:// localhost /historian-visualization/hwa | |

| Parameter | Description | Default Value |
| --- | --- | --- |
| | **Important:**<br>◦ Do not enter a space between the values.<br>◦ If you have already installed Operations Hub on the same machine, enter the same value that you have provided while installing Operations Hub.<br>◦ If you will install Operations Hub later on the same machine, the value that you provide for this parameter is used while installing Operations Hub.<br>◦ You must add the IP address and alias name in the `hosts` file located at `C:\Windows\System32\drivers\etc`. The IP address that you add | |

| Parameter | Description | Default Value |
|---|---|---|
| | must be a static or fixed IP address.<br>**Format:** `<IP address> <alias name>`<br>**Example:** `1.2.3.4 myservername`<br>◦ FQDN is not supported for Configuration Hub. | |
| `AdminClientId` | The client ID to connect to the Proficy Authentication service. | |
| `AdminClientSecret` | The password to connect to the Proficy Authentication service. The password must satisfy the following conditions:<br>◦ Must not contain only numbers.<br>◦ Must not begin or end with a special character.<br>◦ Must not contain curly braces.<br>If the password does not satisfy these conditions, the installation may be successful, but Web-based Clients will not work. | Not applicable |

| Parameter | Description | Default Value |
|---|---|---|
| | **Note:**<br>The format of user-name for Historian Web-based Clients is <host name>.admin, where <host name> is the machine on which Web-based Clients are installed. And, the de-fault client ID is admin. Both the host name and client ID are case-sensitive.<br><br>If, however, the Proficy Authentication server hostname is long, re-sulting in a username longer than 20 char-acters, Windows does not allow you to create the user. In that case, you can create a Profi-cy Authentication user, and then create the corresponding Win-dows user, using the uaa_config_tool utility. | |
| `UseExternalUaa` | Identifies whether you want to use an external Proficy Authen-tication service (that is, a Profi-cy Authentication instance al-ready installed by an external application such as Operations | 0 |

| Parameter | Description | Default Value |
|---|---|---|
| | Hub). If you want to use external Proficy Authentication, enter 1. | |
| `ActiveUaaBaseUrl` | The base URL to connect to external Proficy Authentication. A value is required only if you want to connect to an external Proficy Authentication service. Enter a value in the following format: https://*<Proficy Authentication server machine name>*:*<public https port number>*. By default, the port number is 443. | |
| `DoConfigHubNeedToInstall` | Identifies whether you want to install Configuration Hub as part of the web-based clients. You can enter 1 to install, or 0 to not install Configuration Hub as you may already have it installed. | 0 |
| `ConfigHubContainerPort` | The web server (https) port that you want to use for Configuration Hub. | 5000 |
| `ContainerPort` | The port for the Configuration Hub container. | 4890 |
| `ConfigHubAdminPort` | The port for the administration service for Configuration Hub. | 4920 |
| `ConfigHubInstallFolder` | The installation folder for Configuration Hub. | `C:\Program Files (x86)\GE\Configuration-Hub` |
| `UseExternalConfigHub` | Identifies whether you want to use Configuration Hub in- | 0 |

| Parameter | Description | Default Value |
|---|---|---|
| | stalled with iFIX or on a remote machine. If you want to use an external Configuration Hub, enter 1. | |
| `RootDrive` | Enter the path pointing to a remote Configuration Hub. For example `E:\` | |
| `ExternalConfigHubMachineName` | Enter the server name or the FQDN of the existing Configuration Hub server, as displayed in the address bar of the browser when you access Configuration Hub from the machine where Configuration Hub is installed. | |
| `ConfigHubClientID` | The username to connect to Configuration Hub. The value that you enter can contain:<br>◦ All English alphanumeric characters (ABCDEFGHIJKLMNOPQRSTUVXYZ abcdefghijklmnopqrstuvwxyz_0123456789)<br>◦ The following special characters: ><:~!@#$%^&*?\| | admin |
| `ConfigHubClientSecret` | The password to connect to Configuration Hub. The value that you enter can contain:<br>◦ Must contain at least eight characters.<br>◦ All English alphanumeric characters (ABCDE- | |

| Parameter | Description | Default Value |
|---|---|---|
| | FGHIJKLMNOPQRS-TUVXYZ abcde-fghijklmnopqrstu-vwxyz_0123456789)<br>◦ The following special characters: ><:~!@#$ %^&*?\| | |
| `IsClusterNode` | Indicates whether you want to install Web-based Clients in a cluster environment. If you want to install Web-based Clients in a cluster environment, enter 1. This is not applicable if you are upgrading Web-based Clients. | 0 |
| `PostgresBaseDir` | The folder in the shared drive that you want to use for Historian database if you want to add the Web-based Clients server to a cluster. | C:\ProgramData\GE\Operations Hub |
| `ClusterFQDN` | Enter the client access point of the role for which you have added the resources while setting up high availability *(on page 182)*. | |
| `MulticastAddress` | The common IP address that all the nodes in the cluster can use. Enter a value between 224.0.0.0 and 239.255.255.255 (or a hostname whose IP address falls in this range). | 228.0.0.4 |
| `HistorianClusterMembership-Port` | The common port number that all the nodes in the cluster can use. This port number, in con- | 45564 |

| Parameter | Description | Default Value |
|---|---|---|
| | junction with the multicast address, is used to create the cluster. | |
| `HistorianClusterReceiverPort` | The multicast port number that you want to use for incoming Historian data. | 4000 |
| `DataPath` | The path to the log files. | `C:\ProgramData\HistorianWebBasedClientsLogs` |
| `DestinationHistorian` | The name of the destination Historian server.<br><br>**Note:**<br>If you want to connect to a remote Historian server, you must disable the **Enforce Strict Client Authentication** and **Enforce Strict Collector Authentication** options using Historian Administrator in the remote server. | |

To install Web-based Clients with local Proficy Authentication and local Configuration Hub, run the following command:

```
Install.exe /quiet PublicPort=443 UAAHttpPort=9480 UAADatabasePort=9432 HistorianHttpPort=8070
HistorianDatabasePort=8432 AdminClientId=admin AdminClientSecret=abc
ConfigHubContainerPort=5000 ContainerPort=4890 ConfigHubInstallFolder="C:\Program Files
 (x86)\GE\ConfigurationHub"
ConfigHubClientID=admin ConfigHubClientSecret=xyz
DataPath="C:\ProgramData\HistorianWebBasedClientsLogs"
```

To install Web-based Clients with local UAA and an external Configuration Hub, run the following command:

```
Install.exe /quiet PublicPort=443 UAAHttpPort=9480 UAADatabasePort=9432 HistorianHttpPort=8070
HistorianDatabasePort=8432 AdminClientId=admin AdminClientSecret=secret123 UseExternalUaa=1
 ActiveUaaBaseUrl=https://TESTHIST4.test.com:443
UseExternalConfigHub=1 ExternalConfigHubMachineName=test2019-2 ConfigHubClientID=admin
ConfigHubClientSecret=secret321itc DataPath="E:\ProgramData\HistorianWebBasedClientsLogs"
 DoConfigHubNeedToInstall=0 RootDrive="E:\"
```

To install Web-based Clients with external Proficy Authentication and a local Configuration Hub, run the following command:

```
Install.exe /quiet PublicPort=443 UAAHttpPort=9480 UAADatabasePort=9432 HistorianHttpPort=8070
HistorianDatabasePort=8432 AdminClientId=admin AdminClientSecret=abc UseExternalUaa=1
 ActiveUaaBaseUrl=https://<extrenal UAA machine hostname>:443
ConfigHubContainerPort=5000 ContainerPort=4890 ConfigHubInstallFolder="C:\Program Files
 (x86)\GE\ConfigurationHub"
ConfigHubClientID=admin ConfigHubClientSecret=xyz
DataPath="C:\ProgramData\HistorianWebBasedClientsLogs"
```

To install Web-based Clients in a cluster environment with local Proficy Authentication and local Configuration Hub, run the following command:

```
Install.exe /quiet PublicPort=443 UAAHttpPort=9480 UAADatabasePort=9432
HistorianHttpPort=8070 HistorianDatabasePort=8432 AdminClientId=admin AdminClientSecret=abc
 ConfigHubContainerPort=5000
ContainerPort=4890 ConfigHubInstallFolder="C:\Program Files (x86)\GE\ConfigurationHub"
ConfigHubClientId=admin ConfigHubClientSecret=xyz DataPath="C:\ProgramData\HistorianWebBasedClientsLogs"
DestinationHistorian=<Historian server host name> IsClusterNode=1
PostgresBaseDir="E:\pgsql" ClusterFQDN="cluster.domain.com"
HistorianClusterMembershipPort=45564 HistorianClusterReceiverPort=4000
UAAClusterMembershipPort=45565 UAAClusterReceiverPort=4005 MulticastAddress=228.0.0.4
```

Web-based Clients are installed in the following folder: `<installation drive>:\Program Files \GE`, and the following registry paths are created:

- `HKEY_LOCAL_MACHINE\SOFTWARE\GE Digital`
- `HKEY_LOCAL_MACHINE\SOFTWARE\GE`

If you want to use Configuration Hub installed using other products such as iFIX, Plant Applications, and so on, set up authentication to point to the Proficy Authentication instance.

## Connect to a Remote Proficy Authentication Service

Provide the details of the external Proficy Authentication instance while installing Web-based Clients *(on page 154)*.

To host a Proficy Authentication service, you can use the same machine on which Web-based Clients are installed or a different one. This topic describes how to connect to a Proficy Authentication service that is set up on a machine different from the one on which you have installed Web-based Clients.

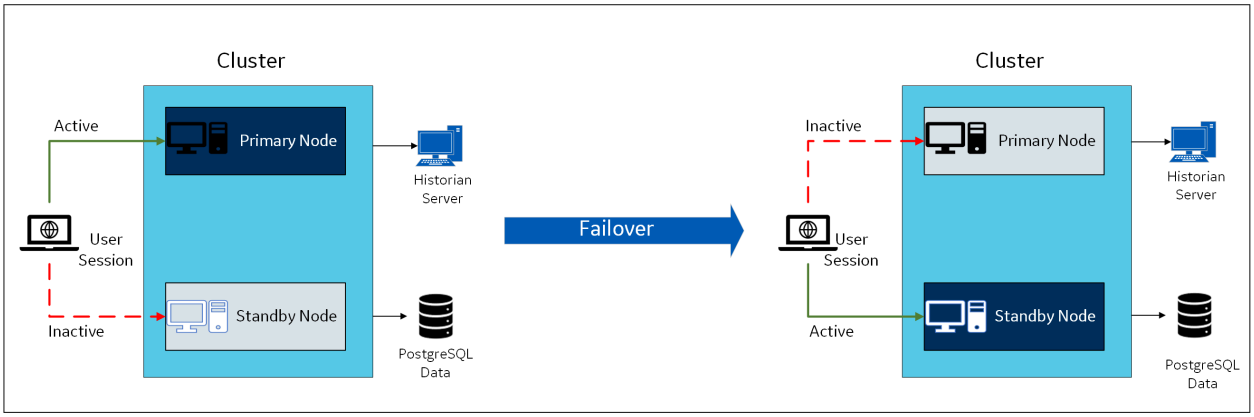
1. Access the Web Admin console.
2. Select **Not Secure**, and then select **Certificate**.
3. Select the root CA certificate in the **Certificate Path** section.
   The **Certificate Export Wizard** window appears.
4. Select the **Base-64 encoded X.509 (.CER)** format.
5. Install the certificate in Trusted Root Certification Authorities store.
6. Rename the certificate file from `.cer` to `.pem`.
7. Access Certificate Management Tool.
8. Access the **External Trust** section and import the renamed certificate.
9. Select **No** when prompted to restart GeOphubMasterStarter.
10. Restart the **Proficy Historian Tomcat** service.
11. Reopen the browser.

Map the user groups of the remote Proficy Authentication service with the Historian Proficy Authentication group

## Set Up High Availability of Web-based Clients

1. Create a shared drive on your network that all the nodes in the cluster can access, and create a folder in that drive.
2. On each node that you want to add to the cluster:
   a. Install the Failover Clustering feature.
   b. If you want to use an existing Proficy Authentication instance, ensure that all the cluster nodes point to the same Proficy Authentication instance. Note that for *all* the cluster nodes, the Proficy Authentication credentials of the node on which you installed Proficy Authentication *last* will be considered.

In a cluster environment, multiple servers are installed, which share the same data. Each of these servers is called a node. One of them acts as the primary node, while the others are standby nodes. If the primary node is down, one of the standby nodes is used.




When you install Web-based Clients in a cluster environment, the web servers are added to the cluster. You can then achieve high availability of connection between the Historian server and the client applications.

For example, if Configuration Hub on the primary node is unable to connect to the Historian server, the user session on the standby node is activated. Therefore, you will still be able to connect to the Historian server using Configuration Hub installed on the standby node.

The following services are shared between the primary and standby nodes in a cluster:

- Historian Indexing Service
- Proficy Historian PostgreSQL Database

Historian works with Microsoft Failover Cluster Manager to ensure high availability of Web-based Clients. Using Failover Cluster Manager, you must add these services to the cluster.

1. Access the primary node of the cluster.
2. Create a failover cluster.
3. Add a storage to the failover cluster.
4. Select **Roles > Create Empty Role**.
   A role is created.
5. Add a client access point to the role:
   a. Select the role.
   b. In the **Actions** section, select **Add Resource > Client Access Point**.
   c. Follow the on-screen instructions to add a client access point to the role.
6. Add a storage to the role:

a. Select the role.
b. In the **Actions** section, select **Add Storage**.
c. Follow the on-screen instructions to add the storage that you have created in step 3. You can use a storage only once.

7. Ensure that your network is enabled for multicast traffic. To do so, run the following command:

```
netsh <interface name> <IP address> show joins
```

A list of IP multicast groups that have been joined through an interface appears. If you do not specify an interface name, a list of multicast groups for all interfaces appears.

8. Install Web-based Clients *(on page 155)*. During the installation, select the **Cluster Node** check box, and provide the details.
9. Import the Proficy Authentication certificate *(on page 182)* into all the cluster nodes. Copy the certificate in the following path from any node in the cluster and paste it in the same folder in all the other nodes: `C:\Program Files\GE\Operations Hub\httpd\conf\cert`.
10. Add resources to the role:

a. Select the role.

b. In the **Actions** section, select **Add Resource > Generic Service**.
The **New Resource Wizard** window appears.

c. In the list of resources, select **Historian Indexing Service**, and then follow the on-screen instructions to add the service.

11. Perform the previous step to add the Proficy Historian PostgreSQL Database resource as well.
12. Add the following dependencies for each of these resources:

a. Double-click a resource.
The **<resource name> Properties** window appears.

b. Select **Dependencies**.

c. Select **Insert**, and add dependencies for each resource as described in the following table, using the AND operation.

| Resource Name | Dependencies |
|---|---|
| Proficy Historian ProstgreSQL Database | ▪ IP Address<br>▪ Storage<br>▪ The network name |

| Resource Name | Dependencies |
|---|---|
| Historian Indexing Service | ▪ IP Address<br>▪ Storage<br>▪ The network name<br>▪ Proficy Historian ProstgreSQL Database |

13. Select the role, and then in the **Actions** section, select **Start Role**.
    When you later install Web-based Clients and provide the cluster details, Web-based Clients will be part of the cluster, thus achieving high availability.

1. Restart the following services on all the cluster nodes:
    - Historian Indexing Service
    - Proficy Historian PostgreSQL Database
2. On the machine on which you have installed the Historian server, update the URI of the following registry key to point to the cluster FQDN: `HKEY_LOCAL_MACHINE\SOFTWARE\Intellution, Inc\iHistorian\SecurityProvider\OAuth2`.

## Upgrade Web-based Clients

- When you upgrade, Web-based Clients and associated data will be lost. Therefore, back up Web-based Clients and associated data using the `uaa_config_tool` utility provided in the `Utilities` folder of the ISO package. For information, refer to Migrate User Authentication Data from Historian to Common Proficy Authentication Service *(on page 248)*.

  > **Tip:**
  > After installation, `uaa_config_tool` is available in the following folder as well:
  > `<installation drive>\Program Files\GE Digital\Historian Config`

- You cannot upgrade Web-based Clients from a version earlier than 8.0. This is because, starting 8.0, Web-based Clients are installed separately (not as part of the Historian server installation). Therefore, you must do either of the following:
    - Install Web-based Clients on a new machine.
    - Uninstall the Historian server (remember to back up the Proficy Authentication using the UAA_config_tool utility), and then install Web-based Clients.
- If the machine name has changed, you must uninstall and reinstall Web-based Clients.

- If you want to use a different Proficy Authentication server from the previous one, you must manually migrate the Proficy Authentication data to new Proficy Authentication server using the uaa_config_tool utility.
- If you want to switch from using a local Proficy Authentication to using an external Proficy Authentication (or vice versa), you must manually change the Proficy Authentication details.

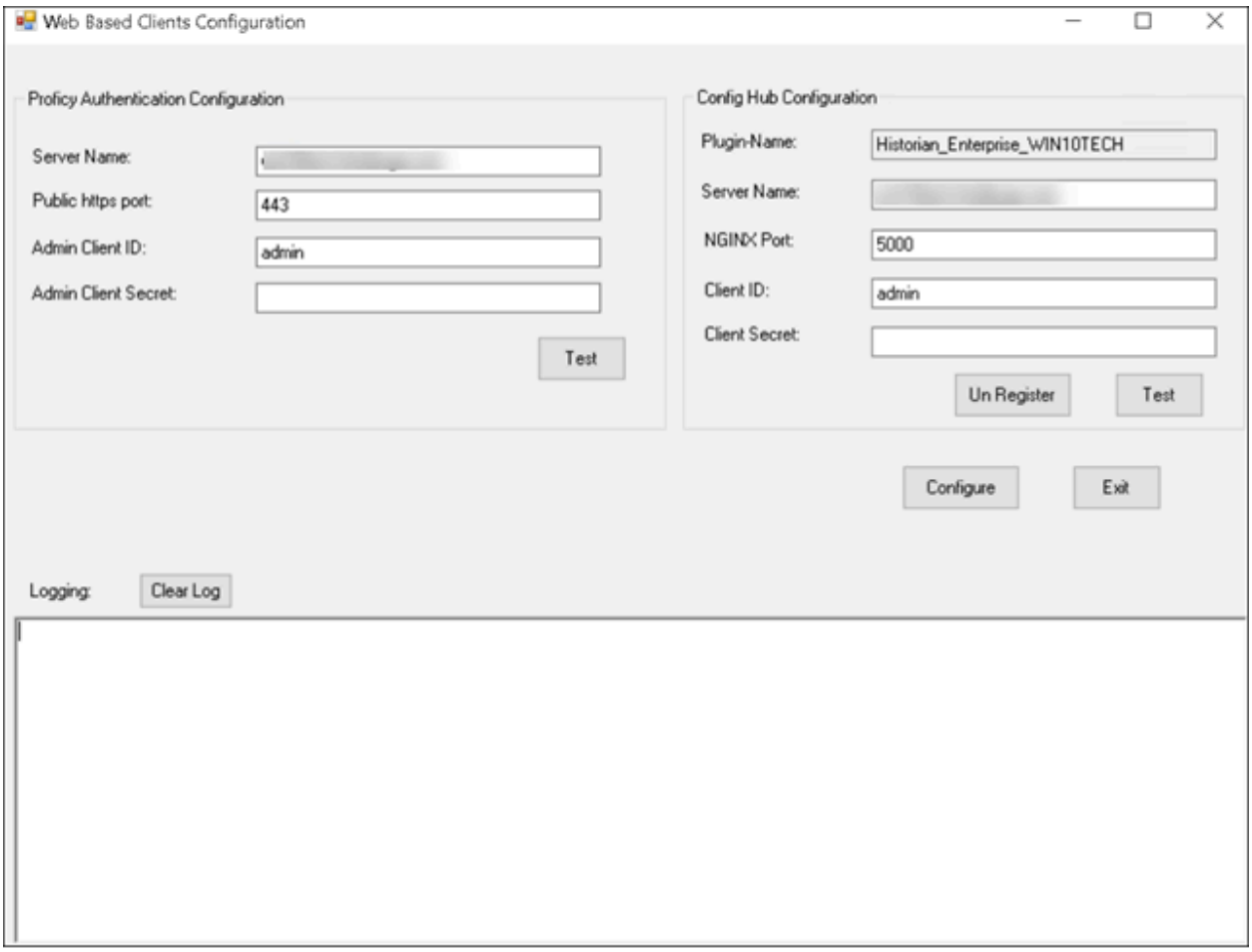
1. Install Web-based Clients *(on page 154)*.
   Web-based Clients will be upgraded to the latest version.
2. Clear the browser cache.
   You can now access Web-based Clients.

## Configure Web-based Clients

You can configure the following settings for Web-based Clients:

- Reconfigure Proficy Authentication to point to a different Proficy Authentication server.
- Reconfigure the same Proficy Authentication instance to resolve any issues with login.
- Re-register Configuration Hub to resolve any issues.
- Unregister Configuration Hub, and register another one.

To perform these tasks, Historian provides a utility called Web Based Clients Configuration. It is installed during Web-based Clients installation.




To run this utility, run the `Web_Clients_Configuration_Tool.exe` file. By default, it is located in the following folder: `C:\Program Files\GE Digital\Historian Config`

## Remote Management Agents

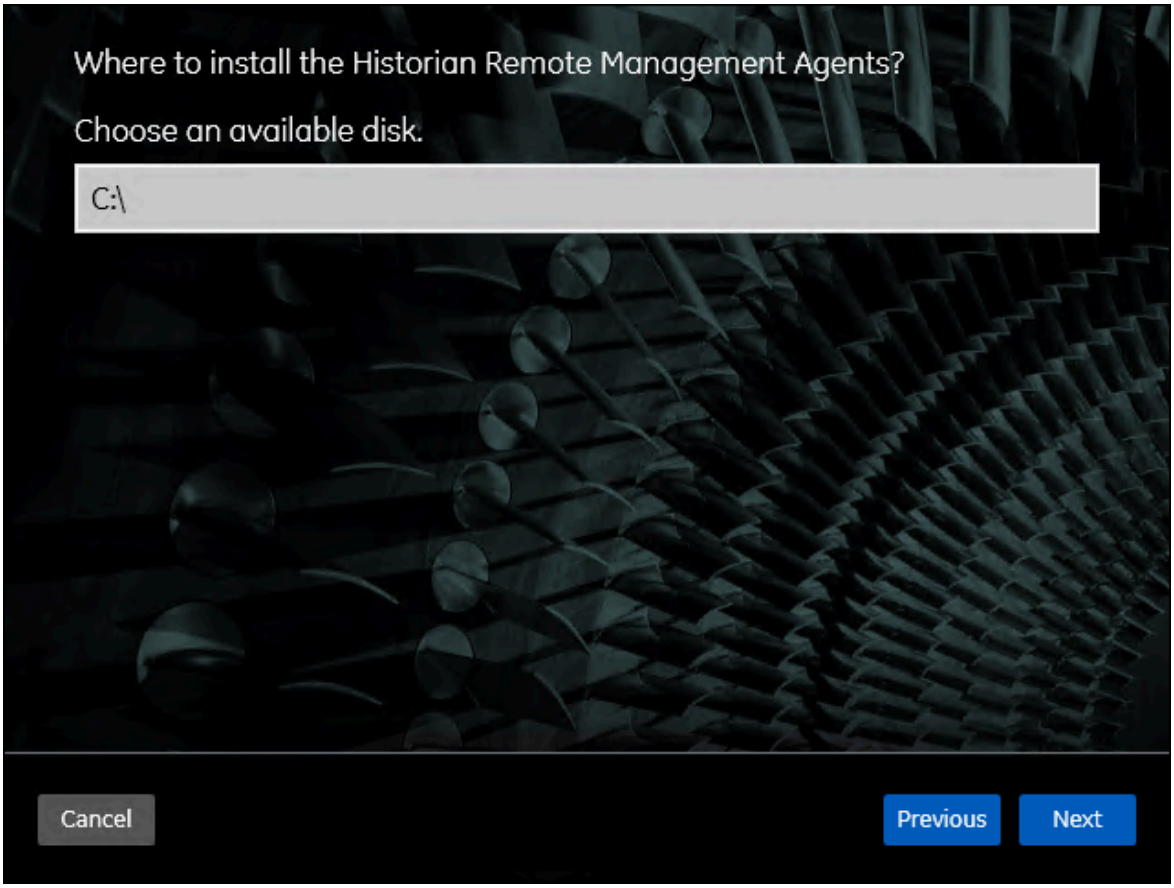
## Install Remote Management Agent Using the Installer

Ensure that all the collectors that you want to manage remotely are in the running state.

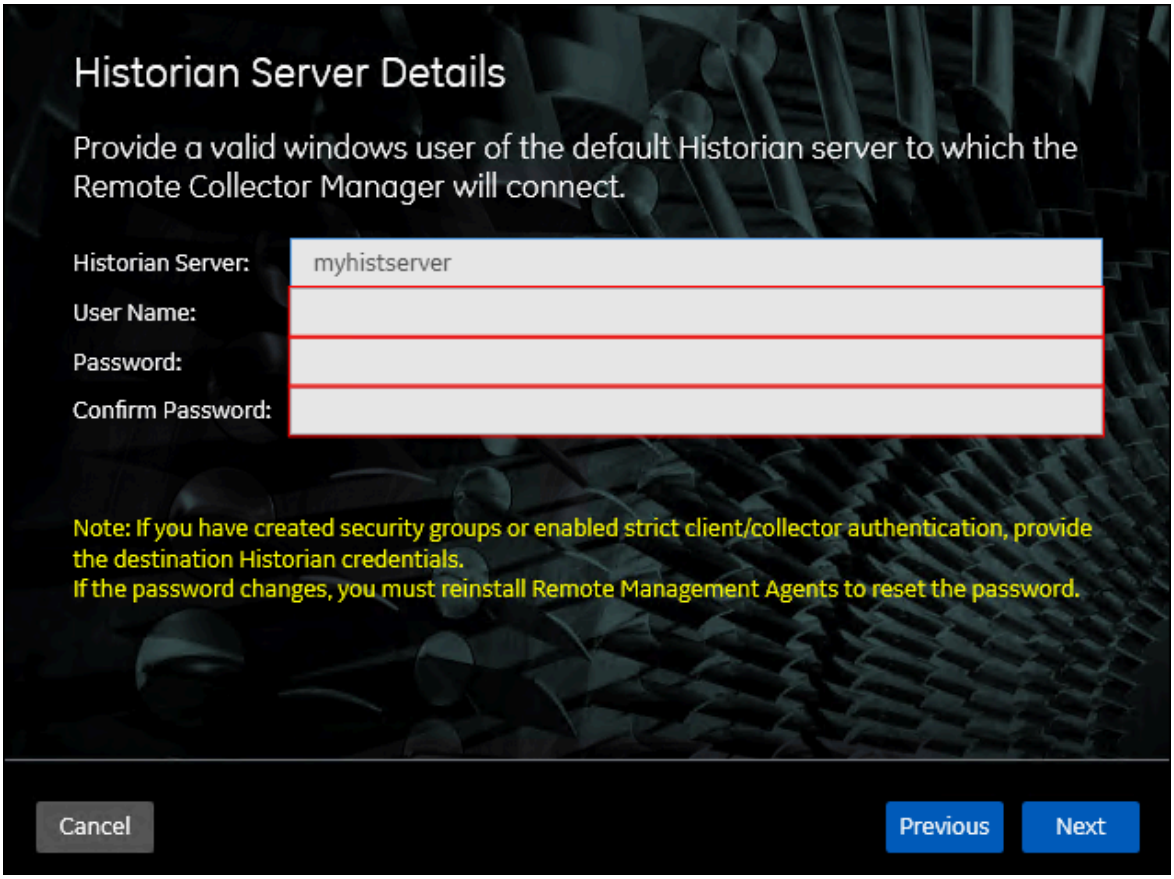
If the collectors that you have installed are earlier than version 9.0, you must install Remote Management Agent on each machine on which the collectors that you want to manage are installed. For collectors version 9.0 or later, Remote Management Agent are automatically installed when you install collectors.

This topic describes how to install Remote Management Agent using the installer. You can also install them at a command prompt *(on page 190)*.

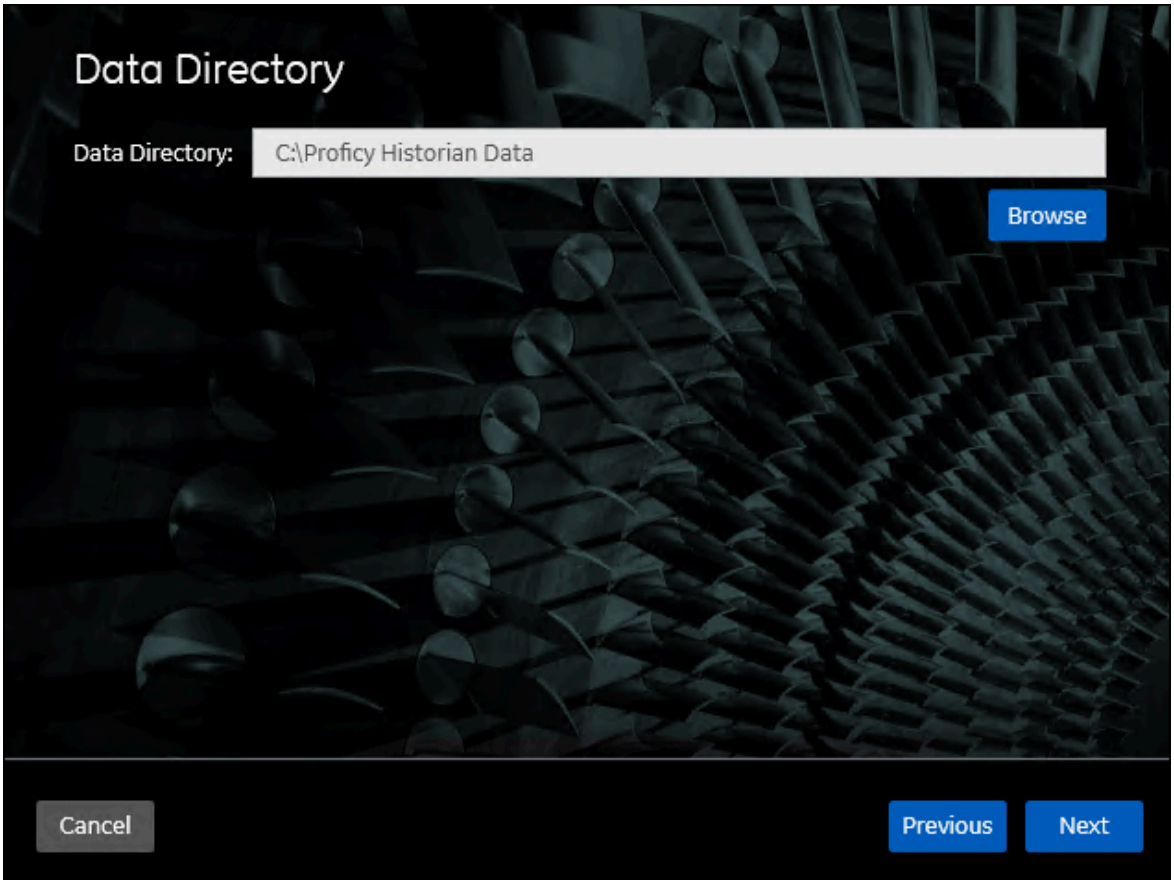
1. Run the `InstallLauncher.exe` file.
2. Select **Install Remote Management Agents**.
   The welcome page appears.
3. Select **Next**.
   The license agreement appears.
4. Select the **Accept** check box, and then select **Next**.
   The default installation drive appears.




5. If needed, modify the installation drive, and then select **Next**.
   The destination Historian server page appears.




6. Enter the details of the Historian server to which Remote Management Agent will connect, and then select **Next**.
   The **Data Directory** page appears.




7. As needed, modify the location of the data directory, or leave the default value, and then select **Next**.
   The **You are ready to install** page appears.
8. Select **Install**.

- Remote Collector Management is installed on your machine.
- A folder named `Historian Remote Management Agents` is created in the `GE Digital` folder in the installation location that you specified.
- Remote Collector Management is running, and a .shw file is created in the log folder. This file contains the details of the collectors that are running on the machine.
- For each collector that you manage using Remote Collector Management, a new entry named ServiceName is created in the collector registry. If the ServiceName key is not created or updated incorrectly, refer to Troubleshooting Remote Collector Management Issues *(on page 820)*.

## Install Remote Management Agent at a Command Prompt

Ensure that all the collectors that you want to manage remotely are in the running state.

If the collectors that you have installed are earlier than version 9.0, you must install Remote Management Agent on each machine on which the collectors that you want to manage are installed. For collectors version 9.0 or later, Remote Management Agent are automatically installed when you install collectors.

This topic describes how to install Remote Management Agent at a command prompt. You can also install them using the installer *(on page 187)*.

1. Access the command prompt, and navigate to the `RMA` folder in the install media.
2. Run the following command, replacing the values in angular brackets with the appropriate values:

```
HistorianRMA_Install.exe -s RootDrive=<installation drive> DestinationServerName=<Destination Historian server name> UserName1=<Windows username> Password=<Windows password> DataPath="C:\Proficy Historian Data\LogFiles"
```

```
HistorianRMA_Install.exe -s RootDrive=C:\ UserName1=Administrator Password=AdminPassword DestinationServerName=VMHISTWEBAUTO DataPath="C:\Proficy Historian Data\LogFiles"
```

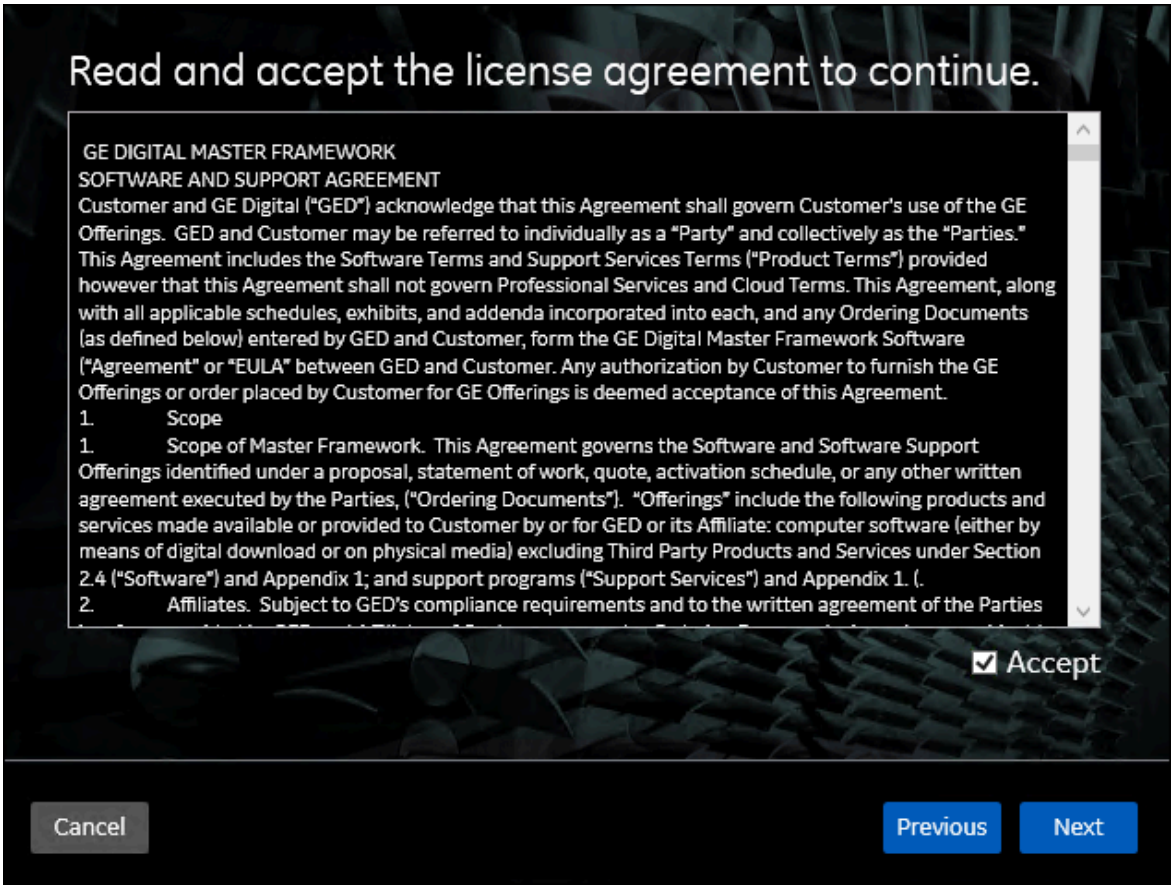
- Remote Collector Management is installed on your machine.
- A folder named `Historian Remote Management Agents` is created in the `GE Digital` folder in the installation location that you specified.
- Remote Collector Management is running, and a .shw file is created in the log folder. This file contains the details of the collectors that are running on the machine.
- For each collector that you manage using Remote Collector Management, a new entry named ServiceName is created in the collector registry. If the ServiceName key is not created or updated incorrectly, refer to Troubleshooting Remote Collector Management Issues *(on page 820)*.

## Install the OPC UA HDA Server

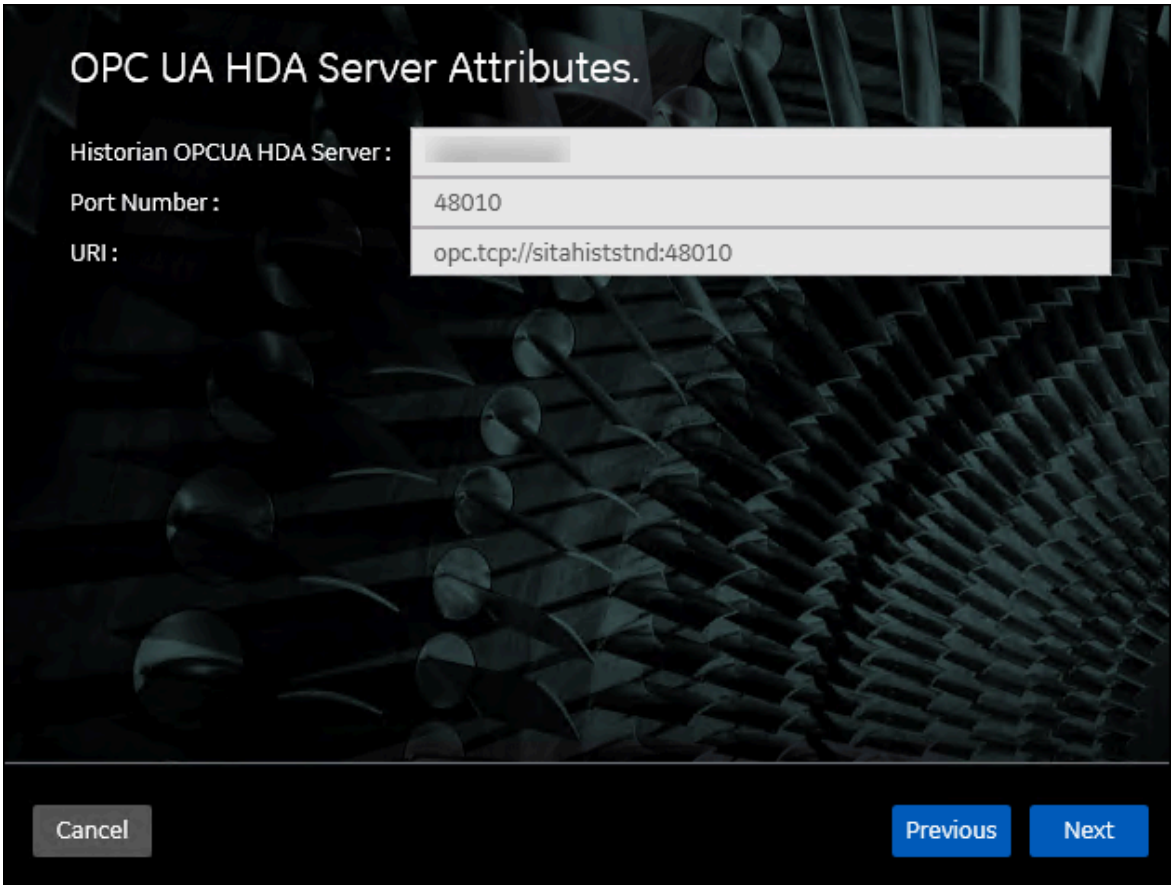
Install Historian *(on page 104)*.

**Note:**
You can install Historian and the OPC UA HDA server on the same machine or on different machines.

1. Run the Historian installer.
2. Select **Install Historian OPC UA HDA Server**.
   The welcome page appears.
3. Select **Next**.
   The license agreement appears.



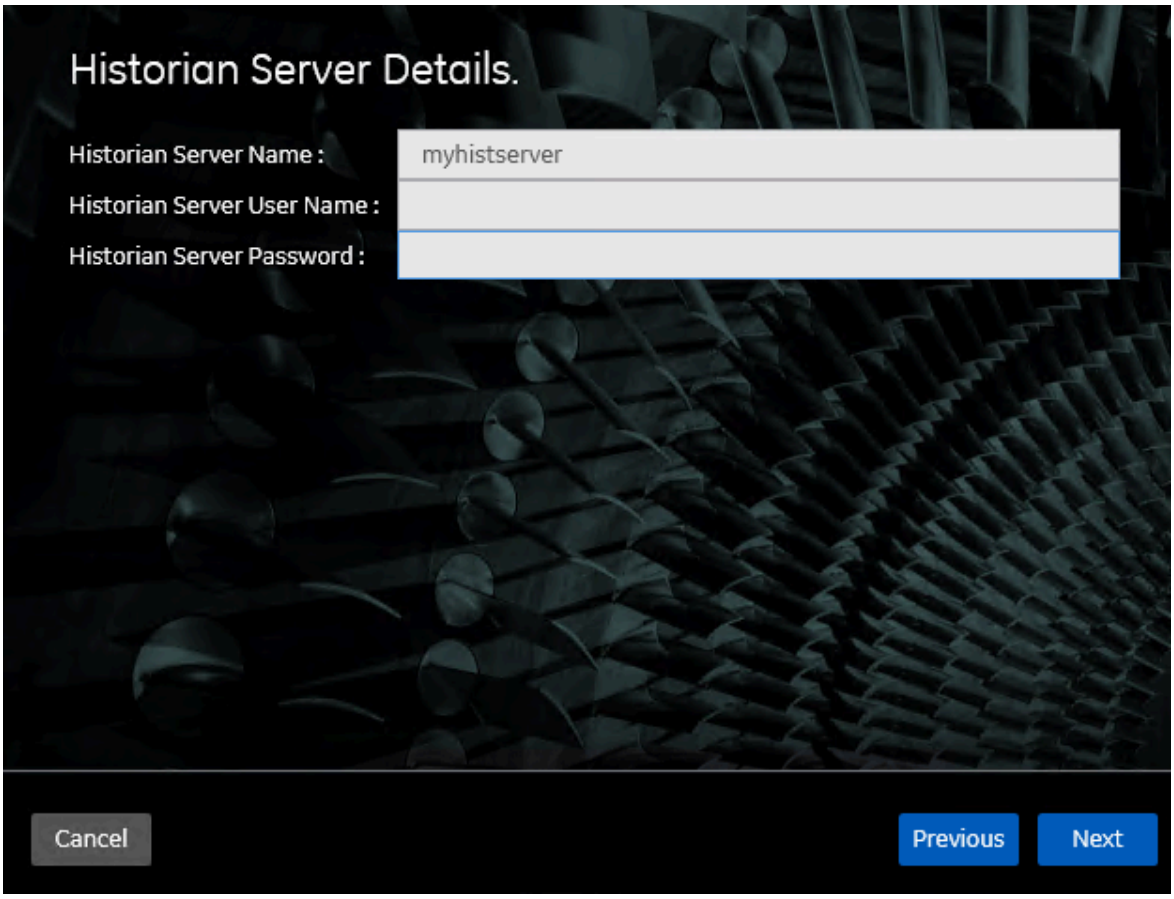

4. Select the **Accept** check box, and then select **Next**.
   The **Where to install the Historian OPC UA HDA Server** page appears, asking you to select the installation drive.
5. Select the installation drive, and then select **Next**. You can retain the default one, or choose a different one.
   The **OPC UA HDA Server Attributes** page appears.




6. Provide values as described in the following table, and then select **Next**.

| Field | Description |
|---|---|
| **Historian OPCUA HDA Server** | Enter the host name or the IP address of the machine on which you want to install the OPC UA HDA server. By default, the local host name appears. |
| **Port Number** | Enter the port number that you want the OPC UA HDA server to use. |
| **URI** | The URI to access the OPC UA HDA server. This field is disabled and populated with a value in the following format: opc.tcp://*<host name>*:*<port number>*, where *<host name>* and *<port number>* are the values that you have entered in the preceding fields. |

The **Historian Server Details** page appears.




7. Provide values as described in the following table, and then select **Next**.

| Field | Description |
|---|---|
| **Historian Server Name** | Enter the name of the Historian server that you want to connect to the OPC UA HDA server. |
| **Historian Server User Name** | Enter the username of the Historian server. |
| **Historian Server Password** | Enter the password of the Historian server. |

The **You are ready to install** page appears.

8. Select **Install**.

The Historian OPC UA HDA server is installed. Reboot the machine when prompted to do so.

- If you have installed the OPC UA HDA server on a remote machine, enable the firewall.
- Install an OPC UA client.
- Configure the OPC UA HDA server.

# The Excel Add-In for Historian

## Install the Historian Excel Add-in Using the Installer

Install one of the following 32-bit or 64-bit Microsoft® Excel® applications:

- Microsoft® Excel® 2019
- Microsoft® Excel® 2016
- Microsoft® Excel® 2013
- Microsoft® Excel® 2010

You can install Excel Add-In separately or during Client Tools installation. This topic describes how to install Excel Add-In separately using the installer. You can also install it at a command prompt *(on page 195)*. However, if installing via a command prompt, do not install the Excel Add-In on the machine that you have installed the Historian Administrator or data archiver.

1. Run the `InstallLauncher.exe` file.
2. Select **Install Excel Add-in for Historian**.
   The installer runs through the installation steps.

   > **Note:**
   > If using certain versions of Windows (like Windows 10 or Windows 2019), you may receive an error message, stating that some of the DLL files are not registered. You can ignore these messages.

3. When prompted to reboot your system, select **Yes**.

Excel Add-In is installed.

Activate Excel Add-In *(on page 197)*.

## Install the Historian Excel Add-in at a Command Prompt

1. Install one of the following 32-bit or 64-bit Microsoft® Excel® applications:
   - Microsoft® Excel® 2019
   - Microsoft® Excel® 2016
   - Microsoft® Excel® 2013
   - Microsoft® Excel® 2010
2. Install Excel Add-in using the installer *(on page 195)* on your machine:

a. Open the Command Prompt as an Administrator. (From the Start menu, right-click the Command Prompt app, and select Run as Administrator.)
b. On the Historian install media, navigate to the Excel folder.
c. Run the following command line: setup.exe -r.
d. When the install runs, select the Excel option and leave the other options greyed out as they cannot be removed.

   When you run this install, a template file named setup.iss is created at `C:\Windows`. This file stores the installation options that you have provided during the installation.

   After completing the installation, you can then use this `.iss` file to install Excel Add-in at a command prompt on other machines.

   On the other Client machine where you want to install the Excel Add-in, place a copy of the **setup.iss** on the local directory and open a Command Prompt (Run as Administrator). Run the silent install with following command:

   ```
   setup.exe /s
   ```

   If your iss file is not called **setup.iss** or is in different directory than setup.exe you need to add the flag: /f1. For example:

   ```
   setup.exe /s /sms /f1"C:\Temp\setup.iss
   ```

   Wait for few minutes (probably 3-5 minutes) for the install to complete.

You can install Excel Add-In separately or during Client Tools installation. However, do not install Excel Add-In on the machine on which you have installed Historian Administrator or data archiver.

1. Copy the `setup.iss` file to each machine on which you want to install Excel Add-in at a command prompt.
2. In the folder that contains the `setup.iss` file, run the following command: `setup.exe /s /sms`
   The installer runs through the installation steps.

   > **Note:**
   > If using certain versions of Windows (like Windows 10 or Windows 2019), you may receive an error message, stating that some of the DLL files are not registered. You can ignore these messages.

3. When prompted to reboot your system, select **Yes**.

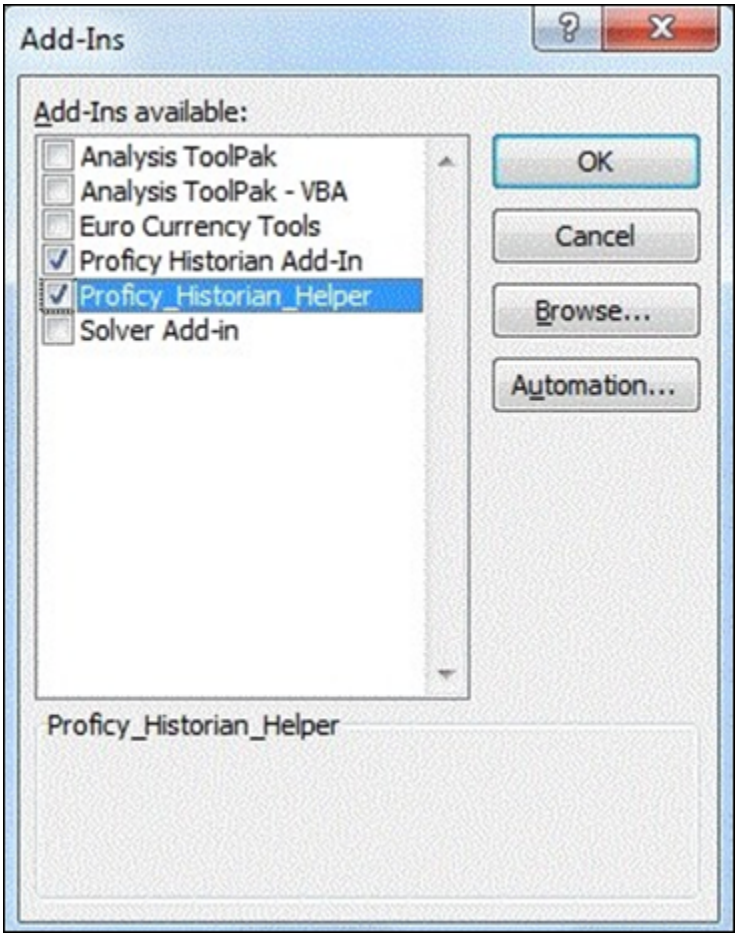
Excel Add-In is installed.

Activate Excel Add-In *(on page 197)*.

## Activate Excel Add-In

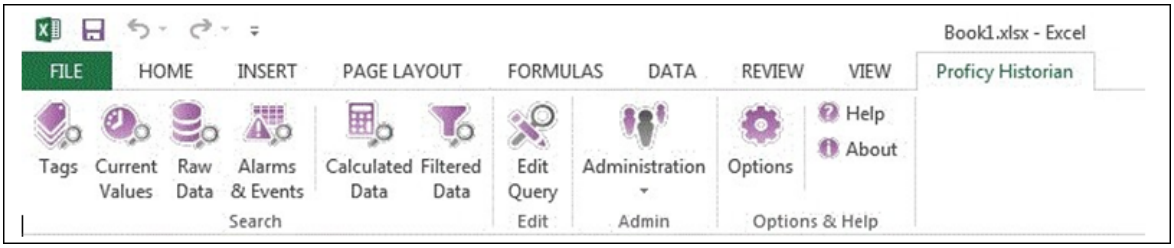
Install Excel Add-In *(on page 195)*.

1. Open a new Microsoft Excel worksheet.
2. Select **File > Options**.
   The **Excel Options** window appears.
3. Select **Add-Ins**.
4. In the **Manage** box, select **Excel Add-ins**, and then select **Go**.
   The **Add-Ins** window appears.




5. Select the **Proficy Historian Add-In** and **Proficy_Historian_Helper** check boxes, and then select **OK**.
   If the **Proficy Historian Add-In** and **Proficy_Historian_Helper** check boxes do not appear, select **Browse** to locate the `Historian.xla` file for the check boxes to appear. This file is created if you have installed Microsoft Excel after installing Excel Add-In. By default, the `Historian.xla`

file is located in the `C:\Program Files\Proficy\Historian or C:\Program Files (x86)\Proficy\Historian` folder.
Excel Add-In is now ready to use and the **Proficy Historian** menu is now available in the Microsoft Excel toolbar.




## Software Requirements

The following components are required to use Excel Add-in for Operations Hub:

| Component | Version | Description |
|---|---|---|
| Operations Hub | 2.0 and above. | If you have purchased the standard or enterprise license of Historian, you receive a no-cost license for Operations Hub, which enables you to:<br>• Access to the Historian Analysis run-time application, which is an in-built HTML5 application in Operations Hub.<br>• Perform advanced trend analysis, including inserting annotations.<br>• Define an asset model including tag mapping. |
| Microsoft Excel | 2016 and 2019 (32 bit or 64 bit) | |
| Historian REST APIs | | Historian REST APIs are required to integrate between Historian and Operations Hub. Historian REST APIs are installed automat- |

| Component | Version | Description |
|---|---|---|
| | | ically when you install Historian Web-based Clients *(on page 155)*. |

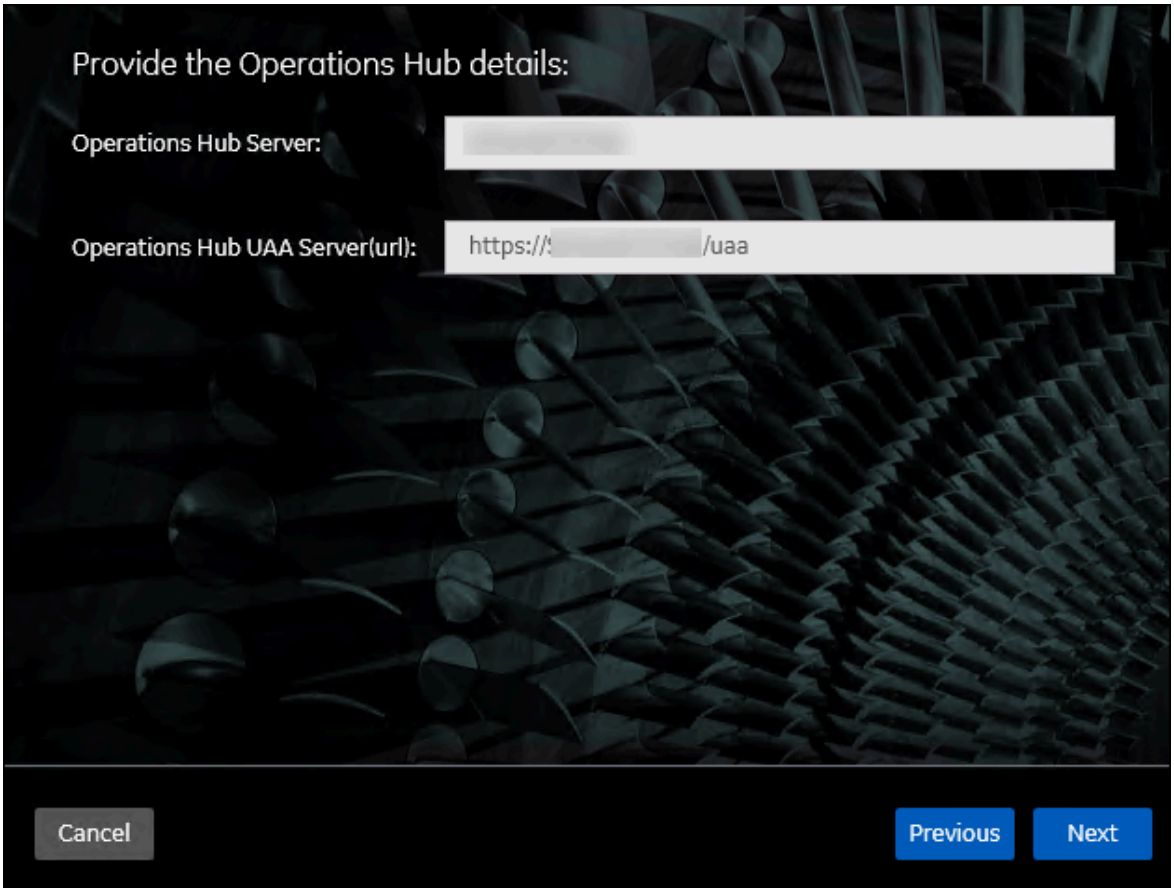
## Install Excel Add-In for Operations Hub

Install the Historian server *(on page 104)* and other software requirements *(on page 198)*.

1. Run the `InstallLauncher.exe` file.
2. Select **Install Excel Add-in for Operations Hub**.
   The welcome page appears.
3. Select **Next**.
4. Read and accept the license agreement, and then select **Next**.
5. Select the available disk to install the Excel Add-in for Operations Hub, and then select **Next**.

   **Note:**
   We recommend that you select the drive where Microsoft Excel is installed.

6. Provide the details of Operations Hub, and then select **Next**.




The **You are ready to install** page appears.

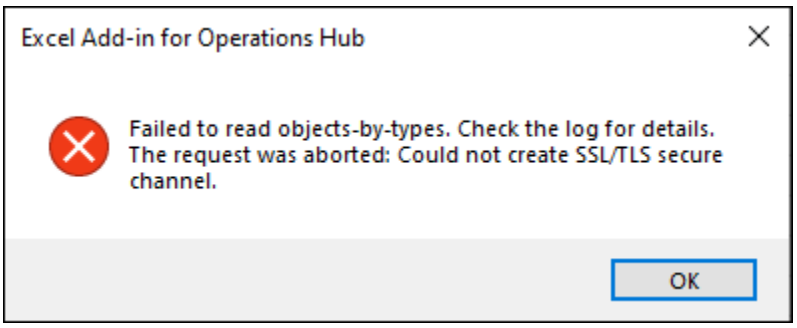
7. Select **Install**.
   Excel Add-In for Operations Hub is installed.

Copy/export the issuer certificate *(on page 200)*, and then install/import it *(on page 201)*.

## Copy or Export the Issuer Certificate on Server

Install Excel Add-In for Operations Hub *(on page 199)*.

This topic describes how to copy or export the issuer certificate on Server. If this is not done, you will get an error while querying data as shown in the image below.




1. Navigate to the machine where Operations Hub is installed.
2. Select **Site Information (Not secure)**.
3. Select **Certificate (invalid)**.
   The **Certificate** window appears.
4. Select **Certificate Path**.
5. Select the Root CA certificate.
6. Select **Details**.
7. Select **Copy to file**.
   The **Certificate Export Wizard** window appears.
8. Select **DER encoded binary X.509(.CER)** format and select **Next**.
9. Select **Browse** to save the certificate file at desired location.
10. Complete the certificate export.

Install or import the certificate *(on page 201)*.

## Install/Import the Issuer Certificate

Copy or export the issuer certificate *(on page 200)* on the machine on which Excel Add-In for Operations Hub is installed.

1. Right-click the certificate, and then select **Install Certificate**.
   The **Certificate Import Wizard** page appears
2. Select **Local Machine**, and then, select **Next**.
3. Select **Place all certificates in the following store**.
4. Select **Trusted Root certification Authorities**, and then select **OK**.
5. Select **Next**, and then select **Finish**.
   The certificate is imported.

Configure the Operations Hub server *(on page 202)*.

## Connect to Operations Hub

To query a model defined in Operations Hub, you must first connect to the Operations Hub server. You will then receive a token from the server, which will be used for authentication.

1. Select **Configuration** menu in Admin.
   The **Operations Hub Configuration** window appears.
2. Provide values as described in the following table.

| Field | Details |
|---|---|
| Operations Hub Server | The Operations Hub server name to which you want to connect and get the data. |
| Operations Hub Proficy Authentication Server (url) | The URL of the Proficy Authentication service of Operations Hub.<br><br>Example: https://<ophubservername>/uaa |

> **Note:**
> The **Token Status** field indicates the status of the connection with Operations Hub server.

3. Select **Connect**.
   The login page appears.
4. Provide the **User Identifier** and **Password** to connect to Operations Hub.
5. Select **Open UaaAuthSchemeHandler**.
   Operations Hub Server to which you are connected and the status of the token appears.
6. Select **Save** to save the Operations Hub server details. The configuration will be retained and used when you open excel add-in again.

## Install the Historian ETL Tools

- If you want to use the Historian Extract, Transform, and Load (ETL) tools to transfer data from a PI Historian server, install the PI SDK package.
- If you want to use the ETL tools to transfer data from an eDNA server, copy the following eDNA binaries to the `<installation drive>\Program Files\GE Digital\Historian ETL eDNA Extract` folder:

    - `EzDnaApi.dll`
    - `EzDNAApiNet.dll`

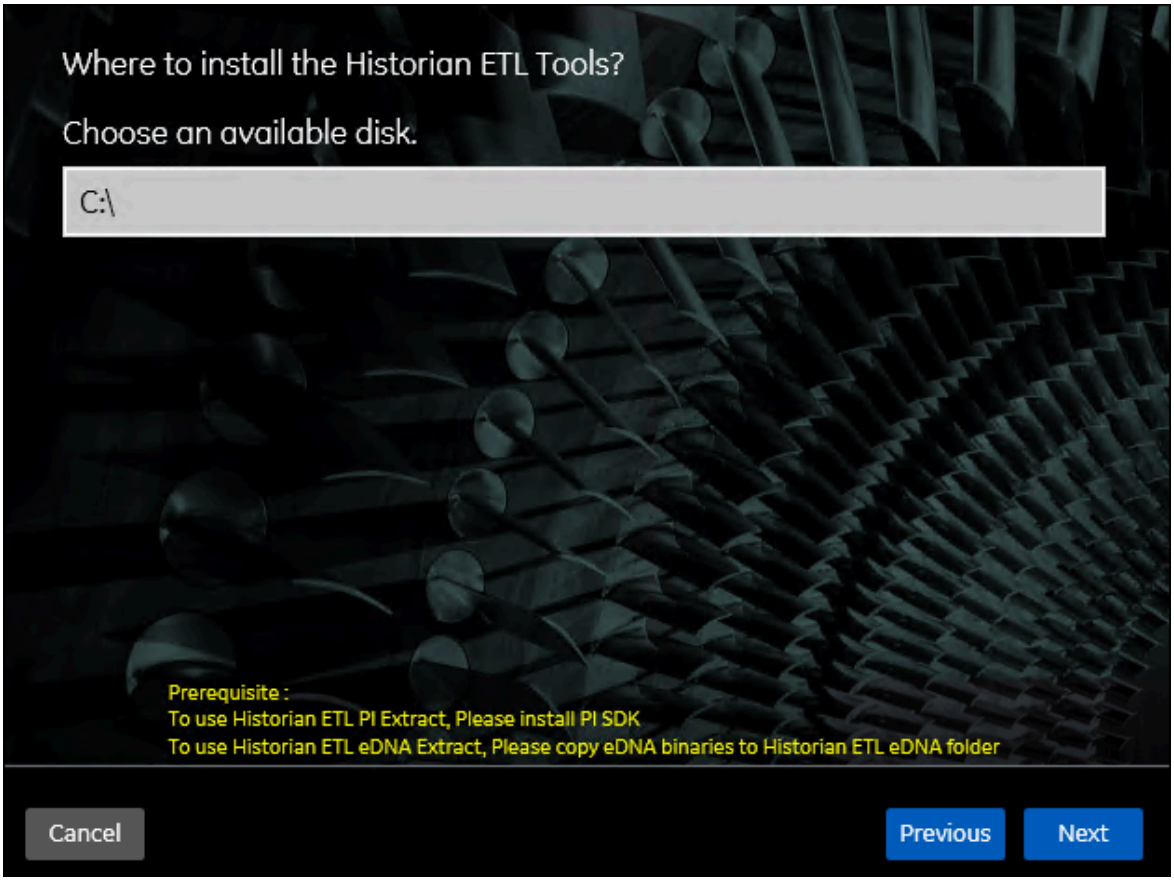
  By default, these files are available in the following folder on the machine on which the eDNA server is installed: `C:\Program Files(x86)\eDNA`

Installing ETL installs the following tools:

- The Extract tool
- The Transform tool
- The Load tool

This topic describes how to install ETL to extract, transform, and load data from an onsite Historian machine to the destination Historian server. You must install Historian ETL on both the onsite Historian machine and the destination Historian server (that is, the source and destination machines for data transfer).

1. Run the `InstallLauncher.exe` file.
2. Select **Install Historian ETL Tools**.
   The welcome page appears.
3. Select **Next**.
   The license agreement appears.
4. Select the **Accept** check box, and then select **Next**.
   The default installation drive appears.




5. If required, modify the installation drive for Historian ETL, and then select **Next**.
   A message appears, stating that you are ready to install ETL.
6. Select **Install**.

The Historian ETL tools are installed on your machine.

- The following folders are created in the `<installation drive>/Program Files/GE Digital` folder:
  - `Historian ETL eDNA Extract`
  - `Historian ETL Extract`
  - `Historian ETL Load`
  - `Historian ETL ODBC Extract`
  - `Historian ETL PI Extract`
  - `Historian ETL Transform`
- The following services are installed:
  - Historian ETL eDNA Extract
  - Historian ETL Extract
  - Historian ETL ODBC Extract_x64

    - Historian ETL ODBC Extract_x86
    - Historian ETL Load
    - Historian ETL PI Extract
- The following registry paths are created:
    - `HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\GE Digital\Historian ETL eDNA Extract`
    - `HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\GE Digital\Historian ETL ODBC Extract`
    - `HKEY_LOCAL_MACHINE\SOFTWARE\GE Digital\Historian ETL Extract`
    - `HKEY_LOCAL_MACHINE\SOFTWARE\GE Digital\Historian ETL ODBC Extract`
    - `HKEY_LOCAL_MACHINE\SOFTWARE\GE Digital\Historian ETL PI Extract`
    - `HKEY_LOCAL_MACHINE\SOFTWARE\GE Digital\Historian ETL Load`

1. If you want to extract data from an eDNA server, copy the `Customdna.ini` file in the eDNA server into the `C:\Windows` folder on the machine on which you have installed ETL.
2. Extract data from an eDNA server *(on page 2910)*, ODBC data source *(on page 2924)*, Proficy Historian *(on page 2938)*, or PI Historian *(on page 2948)*.

## About Installing Help

Historian documentation is available both online and offline. This topic describes how to install the offline Help documentation. Online Help is available here: https://www.ge.com/digital/documentation/historian/

You can install Help using a GUI-based installer or at the command prompt.

## Install Standalone Help Using the Installer

This topic describes how to install Help using the installer. You can also install Help at a command prompt *(on page 206)*.

1. Run the `InstallLauncher.exe` file.
2. Select **Install Standalone Help**.
   The welcome page appears.
3. Select **Next**.
   The license agreement appears.
4. Select the **Accept** check box to accept the license agreement, and select **Next**.
5. If needed, change the installation drive, and then select **Next**.
6. If needed, change the port number for the NodeJS server to run. This step is required if the default port number is not available.

7. Select **Next**.
8. Select **Install**.
   The Help is installed.
9. Select **Next**.

The standalone Help is installed in the following folder: `<installation drive>:\Program Files \Proficy\Proficy Historian\ProficyDoc`. You can access the Help from any of the Historian applications or by accessing the `index.html` file.

### Install Help at a Command Prompt

This topic describes how to install Help at a command prompt. You can also install Help using the installer *(on page 205)*.

1. Navigate to the `Help` folder.
2. If you want to use the default installation drive (C:/) and port number (7070), run the following command:

   ```
   Help_Install.exe -s
   ```

   Otherwise, run the following command:

   ```
   Help_Install.exe -s RootDrive=<installation drive> PortNumber=<port number>
   ```

The Help is installed. You can access the Help from any of the Historian applications or by accessing the `index.html` file. By default, this file is available in the `C:\Program Files (x86)\GE Digital \Historian Help` folder.

## Configuring Certificate-based Security to use MTLS

### Overview of the Certificate-based Security

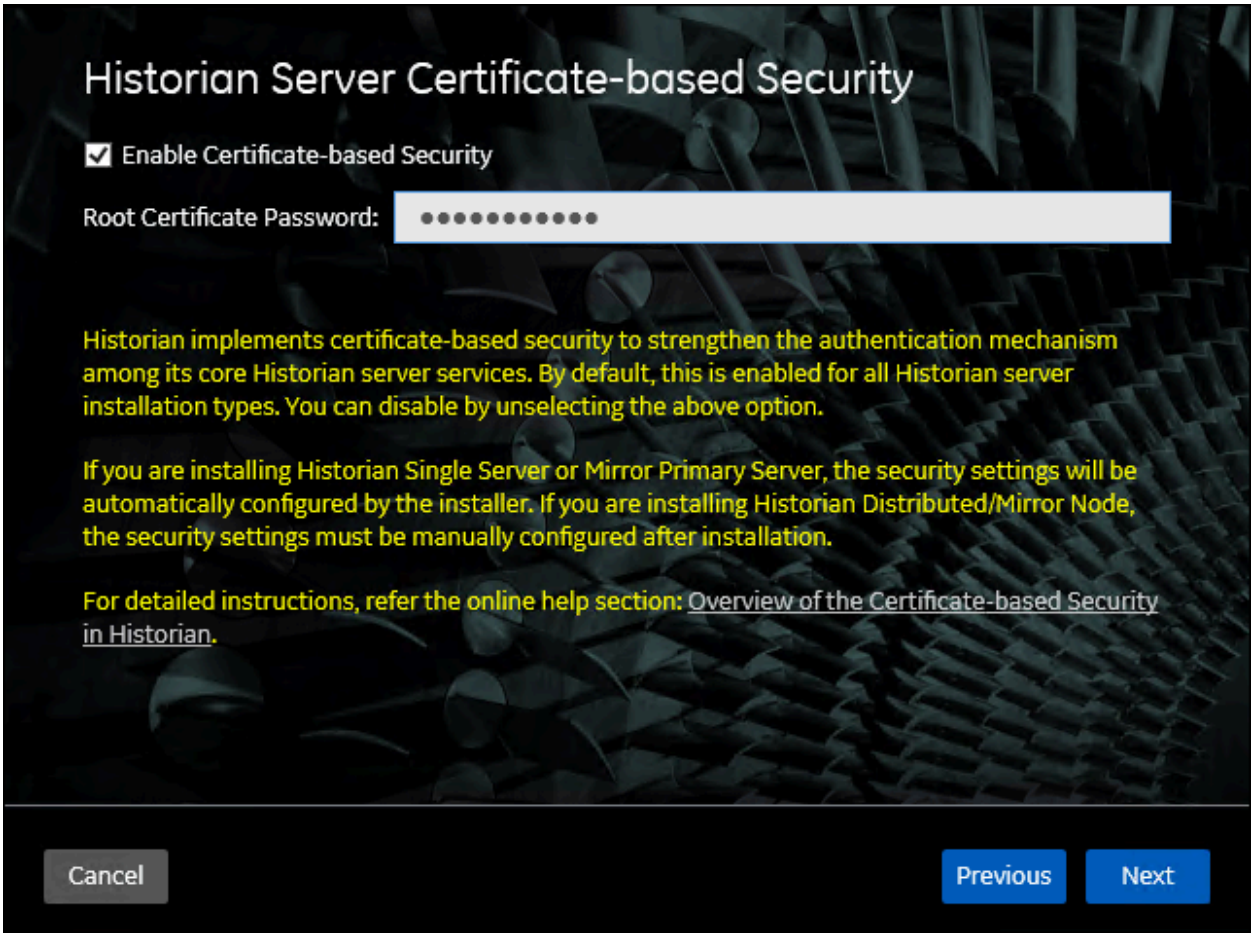
Historian uses the Mutual Transport Layer Security (MTLS) protocol along with certificate-based security to strengthen the authentication mechanism and build trusted connections among the core Historian services, and also collectors.

> **Note:**
> **MTLS Security** and **MTLS Data Encryption** are not applicable to the File collector.

The core Historian services include:

- Data Archiver
- Client Manager
- Configuration Manager
- Diagnostic Manager

When you install Historian, you are provided with the **Enable Certificate-based Security** check box to enable Certificate-based Security and generate root certificate with a password for Server and the core services. By selecting this option, the installer will generate the root certificates, machine specific certificates, and the core services certificates in the `<Install directory>\Program Files \Proficy\Proficy Historian\MTLS` folder, and add it to Trusted Root Certification Authorities in the machine.




**Warning:**
If you do not select the **Enable Certificate-based Security** check box during installation, you must generate the root certificates manually, as described in the Manually Install Certificates for Historian section. However, this is not recommended.

| Certificate | Applicability |
|---|---|
| ica_key.cer and ica_key.pfx | Root certificates. |
| ClientManager.cer, and ClientManager.pfx | Historian Core Services specific certificates. |
| ConfigManager.cer, and ConfigManager.pfx | |
| DataArchiver.cer, and DataArchiver.pfx | |
| DiagnosticManager.cer, and DiagnosticManager.pfx | |
| <Machine name>.cer, and <Machine name>.pfx | Machine specific certificates. |

If you want to connect a distributed/mirror node to a Historian primary mirror server, or you want to connect your collectors to a remote Historian server, you need the server specific root certificates (ica_key.cer and ica_key.pfx) on the client machine to establish a successful handshake. To establish a successful handshake, you must copy the root certificates from the server machine and place them in the machines where the mirror nodes or collectors are installed. For more information, see the table below.

After installation, based on the install type, you must perform the following configuration:

| Installation Type | Description | Configurations |
|---|---|---|
| **Historian Single Server** | This is for a stand-alone Historian system, which contains only one Historian server. This type of system is suitable for a small-scale Historian setup. | **Collectors and server are installed on the same machine**<br><br>You do not have to perform any additional configurations.<br><br>To use MTLS for collectors, you must enable the MTLS security for the collector instance as needed. For more information, refer to Enable MTLS Security for Collectors.<br><br>**Collectors and server are installed on different machines (Collectors trying to connect to a remote Historian)**<br><br>1. On the server machine, from the `<Install directory>\Program Files\Proficy\Proficy Historian\MTLS` folder, copy |

| Installation Type | Description | Configurations |
|---|---|---|
| | | the following root certificate files: **ica_key.cer** and **ica_key.pfx**, and then place them in the `<Install directory>\Program Files \Proficy\Proficy Historian\MTLS` folder on the collector machine. For more information, refer to Copy server root certificate to the client machine.<br>2. On the collector machine, add the copied certificates to the Trusted Root Certification Authorities folder. For more information on how to add a certificate to the Trusted Root Certification Authorities folder, refer to Adding a Certificate to the Trusted Root Certification Authorities Folder.<br>3. Generate MTLS certificate (client certificate) on the collector machine. For more information on how to generate MTLS certificate (client certificate), refer to Generate MTLS certificate.<br>4. To use MTLS for collectors, you must enable the MTLS security for the collector instance as needed. For more information, refer to Enable MTLS Security for Collectors. |
| **Historian Mirror Primary Server** | This is for a horizontally scalable Historian system, which contains multiple Historian servers, all of which are connected to one another. This will be the primary server for the distributed/mirror node(s). | **Collectors and Historian primary mirror server are installed on the same machine**<br><br>You do not have to perform any additional configurations.<br><br>To use MTLS for collectors, you must enable the MTLS security for the collector in- |

| Installation Type | Description | Configurations |
|---|---|---|
| | | stance as needed, for more information, refer to Enable MTLS Security for Collectors.<br><br>**Collectors and Historian primary mirror server are installed on different machines (Collectors trying to connect to a remote Historian primary mirror server)**<br><br>1. On the mirror server machine, from the `<Install directory>\Program Files\Proficy\Proficy Historian\MTLS` folder, copy the following root certificate files: **ica_key.cer** and **ica_key.pfx**, and then place them in the `<Install directory>\Program Files\Proficy\Proficy Historian\MTLS` folder on the collector machine. For more information,refer to Copy server root certificate to the client machine.<br>2. On the collector machine, add the copied certificates to the Trusted Root Certification Authorities folder. For more information on how to add a certificate to the Trusted Root Certification Authorities folder, refer to Adding a Certificate to the Trusted Root Certification Authorities Folder.<br>3. Generate MTLS certificate (client certificate) on the collector machine. For more information on how to generate MTLS certificate (client certificate), refer to Generate MTLS certificate.<br>4. To use MTLS for collectors, you must enable the MTLS security for |

| Installation Type | Description | Configurations |
|---|---|---|
| | | the collector instance as needed, for more information, refer to Enable MTLS Security for Collectors. |
| **Historian Distributed/Mirror Node** | This is for a horizontally scalable Historian system. Installing this server will allow you to add this node to a primary server. | **Configuration on the distributed/ mirror node machine(s)**<br>1. On the Historian mirror primary server machine, from the `<Install directory>\Program Files\Proficy\Proficy Historian\MTLS` folder, copy the following root certificate files: **ica_key.cer** and **ica_key.pfx**, and then place them in the `<Install directory>\Program Files\Proficy\Proficy Historian\MTLS` folder on the distributed/mirror node machine(s). For more information, refer to Copy server root certificate to the client machine.<br>2. On the distributed/mirror node machine(s), add the copied certificates to the Trusted Root Certification Authorities folder. For more information on how to add a certificate to the Trusted Root Certification Authorities folder, refer to Adding a Certificate to the Trusted Root Certification Authorities Folder.<br>3. Generate MTLS certificate (client certificate) on the distributed/mirror node machine(s). For more information on how to generate MTLS certificate (client certificate), refer to Generate MTLS certificate. |

| Installation Type | Description | Configurations |
|---|---|---|
| | | **Collectors and distributed/mirror node are installed on the same machine**<br><br>You do not have to perform any additional configurations.<br><br>To use MTLS for collectors, you must enable the MTLS security for the collector instance as needed, for more information, refer to Enable MTLS Security for Collectors.<br><br>**Collectors and distributed/mirror node are installed on different machines (Collectors trying to connect to a remote distributed/mirror node)**<br><br>1. On the Historian mirror primary server machine, from the `<Install directory>\Program Files\Proficy\Proficy Historian\MTLS` folder, copy the following root certificate files: **ica_key.cer, ica_key.pfx**, and then place them in the `<Install directory>\Program Files\Proficy\Proficy Historian\MTLS` folder on the distributed/mirror node machine(s). For more information, refer to Copy server root certificate to the client machine.<br>2. On the distributed/mirror node machine(s), add the copied certificates to the Trusted Root Certification Authorities folder. For more information on how to add a certificate to the Trusted Root Certification Authorities folder, refer to Adding a Certifi- |

| Installation Type | Description | Configurations |
|---|---|---|
| | | cate to the Trusted Root Certification Authorities Folder.<br>3. On the Historian distributed/mirror node machine, from the `<Install directory>\Program Files \Proficy\Proficy Historian\MTLS` folder, copy the following root certificate files: **ica_key.cer, ica_key.pfx**, and then place them in the `<Install directory>\Program Files\Proficy\Proficy Historian\MTLS` folder on the collector machine. For more information, refer to Copy server root certificate to the client machine.<br>4. On the collector machine, add the copied certificates to the Trusted Root Certification Authorities folder. For more information on how to add a certificate to the Trusted Root Certification Authorities folder, refer to Adding a Certificate to the Trusted Root Certification Authorities Folder.<br>5. Generate MTLS certificate (client certificate) on the collector. For more information on how to generate MTLS certificate (client certificate), refer to Generate MTLS certificate.<br>6. To use MTLS for collectors, you must enable the MTLS security for the collector instance as needed, for more information, refer to Enable MTLS Security for Collectors. |

**Note:**
If you are using a cluster node setup, you can follow the configurations similar to the Historian Single Server installation type on all the nodes.

## Certificate-based Security Specifc Configurations

## Copy Server root certificate to the Client Machine

If you want to connect a distributed/mirror node to a Historian primary mirror server, or you want to connect your collectors to a remote Historian server, you need the server specific root certificates (ica_key.cer and ica_key.pfx) on the client machine to establish a successful handshake. To establish a successful handshake, you must copy the root certificates from the server machine and place them on the machines where the mirror nodes or collectors are installed.

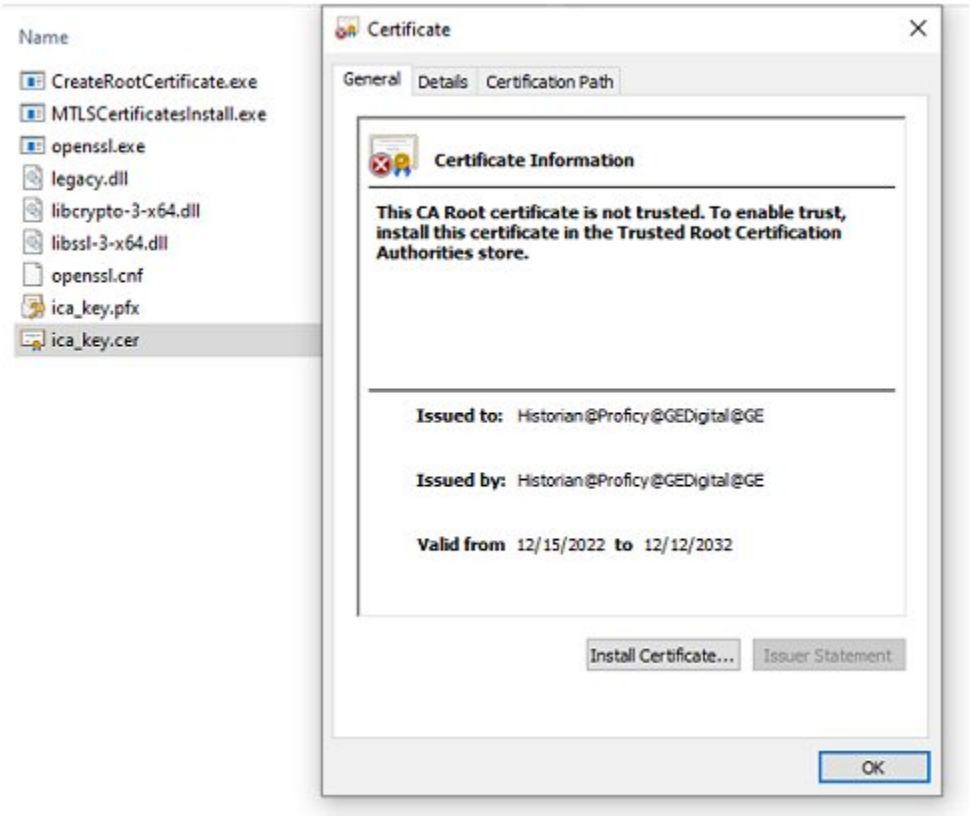
1. On the server machine, from the `<Install directory>\Program Files\Proficy\Proficy Historian\MTLS` folder, copy the following root certificate files: ica_key.cer and ica_key.pfx.
2. On the client machine (remote mirror nodes or collectors) place them in the `<Install directory>\Program Files\Proficy\Proficy Historian\MTLS` folder.
3. Add the certificate to the Trusted Root Certification Authorities folder.

## Adding a Certificate to the Trusted Root Certification Authorities Folder

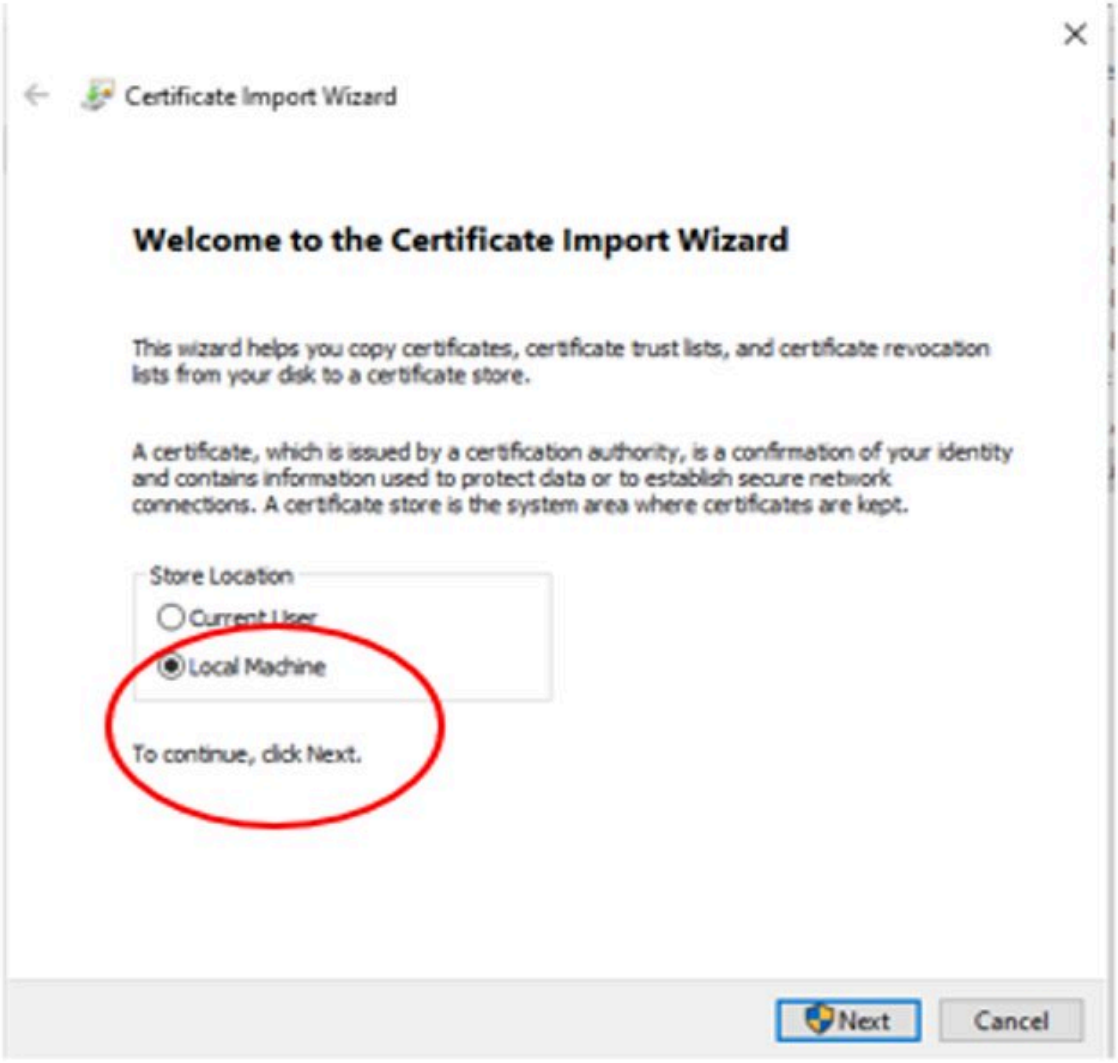
If you want to connect a distributed/mirror node to a Historian primary mirror server, or you want to connect your collectors to a remote Historian server, the root certificate needs to be added to the Trusted Root Certification Authorities folder on the local machine (that is distributed/mirror node or collector machine).

On the primary server machine, from the `<Install directory>\Program Files\Proficy\Proficy Historian\MTLS` folder, copy the following root certificate files: ica_key.cer and ica_key.pfx, and then place them in the `<Install directory>\Program Files\Proficy\Proficy Historian\MTLS` folder on the local machine (that is distributed/mirror node or collector machine).

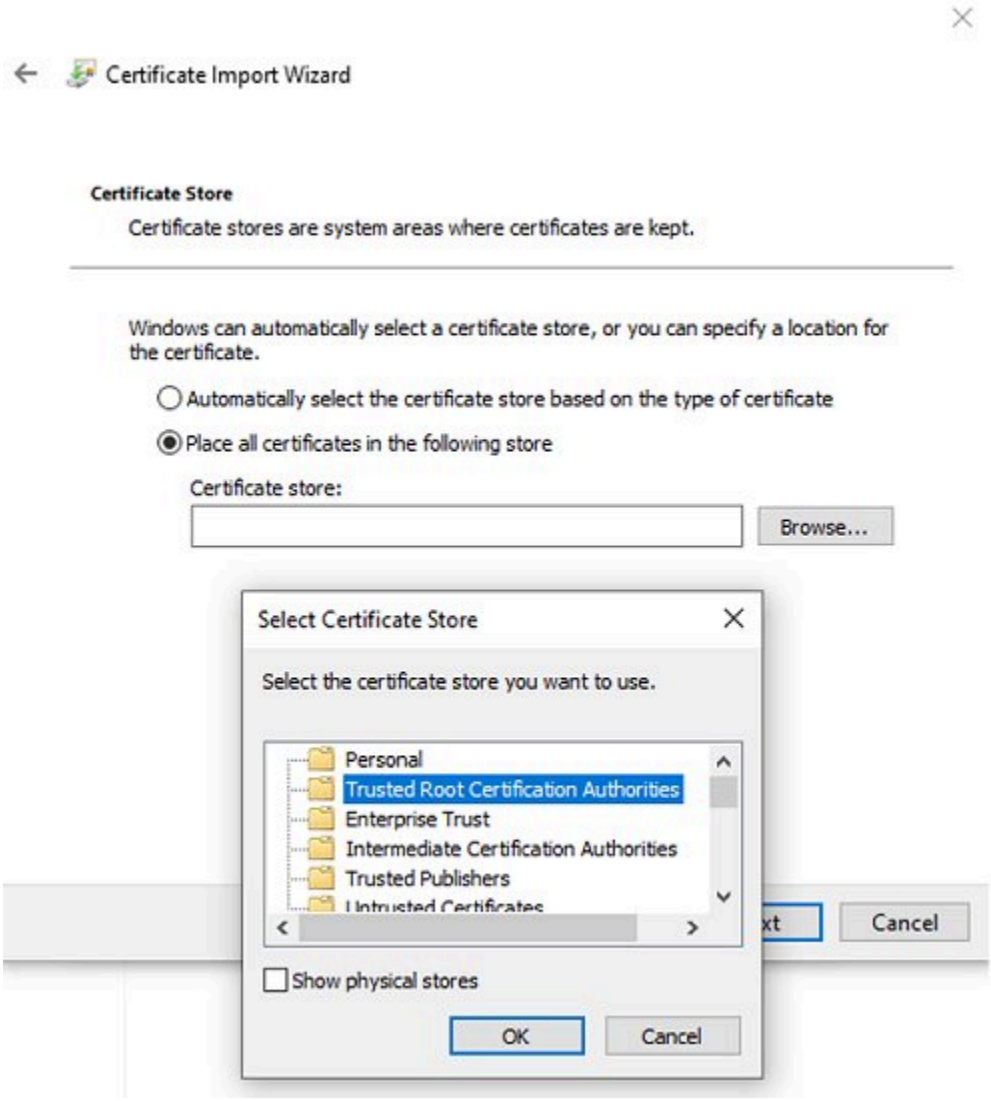
1. Double-click the `ica_key.cer` file. The certificate dialog appears as shown in the following image.



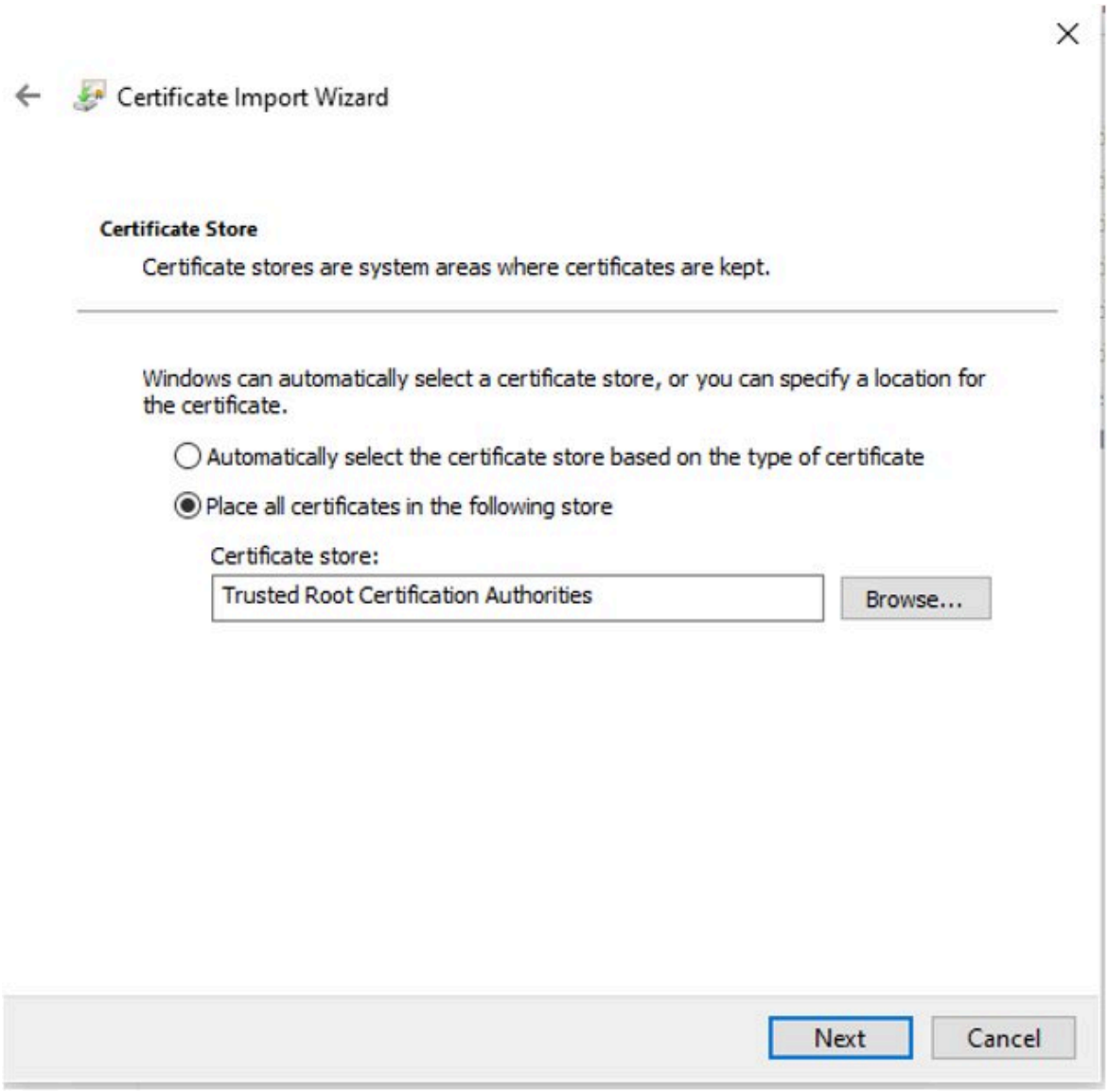

2. Select **Install Certificate** to launch the Certificate Import Wizard.
3. Click **Next** to continue. The **Certificate Import Wizard** window appears as shown in the following image.



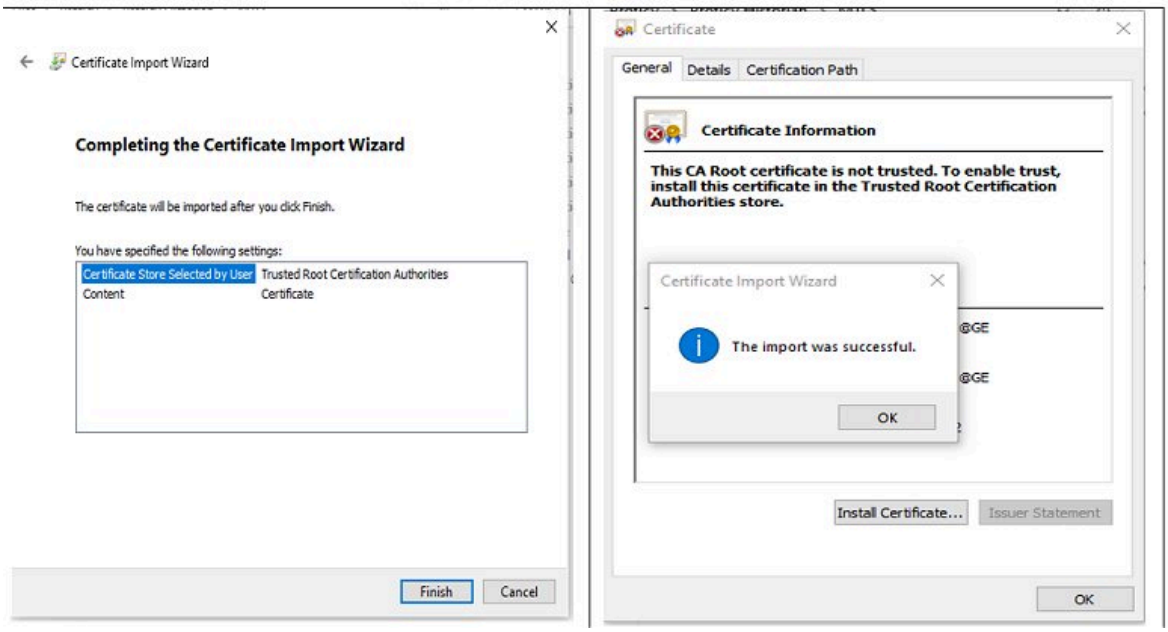

4. Select **Local Machine** and click **Next** to continue. The following screen appears.




5. Select **Place all certificates in the following store**, and click **Browse** to display the list of stores from where Trusted Root Certification Authorities can be selected.
6. Select the **Trusted Root Certification Authorities**, and click **OK**. The following dialog box appears.




7. Click **Next** to continue. The **Completing the Certificate Import Wizard** window appears.




8. Click **Finish** to add the certificate to the **Trusted Root Certification Authorities**.

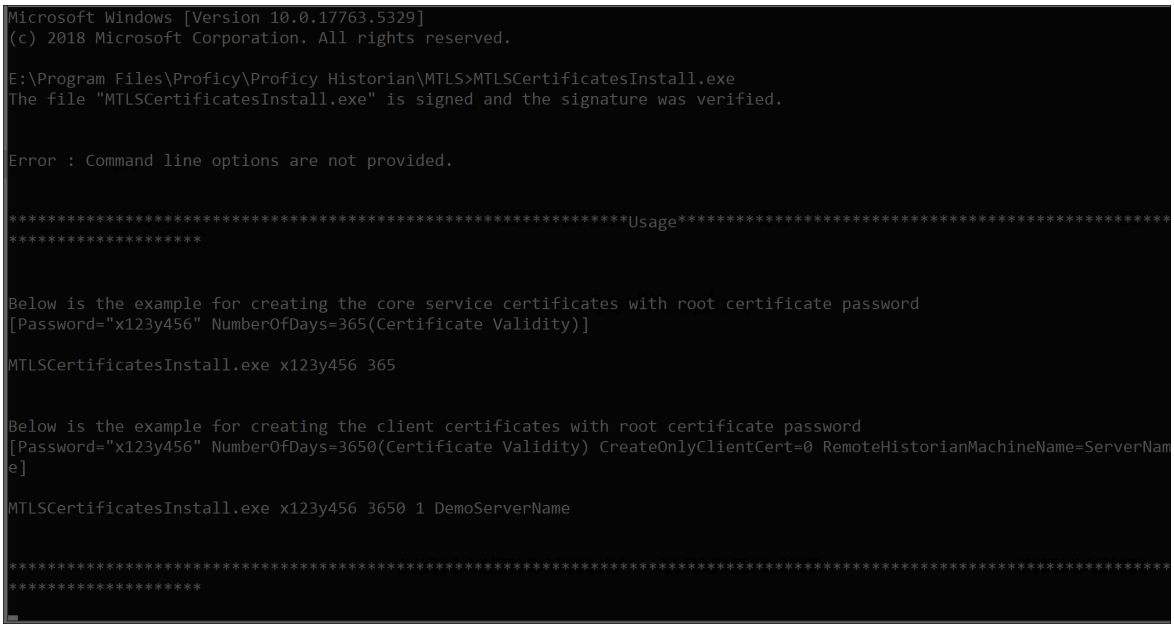
## Generate MTLS Certificate

If you want to connect a distributed/mirror node to a Historian primary mirror server, or if you want to connect your collectors to a remote historian server, you must generate the client certificates (MTLS Certificate) specific to the distributed/mirror node or collector machine. This will create the needed machine specific certificates and registries. You can also generate core services certificates if needed.

You must install the distributed/mirror node, or in the case of collectors, you must install collectors.

To generate client certificates (MTLS certificates), run the `MTLSCertificatesInstall.exe` utility from the command prompt with Administrator privileges.

1. Launch from the command prompt with Administrator privileges in the following format: `C:\Program Files\Proficy\Proficy Historian\MTLS\MTLSCertificatesInstall.exe [Password] [Certificate Validity] [Create only client Certificate] [Remote Historian Machine Name]`




   For example,

   ```
   C:\Program Files\Proficy\Proficy Historian\MTLS\MTLSCertificatesInstall.exe P@55W0RD 3650 1 DemoServerName
   ```

   The **MTLSCertificatesInstall.exe** utility takes the following arguments:

| Argument | Description |
|---|---|
| Password | Specifies the word or phrase that you use to protect your certificate. The Password argument is mandatory, whereas Number of Days is optional. An example Passphrase is: P@55w0rd.<br><br>**Note:**<br>The same Password used for creating the root certificate needs to be used here. This is so that the Password will be same while passing the argument between executables. The MTLSCertificateInstall.exe utility uses this pass- |

| Argument | Description |
| --- | --- |
| | word to open the root certificate private key (ica_key.pfx) and sign the core services certificates. |
| NumberOfDays | Optional. Specifies the Number of Days for the root certificate to be valid. After the specified days, the certificate validity expires.<br><br>If you do not pass any value for Number of Days, the setting defaults to 365 days. For example, if the Number of Days is 3650, the certificate is valid for 10 years from the generated date. |
| CreateOnlyClientCert | Specifies that only client certificates with a root certificate password must be created. You must enter 1 to install only the client certificates. |
| RemoteHistorianMachineName | Specifies the remote Historian server to which you want to connect to. |

The client certificates are created and added to the Trusted Root Certification Authorities Folder. For example, if the machine name is **Node1**, the certificates are created as **Node1.cer** and **Node1.pfx**.

2. After the required certificates are generated, restart the machine. Without valid certificates, core services cannot establish connections to each other.

## Generate Root Certificates for Proficy Historian Server Manually (Not recommended)

This topic describes how to install root certificates and the certificates for core services, for use with the MTLS feature for Proficy Historian.

**Note:**
This topic is applicable if you did not enable certificate-based security during the installation of Historian server. If you enabled certificate-based security during the installation, the root

certificate and core services certificates are generated by default and stored at `<Install directory>\Program Files\Proficy\Proficy Historian\MTLS` folder.

**MTLS Binaries**

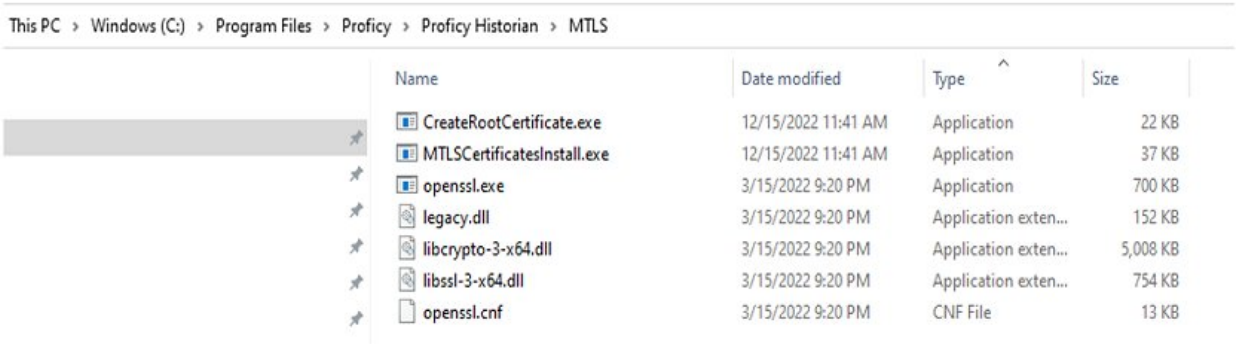
To support MTLS, the Historian install media includes the following files. These files are located in the MTLS folder in the Proficy Historian install folder:

- CreateRootCertificate.exe
- MTLSCertificatesInstall.exe
- openssl.exe
- legacy.dll
- libcrypto-3-x64.dll
- libssl-3-x64.dll
- openssl.cnf

**CreateRootCertificate.exe** and **MTLSCertificatesInstall.exe** are the two command-line utilities for generating the certificates. The other binaries are the dependent components.

**Location of MTLS Binaries**

The following figure shows an example of the binaries folder for MTLS feature, when Proficy Historian 2024 is installed in “C” drive:




To generate root certificate, use the `CreateRootCertificate.exe` utility (in the MTLS folder in the Historian install folder) from a command prompt with **Administrator privileges**, as described in the following steps.

1. Right-click the Command Prompt, and select **Run as Administrator**.
2. Navigate to the MTLS folder in the Historian installed path. For example:

```
cd C:\Program Files\Proficy\Proficy Historian\MTLS
```

3. Run the `CreateRootCertificate.exe` command using the following arguments:

| Argument | Description |
|---|---|
| EnableMTLS | Specifies whether MTLS is enabled. If you do not specify a value, MTLS feature is enabled by default (and set to 1 by default):<br>◦ 0 – MTLS feature is disabled<br>◦ 1 – MTLS feature is enabled<br>For example, if you want to disable the certificate-based security, simply pass “0” to the tool. |
| Password | Specifies the word or phrase that you use to protect your certificate. The Password argument is mandatory, whereas Number of Days is optional. An example Passphrase is: P@55w0rd. |
| The Number of Days | Optional. Specifies the Number of Days for the root certificate to be valid. After the specified days, the certificate validity expires.<br>If you do not pass any value for Number of Days, the setting defaults to 365 days. For example, if the Number of Days is 3650, the certificate is valid for 10 years from the generated date. |

**Note:**

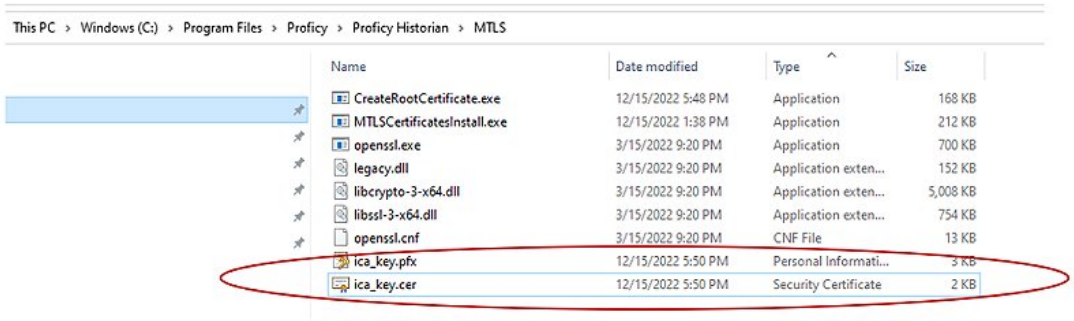
If you fail to pass any values to this command-line, the command will fail to create the root certificate.

The following is an example of the command-line. In this example, MTLS is enabled, the passphrase is P@55w0rd, and the certificate will be valid for 3650 days (10 years):

```
C:\Program Files\Proficy\Proficy Historian\MTLS CreateRootCertificate.exe 1 P@55w0rd 3650
```

4. After executing `CreateRootCertificate.exe`, locate the root keys generated in the same MTLS folder:
    - **ica_key.pfx** – Password protected certificate that contains the private key to sign the core services certificates.
    - **ica_key.cer** – Root certificate contains the public and different attributes of the certificate.

    The following figure shows examples of the root certificates.




    After generating the root certificate, this certificate needs to be added to the “Trusted Root Certification Authorities” certificate store on the Local Machine.

## Installing Certificates for Core Services

For generating certificates for core service, run the `MTLSCertificatesInstall.exe` utility from the command prompt with Administrator privileges.

1. Launch from the command prompt with Administrator privileges. For example:

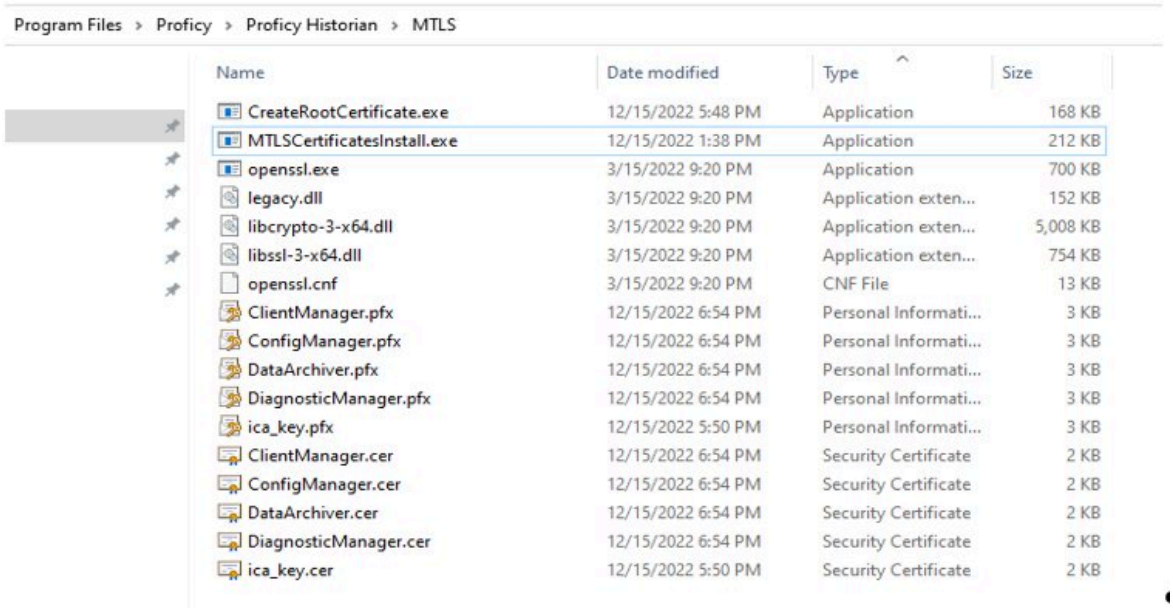
```
C:\Program Files\Proficy\Proficy Historian\MTLS\MTLSCertificatesInstall.exe P@55w0rd 3650
```

    The MTLSCertificatesInstall.exe utility takes the following arguments:

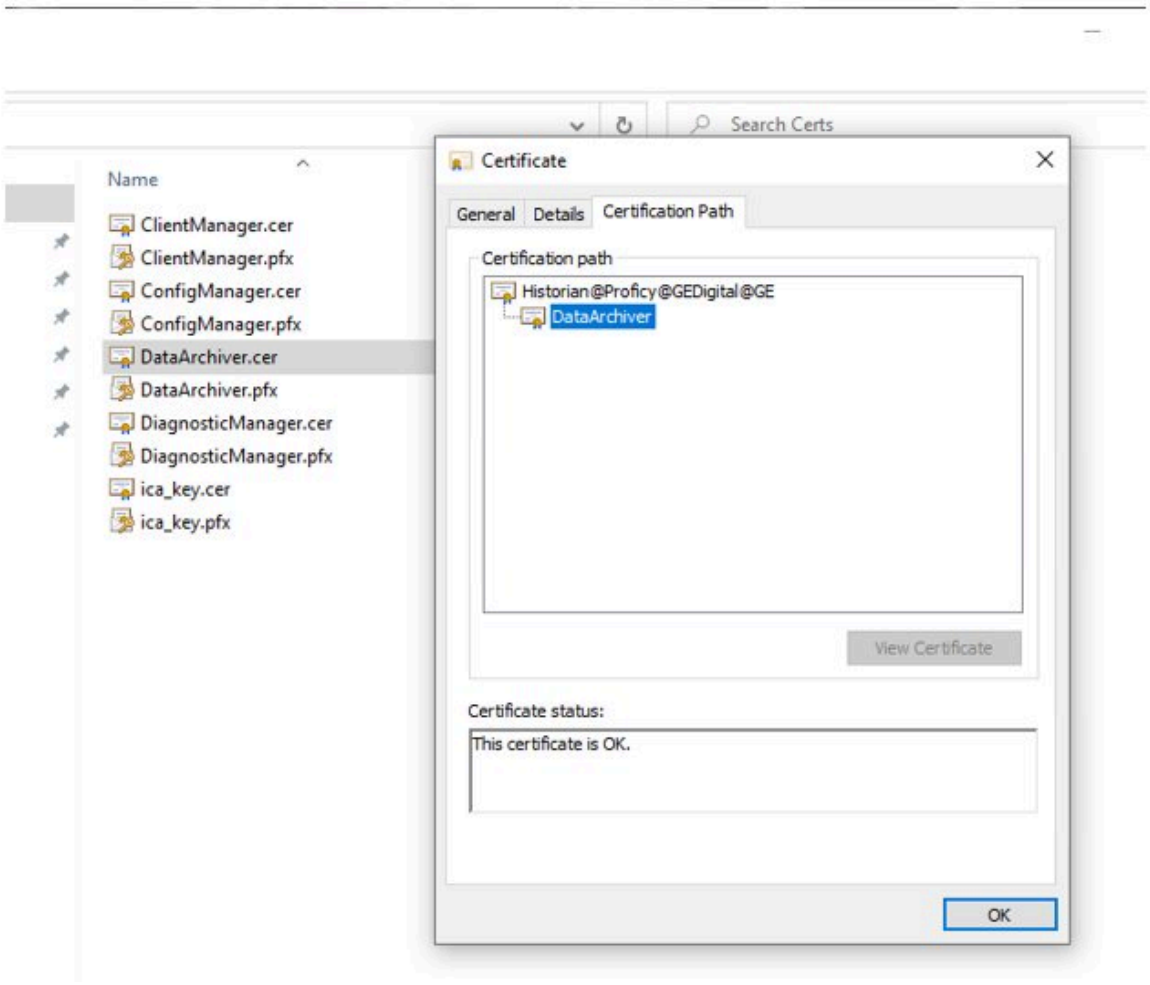
| Argument | Description |
|---|---|
| Password | Specifies the word or phrase that you use to protect your certificate. The Password argument is mandatory, whereas Number of Days is optional. An example Passphrase is: P@55w0rd. |

| Argument | Description |
|---|---|
| | **Note:** The same Password used for creating the root certificate needs to be used here. This is so that the Password will be same while passing the argument between executables. The MTLSCertificateInstall.exe utility uses this password to open the root certificate private key (ica_key.pfx) and sign the core services certificates. |
| The Number of Days | Optional. Specifies the Number of Days for the root certificate to be valid. After the specified days, the certificate validity expires.<br><br>If you do not pass any value for Number of Days, the setting defaults to 365 days. For example, if the Number of Days is 3650, the certificate is valid for 10 years from the generated date. |

For each service, two certificates will be generated as shown in the following figure:




2. Double-click each service .cer file as shown in the following figure, and check whether each generated certificate has a valid root certificate chain.




3. After all required certificates are generated, restart the core Historian services. Without valid certificates, core services cannot establish connections to each other.

## Upgrade Scenarios when Working with Historian Server Certificates

Be aware of the following when upgrading your Historian:

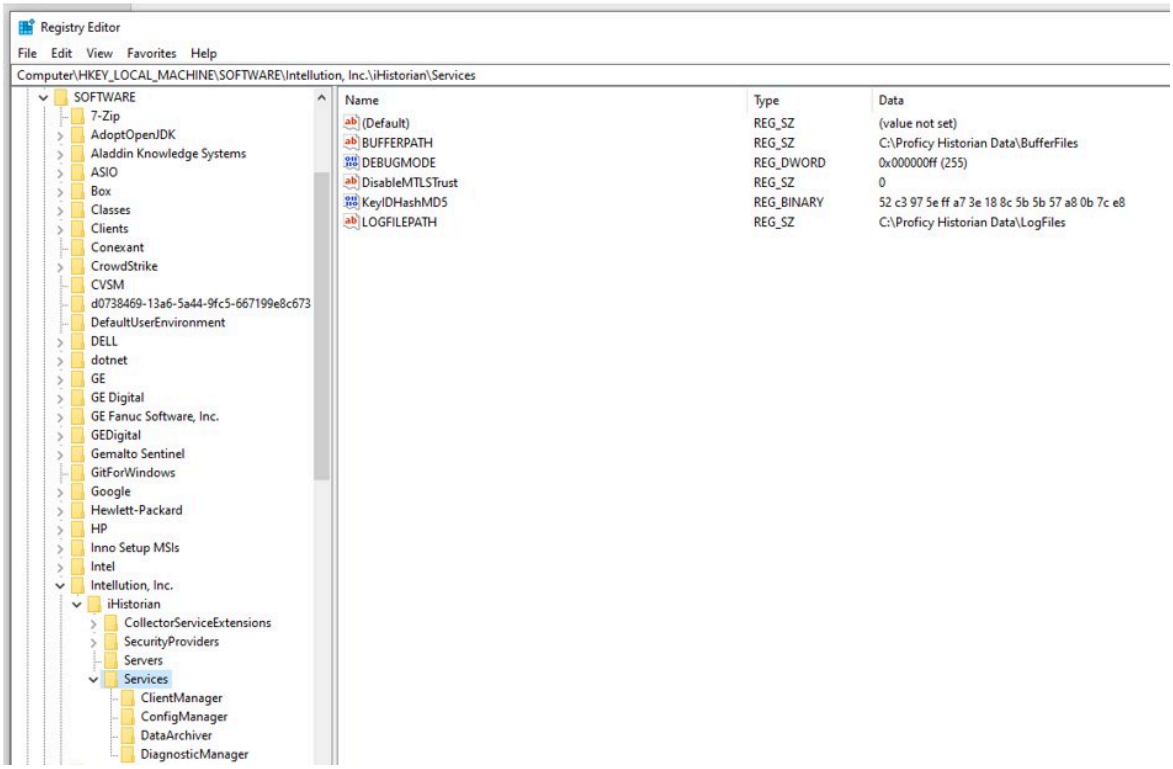
- By default, certificate-based security is enabled. Your configuration for certificates is automatic, unless you are installing or upgrading a distributed/mirror node. If you do not configure your certificates in this scenario, your Historian services may not start when you run it for the first time.
- You will need to regenerate the certificates if you uninstall the existing version and reinstall any version that supports certificate-based security.
- You will need to regenerate the certificates for Historian in-place upgrades that support certificate-based security.
- You will NOT need to regenerate the certificates for SIM over SIM upgrades that support the MTLS feature.

## Troubleshooting Historian Server Certificates

Be aware of the following when working with certificates in Historian:

- When you install Historian, you are presented with three install types: **Historian Single Server**, **Historian Mirror Primary Server**, and **Historian Distributed/Mirror Node**. The MTLS protocol and certificate-based security is enabled by default for all install types. If you are installing a **Historian Single Server** or the **Historian Mirror Primary Server**, the security settings will be automatically configured by the installer. However, if you are installing a **Historian Distributed/Mirror Node**, you must configure the security settings manually after installation, for more information, refer to Overview of certificate-based security.
- You will need to follow the same procedure for installing certificates when certificates expire in the future.
- Prior to adding any new root certificate to the Trusted Root Certification Authorities folder, it is a better practice to remove an existing root certificate from the store first.
- After the root certificates are added to the Trusted Root Certification Authorities folder, all core services need to be restarted.
- If the MTLS authentication fails due to a mismatch in certificates, improperly generated certificates, expired certificates, or any issue related certificates, it will not cause the core Historian Services to stop. These services will be in running state, but the trusted connections among these services will fail. In this scenario, sometimes client tools will not be able to connect to the services. Sometimes client tools connect to Client Manager but cannot do any operations on Historian server. They simply show the “Not Connected” error.
- After trusted connections among core Historian services succeed, they will be in the same trusted state until these services are stopped or restarted.
- It is strongly recommended to provide same expiration date (in the Number of days field) for the root and all other core services certificates.
- If you forget to install the certificates, some of your core Historian services may not start after you complete your Historian install or upgrade.
- If you do not provide the same root certificate password while creating other services certificates, the MTLSCertificatesInstaller.exe tool returns a "Wrong Password" error.

- To see any specific errors that may be caused by certificate-based security, you need to enable full debugging by adding the “FF” hexadecimal value to the DEBUGMODE registry as shown in the following figure.




# Using Proficy Authentication or LDAP Groups

## About Proficy Authentication

In Historian, user authentication is handled using Proficy Authentication, which provides user account and authentication (UAA) service. Proficy Authentication provides identity-based security for applications and APIs. It supports open standards for authentication and authorization, including Oauth2.

When a user is created, modified, or deleted in Historian, the associated user account is being created, modified, or deleted in the Proficy Authentication instance, respectively.

**Note:**
This is done in the back-end automatically. Therefore, most users will not require knowledge on UAA to perform basic user management, except when additional configuration is required.

To use Proficy Authentication, you can choose between the following options while installing Web-based Clients:

- **Use a local Proficy Authentication service:** Use this option if you want to create a local Proficy Authentication instance. This is the default option. You can create this while installing Web-based Clients.
- **Using an existing Proficy Authentication service:** Use this option if you are currently using a Proficy Authentication service on a remote machine. You can install this service using Historian Web-based Clients, or you can use any other UAA service (such as Proficy Authentication installed using Operations Hub). You can then manage these users in Web-based Clients. The users in the remote Proficy Authentication service can then use Web-based Clients.

You can use Proficy Authentication plugin in Configuration Hub to manage groups, manage users, and map LDAP groups, and SAML groups with the Proficy Authentication groups.

For more information, refer to Proficy Authentication in Configuration Hub at https://www.ge.com/digital/documentation/confighub/.

- **Manage Identity Providers**
  - Add LDAP identity provider
  - Modify LDAP identity provider
  - Enable SAML
  - Add SAML identity provider
  - Modify SAML identity provider
  - Delete Identity Provider
- **Manage Groups**
  - Overview of Historian groups in Proficy Authentication.
  - Create groups
  - Map groups
  - Modify groups
  - Add or remove users in a group
  - Add or remove sub-groups in a group
  - Delete group
- **Manage Users**
  - Create users
  - Add LDAP or LDAPS users
  - Add or remove groups for a user

- Reset user password
- Delete user

## About Proficy Authentication Groups

A Proficy Authentication group is created for a specific type of users who will likely perform the same type of activities.

If you have groups in a remote Proficy Authentication service, you can use them with Historian using the Proficy Authentication LDAP Integration tool. This section describes how to map the groups in the remote Proficy Authentication service with Historian counterparts. By default, Historian contains the following Proficy Authentication groups:

- **historian_visualization.admin:** Provides access to Trend Client and the Web Admin console.
- **historian_visualization.user:** Allows access to Trend Client.
- **historian_rest_api.read:** Provides read access to public REST API.
- **historian_rest_api.write:** Provides write access to public REST API.
- **historian_rest_api.admin:** Provides read/write access to public REST API.
- **historian_enterprise.admin:** Provides read/write access to Configuration Hub APIs.
- **historian_enterprise.user:** Provides access to view Configuration Hub APIs.
- **ih_archive_admins:** Provides the ability to create, modify, and remove archives.
- **ih_audited_writers:** Allows data writes and to produce a message each time a data value is added or changed.
- **ih_collector_admins:** Allows the ability to add collector instances and change their destination.
- **ih_readers:** Provides access to the ability to read data and system statistics. Also allowed access to Historian Administrator.
- **ih_security_admins:** Provides access to Historian power security users. Security administrators have rights to all Historian functions.
- **ih_tag_admins:** Provides access to allow the ability to create, modify, and remove tags. Tag-level security can override rights given to other Historian security groups. Tag admins can also browse collectors.
- **ih_unaudited_logins:** Allow connenctions to the Data Archiver without creating login successful audit messages.
- **ih_unaudited_writers:** Provides the ability to write data without creating any messages. Tag, archive, and collector changes log messages regardless of whether the user is a member of the ih_audited_writers group.

**Note:**
Instead of mapping the groups, you can choose to map individual users with Historian users. For instructions, refer to Managing Proficy Authentication Users Using the Configuration Tool *(on page 284)*.

### Workflow

1. Provide the details of the remote Proficy Authentication service while installing Web-based Clients *(on page 154)*.
2. Connect to the remote Proficy Authentication service *(on page 182)*.
3. Map the Proficy Authentication groups with that of the Historian Proficy Authentication instance. You can map the groups in LDAP and LDAPS (LDAP via SSL).

## Configurations Checklist to use Proficy Authentication Security Groups

In Configuration Hub, the **Global Security** section in the system **DETAILS** section enables you to specify the authorization for Historian security groups. To use Proficy Authentication security groups, you must perform the configurations listed in this topic.

This topic intends to be a checklist for you to perform the configurations that are needed to use Proficy Authentication security groups for authorization.

1. Access Configuration Hub using ch_admin as the username.

2. In the **NAVIGATION** pane, expand **Proficy Authentication**, and then select **Security**.
3. Add the same Windows Login user (For example, if you logged in as Administrator, use Administrator), and then map the user with all admin groups. For more information on adding users and mapping the users to groups, refer to Create groups, Add or remove users in a group and Map groups. For more information on Historian security groups, refer to Overview of Historian groups in Proficy Authentication.
4. In the **Users** tab, from the list of users, select the predefined user, that is <hostname>.admin, and then map it to **ih_security_admins**. For more information, refer to Map groups.

5. Add the users defined in Proficy Authentication using the same password to **Local users & Group**. This will enable you to access thick clients (VB admin, iHSQL). For more information on how to add a user to **Local users & Group**, refer to Adding Users to Windows Security Group.

   Alternatively, you can add the users by adding registry entries.

   a. In the `HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Intellution, Inc.\iHistorian\Servers` and `HKEY_LOCAL_MACHINE\SOFTWARE\Intellution, Inc.\iHistorian\Servers` registry paths, add the following registries.

| Name | Type | Data |
|---|---|---|
| **UAA** | **REG_SZ** | **https://hostname.domain:433/uaa/oauth/token**<br><br>For example,<br><br>https://testmachine.ge-.com:443/uaa/oauth/token. |
| **UseUaaAuthentication** | **REG_DWORD** | **0x00000001**. |

   b. Save the registries.

6. Now, you can Access Configuration Hub *(on page 336)* using the predefined username, that is, <hostname>.admin and password.
7. In the **NAVIGATION** pane, in the Configuration Hub plugin for Historian, access **Systems**, and then select the system as needed.

   The system appears in the main section. The system details appear in the **DETAILS** section.

8. In the **GLOBAL SECURITY** section, select **Use Proficy Authentication**.
9. On the top-right corner, select **Save**.
10. Restart all the services, including the Remote Collector Management (RCM) service.

Now you can use the Proficy Authentication security groups for authorization.

To know if your configurations were saved, check the following:

- The CollectorManager.shw status must be **Connected**.
- The DataArchiver.shw security settings must be as shown below.

```
Security Settings
=================
                    Security Group Source : Proficy Authentication
                        Security Provider : OAuth2
                                      URI : https://hostname:domain:443/uaa/check_token
                                      UAA : https://hostname:domain:443/uaa/oauth/token
        Use Client Windows User for Logon : TRUE
                            Use ADSI calls : FALSE
             Strict Client Authentication : TRUE
           Strict Collector Authentication : TRUE
               Allow Anonymous WCF Clients : FALSE
                       Anonymous User Name : Guest
```

## Using Server Certificates

To use server certificates with Historian, use the Certificate Management tool. This tool supports the following combination of files to import the certificate chain and the private key:

- A PEM file that contains the certificate chain and the private key.
- A PEM file that contains the certificate chain, and another PEM file for the private key.
- A PFX file that has the certificate chain and the private key.

For instructions on using the Certificate Management tool, refer to https://www.ge.com/digital/documentation/opshub/windows/windows/c_about_certificate_management.html.

## Change the Log Levels of Proficy Authentication

1. Access the `log4j.properties` file in the following folder: `C:\Program Files\GE \Operations Hub\uaa-tomcat\webapps\uaa\WEB-INF\classes`
2. For each module, select one of the following log levels depending on your requirement:
    - TRACE
    - DEBUG
    - INFO
    - WARN
    - ERROR
    - FATAL
    - OFF
3. If you want to disable Tomcat logging:

a. Stop the Proficy Authentication Tomcat Web Server service.
b. In the `C:\Program Files\GE\Operations Hub\uaa-tomcat\bin` folder, rename the `tomcat8w.exe` file `UaaTomcat.exe`, and run this application as an administrator.
c. Select **Logging**.
d. Remove the auto keyword from the Redirect Stdout and Redirect Stderr labels.
e. Start the Proficy Authentication Tomcat Web Server service.

4. If you want to change the Tomcat log level:

a. Stop the Proficy Authentication Tomcat Web Server service.

b. Access the `context.xml` file located in the `C:\Program Files\GE\Operations Hub \uaa-tomcat\conf`.

c. In the Context tag, add: `swallowOutput="true"`

d. Access the `logging.properties` file in the same folder, and set the **2localhost.org.apache.juli.AsyncFileHandler.level** to one of the following values, which are in the order of less verbose to more verbose:
   - SEVERE
   - WARNING
   - INFO
   - CONFIG
   - FINE
   - FINER
   - FINEST
   - ALL

e. Start the Proficy Authentication Tomcat Web Server service.

# Migrating Historian Data

## Migrating the Alarms and Events Data

If you have upgraded Historian, you must migrate the alarms and events data as well. Only then you can retrieve the data.

The steps to migrate depend on the Microsoft SQL version in which the alarms and events data is stored:

- If using Microsoft SQL 2008 or later, you can install Historian and its components, install a supported version of Microsoft SQL, and then migrate the data.
- If using a version earlier than Microsoft SQL 2008, you must first migrate the data to Microsoft SQL 2008, and then migrate it to a supported version of Microsoft SQL.

To migrate data, you can choose one of the following options:

- **Before upgrading to the latest version of Historian:** In this case, you can use Historian Administrator of the older version of Historian to migrate the data.
- **After upgrading to the latest version of Historian:** In this case, you can use the Proficy Alarm Database Migration tool, which is provided with Historian.

This section describes how to migrate data using the migration tool.

## Workflow for Migrating Alarms and Events Data

If the alarms and events data is currently in Microsoft SQL 2008 or later:

1. Install the following components on the target machine in the given sequence:
    a. Historian *(on page 104)*
    b. Alarms and Events *(on page 140)*
    c. Collectors *(on page 142)*
    d. Client Tools *(on page 150)*
    e. Standalone Help *(on page 205)*
2. Back up the alarms and events data *(on page 237)*.
3. Install Microsoft SQL on the target machine. Refer to Software Requirements *(on page 92)* for a list of supported versions.
4. Restore the data that you have backed up to Microsoft SQL.
5. As needed, perform calculations on the migrated data. Ideally, you must create an archive already to store the calculated data. For unsolicited calculation tags, migration of data will cause the calculation to be triggered automatically for the time associated with the migrated data points. Archives will potentially grow beyond the configured default size. To avoid this issue, adjust the value for the `DataIsReadOnlyAfter` field on the **Security** section of the **Data Store Maintenance** page of Historian Administrator (or the `ActiveHours` property) so that the value is large enough to contain the calculated data. By default, this value is 1 month.

If the alarms and events data is currently in a version earlier than Microsoft SQL 2008:

1. Using Microsoft SQL Server Management Studio, back up the alarms and events data. If the database is large, consider taking a partial backup instead of a full backup.
2. Install Microsoft SQL Server 2008 on the target machine.
3. Restore the data that you have backed up in step 1.
4. In Microsoft SQL Server Management Studio, under **Databases**, right-click the database that you have restored, and then select **Properties > Options**.
5. In the **Compatibility level** field, select **SQL Server 2008**.
6. Install the following components on the target machine in the given sequence:
    a. Historian *(on page 104)*
    b. Alarms and Events *(on page 140)*
    c. Collectors *(on page 142)*
    d. Client Tools *(on page 150)*
    e. Standalone Help *(on page 205)*
7. Back up the alarms and events data *(on page 237)* that you have restored in step 3.
8. Install a supported version of Microsoft SQL on the target machine. Refer to Software Requirements *(on page 92)* for a list of supported versions.
9. Restore the data that you have backed up in step 7 to Microsoft SQL.
10. As needed, perform calculations on the migrated data. Ideally, you must create an archive already to store the calculated data. For unsolicited calculation tags, migration of data will cause the calculation to be triggered automatically for the time associated with the migrated data points. Archives will potentially grow beyond the configured default size. To avoid this issue, adjust the value for the `DataIsReadOnlyAfter` field on the **Security** section of the **Data Store Maintenance** page of Historian Administrator (or the `ActiveHours` property) so that the value is large enough to contain the calculated data. By default, this value is 1 month.

## Back Up the Alarms and Events Data

Install the following components in the given sequence:

1. Historian *(on page 104)*
2. Alarms and Events *(on page 140)*
3. Collectors *(on page 142)*
4. Client Tools *(on page 150)*
5. Standalone Help *(on page 205)*

If, however, the alarms and events data is currently in a version earlier than Microsoft SQL 2008, you must first migrate the data to Microsoft SQL 2008, and change the compatibility level to **SQL Server 2008**.

1. Go to the <*Historian installation folder*>\Proficy DataBase folder, and open the Proficy.Historian.AandE.Migration.exe file.
   The **Backup Existing Alarms and Events** window appears.
2. In the **Time Range** section, in the **From** and **To** fields, select the start time and end time of the backup duration.
   We recommend that you select small duration if you have many alarms. If you want to migrate the alarms in blocks of time, choose the oldest alarms first.
3. In the **Database Name** field, enter the name of the database that you want to back up. Typically, this will be the same as the Microsoft SQL server you are using.
4. Depending on whether you want to use Windows credentials or Microsoft SQL credentials, select either **Use Windows Authentication** or **Use SQL Authentication**, respectively.
5. In the **User Id** and **Password** fields, enter the login credentials. Provide the username of a user who has the permission to connect and back up alarms.
6. In **Backup Folder Path** field, provide the absolute path, including the file name, to store the backed up alarms (for example, c:\temp\March2010.bak). You can enter the path of a local file or a remote one, depending on whether the Microsoft SQL server is installed on the local machine or a remote machine.
7. Select **Test Connection** to check if the source database is active and the information is accurate.
   The **Begin Backup** button is activated.
8. Select **Begin Backup**.
   The alarms and evens data is backed up. The count of the rows that are backed up appears.

Install a supported version of Microsoft SQL, and restore the data *(on page 238)* that you have backed up. Refer to Software Requirements *(on page 92)* for a list of supported versions.

## Restore the Alarms and Events Data

Install Microsoft SQL. Refer to Software Requirements *(on page 92)* for a list of supported versions.

1. Go to the <*Historian installation folder*>\Proficy DataBase folder, and open the Proficy.Historian.AandE.Migration.exe file.
   The **Backup Existing Alarms and Events** window appears.
2. Select **Migrate Alarms and Events Backup**.
3. In the **Backup Folder Path** field, provide the absolute path of the file (including the file name) in which you want to restore the data (for example, c:\temp\March2010.bak). You can enter the path of a local file or a remote one, depending on whether the Microsoft SQL server is installed on the local machine or a remote machine.
4. In the **Database Name** field, enter the name of the database that you want to back up. Typically, this value is the same as the Microsoft SQL server you are using.

5. Depending on whether you want to use Windows credentials or Microsoft SQL credentials, select either **Use Windows Authentication** or **Use SQL Authentication**, respectively.
6. In the **User Id** and **Password** fields, enter the login credentials. Provide the username of a user who has the permission to connect and back up alarms.
7. Select **Test Connection** to check if the source database is active and the information is accurate. The **Begin Migrator** button is activated.
8. Select **Begin Migrator**.
   The alarms and evens data is restored. The count of the rows that are restored appears.

## Using the Migration Tool

The IHA Migration Tool (`MigrateIHA.exe` for 32 bit or MigrateIHA_x64.exe for 64 bit) allows you to migrate data up to 30 years old if the data is already stored in `IHA` files from any version of Historian. Use the Migration Tool to move data from one archiver to another when you cannot simply restore the `IHA` in Historian Administrator.

The Migration Tool opens an `IHA` file as a binary data file and reads the raw samples from it. Those raw samples are then written to a destination archiver, in a similar way to how an OPC collector or File collector would write data. Any errors returned from the data archiver are reported in the main window and repeated in the log file.

**Note:**

- You can migrate UserDefined types, MultiField tags, and Array tags.
- When you are migrating the data stores, the source data store is created in the destination.
- Using this Migration Tool, you can upgrade from two previous versions of Historian to the latest version.
- The performance of this tool is impacted with the addition of Client Manager and Configuration Manager. For best performance, use this on a Single Server install only.

## Migrating Historical Data

You need to run this tool as an administrator to migrate and create the log files in the `C:\` directory.

To migrate historical data stored in `IHA` files from any version of Historian:

1. In the `Historian` folder, double-click the Migration Tool executable (`MigrateIHA.exe` for 32-bit or `MigrateIHA_x64.exe` for 64-bit) to open the IHA Migration Utility.
   The icon for the executable looks as follows: .
2. Select **Configure Options** from the **Options** menu.
3. Enter or modify any specific configuration information.
   When choosing an `IHC` file, do not specify one currently in use by the Data Archiver. (For more information, see Configuring Migration Options *(on page 240)*.)
4. Select **File > Migrate Historical Data** .
   The **Select Historical Data File(s)** window appears.
5. Select a historical file and select **Open**.
   Refer to the **IHA Migration Utility** main page for information on the progress of the migration and any encountered errors.

   **Note:**

   The IHA Migration Utility page only displays the most recent lines of the log file. For the full set of logged messages, refer to the log file, typically located in `C:\IHAMigration.Log`.

6. Optionally, perform these steps:

   a. You can upgrade the older version's archive files to the latest version by selecting the bulk upgrade option.
      Stop the Data Archiver service and select **File > Bulk Upgrade Historical Data**.

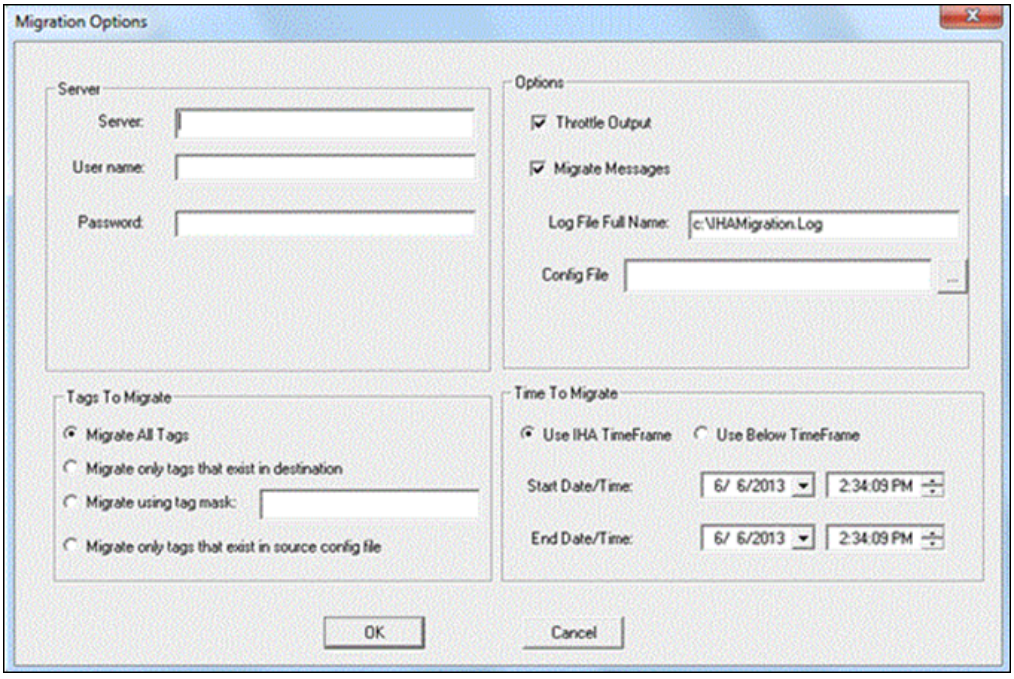
      If you do a bulk upgrade of historical data immediately after you install the latest version on Historian, then save on upgrading while the system reboots.

   b. To clear the log messages displayed in the page, select **File > Clear Display**.

   c. To view the logs saved in the `IHAMigration.log` file, select **File > View Log File > .**.

## Configuring Migration Options

1. In the Migration tool (`MigrateIHA.exe` for 32 bit or `MigrateIHA_x64.exe` for 64-bit), select **Options > Configure Options**.
   The **Migration Options** window appears showing the default server information and the default migration options.




2. Enter options the following options.

---

**Related reference**

Server Pane *(on page 241)*

Options Pane *(on page 242)*

Tags to Migrate Pane *(on page 242)*

Time to Migrate Pane *(on page 242)*

## Server Pane

| Field | Description |
|---|---|
| **Server** | The default server (set during installation). If you do not want to write data to the default server, enter the desired server in this field. |
| **Username and Password** | If you have created and established Security Groups in your Historian Security Environment, you may need to enter the user name and password here. By default, if you do not supply any information, the current logged in user will be used in security checking. |

## Options Pane

It is always advisable to take a copy of the configuration file and work on the copy rather than working on the original file.

## Tags to Migrate Pane

| Option | Description |
|---|---|
| **Migrate All Tags** | Select this option to migrate all the tags from the selected archiver. |
| Migrate only tags that exist in destination | Select this option to migrate all the tags that exists in the source destination. |
| Migrate using tag mask | Select this option to migrate tags with the mask specified. You can specify an exact tag name to migrate that tag only. |
| **Migrate only tags that exist in source config file** | To migrate the tags that are present only with the source config file. |

## Time to Migrate Pane

| Option | Description |
|---|---|
| **Use IHA TimeFrame** | Select this option to migrate all the tags which has the IHA time frame. |
| **Use Below TimeFrame** | Select this option to migrate all the tags in the specified time frame. You need to specify the Start Date/Time and End Date/Time if you select this option. |

## Data Migration Scenarios

You can migrate tags and their data on the same Historian Server or between servers. When migrating your data, consider the following guidelines:

- Get new collection working first

  When the data is collected from the collectors or the API programs, then you should consider adding the tag definitions into the destination server and directing data to be written there before you start migration, because migration may take several hours or days.

- Migrate data from oldest to newest

It is advisable to migrate the oldest data first and then the newest, to make the optimal use of archive space.

- Pay attention to TagID

  Every tag in Historian 4.5 and above has a property called TagID, that uniquely identifies it and allows data retrieval to locate the data. Even if you have a tag of with the same name in another archiver, that tag has a different TagID and is considered as a different tag. You can see the TagID of a tag in the Excel Tag Export. Preserve that number when moving a tag from one system to another.

The following are commonly used scenarios while migrating data on the same Historian server or between servers.

- Migrating a Tag and its data from one data store into another data store.
- Merging a Historian Server into an existing data store on another machine. .

## Migrating a Tag and its Data

If you want to separate a single large user data store of tag into multiple smaller data stores on the same machine, and if your software license allows it, then you should assign the tag to the new data store and then migrate the data.

Consider when data is collected for the year 2009 in `Tag1`. The collected data is archived in the default User data store. If you want to move `Tag1` residing in the `User` data store to another data store, (for example, the `Motor` data store), then you must create the `Motor` data store if it does not already exist and if your license allows it.

The next step is to change the data store of the tag. You can change the data store of the tag either using Historian Administrator or using Excel Tag Import. The new incoming data gets collected in the Motor data store. If you do a raw data query, you will get only the latest data and the previous data will not be available. To get the old data, you must migrate the data residing in the `User` data store to the `Motor` data store.

To migrate a tag and its data from one data store to another data store on the same server:

1. Use `iharchivebackup -c` to make a backup of the `.ihc` file.
   The backup of the Config file is automatically created in the `Archives` folder.
2. In Historian Administrator, back up each archive from oldest to newest.

3. Launch the Migration Tool (`MigrateIHA.exe` for 32-bit or `MigrateIHA_x64.exe` for 64-bit) using Administrator privileges.
4. Select **Options > Configure Options**.
5. In the **Server** pane, enter the **Server name**.
6. In the **Options** pane, enter the **IHC File** path in the **Config File** path field, using the browse button. This is the path to the `IHC` backup that you made in step 1.
7. In the **Tags to Migrate** pane, select the **Migrate Using Tag Mask** option and enter the **Tag Name** you moved to the new data store.
8. In the **Time to Migrate** pane, ensure the **Use IHA TimeFrame** option is selected.
9. Select **File > Migrate Historical Data**.
10. Select the archive file that you backed up in Step 2 and monitor the progress of the migration. When the migration is complete, query the data to see the migrated data can be queried. Repeat with the remaining archives from oldest to newest.

As needed, perform calculations on the migrated data. Ideally, you must create an archive already to store the calculated data. For unsolicited calculation tags, migration of data will cause the calculation to be triggered automatically for the time associated with the migrated data points. Archives will potentially grow beyond the configured default size. To avoid this issue, adjust the value for the `DataIsReadOnlyAfter` field on the **Security** section of the **Data Store Maintenance** page of Historian Administrator (or the `ActiveHours` property) so that the value is large enough to contain the calculated data. By default, this value is 1 month.

## Merging a Historian Server

A typical scenario is to merge a Historian Server into an existing data store on another machine.

If your system architecture has evolved from multiple smaller servers into fewer large archives, you can eliminate the smaller machines while preserving all your tag configuration and collected data.

Consider the following example. You have two machines, Machine A and Machine B. Machine A is running current or any earlier version of Historian and has 100 tags and 10 archive files. The data of these tags are collected from the collector and is being queried by users. Machine B is running the current version of Historian.

**Note:**

- This example does not include Alarm migration. If Machine A was being used to store alarms, then you need to migrate those before eliminating Machine A.
- You cannot migrate tags with Enumerated Data Sets. If you want to migrate data for Enumerated Data Sets, then you must create the Enumerated Data Sets in Historian Administrator or Microsoft Excel and then migrate the tags.
- To migrate tags which are condition based triggers, then you must create the condition-based triggers for that tag in Historian Administrator or Microsoft Excel and then migrate the tags.

You can migrate data only if the file format of the archive files format is `.IHA`. If the back-up archive is in `.zip` format, extract the `zip` files and copy all the `.IHA` files separately in a folder.

1. Before migrating, copy the `.IHC` and all the `.IHA` files from Machine A to Machine B.
2. Launch the Migration Tool (`MigrateIHA.exe` for 32-bit or `MigrateIHA_x64.exe` for 64-bit) with Administrator privileges.
3. Select **Options > Configure Options**.
4. In the **Server** pane, enter the **Server name**.
5. In the **Tags to Migrate** pane, ensure that the **Migrate All Tags** option is selected
6. In the **Options** pane, enter the **IHC File** path in the **Config File** path field, using the browse button. The path you enter is the path to the .IHC file brought over from Machine A.
7. In the **Time to Migrate** pane, ensure the **Use IHA TimeFrame** option is selected.
8. Ensure **Throttle Output** is selected.
9. To migrate the data, select **File > Migrate Historical Data** and select the archive file that has the oldest data.
   The tags and data are migrated to the default data store in time slices. The **MigrateIHA** window displays the progress and any Tag Add or Data Add errors are displayed in the log file. You can estimate the remaining time by watching the progress.
10. Repeat the previous steps for each of the remaining archives, from oldest to newest data.
11. Add the collector to the Historian Server on Machine B.
    See the *Adding a Data Collector to an Historian Server* topic in *Data Collectors - General*.

# Migration Tool Command-Line Syntax

## Command Syntax

- For 32-bit:

```
MigrateIHA.exe "<IHA file name with full path>" "<IHC filename with full path>"
```

- For 64-bit:

```
MigrateIHA_x64.exe "<IHA file name with full path>" "<IHC filename with full path>"
```

## Command-line Options

| Option | Description |
|---|---|
| `/NOTHROTTLE` | This does not throttle any part of the migration process, but may impact resources on the server. Optionally, you can remove this switch as required. By default, throttling is rated at 5000 events per second. |
| `/NOMESSAGES` | This does not migrate messages into the newly created archive. Using this switch may or may not reduce the size of your archives, depending on the number of messages stored in the archive. By default, messages are migrated if this switch is not used. |
| `/EXISTINGTAGS` | This will migrate data for only those tags that exist in the destination archiver. |
| `/b` | This option of the `start.exe` file allows the IHA Migration tool to start without opening a new window for each instance.<br><br>If you are migrating a pre 4.5 IHA file you will need to have the IHC file for that IHA and specify the IHC file in the Options window or on the command line. Otherwise, you will get a warning message. |
| `/wait` | This option of the `start.exe` file allows each instance of the IHA Migration tool to complete the migration before starting the next migration in the sequence. |
| `/NOIHC` | This option skips verifying for IHC file and proceed with the migration. IHC file is not required, if batch command have /NOIHC option. |

### Notes

- If you are migrating from a command line, provide IHC file, else, use /NOIHC option to omit the IHC file.
- If you do not have the IHC or you are not sure you have the correct IHC then you should use the pre-4.5 version of MigrateIHA to migrate the IHA. Otherwise, the data will not migrate correctly.
- You should keep a copy of the original IHA file.
- The IHC must contain all the tags that are in the IHA file, so use the most current IHC you have.
- You must use double quotes when you enter the IHA and IHC file even if you do not have spaces in your file path or file name.
- Migrating an IHA will upgrade it to 4.5 format.
- If you are migrating a 4.5 IHA you should provide the IHC file in the Options window but if you do not have the IHC you can safely continue past the warning message.

## Creating a Batch File to Migrate Multiple IHA Files

The IHA Migration utility migrates only one archive at a time by design. However, if you need to add more than one archive at a time, you can create a batch file to automate multiple archive merges.

When creating a batch file you need to provide the batch file with a logical name and save the batch (`.bat`) file in a location that can be easily accessed using the command prompt.

> **Note:**
> When migrating any archive, you should start with the archive with the oldest data first, followed by newer data, in sequence, to minimize the amount of disk space used in the Data Archiver.

For example:

```
cd c:\Program Files\Historian

start /b /wait migrateiha /NOTHROTTLE /NOMESSAGES

"c:\Historian Data\Backups\server_Archive001.iha"

"c:\Historian Data\Backups\server_Config.ihc".
```

## Interoperability of Historian Versions

Interoperability guidelines for Historian versions include:

- Historian Collectors below v6.0 can write to Historian v7.0 Archivers; however, since the earlier collector versions cannot automatically connect to a mirror, users need to point those collectors to the mirror system.
- Historian Clients below v6.0 can retrieve data from Historian v7.0 Archivers.
- Historian v7.0 or later Clients can retrieve data from a single Historian Data Archiver below v6.0.
- Historian v7.0 or later Collectors can write to a single Historian Data Archiver below v6.0.
- An SDK program built on an Historian v7.0 or later node does not run on an Historian below v6.0.
- An SDK program that you created in Historian below v7.0 must be rebuilt on a computer with Historian v7.0 or later if you want to run it on that version.
- It is recommended that you use consistent versions of client and server applications. If you do use different client and server versions of the Historian, regularly back up all archives and tag configurations.

**Note:**
To determine the version of the server, client, and SDK, select the **About** link in Historian Administrator. The version of the Historian installer can be seen in the **Control Panel / Uninstall programs**; this version is different from the Historian core version seen in Historian Administrator **About** link.

## Migrate User Authentication Data from Historian to Common Proficy Authentication Service

Starting Historian 8.0 version, you can use the common Proficy Authentication service.

**Note:**
You can either choose the common Proficy Authentication deployed by any other products such as, Operations Hub, Plant Apps; or the common Proficy Authentication deployed by Historian.

To use the common Proficy Authentication service, you must migrate the Proficy Authentication data should from Historian to the new local or external Proficy Authentication.

To enable this migration of data the `uaa_config_tool` is introduced.

**Note:**

- While migrating Proficy Authentication details, we are setting default password as user123 for all the users. You should change it using Proficy Authentication config tool once migration is done.
- While setting up new password, it may ask you to enter the port number. By default, the port number is 443. If you have provided a different value in the Public https port field in the TCP port assignments page while installing Web-based Clients, you must provide that value.
- Using this tool, you can back up the Proficy Authentication data only for Historian 7.2 or earlier. And, you can migrate the data to Historian 8.0 or later.

  **Important:**
  Back up the data before upgrading to the latest version of Web-based Clients.

It includes the following tasks:

1. Back up Historian Proficy Authentication data.
2. Migrate the data to an existing common Proficy Authentication service.

1. Open Command Prompt.
2. Mount the ISO and navigate to the Utilities folder where the `uaa_config_tool.exe` file is located. After installation, uaa_config_tool is available in the following folder as well: `<installation drive of Historian>\Program Files\GE Digital\Historian Config`

   **Important:**
   You should back up the data before upgrading to 8.0 (Web-based Clients)

3. Enter the following command to take a back up of the data:

   ```
   C:\ uaa_config_tool.exe backup_data -d "<destination folder>"
   ```

   `<destination folder>` is the location where you want to save the back up files. Note that you must enter the value in quotes.

> **Note:**
> You may be prompted to enter the password twice for Historian Database and Proficy Authentication. Enter the password as GEIP123User

Back up is generated and the data is saved in the destination folder.

4. Copy the backup files to the machine where Web-based Clients are installed.
5. Enter the following command to migrate the data:

```
uaa_config_tool.exe migrate_data -h <host-addr> -m <portnum> -u <username> -s <secret> -l "<location path of backup files>"
```

Example: `uaa_config_tool.exe migrate_data -h vmhistwin2016 -m 443 -u admin -s gowt43df -l "c:\myuaabackupfiles"`

> **Note:**
> While migrating Proficy Authentication details, it may ask for port number at migrating historian database. The default value is 8432. If, however, you have provided a different value in the **Historian database port** field in the **TCP port assignments** page while installing Web-based Clients, provide that value.

| Value | Description |
|---|---|
| <host-addr> | Address of the host or destination where you want to migrate the data. |
| <portnum> | Port number of the destination. |
| <username> | Username |
| <secret> | Client Secret |
| <location path of backup files> | Source location where the data is backed up. |

- The data is migrated to the destination that is, common Proficy Authentication service.
- The `umt log` file is generated containing the details about backup and migration for users and groups.
- The `User_migration_report` is generated which contains the migration status of the user data.

**Note:**
Creation and migration of users and groups can fail of the user or group already exists in the destination.

If user migration fails, you can change the username in the backup file and repeat Step 5.

# Implementing Historian Security

## Implementing Historian Security

Historian is a high performance data archiving system designed to collect, store, and retrieve time-based information efficiently. By default, access to these Historian archives, tags, and data files is available to any valid operating system user account. In this default environment, all users are allowed to read, write, change, and delete archives, tags, or data files in Historian Administrator, SDK, migration tools, and Excel Add-In. However, you may want to make these features and data available only to authorized personnel. You can do this by creating and defining Historian security groups in your Windows Security.

Historian includes an Electronic Signature and Electronic Records security feature. This option provides installations related to the FDA's 21 CFR Part 11 regulation or any site interested in added security or tracking the ability to require a signature and password every time a change in data or configuration is requested. For more information on the Electronic Signature and Electronic Records feature, refer to Historian in a Regulated Environment *(on page 824)*.

To ensure a secure environment when using Historian security, do not create any local user accounts unless Historian is set up on a standalone machine.

Whether or not you use Historian security, make sure that you disable Guest accounts on your computer to limit access to valid Windows user accounts.

To run Proficy Authentication commands, refer to Managing Proficy Authentication Users Using the Configuration Tool *(on page 284)*.

## About Protecting Your Process

If you want to restrict access to Historian archives, files, and tags, or protect your data files from unauthorized changes, you can enable Historian security. Using security is optional and is disabled by default. By enabling security, you can restrict access to:

- Modifying data using Excel Add-In
- Updating security for individual tags or groups of tags

- Creating, modifying, and removing tags
- Tag protection (adding, modifying, removing, and so on) can be applied at a global level to all tags or at the individual tag level.

  Refer to Implementing Tag Level Security for more information.
- Reading data in the iFIX Chart object, Excel Add-In, and Migration Utilities
- Writing data
- Starting and stopping collectors
- Creating and deleting collectors
- Creating, modifying, and deleting archives

Historian uses the operating system security groups to create a security structure. You can enable security for a particular set of functions by adding specific Historian Security Groups to your groups. You can also add security groups to your domain controller.

By defining one or all of the groups, you begin to set up a security structure. Refer to the *Historian Security Groups* section for more information on the Historian Security Groups available.

## Strict Authentication

With Historian's strict user account authentication features, `Enforce Strict Client Authentication` and `Enforce Strict Collector Authentication`, you can control access to the Historian server and safeguard user account credentials.

With strict authentication enabled, only known user accounts configured on the Data Archiver server computer will be able to access a Historian server. Similarly, enabling strict collector authentication enforces the same requirement for incoming collector connections.

For an account to be known at the Data Archiver, it has to exist on that archiver as a local account or exist on a Domain Controller available to the data archiver. Historian will access the local accounts or Domain Controller via Microsoft’s Security Support Provider Interface (SSPI) and this involves having a Kerberos server setup optionally to assist in account validation.

By default, strict client and collector authentication is enabled on new installations to maximize security. When upgrading from a previous version of Historian, strict client and collector authentication is disabled to allow compatibility with older clients or collectors that cannot be upgraded concurrently.

It is recommended that all clients and collectors receive timely upgrade to the latest version, which permits enabling both strict client and collector authentication on the server for the highest security configuration.

By treating clients and collectors separately, it is possible to accommodate new and legacy authentication during the upgrade process. However, upgrading all clients and collectors to the latest version immediately will achieve a high level of security. The two options, Enforce Strict Client Authentication and Enforce Strict Collector Authentication, permit flexibility during the upgrade process by selectively accommodating legacy clients and collectors.

**Local and Domain Security Groups:**

You can choose local or domain security groups to access Historian. To do so, in **Historian Administrator > Data Stores > Security**, select **Use Local** or **Use Domain**. The following table provides recommended group to use based on the machine configuration and the security group of the logged-in user.

| Machine Configuration | Security Group of the Logged-In User | Recommended Security Group |
|---|---|---|
| Workgroup | Local | Local |
| Domain | Local | Domain<br><br>For domain machines, we recommend that you log in with a domain-level user and create security groups in the domain controller machine. |
| Domain | Domain | Domain |

**Strict Authentication Options:**

This table provides guidelines about the different combinations of strict client and collector authentication options and their use:

| Strict Client Authentication | Strict Collector Authentication | Comment |
|---|---|---|
| Enabled | Enabled | Use this for highest available security. You will need to install SIMs, if available on all pre-6.0 collectors and clients. Clients can refer to any program that connects to the Data Archiver. This includes Historian Administrator, Microsoft Excel, any OLE DB program, user written programs, or any other Proficy software. |

| Strict Client Authentication | Strict Collector Authentication | Comment |
|---|---|---|
| Enabled | Disabled | Use this if you are unable to upgrade collectors to the latest version if there is no SIM update for your collector. |
| Disabled | Enabled | Use this if you have to support legacy clients and you are unable to install the SIM update on all clients. |
| Disabled | Disabled | Use this for maximum compatibility with existing systems. |

**Trusted Connections in Distributed Historian Service Environment:**

This trusted connection works only in the Domain environment and it is enabled by default.

**Note:**
If you are adding a mirror copy to an existing node, make sure that both the nodes are in the same domain.

If you want to work in the workgroup setup, contact Online technical support & GlobalCare:www.digitalsupport.ge.com.

## Disabling Strict Client and Collector Authentication

To permit older versions of clients and collectors to access a Historian 7.0 (or later) server, disable strict client and collector authentication.

1. Open the page and select **DataStore Maintenance Security**.
2. In the **Global Security** section:
    - Select the **Disabled** option button for **Enforce Strict Client Authentication**.
    - Select the **Disabled** option button for **Enforce Strict Collector Authentication**.

## Security Strategy Guidelines

When you begin to implement security, you should first define a clear strategy. Consider the following when beginning to set up your security strategy:

- If you disabled the Guest account, a user must provide a valid username and password even if no groups are created.
- Protection is only provided for the functional areas for which you have built the associated Historian Security Groups.

- If you only choose to define some of the security groups, all users still have all access to any uncreated groups. All users are still assumed to be a member of a group unless that group has been created, with the exception of iH Audited Writers group. You must add the iH Audited Writers group to the Windows security groups so that a user can become a member of this group.

  For example, if you elect to define the iH Security Admins group and iH Archive Admins group, both the members associated with those defined groups and all other valid users still have access to such functions as creating and modifying tags until you create the iH Tag Admins security group.
- If you implement any Historian Security groups, you must first add and define the iH Security Admins group.

> **Note:**
> If you do not create and define the iH Security Admins group, all valid users are assumed to be members of this group. This membership overrides any other security group that you set.

See also Historian Security Groups *(on page 256)*.

## Setting Historian Login Security

Use Historian Login Security settings if you want to validate users at the Data Archiver, instead of at the client. By applying these settings, users and applications are forced to provide a user name and password at connect time so that the archiver can validate them. For example, users in the security group such as `ih Security Admins` will be checked by the Archiver.

For Historian Login Security settings, you can view and set the property from the HistorianSDKsample server properties. The current setting is shown in the data archiver `SHW` file.

Historian Login Security property is available only in Historian SDK.

To set login security using the Historian SDK:

1. Run the SDK sample.
2. Connect to a server.
3. Select the server in the list box.
   The **Server Properties** window appears.
4. On the right side of the window, locate the **AllowClientValidation** setting. By default, this value is set to `TRUE`. Select to set to `FALSE`, and select **OK**.

## Historian Security Groups

Historian provides the following security groups:

**iH Security Admins**

Historian power security users. Security Administrators have rights to all Historian functions. This group also has the ability to change tag level security, archive security, and modify the Electronic Records and Signatures option. This is the only Historian security group that overrides tag level security.

**iH Collector Admins**

Allowed to start and stop collectors, browse collectors, configure collectors, and add new collectors.

**iH Tag Admins**

Allowed to create, modify, and remove tags. Tag level security can override rights given to other Historian security groups. Tag Admins can also browse collectors.

iH Tag Admins are not responsible for setting Tag Level Security. This task can only be performed by an iH Security Admins. For more information on setting Tag Level Security, refer to the *Implementing Tag Level Security* section.

**iH Archive Admins**

Allowed to create, modify, remove, backup, and restore archives.

**iH UnAudited Writers**

Allowed to write data without creating any messages.

**iH UnAudited Logins**

Allowed to connect the DataArchiver without creating login successful audit messages.

**iH Audited Writers**

Allowed to write data and to produce a message each time a data value is added or changed.

Tag, archive, and collector changes log messages regardless of whether the user is a member of the iH Audited Writers Group.

**iH Readers**

Allowed to read data and system statistics. Also allowed access to Historian Administrator.

Use this table to identify the types of user groups you need to create and define in your security system.

| Function | iH Security Admins | iH Un-Audited Writers | iH Un-Audited Login | iH Audited Writers | iH Readers | iH Archive Admins | iH Tag Admins | iH Collector Admins |
|---|---|---|---|---|---|---|---|---|
| Create Tags:<br>• Excel Add-In<br>• SDK<br>• Historian Admins<br>• File collector | X | | | | | | X | |
| Remove Tags:<br>• Historian Admins<br>• SDK | X | | | | | | X | |
| Modify Tags: | X | | | | | | X | |

| Function | iH Security Admins | iH Un-Audited Writers | iH Un-Audited Login | iH Audited Writers | iH Readers | iH Archive Admins | iH Tag Admins | iH Collector Admins |
|---|---|---|---|---|---|---|---|---|
| • Excel Add-In<br>• SDK<br>• Historian Admins<br>• File collector | | | | | | | | |
| Modify Archive Security:<br>• SDK<br>• Historian Admins | X | | | | | | | |
| Backup Archive:<br>• SDK<br>• Historian | X | | | | | X | | |

| Function | iH Security Admins | iH Un-Audited Writers | iH Un-Audited Login | iH Audited Writers | iH Readers | iH Archive Admins | iH Tag Admins | iH Collector Admins |
|---|---|---|---|---|---|---|---|---|
| an Admins | | | | | | | | |
| Restore Backup:<br>• SDK<br>• Historian Admins | X | | | | | X | | |
| Create Archive:<br>• SDK<br>• Historian Admins | X | | | | | X | | |
| Start/ Stop Collector:<br>• SDK<br>• Historian | X | | | | | | | X |

| Function | iH Security Admins | iH Un-Audited Writers | iH Un-Audited Login | iH Audited Writers | iH Readers | iH Archive Admins | iH Tag Admins | iH Collector Admins |
|---|---|---|---|---|---|---|---|---|
| Admins<br>• Mission Control (iFIX) | | | | | | | | |
| Browse Collector:<br>• Historian Admins | X | | | | | | | X |
| Read Data:<br>• Chart Object<br>• Excel Add-In<br>• SDK | X | | | | X | | | |
| Write Data (UnAudited): | X | X | X | | | | | |

| Function | iH Security Admins | iH Un-Audited Writers | iH Un-Audited Login | iH Audited Writers | iH Readers | iH Archive Admins | iH Tag Admins | iH Collector Admins |
|---|---|---|---|---|---|---|---|---|
| • Excel Add-In<br>• SDK | | | | | | | | |
| Write Data (Audited):<br>• Excel Add-In<br>• SDK | X | | | X | | | | |
| Modify Data:<br>• Excel Add-In<br>• SDK | X | X | X | X | | | | |
| Update Security for Tag:<br>• Excel Add-In<br>• SDK<br>• Histo- | X | | | | | | | |

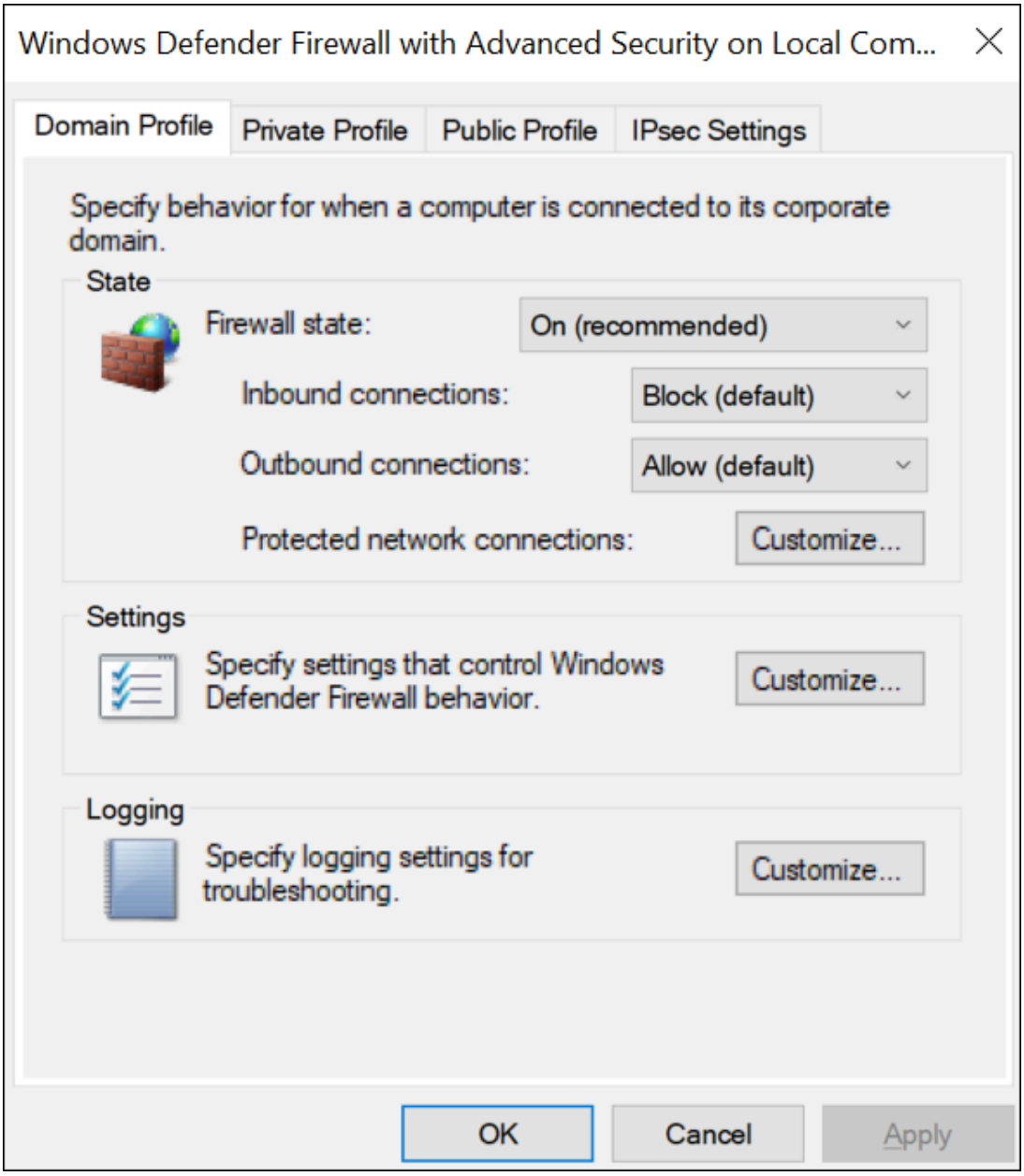
| Function | iH Security Admins | iH Unaudited Writers | iH Unaudited Login | iH Audited Writers | iH Readers | iH Archive Admins | iH Tag Admins | iH Collector Admins |
|---|---|---|---|---|---|---|---|---|
| rian Admins | | | | | | | | |
| Migrate:<br>• Migration Tools | X | | | | | | | |
| Login Connection Messages | X | X | | X | X | X | X | X |
| Recalculate Data | X | | X | X | | | | X |

## Configure Internet Protocol Security (IPSEC)

Historian supports encryption based on Internet Protocol Security to secure traffic between various Historian components and collectors without the need to use VPN or other security protocols.

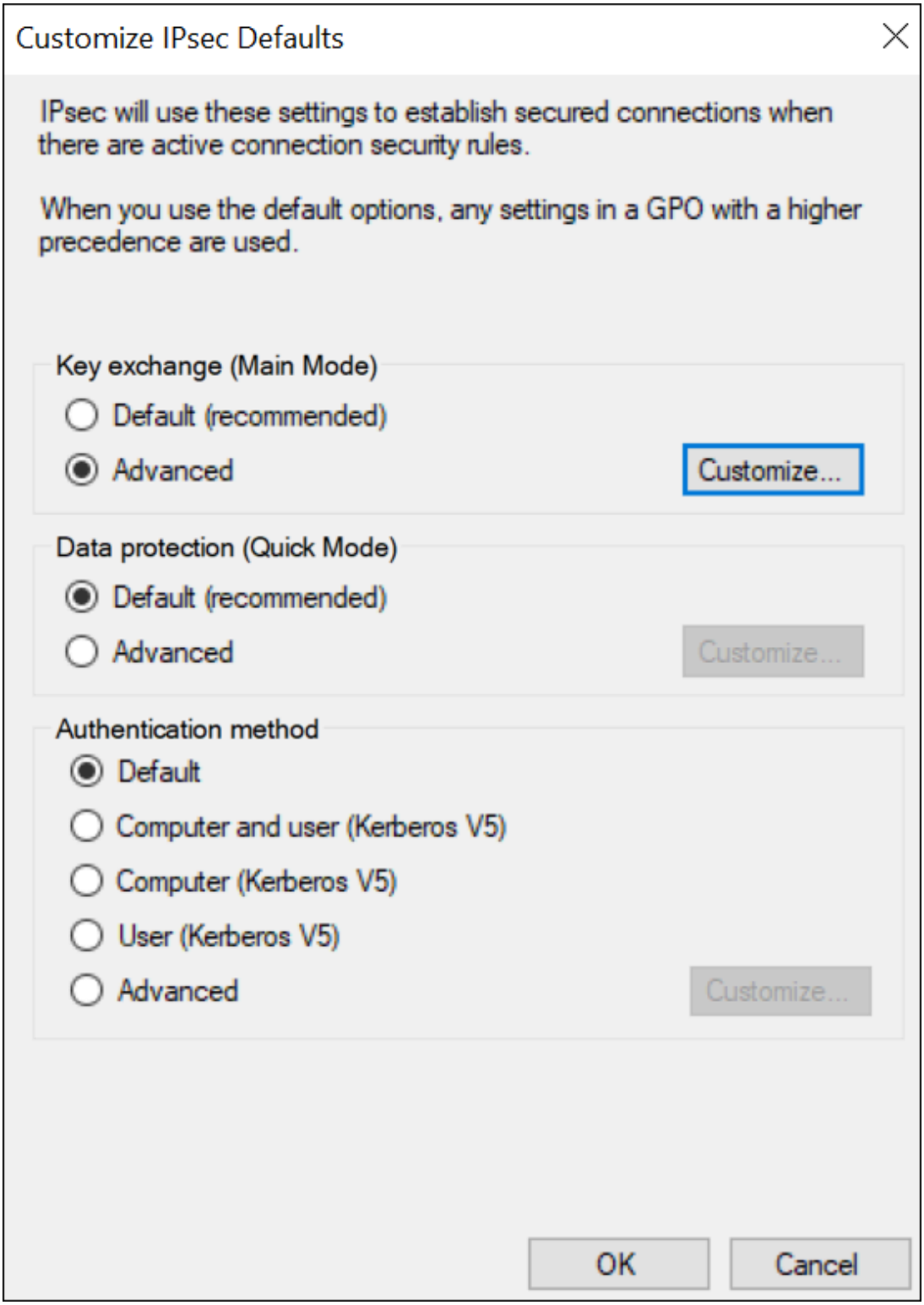
1. Run `wf.msc`.
   The **Windows Defender Firewall with Advanced Security** window appears.
2. Create a security method:

   a. Select **Actions > Properties**.
   The **Windows Defender Firewall with Advanced Security on Local Computer** window appears.



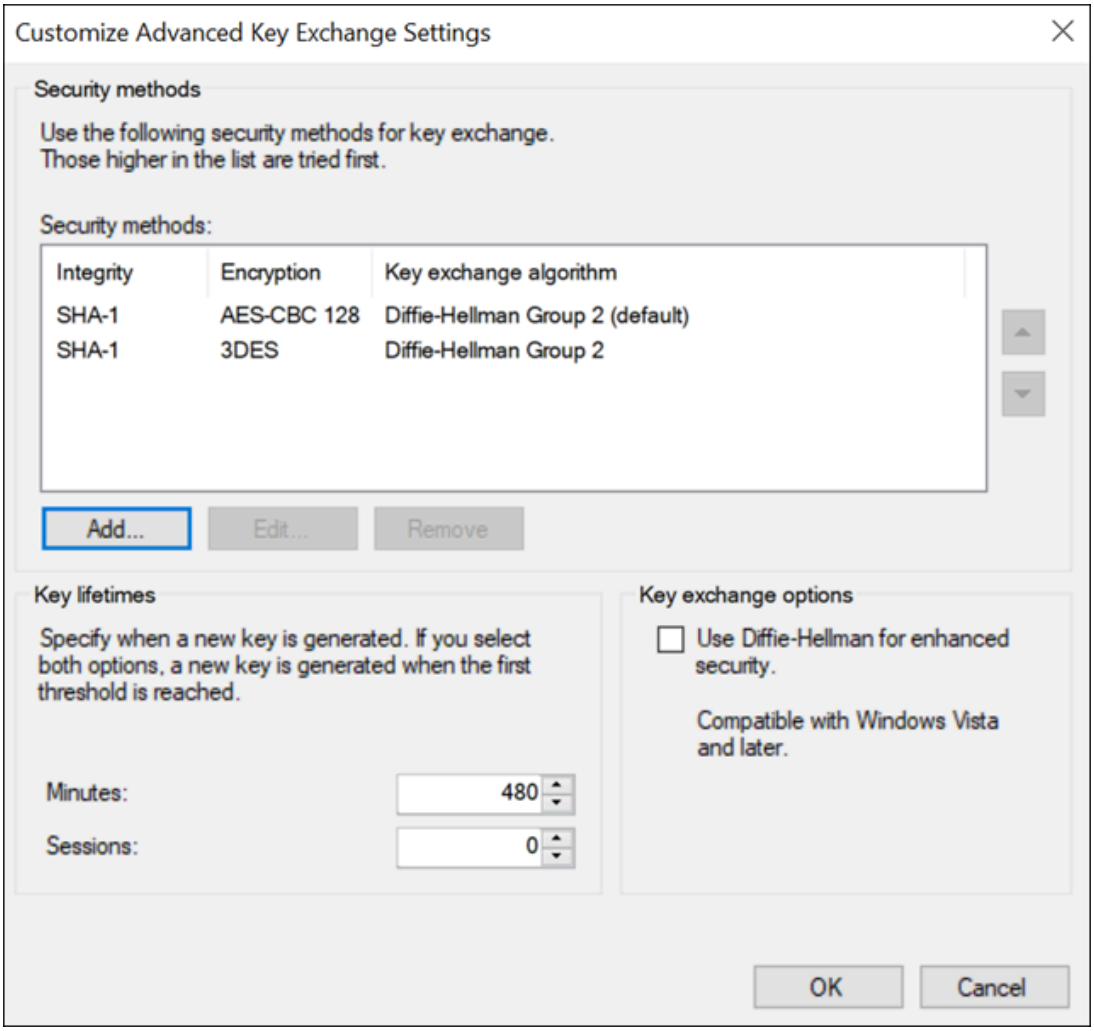

b. Select **IPsec Settings > Customize**.

The **IPsec Defaults** window appears.




c. Under **Key exchange (Main Mode)**, select **Advanced > Customize**.

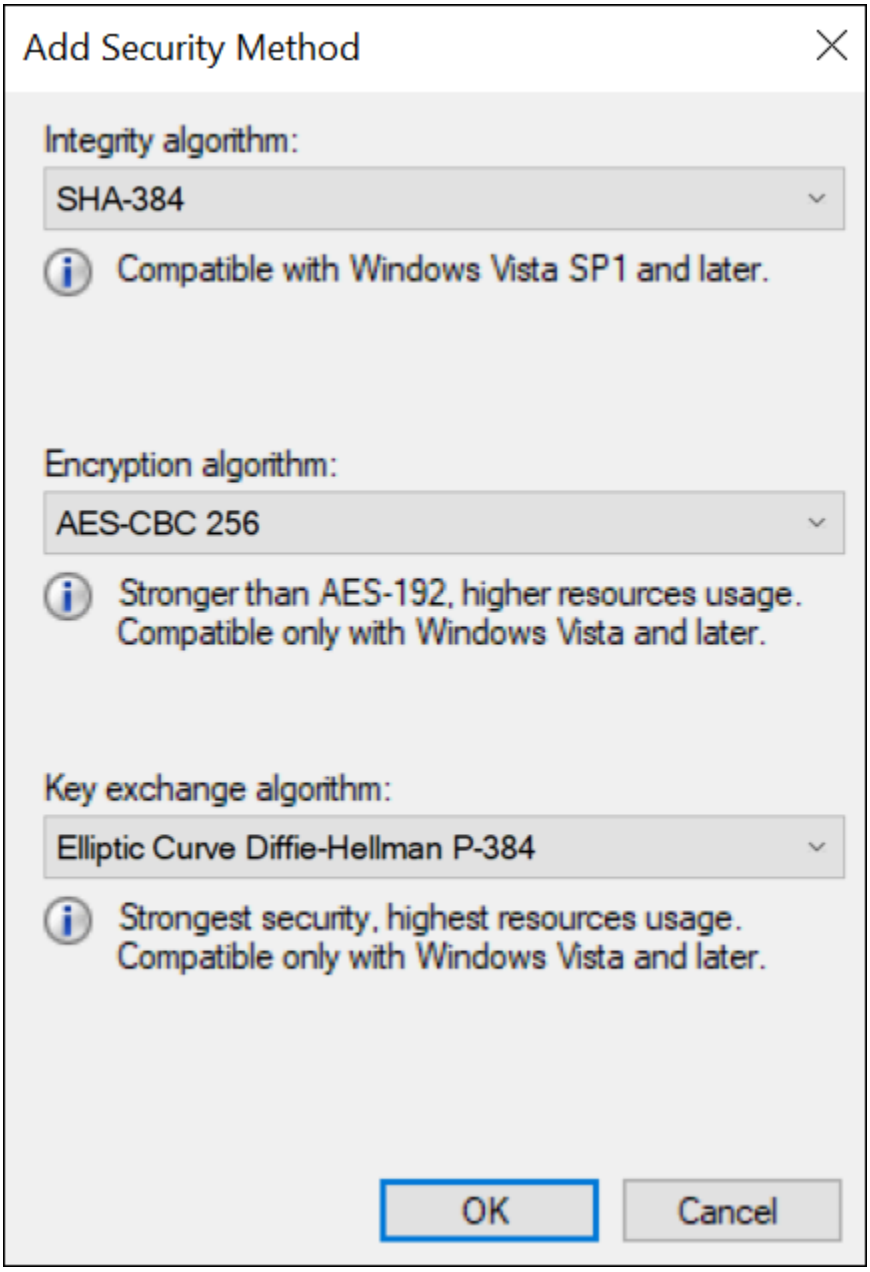
The **Customize Advanced Key Exchange Settings** window appears.




d. Select **Add**.

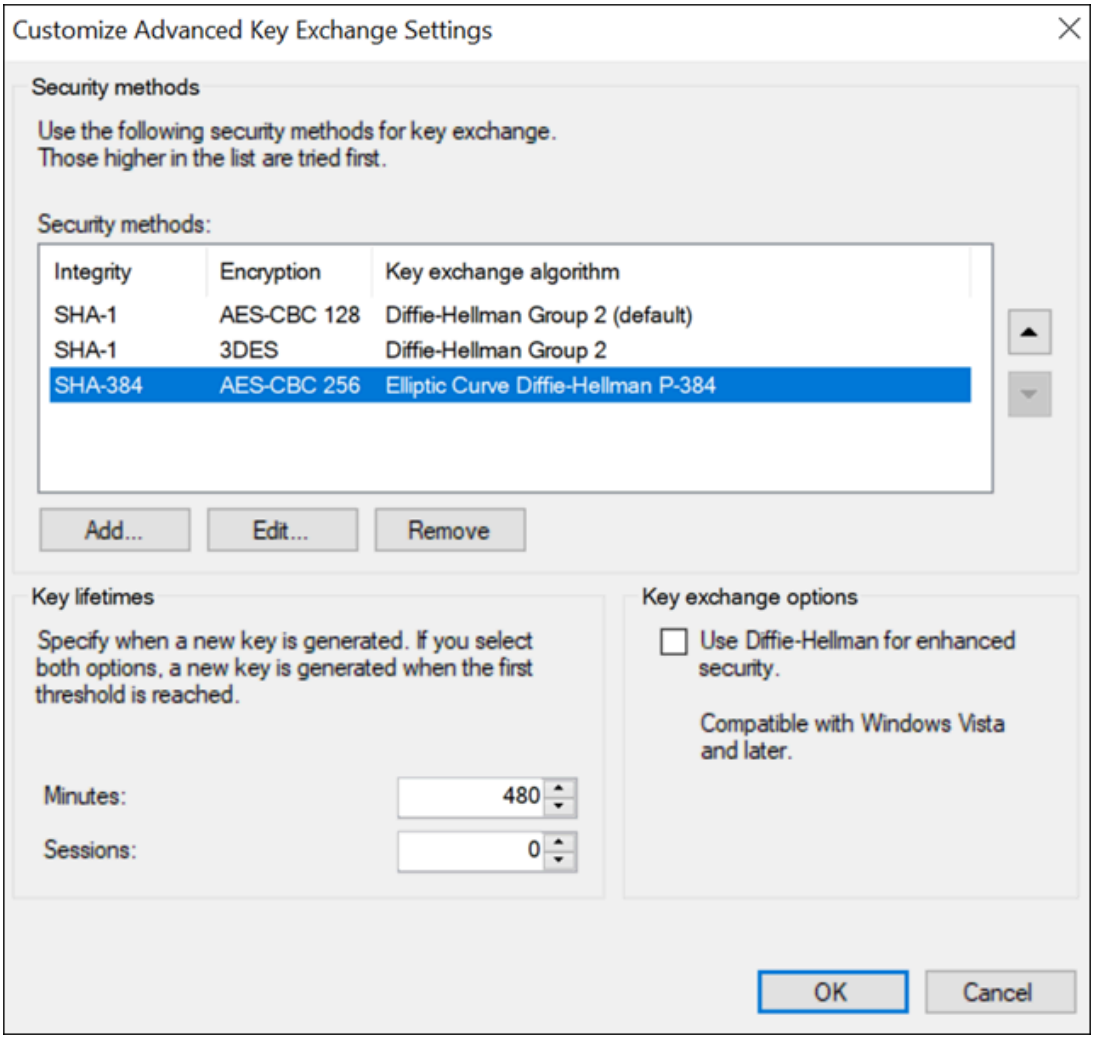
   The **Add Security Method** window appears.

e. Select the algorithms that you want to use for each purpose. The following image shows an example.




**Important:**
You must provide the same values for all the machines for which you want to configure IP security.

The security method that you have added appears in the list.

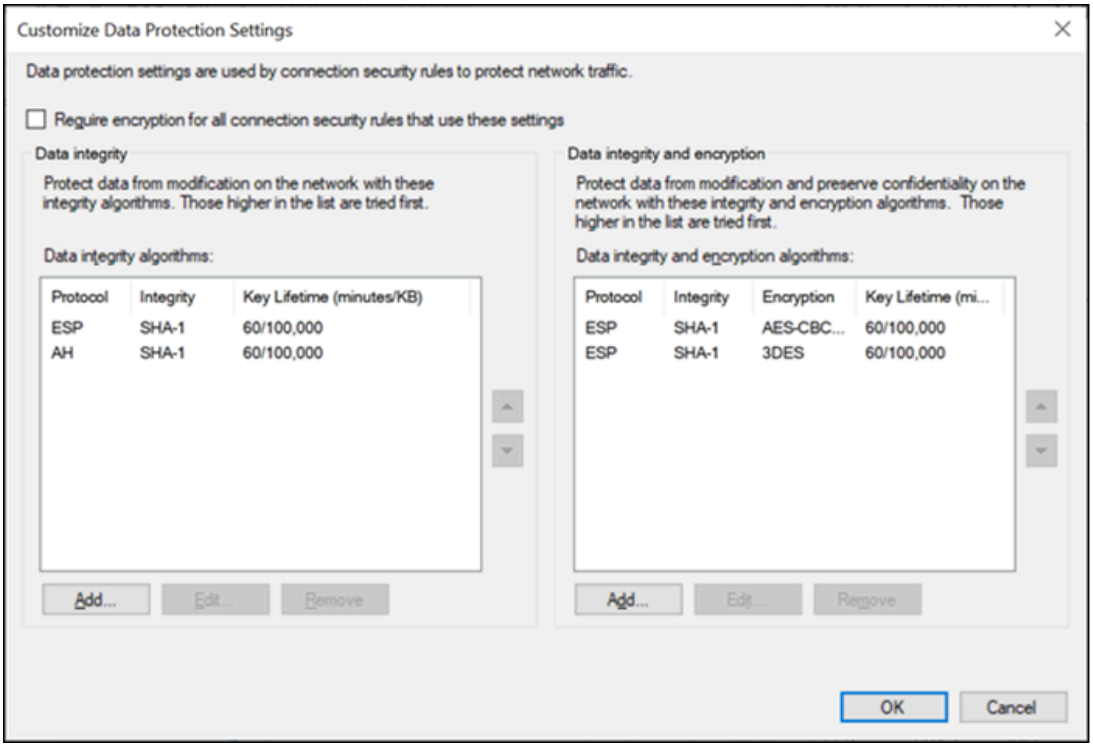



f. Move the security method that you have added to the top of the list. We recommend that you remove the other methods.

g. Select **OK**.

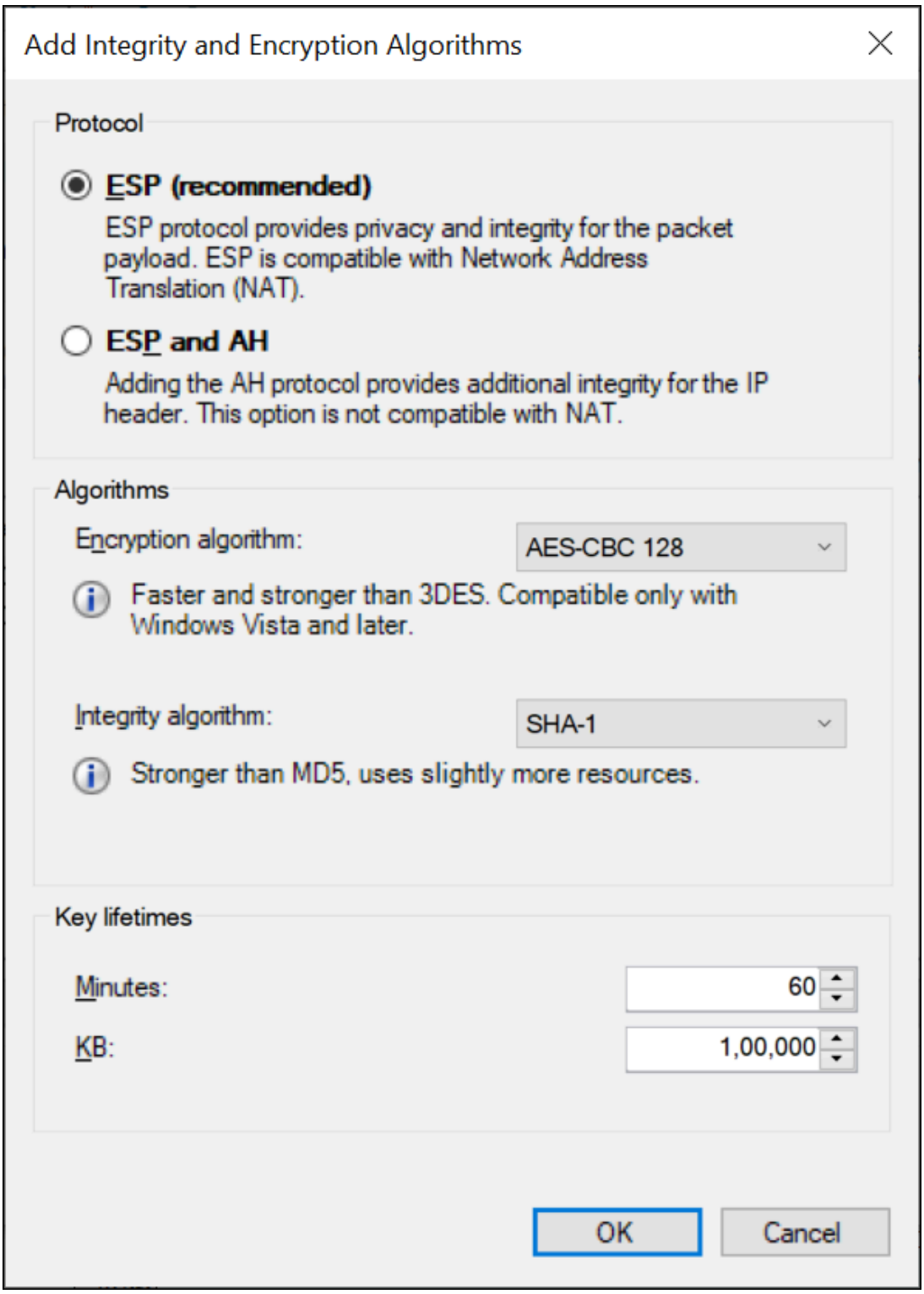
3. Add integrity and encryption algorithms:

a. In the **Customize IPsec Defaults** window, under **Data protection (Quick Mode)**, select **Advanced > Customize**.
The **Customize Data Protection Settings** window appears.




b. Select the **Require encryption for all connection and security rules that use these settings** check box.

c. Under **Data integrity and encryption**, select **Add**.
The **Add Integrity and Encryption Algorithms** window appears.




d. Under **Protocol**, ensure that **ESP** is selected.

e. Select the algorithms that you want to use for each purpose, and then select **OK**.

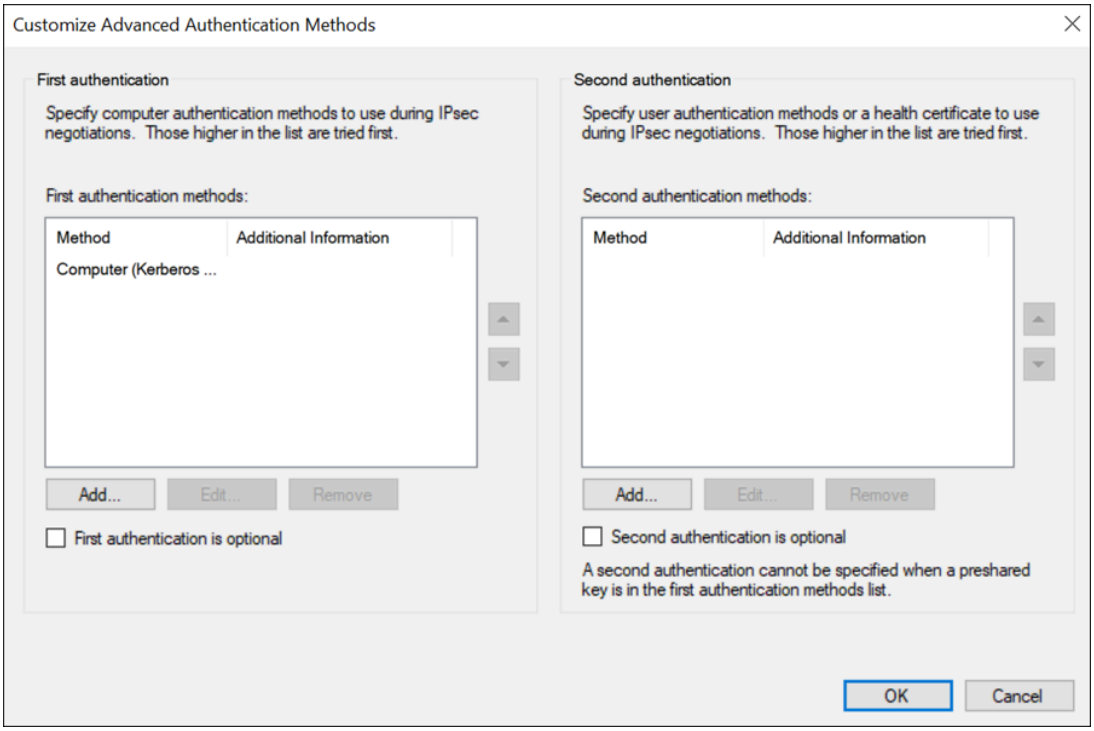
The algorithms that you have selected appear in the list.

f. Move the algorithms to the top of the list. We recommend that you remove the remaining items in the list.

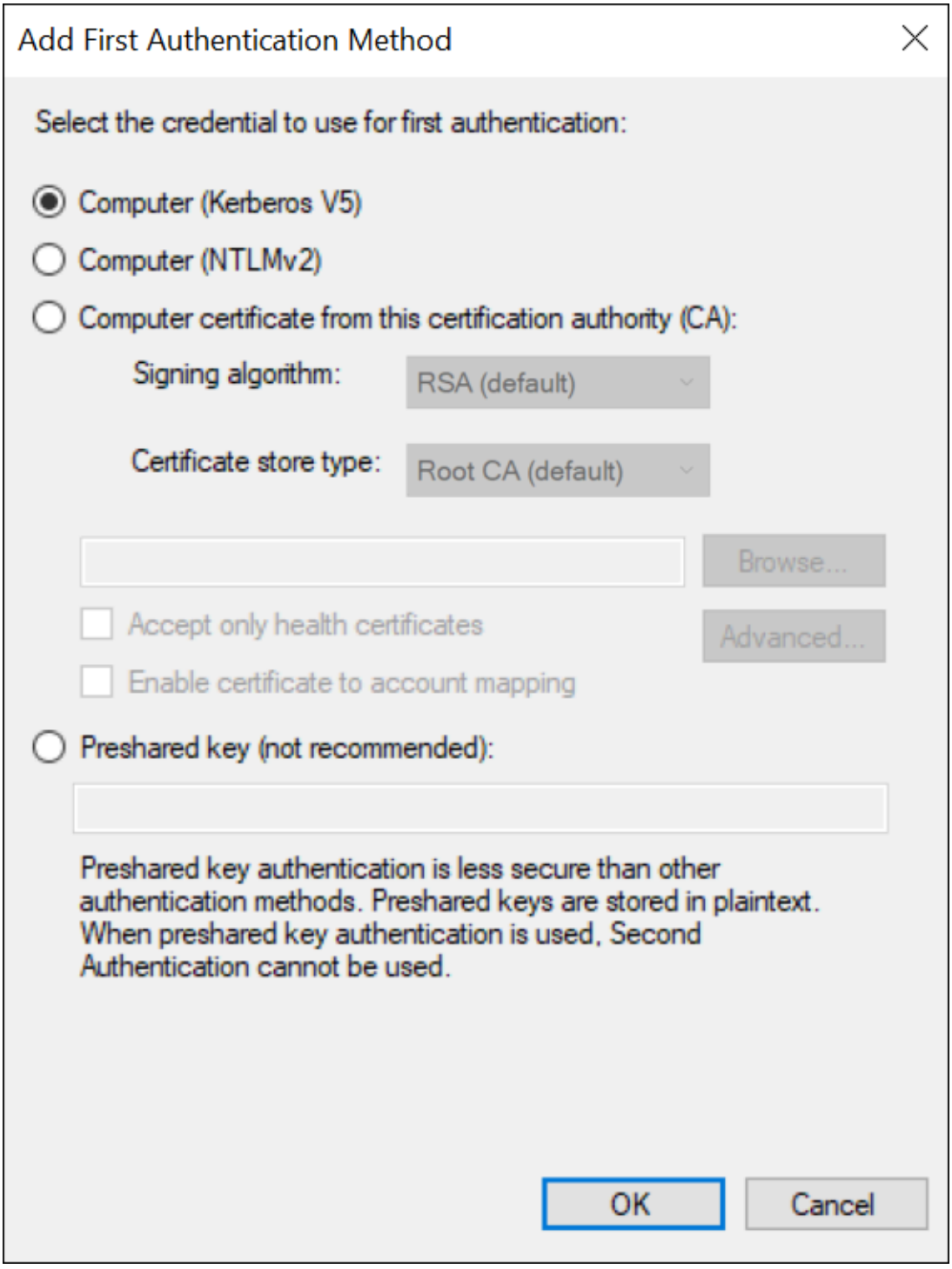
g. Select **OK**.

4. Create a first authentication method:

a. In the **Customize IPsec Defaults** window, under **Authentication Method**, select **Advanced > Customize**.
The **Customize Advanced Authentication Methods** window appears.




b. Under **First authentication methods**, select **Add**.
The **Add First Authentication Method** window appears.




c. Provide the CA certificate that you want to use, and then select **OK**.

The certificate that you have provided appears in the list.

d. Move the certificate to the top of the list. We recommend that you remove the remaining items in the list.

e. Select **OK**.

5. Create a connection security rule:

   For Windows x86, run the following set of commands to create a rule:

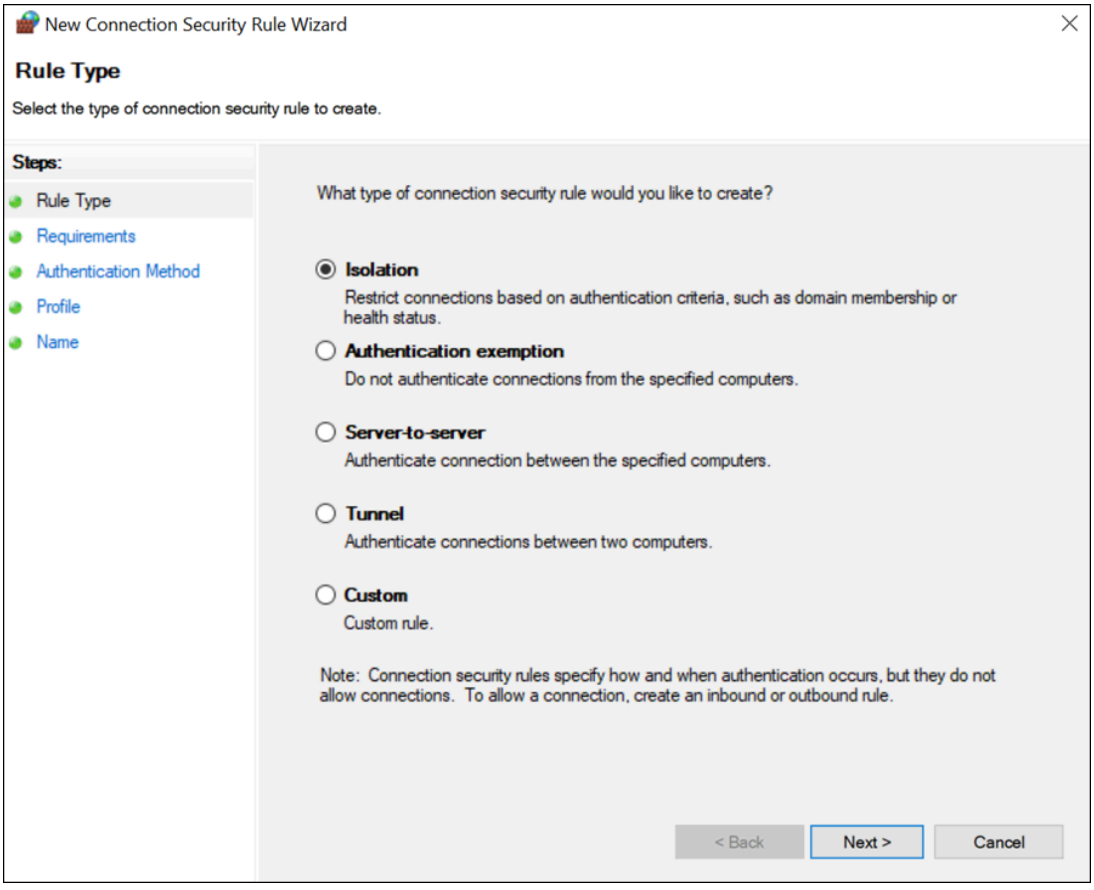
```
netsh  advfirewall

consec

add rule name=""<rule name>"" endpoint1=any endpoint2=any protocol=tcp port1=any port2=2010

action=requestinrequestout
```

   For other versions, perform the following steps:

   a. In the **Windows Defender Firewall with Advanced Security** window, select **Connection Security Rules**.

   b. Select **Actions > New Rule**.

   The **New Connection Security Rule Wizard** window appears.



c. Select **Custom**, and then select **Next**.

d. Both for Endpoint 1 and Endpoint 2, select **Any IP Address**, and then select **Next**.

e. Select **Require authentication for inbound and outbound connections**, and then select **Next**.

f. Select **Default**, and then select **Next**.

g. Enter values as described in the following table, and then select **Next**.

| Field | Description |
|---|---|
| **Protocol type** | Select **TCP**. |
| **Endpoint 1 port** | Select **All Ports**. |
| **Endpoint 2 port** | Select **Specific Ports**, and then enter 2010. |

h. Select when to apply the rule, and then select **Next**.

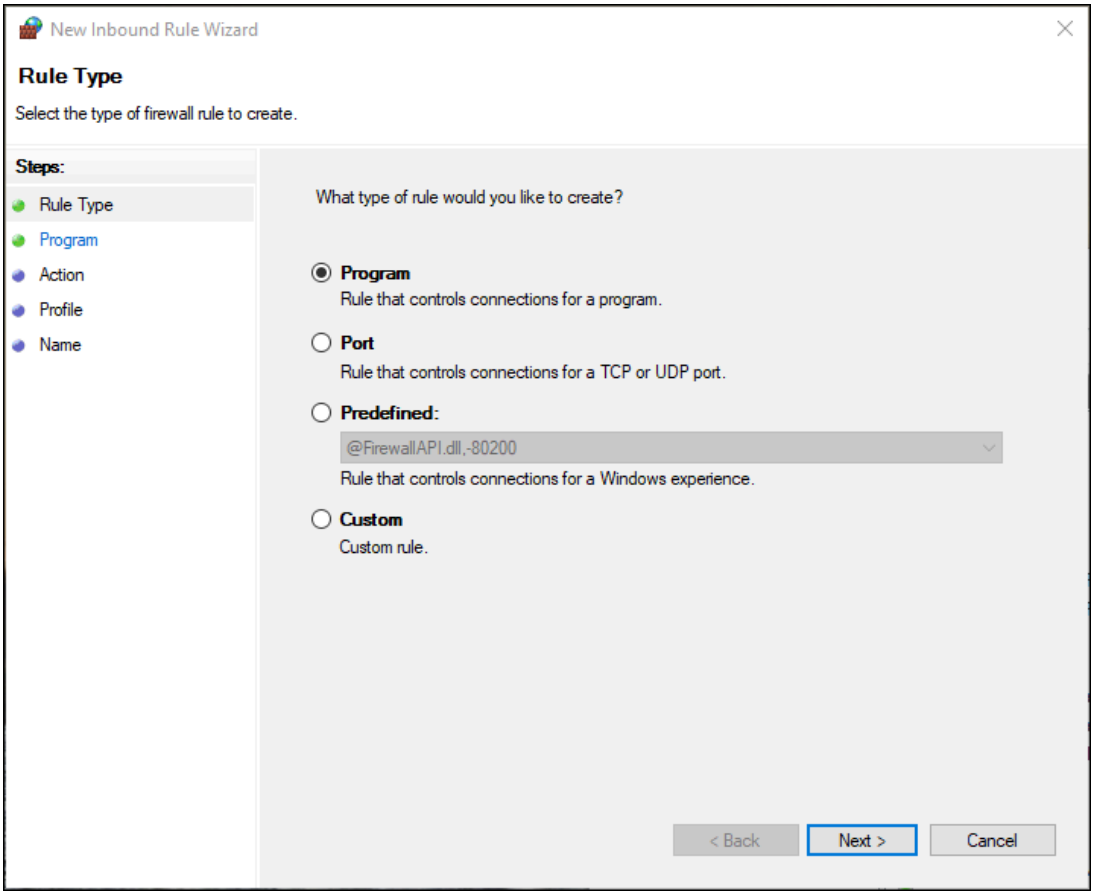
i. Enter a name and description for the rule, and then select **Finish**.
The rule appears in the **Connection Security Rules** window.

j. Ensure that the rule is enabled.

6. If using Microsoft Windows Server 2019, 2016, 2012 R2 and/or Windows 8, 8.1, open up port number 5000:

a. In the **Windows Defender Firewall with Advanced Security** window, select **Inbound Rules**.

b. Select **Actions > New Rule**.
The **New Inbound Rule Wizard** window appears.




c. Select **Custom**, and then select **Next**.

d. Select **All programs**, and then select **Next**.

e. Enter values as described in the following table, and then select **Next**.

| Field | Description |
|---|---|
| **Protocol type** | Select **UDP**. |
| **Protocol number** | Leave the default value as is. |
| **Local port** | Select **Specific Ports**, and then enter `5000`. |
| **Remote port** | Leave the default value as is. |

f. Both for the local and remote IP addresses, set the scope to **Any IP address**, and then select **Next**.

g. Select **Allow the connection**, and then select **Next**.

h. Select when to apply the rule, and then select **Next**.

i. Enter a name and description for the rule, and then select **Finish**.
The rule appears in the **Inbound Rules** window.

j. Ensure that the rule is enabled.

IPSEC is now configured on the machine.

7. Repeat all the steps above on all the machines that host the Historian server and/or its components/clients.
8. To verify that the IPSEC cryptography is used:

a. Ensure that the Historian server is running.

b. Ensure that the collectors are connected to the Historian server, and that the collectors are running.

c. Specify the tags for data collection. You can do so using Configuration Hub *(on page 357)* or Historian Administrator *(on page 909)*.

d. Verify that the collector is collected data.

e. On each machine on which you configured IPSEC, run `wf.msc`.
The **Windows Defender Firewall with Advanced Security** window appears.

f. Select **Monitoring > Security Associations > Main Mode**.
The **Main Mode** section displays the connection that you have created.

## Security Setup Example

The following example takes you through the process of establishing your security needs and defining and setting up the levels of security.

For this example, assume the following user needs in a plant of 14 users:

| User | Needs | Added to Security Group |
|---|---|---|
| USER1 | Power user. Needs total access to security. | iH Security Admins |
| USER2<br><br>USER3 | • Read/Write Data (no messages).<br>• Create, modify, and delete tags. | • iH UnAudited Writers<br>• iH Tag Admins |

| User | Needs | Added to Security Group |
|---|---|---|
| USER5<br>USER6<br>USER8 | • Backup, restore, and create archives.<br>• Connect to Data Archiver without creating login successful audit messages | • iH Archive Admins<br>• iH UnAudited Logins |
| USER4<br>USER7 | • iRead/Write Data (no messages).<br>• iCreate, modify, and delete tags.<br>• iStart/Stop Collectors.<br>• iBackup, restore, and create archives. | • iH UnAudited Writers<br>• iH Tag Admins<br>• iH Collector Admins<br>• iH Archive Admins |
| USER9-14 | Read Data. | iH Readers |

1. Establish the needs of your users. For this example, assume the user needs in a plant of 14 users, as described in the previous table.
2. Add and define the iH Security Admins Group.
   Once you determine that you want to establish a security structure, you must create and define the iH Security Admins group. This group of users is typically the "power users" of the Historian. Security Administrator rights allow them to manage configuration and give them free rein to the entire system. For this example, only USER1 would be added to the iH Security Admins group.
3. Establish and create any other Historian Security Groups as needed.

   **Note:**
   Any user with Windows administrative permissions can add or remove Windows groups and users. As such, an administrator on a Windows computer, can add himself to any Historian security group.

   Set up the functional security groups as needed. For this example, Write, Tag, Archive, and Collector security is required, so the groups associated with those functions should be added and defined. There is no need for Audited Writers and all valid users can read data, so neither the iH Audited Writers Group nor the iH Readers Group need to be added.
4. Define any individual Tag Level security.

   In addition to defining iH Tag Admins that have the power to create, modify, and remove tags, you can also define individual tag level security to restrict access to sensitive tags. You can grant read, write, or administrative privileges per tag. For more information on setting Tag Level security, refer to the *Implementing Tag Level Security* section.

## Setting Up Historian Security Groups

This section describes how to add the Historian Security Groups to your local and domain Windows security systems.

You can choose whether Historian uses LOCAL or DOMAIN security by selecting an option on the **Security** section of the **Data Store Maintenance** page in Historian Administrator. If you select the local security option, the groups are defined as local groups on the Historian server. If you select the Domain security option, the groups are defined as global groups in the primary domain controller of the Historian server. With domain security, Historian locates the Primary Domain Controller (PDC), if available, or a Backup Domain Controller (BDC) in order to establish groups. If the PDC and all BDCs are unavailable, the system locks all users out until rights can be established with a valid PDC or BDC.

**Note:**
If you change this setting, you must stop and re-start the Historian server for this change to take effect.

## Creating a Local Group on Windows

1. Do a Windows search and open **Computer Management**.
   The **Computer Management** console appears.
2. In the left-side pane, expand **System Tools**, and then expand **Local Users and Groups**.
3. Select **Groups** in the system tree.
   The list of all the available groups appear.
4. In the right-side pane, select **Groups** > **More Actions**, and then select **New Group**.
   The **New Group** window appears.
5. Enter the Historian Security Group name in the **Group Name** field.
   For a list of available Historian Security Groups and their functions, see Historian Security Groups *(on page 256)*.

   **Note:**
   You must enter the Historian Security Group name exactly as it appears. The security groups are case sensitive.

6. Optionally, enter a description of the Historian Security Group in the **Description** field.
7. Select **Create**.
8. Select **Close**.

## Adding Users to Windows Security Group

Add your users to the Windows system.

1. Do a Windows search and open **Computer Management**.
   The **Computer Management** console appears.
2. In the left-side pane, expand **System Tools**, and then expand **Local Users and Groups**.
3. Select **Groups** in the system tree.
   The list of all the available groups appear.
4. Select the group to which you want to add users.
5. In the right-side pane, select the user > **More Actions**, and then select **Properties**.
   The user **Properties** window appears.
6. Select **Add**.
7. Select the users or groups to add from the listed users or enter the names of the users or groups you want to add in the bottom field.
8. Select **Add**.

   **Note:**
   To validate the user or group names that you are adding, select **Check Names**.

9. When you have added all users to the group, select **OK**.

## Adding a Local or a Domain User

1. Verify object type is **Users** or **Groups**.
2. If you want to add a local user, verify the **From This Location** setting is your local machine. (Select **Locations** to specify the local machine, if required.). If you want to add a domain user:
   a. Select **Locations** to specify the domain, if required.
   b. Select **Entire Directory** or the specific domain underneath **Entire Directory**.
   c. Select **OK**.
3. Select **Advanced**.
   The **Advanced** window appears.
4. Select **Find Now**.
5. From the list of users, select the users or groups to add or enter the names of the users or groups you want to add in the bottom field.
6. In the **Advanced** window, select **OK**.
7. In the **Select Users** window, select **OK**.
8. In the **Group Properties** window, select **OK**.

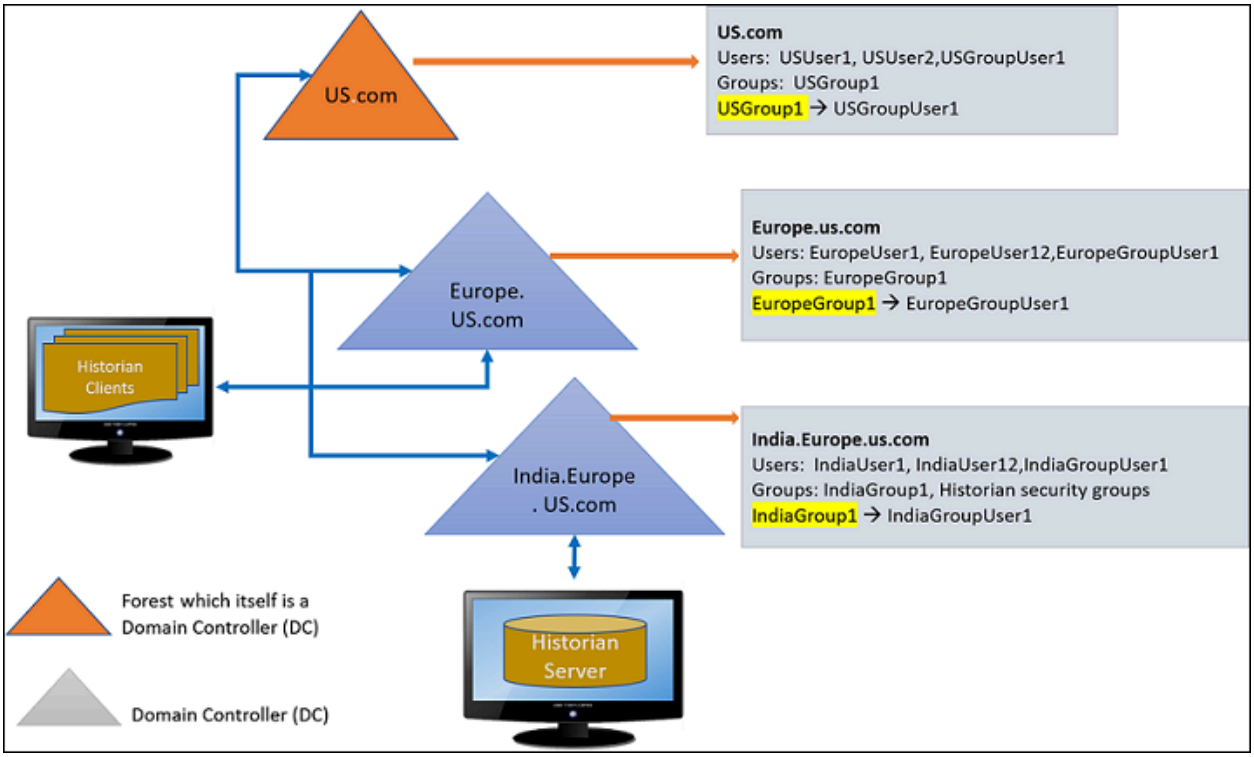
## Active Directory Setup - an Overview

Historian Active Directory setup supports integration with complex models that include the following complexities:

- Users and administrators may belong to different domains within a forest.
- Domains may have sub-domains (multi-level) that need to inherit or refine on inherited permissions
- Group names may be longer than average to cater for group differentiation

The Active Directory setup supports authentication and authorization of users as members of groups from trusted or sub-domains (including assigning appropriate Historian access rights in line with Historian security roles/groups access).

The following figure provides an overview of the Active Directory setup with examples:




## Configuring the Domain Users for active directory setup

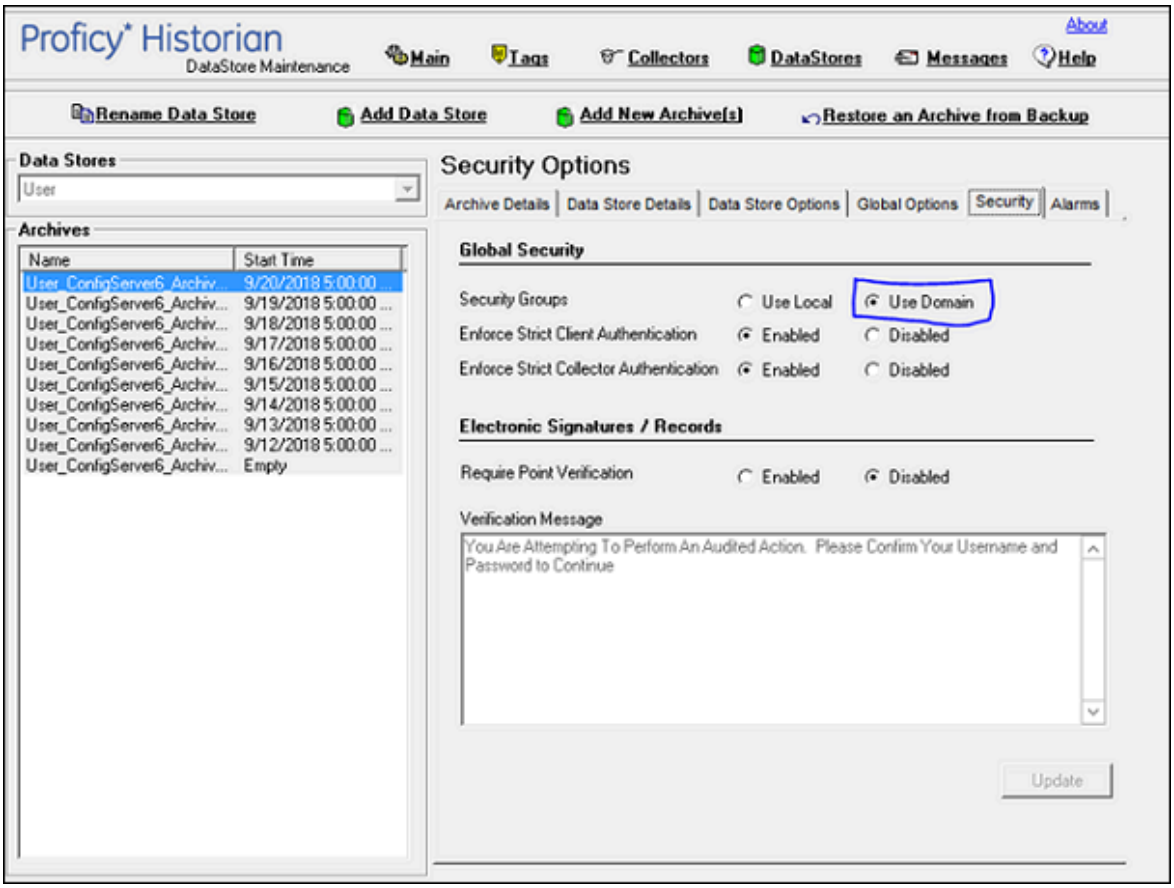
To configure the domain (single\multi) environment in Historian Administrator:

Historian Security Groups should be created on the machine where the Domain controller of the Historian Server is installed. The India.Europe.US.com domain controllers must contain the following Historian groups:

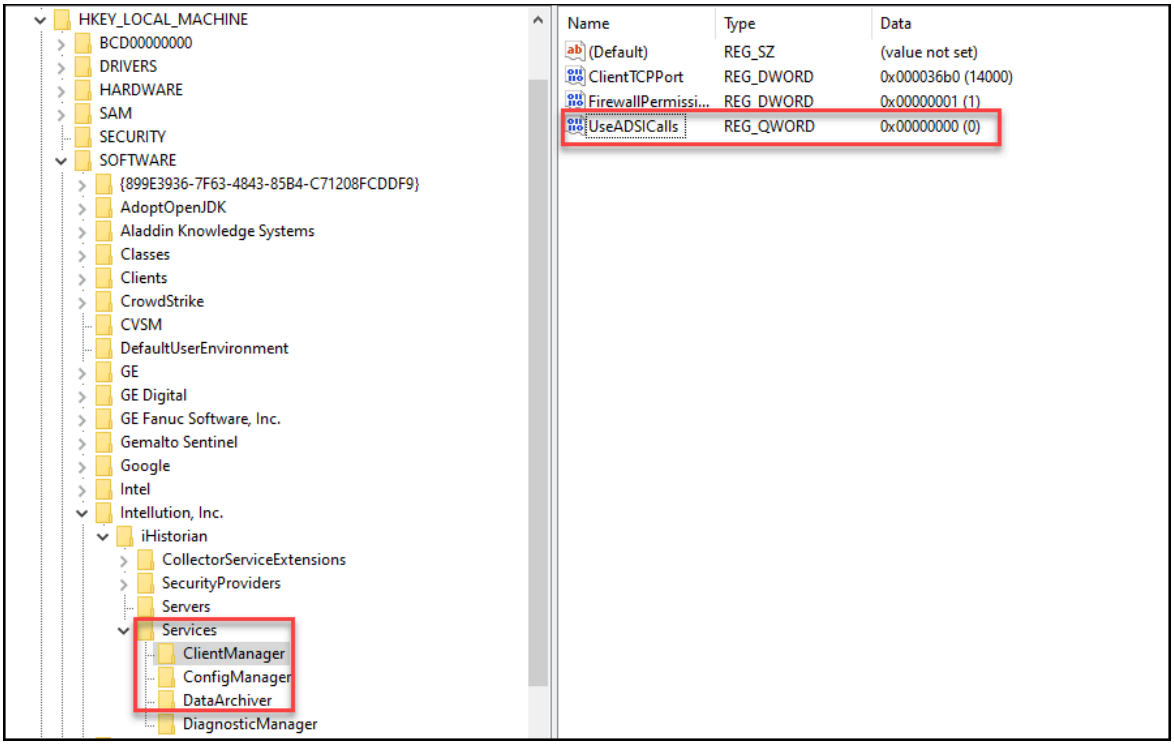
- iH Security Admins
- iH Collector Admins
- iH Tag Admins
- iH Archive Admins
- iH UnAudited Writers
- iH UnAudited Logins
- iH Readers
- iH Audited Writers

> **Note:**
> Historian Security Groups should be of type **Domain-Local** only.

1. On the **Data Stores** section, under **Security** > **Global Security**, select the **Use domain** option.




2. Stop the Historian Services.
3. Add the Registry Entries for ClientManager, ConfigManager and DataArchiver as shown below.
   Registry path: *\HKEY_LOCAL_MACHINE\SOFTWARE\Intellution, Inc.\iHistorian\Services*




4. Start the Historian Services. The new registry entries will now be read by the corresponding Historian service.

## Accessing Historian Server using Domain Users - Examples

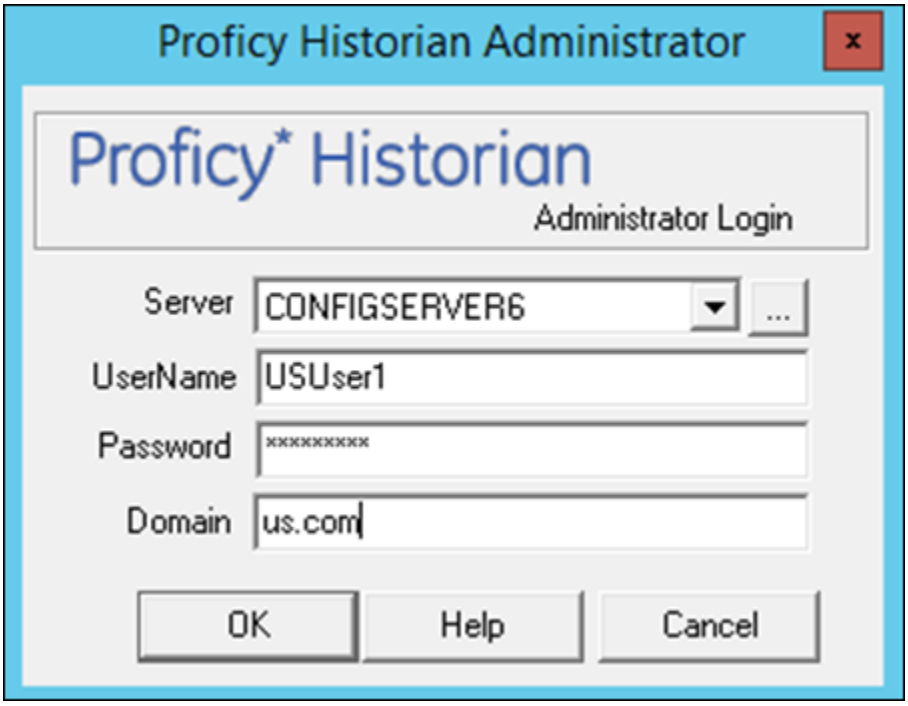
**Example 1**: European domain user trying to connect to Historian Server installed in India.Europe.US.com Domain Controller (DC).

1. Create a user **EuropeUser1** in Europe.US.com DC.
2. Add the user (i.e. EuropeUser1) from Europe.US.com to “iH Security Admin” group in India.Europe.US.com DC.
3. Log in to Historian Client using EuropeUser1 as shown below.

4. EuropeUser1 is added to iH Security Admin. The user will get full access to Historian Server.

**Example 2: US domain user trying to connect to Historian Server (which is installed in India.Europe.US.com Domain Controller (DC).**

a. Create a user **USUser1** in US.com DC.
b. Add the user (i.e. USUser1) from US.com to “iH Readers” group in India.Europe.US.com DC
c. Log in to Historian Client using USUser1 as shown below.




5. USUser1 is added to iH Readers. The user will get only data read access to Historian Server.

## Adding Nested Domain Groups to Historian Security Groups

The following procedure describes, a European domain group user trying to connect to Historian server, installed in India.Europe.US.com Domain Controller.

1. Create a user **EuropeGroupUser1** in Europe.US.com DC.
2. Create a group **EuropeGroup1**.
3. Add the EuropGroupUser1 to EuropeGroup1.
4. Add the group (i.e. EuropeGroup1) from Europe.US.com to “iH Security Admin” group in India.Europe.US.com DC
5. Log in to Historian Client using EuropeGroupUser1.
6. If a new user gets added to EuropeGroup1, then the user gets automatically synced with Historian Security Groups.

> **Note:**
> As EuropeGroupUser1 is added to iH Security Admin, user will get full access to Historian Server.

US domain group user trying to connect to Historian Server installed in India.Europe.US.com Domain Controller.

a. Create a user **USGroupUser1** in US.com DC (if not exist).
b. Create a group USGroup1.
c. Add the USGroupUser1 to USGroup1.
d. Add the group (i.e. USGroup1) from US.com to “iH Security Admin” group in India.Europe.US.com DC
e. Login to Historian Client using USGroupUser1.
f. If a new user gets added to USGroup1, then it gets automatically synced with Historian Security Groups.

> **Note:**
> As USGroupUser1 is added to iH Readers, user will get data read access to Historian Server

## Managing Proficy Authentication Users Using the Configuration Tool

Use the Proficy Authentication Config tool to perform the following tasks:

- Add a local Proficy Authentication user.

> **Note:**
> Here a local Proficy Authentication user means a user defined by Proficy Authentication, not by an external identity provider such as LDAP.

- Remove a local Proficy Authentication user.
- Reset the password for a local Proficy Authentication user.
- Add a local Proficy Authentication user to an existing group.

  Since OAuth2 scopes are implemented as Proficy Authentication groups, this means the same as adding a scope to a user.
- Remove a local Proficy Authentication user from an existing group.

A user who performs these functions acts as the admin client and needs to know the secret of the admin client. The tool does provide a way for the user to cache the secret safely to be used later.

By default, this tool is available in the following folder: `C:\Program Files\GE Digital\Historian Config`. Run the tool from a Windows command prompt window.

**Syntax**

The tool's syntax follows this format:

```
uaa_config_tool verb [options]
```

where verb is one of the following:

- `add_user`
- `remove_user`
- `set_user_password`
- `add_user_to_group`
- `remove_user_from_group`
- `clear_secret`

Run the tool without a verb or any other options to view the help page.

The uaa_config_tool utility prompts for a port number. This is the port number that you have specified in the Public HTTS Port field in the **TCP PORT ASSIGNMENTS** page. By default, it is set to 443. If you have changed the public HTTPS port number, enter the number. Otherwise, enter 443.

Options can be specified in the form of single dash followed by a short name, or double dash followed by a long name, followed by the value of the option, if any. For example, you can specify the user name `Alice` by either

```
-u Alice
```

or

```
--UserName Alice
```

**Table 11. Options**

| Short name | Long name | Remark |
|---|---|---|
| `-t` | `--Target` | URL of the Proficy Authentication instance that the command should be performed on. Typically, the URL is `https://localhost:443/uaa`, which is the default value. This option is optional and is only needed when the user wants to run the command against a remote Proficy Authentication instance (which is not recommended due to security concerns). |
| `-n` | `--ClientId` | ID of the client that the user is acting as. By default, it is `admin`. This option is optional and is only needed when the ad- |

**Table 11. Options (continued)**

| | | |
|---|---|---|
| | | min has set up the Proficy Authentication to delegate certain operations to others. |
| `-s` | `--ClientSecret` | This is the secret used to authenticate the user for acting as the admin client (or an alternative client given in a `--ClientId` option). If the user has elected to cache the secret previously, then this option can be omitted. Otherwise, it has to be provided.<br><br>The password must satisfy the following conditions:<br><br>• Must not contain only numbers.<br>• Must not begin or end with a special character.<br>• Must not contain curly braces. |
| `-c` | `--CacheSecret` | This option is not followed by a value and is optional. If specified, the tool will cache the client secret so when the next time this tool is invoked the secret does not have to be specified. Note that the secret is encrypted and only the current Windows logon user can access and decrypt. |
| `-u` | `--UserName` | Name of the user that the tool is being invoked for. For example, the user that is being added or removed. |
| `-p` | `--UserPassword` | The password for the user being added or whose password is being reset. The option is only needed for the `add_user` and `set_user_password` commands. |
| `-g` | `--Group` | Name of the Proficy Authentication group (scope) that the user is being added to or removed from. The option is only needed for the `add_user_to_group` and `remove_user_from_group` commands. |

**Examples**

- To add a user named alice with the password Pa55word and the admin client secret myclientsecret (this is the admin client secret that you entered while installing Web-based Clients):

```
uaa_config_tool add_user -u alice -p Pa55word -s myclientsecret  -c
```

If the Proficy Authentication server is on a remote machine named webhost.lab:

```
uaa_config_tool add_user -u alice -p Pa55word -s myclientsecret -t https://webhost.lab:443/uaa -c
```

- To provide user privileges to access the Web Admin console and Trend Client:

```
uaa_config_tool add_user_to_group -u alice -g historian_visualization.user -t https://webhost.lab:443/uaa
```

- To provide admin privileges to access the Web Admin console and Trend Client:

```
uaa_config_tool add_user_to_group -u alice -g historian_visualization.admin -t https://webhost.lab:443/uaa
```

- To provide Configuration Hub privileges, add alice to the group historian_enterprise.admin, using the previously cached admin secret:

```
uaa_config_tool add_user_to_group -u Alice -g historian_enterprise.admin -t https://webhost.lab:443/uaa
```

- To remove alice from a remote instance of Proficy Authentication as an alternative client (that is, other than `admin`) `useradmin`:

```
uaa_config_tool remove_user -u alice -t https://webhost.lab:443/uaa -n useradmin -s MyOtherNonSecret
```

- To clear any cached client secret:

```
uaa_config_tool clear_secret
```

**Note:**
If the Windows logon account is not shared, it is not necessary to clear cached secret, since the cache is encrypted and only the same Windows user account can decrypt.

When there are Historian security groups on the local historian machine or on the domain server:

1. Create a new user account on the local Historian machine or on the domain server with same login name and password as the local Proficy Authentication user.
2. Add the new user to the appropriate Historian Security group on the local historian machine or on the domain server.

## Create a Proficy Authentication Reader Client

To fetch Historian data, previously, you had to manually add each user to iH security groups. Only then the users could fetch the data. Now, this process has been simplified. All you must do is create a Proficy Authentication client. The Proficy Authentication client will be automatically added to the iH Readers security group on the server side. Therefore, you can use the client to fetch Historian data using REST APIs.

1. Access the Proficy Authentication Config tool, which is located in the following folder by default: `C:\Program files\GE Digital\Historian Config\uaa_config_tool`
2. Access Command Prompt as an administrator, and then run the following command:

   ```
   uaa_config_tool add_historian_reader_client -u <client name> -p <client password>
                           -n <username> -s <password> -t https://<Proficy Authentication URI>:443/uaa
   ```

   To create a client named client1:

   ```
   uaa_config_tool add_historian_reader_client -u client1 -p password123
                           -n user1 -s userpassword123 -t https://Proficy Authentication URI2010:443/uaa
   ```

   The Proficy Authentication reader client is created. However, if it already exists, a message appears, stating the same.

## About Accessing Cross-Domain Historian

You can now access Historian across various domains regardless of the domain to which your user account belongs. To do so:

- Ensure that all the domain controllers across various domains trust one another.
- Create iH security groups on each leaf node that contains the Historian server that you want to access.
- Ensure that you are part of at least one iH security group. You can be part of an iH security group directly, or you can be part of a universal domain group, which must be part of the iH security group.

## About Domain Security Groups

When you configure Historian to use domain security groups, the data archiver attempts to locate the groups on the primary domain controller (PDC) or one of the backup domain controllers (BDC). When using a PDC, if a primary or backup domain controller cannot be located when the Historian Data Archiver service starts, access to Historian is denied to all users.

For troubleshooting, .shw file of the data archiver lists all PDCs and BDCs available at the time of archiver startup. Use this list to verify that the Historian server has visibility into the appropriate domain.

When using a PDC, after the list of Domain Controllers has been established, the Historian Server will use that list to query for Security Group Membership on an as needed basis. If at any time a request for Group Membership information is made and the Primary Domain Controller is not available, Historian selects the first Backup Domain Controller and attempts the same request. If a Backup Domain Controller successfully responds to the request, the process of querying for Group Membership can stop. Otherwise,

Historian will attempt to query Group Membership information from the next available Backup Domain Controller. If no Backup Domain Controller successfully responds, access to the system is denied.

Changing security group configuration from Local to Domain or vice versa requires that the Historian Data Archiver service be restarted for the change to take effect.

## Establishing Your Security Rights

Your security identity is established upon connecting to the server. This occurs through the following steps:

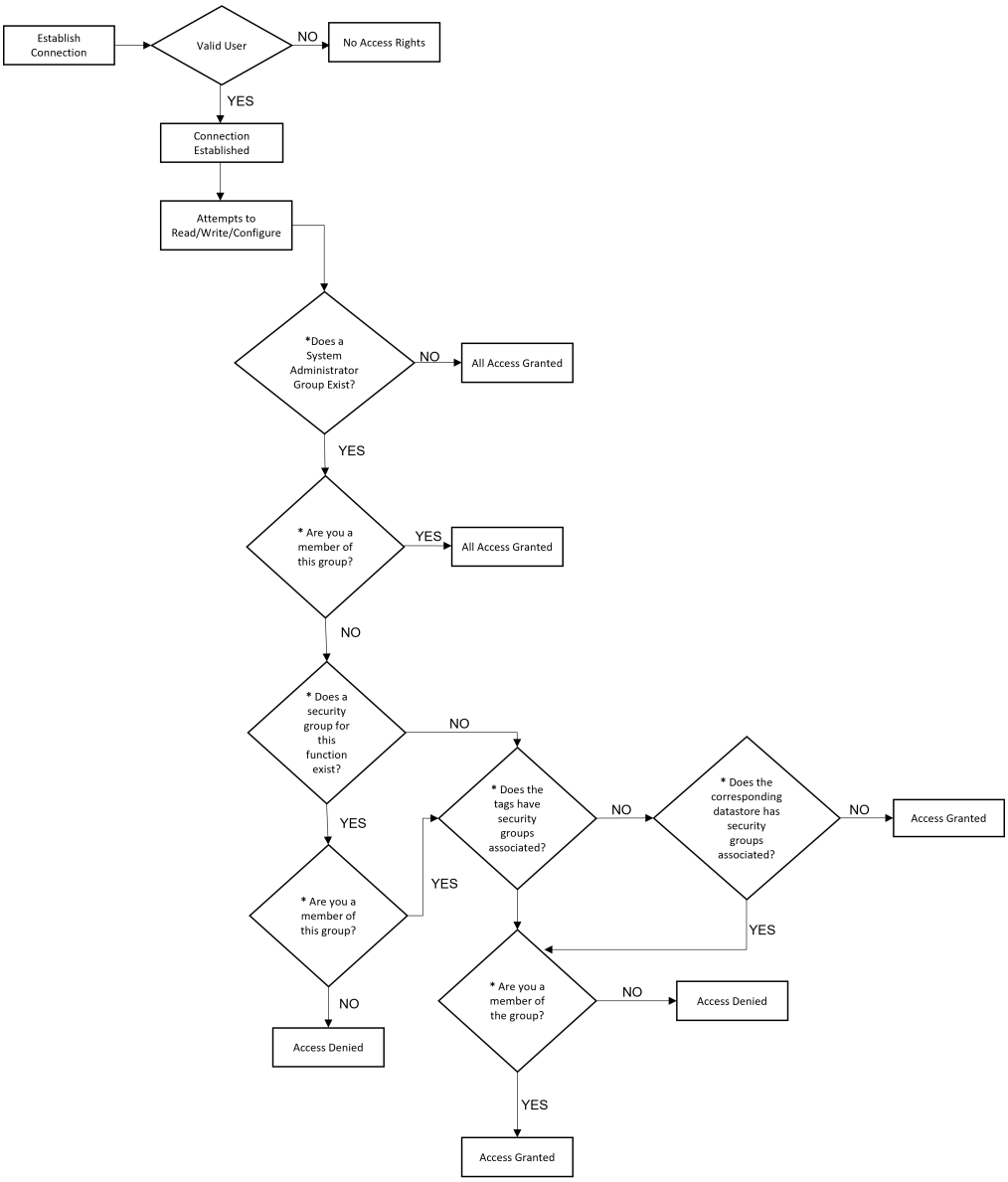
1. Specifying a user name and password of an account.
   Upon connection, the system checks to see if you have a valid Windows 2003 account. If you have supplied a username and password (through the Excel Add-In for example), security checks that user. If username and password are not supplied and you are on a Windows 2003 or Windows 2008 machine or higher, security checks the currently logged in user.

   > **Note:**
   > If you do not pass a domain name the account will be checked locally in the same way a mapped drive attempt happens. You have to specify a username and password that exists on the server.

2. Determining group membership of that account.
   Once the account is validated, the server determines group membership. For more information on the process and hierarchy of the groups, refer to the Security Checking Process diagram below.
3. Caching membership profile.
   Once the group and tag membership are determined, it is cached for the connection and not looked up again. If users are added to or deleted from a group, the cache is not updated.

**Note:**
The cache information is per connection, and not per IP address. In other words, it is cached per application and not per system.

Figure 1. Security Checking Process



## Implementing Tag-level Security

In addition to defining the iH Tag Admins who have the power to create, modify, and remove tags, you can also define individual tag level security to protect sensitive tags.

Set tag level security in Historian Administrator. You need the Historian Security Groups to implement tag-level security. You can use a Windows pre-defined group (power users, for example) or create your own separate group specifically for this function. For more information on creating and adding groups, refer to Setting Up Historian Security Groups *(on page 277)*.

Users must have iH Security Admins rights to set individual tag level security, browse, or query tags in Historian Administrator.

> **Note:**
> Tag security is not enforced in the Trend Client when it comes to browsing the full list of tags. Security, however, is enforced when it comes to trending data for tags for which you have permission. For example, if you are logged into the Trend Client as a user that is a member of the User Group assigned to a tag's security Read Group, you will still be able to browse all Historian tags. However, you are only allowed to trend the tags for which the user is a member of the User Group assigned to the tag's security Read Group,

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
   A list of all the tags appears.
3. Select the row containing the tag whose security you want to define.
   The tag details appear in the **DETAILS** section.
4. Enter values as described in the following table.

| Field | Description |
|---|---|
| **Read Group** | The Windows security group that can retrieve the tag data and plot it in a trend chart.<br><br>For example, if you select a group with power users, in addition to members of the iH Security Admins group, only a member of the power users group will be able to read data for that tag. Even a member of the iH Readers group will not be able to access data for that tag, unless they are also defined as a member of the power users group. |

| Field | Description |
|---|---|
| **Write Group** | The Windows security group that can write tag data (for example, using the Excel Add-in for Historian). |
| **Administer Group** | The Windows security group that can create, modify, and delete the tag. |

**Note:**

- If you are using domain groups (instead of local groups), the **Read Group**, **Write Group**, and **Administer Group** fields contain only the groups whose names begin with iH<space> (case-sensitive). Therefore, ensure that the group that you want to use begins with iH<space>. For more information on the security groups, refer to Historian security groups *(on page 256)*.
- If you are using Proficy Authentication, any custom groups that you created must be defined in Proficy Authentication and added to the group whose names begin with ih_<group_name>. For more information on the Proficy Authentication groups, refer to about Proficy Authentication groups *(on page 231)*.

## Implementing Data Store-level Security

In addition to defining the iH Tag Admins who have the power to create, modify, and remove tags, the individual tag-level security, you can also define individual data store-level security to protect sensitive tags.

**Note:**
The security settings defined at the tag level, if any, take the precedence over those at the data store-level.

You need the Historian Security Groups to implement data store-level security. You can use a Windows pre-defined group (power users, for example) or create your own separate group specifically for this function. For more information on creating and adding groups, refer to Setting Up Historian Security Groups *(on page 277)*.

Users must have iH Security Admins rights to set individual data store level security.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Data Stores**. A list of all the data stores appears.

3. Select the row containing the data store whose security you want to define.
   The data store details appear in the **DETAILS** section.
4. In the **SECURITY** section, enter values as described in the following table.

| Field | Description |
|---|---|
| **Read Group** | The Windows security group that can retrieve the tag data and plot it in a trend chart for the selected data store.<br><br>For example, if you select a group with power users, in addition to members of the iH Security Admins group, only a member of the power users group will be able to read data of the tags for that data store. Even a member of the iH Readers group will not be able to access data of the tags for the selected data store, unless they are also defined as a member of the power users group. |
| **Write Group** | The Windows security group that can write tag data for the selected data store (for example, using the Excel Add-in for Historian). |
| **Administer Group** | The Windows security group that can create, modify, and delete the tags for the selected data store. |
| **Note:**<br>When it comes to the group security, the security settings applied at the tag level, if any, take the precedence over those at the data store level. | |

**Note:**

- If you are using domain groups (instead of local groups), the **Read Group**, **Write Group**, and **Administer Group** fields contain only the groups whose names begin with iH<space> (case-sensitive). Therefore, ensure that the group that you want to use begins with iH<space>. For more information on the security groups, refer to Historian security groups *(on page 256)*.
- If you are using Proficy Authentication, any custom groups that you created must be defined in Proficy Authentication and added to the group whose names begin with ih_<group_name>. For more information on the Proficy Authentication groups, refer to about Proficy Authentication groups *(on page 231)*.

For more information on System Global security settings, refer to access system *(on page 411)*.

# Uninstalling Historian

## Uninstalling Historian

Uninstalling Historian removes all saved Favorites from your Trend Client and all Users and Scopes you created. To keep these and other configurations on an upgrade, do not uninstall Historian unless you are changing server roles as previously described. If you must uninstall Historian on an upgrade, you can Export your favorites and save your data and tag configuration files for future use.

For information on uninstalling OPC Data Collectors, refer to the *Modifying and Uninstalling OPC Collectors* section of the *Historian Data Collectors* manual.

- When you want to uninstall Web-based Clients:
  - If you select the **Purge database during uninstall** check box, the entire database will be purged, and you must recreate the Proficy Authentication details, favourites, and so on. Therefore, if you want to retain Proficy Authentication details, do not select that check box.
  - If you have installed Operations Hub on the same machine, and if there is a shared Proficy Authentication package between Operations Hub and Web-based Clients, in some cases, a message appears, asking you to first uninstall Operations Hub before uninstalling Web-based Clients. You can, however, uninstall Web-based Clients first; the shared Proficy Authentication package will not be deleted in that case.
  - Configuration Hub will not be uninstalled because it is a common component. If needed, you can uninstall it separately.
- If you uninstall Historian after installing the Excel Add-In as described, ensure that you clear the **Historian** check box in the Microsoft Excel **Add-Ins** window. If you do not clear this option, you will receive an error each time you open Microsoft Excel.

1. To uninstall Historian from your computer:

   a. Double-click the **Programs / Uninstall a Program** link in the Control Panel.

   b. Select **Historian** and select **Uninstall**.

   > **Note:**
   > Historian archives are not removed by default. If you need to remove them, delete the folder manually.

A progress bar appears, showing that the software is being uninstalled. This may take some time.

To abort the uninstall, select **Cancel**.

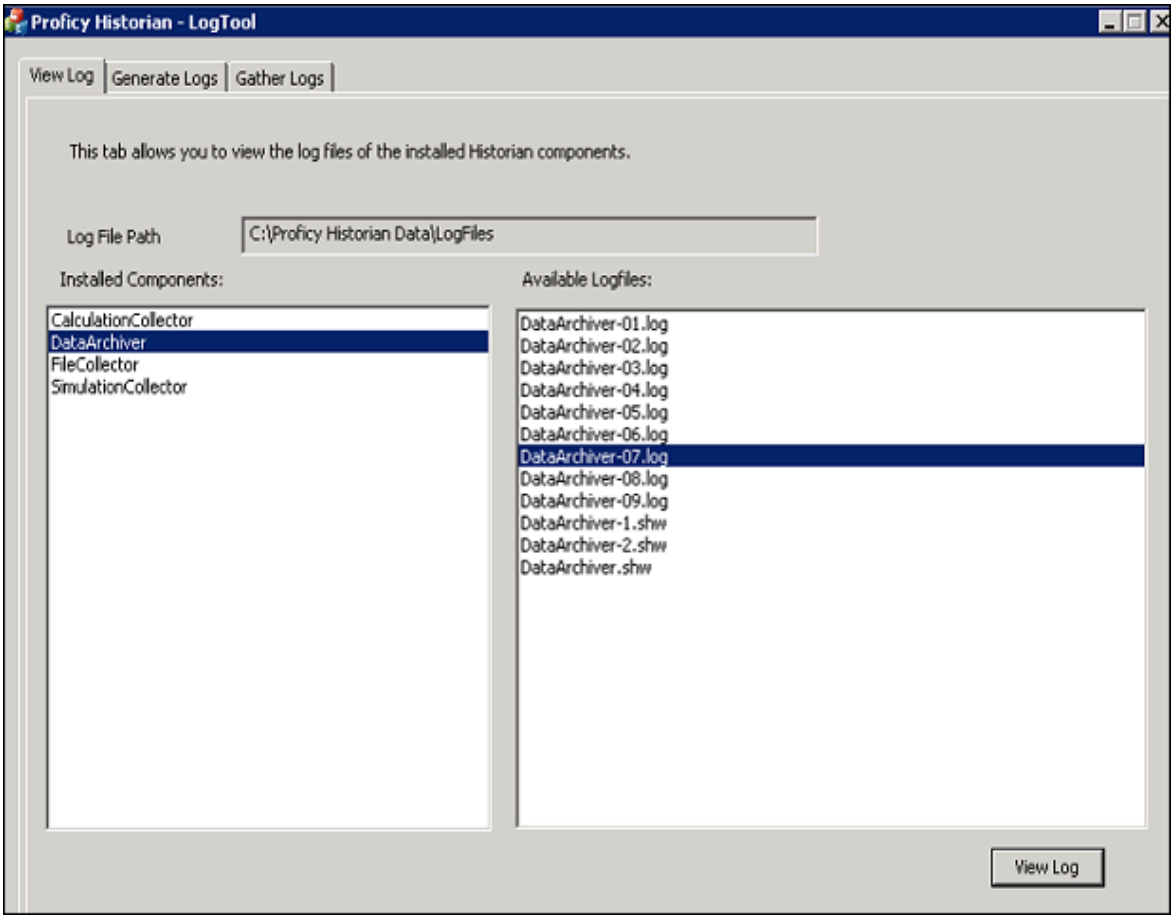
2. To remove all related software from your computer:
    a. Double-click the **Programs / Uninstall a Program** link in the Control Panel.
    b. Select **Proficy Common Licensing**, and select **Uninstall**.

# Troubleshooting

## Managing Historian Log Files

Use the Historian LogTool to view, generate, or compress log files to use for troubleshooting. `Logtool.exe` is located in the historian installation directory, for example: `C:\Program Files\Proficy \Proficy Historian\X64`.

1. Go to your installation directory and execute the `Logtool.exe` file.
   The **LogTool** appears, displaying the **View Log** section.



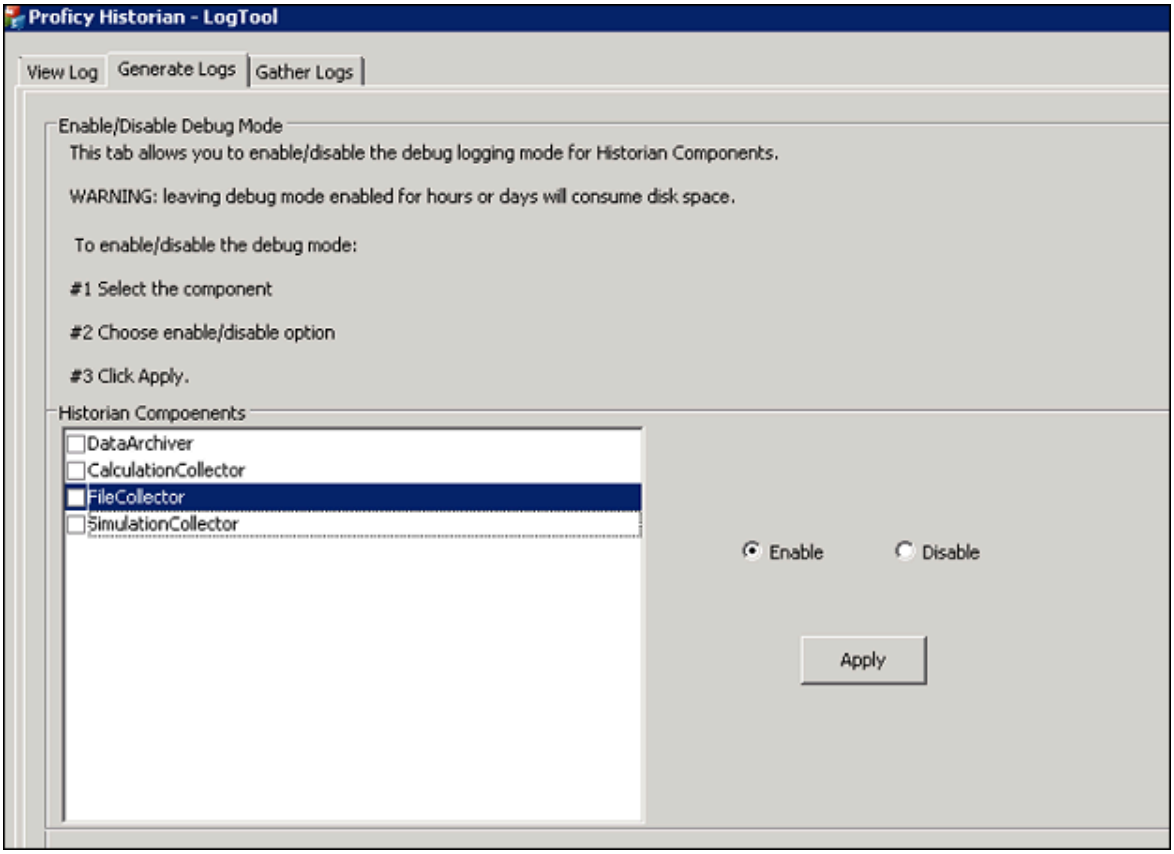

2. Select a component from the left panel to see the available log files, and select **View Log**.
3. Select **Generate Logs** to enable or disable the debug logging mode for Historian components:

   This tool will enable/Disable the debug mode for Historian components. However, leaving the debug mode enabled for longer time consume the disk space

   1.Select the component

   2.Choose enable/disable option

   3.Select apply

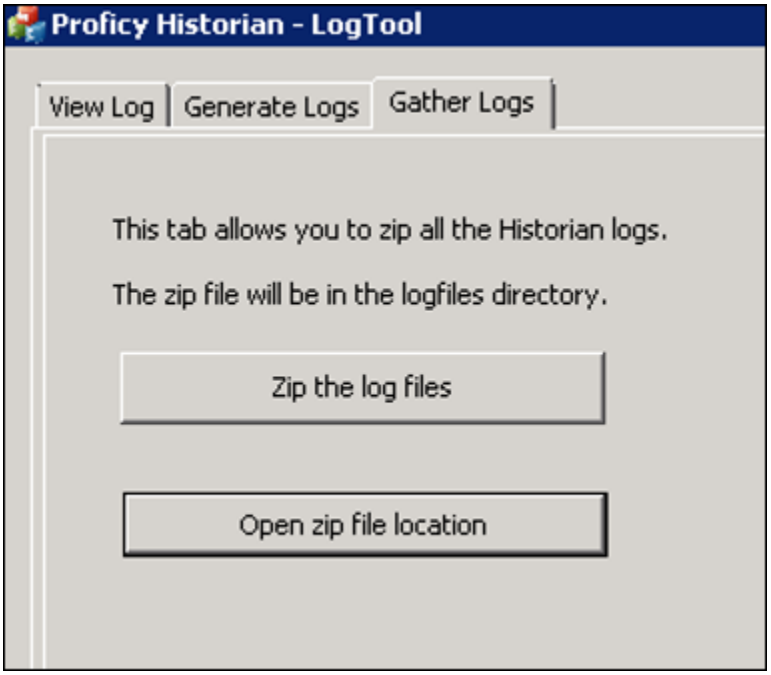



a. Select a Historian Component and select **Enable** or **Disable**.

> **Note:**
> Leaving debug mode enabled for a component consumes disk space.

b. Select **Apply**.

4. Select **Gather Logs** and select **Zip the log files** to compress the log files and select **Open zip file location** to view the zip files.




## Troubleshooting the Historian Server

**iFIX-Related Files in C Drive Even if iFIX is not Installed**

**Description**

If you install only Historian without installing iFIX, you may find some iFIX-related files in the C drive.

**Workaround**

You can ignore/delete them. If, however, you plan to install iFIX later, you must reinstall Historian Client Tools after installing iFIX.

**Error Message when Upgrading the Historian Server on a Passive Node in a Cluster**

**Description**

If you are upgrading the Historian server on a passive node, an error message may appear behind the installer screen, stating that the Archives directory is not created.

**Error Message**

Unable to create Archives directory.

**Workaround**

You can ignore this message, or you can make the node active before upgrading the Historian server.

### Historian Server Rejects Collector or Client User Credentials

#### Description

If a client or collector is attempting to connect to the Historian server with strict authentication enabled on a Kerberos configuration, the server rejects valid credentials and does not allow the connection.

#### Workaround

Ensure that the server time and the domain controller time match with each other.

### Historian Resource in a Cluster Environment is not Online

#### Workaround

Ensure that the cluster feature is included in your license.

### Historian Resource Runs for a Long Duration and Fails Over

#### Workaround

Debug the log messages of the Data Archiver and the Clusters before taking appropriate actions.

### While Label Error

#### Description

If the PFX file that you want to use with Historian does not contain a full-chained certificate, a while label error message appears.

#### Workaround

1. Create a PEM file from the PFX file by copying the content from all the certificates (such as root, intermittent, and leaf certificates) to the PEM file. It results in a full-chained certificate.
2. Get a KEY file from the vendor. Or, use a KEY file extracted from the PFX file using the Certificate Management tool. To do so, import the PFX file. The certificate and the KEY file will then be available in the `<Operations Hub installation`

`location>\httpd\conf\cert` folder. You can then use the `server.key` file in the folder.

3. Using the Certificate Management tool, import the PEM and KEY files to the machines on which the Historian server, the Operations Hub server, and clients are installed.

## Troubleshooting Web-based Clients

#### Unable to Access Configuration Hub After Upgrading Web-based Clients

**Description**

After you upgrade Web-based Clients, you cannot access Configuration Hub.

**Workaround**

Clear your browser cache.

#### Error Occurs When You Try to Access Web-based Clients

**Description**

If you have logged in to Operations Hub, and then if you try to access Web-based Clients, and vice versa, an error occurs. This is because the user credentials of the first application to which you have logged in are used to log in to the other one as well.

**Workaround**

Try one of the following steps:

- Add the scopes of each application user to the other application.
- If you log in to Web-based Clients first, on the login page of Operations Hub, re-log in with the Operations Hub user. If you log in to Operations Hub first, log out of Operations Hub, log in to Web-based Clients, and then log in to Operations Hub, using the credentials of the respective users for each application.

#### Unable to Access Web-based Clients

**Description**

When you upgrade Historian, after installing Web-based Clients, a message asking you to restart the machine does not appear. Because of this, sometimes, you cannot access Web-based Components such as Configuration Hub, Trend Client, the Web Admin console, and REST APIs.

**Workaround**

Restart the machine, or start the following services:

- Proficy Historian PostgreSQL Database
- Proficy Historian Tomcat Server
- Proficy Operations Hub Httpd Reverse Proxy
- Proficy Authentication PostgreSQL Database
- Proficy Authentication Tomcat Web Server

### Certificate Issues When Trying to Log in

#### Workaround

Connect to a Remote Proficy Authentication Service *(on page 182)*

### Web-based Clients are not connected to the Historian server

#### Workaround

- Ensure that the **Enforce Strict Client Authentication** and **Enforce Strict Collector Authentication** options are disabled in the **Data Stores Security** section in Historian Administrator.
- Ensure that the Proficy Authentication URL used by Web-based Clients matches the one you provided while installing the Historian server. If not, change the Proficy Authentication server *(on page 137)* so that the Historian server points to the same Proficy Authentication server as Web-based Clients.

### Clients or Users Not Created

#### Description

Clients or users are not created because the required services were not running during the installation of Web-based Clients.

- An error message appears when you access Web-based Clients or Trend Client.
- The visualization client is not found. The following error message appears: No client with requested id: historian_visualization
- When you attempt to log in to Proficy Authentication, a message appears, stating that the credentials are incorrect.

#### Workaround

Configure Web-based Clients *(on page 186)* to point to a different Configuration Hub and Proficy Authentication instance.

### The Reverse Proxy Service Stops Working

#### Description

If you install iFIX on a machine that has Historian Web-based Clients, the reverse proxy service stops working.

#### Workaround

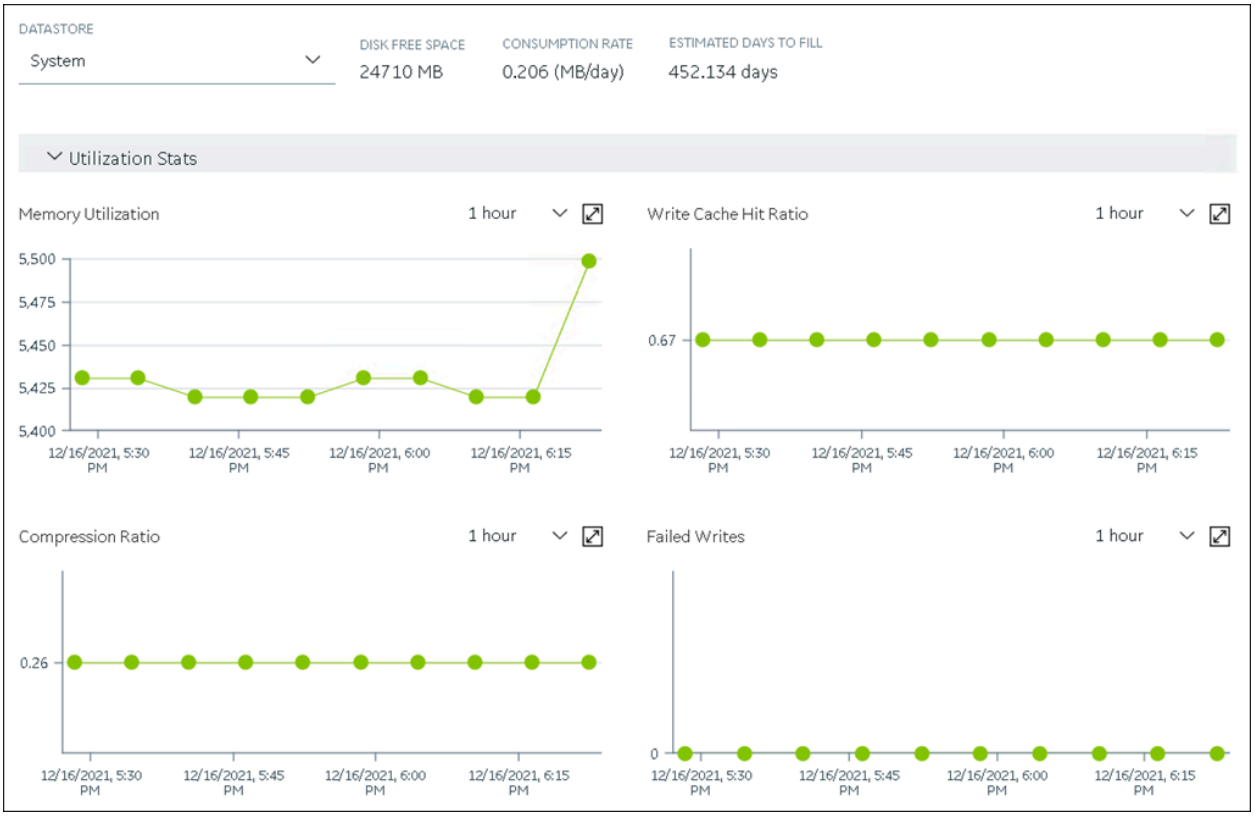
Restart the reverse proxy service - Proficy Operations Hub Httpd Reverse Proxy.

# Chapter 3. Configuration Hub

## Overview

### About Configuration Hub

The Configuration Hub application allows you to manage the Historian models, Historian systems, and their components.




**Advantages of Using Configuration Hub:**

- **Creating a Historian model:** You can create and manage object models, which is a hierarchical classification of objects. A model contains object types, variables, and instances.
- **A single application that enables you to manage multiple Historian systems:** A Historian system is a network of Historian servers that collect, store, and retrieve data related to tags, alarms, and events.You can create and manage Historian systems using Configuration Hub. In addition, you can manage collectors, data stores, and tags.
- **Horizontal scalability:** You can increase the storage capacity of a Historian system by connecting multiple software entities so that they work as a single logical unit. This will improve the

performance of the Historian system. The storage capacity depends on the number of Historian licenses that you have purchased.

- **High availability:** You can create mirror locations in a Historian system to achieve high availability of the server. If one of the servers is not available, you can retrieve data from the remaining servers in the mirror location.
- **Ease of setting up:** You can install all the collectors used in a Historian system easily by providing the required details with the help of the user-friendly interface.

**Types of Historian Systems**

- **Stand-Alone:** In a stand-alone Historian system, there is only one Historian server. This type of system is suitable for a small-scale Historian setup. For information on setting up a stand-alone Historian system, refer to About Setting up a Stand-Alone Historian System *(on page 356)*.
- **Horizontally scalable:** In a horizontally scalable Historian system, there are multiple Historian servers, all of which are connected to one another. This type of system is used to scale out the system horizontally. For example, if you have 5,00,000 tags in your Historian system, you can distribute them among the various servers to improve performance. For information on setting up a horizontally scalable system, refer to About Setting up a Horizontally Scalable System *(on page 359)*.

**Limitations**

- If only one machine remains in a mirror location, you cannot remove it.
- If you install Configuration Hub and the Web Admin console on the same machine, and use self-signed certificates for both of them, the login page for Configuration Hub does not appear. To prevent this issue, disable the domain security policies:
    1. Access the following URL: chrome://net-internals/#hsts
    2. In the **Domain Security Policy** section, in the **Delete domain security policies** field, enter the domain name for Configuration Hub, and then select **Delete**.
- If the primary server is down, you cannot add tags using a distributed node because the Configuration Manager service is down.

## Configuration Hub Workflow

This topic provides the high-level steps in using Configuration Hub.

1. Set up Configuration Hub. This involves installing the Historian server, the collectors, and Web-based Clients.
2. Apply the license *(on page 82)*.

3. Depending on your requirements, set up a stand-alone system *(on page 356)* or a horizontally scalable system *(on page 359)*. This involves adding the required components.

   **Note:**
   When you set up Configuration Hub, by default, a system and a data store are created. You can add more systems and data stores as needed.

4. For a horizontally scalable system, you can choose to set up high availability *(on page 367)*.
5. As needed, create an object model *(on page 371)*.
6. Specify the tags for data collection *(on page 357)*.

After you perform these initial steps, data is collected and stored in the Historian server. You can then retrieve and analyze the data.

# Setting up Configuration Hub

## About Setting up Configuration Hub

To set up Configuration Hub, you must perform the following steps:

1. Install the Historian server *(on page 105)*.
   - For a stand-alone Historian server, install single-server Historian.
   - For a horizontally scalable Historian server, install the mirror primary server and distributed/ mirror servers.
2. Install Web-based Clients *(on page 155)*.
3. Install collectors *(on page 143)*.
4. Perform the post-installation tasks *(on page 335)*.

If you want to upgrade Configuration Hub, refer to Upgrade Configuration Hub *(on page 336)*.

After you install or upgrade the required components, you can access Configuration Hub *(on page 336)*.

## Install the Historian Server Using the Installer

- set up the Historian environment.
- If you want to install web-based clients and view Historian license information on Configuration Hub, you must provide the Configuration Hub and Proficy Authentication server details during installation. Therefore, ensure that you have already set up Proficy Authentication in Configuration

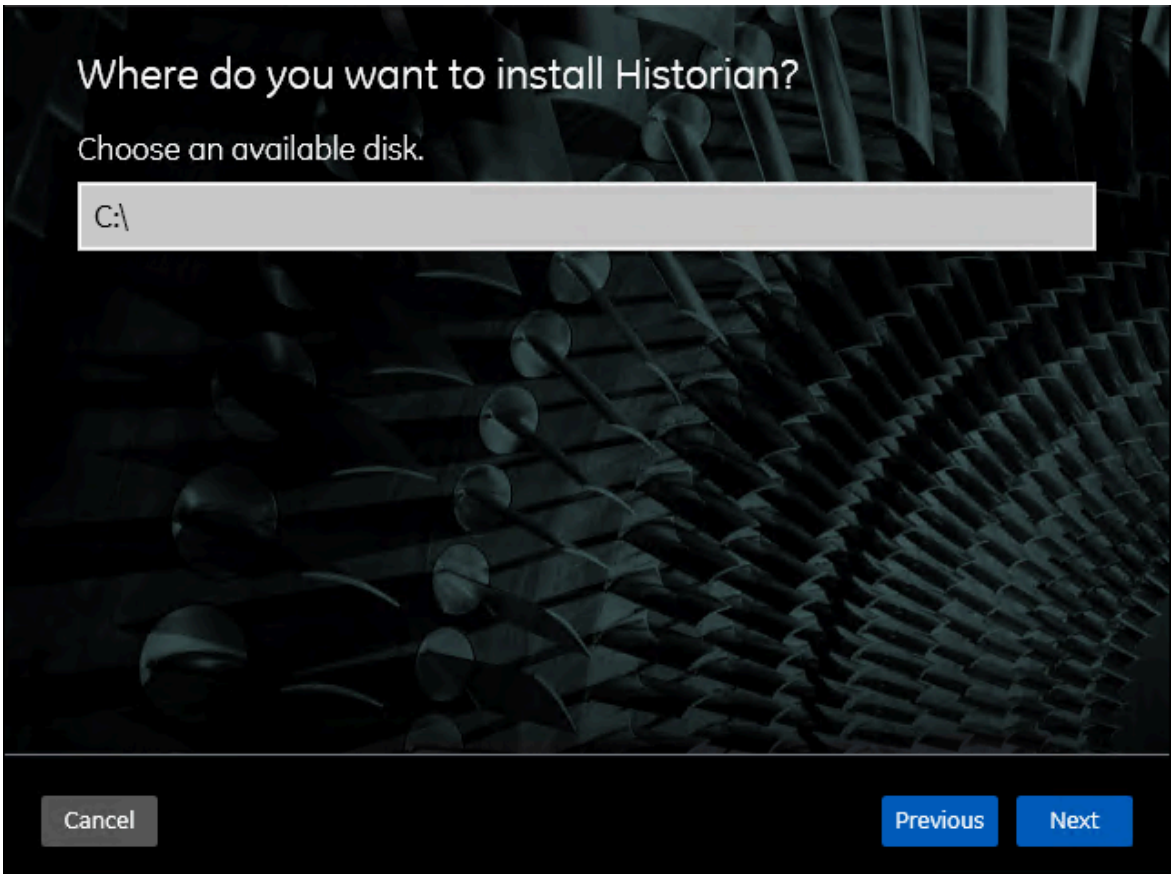
Hub. For more information on setting up Proficy Authentication in Configuration Hub, refer to https://www.ge.com/digital/documentation/confighub/version2023/t_authentication_setup.html.

- If you are changing the role of a Historian server that was previously a distributed/mirror server to any other configuration (single-server or mirror primary server), you must first uninstall Historian.
- If you are installing a distributed/mirror server, use the same configuration, license key, installation drive, Proficy Authentication instance, and domain as the primary server.

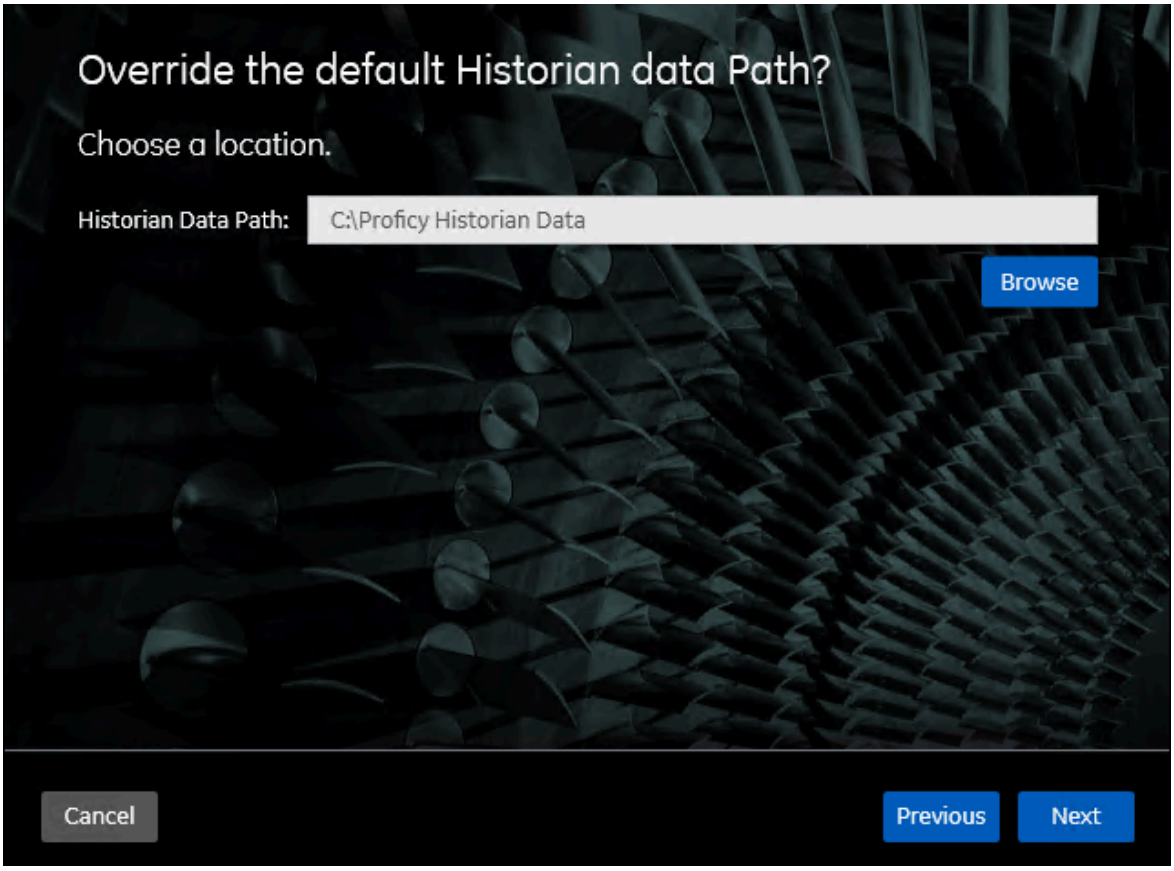
This topic describes how to install the Historian server using the installer.

You can also install it at a command prompt.

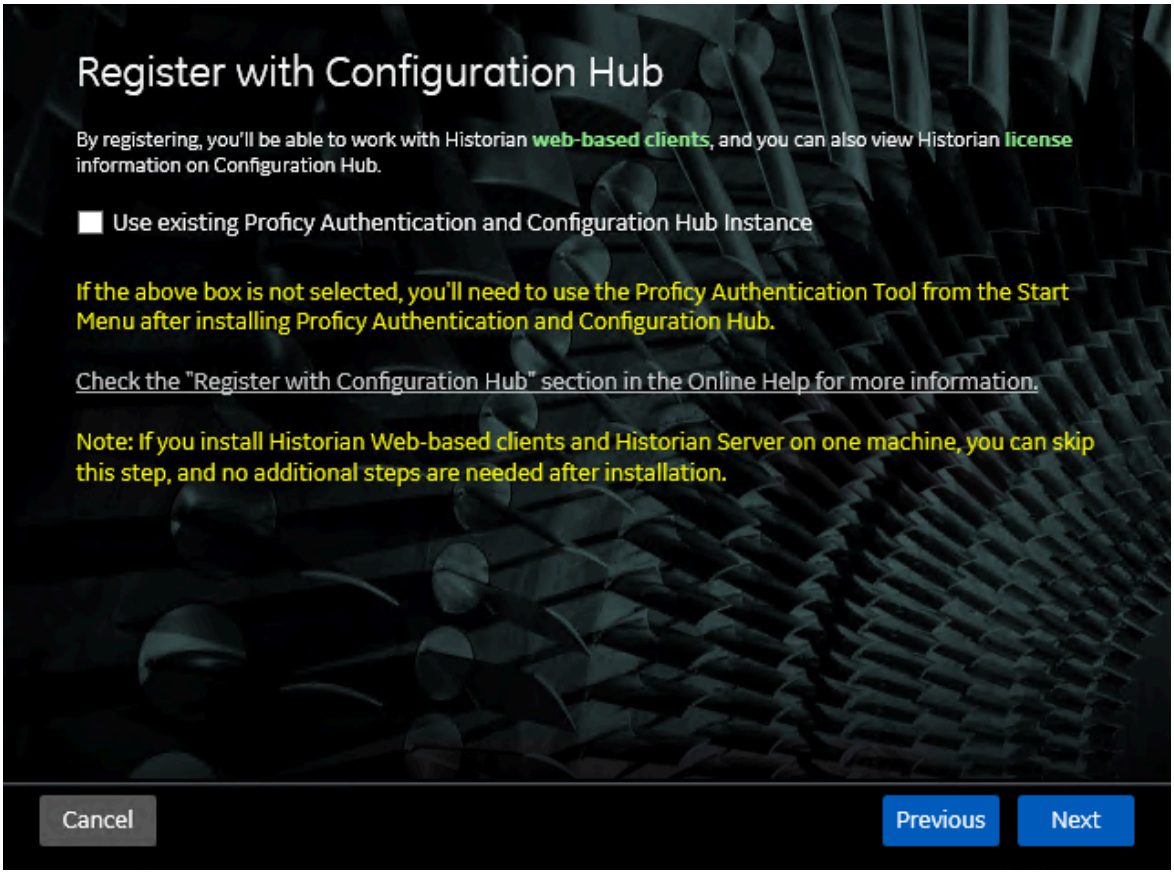
1. Log in as an administrator to the machine on which you want to install the Historian server.
2. Run the `InstallLauncher.exe` file.
3. Select **Install Historian Server**.
   The welcome page appears.
4. Select **Next**.
   The license agreement appears.
5. Select the **Accept** check box, and then select **Next**.
   The **Where do you want to install Historian?** page appears.



6. If needed, change the default installation drive of the Historian server, and then select **Next**.
   The **Override the default Historian data Path?** page appears.




7. If needed, change the default folder of the log files, and then select **Next**. If you want to include the Historian server in a cluster, enter the path to the shared folder of the cluster.
   The **Register with Configuration Hub** page appears.



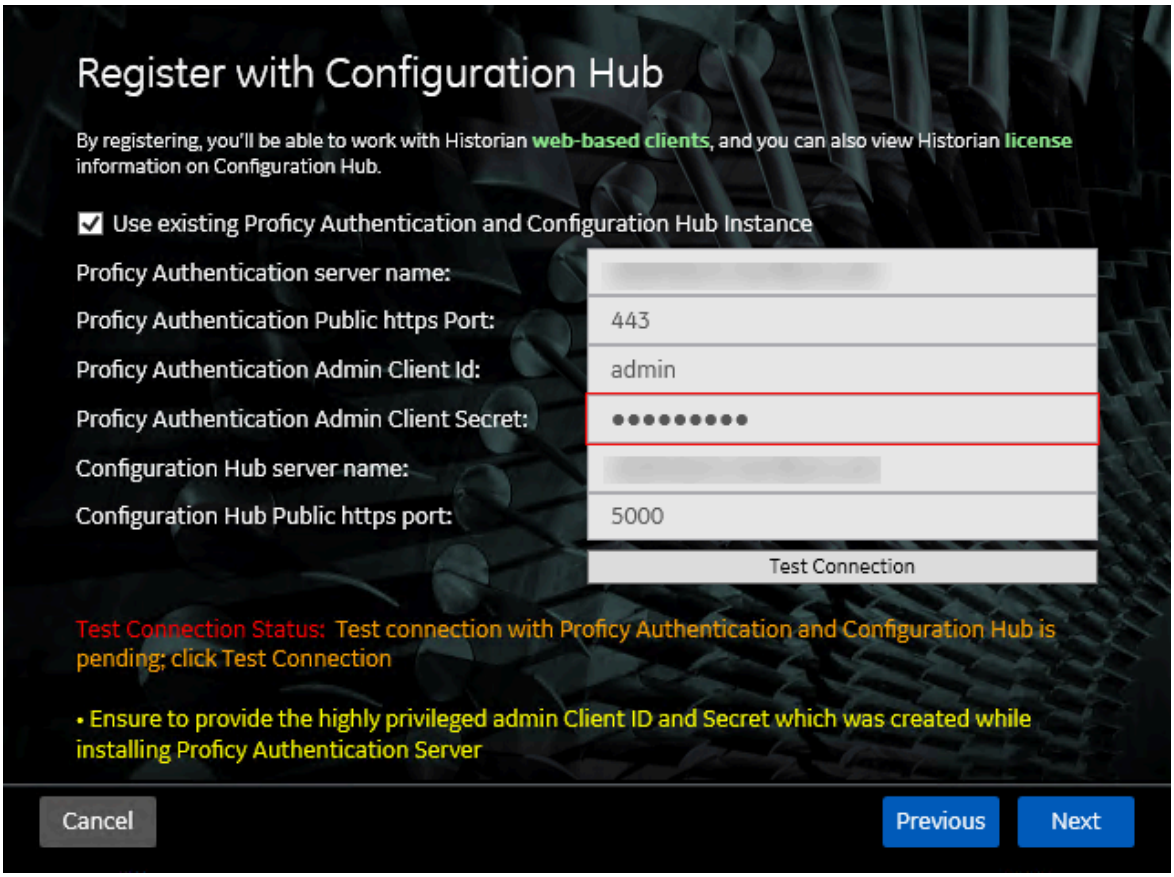

8. Select the **Use existing Proficy Authentication and Configuration Hub Instance** check box, and provide values as described in the following table.

   This step is needed only in the following cases:
   - The Historian Server and the Web-based clients will be on two different machines.
   - You already installed Proficy Authentication and Configuration Hub, and you want to use the Historian Web-based clients and view the Historian license information on Configuration Hub.

     If you do not select the **Use existing Proficy Authentication and Configuration Hub Instance** during the installation, to use the Historian Web-based clients and view the Historian license information on Configuration Hub, you must use the Proficy Authentication Tool from the Start menu to register with Proficy Authentication and Configuration Hub servers.

     If you do not have Proficy Authentication and Configuration Hub installed, and you intend to use the Historian Web-based clients and view the Historian license information on Configuration Hub, you can install them while installing the Web-based clients.

**Note:**
Proficy Authentication is required for user authentication. It provides identity-based security for applications and APIs. It supports open standards for authentication and authorization, including Oauth2.




| Field | Description |
|---|---|
| **Proficy Authentication server name** | Enter the name of the machine on which the Proficy Authentication server is installed. If the machine uses a fully qualified domain name (FQDN), provide the FQDN. By default, the local hostname is considered. |
| **Proficy Authentication Public https port** | Enter the port number used by the Proficy Authentication service. The default value is 443. Ensure that this port number matches the one |

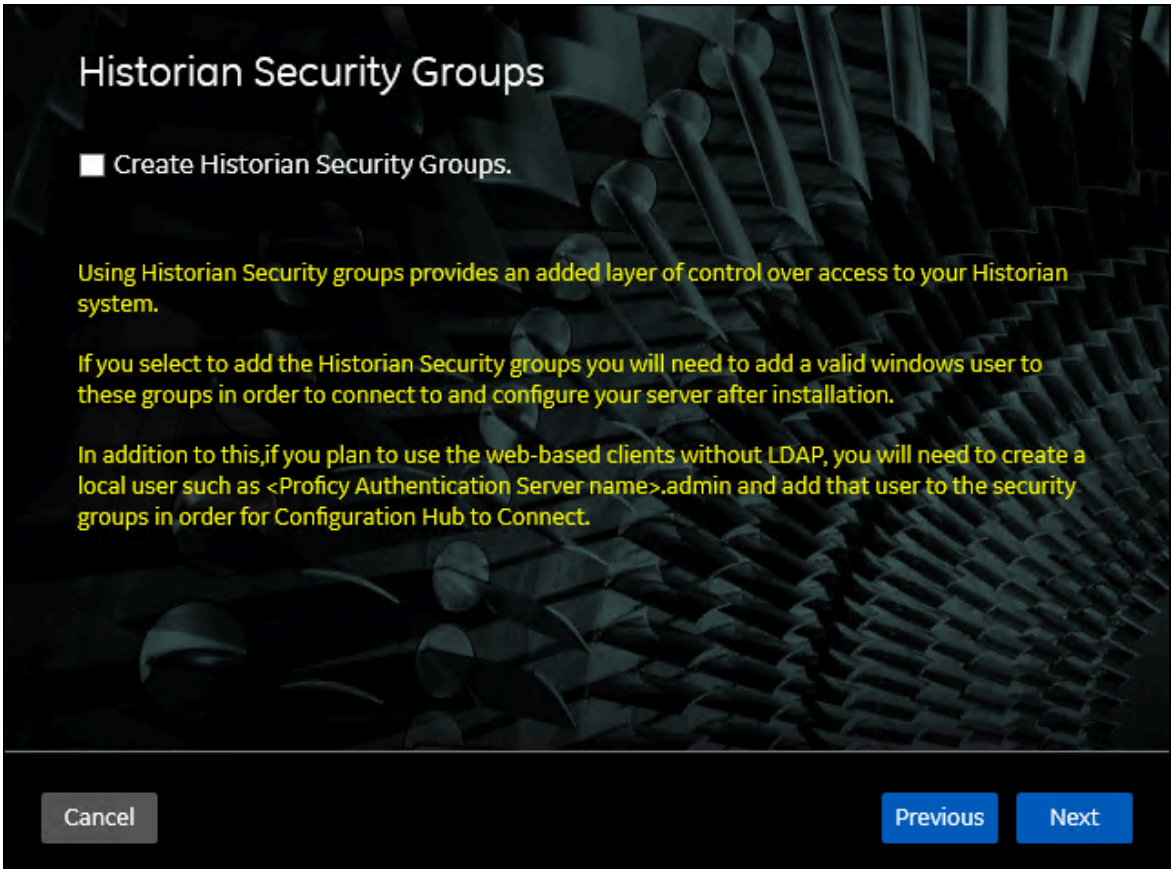
| Field | Description |
|---|---|
| | on the **TCP Port Assignments** page during Web-based Clients installation. |
| **Proficy Authentication Admin Client Id** | The client ID to connect to the Proficy Authentication service. |
| **Proficy Authentication Admin Client Secret** | The password to connect to the Proficy Authentication service. |
| **Configuration Hub server name** | The server name or the FQDN of the existing Configuration Hub server, as displayed in the address bar of the browser when you access Configuration Hub from the machine where Configuration Hub is installed. |
| **Configuration Hub Public https port** | The web server (https) port that you want to use for Configuration Hub. By default, it is 5000. |
| **Test Connection** | Option to test the status of the connection with external Proficy Authentication server. |

**Note:**
If you change the Proficy Authentication server for Web-based Clients later, you must also change the Proficy Authentication server for the Historian server. This can be done using the Proficy Authentication Configuration Tool located at `<Install Drive>:\Program Files\Proficy\Proficy Historian\x64\Server` without the need to install the Historian server again. Alternatively, you can search for the Proficy Authentication Tool in the Windows search bar and open it.

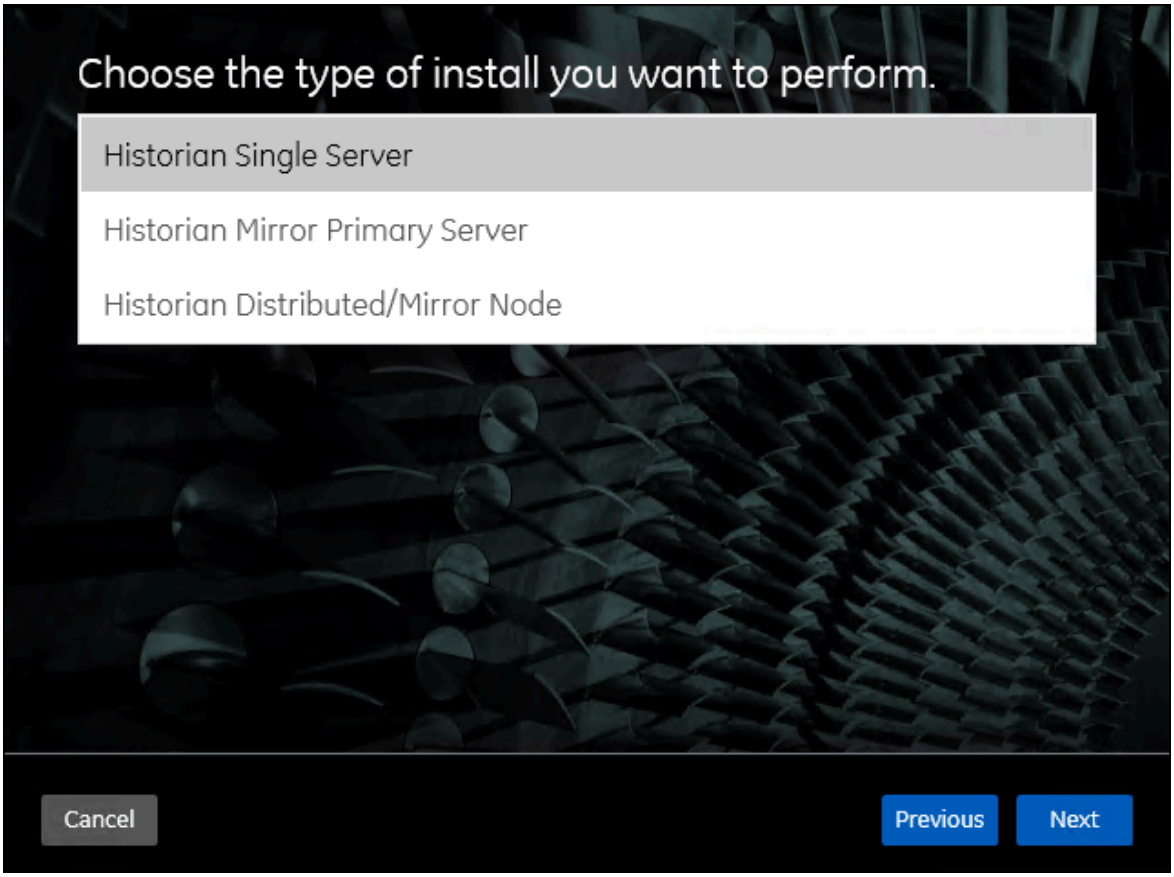
9. Select **Next**.
   The **Historian Security Groups** page appears.

   Using Historian security groups provides an added layer of control over access to your Historian system.

   By default, the option to create Historian security groups is not selected.

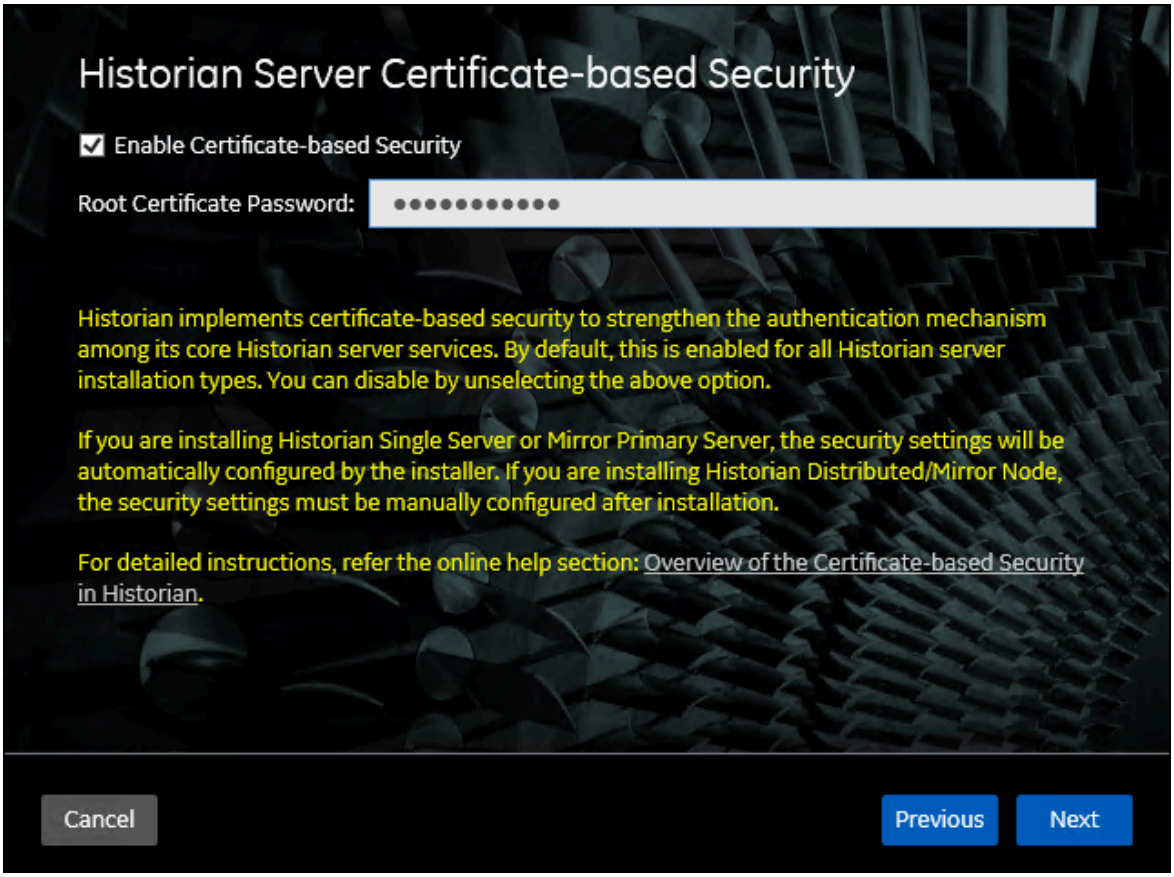



10. If you want the installer to create Historian security groups *(on page 256)*, select the corresponding check box, and then select **Next**.
    The **Choose the type of install you want to perform** page appears.




11. Select the type of the Historian server that you want to install, and then select **Next**.
    - **Historian Single Server**: This is for a stand-alone Historian system, which contains only one Historian server. This type of system is suitable for a small-scale Historian setup.
    - **Historian Mirror Primary Server**: This is for a horizontally scalable Historian system, which contains multiple Historian servers, all of which are connected to one another. Installing this server will allow you to add machines and distributed/mirror servers to this system.
    - **Historian Distributed/Mirror Node**: This is for a horizontally scalable Historian system. Installing this server will allow you to add this node to a primary server.

    The **Historian Server Certificate-based Security** page appears.




12. If you want to enable certificate-based security (MTLS-based security), leave the **Enable Certificate-based Security** check box selected, and then enter **Root Certificate Password**.

> **Note:**
> Ensure to use the same password to create MTLS (client) certificates. For more information on creating MTLS (client) certificates, refer to Generate MTLS certificate.

For more information on certificate-based security, refer to overview of the Certificate-based Security in Historian *(on page 206)*.

> **Warning:**
> If you do not select the **Enable Certificate-based Security** check box during the installation, you must generate the root certificates manually, as described in the Manually Install Certificates for Historian section. However, this is not recommended.

13. Select **Next**.
    The **You are ready to install** page appears.

14. Select **Install**.
    The installation begins.
15. After the installation, when you are asked to reboot your system, select **Yes**.
    The Historian server is installed on your machine in the following folder: `<installation drive>:\Program Files\Proficy\Proficy Historian\x64\Server`, and the following registry path is created: `HKEY_LOCAL_MACHINE\SOFTWARE\Intellution, Inc.\iHistorian\Services`.

    In addition, the following components are installed:
    - **The RemoteCollectorConfigurator utility:** A command-line tool, which allows you to manage collectors remotely. By default, it is located in the `C:\Program Files\GE Digital\NonWebCollectorInstantiationTool` folder. For instructions on using this utility, refer to About Installing and Managing Collectors Remotely.
    - **The Proficy Authentication Configuration tool:** A utility that allows you to specify the Proficy Authentication server details to match with the Proficy Authentication server used by Web-based Clients. By default, it is located in the `C:\Program Files\Proficy\Proficy Historian\x64\Server` folder. For instructions on using this tool, refer to Register with Configuration Hub *(on page 137)*.

## Install Web-based Clients Using the Installer

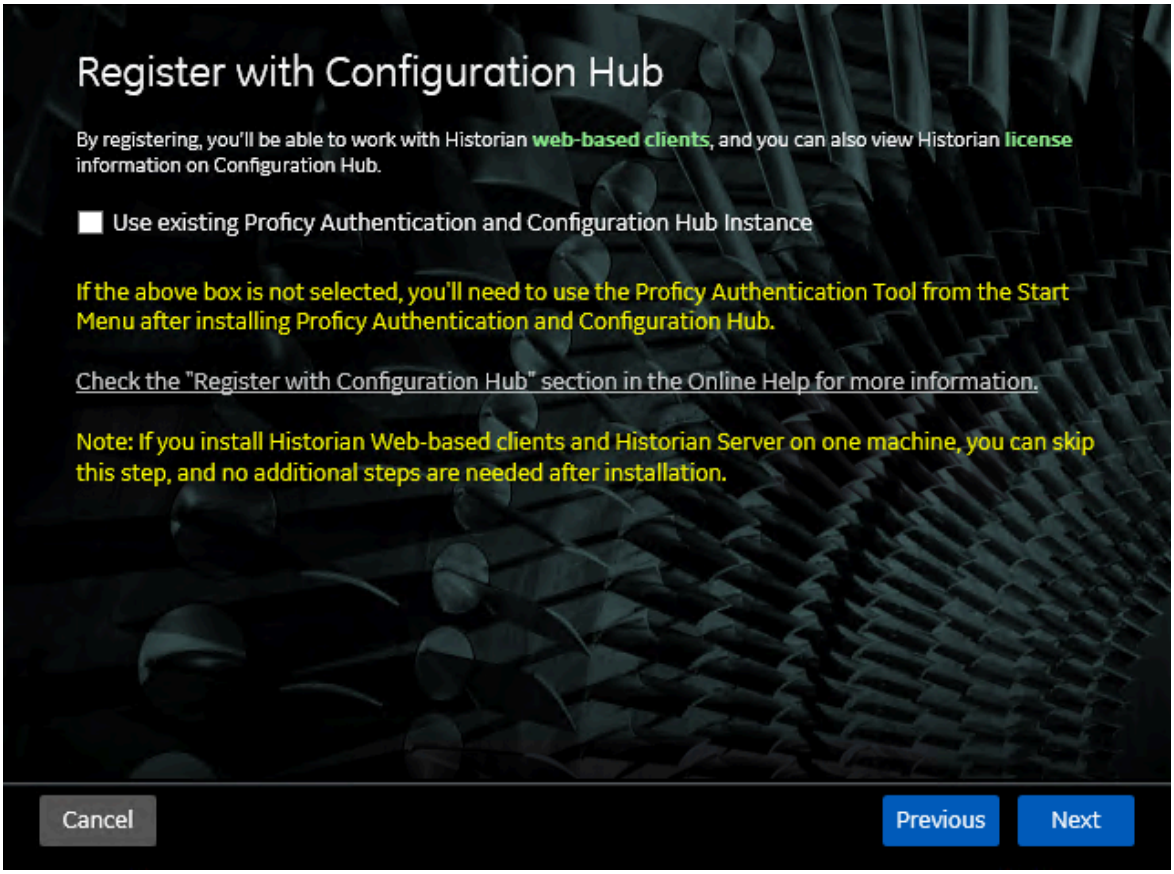
1. Install the Historian server *(on page 105)*. During the installation, in the **Register with Configuration Hub** page, select the **Use existing Proficy Authentication and Configuration Hub Instance** check box, and the provide the Proficy Authentication server and Configuration Hub server details.

**Note:**

This step is needed only in the following cases:
- The Historian Server and the Web-based clients will be on two different machines.
- You already installed Proficy Authentication and Configuration Hub, and you want use the Historian Web-based clients and view the Historian license information on Configuration Hub.

If you do not select the **Use existing Proficy Authentication and Configuration Hub Instance** during the installation, to use the Historian Web-based clients and view the Historian license information on Configuration Hub, you must use the Proficy Authentication Tool *(on page 137)* to register with Proficy Authentication and Configuration Hub servers.

If you do not have Proficy Authentication and Configuration Hub installed, and you intend to use the Historian Web-based clients and view the Historian license information on Configuration Hub, you can install them while installing the Web-based clients.




2. If you want to use Web-based Clients in a cluster environment, ensure that your network is enabled for multicast traffic, and set up high availability on each node in the cluster.

This topic describes how to install Web-based Clients using a GUI-based installer.

You can also install Web-based Clients using the command line.

During the installation, you can choose to use Web-based Clients in a cluster environment, thus ensuring high availability of connection to the Historian server using the client applications.

1. Run the `InstallLauncher.exe` file.
2. Select **Install Web-based Clients**.

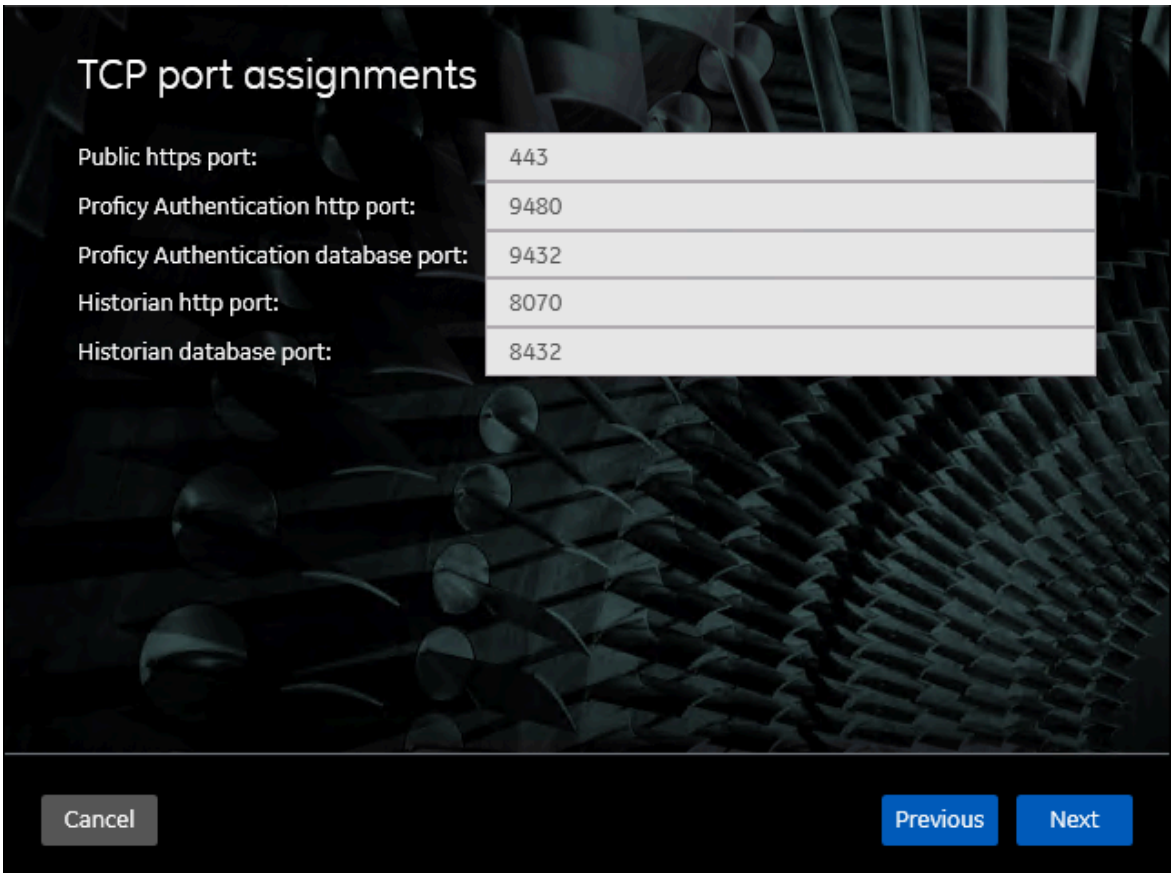
The welcome page appears.

3. Select **Next**.

   The license agreement appears.

4. Select the **Accept** check box, and then select **Next**.

   The **TCP port assignments** page appears.



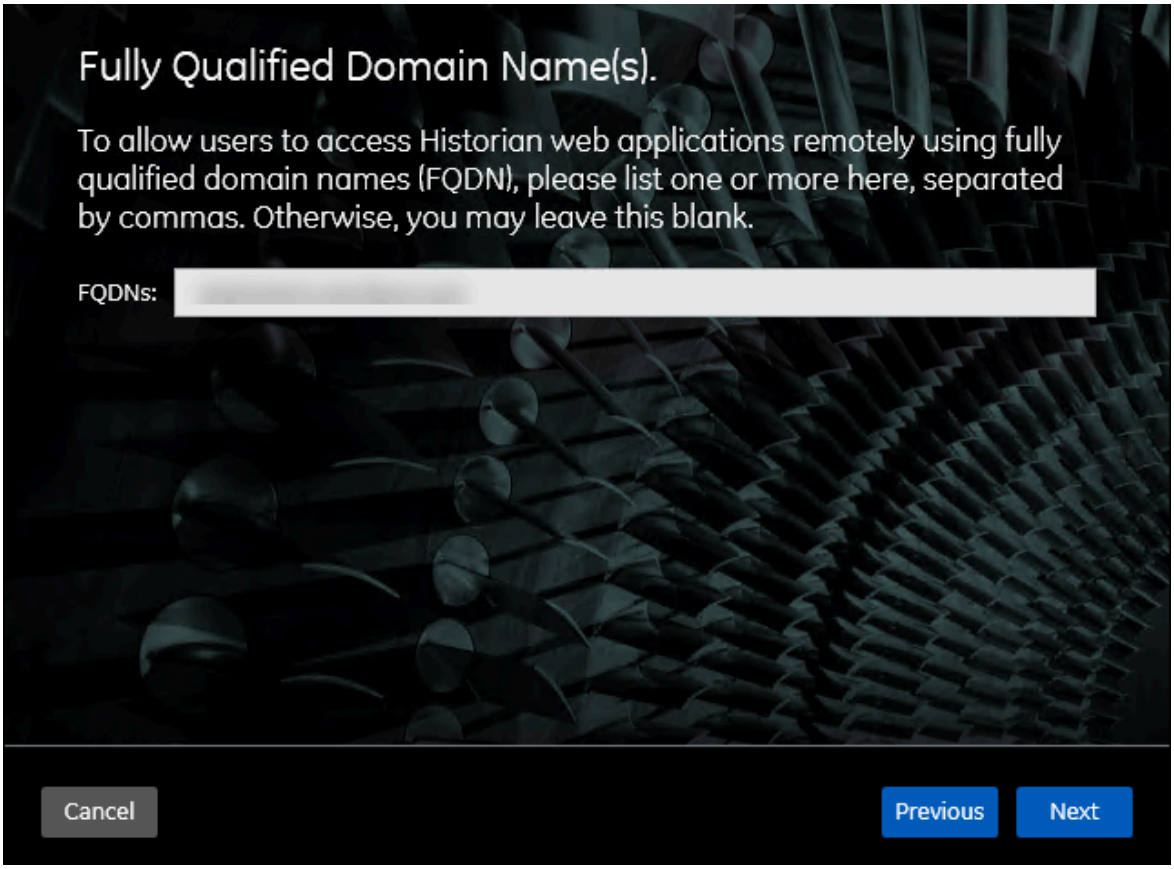

5. As needed, change the values for TCP port assignments as described in the following table, and then select **Next**.

| Field | Description |
|---|---|
| **Public https port** | Port for https protocol communication used by Web-based Clients (through a firewall). The default value is 443. Ensure that this port number matches the one you specify while installing the Historian server. In addition:<br>◦ If you will install Operations Hub later on the same machine, the value that you provide in this field is populated while installing Operations Hub.<br>◦ If you have already installed Operations Hub on the same machine, this field is disabled and populated with the value you have provided while installing Operations Hub. |

| Field | Description |
|---|---|
| **Proficy Authentication http port** | Port for http protocol communication used by the Proficy Authentication service. The default value is 9480. |
| **Proficy Authentication database port** | Port for the Proficy Authentication database. The default value is 9432. |
| **Historian http port** | Port for the http protocol communication used by Web-based Clients. The default value is 8070. |
| **Historian database port** | Port for the PostgreSQL Historian database. The default value is 8432. |

The **Fully Qualified Domain Name(s)** page appears.

- If you will install Operations Hub later on the same machine, the value that you provide in the **FQDNs** field is populated while installing Operations Hub.
- If you have already installed Operations Hub on the same machine, the **FQDNs** field is disabled and populated with the value you have provided while installing Operations Hub.




6. In the **FDQNs** field, enter the fully qualified domain names, and then select **Next**.
   This enables you to access Historian web applications remotely. You can use it to access the Web Admin console using alias names. Enter the values separated by commas.
   To access the Web Admin console using any of the following URLs, enter `Test.abc.ge.com,localhost,127.0.0.1,aliasName`
   - https:// Test.abc.ge.com /historian-visualization/hwa
   - https:// 127.0.0.1 /historian-visualization/hwa
   - https:// aliasName /historian-visualization/hwa
   - https:// localhost /historian-visualization/hwa

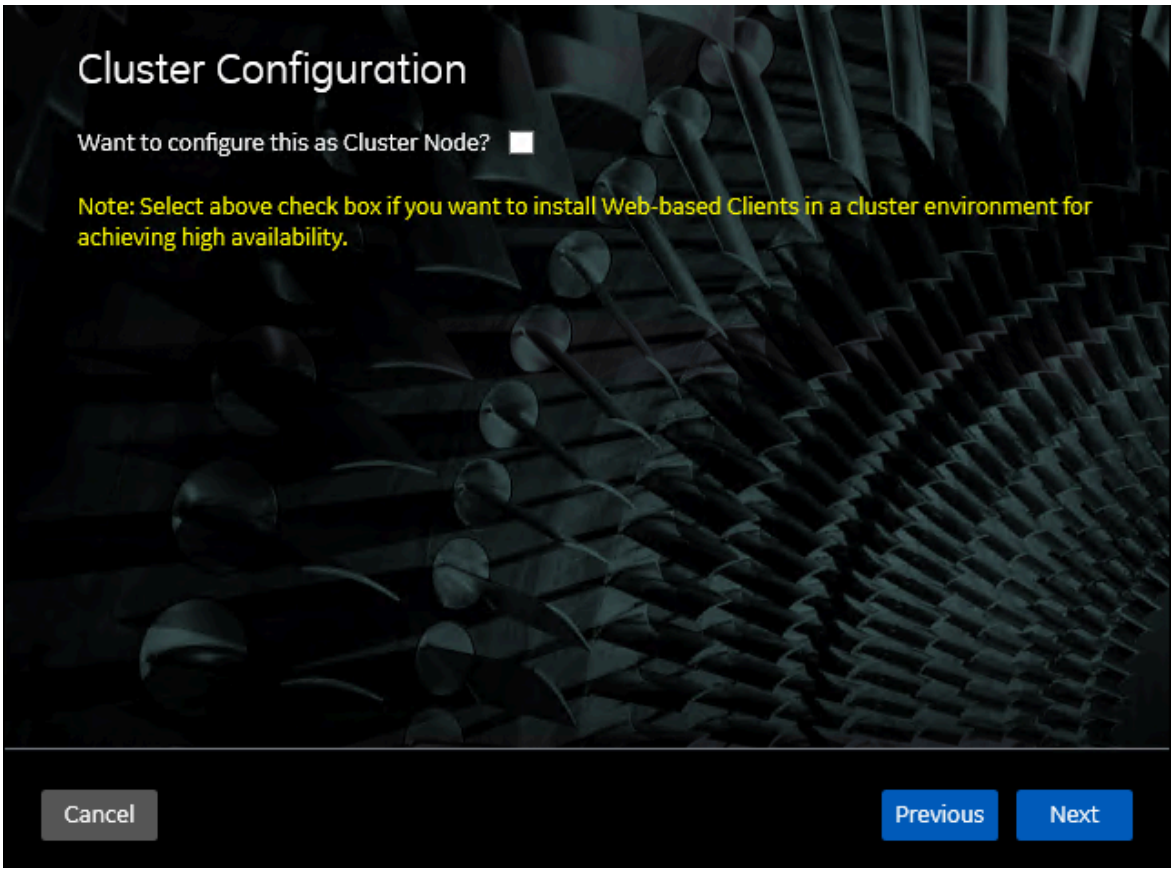
   **Important:**
   - Do not enter a space between the values.
   - You must add the IP address and alias name in the `hosts` file located at `C:\Windows\System32\drivers\etc`. The IP address that you add must be a static or fixed IP address.

     **Format:** `<IP address> <alias name>`

**Example:** `1.2.3.4 myservername`

- FQDN is not supported for Configuration Hub.

The **Cluster Configuration** page appears.



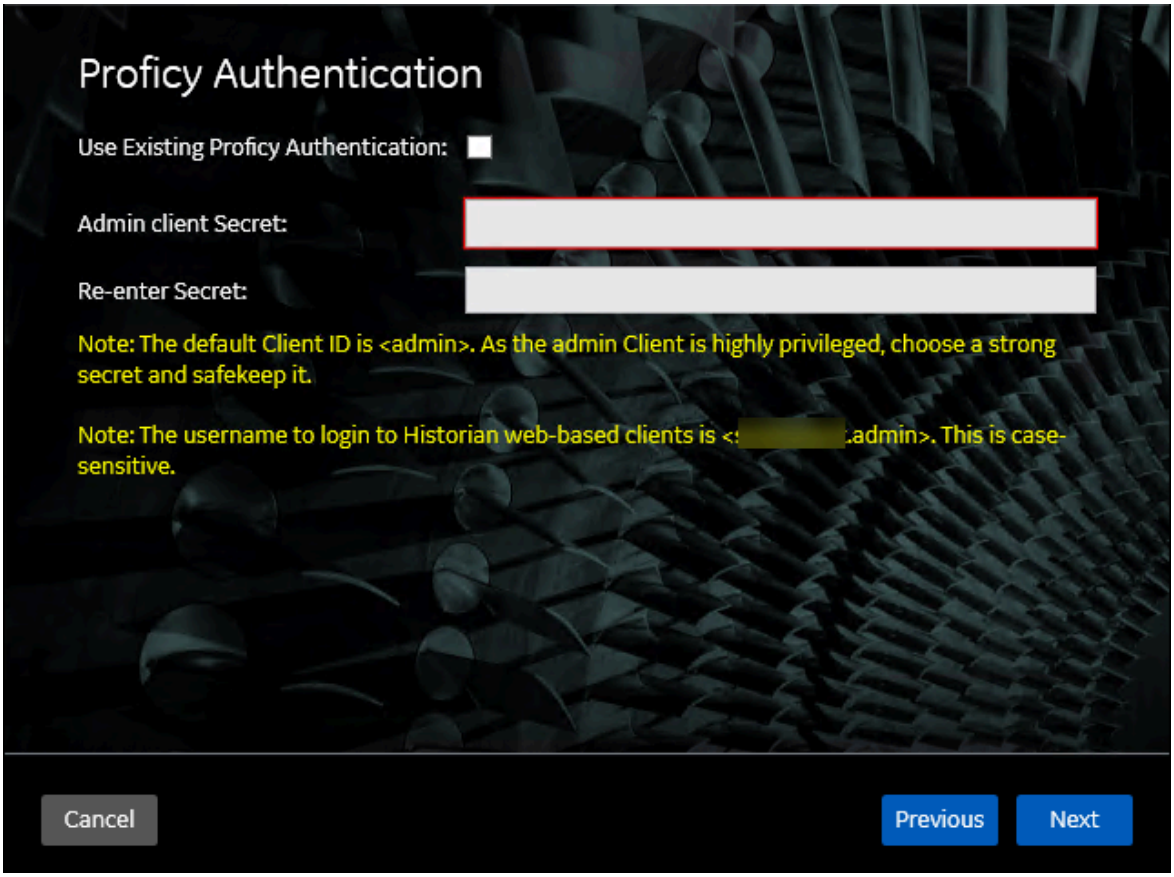

If, however, you are upgrading Web-based Clients, this page does not appear. In that case, skip the next step.

7. If you want high availability of Web-based Clients, select the **Cluster Node** check box, and enter values as described in the following table.

| Field | Description |
|---|---|
| **Historian Database Folder** | Provide the database folder in the shared drive that you have created. The default value is `C:\ProgramData\GE\OperationsHub`. You *must* change this value. |

| Field | Description |
|---|---|
| **Cluster FQDN** | Enter the client access point of the role for which you have added the resources while setting up high availability *(on page 182)*. |
| **Multicast Address** | If needed, modify the common IP address that all the nodes in the cluster can use. Enter a value between 224.0.0.0 and 239.255.255.255 (or a hostname whose IP address falls in this range).The default value is 228.0.0.4. |
| **Historian Cluster Membership Port** | If needed, modify the common port number that all the nodes in the cluster can use. The default value is 45564. This port number, in conjunction with the multicast address, is used to create the cluster. |
| **Historian Cluster Receiver Port** | If needed, modify the multicast port number that you want to use for incoming Historian data. The default value is 4000. |

8. Select **Next**.
   The **Proficy Authentication** page appears, allowing you to choose whether you want to install Proficy Authentication along with Web-based Clients installation or use an existing Proficy Authentication.




- If you want to install Proficy Authentication, clear the **Use Existing Proficy Authentication** check box. If you want to include Proficy Authentication in the cluster, you must install Proficy Authentication locally on each cluster node.
- If you want to use an existing Proficy Authentication server, select the **Use Existing Proficy Authentication** check box. Proficy Authentication is detected if you installed it using a unified installer or Operations Hub, or if Historian uses Proficy Authentication installed remotely from an earlier version.

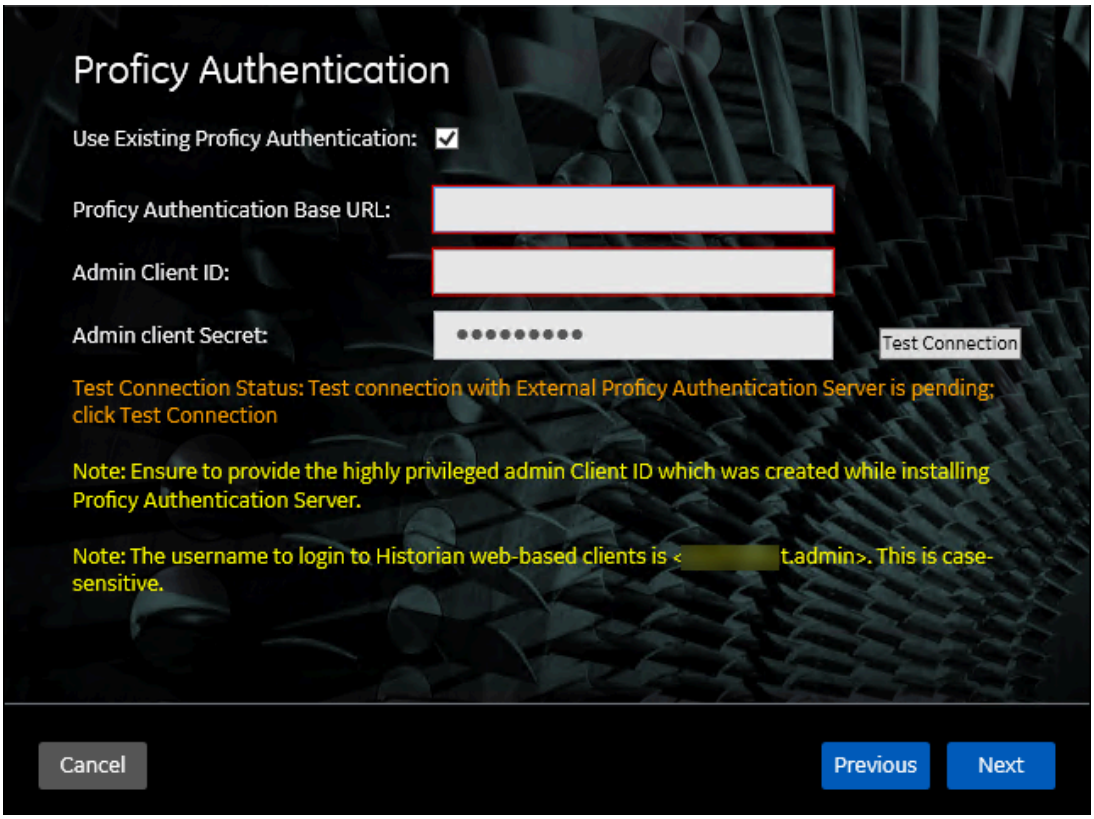
9. If you want to install Proficy Authentication, enter the **Admin client secret**, re-enter the secret, and then select **Next**.
   The admin client secret must satisfy the following conditions:
   - Must not contain only numbers.
   - Must not begin or end with a special character.
   - Must not contain curly braces.

**Note:**
The format of username for Historian Web-based Clients is <host name>.admin, where <host name> is the machine on which Web-based Clients are installed. And, the default client ID is admin. Both the host name and client ID are case-sensitive.

If, however, the Proficy Authentication server hostname is long, resulting in a username longer than 20 characters, Windows does not allow you to create the user. In that case, you can create a Proficy Authentication user, and then create the corresponding Windows user, using the uaa_config_tool utility.

10. Alternatively, if you want to use an existing Proficy Authentication service (that is, a Proficy Authentication instance already installed by an external application such as Operations Hub):

    a. Select the **Use Existing Proficy Authentication** check box.
    The fields for the existing Proficy Authentication service appear.

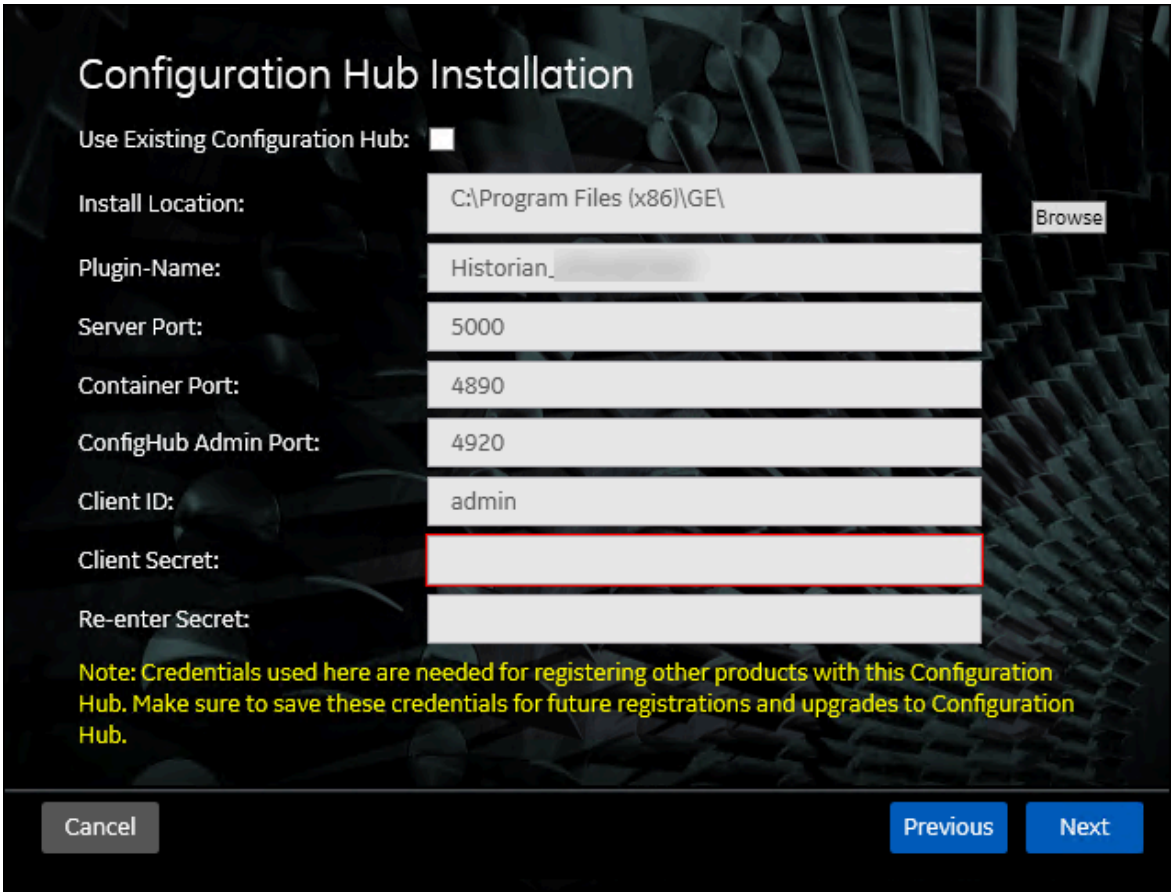



    b. Enter values as described in the following table.

| Field | Description |
|---|---|
| **Proficy Authentication Base URL** | Enter the URL of the external Proficy Authentication server in the following format: https://*<Proficy Authentication server name>*:*<port number>*, where *<Proficy Authentication server name>* is the FQDN or hostname of the machine on which Proficy Authentication is installed. By default, the port number is 443.<br><br>**Note:**<br>Do not enter a trailing slash character. |
| **Admin Client ID** | Enter the client name that you provided while installing the external Proficy Authentication. The default value is admin. |
| **Admin Client Secret** | Enter the client secret that you provided while installing the external Proficy Authentication. |

c. Select **Test Connection**.
The results of the connection test appear. You cannot proceed until the connection is successful.

11. Select **Next**.
The **Configuration Hub Installation** page appears, allowing you to choose whether you want to install Configuration Hub along with Web-based Clients or use an existing Configuration Hub.




Configuration Hub allows you to add and manage a collector instance remotely. For more information, refer to About Configuration Hub.

If, however, an earlier version of Configuration Hub is available on the same machine, you will be prompted to enter the details of the existing Configuration Hub, and it will be upgraded to the latest version. If that happens, skip the next step.

**Important:**
By default, Configuration Hub points to the same Proficy Authentication server as the one you provided during the Historian server installation. If you want to install Web-based Clients in a cluster environment, ensure that:

- Configuration Hub does not use the same Proficy Authentication server as that used by the cluster.
- The Proficy Authentication and Configuration Hub details must be the same for all cluster nodes.

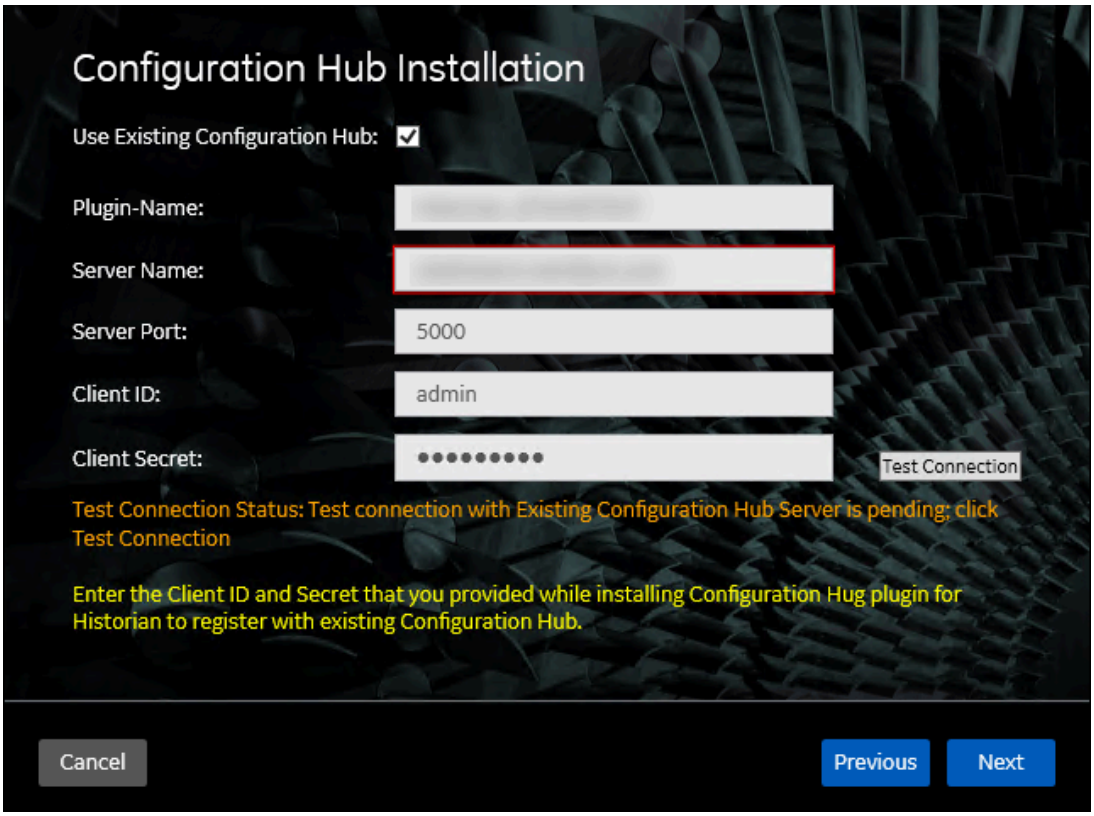
12. If you want to install Configuration Hub, ensure that the **Use Existing Configuration Hub** check box is cleared, and then provide values as described in the following table.

| Field | Description |
|---|---|
| **Install Location** | If needed, modify the installation folder for Configuration Hub. |
| **Plugin Name** | If needed, modify the name of the Configuration Hub plugin for Historian. The default value is in the following format: Historian_<*host name*>. If, however, you are installing Web-based Clients in a cluster environment, the default value is Historian_<*cluster name*>. You can modify this value, but provide the same value for all the nodes in the cluster. |
| **Server Port** | If needed, modify the port number that you want to use for the web server. The default value is 5000. If you want to install Web-based Clients in a cluster environment, provide the same value for all the nodes in the cluster. |
| **Container Port** | If needed, modify the port number for the Configuration Hub container. The default value is 4890. |
| **ConfigHub Admin Port** | This is the port number of the Configuration Hub admin. The default value is 4890. If needed, you can change the port number. |
| **Client ID** | Enter the username to connect to Configuration Hub. The default value is admin. The value that you enter can contain:<br>◦ All English alphanumeric characters (ABCDEFGHIJKLMNOPQRSTUVXYZ abcdefghijklmnopqrstuvwxyz_-0123456789)<br>◦ The following special characters: ><:~!@#$%^&*?\| |

| Field | Description |
|---|---|
| **Client Secret** | Enter the password to connect to Configuration Hub. The value that you enter can contain:<br>◦ Must contain at least eight characters.<br>◦ All English alphanumeric characters (ABCDEFGHIJKLMNOPQRSTUVXYZ abcdefghijklmnopqrstuvwxyz_-0123456789)<br>◦ The following special characters: ><:~!@#$%^&*?\| |
| **Re-enter Secret** | Re-enter the password to connect to Configuration Hub. |

13. Alternatively, if you want to use an existing Configuration Hub:

    a. Select the **Use Existing Configuration Hub** check box. This check box is disabled if an existing Configuration Hub is detected.
    The fields for the existing Configuration Hub appear.

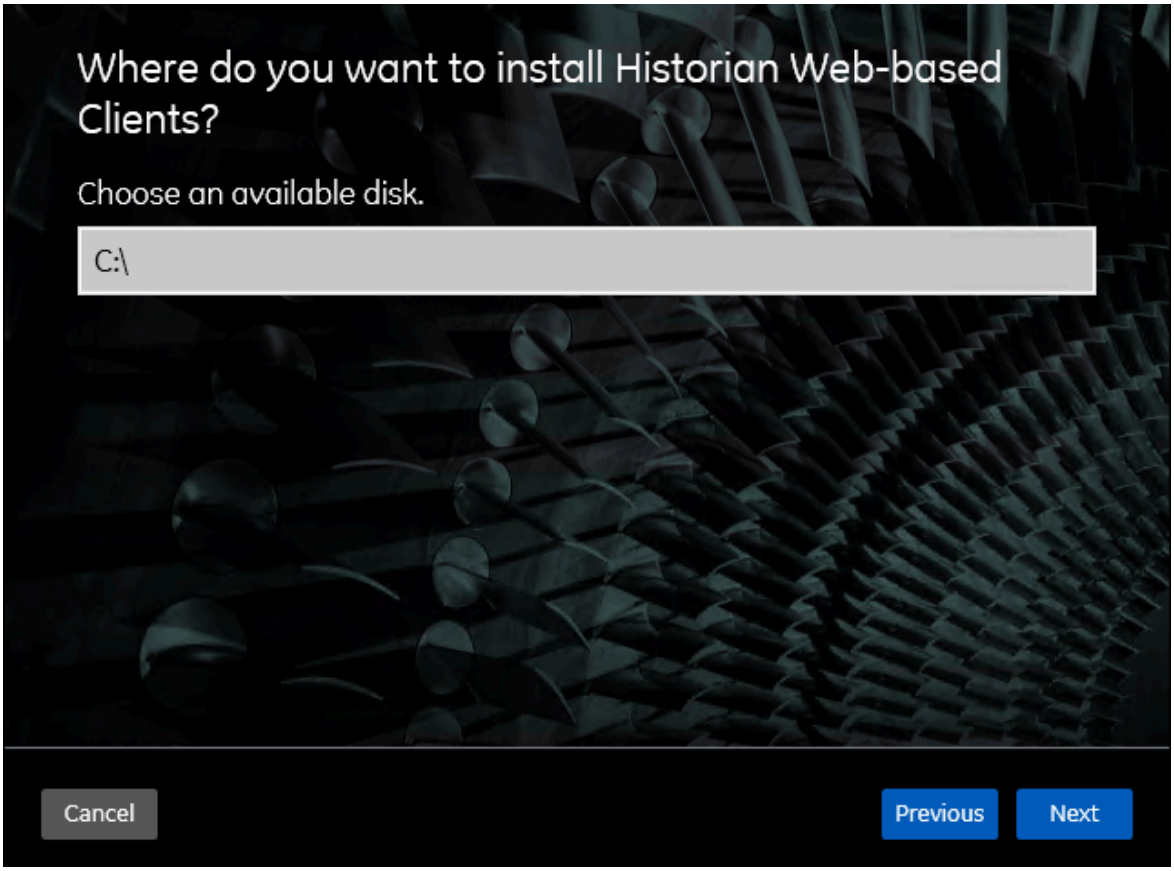


b. Provide values as described in the following table.

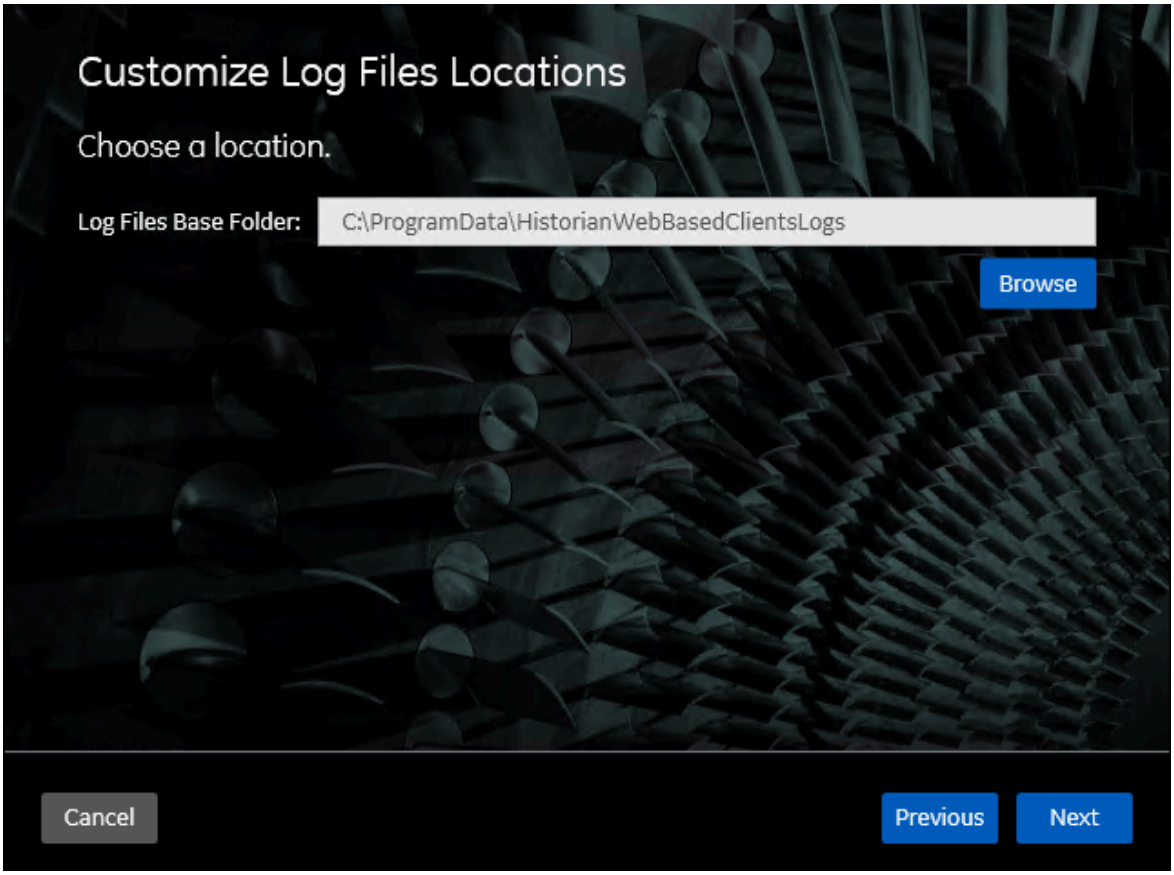
| Field | Description |
|---|---|
| **Plugin Name** | If needed, modify the name of the Configuration Hub plugin for Historian. The default value is in the following format: Historian_-*<host name>* |
| **Server Name** | Enter the server name or the FQDN of the existing Configuration Hub server, as displayed in the address bar of the browser when you access Configuration Hub from the machine where Configuration Hub is installed. |
| **Server Port** | If needed, modify the port number that you want to use for the web server. The default value is 5000. |
| **Client ID** | If needed, modify the username to connect to Configuration Hub. The default value is admin. |
| **Client Secret** | Enter the password to connect to Configuration Hub. |

c. Select **Test Connection**.
The results of the connection test appear. You cannot proceed until the connection is successful.

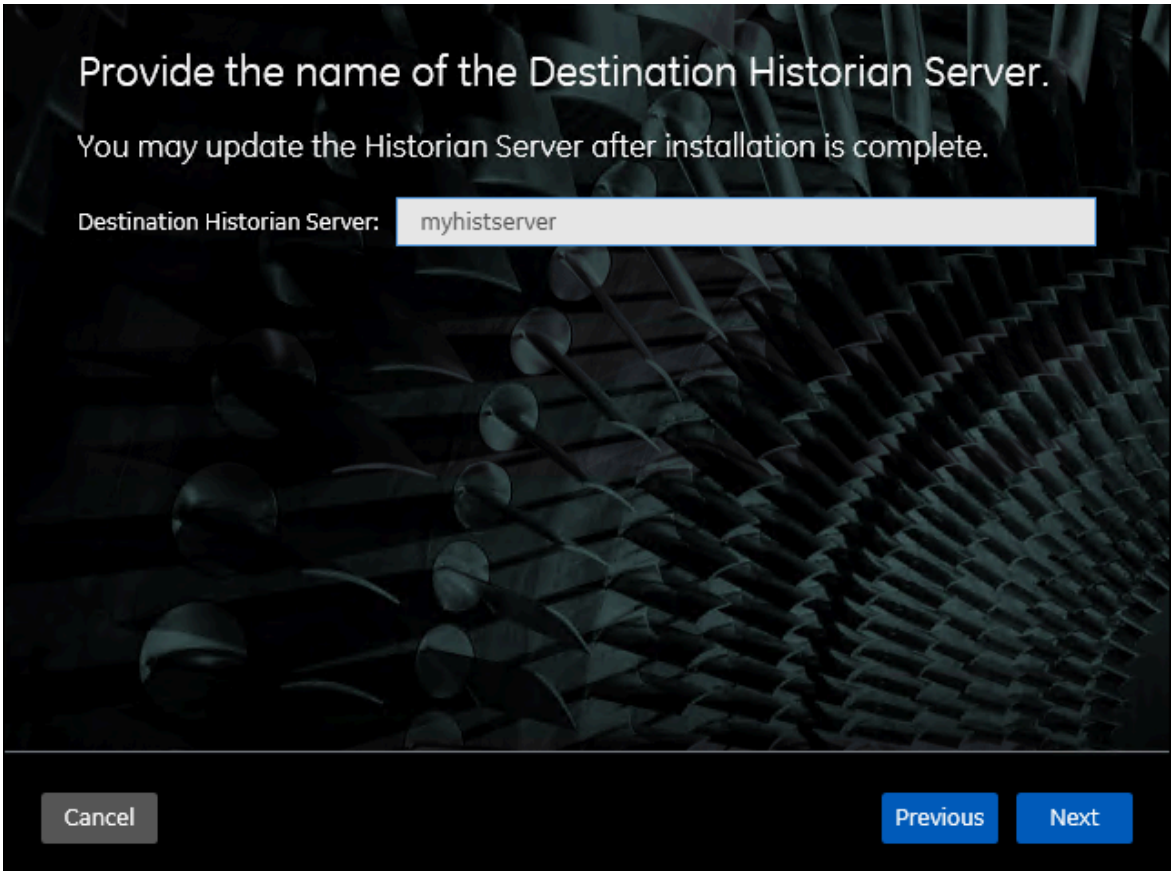
14. Select **Next**.
The default installation drive appears.




15. If needed, change the installation drive for Web-based Clients, and then select **Next**.
    The log files location page appears.




16. If needed, change the location for log files, and then select **Next**.
    The destination Historian server page appears.




17. Provide the name of the destination Historian server to which Web-based Clients are connected by default. When you login to Configuration Hub, the default system will point to this server.

> **Note:**
> - Provide the name of either Historian single-server or mirror primary server because the systems in Configuration Hub will be either a stand-alone system or a horizontally scalable system.
> - If you want to connect to a remote Historian server, you must disable the **Enforce Strict Client Authentication** and **Enforce Strict Collector Authentication** options using Historian Administrator in the remote server.

18. Select **Next**.
    The **You are ready to install** page appears.
19. Select **Install**.
    The Web-based Clients installation begins.
20. When you are prompted to reboot your machine, select **Yes**.

Historian Web-based Clients are installed in the following folder: `<installation drive>:\Program Files\GE`, and the following registry paths are created:

- `HKEY_LOCAL_MACHINE\SOFTWARE\GE Digital`
- `HKEY_LOCAL_MACHINE\SOFTWARE\GE`

If you want to use Configuration Hub installed using other products such as iFIX, Plant Applications, and so on, set up authentication to point to the Proficy Authentication instance.

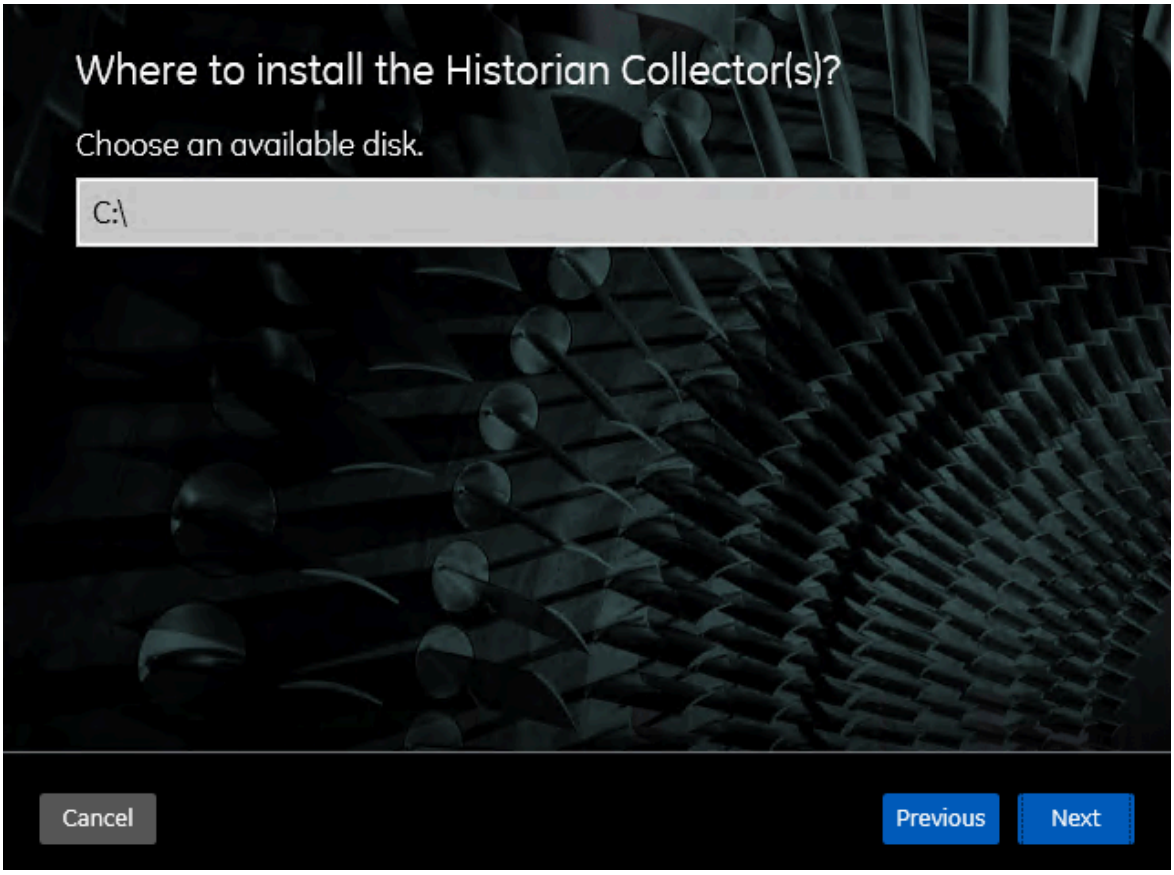
## Install Collectors Using the Installer

After you install collectors, the following artefacts will be available:

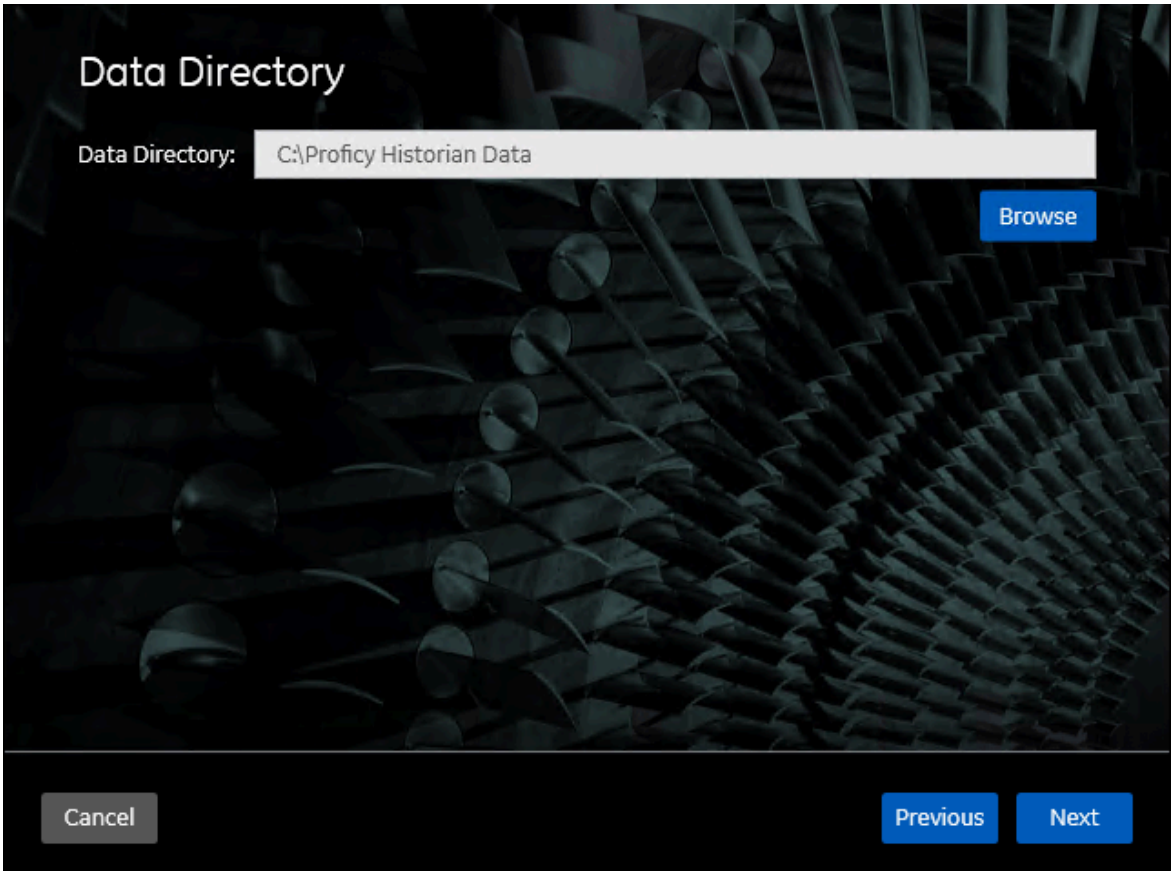
- **Executable files:** These files are required to add a collector instance.
- **Instances of the following collectors:**
  - The iFIX collector
  - The iFIX Alarms & Events collector
  - The OPC Classic Data Access collector for CIMPLICITY
  - The OPC Classic Alarms and Events collector for CIMPLICITY

  These instances will be created only if iFIX and/or CIMPLICITY are installed on the same machine as the collectors.
- **The Remote Collector Management agent:** Provides the ability to manage collectors remotely.

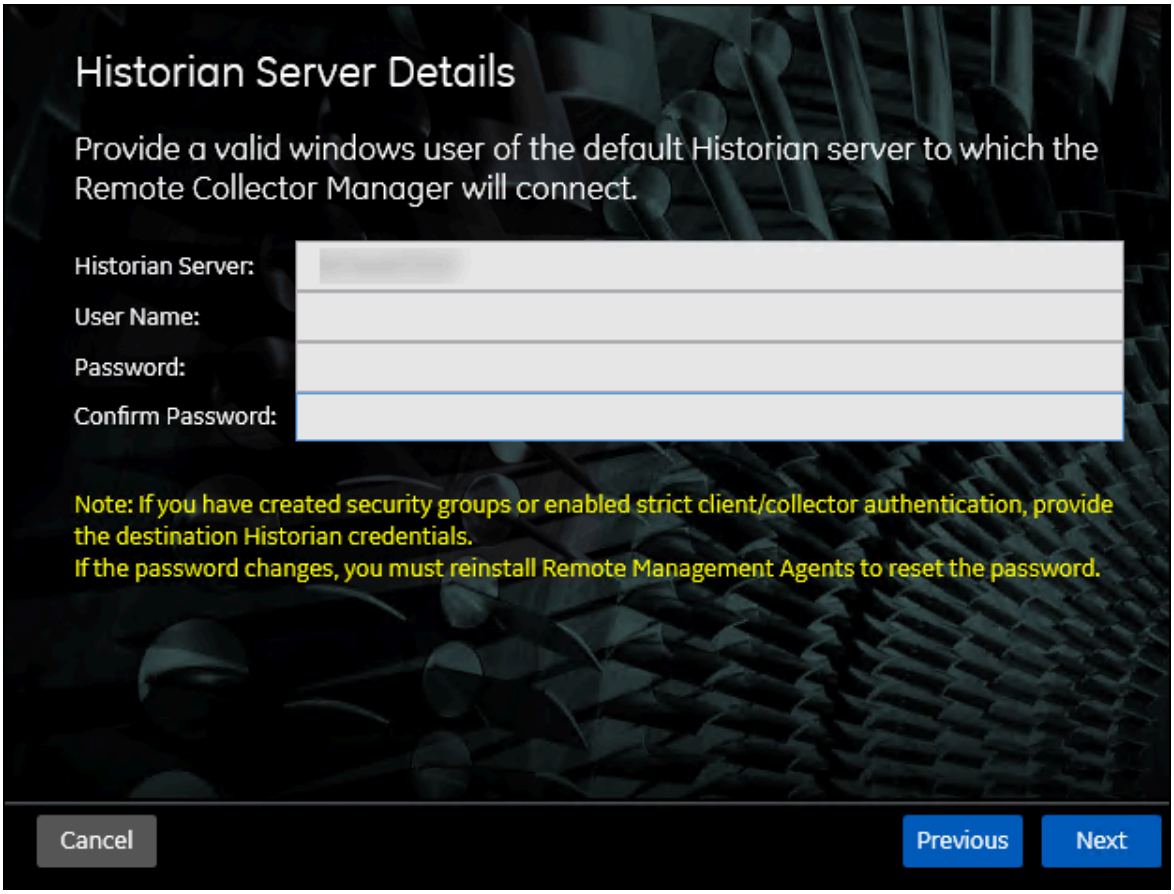
1. Run the `InstallLauncher.exe` file.
2. Select **Install Collectors**.
   The welcome page appears.
3. Select **Next**.
   The license agreement appears.
4. Select the **Accept** check box, and then select **Next**.
   The default installation drive appears.




5. If needed, modify the installation drive, and then select **Next**.
   The data directory page appears.




6. If needed, change the folder for storing the collector log files, and then select **Next**.
   The destination Historian server page appears.




7. Provide the credentials of the Windows user account of the destination Historian server to which you want Remote Management Agent to connect.

   These details are required for Remote Collector Manager to connect to Historian to manage the collectors remotely. If you are installing collectors on same machine as the Historian server, and if strict collector authentication is disabled, you need not provide these details; by default, the machine name of the local Historian server is considered. If, however, they are installed on different machines, or if strict collector authentication is enabled, you must provide the credentials of the Historian server user.

8. Select **Next**.
   The **You are ready to install** page appears.
9. Select **Install**.
   The installation begins.
10. When you are prompted to reboot your system, select **Yes**.

The collector executable files are installed in the following folder: `<installation drive>:\Program Files (x86)\GE Digital\<collector name>`. The iFIX collectors are installed in the following folder: `C:\Program Files\GE\iFIX`. The following registry paths are created:

- `HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\ GE Digital\iHistorian\Services\<collector type>`
- `HKEY_LOCAL_MACHINE\SOFTWARE\GE Digital\iHistorian\Services\<collector type>`

In addition, if iFIX and/or CIMPLICITY are installed on the same machine as the collectors, instances of the following collectors are created:

- The iFIX collector
- The iFIX Alarms & Events collector
- The OPC Classic Data Access collector for CIMPLICITY
- The OPC Classic Alarms and Events collector for CIMPLICITY

## Perform Post-Installation Tasks

1. If you do not want strict authentication, disable the **Enforce Strict Client Authentication** and **Enforce Strict Collector Authentication** options under Historian Administrator > **Data Stores** > **Security**.
2. While installing the Historian server, if you have allowed the installer to create Historian security groups, create a local Windows user with the format <Web-based Clients server name>.admin, and add the user to the ihSecurity Admins group. This user will log in to Web-based Clients. Alternatively, you can create Proficy Authentication users in an external Proficy Authentication and map their security groups. For information, refer to About Proficy Authentication Groups.

   Depending on whether the Historian server will use local or domain security groups, select the appropriate option in Historian Administrator.
3. Ensure that the Windows user that you have specified while installing collectors is added to the iH Security Admins and iH Collector Admins groups.
4. Enable trust for a client certificate for Configuration Hub.
5. Enable trust for a self-signed certificate on Chrome.
6. Import an issuer certificate.

You are now ready to use Configuration Hub.

Access Configuration Hub *(on page 336)*.

## Upgrade Configuration Hub

If you install Web-based Clients before uninstalling the previous version, you cannot modify the Configuration Hub credentials. If an earlier version of Configuration Hub is available on the same machine, you will be allowed to use the same; you cannot install Configuration Hub again.

1. Uninstall Configuration Hub.
2. Set up Configuration Hub *(on page 305)*.

## Access Configuration Hub

Perform the tasks outlined in About Setting up Configuration Hub *(on page 305)*.

1. Double-click the Configuration Hub icon on your desktop ( ).
   The Configuration Hub login page appears.
2. Depending on whether you want to use Proficy Authentication or custom authentication, select the appropriate tab. If custom authentication is not applicable, skip this step.

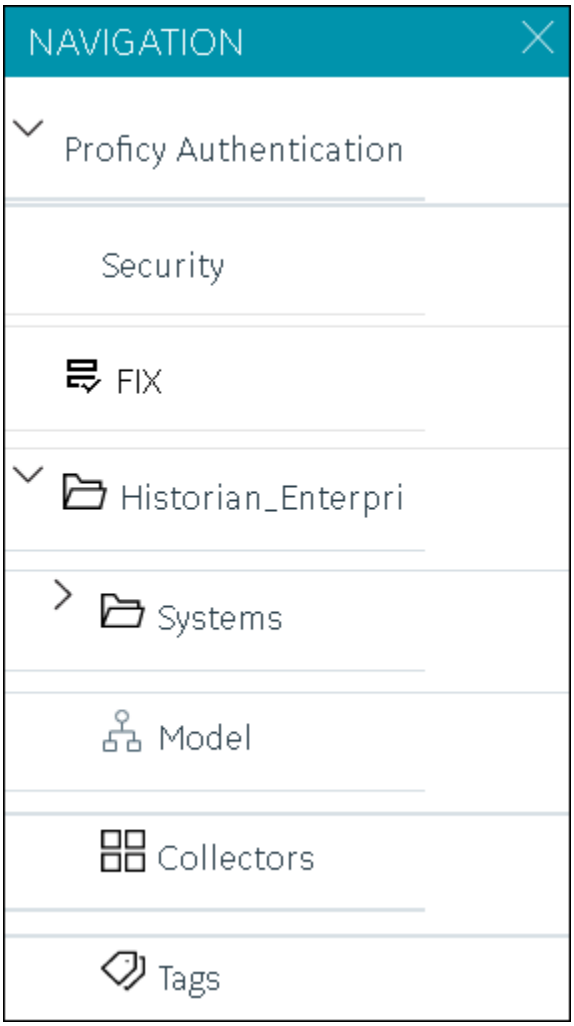
   **Note:**
   For instructions on setting up authentication, refer to https://www.ge.com/digital/documentation/confighub/version2024/t_authentication_setup.html

3. Select the Configuration Hub node that you want to access, and then select **Continue to Login**.
   The Proficy Authentication login page appears.
   If you cannot access the login page, start the GE Operations Hub Httpd Reverse Proxy and the Data Archiver services.
4. Log in with your credentials.

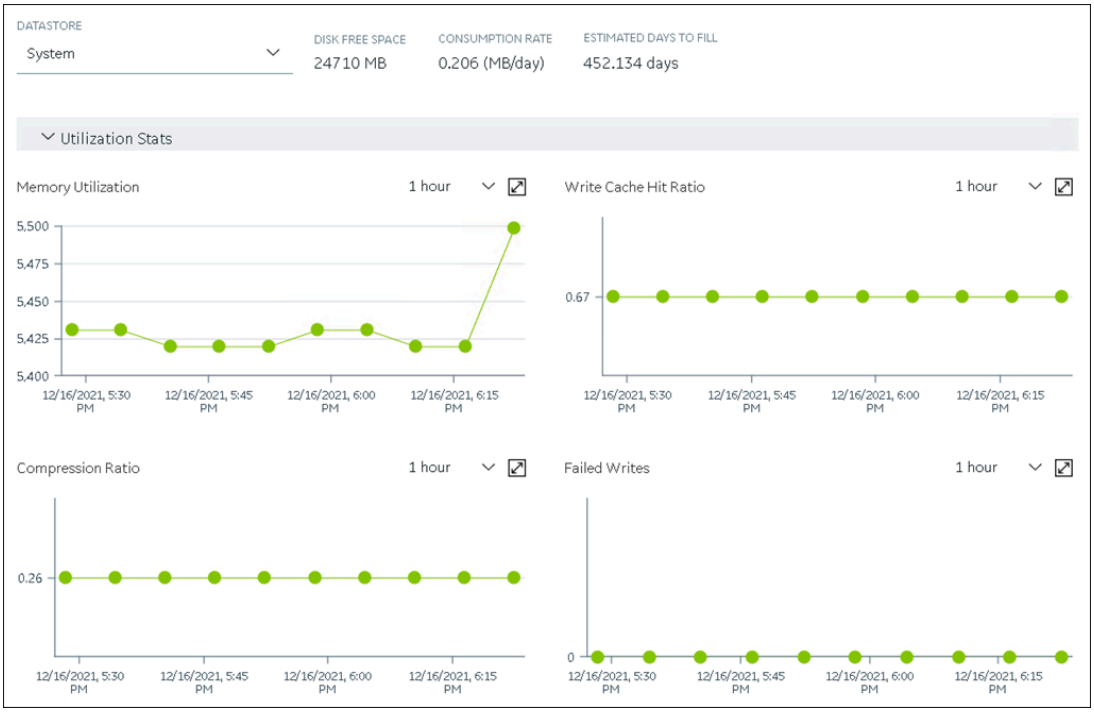
   **Note:**
   By default, the username is *<host name>*.admin, and the password is the value that you have entered in the **Admin client secret** field on the **Proficy Authentication Service** page during Web-based Clients installation.

   The Configuration Hub application appears, displaying the following sections:

- **The Navigation section:** Contains a list of systems that you have added. In addition, it helps you navigate to the Model, Collectors, and Tags sections. You can also access Proficy Authentication to create users and groups.

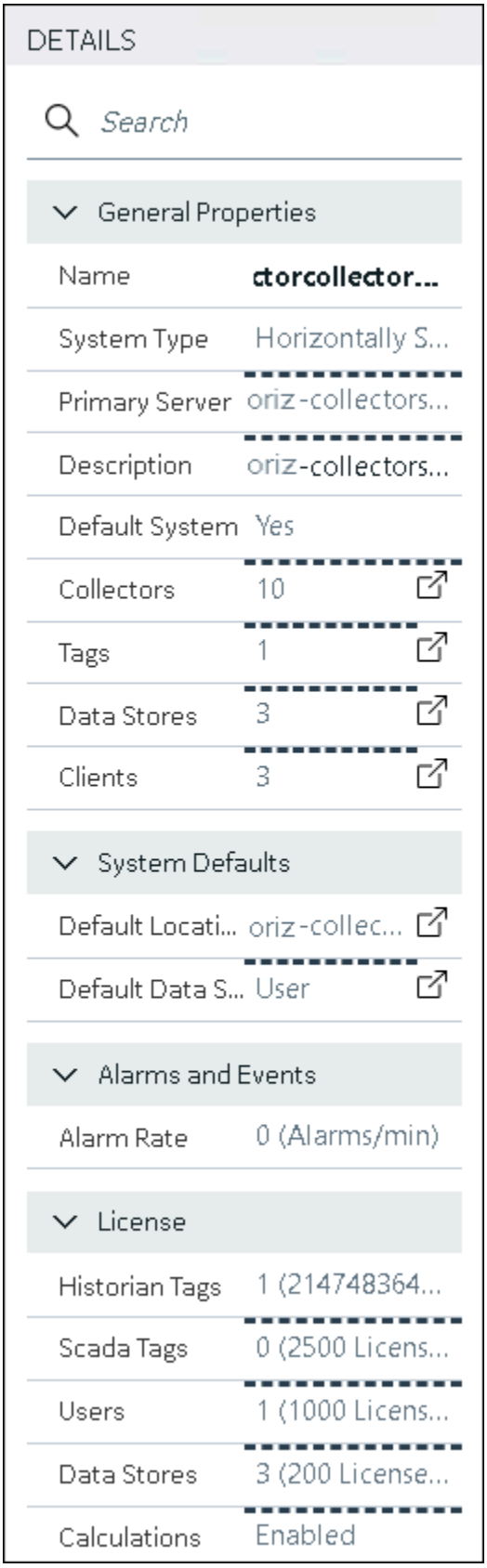



- **The main section:** Displays content based on your selection in the **NAVIGATION** section. For example,if you select a Historian system, you can access a list of servers in the system. You can also navigate to the system statistics as shown in the following image.




Similarly, if you select **Model** in the **NAVIGATION** section, you can access the Historian model.

- **The Details section:** Contains the details of the item selected in the main section. For example, If you select a system, you can view the description of the system, and add data stores and mirror locations using the **Details** section.


DETAILS
Search
General Properties
Name
ctorcollector...
System Type
Horizontally S...
Primary Server
oriz-collectors...
Description
oriz-collectors...
Default System
Yes
Collectors
10
Tags
1
Data Stores
3
Clients
3
System Defaults
Default Locati...
oriz-collec...
Default Data S...
User
Alarms and Events
Alarm Rate
0 (Alarms/min)
License
Historian Tags
1 (214748364...
Scada Tags
0 (2500 Licens...
Users
1 (1000 Licens...
Data Stores
3 (200 License...
Calculations
Enabled

Depending on your requirements, set up a stand-alone system or a horizontally scalable system.

# Historian Plugin Management in Configuration Hub

## About Historian Plugin Management in Configuration Hub

The Historian plugin in Configuration Hub enables you to perform several tasks like monitor, supervise, retrieve, and control gathering functions from a server, client, or one or more remote nodes. However, it operates as a web-based application.

When you install the Historian server and web-based clients, the Configuration Hub application is installed along with Proficy Authentication. Additionally, the Historian node and plugin are automatically registered with the Proficy Authentication and Configuration Hub based on the Historian HTTP and database port numbers.

After installing the Historian Server and the Web-based clients, if you access the Configuration Hub, you can see the Historian plugin displayed in the **NAVIGATION** pane. Also, the Historian node and the plugin can be seen in the **Administration** plugin > **Node Manager**.

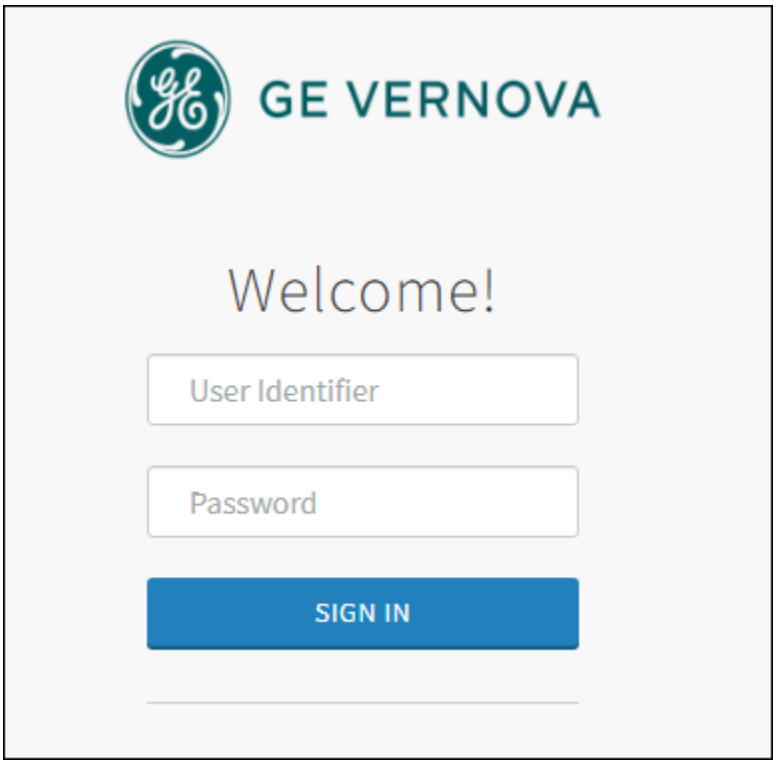
**Note:**
However, if during installation, the Historian node did not properly register with the Proficy Authentication and Configuration Hub, and did not appear in the Node Manager, you can add a Historian node *(on page 344)* on Configuration Hub using the Node Manager.

The Node Manger consolidates control over product and license details, and you can view the corresponding Historian node's certificate, and license details. Using the Node Manager, if needed, you can modify a plugin display name *(on page 347)*, unregister a plugin *(on page 350)*, and if you want to register it at a later stage, you can register the plugin *(on page 348)* again.

## View Historian Node Details

Using the Historian node, you can view the Historian node-specific certificate and license details, and also plugin-specific details.

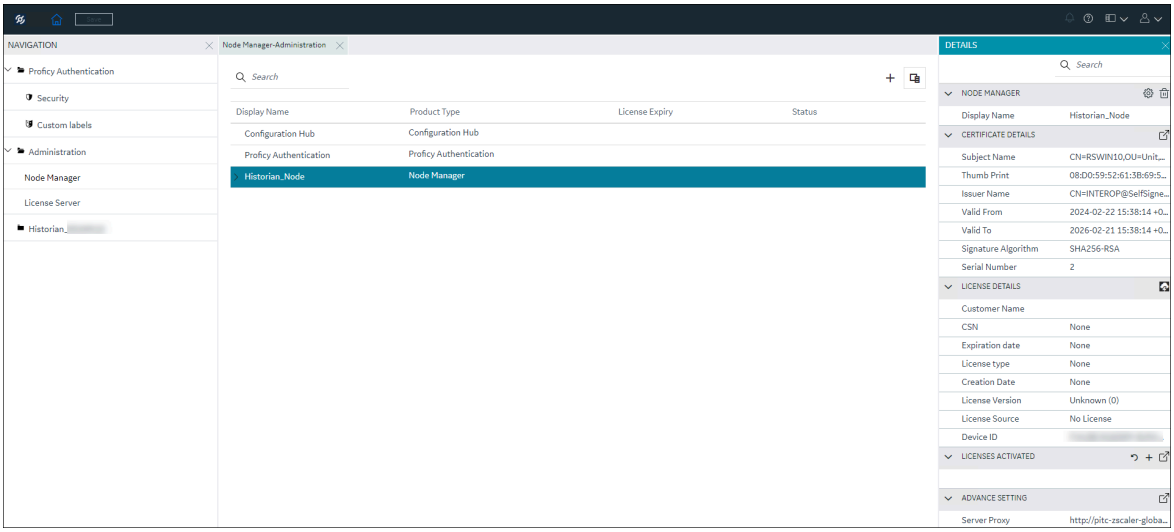
1. Double-click the Configuration Hub icon on your desktop ( ).
   The Configuration Hub login page appears.



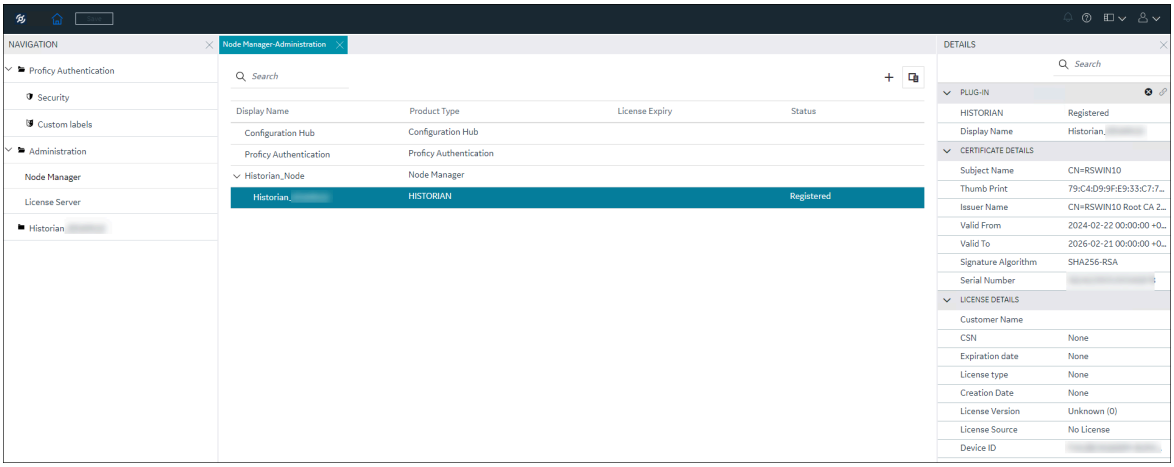

2. Login with the default user credentials. That is, <hostname>.admin.

The Configuration Hub application appears, listing the Historian plugin in the **NAVIGATION** pane.

3. In the **NAVIGATION** pane, select and expand **Administration**, and then select **Node Manager**. The Node Manager administration page appears, listing the available Historian node and the plugin.
4. To view the Historian node-specific license details, select the node. The node-specific details appear in the right-side section. Also, the license details are displayed in the **License Expiry** and **Status** columns.




5. To view the plugin-specific details, expand the node and select the plugin. The plugin-specific details appear in the right-side section.

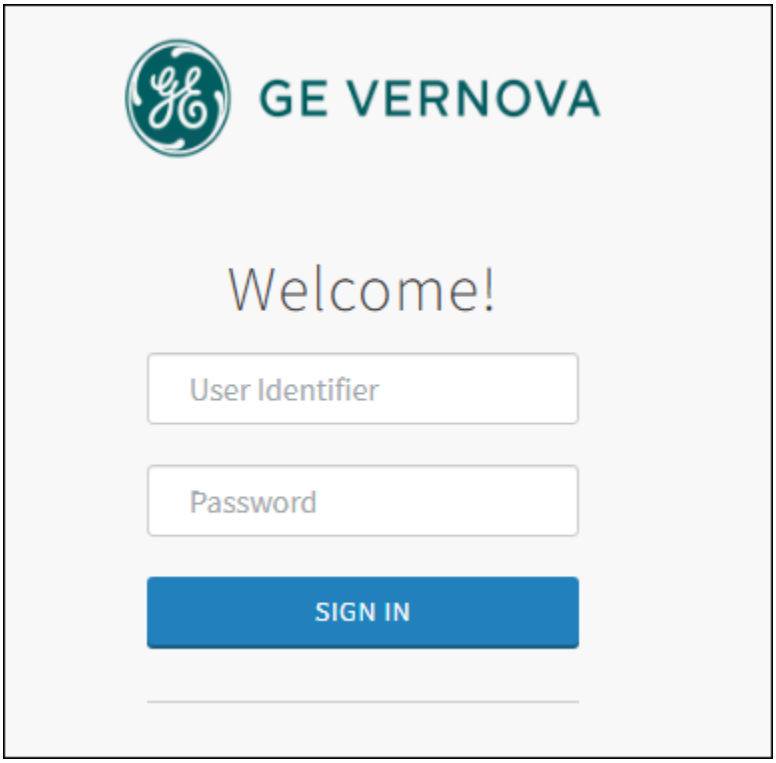



If you have the needed permission, you can perform additional certificate and node management tasks. For more information, refer to **Administration Plugin** in the Configuration Hub help.

## Add a Historian Node (Optional)

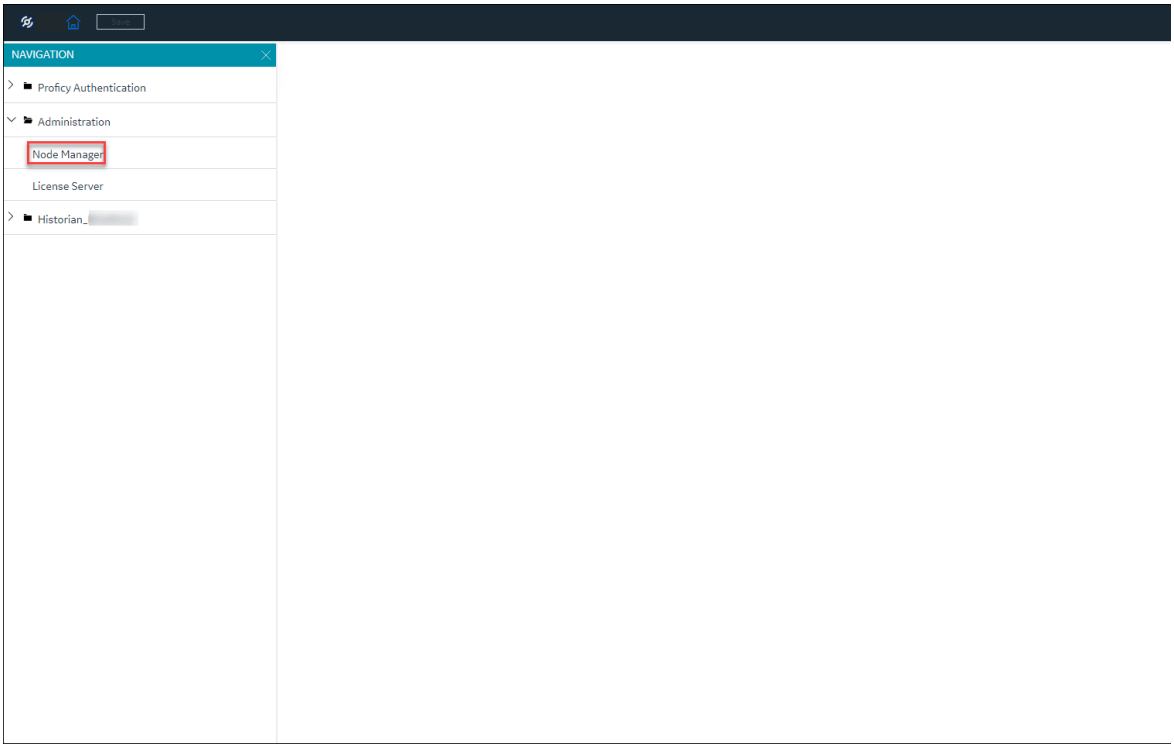
The steps listed in this topic are only needed if the Historian node was not properly configured during the installation and is not being displayed in the Node Manager.

1. Double-click the Configuration Hub icon on your desktop ( ).
   The Configuration Hub login page appears.

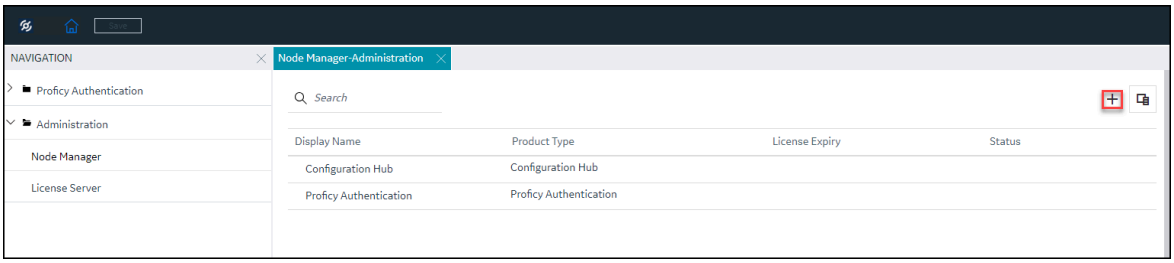



2. Login with the default user credentials. That is <hostname>.admin.
   The configuration hub application appears, listing the Historian plugin in the **NAVIGATION** pane.

3. In the **NAVIGATION** pane, select and expand **Administration**, and then select **Node Manager**.

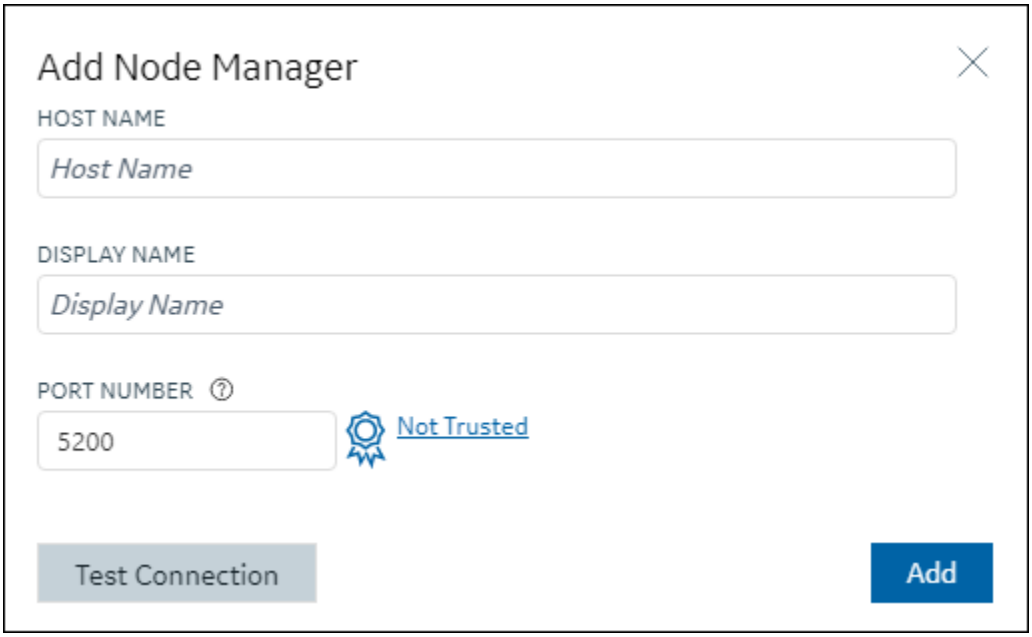



The node manager administration page appears.



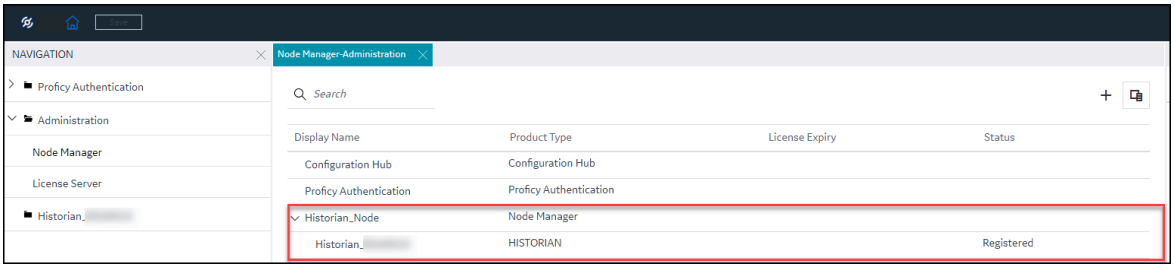

4. In the upper-right corner, select + .

   The **Add Node Manager** window appears.




5. Enter the **HOST NAME**. Here, it is the Historian node host name in a fully qualified domain name format. For example, testmachine123.testdomain.com.
6. Enter the **DISPLAY NAME** for the Historian node.
7. Enter the **PORT NUMBER** of the host that you entered.
8. You must trust the node manager certificate. To trust the certificate, select **Not Trusted**.
   The **Certificate Details** window appears, listing the certificate information.
9. Read the certificate details and if you trust, select **Trust**.
   In the **Add Node Manager** window, the certificate status changes to trusted.
10. Select **Test Connection**.
    If the connection is successful, a success message appears. If not, check the host name and the port are correct.
11. Select **Add**.
    The Historian node along with the plugin are added.

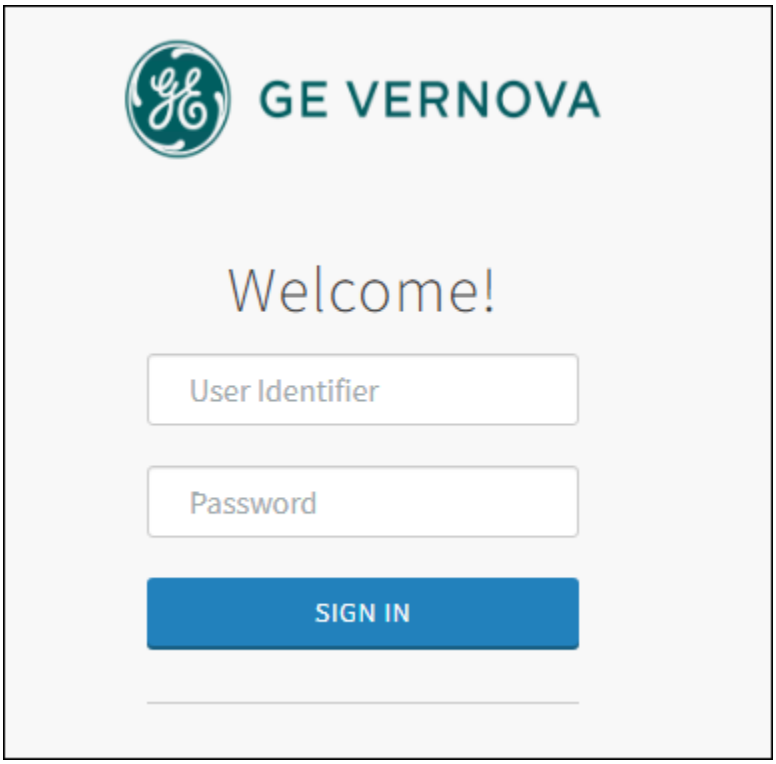
    By default, the plugin gets registered.




In rare cases, for some reason, if the plugin is not registered, you can register the plugin *(on page 348)*.

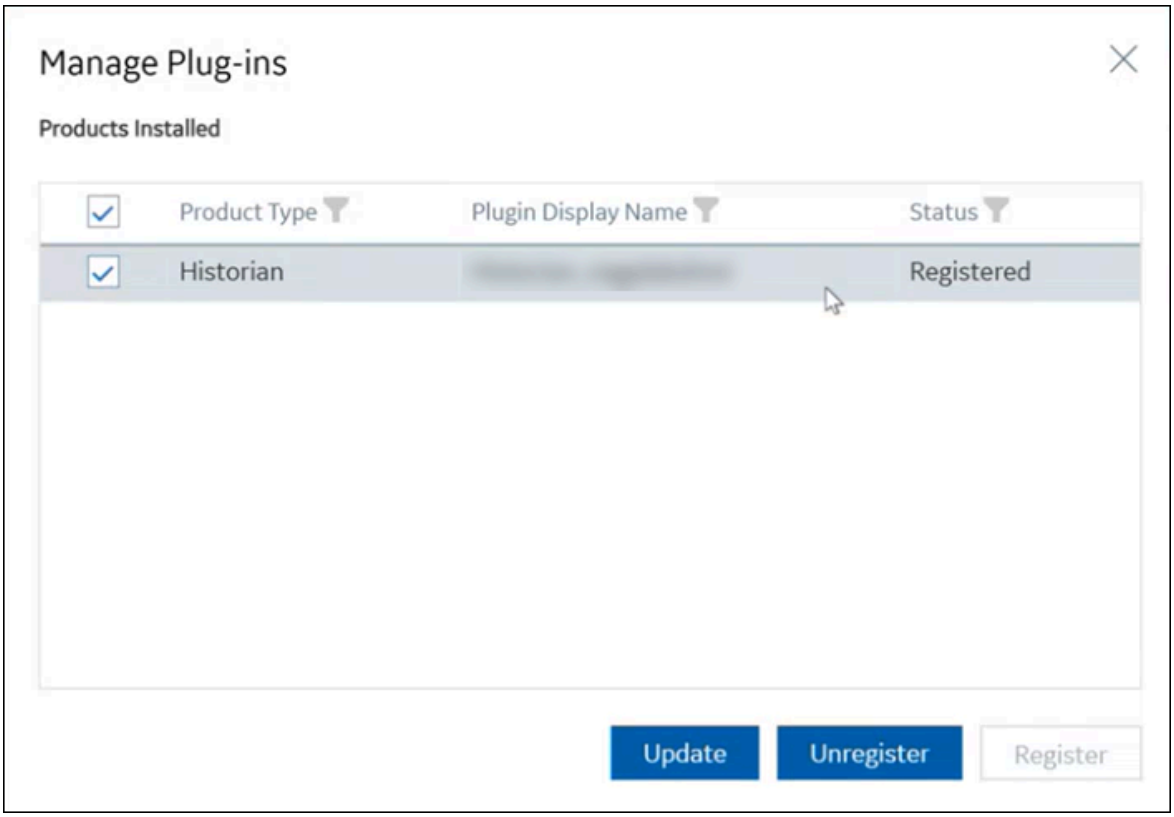
## Modify a Historian Plugin Display Name

1. Double-click the Configuration Hub icon on your desktop ( ).
   The Configuration Hub login page appears.




2. Login with the default user credentials. That is <hostname>.admin.
   The configuration hub application appears, listing the Historian plugin in the **NAVIGATION** section.
3. In the **NAVIGATION** pane, select and expand **Administration**, and then select **Node Manager**.
   The Node Manager administration page appears, listing the available Historian node and the plugin.
4. Select and right-click the node, and then select **Manage**.

The **Manage Plug-ins** window appears.

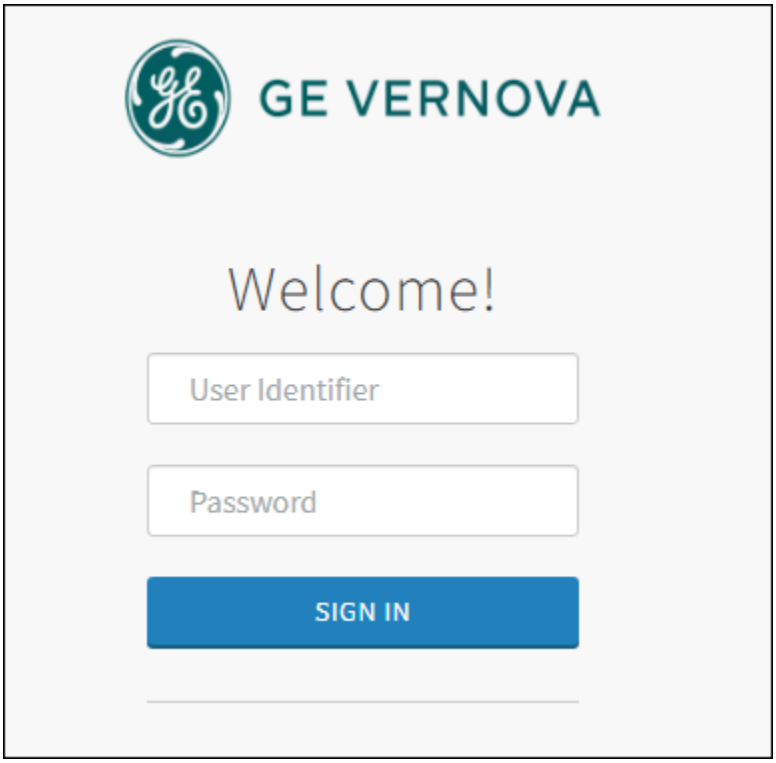



5. Change the plugin display name and select **Update**.
   The changes you made are updated and applied to the plugin.

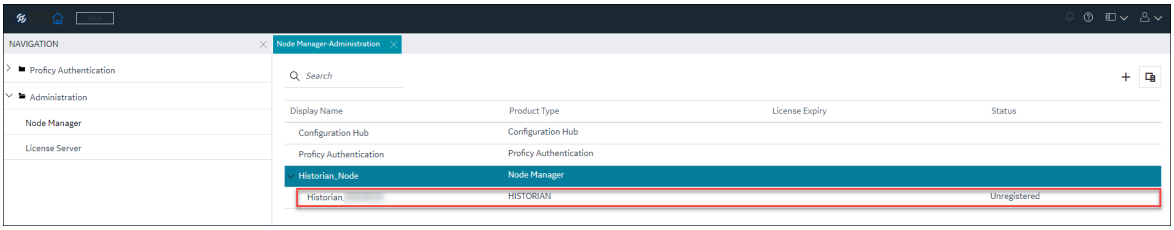
## Register a Historian Plugin

If you had unregistered a plugin and want to register it again, or if, for some reason, the plugin was not registered after installation, you can register the plugin using the Node Manager, provided that you have the Historian node in the Node Manager. If the Historian node did not register properly, you must first add a Historian node *(on page 344)* and then register the Historian node and its plugin.

1. Double-click the Configuration Hub icon on your desktop ( ).
   The Configuration Hub login page appears.



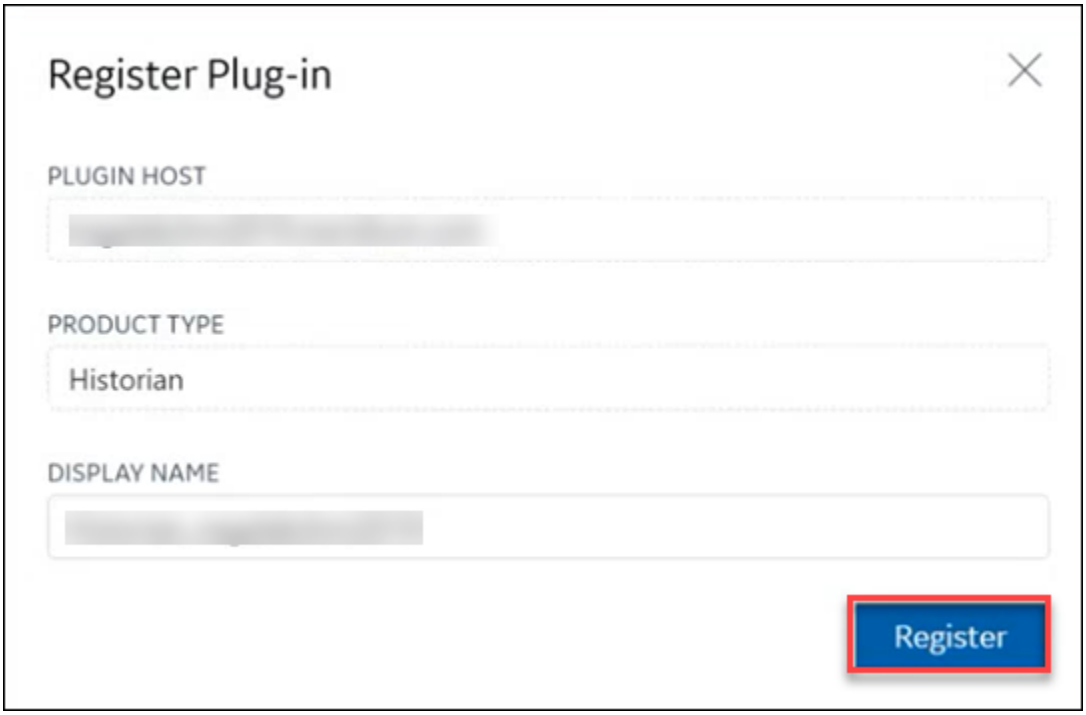

2. Login with the default user credentials. That is <hostname>.admin.
   The configuration hub application appears, listing the Historian plugin in the **NAVIGATION** section.
3. In the **NAVIGATION** pane, select and expand **Administration**, and then select **Node Manager**.
   The Node Manager administration page appears, listing the available Historian node and the plugin.




4. Select and right-click the plugin, and then select **Register**.
   Alternatively, you can select and right-click the node, and then select **Manage**.

   You can register the plugin using the **Register** button available in the **Manage Plug-ins** window.

   The **Register Plug-in** window appears.




5. Enter the values as described in the following table:

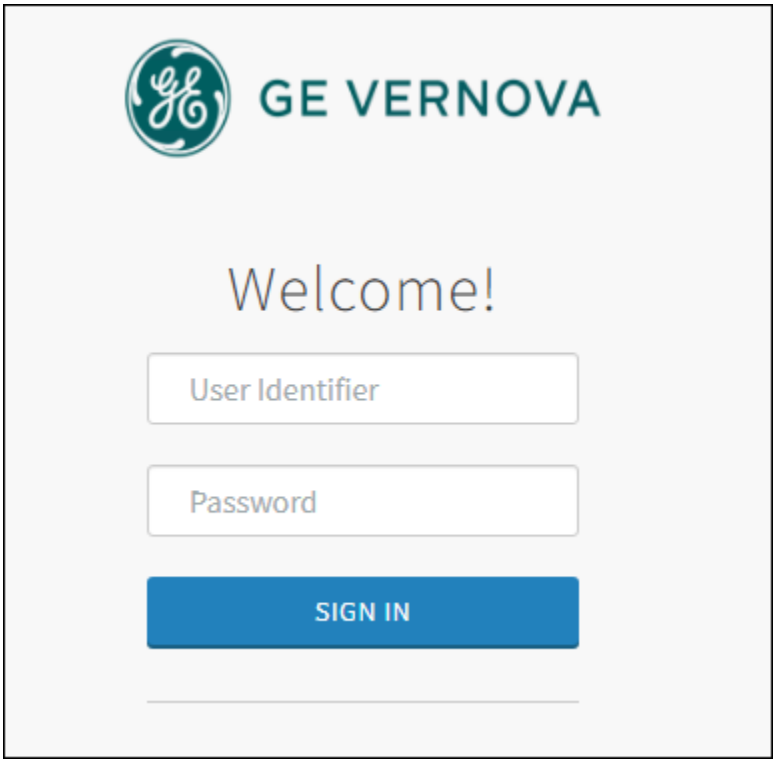
| Field | Description |
|---|---|
| **PLUGIN HOST** | The host name of the plugin in a fully qualified domain name format. |
| **PRODUCT TYPE** | The product type for the plugin. For example, Historian. |
| **DISPLAY NAME** | The plugin name that you want to see below the Node Manager. |

6. Select **Register**.
   The plugin gets registered and added in the **NAVIGATION** pane.
7. For the plugin to work, you must refresh the browser or log out and log in again to restart Configuration Hub.
8. Now try accessing the plugin.

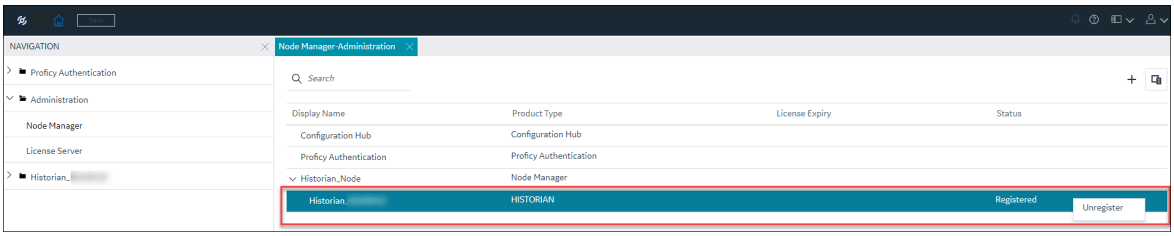
## Unregister a Historian Plugin

If you need to unregister a plugin, you can do that using the Node Manager.

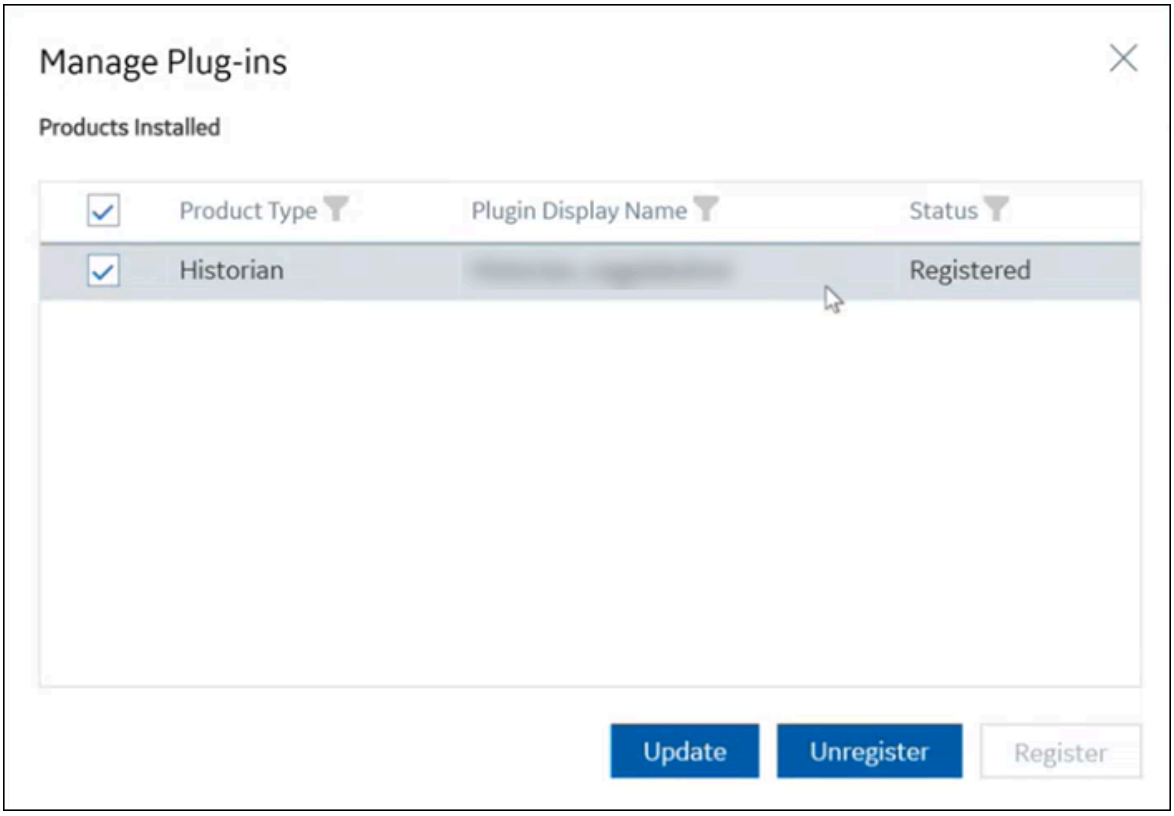
1. Double-click the Configuration Hub icon on your desktop ( ).
   The Configuration Hub login page appears.




2. Login with the default user credentials. That is <hostname>.admin.
   The configuration hub application appears, listing the Historian plugin in the **NAVIGATION** pane.
3. In the **NAVIGATION** pane, select and expand **Administration**, and then select **Node Manager**.
   The node manager administration page appears.




4. Select and right-click the plugin, and then select **Unregister**.
   Alternatively, you can select and right-click the node, and then select **Manage**.
   The **Mange Plug-ins** window appears, listing all the available plugins.

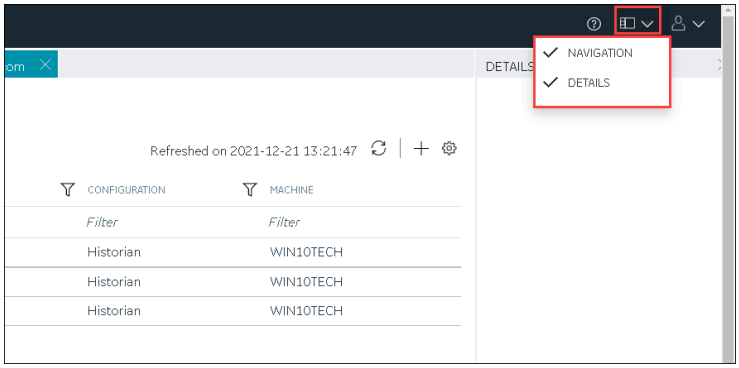
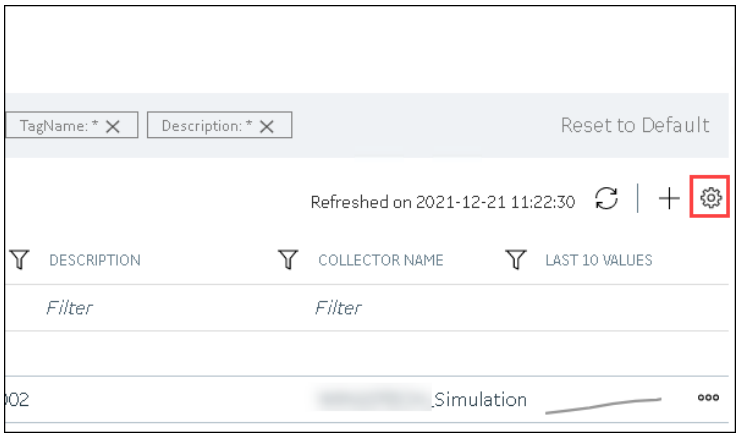


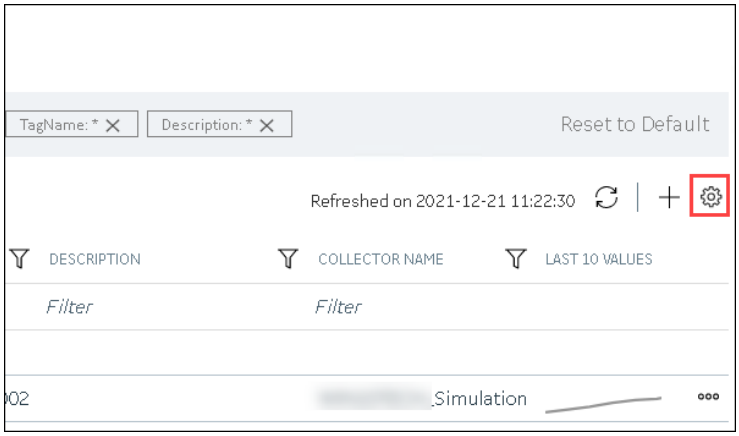
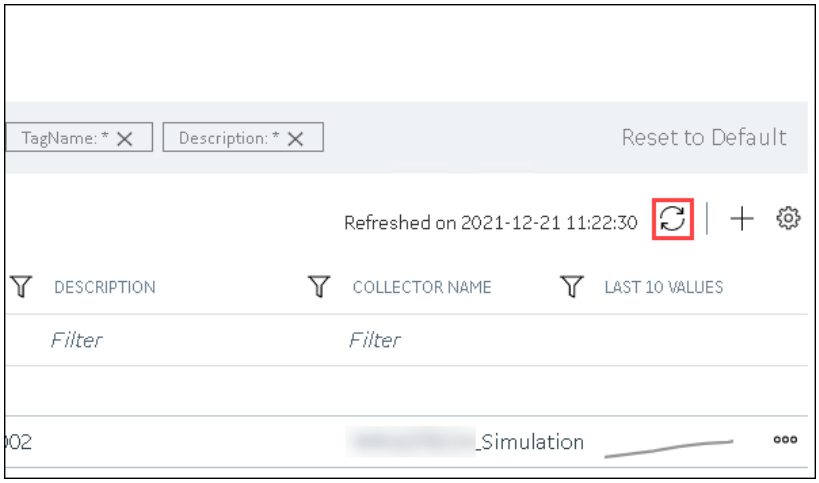

5. Select the plugin as needed.
6. Select **Unregister**.
   The plugin gets unregistered.

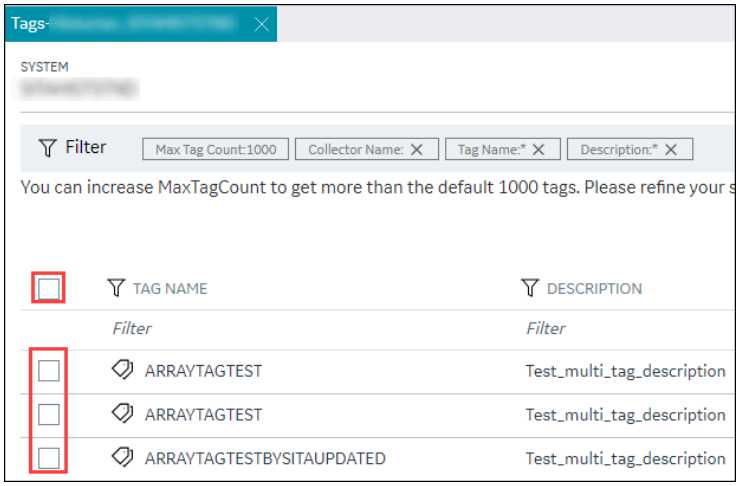
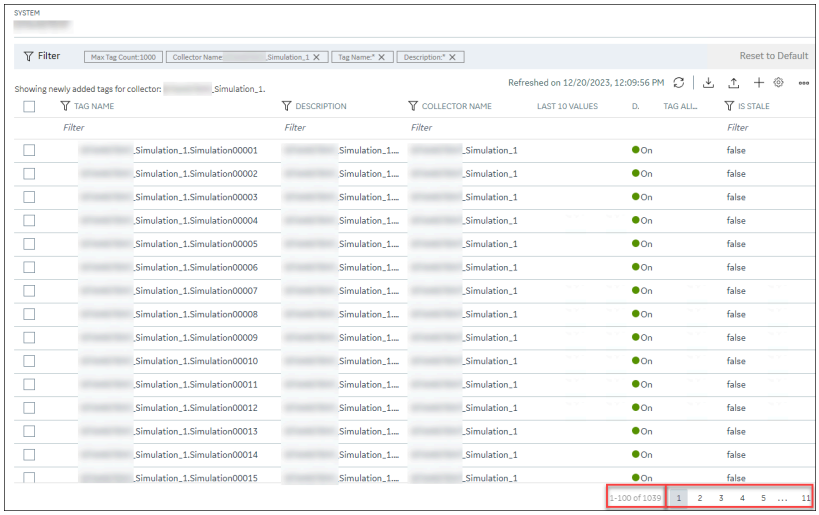
   Alternatively, you can also unregister a plugin from the Plugin **DETAILS** section by selecting **X** on the top-left corner in the **PLUG-IN** section.

   This will prompt you whether to delete the plugin. Selecting **Continue** will unregister the plugin.

# Common Tasks in Configuration Hub

| Task | Procedure |
|---|---|
| Show or hide the **Navigation** or the **Details** section. | 1. In the upper-right corner of the page, select .<br>2. Select the check boxes for the sections that you want to show.<br>  |
| Show or hide columns in a table.<br><br>**Note:**<br>You cannot hide some of the columns (for example, the **COLLECTOR NAME** column). | 1. In the upper-right corner of the table, select .<br>  <br>The **Table Settings** window appears.<br>2. Select the check boxes in the **SHOW COLUMN** column, and then select **Apply**. |

| Task | Procedure |
| --- | --- |
| Reorder columns in a table.<br><br>**Note:**<br>You cannot reorder some of the columns. | 1. In the upper-right corner of the table, select .<br>  <br>The **Table Settings** window appears.<br>2. Use the arrow buttons in the **RE-ORDER** column, and then select **Apply**. |
| Refresh a page/table. | In the upper-right corner of the main section or a table, select .<br>  |

| Task | Procedure |
| --- | --- |
| Select multiple tags or clear the selection. | • To select multiple tags, select the check boxes corresponding to the tags as needed.<br><br>Alternatively, to select all the available tags, select the check box in the upper-left corner of table header.<br><br>  |
| Navigate through grids in a page | When there are more than 100 rows in a grid, page numbers are enabled at the bottom-right corner. You can use these page numbers to navigate through the grid and access other available rows.<br><br>  <br><br>In addition to this, you can also see a grid's count (both total and selected). |

# Setting up a Stand-Alone System

## About Setting up a Stand-Alone Historian System

In a stand-alone Historian system, there is only one Historian server. This type of system is suitable for a small-scale Historian setup.

To set up a stand-alone Historian system, you must first set up Configuration Hub *(on page 305)*.

**Components of a Historian System**: In a Historian system, the following components are used. This list is not comprehensive. For a complete list, refer to System Components *(on page 73)*.

- **The Historian server:** You must install a single-server Historian, and apply the license *(on page 82)*.
- **A Historian system:** A Historian system is a network of Historian servers that collect, store, and retrieve data related to tags, alarms, and events.

  By default, a system is created when you set up Configuration Hub.
- **A data store:** A data store is a logical collection of tags used to store, organize, and manage tags according to your requirements. The primary use of data stores is segregating tags by data collection intervals. For example, you can put name plate or static tags (where the value rarely changes) in one data store, and put process tags in another data store. This can improve the query performance.

  By default, a user data store is created when you set up Configuration Hub. You can add more as needed.
- **A collector instance:** Collectors are the applications that collect data from a data source, and send it to an on-premises Historian server or a cloud destination such as Predix Time Series and Azure IoT hub.

  You must add a collector instance *(on page 357)* to begin collecting data. You can choose the type of the collector depending on your need. You can use any existing instances (created during collector installation or ported during an upgrade).
- **Tags:** Tags are the parameters for which you want to store data (for example, temperature, pressure, torque).

  You must specify the tags *(on page 357)* for which you want to collect data.
- **Data archiver:** This is a service that indexes all the data by tag name and timestamp, and stores the result in an .iha file.

  By default, this is installed when you install the Historian server.

- **Clients:** These are applications that retrieve data from the archive files using the Historian API.

  By default, these are installed when you set up Configuration Hub.

## Add a Collector Instance

Before you begin using a collector, you must add an instance of the collector. You can add multiple instances of the same collector or instances of multiple collectors. To add multiple instances of a collector, perform the steps once again.

You can add and configure the following types of collector instances:

- The Calculation collector *(on page 479)*
- The CygNet collector *(on page 482)*
- The File collector *(on page 486)*
- The HAB collector *(on page 489)*
- The iFIX collector *(on page 505)*
- The MQTT collector *(on page 510)*
- The MQTT Sparkplug B collector *(on page 519)*
- The ODBC collector *(on page 527)*
- The OPC Classic Alarms and Events collector *(on page 532)*
- The OPC Classic DA collector *(on page 534)*
- The OPC Classic HDA collector *(on page 540)*
- The OPC UA DA collector *(on page 544)*
- The OSI PI collector *(on page 549)*
- The OSI PI distributor *(on page 553)*
- The Server-to-Server collector *(on page 559)*
- The Server-to-Server distributor *(on page 563)*
- The Simulation collector *(on page 567)*
- The Windows Performance collector *(on page 571)*
- The Wonderware collector *(on page 575)*

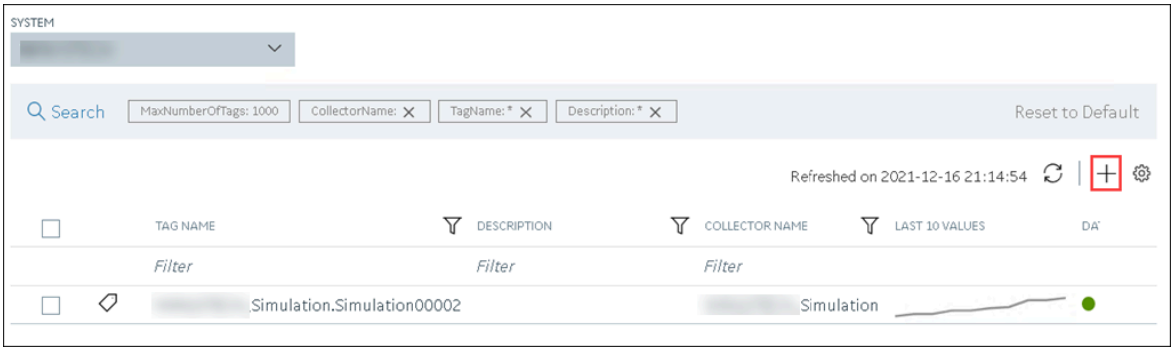
## Add Tags for the Data Store Using Configuration Hub

- Add the collector instance *(on page 357)* using which you want to collect data. Ensure that the collector is running.
- By default, the tag data is stored in the user data store, which is created automatically when you set up Configuration Hub. If, however, you want to store the data in a different data store, create it *(on page 370)*.

This topic describes how to specify the tags for which you want to collect data by browsing through the tags in the data source. For example, for an iFIX collector, if there are 1,00,000 tags in the iFIX server, you must specify the ones for which you want to collect data. Only then data is collected for those tags.

In addition to adding tags from the data source, you can create tags manually *(on page 473)*.

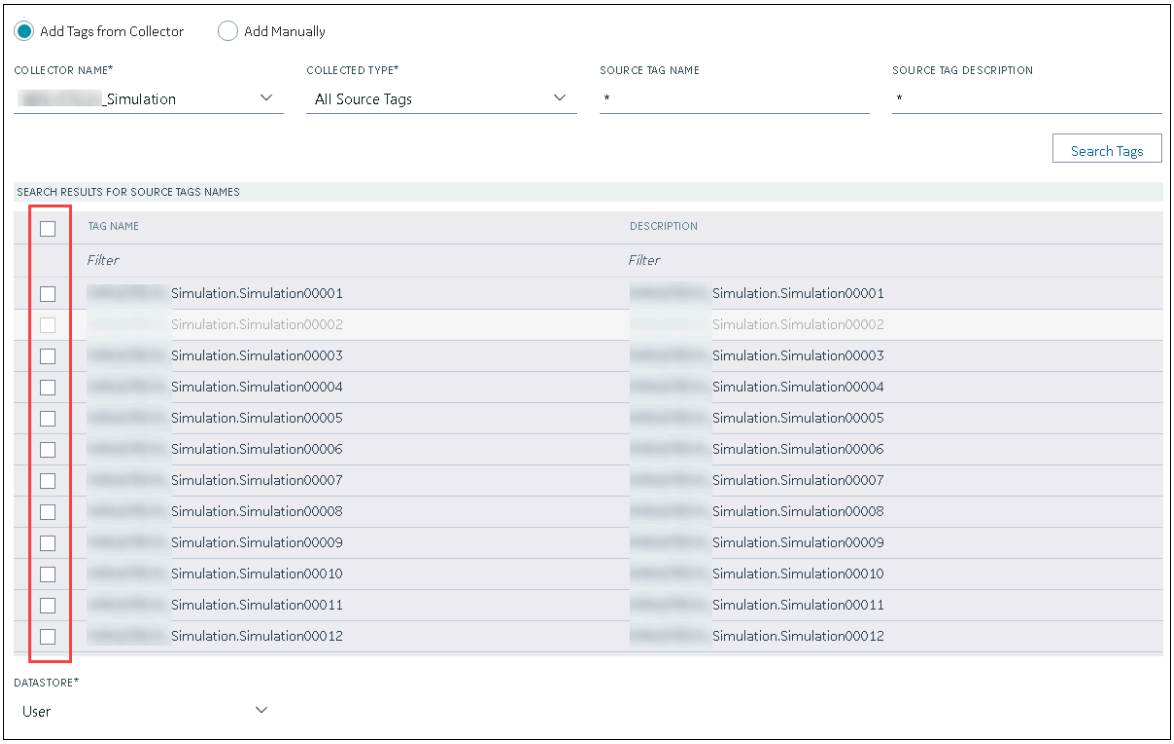
1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
3. Select +.




   The **Add Tag-<system name>** page appears. The **Add Tags from Collector** option is selected by default.
4. Enter values as described in the following table.

| Field | Description |
|---|---|
| **COLLECTOR NAME** | Select the collector instance that you want to use to collect data. A value is required. |
| **COLLECTED TYPE** | Specify whether you want to browse through all the tags in the data source or only from the tags that you have not added yet. A value is required. |
| **SOURCE TAG NAME** | Enter the name of the tag (either completely or partially) to narrow down the search results. |
| **SOURCE TAG DESCRIPTION** | Enter the description of the tag (either completely or partially) to narrow down the search results. |

5. Select **Search Tags**.
   A list of tags that match *all* the criteria that you have specified appears. If a tag is already added, it is disabled.
6. Select the check box corresponding to each tag for which you want to collect data.




7. In the **DATA STORE** field, if you want to store the data in a different data store than the user data store, select the same.
8. Select **Add Tag**.
   Data collection begins for the selected tags.

As needed, configure each tag by providing values for the tag properties. For information on the delta query modes, refer to Counter Delta Queries.

# Setting up a Horizontally Scalable System

## About Setting up a Horizontally Scalable System

In a horizontally scalable Historian system, there are multiple Historian servers, all of which are connected to one another. This type of system is used to scale out the system horizontally. For example, if you have 5,00,000 tags in your Historian system, you can distribute them among the various servers to improve performance.

To set up a horizontally scalable system, you must first set up Configuration Hub *(on page 305)*.

**Components of a Historian System**: In a Historian system, the following components are used. This list is not comprehensive. For a complete list, refer to System Components *(on page 73)*.

- **The Historian servers:** You must install the following types of Historian servers:
  - **Primary:** A primary server is the only server in a system where the Configuration Manager service runs. For the entire system, Configuration Manager manages the system configuration licensed by the user (that is, the number of tags, options, and so on). Each system can have only one primary server.

    You must apply the Enterprise license *(on page 82)* to the primary server.
  - **Distributed/Mirror:** These servers collect and store data. If added to a mirror group/location, you can achieve high availability *(on page 367)*.

    You must apply the Distributed license *(on page 82)* to the distributed/mirror servers.
- **A Historian system:** A Historian system is a network of Historian servers that collect, store, and retrieve data related to tags, alarms, and events.

  By default, a system is created when you set up Configuration Hub.
- **Data stores:** A data store is a logical collection of tags used to store, organize, and manage tags according to your requirements. The primary use of data stores is segregating tags by data collection intervals. For example, you can put name plate or static tags (where the value rarely changes) in one data store, and put process tags in another data store. This can improve the query performance.

  By default, a user data store is created when you set up Configuration Hub. You can add more as needed.
- **Locations:** These are virtual entities in which data stores are created. They are used for storage. The following types of locations are used in a horizontally scalable system:
  - **Distributed location:** This location is created automatically when you install a Historian mirror primary server, or when you install a Historian distributed/mirror node and add it to the primary server. You cannot modify or delete this location, and you cannot create another one.
  - **Mirror location:** This location is used to replicate data collected in a data store. For more information, refer to About Data Mirroring *(on page 367)*.
- **A collector instance:** Collectors are the applications that collect data from a data source, and send it to an on-premises Historian server or a cloud destination such as Predix Time Series and Azure IoT hub.

  You must add a collector instance *(on page 357)* to begin collecting data. You can choose the type of the collector depending on your need. You can use any existing instances (created during collector installation or ported during an upgrade).

- **Tags:** Tags are the parameters for which you want to store data (for example, temperature, pressure, torque).

  You must specify the tags *(on page 357)* for which you want to collect data.

- **Data archiver:** This is a service that indexes all the data by tag name and timestamp, and stores the result in an .iha file.

  By default, this is installed when you install the Historian server.

- **Clients:** These are applications that retrieve data from the archive files using the Historian API.

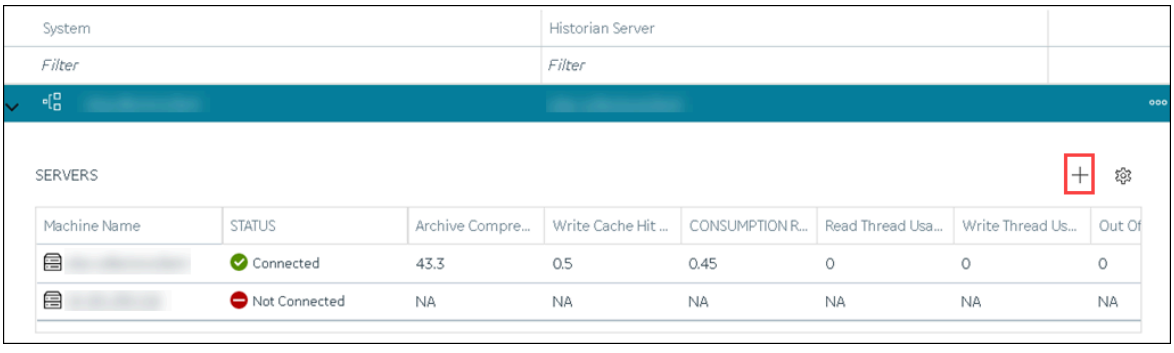
  By default, these are installed when you set up Configuration Hub.

## Add a Distributed/Mirror Server

1. Install Historian server *(on page 105)* on the machine that you want to add as a distributed server.
2. Add a system *(on page 427)*. The server that you specify while adding the system serves as the primary server for the system.

If you want to create a horizontally scalable Historian system, you must first add a primary server, and then add one or more distributed/mirror machines to scale out the primary server horizontally and thus, improve performance.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Systems**.
   A list of systems appears in the main section.
3. Expand the system in which you want to add a distributed/mirror server.
   A list of servers in the system appears.
4. Select +.




The **Add Server Machine: <system name>** window appears.

5. Enter the host name or IP address of the machine that you want to add, and then select **Add**. The distributed server is added to the system. A distributed location is added in the server. You cannot modify or delete this location.

- If you want high availability of one or more data stores in the server, create a mirror location *(on page 368)*, and then add the data stores *(on page 370)*. If not, add the data store *(on page 370)* to the distributed location.
- If you want to set a distributed node as backup to the primary node, set a distributed node as backup *(on page 365)*.

## Add a Collector Instance

Before you begin using a collector, you must add an instance of the collector. You can add multiple instances of the same collector or instances of multiple collectors. To add multiple instances of a collector, perform the steps once again.

You can add and configure the following types of collector instances:

- The Calculation collector *(on page 479)*
- The CygNet collector *(on page 482)*
- The File collector *(on page 486)*
- The HAB collector *(on page 489)*
- The iFIX collector *(on page 505)*
- The MQTT collector *(on page 510)*
- The MQTT Sparkplug B collector *(on page 519)*
- The ODBC collector *(on page 527)*
- The OPC Classic Alarms and Events collector *(on page 532)*
- The OPC Classic DA collector *(on page 534)*
- The OPC Classic HDA collector *(on page 540)*
- The OPC UA DA collector *(on page 544)*
- The OSI PI collector *(on page 549)*
- The OSI PI distributor *(on page 553)*
- The Server-to-Server collector *(on page 559)*
- The Server-to-Server distributor *(on page 563)*
- The Simulation collector *(on page 567)*
- The Windows Performance collector *(on page 571)*
- The Wonderware collector *(on page 575)*

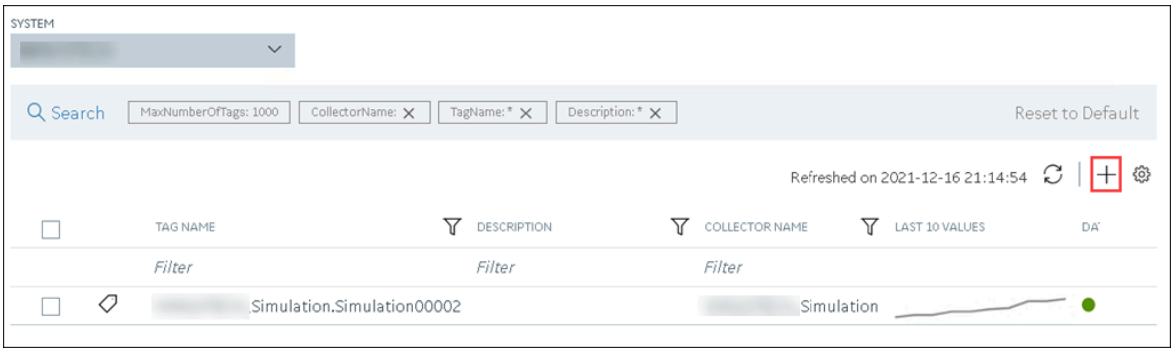
# Add Tags for the Data Store Using Configuration Hub

- Add the collector instance *(on page 357)* using which you want to collect data. Ensure that the collector is running.
- By default, the tag data is stored in the user data store, which is created automatically when you set up Configuration Hub. If, however, you want to store the data in a different data store, create it *(on page 370)*.

This topic describes how to specify the tags for which you want to collect data by browsing through the tags in the data source. For example, for an iFIX collector, if there are 1,00,000 tags in the iFIX server, you must specify the ones for which you want to collect data. Only then data is collected for those tags.

In addition to adding tags from the data source, you can create tags manually *(on page 473)*.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
3. Select +.

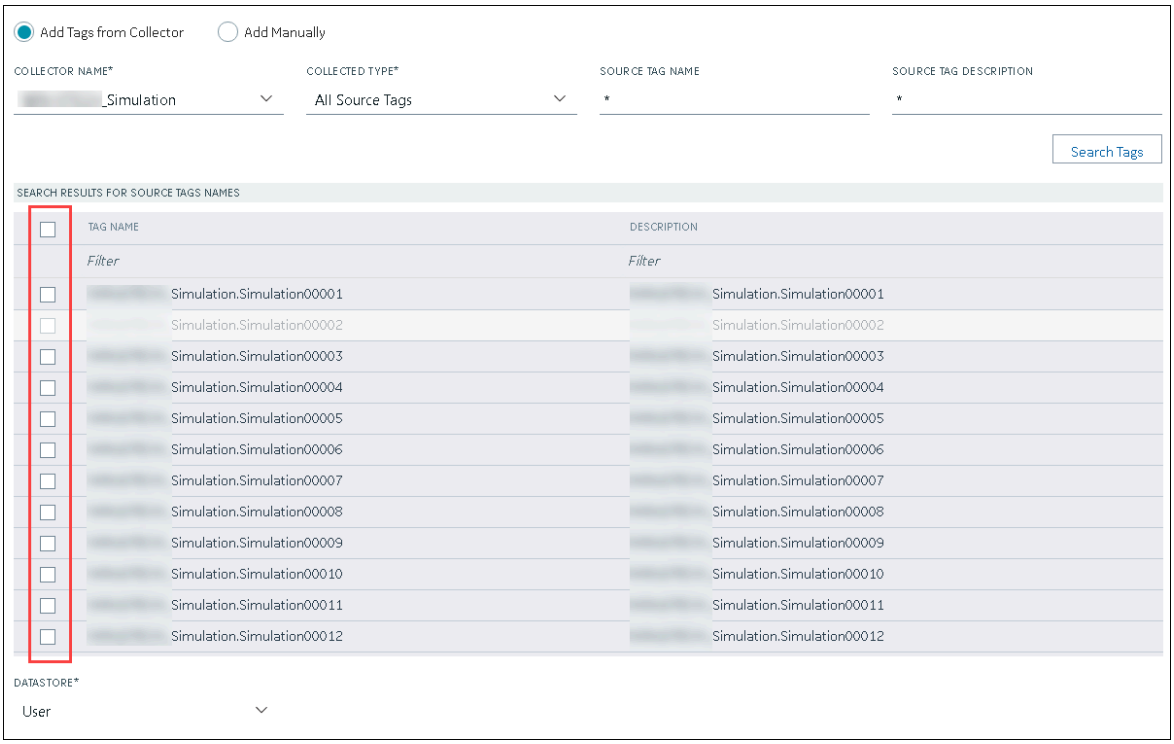



   The **Add Tag-<system name>** page appears. The **Add Tags from Collector** option is selected by default.
4. Enter values as described in the following table.

| Field | Description |
|---|---|
| **COLLECTOR NAME** | Select the collector instance that you want to use to collect data. A value is required. |
| **COLLECTED TYPE** | Specify whether you want to browse through all the tags in the data source or only from the tags that you have not added yet. A value is required. |

| Field | Description |
|---|---|
| **SOURCE TAG NAME** | Enter the name of the tag (either completely or partially) to narrow down the search results. |
| **SOURCE TAG DESCRIPTION** | Enter the description of the tag (either completely or partially) to narrow down the search results. |

5. Select **Search Tags**.
   A list of tags that match *all* the criteria that you have specified appears. If a tag is already added, it is disabled.
6. Select the check box corresponding to each tag for which you want to collect data.



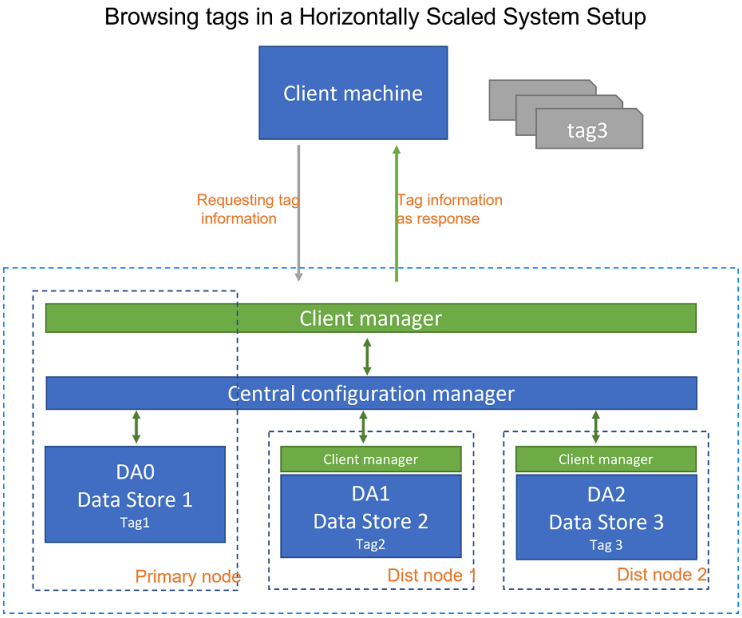

7. In the **DATA STORE** field, if you want to store the data in a different data store than the user data store, select the same.
8. Select **Add Tag**.
   Data collection begins for the selected tags.

As needed, configure each tag by providing values for the tag properties. For information on the delta query modes, refer to Counter Delta Queries.

# Browse Tags using Distributed or Mirror Node Servers when Primary Server is Inactive

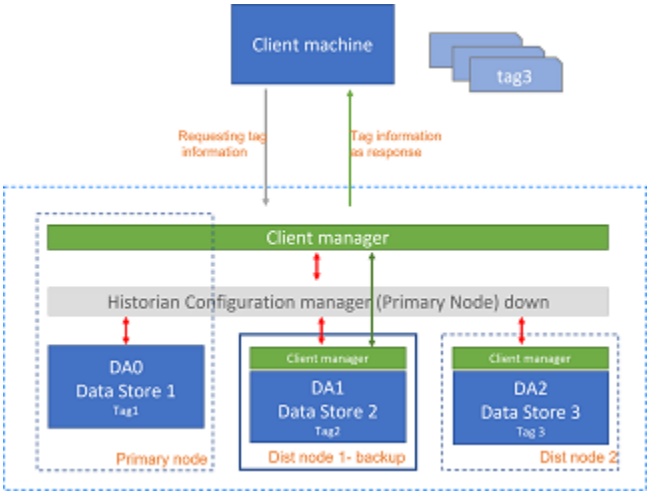
In a horizontally scaled system, there can be multiple Historian servers that are connected to one another. That is, if there are 100000 tags in your system, you can distribute them among the various servers. Similarly, you can view the tags and their values in all the servers.

For example, consider the following setup:



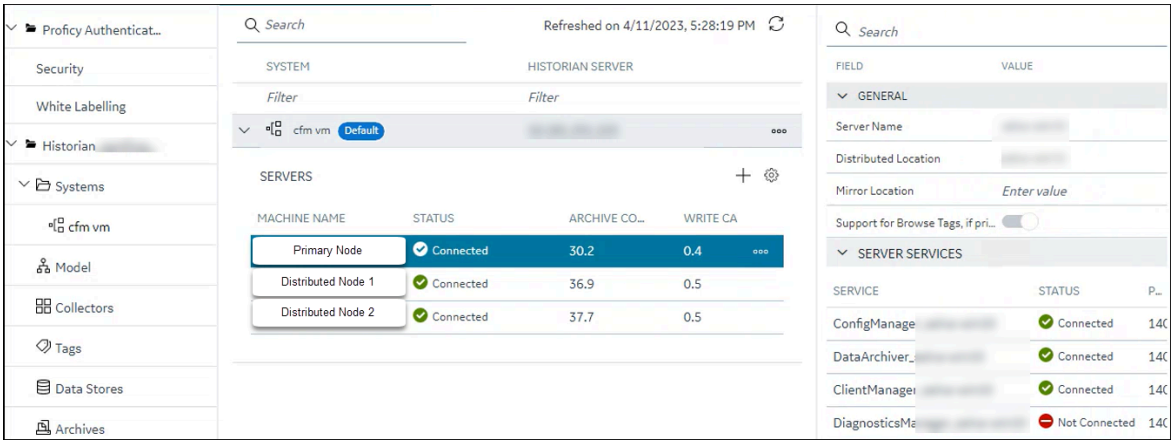

In general, the Historian configuration manager of the primary node stores the tag information of all the data stores in the system. Whenever a client machine requests tag information, the Historian configuration manager sends the corresponding information for all nodes (primary, distributed 1, and distributed 2) to the client machine through the client manager. Suppose the central configuration manager is inactive, in that case, the client manager of the primary node sends the tag information corresponding to the primary node (data store 1) to the client machine. Consequently, the tag information of the other nodes will not be sent to the client machine.

To overcome this, you can assign the other nodes as backups for the primary node's central configuration manager. Therefore, whenever the central configuration manager is inactive, the Data Archiver (DA) of the backup node functions as a configuration manager and sends all tag information to the client manager. The client manager will then forward the tag information response to the client machine. With this, you can still browse tags and their details when the primary node's configuration manager is inactive.

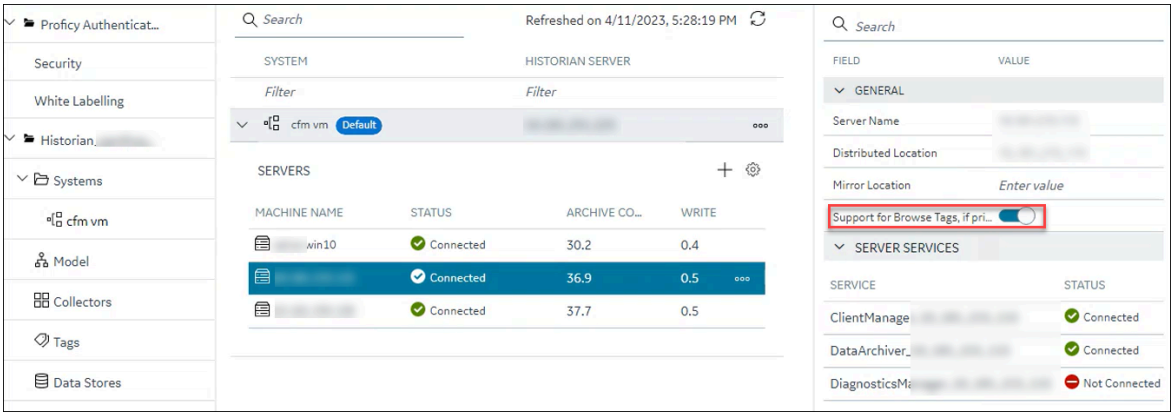



This topic describes how to set a distributed node as a backup node for the primary node.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, expand **Systems**, and then select the system that you want to access.
   The selected system and its details appear.
3. Expand the system to see the corresponding servers.




4. Select a distributed node that you want to set as the backup.
   The selected distributed node's details appear in the **DETAILS** section.




5. In the **GENERAL** section, enable **Support for Browse Tags, if primary server down**.
6. In the upper-left corner, select **save**.
   The selected distributed node is set as a backup for the primary node's configuration manager.

If your primary node's configuration manager is inactive, you can still browse the tags and their details corresponding to all the nodes within the horizontally scaled system.

# Setting up High Availability

## About Data Mirroring

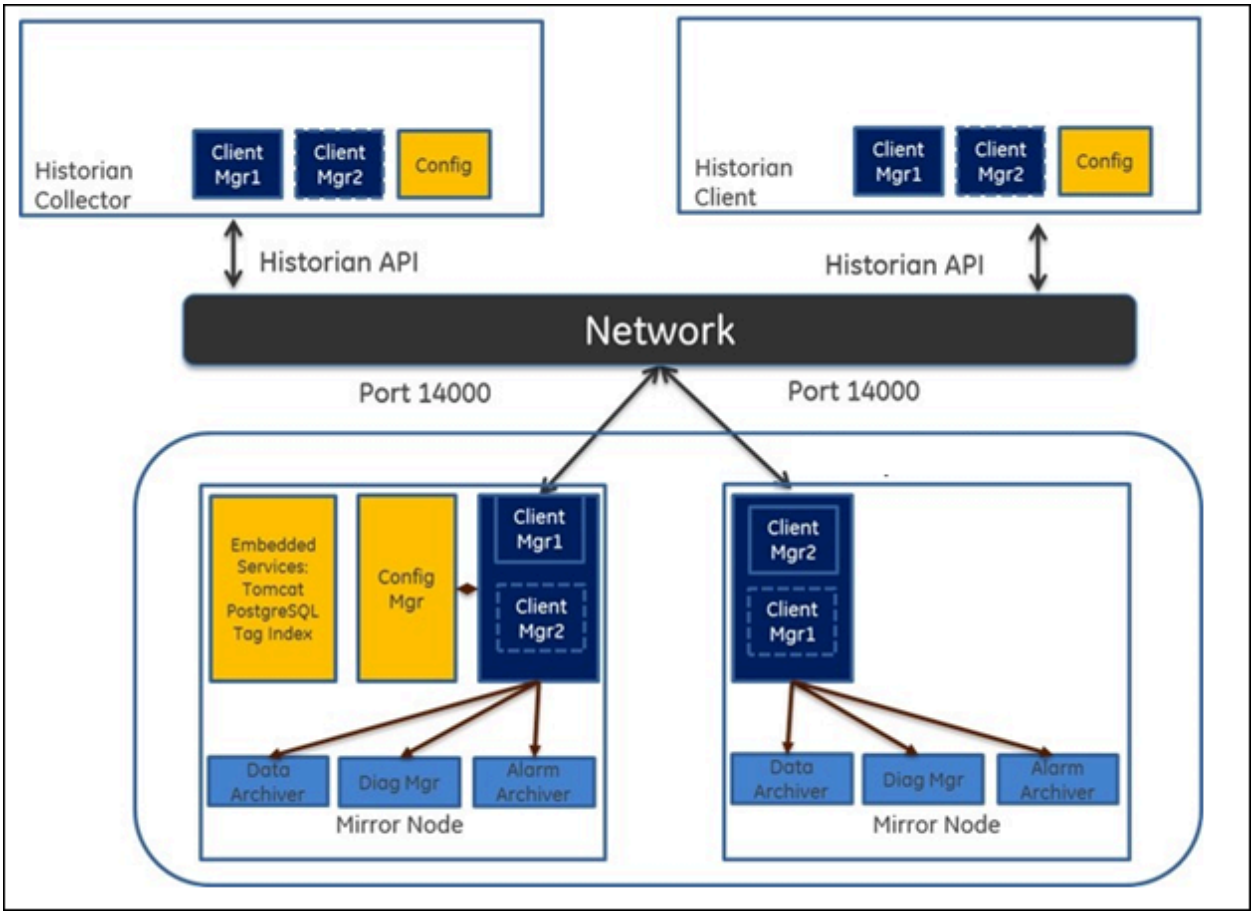
Historian provides mirroring of stored data on multiple nodes to provide high levels of data reliability. Data Mirroring also involves the simultaneous action of every insert, update and delete operations that occurs on any node. Data mirroring provides continuous data read and write functionality.

In a typical data mirroring scenario, one server acts as a primary server to which the clients connect. All communication goes through the Client Manager, and each Client Manager knows about the others. Mirrors must be set up in a single domain.

When you create a mirror location, you add one or more servers to the group, and then create the data stores whose data you want to replicate. For example, suppose you want to create a data store for collecting the data for 100 tags, for which you want high availability. In that case, you must create a mirror location, add two or more servers to the mirror location, and then create the data store. When you do so, the data retrieved in the data store is stored in all the servers in the mirror location. If one of the servers is down, you can retrieve the data from the other servers in the group.

### Mirror Node Setup

The following diagram helps you to understand a typical single mirror node setup.




## Create a Mirror Location

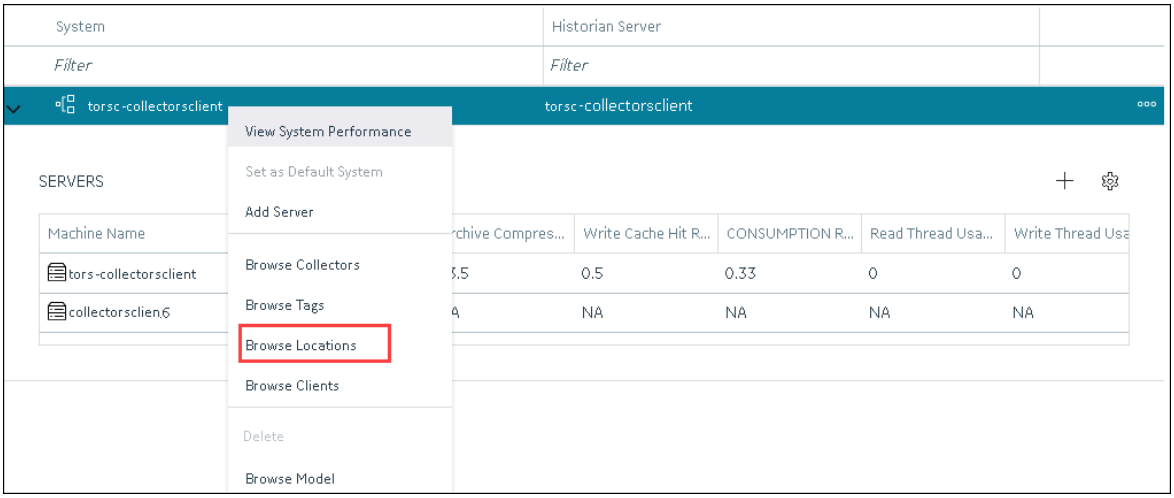
Add one or more distributed servers *(on page 361)* to the system in which you want to create a mirror group.

If you want high availability of one or more data stores, you must create a mirror group (also called a mirror location), and then add servers to it. When you do so, the data in the data stores of the mirror locations is replicated. Therefore, even if one of the servers is down, you can retrieve data from the other servers in the mirror location, thus achieving high availability.

The following conditions apply when you create a mirror location:

- You must add minimum two servers to a mirror location. The maximum number of servers that you can add depends on your Historian license.
- You can add a mirror location only in a horizontally scalable Historian system.
- You can rename a mirror location, remove a machine from a mirror location, or add an additional one even after you create the mirror location. However, if only one machine remains in the group, you cannot remove it.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Systems**.
   The **Systems** section appears, displaying a list of systems.
3. Right-click the system in which you want to create a mirror location (or select ), and then select **Browse Locations**.




   A list of distributed locations in the system appears.
4. Select **Mirror Locations**.
   A list of mirror locations in the system appears.
5. In the upper-right corner of the main section, select .
   The **Add Mirror Location** window appears.
6. Provide values as described in the following table.

| Field | Description |
|---|---|
| **MIRROR LOCATION NAME** | Enter a name for the mirror location. A value is required and must be unique for the system. |
| **SERVER MACHINES** | Select the servers that you want to add to the mirror group. This box contains a list of all the servers in the system. You must add minimum two servers to a mirror location. |

7. Select **Add**.
   The mirror location is created.

Add a data store to the mirror location *(on page 370)*.

## Create a Data Store

The number of data stores that you can create depends on your license.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Data Sores**.
   Alternatively, you can select **Systems**, right-click the system in which you want to create a data store (or select ), and then select **Browse Data Stores**.
   The **Data Stores** section appears.
3. Select .
   If Historian Standard version, then the **Add Data Store** window appears.

   If Historian Enterprise version, then the **Add Data Store: <location name>** window appears.
4. Enter values as described in the following table.

| Field | Description |
|---|---|
| **DATA STORE NAME** | Enter a unique name for the data store. A value is required. You can use all alphanumeric characters and special characters except / \ * ? < > \| |
| **DESCRIPTION** | Enter a description for the data store. |
| **LOCATION** | Enter the host name or IP address of the distributed location on which you want to create the data store. This field is available only for a horizontally scalable system. |
| **Is Default** | Switch the toggle on if you want to set this data store as the default one. A default data store is the one that is considered if you do not specify a data store while adding a tag. You can set only one data store as default. |

5. Select .
   The data store is created.

   When you add tags to the data store, it will have its own set of .IHA (iHistorian Archive) files. Ensure that you back up the new data store archives periodically.

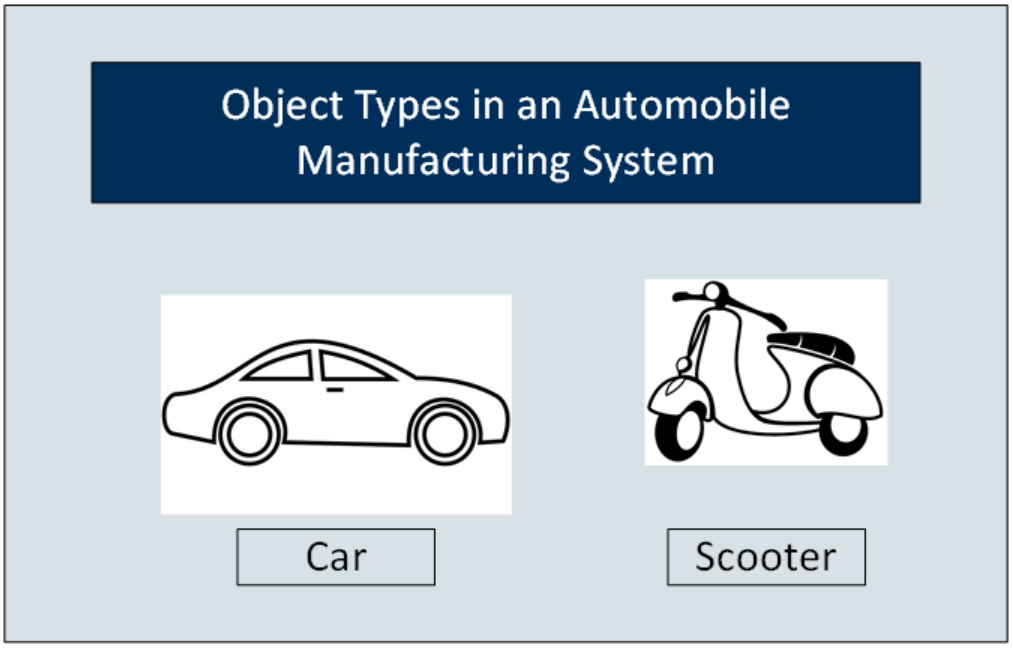
# Creating a Model

## About a Historian Model

A Historian model is a hierarchical classification of various objects in a system. A model contains the following components:

- Object Types
- Contained Types
- Object Instances
- Variables

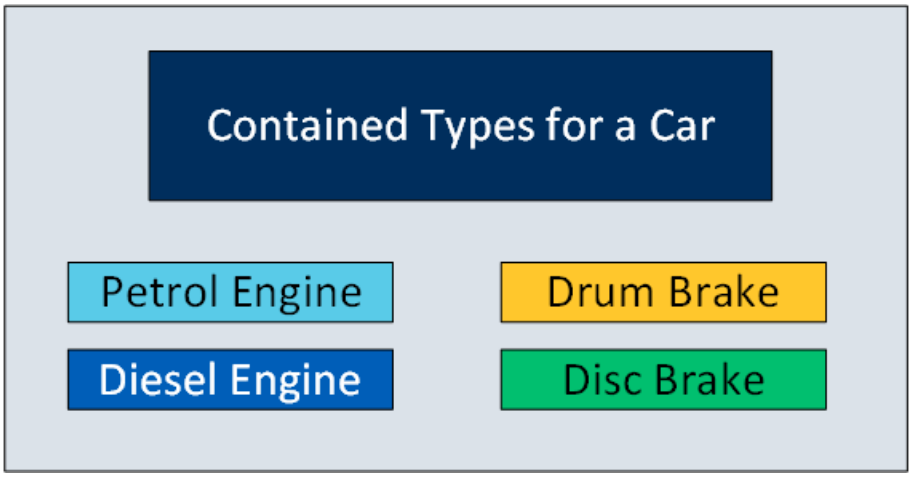
**Object Type:** An object type is a blueprint, which you want to replicate that will have a common structure (common properties/attributes and contained types). These object types can be the products you manufacture, your assets, byproducts, or anything else for which you want to classify information hierarchically and inherit properties/attributes. For example, in an automobile manufacturing unit, the vehicles you manufacture are object types (for example, a car or a scooter).



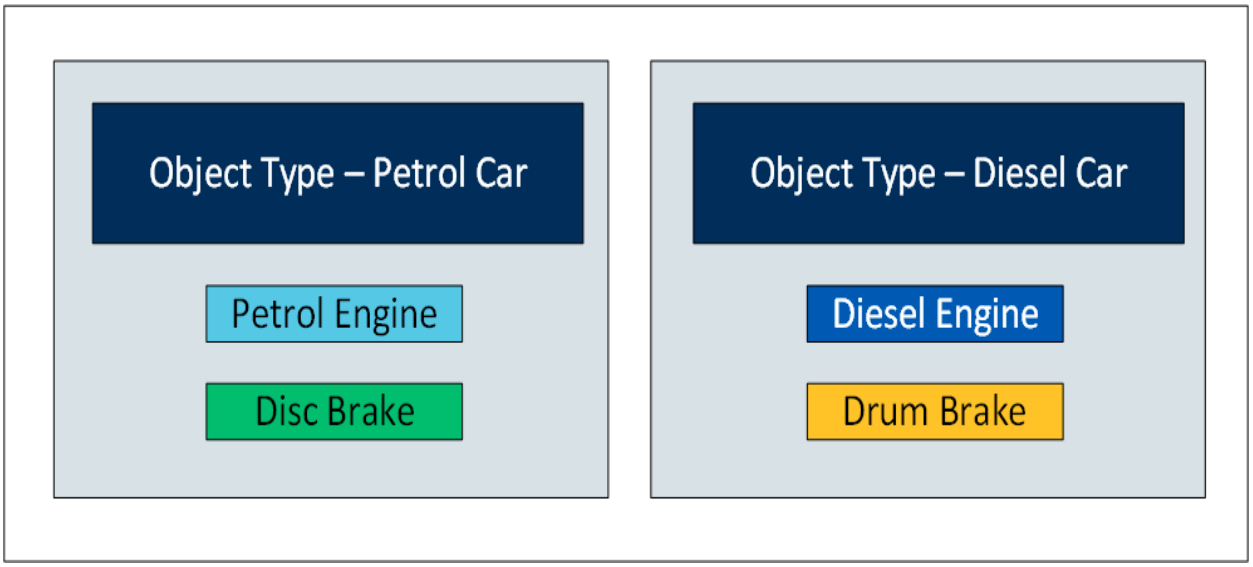

**Contained Types:** A contained type is an object type that you can include in another object type. For example, suppose you manufacture cars with the following types of engines:

- Petrol
- Diesel

You can then create one contained type for a petrol engine and one for a diesel engine. Similarly, you can create contained types for various types of brake systems, testing parameters, and so on.

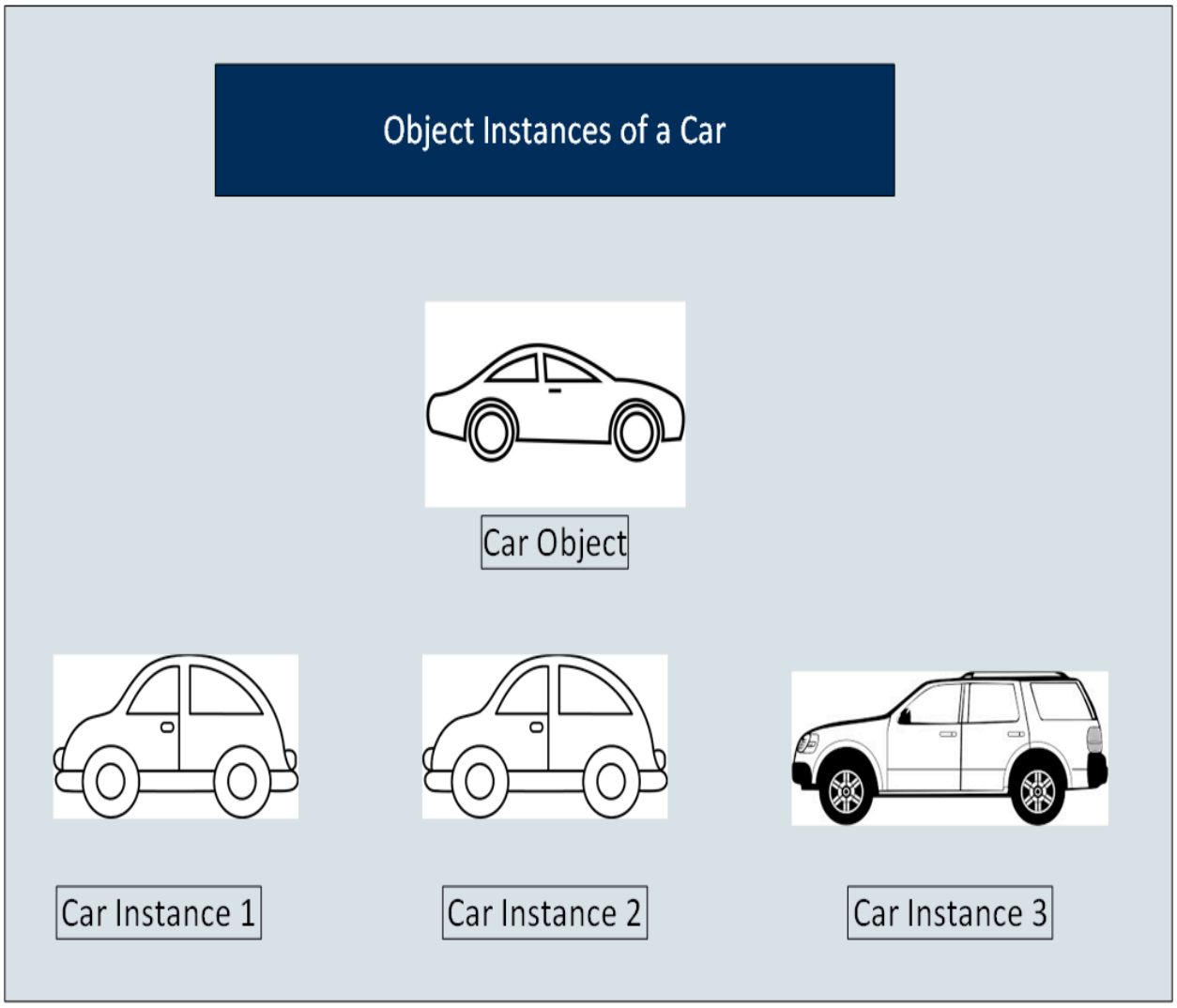



When you create an object type for a car, you can include any of these contained types.



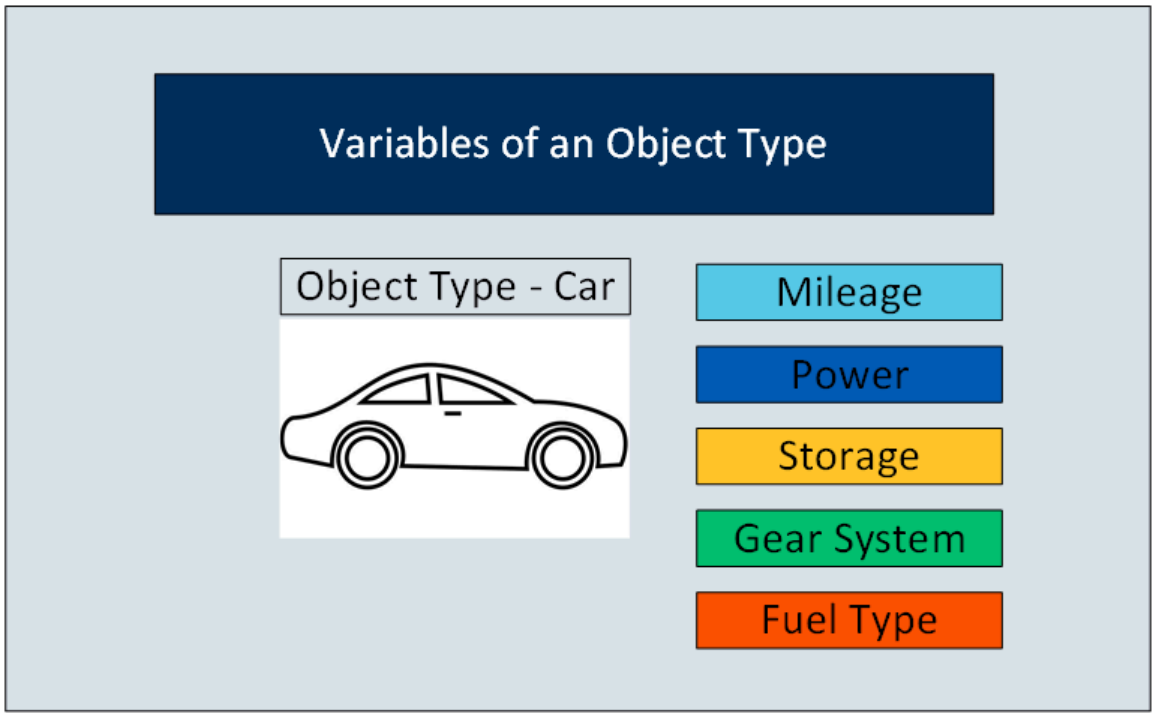

You can include multiple contained types in a single object instance. In addition, you can include a single contained type in multiple object instances.

**Object Instances:** Each item of an object type that you manufacture is called an object instance. For example, if you manufacture three cars, each one is an instance.



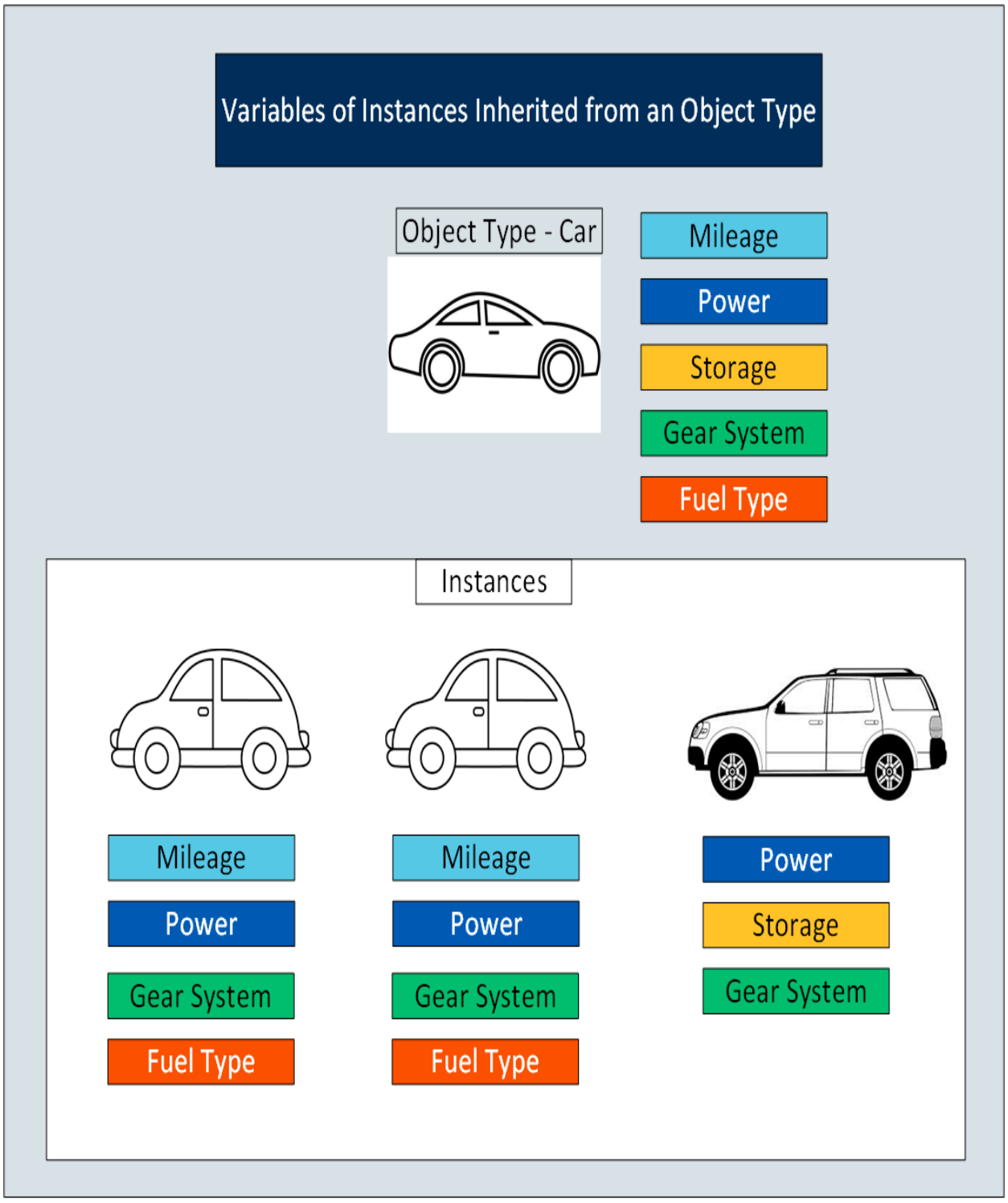

An object instance is specific to a Historian system. An object type, however, is not associated with a system.

**Variables:** Each attribute or property of an object is called a variable. These variables are common across all objects of a certain type. They represent tags whose values are collected by data collectors. For example, a car can have the following variables.




When you create instances of an object type, by default, the variables in the object type are inherited to all the instance as well. You can choose to include or exclude one or more of these variables for each instance.

In the following image, all the variables of the car object type are inherited to each of the instances. However, the first two instances do not include Storage. And the third instance does not include Mileage and Fuel Type.




If an object type contains contained types, the variables in the contained types are inherited as well.

After you create an object instance, you must store the values of each variable of the instance. To do so, you must map each variable with a Historian tag or create one, depending on the type of the variable.

**Types of Variables:**

- **Direct:** Tags for these variables are created in Historian when you select a collector instance. For instructions on collecting data for these types of tags, refer to Collect Data for a Direct Variable *(on page 392)*.
- **Indirect:** These variables are mapped with existing Historian tags. For instructions on collecting data for these types of tags, refer to Collect Data for an Indirect Variable *(on page 395)*.
- **Static:** These variables have a static value, which you provide when you create an object instance. For instructions on providing data for these types of tags, refer to Provide Data for a Static Variable *(on page 390)*.

**Limitations:**

- An OPC UA model is not supported.
- If the name of a tag associated with a variable in a model contains a period (.), you cannot import the tag while importing the model into a Historian system.

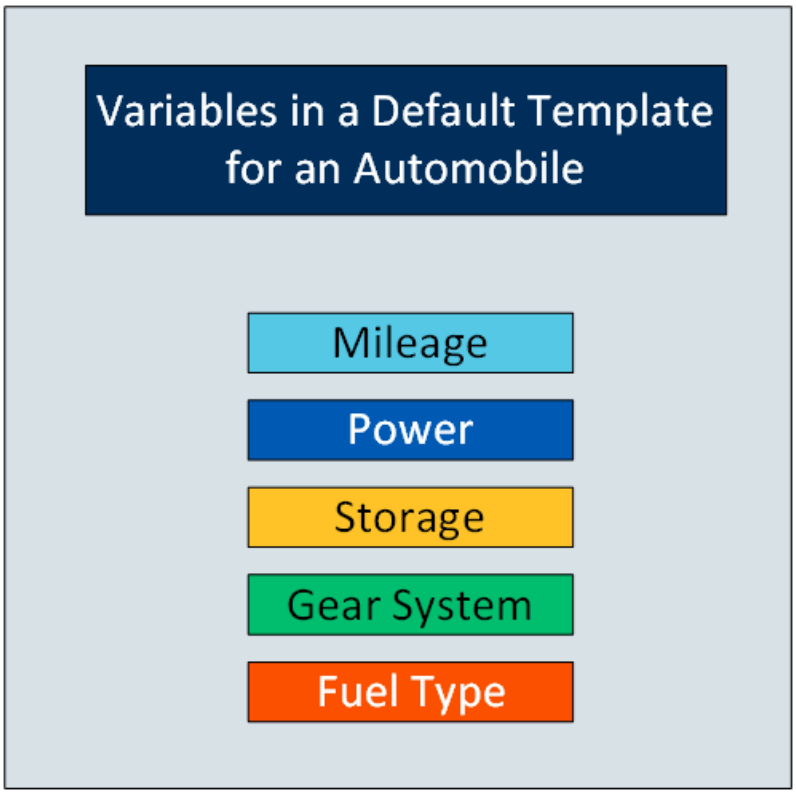
## About Object Templates

When you create an object type, you create a template. A template contains attributes/properties of an object type, called variables. When you apply a template to an object instance, the variables included in the template are added to the object instance.

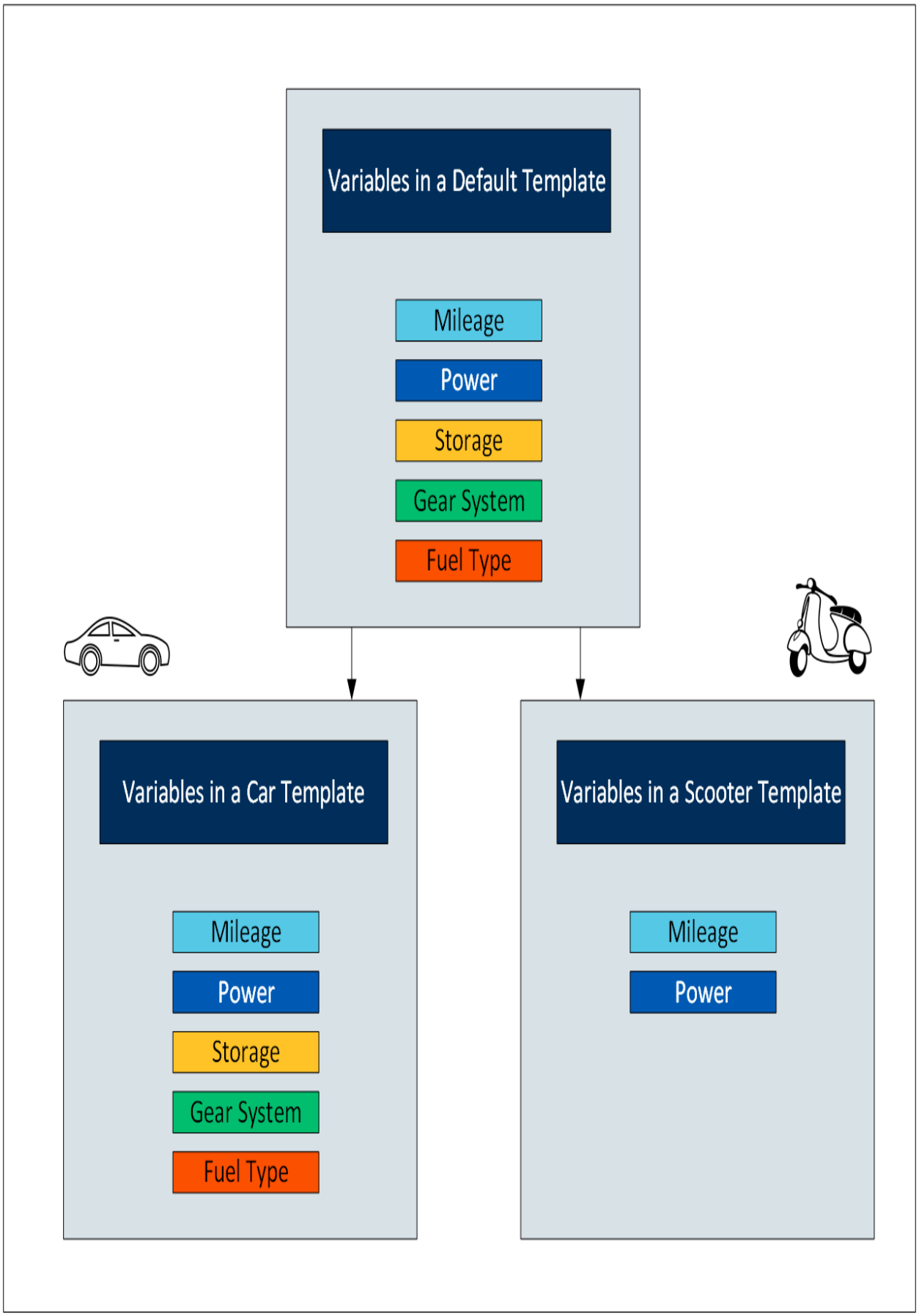
**Types of Templates:**

- **Default Template:** The default template contains generic/common variables of an object type. Each object type contains one and only one default template; you cannot delete it.
- **Custom Template:** In addition to the default template, if needed, you can create one or more custom templates for an object type. When you do so, you can choose to include variables from the default template or other custom templates in the same object type.

For example, suppose you have an automobile manufacturing unit. You can create a default template that contains generic details about an automobile. Each of these details is a variable.




In addition to the default template, you can create custom templates for the object type, and add variables to it. For example, in the automobile object type, you can create a template for a car and another one for a scooter. Some of the variables in the default template (such as storage and gear system) may not be applicable to a scooter. Therefore, you can exclude them.


Variables in a Default Template
Mileage
Power
Storage
Gear System
Fuel Type
Variables in a Car Template
Mileage
Power
Storage
Gear System
Fuel Type
Variables in a Scooter Template
Mileage
Power

When you later create instances of the object type, you can choose any of the three templates. When you do so, all the variables in the template are included in the object instance. You will then capture values of these variables in Historian.

## Workflow for Creating a Historian Model

To create a model and store tag values of object instances, you must perform the following steps.

| Step Number | Description | Notes |
|---|---|---|
| 1 | Set up Configuration Hub *(on page 305)*. | This step is required. It involves installing the required components to get started with Configuration Hub. |
| 2 | Create a collector instance *(on page 357)*. | This step is required. It involves creating a collector instance using which you want to collect data from the object instance and store it in an on-premises Historian server or a cloud destination. |
| 3 | Specify the tags for data collection *(on page 357)*. | This step is required. It involves specifying the tags for which you want to collect data by browsing through the tags in the data source. For example, for an iFIX collector, if there are 1,00,000 tags in the iFIX server, you must specify the ones for which you want to collect data. Only then data is collected for those tags.<br><br>You will later map these tags with the variables in each object instance, thus collecting data for the variables (explained later in this workflow). |
| 4 | Create an object type *(on page 380)*. | This step is required. It involves creating an object type, a default template, one or more custom templates as needed, and adding variables to each object type. |
| 5 | Create an object instance *(on page 388)*. | This step is required. It involves creating an object instance and applying the required |

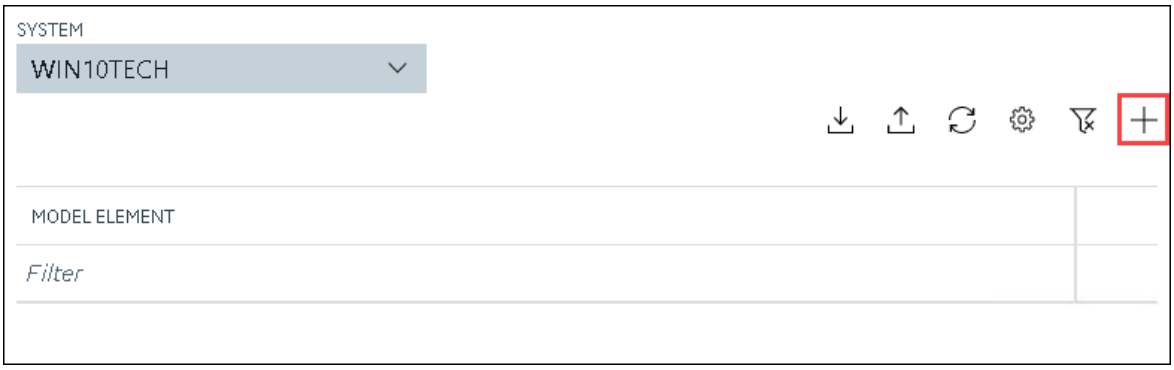
| Step Number | Description | Notes |
|---|---|---|
| | | template from the object type. The object instance then inherits the variables from the object type. |
| 6 | Provide/collect data for static *(on page 390)*, direct *(on page 392)*, and indirect *(on page 395)* variables. | This step is required. You can provide data for the following types of variables:<br>• **Static:** For a static variable, the value does not change; therefore, you just provide the value of the variable.<br>• **Direct:** For a direct variable, you associate the variable with a collector instance and a tag. When you do so, the tag is created in Historian, and values for the variable are collected by the collector instance and stored in Historian.<br>• **Indirect:** For an indirect variable, you associate the variable with a collector instance and an *existing* Historian tag. The values for the variable are then collected by the collector instance and stored in Historian.<br><br>For more information on the types of variables, refer to About a Historian Model *(on page 371)*. |

## Create an Object Type

When you create an object type, you also create the default template, custom templates, and variables for each template. For information on each of these template types and variables, refer to About Object Templates *(on page 376)* and About a Historian Model *(on page 371)*.

This topic describes how to create an object type. You can also copy one *(on page 403)*.

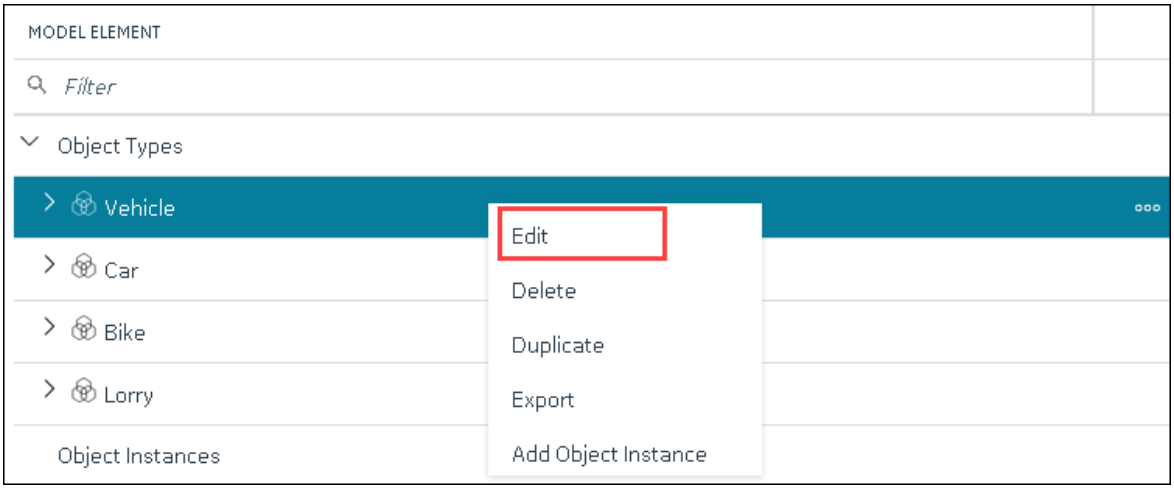
1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Model**.
   Alternatively, you can select **Systems**, and then in the row containing the system in which you want to create a model, select ⋯, and then select **Browse Model**.
   The **Model** section appears.
3. In the upper-right corner of the section, select +.




   The **New Object Type** window appears.
4. Enter values as described in the following table.

| Field | Description |
|---|---|
| **NAME** | Enter a name for the object type. A value is required and must be unique.<br><br>The value that you enter:<br>◦ Must begin with a letter or a number.<br>◦ Can contain up to 256 characters.<br>◦ Can include any of the following special characters: /!\|#{}%$-_<br>◦ Must not include a space or any of the following characters: ~`+^:;.,?"*=@ |
| **DESCRIPTION** | Enter a description for the object type. |

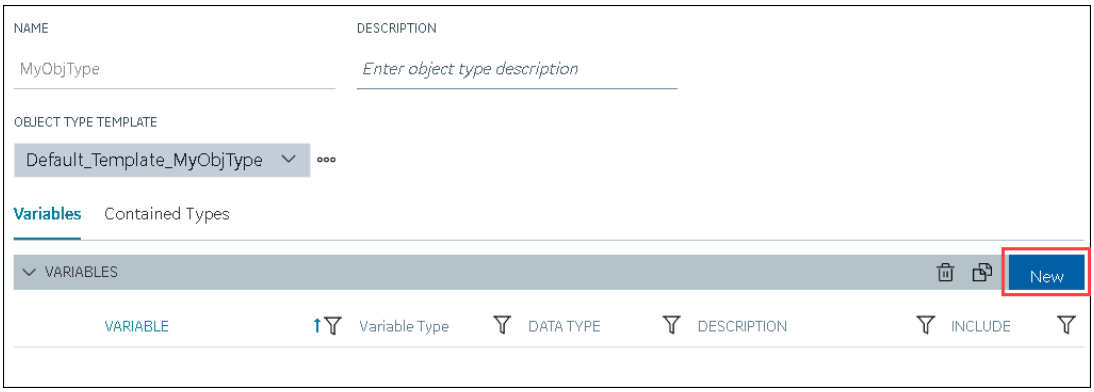
5. Select **Create**.
   The object type is created.
6. In the main section, under **Object Types**, right-click the object type that you have created (or select ⋯), and then select **Edit**.




The **<object type name>** section appears. The **OBJECT TYPE TEMPLATE** field contains the default template.

7. To add variables to the default template:

   a. In the **Variables** table, select **New**.




   A blank row appears in the table.

   b. Enter values as described in the following table.

| Column | Description |
|---|---|
| **VARIABLES** | Enter the name of the variable. A value is required and must be unique for the object type.<br><br>The value that you enter:<br>▪ Must begin with a letter or a number.<br>▪ Can contain up to 256 characters. |

| Column | Description |
|---|---|
| | ▪ Can include any of the following special characters: /!\|#{}%$-_<br>▪ Must not include a space or any of the following characters: ~` +^:;.,?"*={}@ |
| **VARIABLE TYPE** | Choose one of the following types of variables:<br>▪ **Direct:** Tags for these variables are created in Historian for a selected collector instance.<br>▪ **Indirect:** These variables are mapped with existing Historian tags.<br>▪ **Static:** These variables have a static value, which you provide when you create an object instance. |
| **DATATYPE** | Select the data type of the variable. |
| **DESCRIPTION** | Enter a description for the variable. |
| **INCLUDE** | Switch the toggle to indicate whether you want to include the variable in the template. |

**Note:**
After you apply a template to an object instance, you cannot modify or delete a variable in the object type; you can only add more variables. You can, however, copy the object type *(on page 403)*, and modify or delete variables in the copied one.

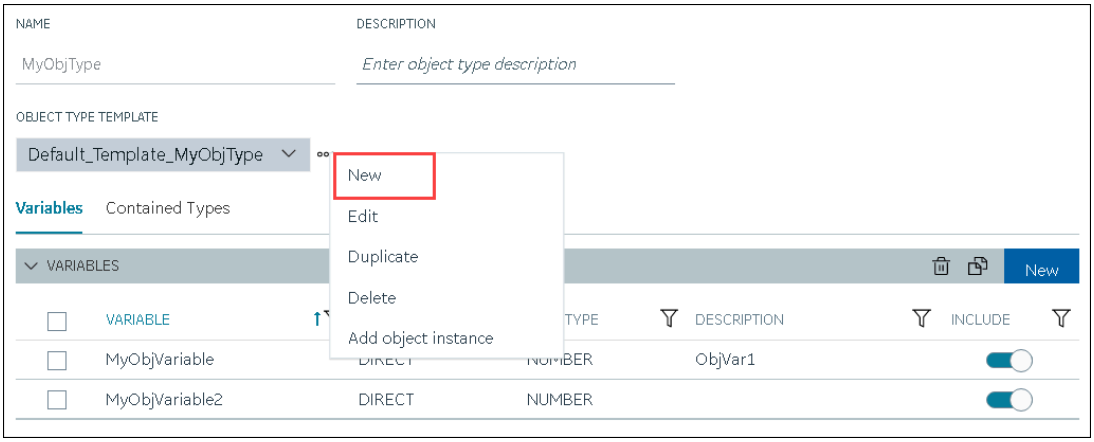
c. Press ENTER.

The default template is created, along with the variable that you have added. You can add more variables or include/exclude variables later too.

**Note:**
If you want to create a variable by copying an existing one, select the check box next to the variable that you want to copy, and then select . You can copy only one variable at a time.

8. To create a custom template:

   a. Next to the **TEMPLATE** field, select ⋯, and then select **New**.




   The **New Object Template** window appears.

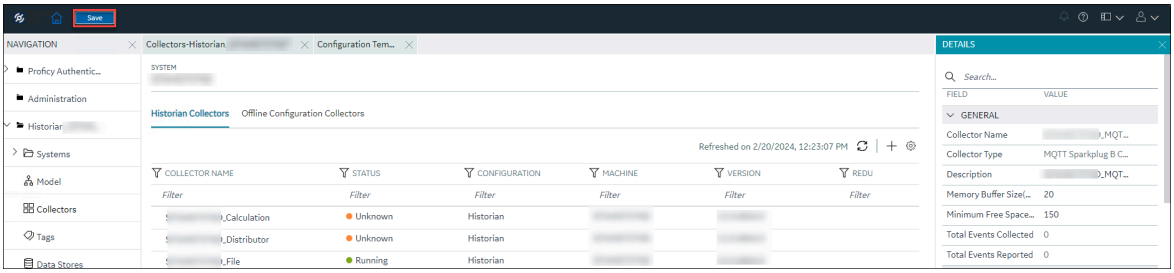
   b. Enter values as described in the following table.

| Field | Description |
|---|---|
| **NAME** | Enter a name for the template. A value is required and must be unique.<br><br>The value that you enter:<br>▪ Must begin with a letter or a number.<br>▪ Can contain up to 256 characters.<br>▪ Can include any of the following special characters: /!\|#{}%$-_<br>▪ Must not include a space or any of the following characters: ~` +^:;.,?"*={}@ |
| **DESCRIPTION** | Enter a description for the template. |

   c. Repeat step 7 to add variables to the custom template.

   The custom template is created, along with the variables. You can add more variables, and include/exclude existing variables later too.

9. In the upper-left corner of the page, select **Save**.




The object type, along with the default template, custom templates, and variables, is created.

Create an object instance *(on page 388)*.

## Include a Contained Type in an Object Type

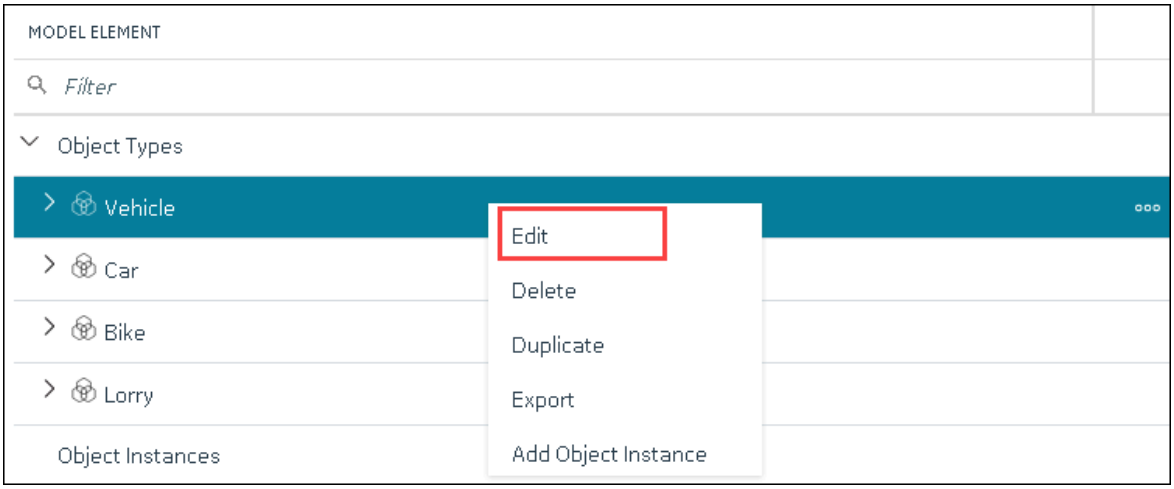
Create an object type *(on page 380)* that you want to use as a contained type.

A contained type is an object type that you can include in another object type. When you do so, you can reuse the variables in the contained type without creating them again manually. These variables are inherited by object instances of the object type in which you include the contained type.

You can include a single contained type in multiple object types and multiple contained types in a single object type.

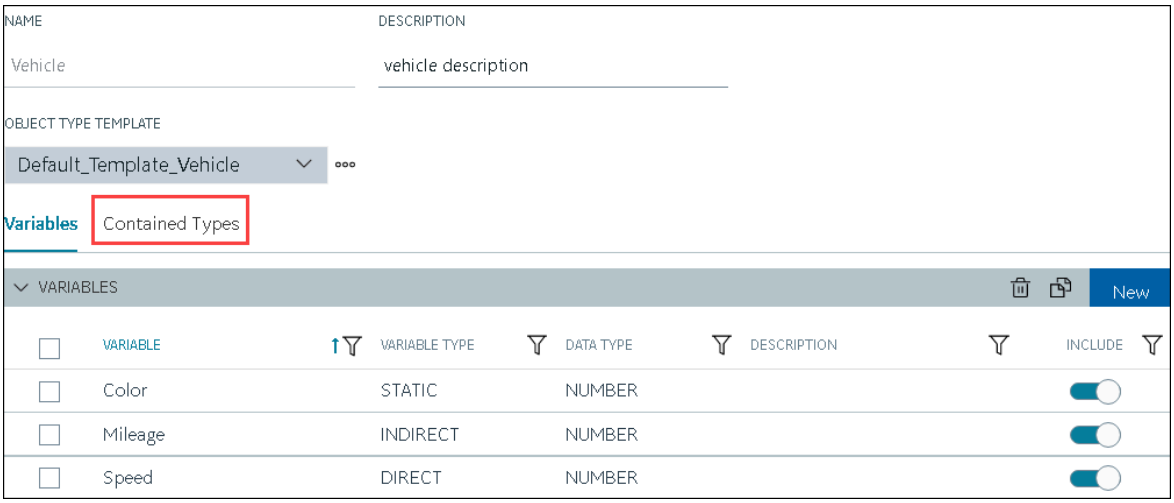
For more information, refer to About a Historian Model *(on page 371)*.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Model**.
   Alternatively, you can select **Systems**, and then in the row containing the system in which you want to create a contained type, select ⋯, and then select **Browse Model**.
   The **Model** section appears.
3. Under **Object Types**, right-click the object type in which you want to include the contained type (or select ⋯), and then select **Edit**.



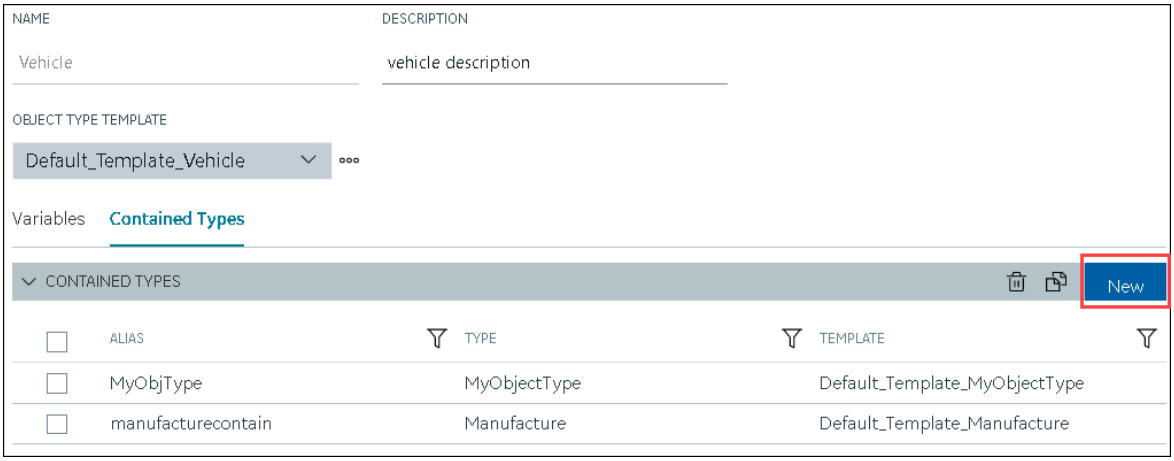

The **<object type name>** section appears, displaying a list of variables in the object type.

4. Select **Contained Types**.




A list of contained types in the object type appears.

5. In the **CONTAINED TYPES** table, select **New**.



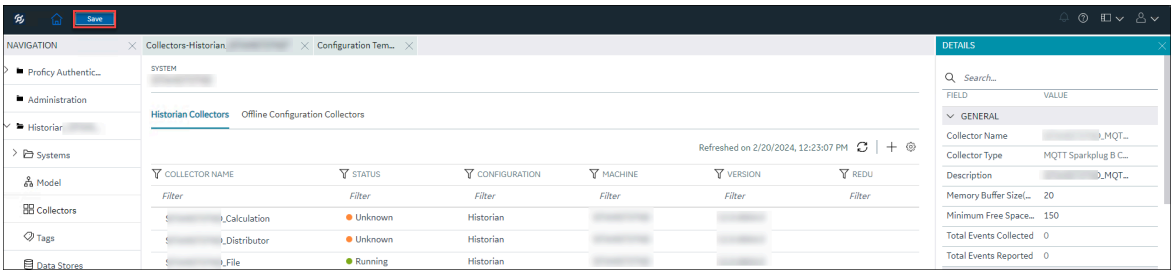

A blank row appears in the table.

6. Enter values as described in the following table, and press ENTER.

| Field | Description |
|---|---|
| **ALIAS** | Enter a name for the contained type. A value is required. It need not match the original name of the contained type that you want to include.<br><br>The value that you enter:<br>◦ Must begin with a letter or a number.<br>◦ Can contain up to 256 characters.<br>◦ Can include any of the following special characters: /!\|#{}%$-_<br>◦ Must not include a space or any of the following characters: ~`+^:;.,?"*=@ |
| **TYPE** | Select the object type that you want to include as a contained type. This field contains a list of all the object types in the Historian system. |
| **TEMPLATE** | Select the template from the object type that you want to include. This field contains a list of templates in the object type. Only the variables in the selected template will be inherited by instances of the object type in which you want to include the contained type. |

The contained type is included in the object instance.

7. In the upper-left corner of the page, select **Save**.




The changes to the object type are saved.

Create an object instance *(on page 388)*. The object instance will include the variables directly added in the object type, along with the ones in the selected template in the contained type.

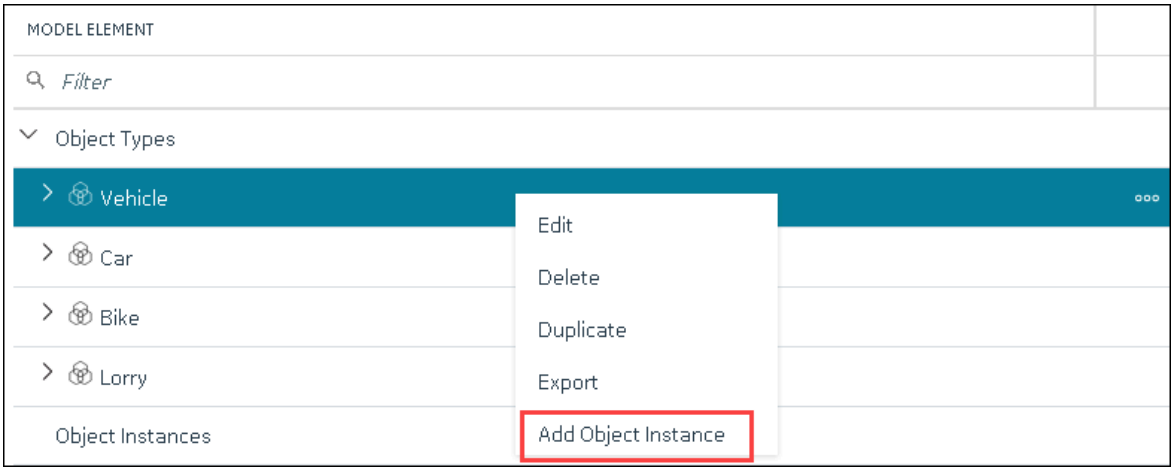
## Create an Object Instance

Create an object type *(on page 380)*.

For each item of an object type, you must create an object instance so that you can capture values of the variables in the instance. These values are then stored in Historian.

**Note:**
After you create an object instance, you cannot rename, import, export, or delete a variable, contained types, or templates in the associated object type; you can only create new ones.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Model**.
   Alternatively, you can select **Systems**, and then in the row containing the system in which you want to create a model, select ⋯, and then select **Browse Model**.
   The **Model** section appears.
3. In the **SYSTEM** list, select the system in which you want to create an object instance.
4. Under **Object Types**, right-click the object type whose instance you want to create (or select ⋯), and then select **Add Object Instance**.




The **New Object Instance** window appears.

5. Enter values as described in the following table.

| Field | Description |
|---|---|
| **NAME** | Enter a name for the object instance. A value is required and must be unique for the object type.<br><br>The value that you enter:<br>◦ Must begin with a letter or a number.<br>◦ Can contain up to 256 characters.<br>◦ Can include any of the following special characters: /!\|#{}%$-_<br>◦ Must not include a space or any of the following characters: ~`+^:;.,?"*=@ |
| **DESCRIPTION** | Enter a description for the object instance. |
| **OBJECT TYPE** | This field is disabled and populated with the object type that you have selected. |
| **OBJECT TYPE TEMPLATE** | Select the template that you want to apply to the object instance.<br><br>**Note:**<br>After you apply a template to an object instance, you cannot modify or delete a |

| Field | Description |
| --- | --- |
| | variable in the object type; you can only add more variables. |

6. Select **Create**.
   The object instance is created.
7. In the **Model** section, under **Instances**, expand the instance that you have created, and then expand **Variables**.
   A list of variables inherited from the template in the object type appear.
8. Select a variable.
   The details of the variable appear in the **DETAILS** section.

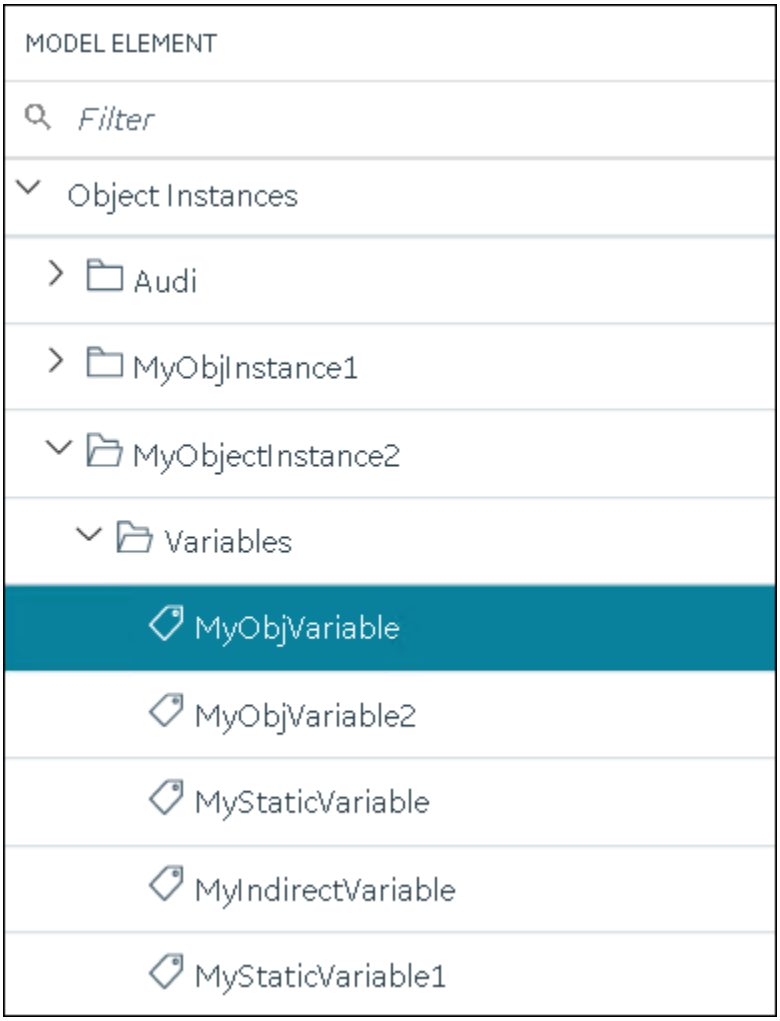
> **Tip:**
> If the **DETAILS** section does not appear, in the upper-right corner of the page, select , and then select **DETAILS**.

- Provide data for static variables *(on page 390)*.
- Collect data for direct *(on page 392)* and indirect *(on page 395)* variables.

## Provide Data for a Static Variable

A static variable contains a fixed value. Therefore, when you create an object instance, you can access the variable, and provide its value.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Model**.
   Alternatively, you can select **Systems**, and then in the row containing the system in which you want to create a model, select , and then select **Browse Model**.
   The **Model** section appears.
3. Under **Object Instances**, expand the object instance, expand **Variables**, and then select the variable whose data you want to provide.

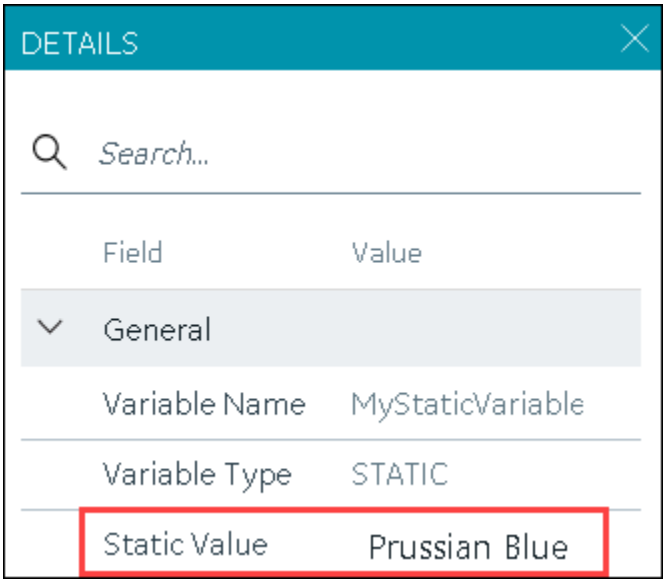



The details of the variable appear in the **DETAILS** section.

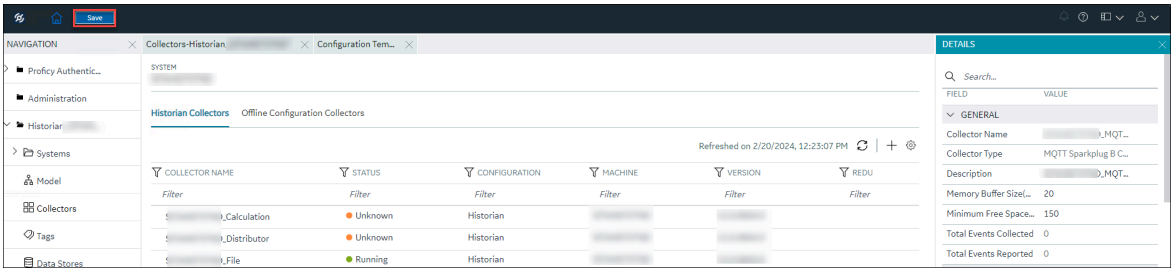
**Note:**

If the **DETAILS** section does not appear, in the upper-right corner of the page, select , and then select **DETAILS**.

4. In the **DETAILS** section, in the **Static Value** field, enter the value that you want to provide for the variable, and then press ENTER.




5. In the upper-left corner of the page, select **Save**.




The value for the variable is saved.

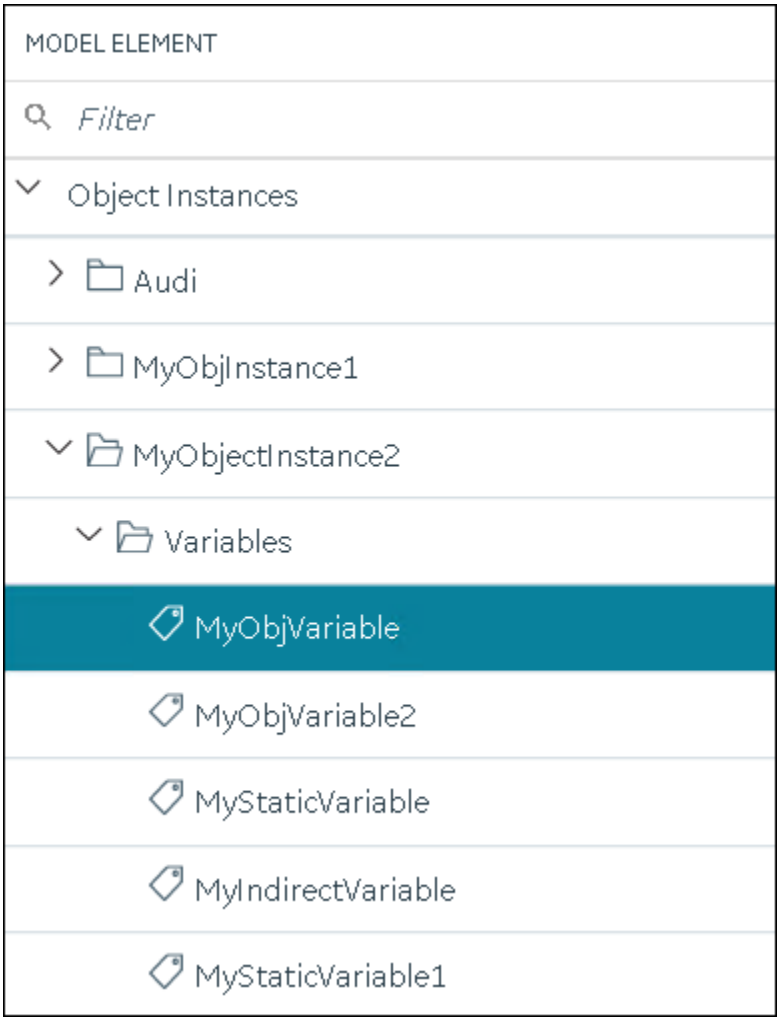
## Collect Data for a Direct Variable

1. Create a collector instance *(on page 357)* using which you want to collect data for the variable.
2. Add the tag *(on page 357)* using which you want to collect data.

To collect data of a direct variable, you must associate the variable with a collector instance and a tag. When you do so, the tag is created in Historian, and values for the variable are collected by the collector instance and stored in Historian.

**Important:**
If the name of a tag associated with a variable in a model contains a period (.), you cannot import the tag while importing the model into a Historian system.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Model**.
   Alternatively, you can select **Systems**, and then in the row containing the system in which you want to create a model, select ⋯, and then select **Browse Model**.
   The **Model** section appears.
3. Under **Object Instances**, expand the object instance, expand **Variables**, and then select the variable whose data you want to collect.

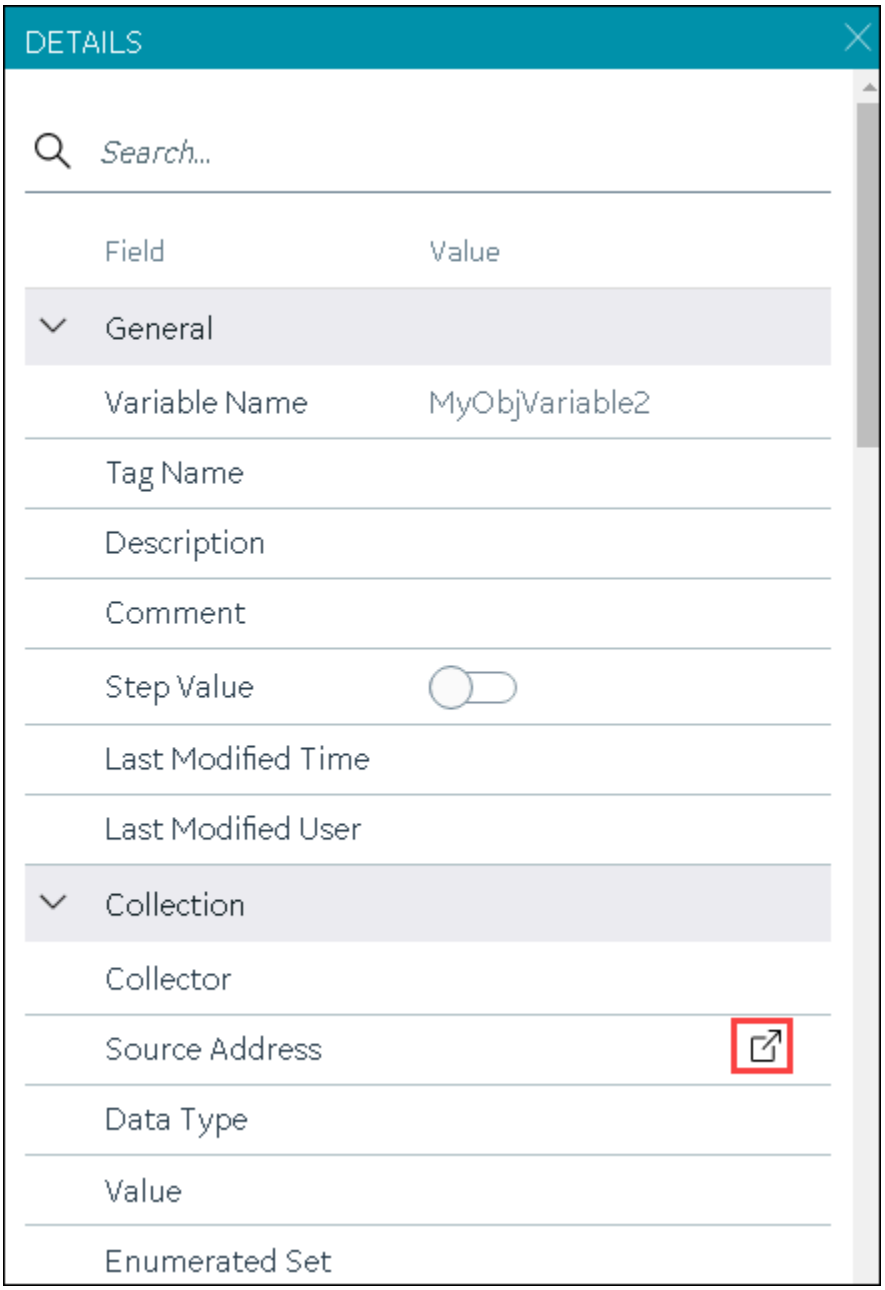



The details of the variable appear in the **DETAILS** section.

**Note:**

If the **DETAILS** section does not appear, in the upper-right corner of the page, select , and then select **DETAILS**.

4. In the **Source Address** field, select .

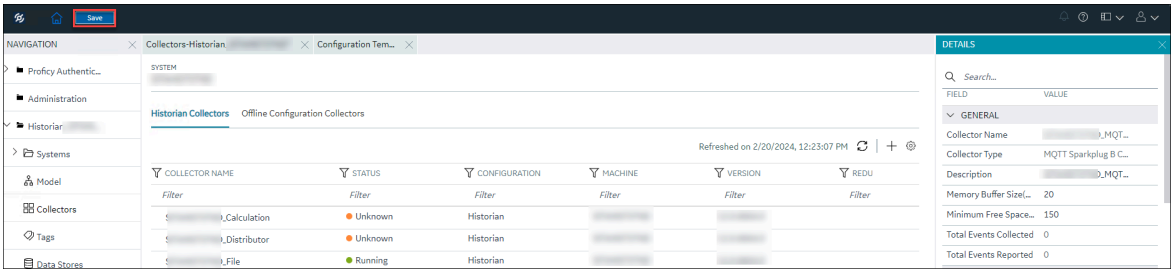



The **Browse Source Tag** window appears.

5. Enter values as described in the following table.

| Field | Description |
|---|---|
| **COLLECTOR NAME** | Enter the name of the collector using which you want to collect data of the variable. A value is required. |
| **COLLECTED TYPE** | Specify whether you want to browse through all the tags in the data source or only from the tags that you have not added yet. A value is required. |
| **SOURCE TAG NAME** | Enter the name of the tag (either completely or partially) to narrow down the search results. |
| **SOURCE TAG DESCRIPTION** | Enter the description of the tag (either completely or partially) to narrow down the search results. |

6. Select **Search Tags**.
   A list of tags that match *all* the search criteria appears.
7. Select the collector tag that you want to map with the variable, and then select **Apply**.
8. In the upper-left corner of the page, select **Save**.




The tag is mapped with the variable. A corresponding tag is created in Historian. The details of the tag and the collector instance are disabled and populated in the **DETAILS** section. All the data that is collected for the tag is now stored in Historian (or in a cloud destination as configured in the collector instance).

## Collect Data for an Indirect Variable

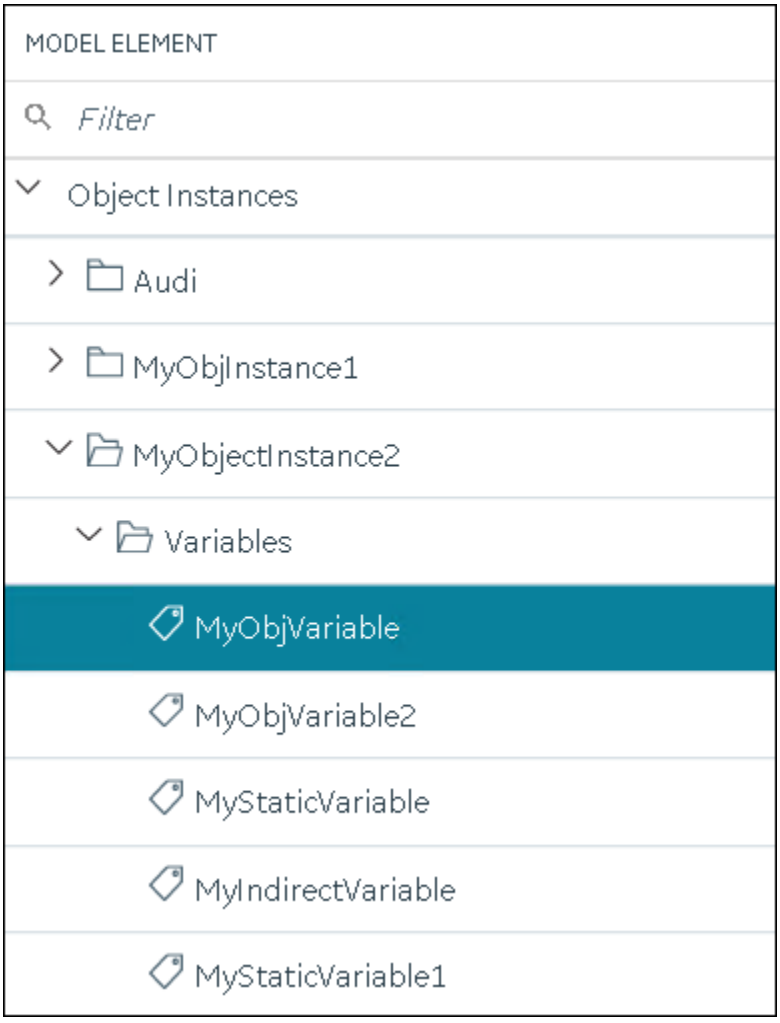
1. Add or ensure there is a collector instance *(on page 357)* which you want to collect data for the variable.
2. Add the tag *(on page 357)* that you want to map with the variable.

To collect data of an indirect variable, you must associate the variable with a collector instance and an existing Historian tag. The values for the variable are then collected by the collector instance and stored in Historian.

**Important:**

- If the name of a tag associated with a variable in a model contains a period (.), you cannot import the tag while importing the model into a Historian system.
- If you want to later delete the tag, first remove the mapping between the tag and the variable.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Model**.
   Alternatively, you can select **Systems**, and then in the row containing the system in which you want to create a model, select ⋯, and then select **Browse Model**.
   The **Model** section appears.
3. Under **Object Instances**, expand the object instance, expand **Variables**, and then select the variable whose data you want to collect.

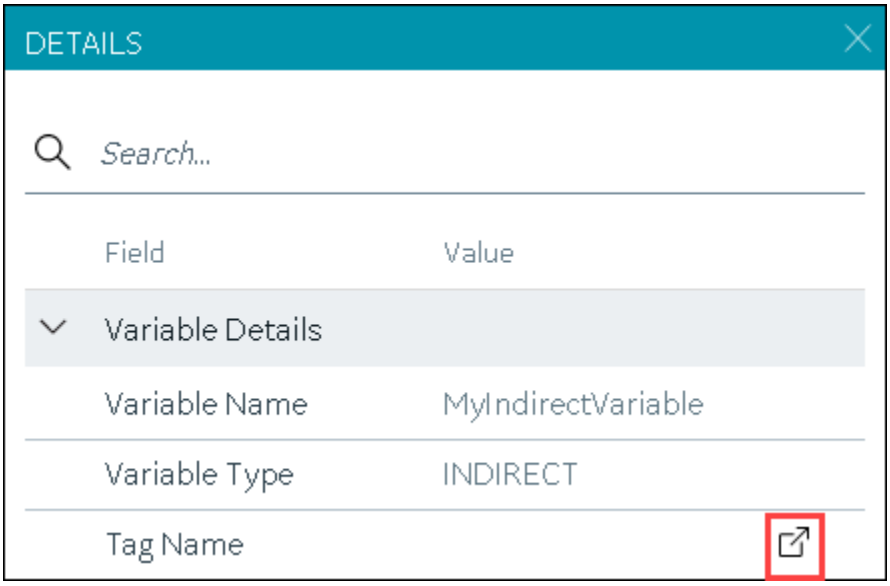



The details of the variable appear in the **DETAILS** section.

**Note:**

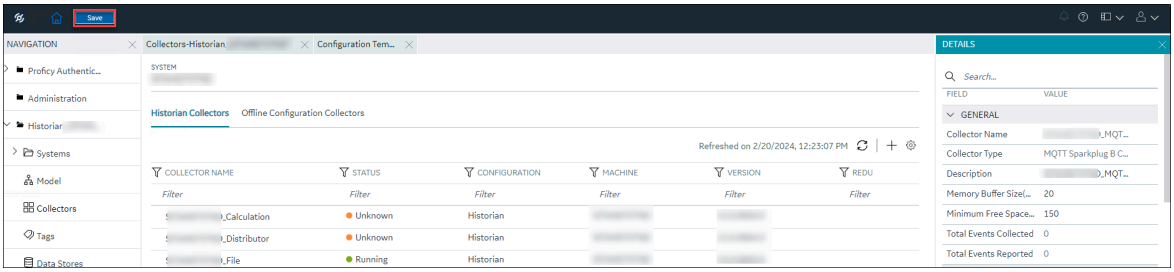
If the **DETAILS** section does not appear, in the upper-right corner of the page, select , and then select **DETAILS**.

4. In the **DETAILS** section, in the **Tag Name** field, select .




The **Tag Selection: <variable name>** window appears.

5. Select **Search** to search for tags.
6. Enter the search criteria, and then select **Search**. You can enter a name or a value partially or use the wildcard character asterisk (*). You can add more search criteria by selecting **Add Attribute**. The list of tags are filtered based on the search criteria.
7. Select the collector tag that you want to map with the variable, and then select **Apply**.
8. In the upper-left corner of the page, select **Save**.




You can choose any of the following options provided while mapping the indirect variable with an existing Historian tag to save in Historian with or without the model hierarchy.

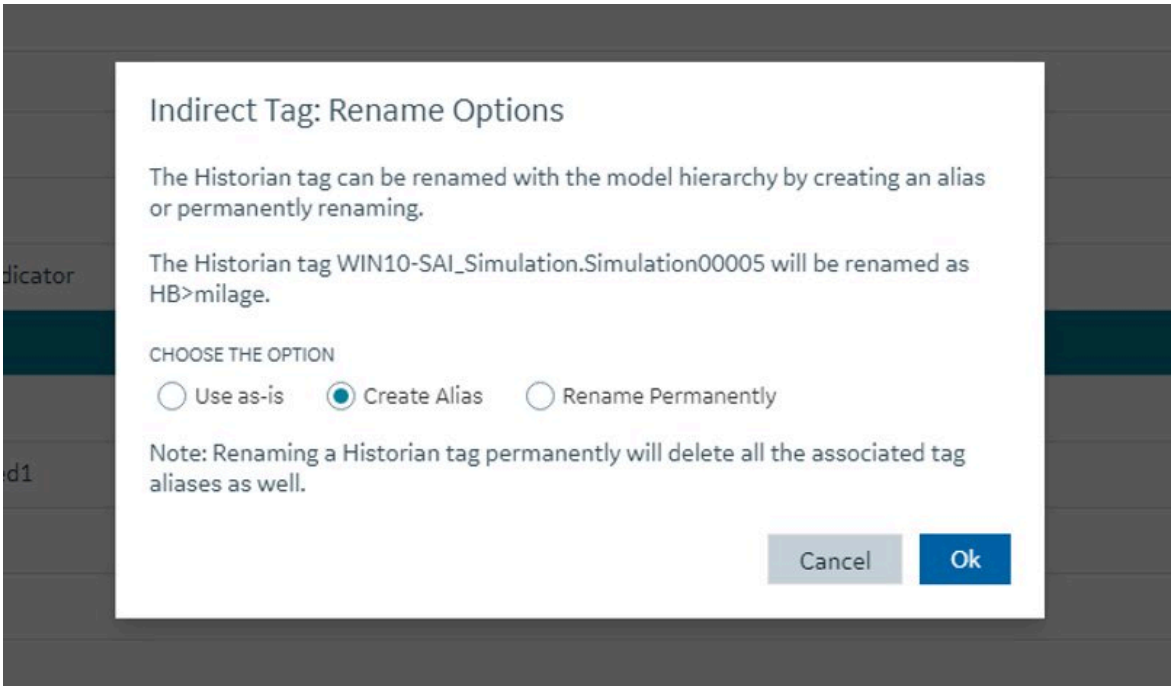



**Table 12.**

| Field | Description |
|---|---|
| **Use as is** | Choosing this option will not update the Historian tag name. Mapping Information will be saved only in model database. Mapping information won't be available with Historian (when you export and try to trend with trending tools like Operations Hub, you will not be able to trend as they are connected to Historian Database). |
| **Create Alias** | Choosing this option creates an alias of Historian tag mapped with indirect variable.<br>This option will save the model tag both in the model database and Historian database as an alias. When you export the model, you will be able to trend and see the data. (This is the default option.) |
| **Rename Permanently** | Choosing this option will permanently rename the Historian tag mapped with the model indi- |

| Field | Description |
|---|---|
| | rect variable and removes the existing tag. So, there will be chance that existing trends using this Historian tag might have impact. |

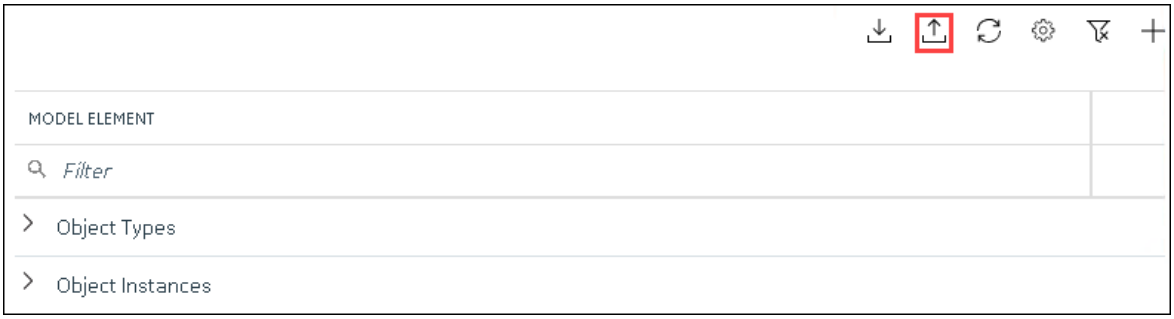
## Export an Object Type/Instance

When you create an object type or an object instance, you can use it only in the Historian system in which you have created it. If, however, you want to use the object type/instance in a different Historian system or Operations Hub, you can export it and then import it into the other Historian system or Operations Hub.

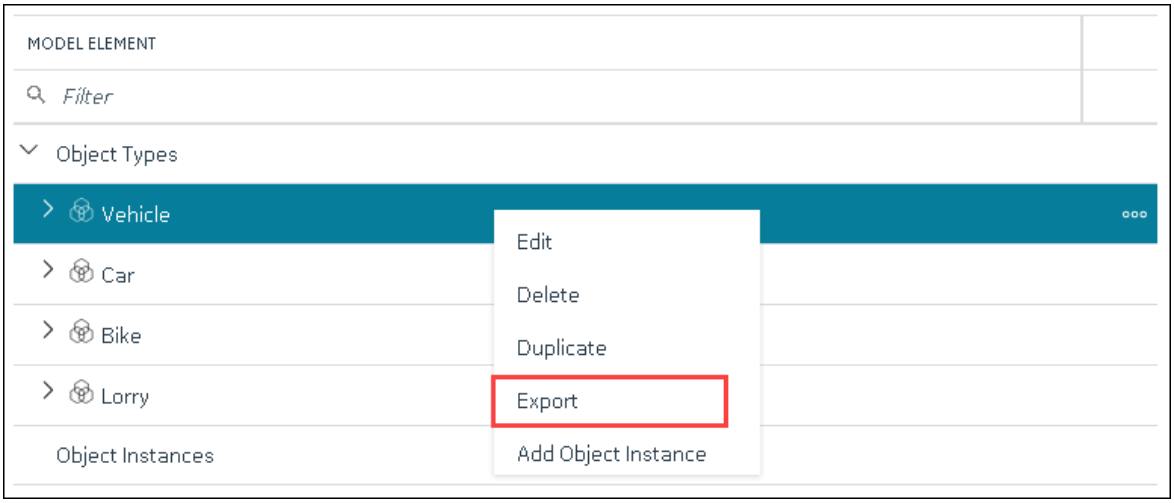
The following conditions apply when you export an object type/instance:

- You can export each object type/instance separately or all the object types and object instances in a Historian system at once.
- You can choose to export the variables or object instances of an object type (or both).
- You can choose to export the variables and templates of an object instance (or both). You cannot, however, export templates to Operations Hub.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Model**. Alternatively, you can select **Systems**, and then in the row containing the system in which you want to create a model, select ⋯, and then select **Browse Model**.
   The **Model** section appears.
3. If you want to export all the object types/instances in the Historian system, in the upper-right corner of the main section, select .

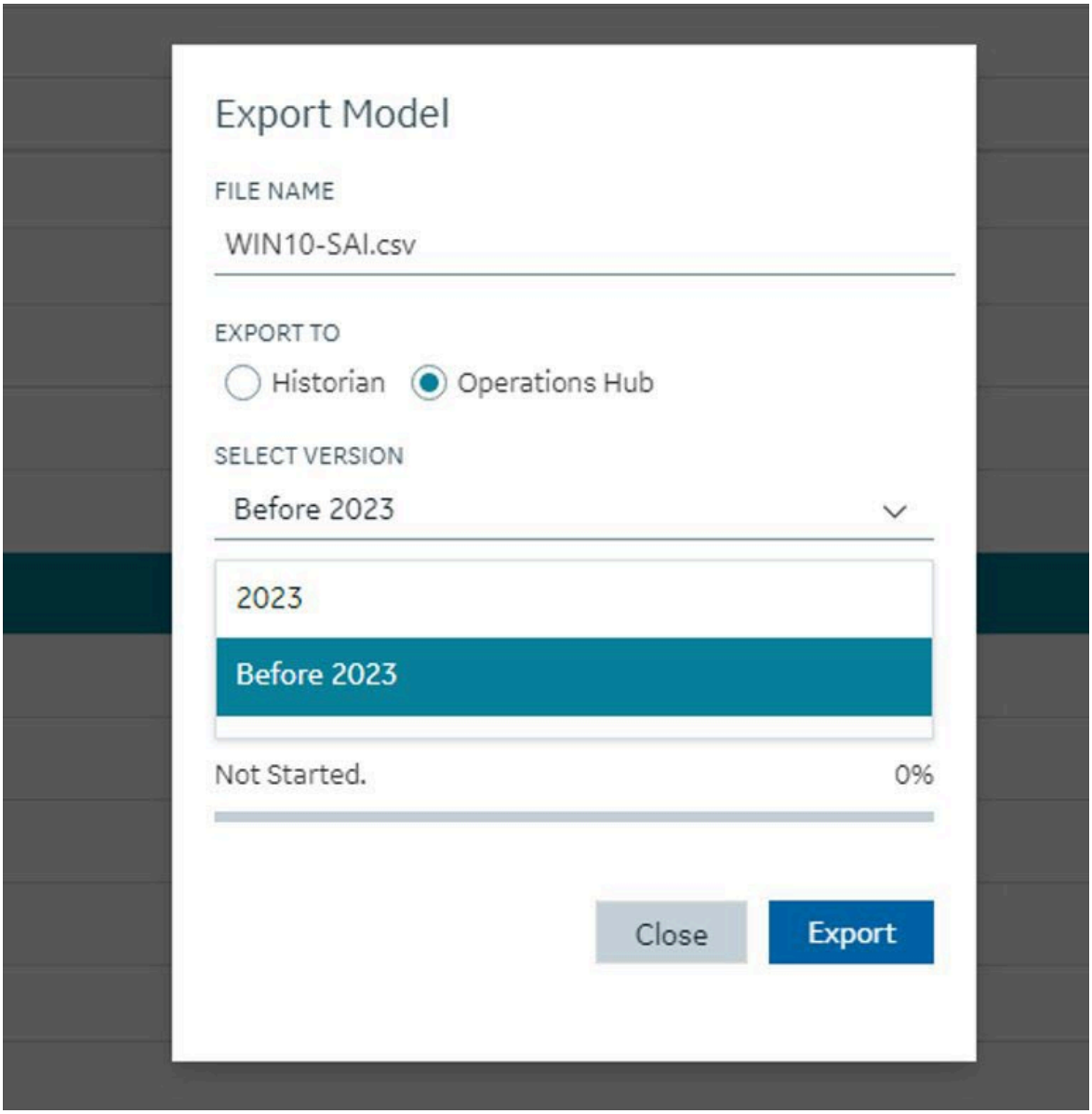



   If you want to export a single object type/instance, under **Object Types** or **Object Instances**, right-click the object type/instance that you want to export (or select ⋯), and then select **Export**.




The **Export Model** window appears. Depending on whether you are exporting all the object types/ instances in the system or just a single one, the **File name** field contains a value in the following format: *<host name>*.csv or *<object type>*.csv or *<object instance*.csv.

4. If needed, modify the value in the **File name** field.
5. Depending on whether you want to export to another Historian system or Operations Hub, select the appropriate option.
6. Operations Hub 2023 supports multi model with multiple roots, you can choose the version while exporting the model either to 2023 or versions prior to 2023.




7. Depending on whether you want to export the templates, the object instances, or both, ensure that the corresponding check boxes are selected. However, you can export templates only to a Historian system, not to Operations Hub.
8. Select **Export**.
   The object types/instances, along with the underlying object variables, are exported in to a .csv file.

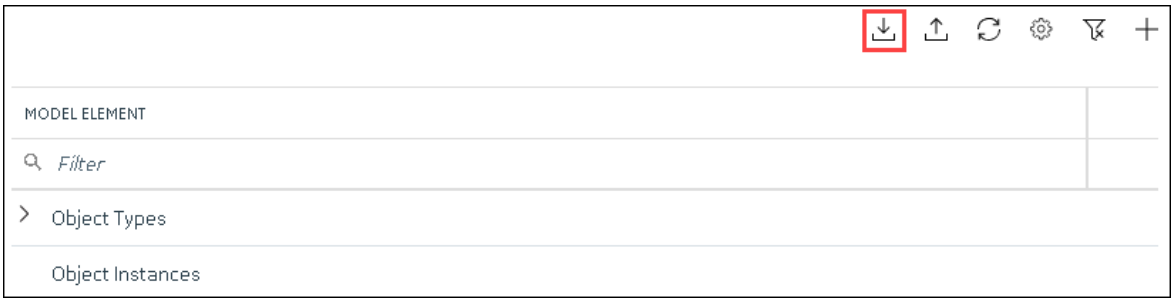
## Import an Object Type/Instance

When you create an object type or an object instance, you can use it only in the Historian system in which you have created it. If, however, you want to use the object type/instance in a different Historian system or Operations Hub, you can export it and then import it into the other Historian system or Operations Hub.

You can import each object type/instance separately or all the object types/instances in a Historian system at once.

**Important:**
If the name of a tag associated with a variable in a model contains a period (.), you cannot import the tag while importing the model into a Historian system.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Model**.
   Alternatively, you can select **Systems**, and then in the row containing the system in which you want to create a model, select , and then select **Browse Model**.
   The **Model** section appears.
3. In the upper-right corner of the main section, select .




   The **Import Model** window appears.
4. Select **Choose File**, and then select the .csv file that contains the object types/instances that you want to import.
5. Depending on whether you want to import the templates, the object instances, or both, ensure that the corresponding check boxes are selected.
6. Select **Import**.
   The object types/instances are imported.

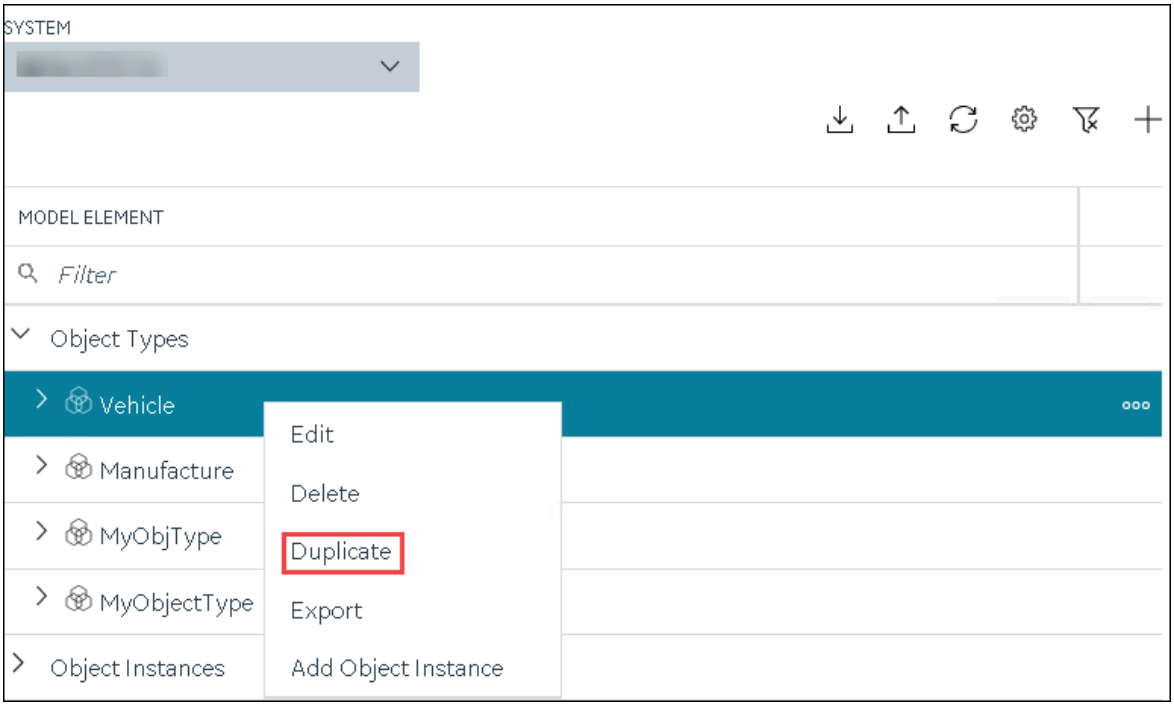
## Copy an Object Type

When you create an object type, you also create the default template, custom templates, and variables for each template. For information on each of these template types and variables, refer to About a Historian Model *(on page 371)* and About Object Templates *(on page 376)*.

When you copy an object type, all the templates and variables are copied too.

This topic describes how to copy an object type. You can also create one *(on page 380)*.

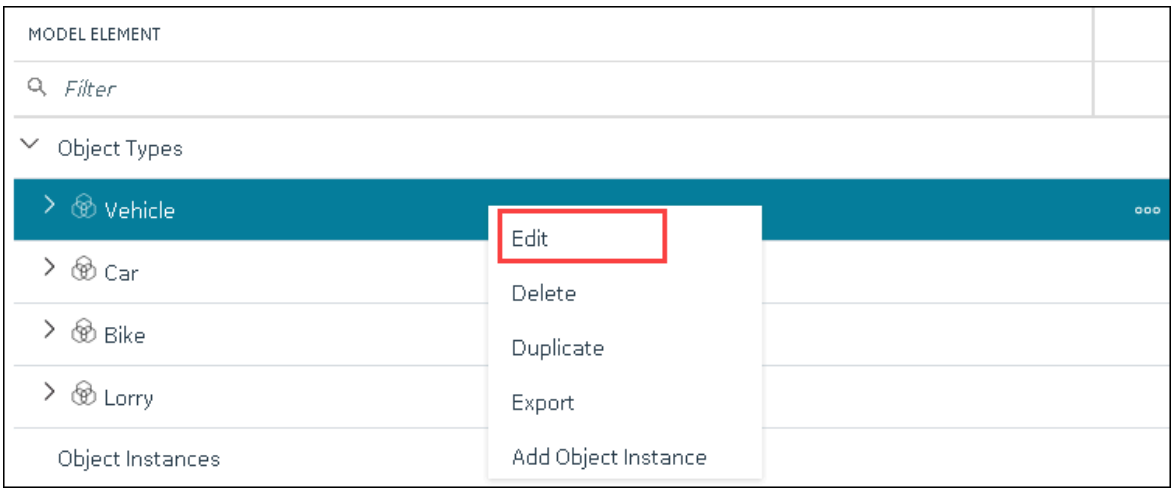
1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Model**.
   Alternatively, you can select **Systems**, and then in the row containing the system in which you want to create a model, select ⋯, and then select **Browse Model**.
   The **Model** section appears.
3. Under **Object Types**, select the object type that you want to copy, and then select **Duplicate**.




   The **Duplicate Object Type** window appears.
4. Enter values as described in the following table.

| Field | Description |
|---|---|
| **NAME** | Enter a name for the object type. A value is required and must be unique.<br><br>The value that you enter:<br>◦ Must begin with a letter or a number.<br>◦ Can contain up to 256 characters.<br>◦ Can include any of the following special characters: /!\|#{}%$-_<br>◦ Must not include a space or any of the following characters: ~`+^:;.,?"*=@ |
| **DESCRIPTION** | Enter a description for the object type. |

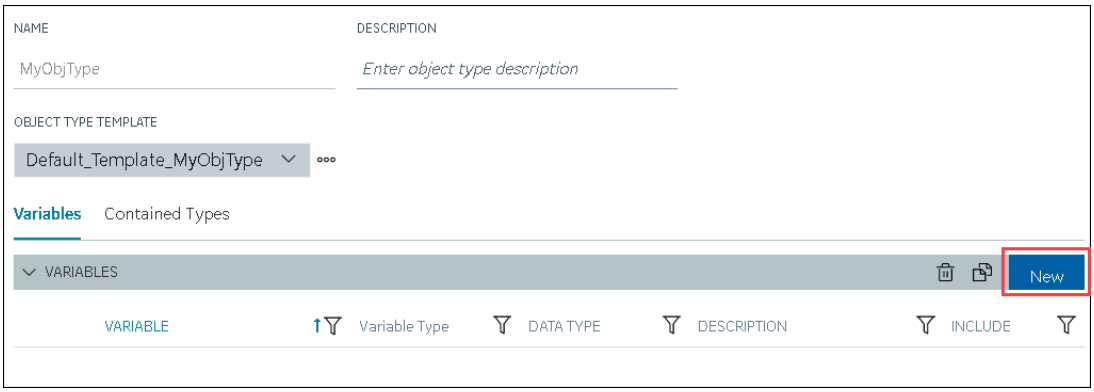
5. Select **Create**.
   The object type is copied, along with the variables and templates in the original object type.
6. Right-click the object type that you have copied (or select ⋯), and then select **Edit**.




   The **<object type name>** section appears. The **OBJECT TYPE TEMPLATE** field contains all the templates in the original object type. In addition, each template contains all the variables as defined in the original object type.
7. To add more variables to a template:

   a. In the **OBJECT TYPE TEMPLATE** field, select the template to which you want to add more variables.

   b. In the **Variables** table, select **New**.




   A blank row appears in the table.

   c. Enter values as described in the following table.

| Column | Description |
|---|---|
| **VARIABLES** | Enter the name of the variable. It must be unique for the object type. |
| **VARIABLE TYPE** | Choose one of the following types of variables:<br>▪ **Direct:** Tags for these variables are created in Historian when you select a collector instance.<br>▪ **Indirect:** These variables are mapped with existing Historian tags.<br>▪ **Static:** These variables have a static value, which you provide when you create an object instance. |
| **DATATYPE** | Select the data type of the variable. |
| **DESCRIPTION** | Enter a description for the variable. |
| **INCLUDE** | Switch the toggle to indicate whether you want to include the variable in the template. |

**Tip:**
If you want to modify a variable, change the values in the aforementioned fields.
If you want to delete a variable, select the check box next to the variable, and then select .

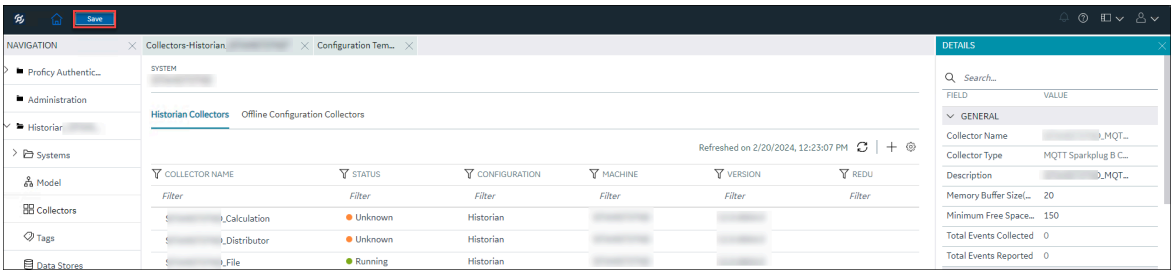
**Note:**
After you apply a template to an object instance, you cannot modify or delete a variable in the object type; you can only add more variables.

d. Press ENTER.

The default template is modified, and the new variables have been added. You can add more variables or include/exclude variables later too.

> **Note:**
> If you want to create a variable by copying an existing one, select the check box next to the variable that you want to copy, and then select . You can copy only one variable at a time.

8. In the upper-left corner of the page, select **Save**.



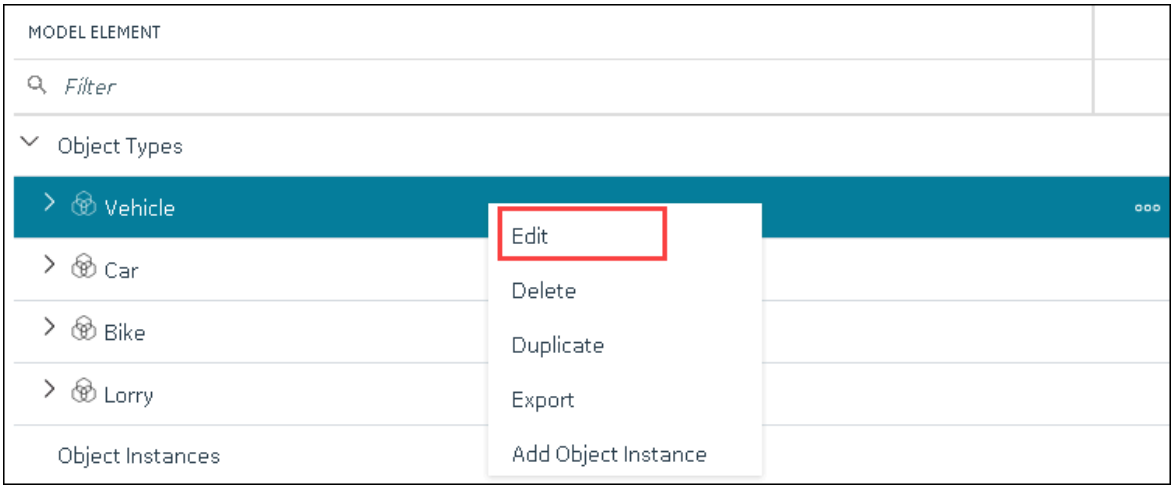

The object type, along with the default template, custom templates, and variables, is created.

Create an object instance *(on page 388)*.

## Delete a Template

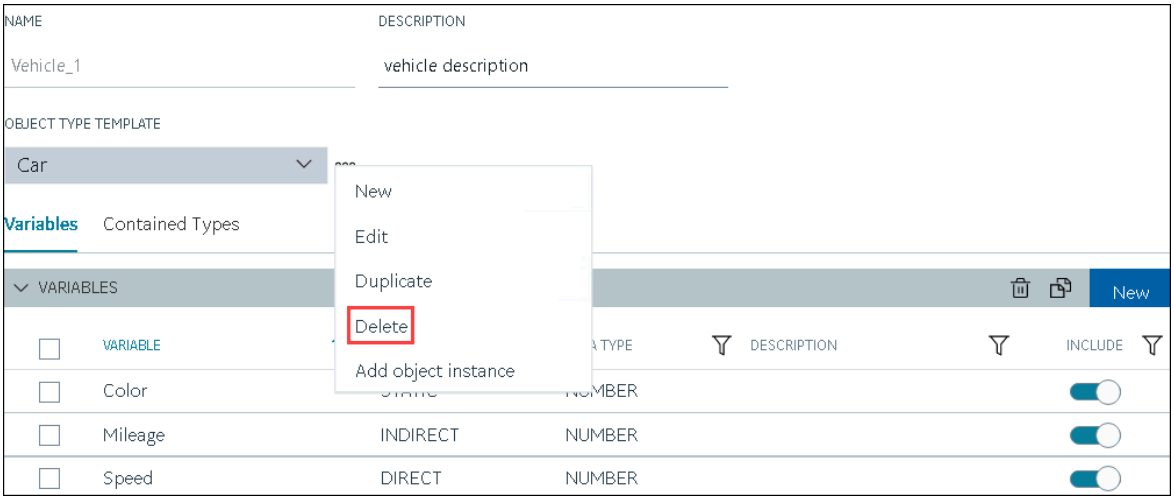
This topic describes how to delete a custom template. You cannot delete a template that is in use (that is, an object instance has been created for the object type). And, you cannot delete the default template in an object type.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Model**.
   Alternatively, you can select **Systems**, and then in the row containing the system in which you want to create a model, select , and then select **Browse Model**.
   The **Model** section appears.
3. In the main section, under **Object Types**, right-click the object type from which you want to delete a template (or select ), and then select **Edit**.




The **<object type name>** section appears. The **OBJECT TYPE TEMPLATE** field contains the default template.

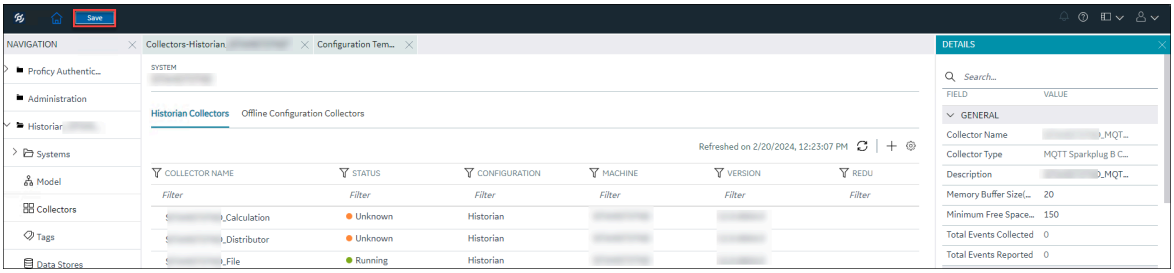
4. In the **OBJECT TYPE TEMAPLTE** field, select the template that you want to delete.
5. Next to the **OBJECT TYPE TEMPLATE** field, select ⋯, and then select **Delete**.




A message appears, asking you to confirm that you want to delete the template.

6. Select **Yes**.

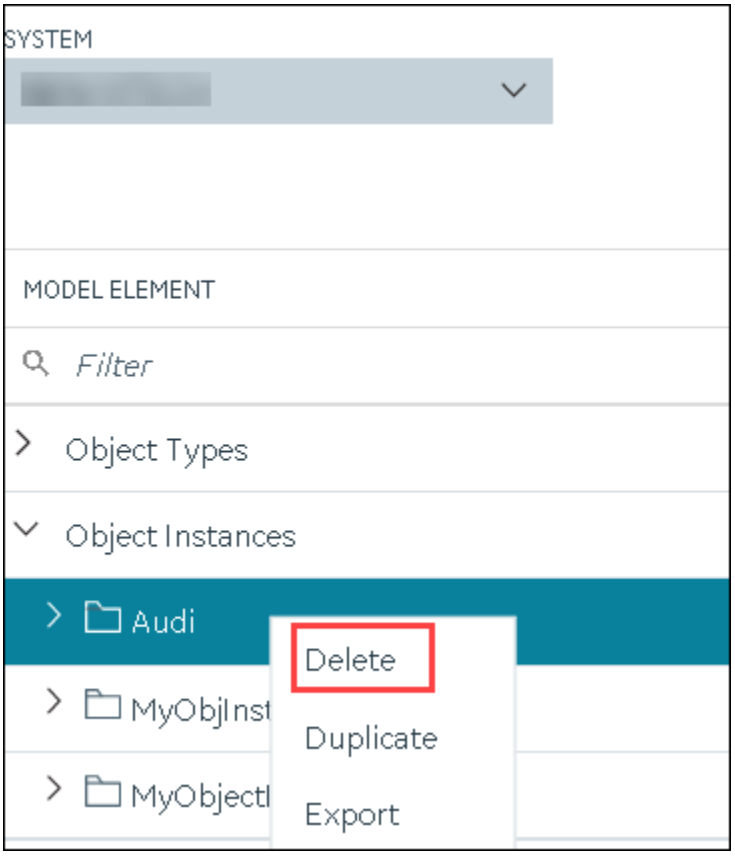
   The template is deleted.
7. In the upper-left corner of the page, select **Save**.




The changes to the object type are saved.

## Delete an Object Instance

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Model**.
   Alternatively, you can select **Systems**, and then in the row containing the system in which you want to create a model, select ⋯, and then select **Browse Model**.
   The **Model** section appears.
3. Under **Object Instances**, right-click the object instance that you want to delete (or select ⋯), and then select **Delete**.



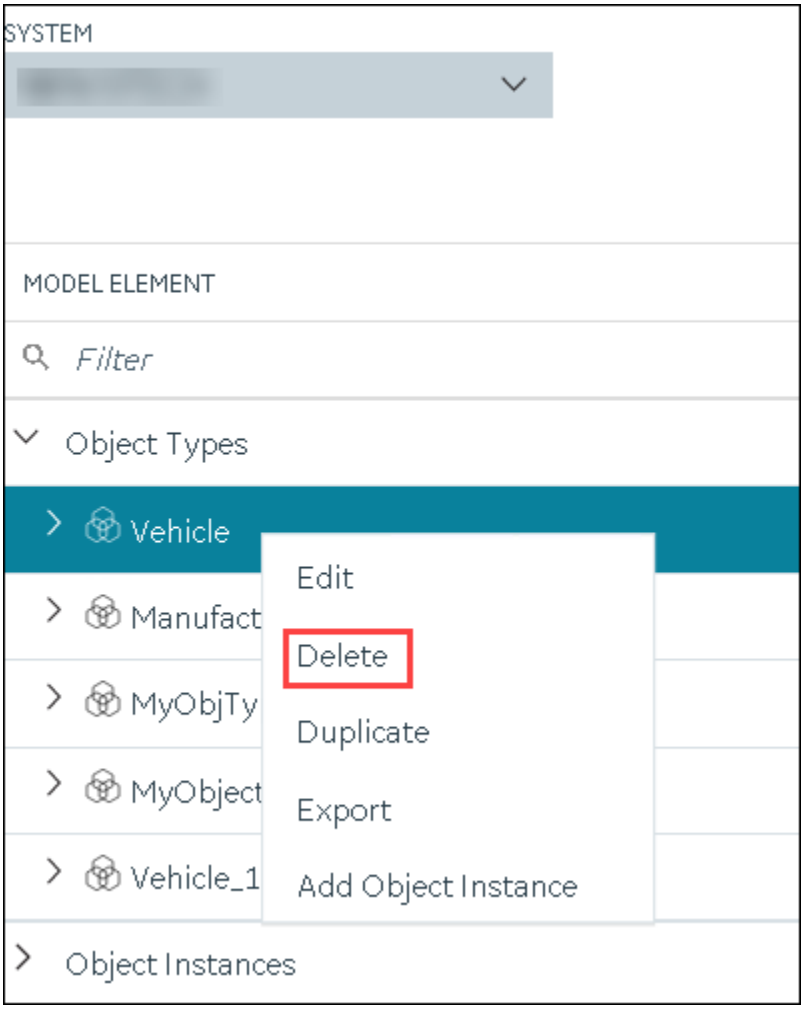
A message appears, asking you to confirm that you want to delete the object instance. If there are direct variables in the object type, you can also choose to delete the tags associated with these variables (along with their data).

4. Select **Yes**.
   The object instance is deleted, along with the underlying variables and templates.

## Delete an Object Type

You cannot delete an object type if it is used in an object instance; you must first delete the object instance *(on page 409)*.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Model**.
   Alternatively, you can select **Systems**, and then in the row containing the system in which you want to create a model, select ⋯, and then select **Browse Model**.
   The **Model** section appears.
3. Under **Object Types**, right-click the object type that you want to delete (or select ⋯), and then select **Delete**.




A message appears, asking you to confirm that you want to delete the object type.

4. Select **Yes**.
   The object type is deleted, along with the underlying variables and templates in the object type. If, however, the object type is used in an object instance, a message appears, asking you to first delete the object instance.

# Managing Historian Systems

## Access a System

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, expand **Systems**, and then select the system that you want to access.
   The system appears in the main section. The following details of the system appear in the **DETAILS** section.

**Table 13. The General Section**

| Field | Description |
|---|---|
| **Name** | The name of the system. |
| **System Type** | The type of the system (whether stand-alone or distributed). |
| **Primary Server** | The primary server of the system. |
| **Description** | The description of the system. |
| **Default System** | Indicates whether the system is a default one. If yes, when you log in to Configuration Hub, this system appears by default. The following conditions apply for a default system:<br>◦ You can have only one default system in Configuration Hub.<br>◦ You cannot delete a default system.<br>For instructions of setting a default system, refer to Set a Default System *(on page 445)*. |
| **Collectors** | The number of collectors in the system. |
| **Tags** | The number of tags in the system. |
| **Data Stores** | The number of data stores in the system.<br>**Tip:**<br>If you hover over, the names of the data stores will be displayed. |
| **Clients** | The number of clients in the system. |
| **Server Time** | The current time of the server. |
| **Server Version** | The version of the server. |
| **Demo Mode** | Indicates whether the server is currently in demo-license mode. |
| **Clustered** | Indicates whether the server is currently in clustered environment. |

**Table 14. The System Defaults Section**

| Field | Description |
|---|---|
| **Default Data Store** | The default data store in the system. A default data store is the one that is considered if you do not specify a data store while adding a tag. For instructions on setting a data store as default, refer to Set a Default Data Store *(on page 462)*. |
| **Default Location** | The default location in the system. A default location is a server in a system which is considered when you do not specify a location while creating a Data Store. By default, the distributed location on the primary server is the default location. You can, however, set a different default location. To set the default location select . |

**Table 15. The Alarms and Events Section**

| Field | Description |
|---|---|
| **Alarms Rate** | The rate at which alarm data *(on page 750)* is collected in the system. |

**Table 16. The License Section**

| Field | Description |
|---|---|
| **Historian Tags** | The number of tags in the system (out of the total number of licensed tags).<br><br>**Note:**<br>If this field displays 100 tags and the **Users** field displays 1 client, you are likely running in demonstration mode and may have incorrectly installed your hardware key. |
| **Scada Tags** | The number of SCADA tags in the system (out of the total number of licensed SCADA tags). |
| **Users** | The number of users in the system (out of the total number of users authorized to access Historian using the software key and license). |

| Field | Description |
|---|---|
| | The number of users that are authorized to access Historian is strictly based on the software key and license. However, if you have utilized your available Client Access Licenses (CAL) and need an additional one to use the system in an emergency, you have an option to reserve a CAL. This reserved CAL allows you to access the server. To do so, provide the reserved CAL to the system administrators and add them to the ih Security Admins group. A system administrator can then connect to Historian in an emergency.<br><br>This facility is optional and does not provide a guaranteed connection. It only eliminates the emergency situations when a CAL is preventing you from accessing the system and may not work if there are other conditions. For example, if the Historian server is busy, you will not be able to connect using this feature. |
| **Data Stores** | The number of data stores in the system (out of the total number of licensed data stores). |
| **Calculations** | Indicates whether the Calculation collector is licensed on the software key. |
| **Server to Server** | Indicates whether the Server-to-Server collector is licensed on the software key. |
| **OPC HDA Server** | Indicates whether the OPC Classic HDA server is licensed on the software key. |
| **OPC UA HDA Server** | Indicates whether the OPC UA HDA server is licensed on the software key. |
| **Model** | Indicates whether the object model is licensed on the software key. |
| **Electronic Signature** | Indicates whether electronic signature is licensed on the software key. |

**Table 17. The Global Security Section**

| Field | Description |
|---|---|
| **Security Group** | Indicates the type of authorization you want to use for the Historian security groups. The following options are available:<br>◦ **Use Domain**- If you select this option, the system will use the groups specific to this domain for authorization. Only the users and groups that belong to the domain will have specific permissions and access rights. For more information on the security groups, refer to Historian security groups *(on page 256)*.<br>◦ **Use Local**- If you select this option, the system will use the groups specific to the local system for authorization.<br>◦ **Use Proficy Authentication**- If you select this option, the system will use the groups specific to Proficy Authentication (UAA) for authorization. Only the users or groups that belong to the Proficy Authentication will have specific permissions and access rights. For more information on the Proficy Authentication groups, refer to about Proficy Authentication groups *(on page 231)*.<br>Before you select this option, ensure that you perform the configurations listed in Configurations to use Proficy Authentication Security Groups *(on page 232)*. |
| **Enforce Strict Client Authentication** | If you enable this option, only known user accounts configured on the Data Archiver server computer will be able to access the Historian server. |
| **Enforce Strict Collector Authentication** | If you enable this option, only known collector connections configured on the Data Archiver server computer will be able to send data to the Historian server. |

For more information on global security, refer to Strict Authentication *(on page 252)*.

**Table 18. The Electronic Signatures/Records Section**

| Field | Description |
|---|---|
| **Require Point Verification** | Indicates whether you must enter identifying information whenever you attempt a restricted action. Whenever you attempt to change the system configuration (for the tag, archive, or collector), a tag value, or another record, you must electronically sign the action with a username and password. If the user is authorized to make this change, the identity of the person, the action performed, and the time it was performed, are all recorded in the audit trail.<br><br>**Note:**<br>◦ The audit features are not dependent on this feature being enabled. Historian audits all user actions regardless of whether this option is enabled.<br><br>Enabling electronic signatures and electronic records also requires you to reverify your identity when you use the Historian Excel Add-in, modify or create a tag, or import data or messages.<br><br>**Note:**This feature is available only if you have purchased the Electronic Signatures and Electronic Records option. |
| **Verification Message** | When point verification is enabled, whenever you attempt to perform an action specified as requiring point verification, you are prompted to authenticate.<br><br>◦ **USERNAME**: This is populated with the user that is logged in to Configuration Hub and disabled.<br>◦ **PASSWORD**: The logged in user's password.<br>◦ **DOMAIN**: The logged in user's domain. |

3. Expand the system in the main section.
   A list of servers in the system appears, displaying the following information.

| Field | Description |
|---|---|
| MACHINE NAME | In a stand-alone Historian system, this column displays the host name of the Historian server. In a horizontally scalable Historian system, this column displays the host name of the primary server. |
| STATUS | The current status of the Historian system. |
| ARCHIVE COMPRESSION | The current effect of archive data compression. At the system level, this value is calculated as the average of the corresponding values of individual servers in the system.<br><br>If the value is zero, it indicates that archive compression is either ineffective or turned off. To increase the effect of data compression, increase the value of archive compression deadbands on individual tags in the **Tags** section to activate compression.<br><br>In calculating the effect of archive compression, Historian counts internal system tags as well as data source tags. Therefore, when working with a very small number of tags and with compression disabled on data source tags, this field may indicate a value other than zero. If you use a realistic number of tags, however, system tags will constitute a very small percentage of total tags and will therefore not cause a significant error in calculating the effect of archive compression on the total system. |
| WRITE CACHE HIT RATIO | The hit ratio of the write cache in percentage of total writes. At the system level, this value is calculated as the average of the corresponding values of individual servers in the system.<br><br>It is a measure of how efficiently the system is collecting data. Typically, this value should range from 95 to 99.99%. If the data is changing rapidly over a wide range, however, the hit percentage drops significantly because current values differ from recently cached values. More regular sampling may increase the hit percentage. Out-of-order data also reduces the hit ratio. |

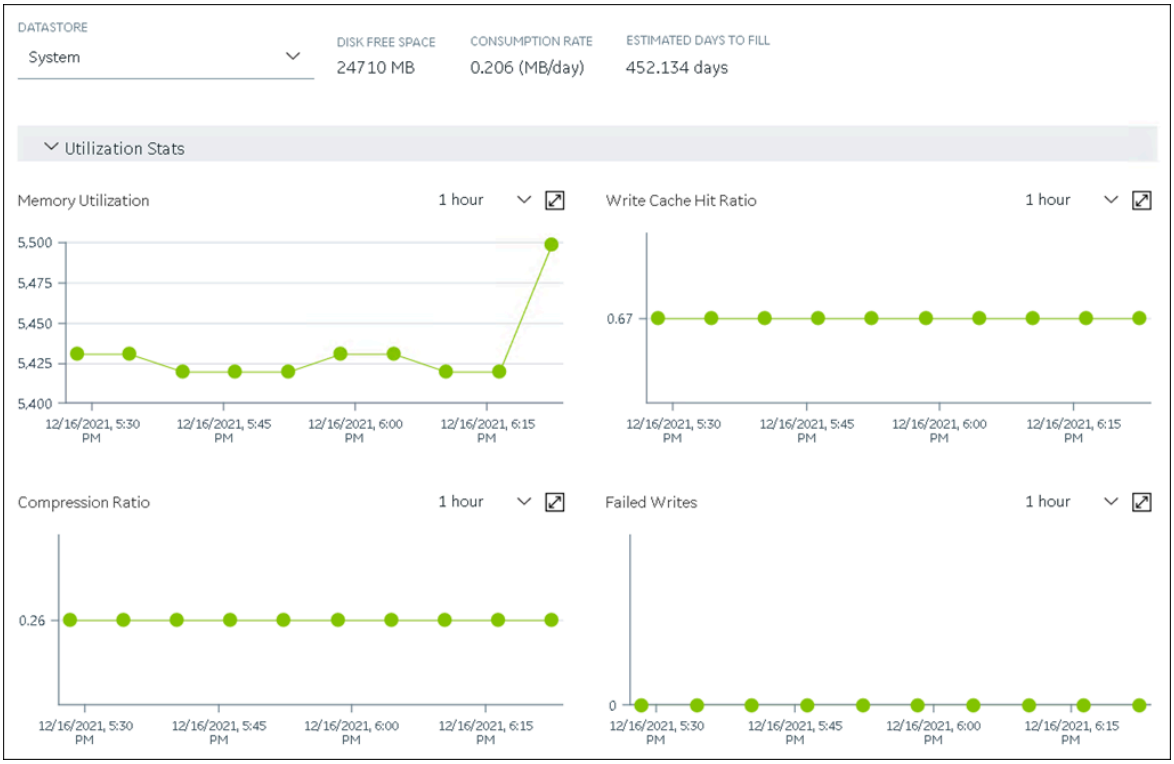
| Field | Description |
|---|---|
| CONSUMPTION RATE | The rate at which the archive disk space is consumed. At the system level, this value is calculated as the sum of the corresponding values of individual servers in the system.<br><br>If the value is too high, you can reduce it by slowing the poll rate on selected tags or data points or by increasing the filtering on the data (widening the compression deadband to increase compression). |
| READ THREAD USAGE | The percentage of the read threads currently in use by the system. At the system level, this value is calculated as the average of the corresponding values of individual servers in the system. |
| WRITE THREAD USAGE | The percentage of the write threads currently in use by the system. At the system level, this value is calculated as the average of the corresponding values of individual servers in the system. |
| OUT OF ORDER WRITE RATE | The number of out-of-order events per minute. At the system level, this value is calculated as the sum of the corresponding values of individual servers in the system. |
| MIN DISK SPACE LEFT | The minimum free disk space in MB that must be available on the computer. If the minimum space required is not available when the collector starts, the collector will shut down. |
| FAILED WRITE RATE (EVENTS/MIN) | The number of samples that failed to be written per minute. At the system level, this value is calculated as the sum of the corresponding values of individual servers in the system.<br><br>Since failed samples are a measure of system malfunctions or an indication of offline archive problems, this value should be zero. If you observe a non-zero value, investigate the cause of the problem and take corrective action.<br><br>Historian also generates a message if a writing a sample fails. Note that the message only appears once per tag, for a succession of failed writes associated with that tag. For example, if the number displayed in this field is 20, but they all pertain to one Historian tag, you will only receive one message until that Historian tag is functional again. |

| Field | Description |
|---|---|
| MEMORY USAGE | Indicates how much server memory is being consumed. |
| READ QUEUE RATE (40 MSG/ MIN) | The number of read requests processed per minute, that came into the archiver from all clients. |
| WRITE QUEUE RATE (MSG/ MIN) | The number of write requests processed per minute, that came into the archiver from all clients. |
| MESSAGE QUEUE RATE (MSG/ MIN) | The number of messages processed per minute. |
| READ QUEUE SIZE (EVENTS) | The total number of messages present in the Read queue. |
| WRITE QUEUE SIZE (EVENTS) | The total number of messages present in the Write queue. |
| MESSAGE QUEUE SIZE (MSG) | The total number of messages present in the Message queue. |

**Tip:**
You can show/hide/reorder columns in the table. For instructions, refer to Common Tasks in Configuration Hub *(on page 353)*.

4. To access the system performance, right-click the system (or select ⋯), and then select **View Server Performance**.

The **<system name> - Performance** section appears, displaying graphs for some of the metrics described in the previous table.




## Access the Collectors in a System

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, expand **Systems**, and then select the system whose collectors you want to access.
   The system appears in the main section.
3. Right-click the system whose collectors you want to access (or select ⋯), and then select **Browse Collectors**.
   This displays the list of Historian collectors and Offline configuration collectors. By default, the Historian collector instances added to the system appear, displaying the following information.

| Column | Description |
|---|---|
| **COLLECTOR NAME** | The name of the collector instance. If you select the link in this column, the details of the collector instance appears. |
| **STATUS** | The status of the collector. Contains one of the following values: |

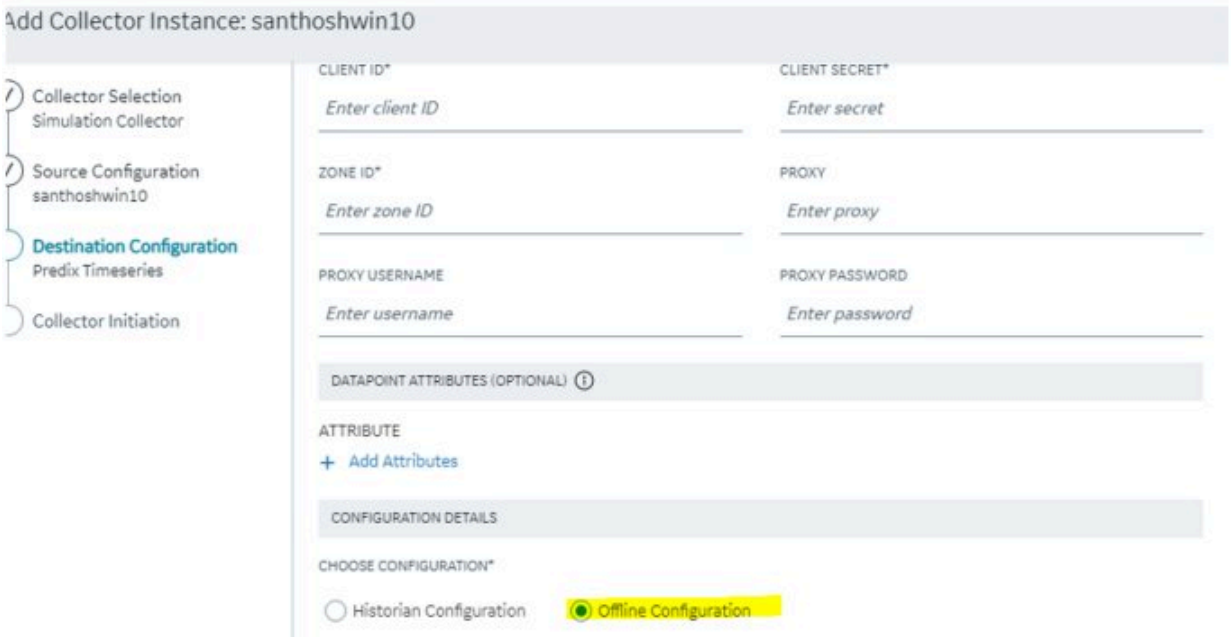
| Column | Description |
|---|---|
| | ◦ **Started**<br>◦ **Stopped**<br>◦ **Running**<br>◦ **Paused**<br>◦ **Unknown** |
| **CONFIGURATION** | The source of the tag configuration for the collector. Contains one of the following values:<br>◦ **HISTORIAN**: Indicates that tags are configured using Historian Administrator.<br>◦ **OFFLINE**: Indicates that tags are configured using an offline configuration *(on page 2039)* file. |
| **MACHINE** | The name of the machine on which the collector is installed. |
| **VERSION** | The version number of the collector. |
| **REDUNDANCY** | Indicates whether collector redundancy is enabled, which decreases the likelihood of lost data due to software or hardware failures. For more information, refer to About Collector Redundancy *(on page 975)*. |
| **REPORT RATE** | The average rate at which the collector is sending data. This is a general indicator of load on the collector. |
| **OVERRUNS** | The total number of data events not collected. In normal operation and under normal conditions, this value should always be zero. If the value is not zero, which indicates that data is being lost, you must take steps to reduce peak load on the system by increasing the collection interval. |
| **COMPRESSION** | The effectiveness of collector compression. If the value is low, you can increase the compression deadbands to pass fewer values and thus increase the effect of compression. |
| **OUT OF ORDER** | The total number of out-of-order samples for the collector. |
| **TAG COUNT** | The number of tags for which the collector collects data. |
| **COMMENTS** | The comments that you have entered for the collector. |

**Tip:**

- To access the details of a collector, select the row containing the collector instance. The details appear in the **DETAILS** section.
- You can show/hide/reorder columns in the table. For instructions, refer to Common Tasks in Configuration Hub *(on page 353)*.

## Access Offline Configuration Collectors

Offline Configuration Collectors are the instances of collectors whose destination is the cloud and display the configuration details as Offline Configuration.



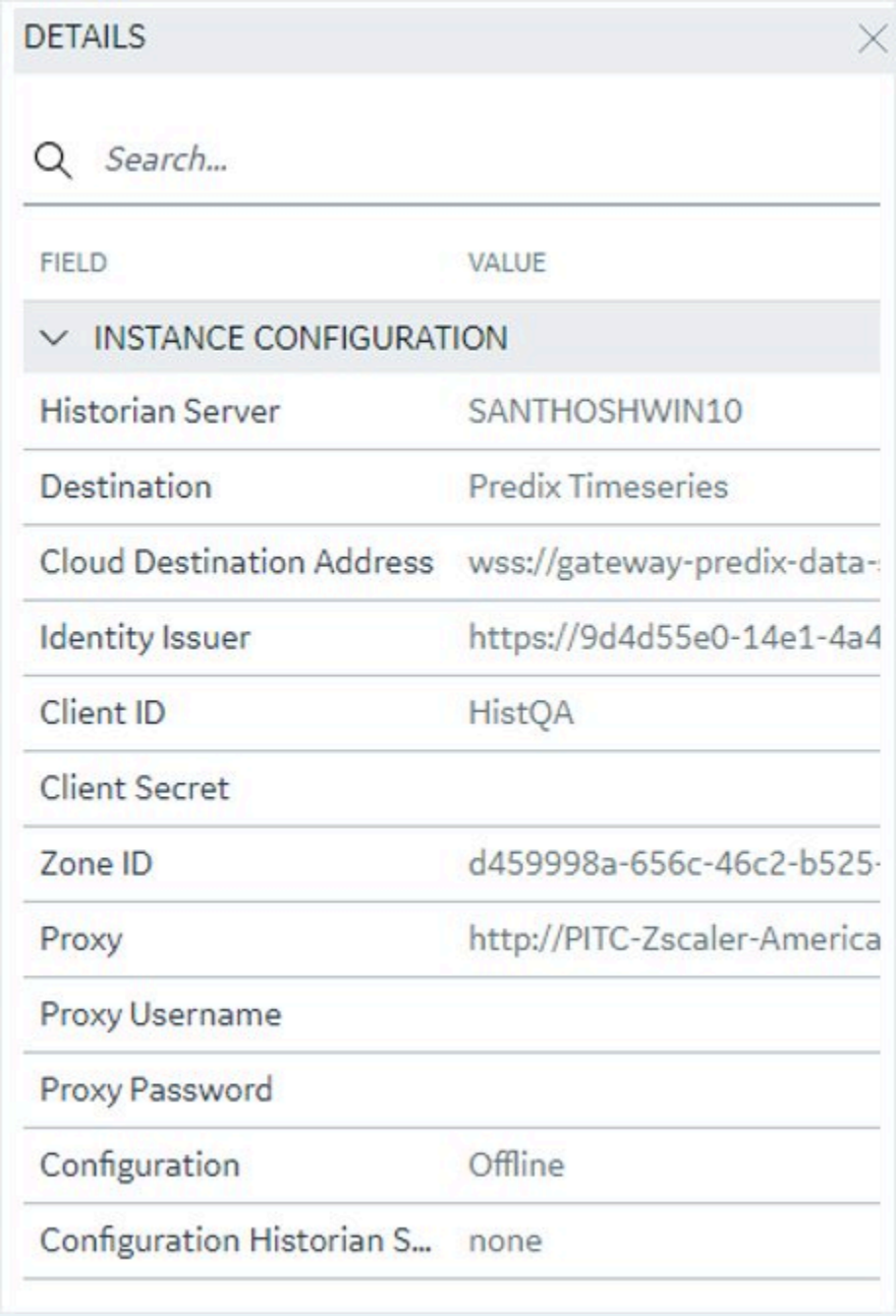

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, expand **Systems**, and then select the system whose collectors you want to access.
   The system appears in the main section.
3. Right-click the system whose collectors you want to access (or select ⋯), and then select **Browse Collectors**.
   This displays the list of Historian collectors and Offline configuration collectors. By default, the Historian collector instances added to the system appear, displaying the following information.

**Tip:**

- To access the details of a collector, select the row containing the collector instance. The details appear in the **DETAILS** section.
- You can show/hide/reorder columns in the table. For instructions, refer to Common Tasks in Configuration Hub *(on page 353)*.

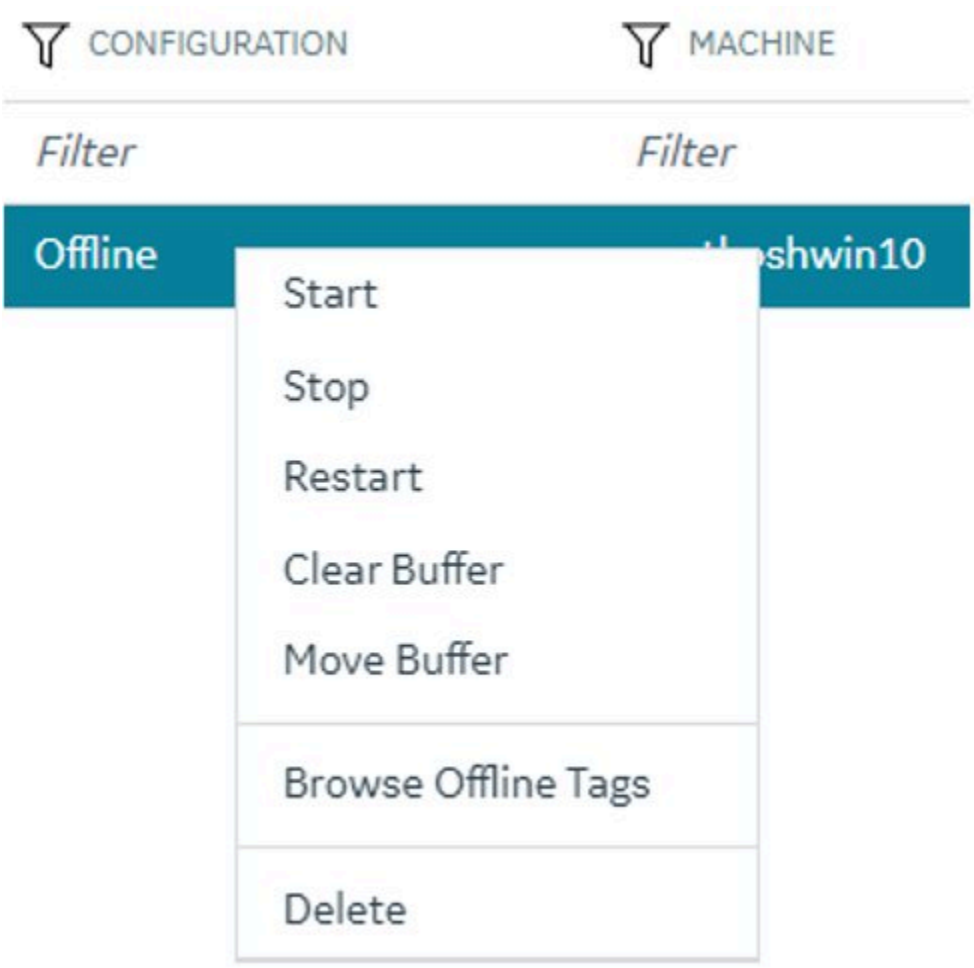
The Details panel for the Offline Configuration Collector Interface contains information regarding the instance configuration provided for the destination while installing software

like Predix, MQTT, or Azure. For example, here is an example of the Details panel for the Offline Configuration Collector:




## Manage Offline Configuration Collectors

Like other Historian collectors, Offline Configuration collectors can also be managed by selecting different options:




Refer to the following sections on how to use these options:

- Start a Collector *(on page 637)*
- Stop a Collector *(on page 638)*
- Restart a Collector *(on page 639)*
- Delete the Buffer Files of a Collector *(on page 643)*
- Move the Buffer Files of the Collector *(on page 644)*
- Delete a Collector Instance *(on page 654)*

## Access the Tags in a System

This topic describes how to access all the tags in a system, regardless of whether they are added to a collector instance. You can also access all the tags added to a collector instance *(on page 631)*.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, expand **Systems**, and then select the system whose tags you want to access.
3. Right-click the system whose tags you want to access (or select ⋯), and then select **Browse Tags**.
   The tags added to the system appears, displaying the following information.

| Column | Description |
|---|---|
| **TAG NAME** | The name of the tag. |
| **DESCRIPTION** | The description of the tag. |
| **COLLECTOR NAME** | The name of the collector instance to which you have added the tag. |
| **LAST 10 VALUES** | The last 10 values collected for the tag, plotted as a trend chart. If you pause over the chart, the minimum, maximum, first, and last values among the 10 values appear. |
| **DATA COLLECTION** | Indicates the status of the data collection. |
| **TAG ALIAS** | Indicates whether the tag contains aliases, which are created when you rename the tag using an alias *(on page 698)*. |

> **Tip:**
> You can show/hide/reorder columns in the table. For instructions, refer to Common Tasks in Configuration Hub *(on page 353)*.

4. To narrow down your search results:
   You can enter a name or a value partially or use the wildcard character asterisk (*).
   a. Select **Search**.
   b. Enter the search criteria, and then select **Apply**. You can add more search criteria by selecting **Add Attribute**.

The list of tags are filtered based on the search criteria. The search criteria that you have provided appear at the top of the page. You can remove any of the criteria as needed.

> **Tip:**
> To access the details of a tag, select the row containing the tag. The details appear in the **DETAILS** section.

## Add a System

Install Historian on the machine that you want to add. If you want to create a stand-alone system, install single-server Historian *(on page 105)*. If you want to create a horizontally scalable system, install Historian primary server *(on page 105)*.

If you want to manage a Historian system using Configuration Hub, you must add it to Configuration Hub.

When you access Configuration Hub for the first time, a default Historian system is available. In a distributed environment, the primary server of this system is the machine whose Configuration Hub details you enter while installing Web-based Clients. This topic describes how to add another system.

> **Note:**
> Adding a Historian system is specific to the logged-in user.

1. Access Configuration Hub *(on page 336)*
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Systems**.
   A list of systems appears in the main section.
3. Select +.
   The **Add System** window appears.
4. Provide values as specified in the following table.

| Field | Description |
|---|---|
| **SYSTEM NAME** | Enter a name for the Historian system. This name must be unique for a user. |
| **HISTORIAN SERVER** | Enter the host name or the IP address of the system that you want to add. This name must be unique for a user. |
| **DESCRIPTION** | Enter a description for the system. |

| Field | Description |
|---|---|
| **Set as Default System** | Select this check box if you want to set this system as the default one. If you do so, when you access Configuration Hub, this system appears by default. The default system varies with the user. |

5. Select **Add**.
   The Historian system is added, and it appears in the **Navigation** section.

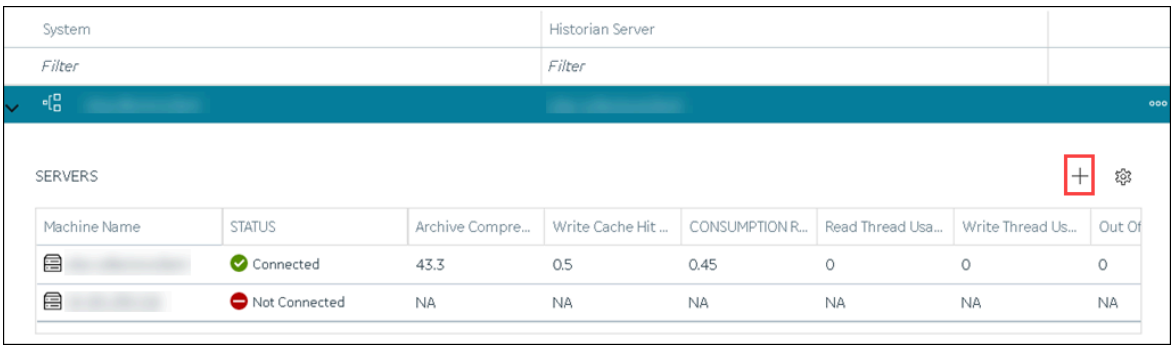
- As needed, add another data store *(on page 370)*.
- If you want to create a horizontally scalable system, the machine that you have added serves as the primary server. On the machines that you want to use as distributed servers, you must install Historian distributed nodes *(on page 105)* and then add them to the system *(on page 361)*.

## Add a Distributed/Mirror Server

1. Install Historian server *(on page 105)* on the machine that you want to add as a distributed server.
2. Add a system *(on page 427)*. The server that you specify while adding the system serves as the primary server for the system.

If you want to create a horizontally scalable Historian system, you must first add a primary server, and then add one or more distributed/mirror machines to scale out the primary server horizontally and thus, improve performance.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Systems**.
   A list of systems appears in the main section.
3. Expand the system in which you want to add a distributed/mirror server.
   A list of servers in the system appears.
4. Select +.



The **Add Server Machine: <system name>** window appears.

5. Enter the host name or IP address of the machine that you want to add, and then select **Add**. The distributed server is added to the system. A distributed location is added in the server. You cannot modify or delete this location.

- If you want high availability of one or more data stores in the server, create a mirror location *(on page 368)*, and then add the data stores *(on page 370)*. If not, add the data store *(on page 370)* to the distributed location.
- If you want to set a distributed node as backup to the primary node, set a distributed node as backup *(on page 365)*.

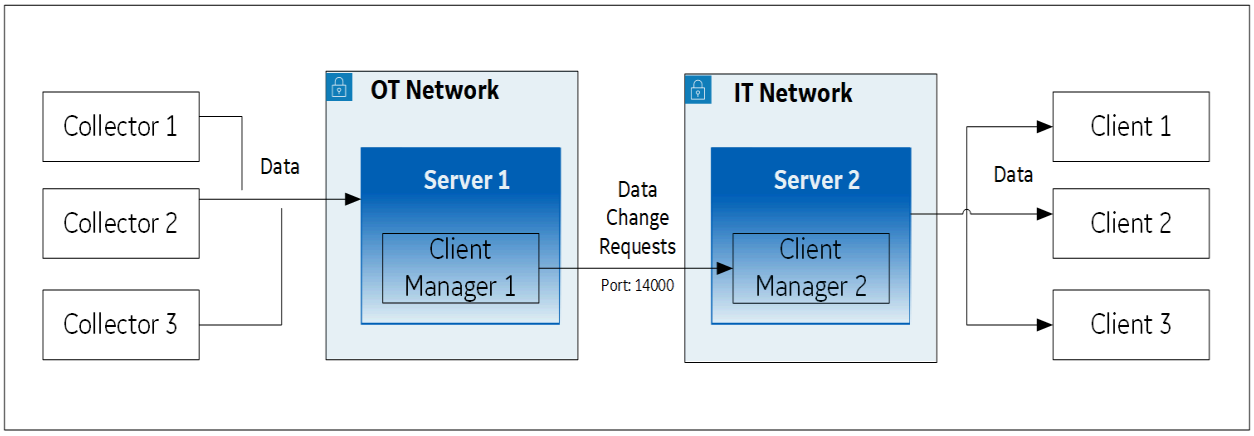
## Set Up a Mirror of Mirror

1. Install Historian server *(on page 105)* on each machine that you want to use in the mirror of mirror setup.
2. Set up Configuration Hub *(on page 305)* on each machine that you want to use in the mirror of mirror setup.
3. Add a system *(on page 427)*. The server that you specify while adding the system serves as the primary server for the system.
4. Create data stores *(on page 370)* in the primary server in the public/IT network with the same name as the data stores in the primary server in your organization network.

You can set up a mirror of the Historian server in a network different from that of your organization. When you do so, any tag/data update requests to the Historian server can be routed to the public and IT network instead of your organization's network.

**Note:**
For the Historian Enterprise Mirror Architecture, only "Time based" archives are supported.

**Single-Node Setup:** The following image shows two networks - OT and IT - with a Historian server installed in each network. These networks communicate using port 14000.



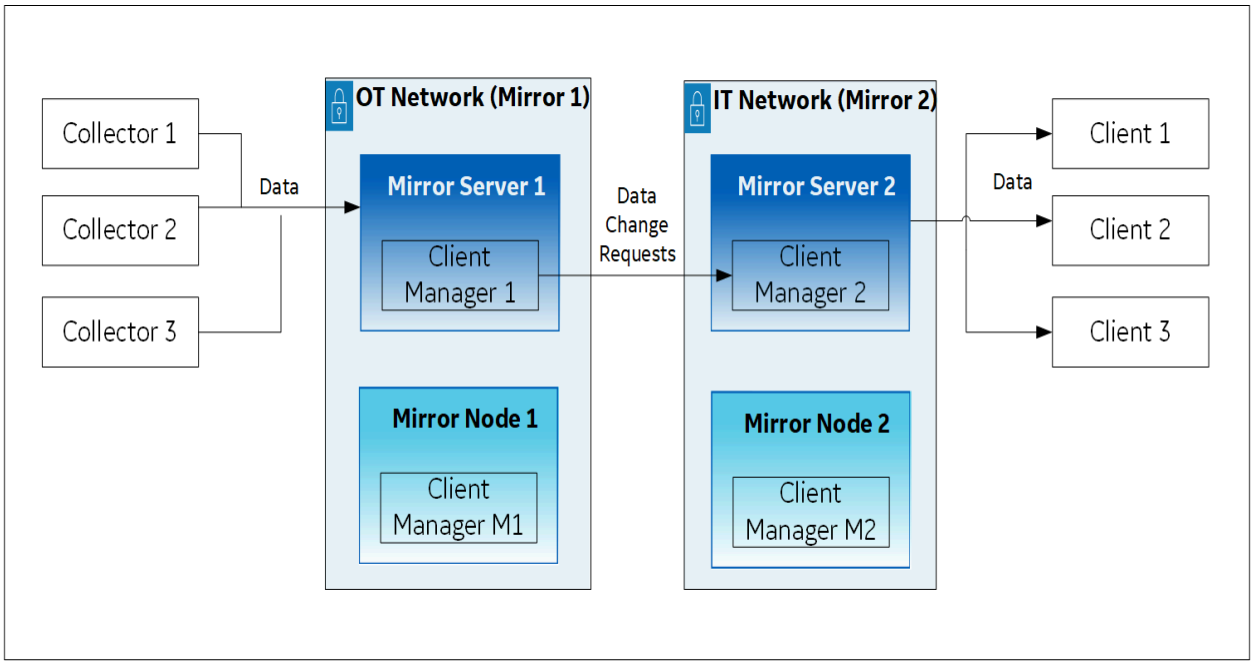

In this setup:

1. Server 1 is the primary server in the OT network; it stores data from collectors.
2. Server 2 is the primary server in the IT network; it is connected to clients.
3. When a tag/data is created, updated, or deleted, Client Manager 1 communicates the same with Client Manager 2 (installed with Server 2 in the IT network).
4. The change in the tag/data is replicated in Server 2 (that is, data is created, updated, or deleted accordingly).
5. The latest data is retrieved from Server 2 using the clients.

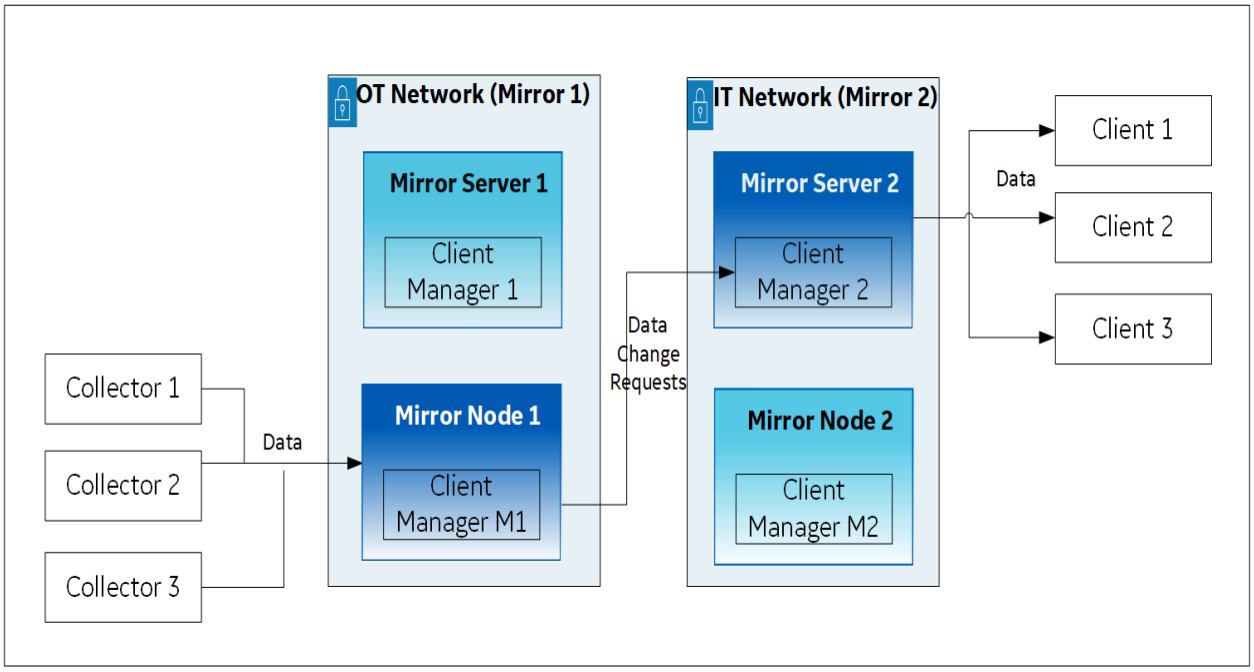
**Mirror Setup:** The following image shows two mirrors:

- Mirror 1 includes Mirror Server 1 and Mirror Node 1, which is a backup/standby node for Mirror Server 1; both these machines are in the OT network.
- Mirror 2 includes Mirror Server 2 and Mirror Node 2, which is a backup/standby node for Mirror Server 2; both these machines are in the IT network.

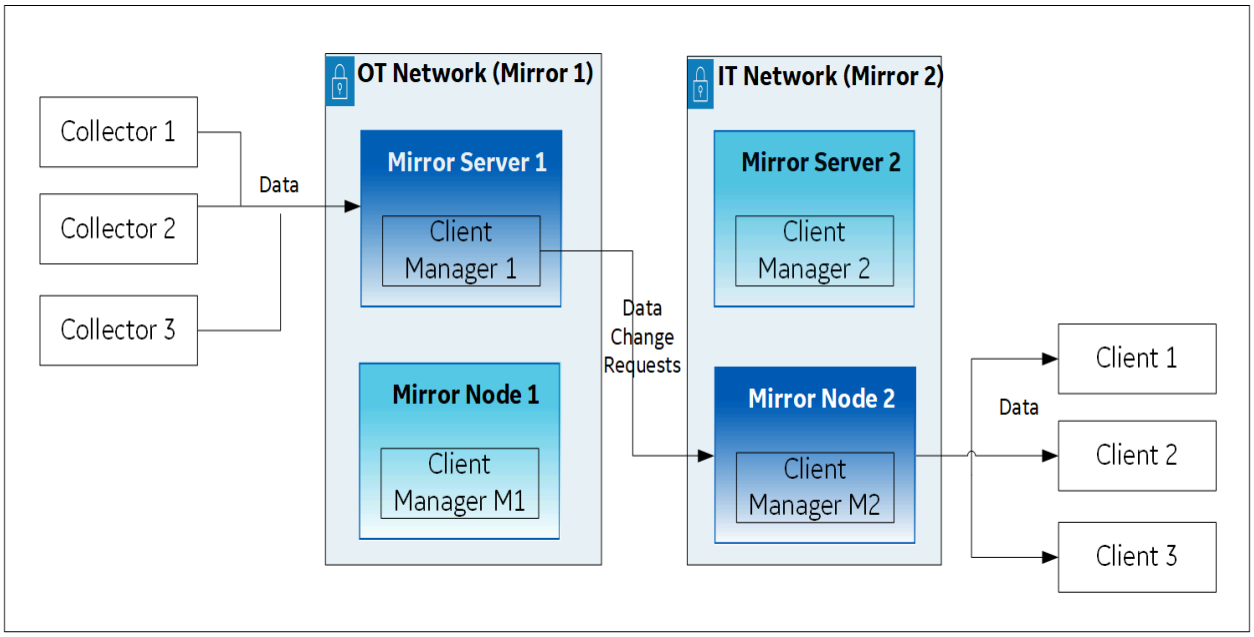
Client Manager 1 in Mirror Server 1 communicates with Client Manager 2 in Mirror Server 2.



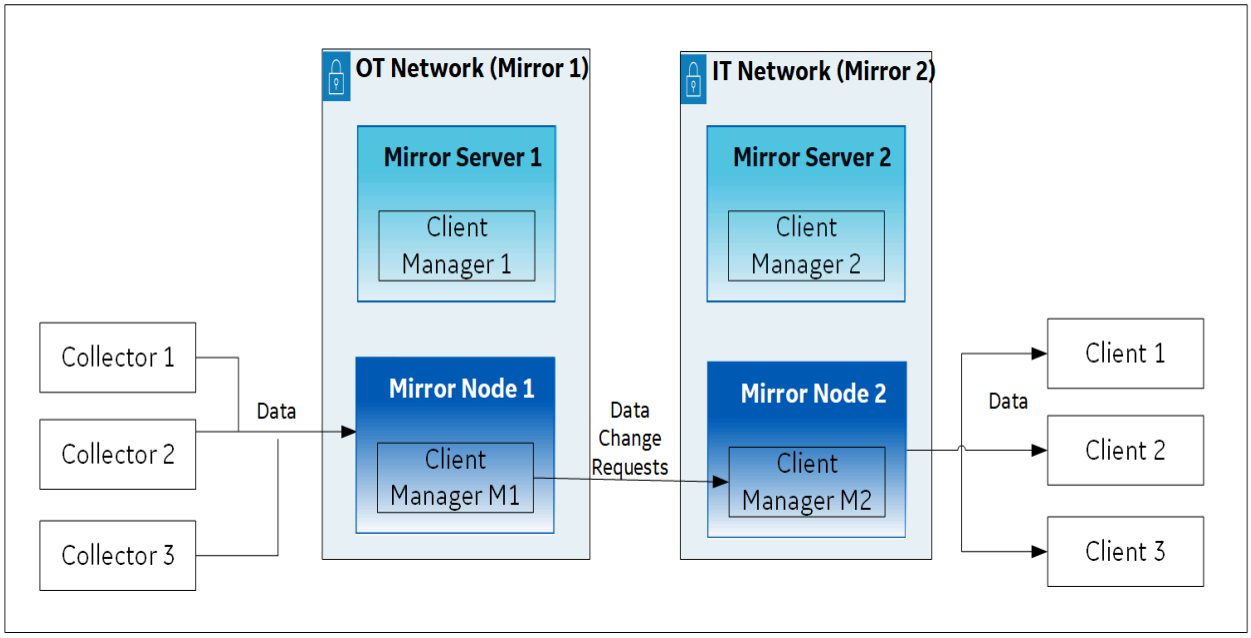

If Mirror Server 1 goes down, Client Manager M1 in Mirror Node 1 communicates with Client Manager 2 in Mirror Server 2.



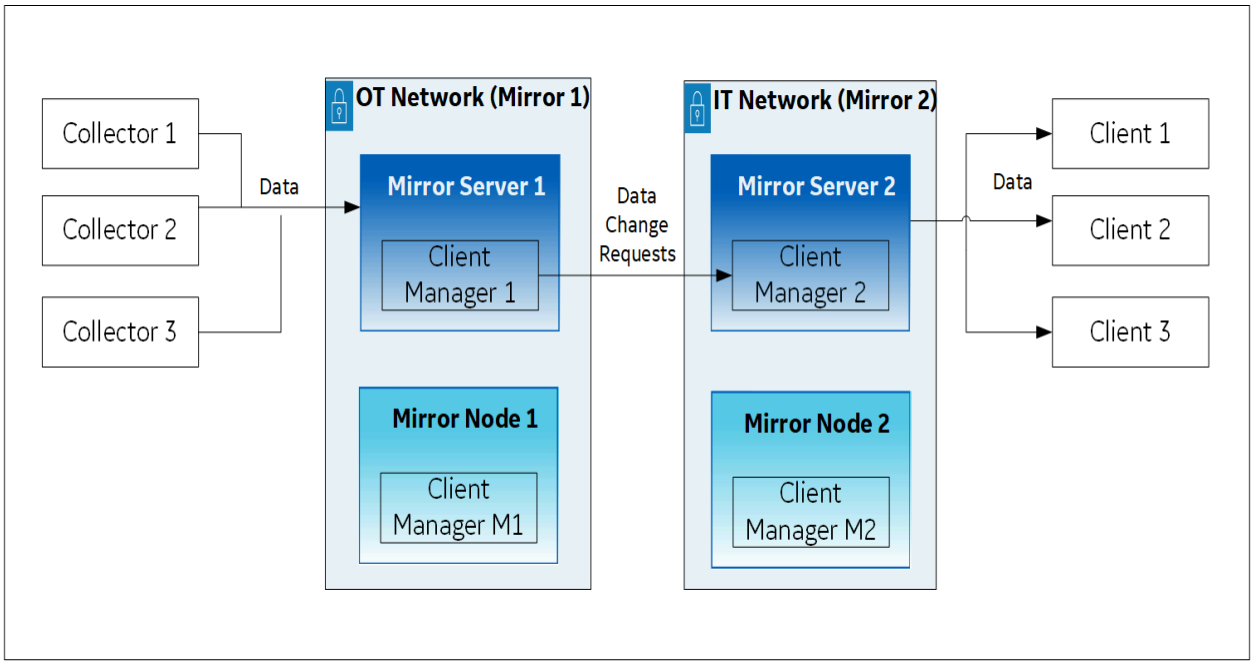

Similarly, if Mirror Server 2 goes down, Client Manager 1 in Mirror Server 1 communicates with Client Manager M2 in Mirror Node 2.




If both Mirror Server 1 and Mirror Server 2 are down, Client Managers M1 and M2 communicate with each other.




If Mirror Server 1 and/or Mirror Server 2 are available, the connection is re-established using these primary servers.



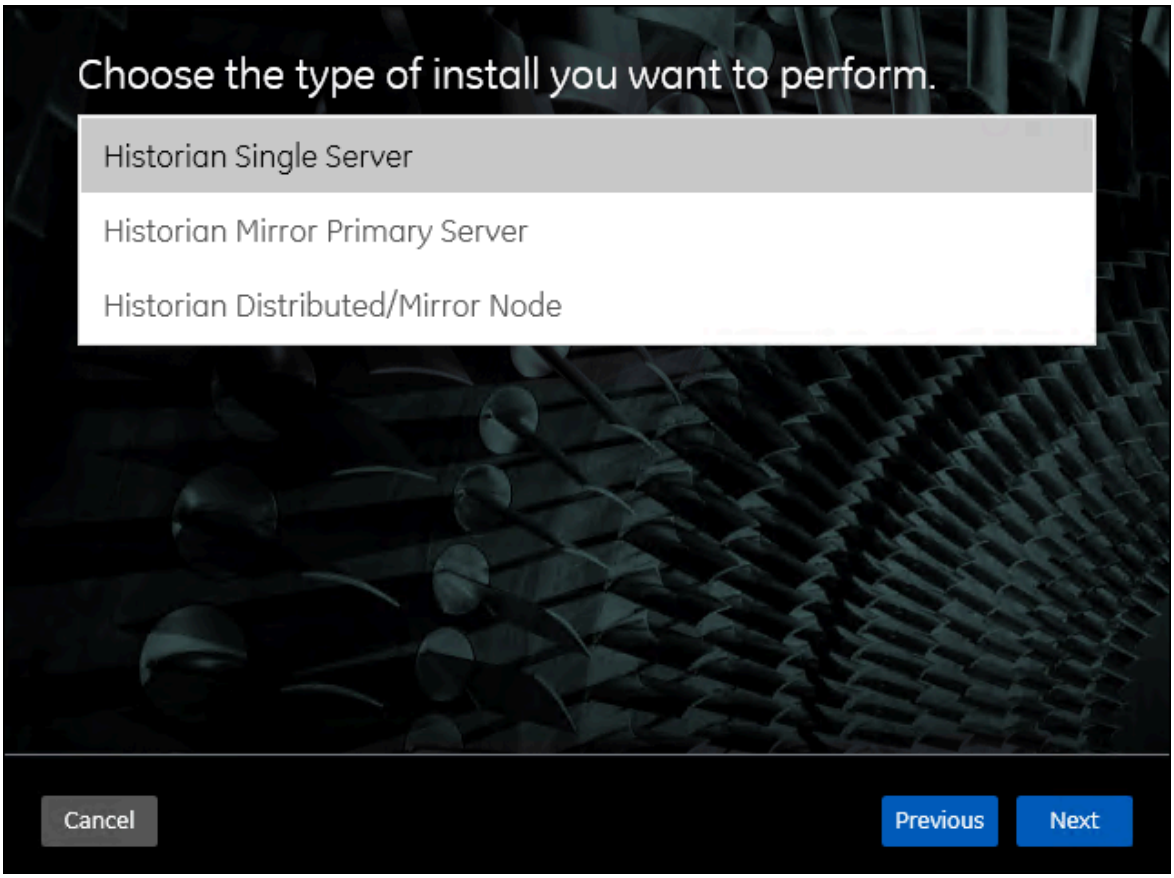

Thus, you can choose to always retrieve data from either Mirror Server 2 or Mirror Node 2. In addition, the store-and-forward functionality is available (in case Client Managers are not yet connected).

This topic describes how to set up a mirror of mirror for the configuration described in the preceding example. It includes the following high-level steps:

1. Installing the Historian server on all the machines
2. Setting up mirror 1
3. Setting up mirror 2
4. Setting up a mirror of mirror

**Installing the Historian server**

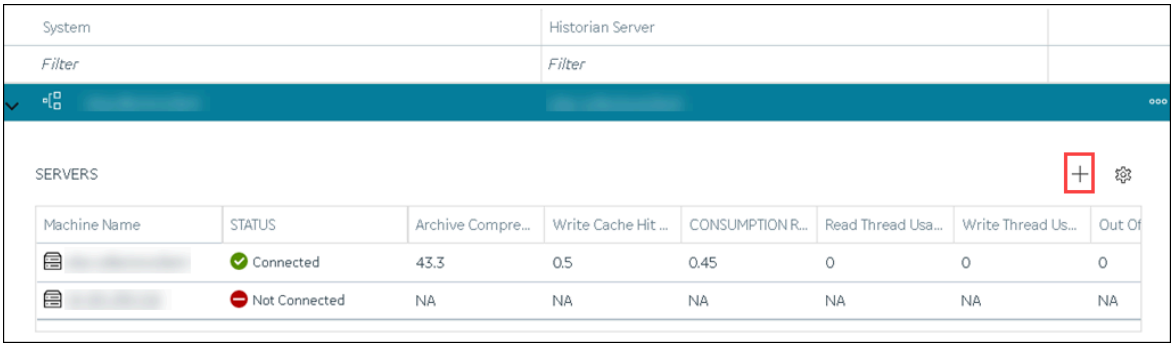
1. On the machines designated as the mirror primary servers (Mirror Server 1 and Mirror Server 2 in the example), install the Historian server *(on page 105)*. During the installation, select **Historian Mirror Primary Server** on the **Choose the type of install you want to perform** page.




2. On the machines designated as mirror nodes (Mirror Node 1 and Mirror Node 2 in the example), install the Historian server *(on page 105)*. During the installation, select **Historian Distributed/ Mirror Node** on the **Choose the type of install you want to perform** page.

**Set up Mirror 1:**

3. On the mirror primary server in your organization's network (Mirror Server 1 in the example), access Configuration Hub *(on page 336)*.
4. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Systems**.
   A list of systems appears in the main section.
5. Expand Mirror Server 1.
   A list of servers in the system appears.
6. Select +.

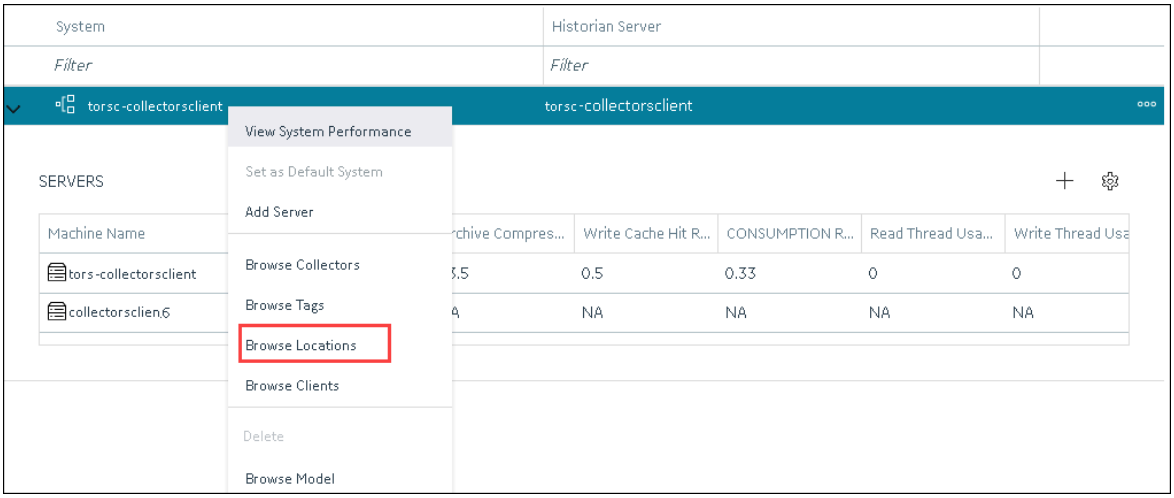



The **Add Server Machine: <system name>** window appears.

7. Enter the host name or IP address of the mirror node in your organization's network (Mirror Node 1 in the example), and then select **Add**.

   The distributed server is added to the system. A distributed location is added in the server.

8. Right-click Mirror Node 1, and then select **Browse Locations**.




A list of distributed locations in the system appears.

9. Select **Mirror Locations**.

   A list of mirror locations in the system appears.

10. In the upper-right corner of the main section, select +.

    The **Add Mirror Location** window appears.

11. Provide values as described in the following table.

| Field | Description |
|---|---|
| **MIRROR LOCATION NAME** | Enter a name for the mirror location. A value is required and must be unique for the system. |

| Field | Description |
|---|---|
| **SERVER MACHINES** | Select the servers that you want to add to the mirror group (Mirror Server 1 and Mirror Node 1 in this example). This box contains a list of all the servers in the system. You must add minimum two servers to a mirror location. |

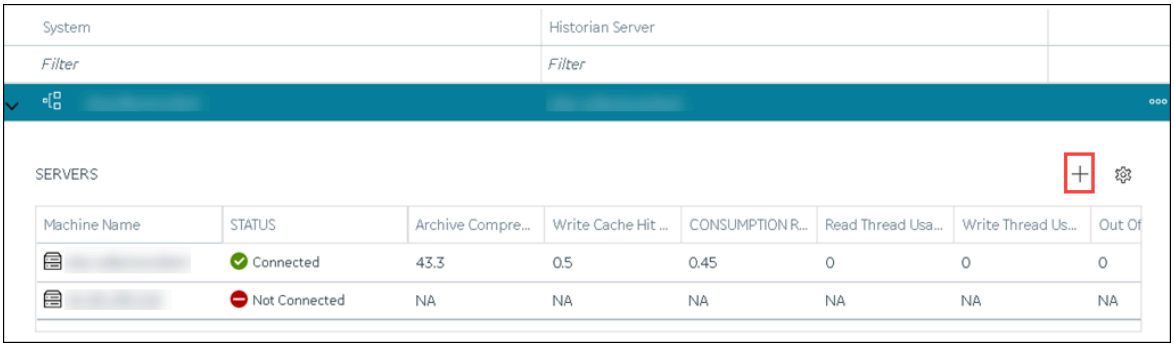
12. Select **Add**.
    Mirror Node 1 is created.
13. Right-click the system name, and then select **Add Data Store**.
    The **Add Data Store: Mirror Node 1** window appears.
14. Enter values as described in the following table.

| Field | Description |
|---|---|
| **DATA STORE NAME** | Enter a unique name for the data store. A value is required. You can use all alphanumeric characters and special characters except / \ * ? < > \|<br><br>You must provide the same name for the mirror setup in the IT network (mirror 2 in the example). |
| **DESCRIPTION** | Enter a description for the data store. |
| **Set as default data store for the system** | Select this check box if you want to set this data store as the default one. A default data store is the one that is considered if you do not specify a data store while adding a tag. You can set only one data store as default. |

15. Select **Add**.
    Mirror 1 is configured.

**Set up Mirror 2:**

16. On the mirror primary server in the IT network (Mirror Server 2 in the example), access Configuration Hub *(on page 336)*.
17. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Systems**.
    A list of systems appears in the main section.
18. Expand Mirror Server 2.
    A list of servers in the system appears.
19. Select +.

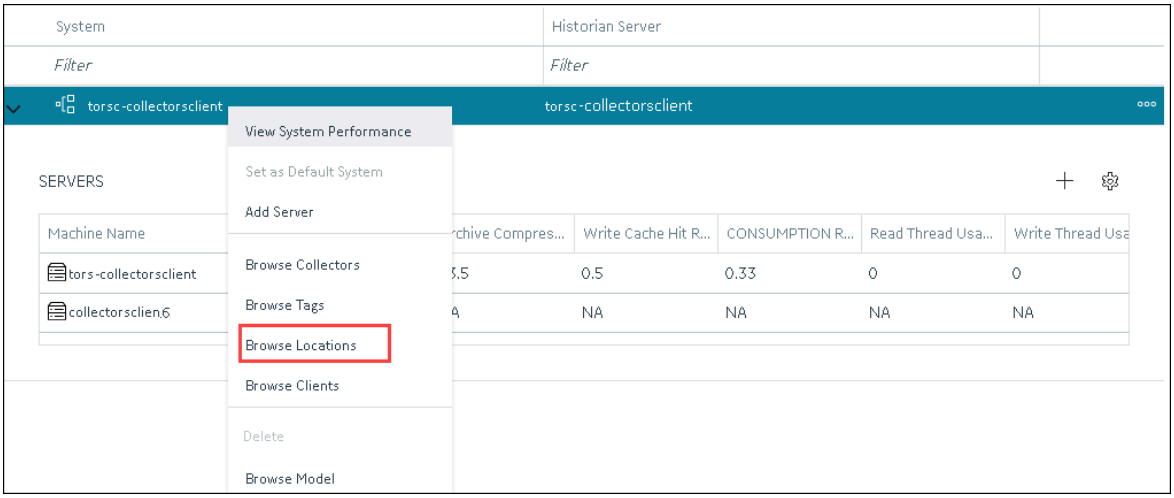



The **Add Server Machine: <system name>** window appears.

20. Enter the host name or IP address of the mirror node in your organization's network (Mirror Node 2 in the example), and then select **Add**.

    The distributed server is added to the system. A distributed location is added in the server.

21. Right-click the system name, and then select **Browse Locations**.




A list of distributed locations in the system appears.

22. Select **Mirror Locations**.

    A list of mirror locations in the system appears.

23. In the upper-right corner of the main section, select +.

    The **Add Mirror Location** window appears.

24. Provide values as described in the following table.

| Field | Description |
|---|---|
| **MIRROR LOCATION NAME** | Enter a name for the mirror location. A value is required and must be unique for the system. |

| Field | Description |
|---|---|
| **SERVER MACHINES** | Select the servers that you want to add to the mirror group (Mirror Server 2 and Mirror Node 2 in this example). This box contains a list of all the servers in the system. You must add minimum two servers to a mirror location. |

25. Select **Add**.
    Mirror Node 2 is created.
26. Right-click Mirror Node 2, and then select **Add Data Store**.
    The **Add Data Store: Mirror Node 2** window appears.
27. Enter values as described in the following table.

| Field | Description |
|---|---|
| **DATA STORE NAME** | Provide the same name that you provided while setting up mirror 1. |
| **DESCRIPTION** | Enter a description for the data store. |
| **Set as default data store for the system** | Select this check box if you want to set this data store as the default one. A default data store is the one that is considered if you do not specify a data store while adding a tag. You can set only one data store as default. |

28. Select **Add**.
    Mirror 2 is configured.

**Set up Mirror of Mirror:**

29. Access Configuration Hub in the primary server in the OT network (Mirror Server 2).
30. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Systems**.
    A list of systems appears in the main section.
31. Expand Mirror Server 1.
    A list of servers in the system appears.
32. Select +.
    The **Add Server Machine: <system name>** window appears.
33. Enter the host name or IP address of the mirror server in the IT network (Mirror Server 2 in the example), select the **Set as Mirror of Mirror** check box, and then select **Add**.
    The distributed server is added to the system. A distributed location is added in the server.
34. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Systems**.

A list of systems appears in the main section.

35. Expand Mirror Server 1.
    A list of servers in the system appears. In the example, Mirror Server 1, Mirror Node 1, and Mirror Server 2 appear.

A mirror of mirror is configured with one primary node and one mirror node each in the OT and IT networks. As needed, you can add more mirror nodes in each network.

## Remove a Distributed/Mirror Server

- Delete the data stores *(on page 2884)* in the machine (using the Web Admin console).
- If the machine is added to a mirror location, remove it from the location *(on page 449)*.

You cannot delete the default server in a system.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Systems**.
   A list of systems appear in the main section.
3. Expand the system from which you want to remove a distributed/mirror server.
   A list of servers in the system appears.
4. In the row containing the server that you want to remove, select ⋯, and then select **Delete Server**.
   A message appears, asking you to confirm that you want to remove the distributed machine from the system.
5. Select **Delete**.
   The machine is removed from the system.

## Set a Default Location

A default location is a server in a system which is considered when you do not specify a location while adding a data store *(on page 370)*. By default, the distributed location in the primary server is the default location. You can, however, set a different default location. The following conditions apply when you set a default location:

- You can have only one default location in a system.
- You cannot delete a default location.
- You can set any of the distributed/mirror locations as default.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Systems**.
   A list of systems appears in the main section.
3. Select the system in which you want to set a default location.
   The details of the system appear in the **DETAILS** section.
4. Expand **System Defaults**, and then next to **Default Location**, select .
   The **Default Location: <system name>** window appears. The **Location** box contains a list of all the servers in the system.
5. Select the location that you want to set as default, and then select **Set as Default**.
   The location is set as default.

## Modify a Historian System

You can change the following details of a system:

- Name
- Description
- Default data store
- Default location (in case of a horizontally scalable system)

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, expand **Systems**, and then select the system that you want to modify.
   The details of the system appear in the **DETAILS** section.
3. Modify values as specified in the following table.

| Field | Description |
|---|---|
| **Name** | Enter a name for the Historian system. This value must be unique for a user. |
| **Description** | Enter a description for the system. |

4. If you want to change the default data store:
   a. Under **System Defaults**, next to **Default Data Store**, select .
      The **Default Data Store: <system name>** window appears.
   b. Select the data store that you want to set as default, and then select **Set as Default**.

The default data store is changed.

5. If you want to change the default location:

    a. Under **System Defaults**, next to **Default Location**, select .
    The **Default Location: <system name>** window appears. The **Location** box contains a list of all the servers in the system.

    b. Select the location that you want to set as default, and then select **Set as Default**.

The changes to the system are saved automatically.

## Configure Advanced Settings of a System

You can now configure a few advanced settings for the Archiver, Collector, and Data Store to achieve some specific functionality in Historian. You must be careful while modifying the configuration as this might impact the stability and security of the Historian System.

Ensure that you are a member of the iH Security Admins group.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Systems**.
   A list of systems appears in the main section.
3. Right-click the system whose advanced settings you want to configure, and then select **Advanced Configuration**.
   The advanced settings of the system appear, displaying the **Server** section by default.
4. Enter values as described in the following table.

**Note:**
If you have changed the values of any * marked fields, restart the Historian Data Archiver service. Only then will your changes be reflected.

| Field | Description |
|---|---|
| **ALARM TIMESTAMP CHECK** | Specify whether you want to speed up, slow down, or disable alarm timestamp checking: |

| Field | Description |
|---|---|
| | ◦ **0 - Disabled**: Select this option if you want to disable alarm timestamp checking. Only the data is sent to the alarm archiver.<br>◦ **1 - Fast**: Select this option if you want to check the time-stamp of the alarm data for the last 10 minutes.<br>◦ **2 - Slow**: Select this option if you want to check the time-stamp of the alarm data beyond the last 10 minutes. |
| **ALLOW DATA OVERWRITES** | Switch the toggle to enable or disable overwriting data. |
| **ARCHIVER MEMORY SIZE** | Enter the memory usage in MB that you want to allocate to an archive. If you enter 0, Data Archiver will dynamically allocate the memory usage. If Data Archiver is running on a 32-bit operating system, you can allocate upto 1800 MB. If Data Archiver is running on a 64-bit operating system, we recommend that you use the default value. |
| **BUFER MEMORY MAX** | Enter the maximum memory buffer size in MB that an archiver queue can use before switching to disk buffer. |
| **COLLECTOR IDLE TIME (SECONDS)** | Enter the number of seconds of no data collection after which a collector is considered idle. |
| **DEBUG MODE** | Specify whether you want to enable or disable the debug mode. If you enable this option, the debug information is included in the Historian log files, which helps you troubleshoot issues. However, this can result in large size of the log files. |
| **DISABLE CA HOSTING** | Enable CA Hosting to continue using the Client Access API based clients when you have converted a Mirror Primary to a standalone server while installing the Historian server. |
| **DO NOT ZIP ARCHIVES** | Switch the toggle to store the archive files as .zip files. This will help optimize the storage space of the archive files. However, if you are collecting alarm data as well, the alarm data may not be backed up. Also, you cannot export alarm data to another Data Archiver. Therefore, exercise caution while enabling this option if you are collecting alarm data as well. |
| **FIREWALL PERMISSION** | Use this parameter to disable port 14000 in the firewall. By default, Historian installation enables port 14000 in the firewall. |

| Field | Description |
|---|---|
| **MAINTAIN AUTO RECOVERY FILE** | Switch the toggle if you want to back up the archive and configuration files (.iha and .ihc files) every hour. This will prevent data loss. However, these files are used by Historian internally. Also, exercise caution in enabling this option because it can impact Historian performance. |
| **MAX QUERY TIME (SECONDS)** | Enter the maximum time in seconds that a query can take to process. After this time limit exceeds, the query is terminated. |
| **MAXIMUM QUERY INTERVALS** | Enter the maximum number of samples per tag that Historian can return from a query on non-raw data. You can use this setting to limit the number of query results on non-raw data. |
| **NUMBER OF READ THREADS** | Enter the number of read threads to use parallel reading of data. The minimum number you can enter is 8. |
| **NUMBER OF WRITE THREADS** | Enter the number of write threads to use parallel writing of data. |
| **USE ADSI CALLS** | Switch the toggle to allow Data Archiver to use Active Directory Service Interfaces (ADSI) calls. |

5. Select **Save**.
6. Expand **Collector**, and then select the collector whose settings you want to configure.
   The fields specific to the selected collector appear.
7. Enter values as described in the following table.

| Field | Description |
|---|---|
| **BUFFER FLUSH MULTIPLIER** | Select the multiplier to the buffer flow speed while using the store-and-forward feature:<br>◦ **0**: Select this option if you want to disable throttling.<br>◦ **1**: Select this option if you want normal speed.<br>◦ **2**: Select this option if you want the collector to never send data faster than twice the normal speed. |
| **NUM INTERVALS FLUSH** | Specify how quickly you want the collector to send data to Data Archiver. The value you enter in this field is multiplied by 100 milliseconds. For example, if you enter 5, the collector sends data to Data Archiver every 500 milliseconds. We recommend that you enter 5. |

8. Select **Save**.
   A message appears, asking you whether you want to save and restart the collector as well.
9. If you want to save your changes and restart the collector as well, select **Save and Restart**. If you want to just save your changes, select **Save**. In that case, you must restart the collector later for the changes to reflect.
   Your changes are saved. If you have selected **Save and Restart**, the collector is restarted.
10. Expand **Data Store**, and then, select the data store whose settings you want to configure.
    The fields specific to the selected data store appear,
11. Enter values as described in the following table.

| Field | Description |
|---|---|
| **ALLOW FUTURE DATA** | Switch the toggle to enable storing future data *(on page 1037)*. |
| **CREATE OFFLINE ARCHIVES** | Switch the toggle to create offline archives. This is to avoid receiving an outside-active-hours error. It happens if you attempt to store data when the current archive file is set to read-only. |

12. Select **Save**.
    The advanced settings are configured for the data store.

## Configure Labels of Spare Fields

For each tag, a set of five spare fields are available, which are named Spare 1 to Spare 5. You can use these fields to enter values for any tag details apart from those captured in the tag fields (for example, the name, location, and phone number of the manufacturer of a device).

This topic describes how to change the label of these spare fields at the system level. The changes are then cascaded to all the tags in the system.

You can configure any or all of these fields. The new labels of the fields then appear under **SPARE FIELDS** in the **DETAILS** section when you access a tag.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, expand **Systems**, and then select the system that you want to modify.
   The details of the system appear in the **DETAILS** section.
3. Right-click the system whose spare fields you want to configure (or select ⋯), and then select **Configure Spare Fields Labels**.
   The **Configure Spare Fields Labels: <system name>** window appears.
4. Enter values in the available fields, and then select **Save**.

For example, if you want to capture the name, location, and phone number of the manufacturer of a device, enter Name, Location, and Phone Number in the **SPARE FIELD 1 LABEL**, **SPARE FIELD 2 LABEL**, and **SPARE FIELD 3 LABEL** fields.
The labels of the spare fields are configured. When you access a tag, the new labels appear under **SPARE FIELDS** in the **DETAILS** section.

> **Note:**
> The labels that you have configured only appear in Configuration Hub. For other applications, such as Historian Administrator, Trend Client, and so on, Spare 1 to Spare 5 are displayed. The values of spare fields can be configured for tags.

## Set a Default System

If you set a system as default, when you log in to Configuration Hub, this system appears by default. The following conditions apply when you set a system as default:

- You can have only one default system in Configuration Hub.
- You cannot delete a default system.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Systems**.
   A list of systems appears in the main section.
3. Right-click the system that you want to set as default (or select ⋯), and then select **Set as Default System**.
   The system is set as default.

## Delete a Historian System

You can delete a Historian system if you no longer want to manage it using Configuration Hub. You cannot, however, delete a system if it is set as default.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Systems**.
   A list of systems appears in the main section.
3. Right-click the system that you want to delete (or select ⋯), and then select **Delete**.
   A message appears, asking if you want to delete the system.
4. Select **Delete**.
   The system is deleted.

# Managing Mirror Locations

## Create a Mirror Location

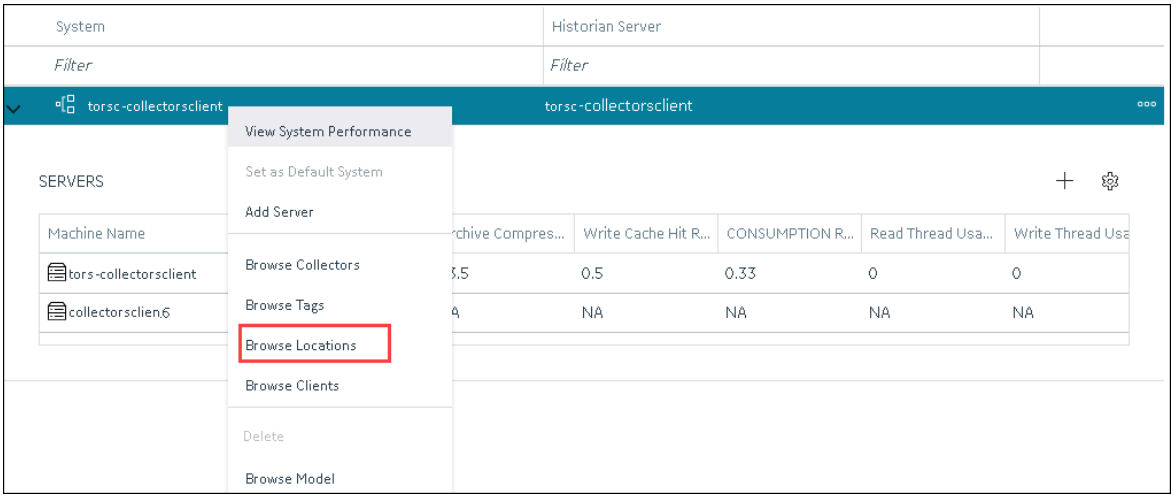
Add one or more distributed servers *(on page 361)* to the system in which you want to create a mirror group.

If you want high availability of one or more data stores, you must create a mirror group (also called a mirror location), and then add servers to it. When you do so, the data in the data stores of the mirror locations is replicated. Therefore, even if one of the servers is down, you can retrieve data from the other servers in the mirror location, thus achieving high availability.

The following conditions apply when you create a mirror location:

- You must add minimum two servers to a mirror location. The maximum number of servers that you can add depends on your Historian license.
- You can add a mirror location only in a horizontally scalable Historian system.
- You can rename a mirror location, remove a machine from a mirror location, or add an additional one even after you create the mirror location. However, if only one machine remains in the group, you cannot remove it.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Systems**. The **Systems** section appears, displaying a list of systems.
3. Right-click the system in which you want to create a mirror location (or select ⋯), and then select **Browse Locations**.




A list of distributed locations in the system appears.

4. Select **Mirror Locations**.
   A list of mirror locations in the system appears.
5. In the upper-right corner of the main section, select +.
   The **Add Mirror Location** window appears.
6. Provide values as described in the following table.

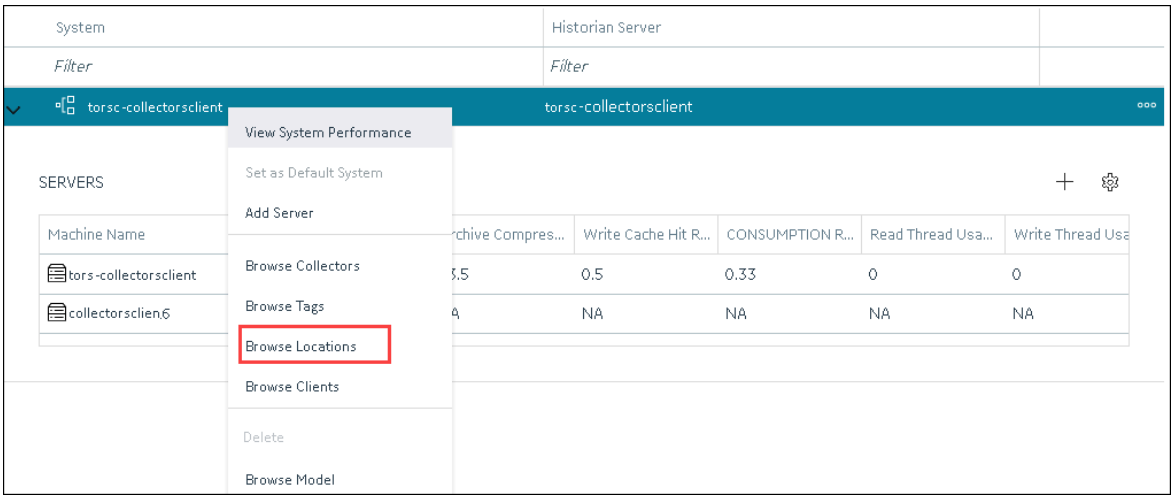
| Field | Description |
|---|---|
| **MIRROR LOCATION NAME** | Enter a name for the mirror location. A value is required and must be unique for the system. |
| **SERVER MACHINES** | Select the servers that you want to add to the mirror group. This box contains a list of all the servers in the system. You must add minimum two servers to a mirror location. |

7. Select **Add**.
   The mirror location is created.

Add a data store to the mirror location *(on page 370)*.

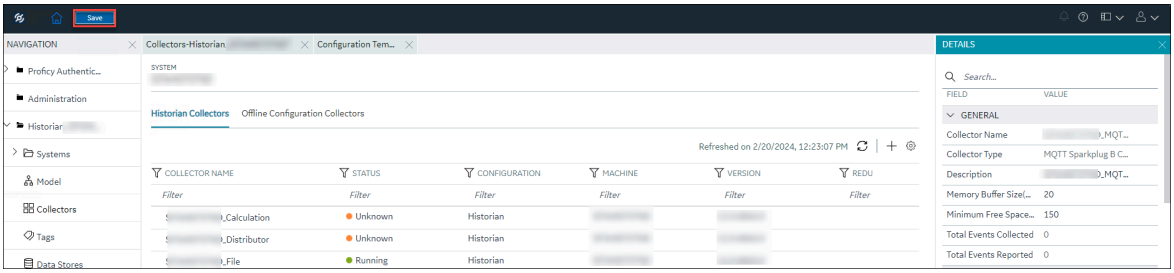
## Rename a Mirror Location

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Systems**.
   The **Systems** section appears, displaying a list of systems.
3. Right-click the system in which you want to rename a mirror location (or select ⋯), and then select **Browse Locations**.



A list of distributed locations in the system appears.

4. Select **Mirror Locations**.
   A list of mirror locations in the system appears.
5. Select the location that you want to rename.
   The details of the mirror location appear in the **Details** section.
6. In the **Name** field, enter the new name of the mirror location.
7. In the upper-left corner of the page, select **Save**.



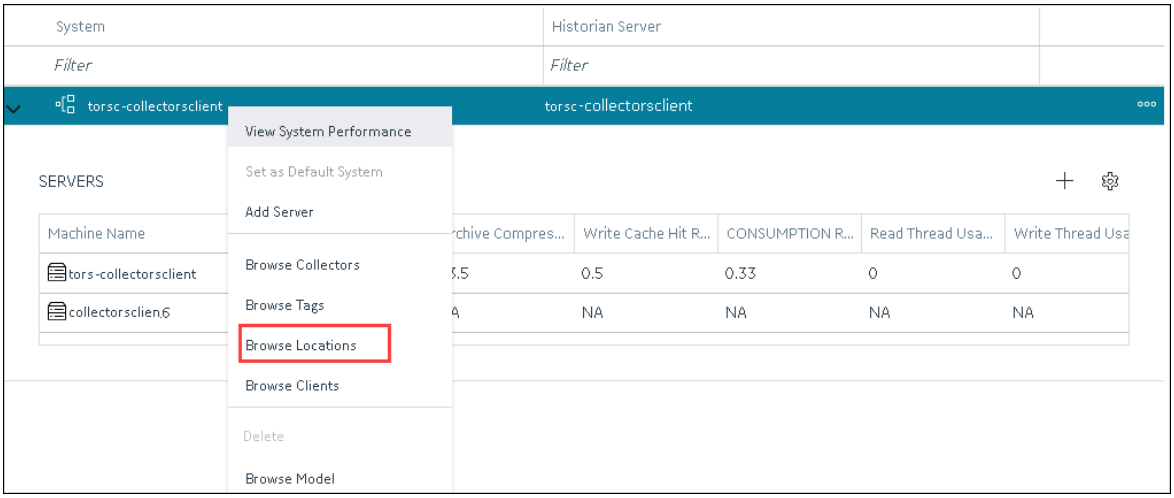

The mirror location is renamed.

## Add a Machine to a Mirror Location

If you want to add machine to a mirror location that already contains machines,and if you want to copy the archive and configuration information from the existing machines to the new machine, perform the following steps:

1. Copy the archive files and configuration files from an existing machine in the mirror location to the one that you have added.
2. Rename the configuration file `<machine name>_Config.ihc`.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Systems**.
   The **Systems** section appears, displaying a list of systems.
3. Right-click the system to which you want to add a machine (or select ⋯), and then select **Browse Locations**.



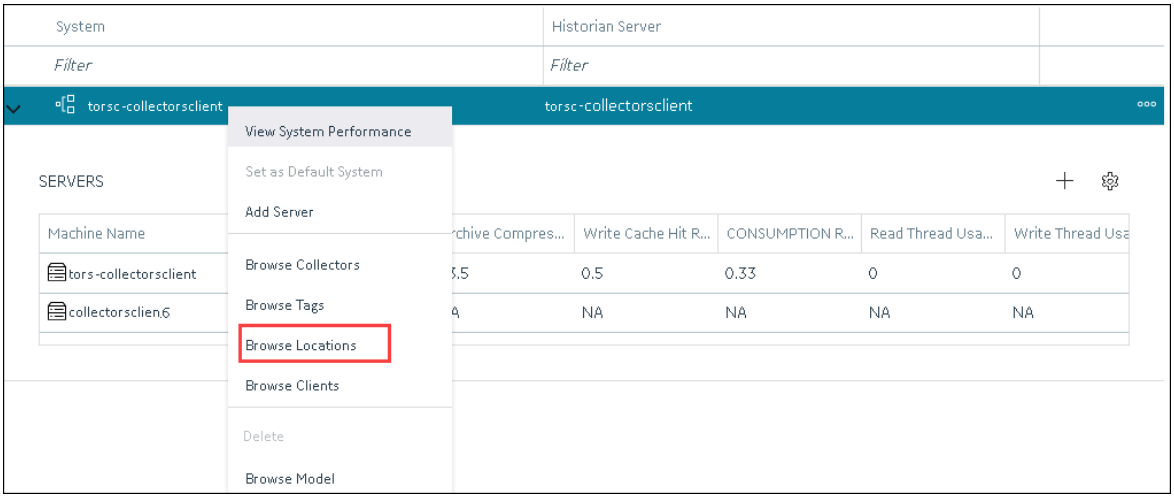

A list of distributed locations in the system appears.

4. Select **Mirror Locations**.
   A list of mirror locations in the system appears.
5. Right-click the mirror location in which you want to add a machine (or select ⋯), and then select **Add Server Machine**.
   The **Add Machine: <mirror location>** window appears. The **SERVER MACHINES** field contains a list of machines in the system that are not yet added to the mirror location.
6. In the **SERVER MACHINES** field, select the machine that you want to add to the mirror location, and then select **Add**.
   The machine is added to the mirror location.

## Remove a Machine from a Mirror Location

If a mirror location contains only one machine, you cannot remove it.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Systems**.
   The **Systems** section appears, displaying a list of systems.
3. Right-click the system from which you want to remove a machine (or select ⋯), and then select **Browse Locations**.



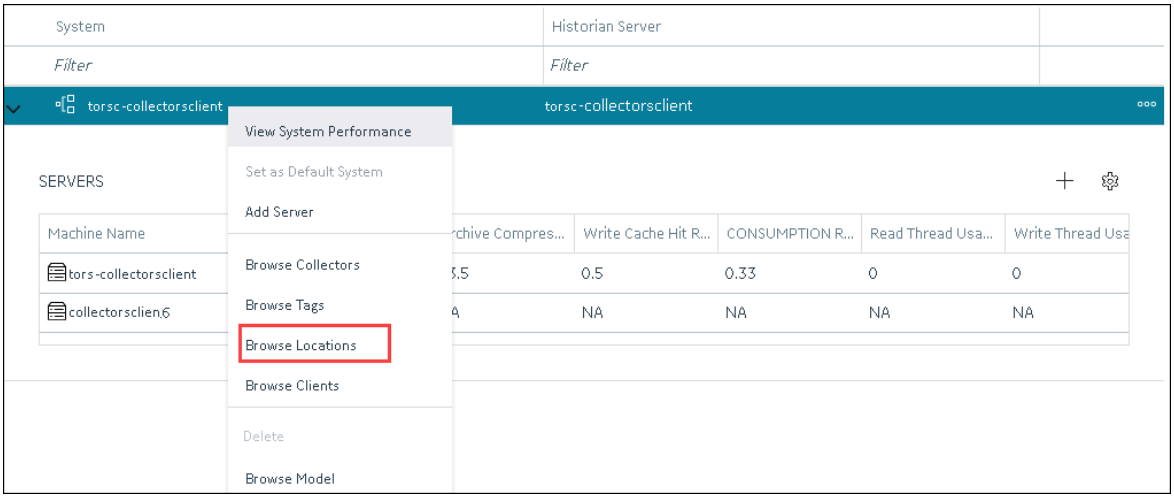

A list of distributed locations in the system appears.

4. Select **Mirror Locations**.
   A list of mirror locations in the system appears.
5. Right-click the mirror location from which you want to remove a machine (or select ⋯), and then select **Remove Server Machine**.
   The **Remove Server Machine: <mirror location>** window appears, displaying a list of machines in the mirror location.
6. Select the machine that you want to remove, and then select **Remove**.
   A message appears, asking you to confirm that you want to remove the machine from the mirror location.
7. Select **Remove**.
   The machine is removed from the mirror location.

## Delete a Mirror Location

Delete all the data stores in the mirror location; you cannot delete a mirror location if it contains a data store.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Systems**.
   The **Systems** section appears, displaying a list of systems.
3. Right-click the system in which you want to delete a mirror location (or select ⋯), and then select **Browse Locations**.




A list of distributed locations in the system appears.

4. Select **Mirror Locations**.
   A list of mirror locations in the system appears.
5. Right-click the mirror location that you want to delete (or select ⋯), and then select **Delete Mirror Location**.
   A message appears, asking you to confirm that you want to delete the mirror location.
6. Select **Delete**.
   The mirror location is deleted.

# Managing Data Stores

## About Data Stores

A data store is a logical collection of tags. It is used to store, organize, and manage tags according to the data source and storage requirements. A data store can have multiple archive files (*.IHA), and includes both logical and physical storage definitions.

Tags can be segregated into separate archives through the use of data stores. The primary use of data stores is to segregate tags by data collection intervals. For example, you can put a name plate or static tags where the value rarely changes into one data store, and your process tags into another data store. This can improve query performance.

Historian data stores are stored as archive files that contain data gathered from all data sources during a specific period of time. You can write and read data from the archive files.

You can define two types of data stores:

- **Historical Data Store**: Tags stored under historical data store will store data as long as the disk space is available. Depending on your license, you may be able to create multiple historical data stores. The maximum number of historical data stores supported depends on the license.
- **SCADA Buffer Data Store**: Tags stored under the SCADA buffer data store will store data for a specific duration of time based on license.

When you install the Historian server, two historical data stores are installed by default.

- **System**: Stores Historian messages and performance tags. This is only for internal usage within Historian, and you cannot add tags to this data store. You cannot rename or delete the system data store.
- **User**: Stores tag data. This is a default data store. You can rename and delete a user data store as long as there is another default data store set for tag addition.

Based on your license, a SCADA Buffer data store may also be installed. It stores short-term tags and data.

## Create a Data Store

The number of data stores that you can create depends on your license.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Data Sores**. Alternatively, you can select **Systems**, right-click the system in which you want to create a data store (or select ⋯), and then select **Browse Data Stores**.
   The **Data Stores** section appears.
3. Select +.
   If Historian Standard version, then the **Add Data Store** window appears.

   If Historian Enterprise version, then the **Add Data Store: <location name>** window appears.
4. Enter values as described in the following table.

| Field | Description |
|---|---|
| **DATA STORE NAME** | Enter a unique name for the data store. A value is required. You can use all alphanumeric characters and special characters except / \ * ? < > \| |
| **DESCRIPTION** | Enter a description for the data store. |

| Field | Description |
|---|---|
| **LOCATION** | Enter the host name or IP address of the distributed location on which you want to create the data store. This field is available only for a horizontally scalable system. |
| **Is Default** | Switch the toggle on if you want to set this data store as the default one. A default data store is the one that is considered if you do not specify a data store while adding a tag. You can set only one data store as default. |

5. Select .
   The data store is created.

   When you add tags to the data store, it will have its own set of .IHA (iHistorian Archive) files. Ensure that you back up the new data store archives periodically.

## Access a Data Store

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Data Stores**. Alternatively, you can select **Systems**, right-click the system in which the data store is available (or select ⋯), and then select **Browse Data Stores**.
   The **Data Stores** section appears.
3. Select the data store that you want to access.
   The **DETAILS** section displays the following details of the data store.

**Table 19. General**

| Field | Description |
|---|---|
| **Data Store Name** | The name of the data store. A value is required and must be unique for the system. |
| **Description** | The description of the data store. |
| **State** | The current state of the data store:<br>◦ Running: Indicates that the data store is actively storing data.<br>◦ Stopped: Indicates that the data store is not storing data.<br>This field is read-only. |

| Field | Description |
|---|---|
| **Location** | The host name or IP address of the distributed location on which the data store has been created. This field is available only for a horizontally scalable system. It is disabled. |
| **Storage Type** | The storage type of the data store, which can be one of the following values:<br>◦ **Historical:** Tags stored in a historical data store will store data as long as disk space is available. The maximum number of Historical data stores supported depends on the license.<br>◦ **SCADA Buffer:** Tags stored under SCADA buffer data store will store data for a specific duration of time based on license.<br>This field is disabled. |
| **System Default Storage** | Indicates whether the data store is a default one. If yes, while creating a tag, this data store will be used by default. |
| **Number of Tags** | The number of tags in the data store. For instructions on how to add a tag, refer to Add Tags for the Data Store Using Configuration Hub *(on page 357)* and Add a Tag Manually *(on page 473)*. |

**Table 20. Archive Creation**

| Field | Description |
|---|---|
| **Create Archive By** | Indicates whether you want to create a new archive automatically after the current one reaches a specific size or after a specific duration. This field is enabled only if you switch the **Automatically Create Archives** toggle on.<br><br>Select one of the following options:<br>◦ **Size**: Select this option if you want to create a new archive when the current one reaches a specific size. Specify the size in the **Default Size (MB)** field (which appears only if you select **Size**). |

| Field | Description |
|---|---|
| | ◦ **Days** or **Hours**: Select one of these options if you want to create a new archive after a specific duration. Specify the duration in the **Archive Duration** field (which appears only if you select **Days** or **Hours**). |
| **Default Size (MB)** | The default size of an archive after which a new one will be automatically created if you switch the **Automatically Create Archives** toggle on. The **Default Size (MB)** field appears only if you select **Size** in the **Create Archive By** field. |
| **Automatically Create Archives** | Indicates whether you want to create an archive automatically *(on page 727)* after the current one is full. An archive file is considered full based on the size or duration you specify in the **Create Archive By** and the **Archive Duration** or **Default Size** fields. |
| **Overwrite Old Archives** | Indicates whether you want to overwrite an old archive file when a new one is created.<br><br>If you enable this option, the oldest archived data is replaced with the latest one when the latest archive default size is reached. Since this action deletes historical data, exercise caution in using this feature. Be sure that you have a backup of the archive so that you can restore it later. Best practice is to create an additional archive to prevent premature loss of data due to overwriting. For example, if you want to save 12 months of data into 12 archives, create 13 archives. |
| **Archive Duration** | The duration after which a new archive will be automatically created if you switch the **Automatically Create Archives** toggle on. The **Archive Duration** field appears only if you select **Days** or **Hours** in the **Create Archive By** field. |
| **Multiple Archive Paths** | Indicates whether you want to create multiple archive paths for the selected data store. This field allows you to back up your high-volume archives to another storage path, for example, an external storage disk. For more information, refer to about multiple archive path *(on page 464)* and configure multiple archive paths *(on page 468)*. |

**Table 21. Maintenance**

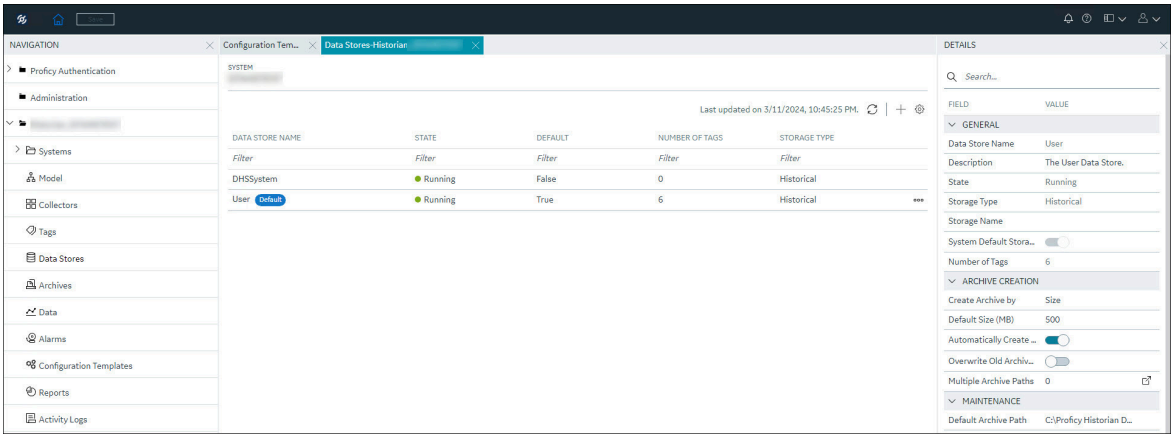
| Field | Description |
|---|---|
| **Default Archive Path** | The default folder in which you want to create archives. |
| **Default Backup Path** | The default folder in which you want to place the backup archives. |
| **Base Archive Name** | A prefix that you want to add to all the archive name. |
| **Base Archive Filename** | A prefix that you want to add to all the archive filenames. |
| **Free Space Required (MB)** | Indicates the remaining disk space required after a new archive is created. If the available space is less than the requirement, a new archive is not created. The default value is 5000 MB.<br><br>This field is not applicable to alarms and events archives. The alarms and events archiver will continue writing to the alarms and events archive until the drive is full. If this occurs, the alarms and events archiver will buffer incoming alarms and events data until the drive has free space. An error message is logged in the Historian message log. |
| **Store OPC Quality** | Indicates whether OPC data quality is stored. |
| **Use Caching** | Indicates whether caching is enabled. When reading data from the archiver, some data is saved in the system memory and retrieved using caching. This results in faster retrieval as the data is already stored in the buffer. |
| **Stale Period (Days)** | Indicates the number of days after which the data store is considered stale. |
| **Stale Period Check (Days)** | Indicates the number of days or the frequency for checking data store validity. |

**Table 22. Security**

| Field | Description |
|---|---|
| **Data is Read-Only After (Hours)** | The number of hours for data to be stored in a read/write archive. After the time lapses, that portion of the archive file is automatically made read-only. Incoming data values with timestamps prior to this time are rejected. A single archive file, there- |

| Field | Description |
|---|---|
| | fore, may have a portion made read-only, another portion that is read/write containing recently written data, and another that is unused free space. |
| **Generate Message on Data Update** | Indicates whether an audit log entry will be made any time the value of a previously archived data point is overwritten. This log entry will contain both the original and new values. |
| **Read Group** | The Windows security group that can retrieve the tag data and plot it in a trend chart for the selected data store.<br><br>For example, if you select a group with power users, in addition to members of the iH Security Admins group, only a member of the power users group will be able to read data of the tags for that data store. Even a member of the iH Readers group will not be able to access data of the tags for the selected data store, unless they are also defined as a member of the power users group. |
| **Write Group** | The Windows security group that can write tag data for the selected data store (for example, using the Excel Add-in for Historian). |
| **Administer Group** | The Windows security group that can create, modify, and delete the tags for the selected data store. |
| For more information, refer to implementing Data store-level security *(on page 292)*.<br><br>**Note:**<br>When it comes to the group security, the security settings applied at the tag level, if any, take the precedence over those at the data store level. | |

4. As needed, modify values in the available fields.
5. In the upper-left corner of the page, select **Save**.




The data store is modified.

## Rename a Data Store

You cannot rename the system data store. You can rename a user data store as long as there is another default data store set for tag addition.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Data Stores**.
   Alternatively, you can select **Systems**, right-click the system in which the data store is available (or select ), and then select **Browse Data Stores**.
   The **Data Stores** section appears.
3. Select the data store that you want to access.
   The details section displays the following details of the data store.

**Table 23. General**

| Field | Description |
|---|---|
| **Data Store Name** | The name of the data store. A value is required and must be unique for the system. |
| **Description** | The description of the data store. |
| **State** | The current state of the data store (whether it is running). This field is disabled. |
| **Location** | The host name or IP address of the distributed location on which the data store has been created. This field is available only for a horizontally scalable system. It is disabled. |

| Field | Description |
|---|---|
| **Storage Type** | The storage type of the data store, which can be one of the following values:<br>◦ **Historical:** Tags stored in a historical data store will store data as long as disk space is available. The maximum number of Historical data stores supported depends on the license.<br>◦ **SCADA Buffer:** Tags stored under SCADA buffer data store will store data for a specific duration of time based on license.<br>This field is disabled. |
| **System Default Storage** | Indicates whether the data store is a default one. If yes, while creating a tag, this data store will be used by default. |
| **Number of Tags** | The number of tags in the data store. For instructions on how to add a tag, refer to Add Tags for the Data Store Using Configuration Hub *(on page 357)* and Add a Tag Manually *(on page 473)*. |

**Table 24. Archive Creation**

| Field | Description |
|---|---|
| **Automatically Create Archives** | Indicates whether you want to create a new archive *(on page 723)* automatically after the current one is full. An archive file is considered full based on the size or duration you specify in the **Create Archive By** and the **Archive Duration** or **Default Size** fields. |
| **Overwrite Old Archives** | Indicates whether you want to overwrite an old archive file when a new one is created.<br><br>If you enable this option, the oldest archived data is replaced with the latest one when the latest archive default size is reached. Since this action deletes historical data, exercise caution in using this feature. Be sure that you have a backup of the archive so that you can restore it later. Best practice is to create an additional archive to prevent premature loss of data due to |

| Field | Description |
|---|---|
| | overwriting. For example, if you want to save 12 months of data into 12 archives, create 13 archives. |
| **Create Archive By** | Indicates whether you want to create a new archive automatically after the current one reaches a specific size or after a specific duration. This field is enabled only if you switch the **Automatically Create Archives** toggle on.<br><br>Select one of the following options:<br>◦ **Size**: Select this option if you want to create a new archive when the current one reaches a specific size. Specify the size in the **Default Size (MB)** field (which appears only if you select **Size**).<br>◦ **Days** or **Hours**: Select one of these options if you want to create a new archive after a specific duration. Specify the duration in the **Archive Duration** field (which appears only if you select **Days** or **Hours**). |
| **Default Size (MB)** | The default size of an archive after which a new one will be automatically created if you switch the **Automatically Create Archives** toggle on. The **Default Size (MB)** field appears only if you select **Size** in the **Create Archive By** field. |
| **Archive Duration** | The duration after which a new archive will be automatically created if you switch the **Automatically Create Archives** toggle on. The **Archive Duration** field appears only if you select **Days** or **Hours** in the **Create Archive By** field. |

**Table 25. Maintenance**

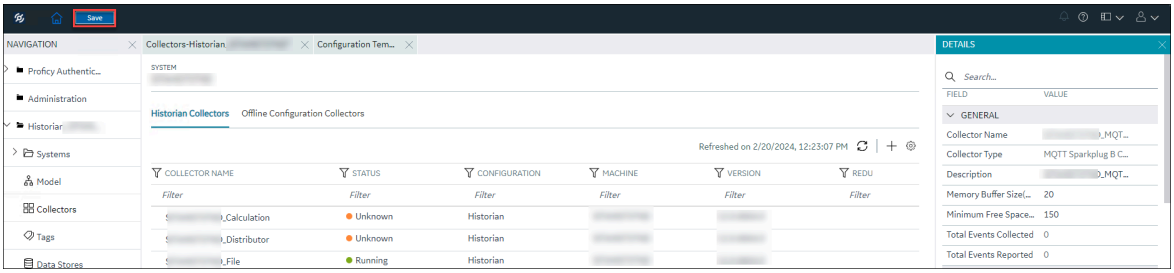
| Field | Description |
|---|---|
| **Default Archive Path** | The default folder in which you want to create archives. |
| **Default Backup Path** | The default folder in which you want to place the backup archives. |
| **Base Archive Name** | A prefix that you want to add to all the archive files. |
| **Free Space Required (MB)** | Indicates the remaining disk space required after a new archive is created. If the available space is less than the requirement, a new archive is not created. The default value is 5000 MB. |

| Field | Description |
|---|---|
| | This field is not applicable to alarms and events archives. The alarms and events archiver will continue writing to the alarms and events archive until the drive is full. If this occurs, the alarms and events archiver will buffer incoming alarms and events data until the drive has free space. An error message is logged in the Historian message log. |
| **Store OPC Quality** | Indicates whether OPC data quality is stored. |
| **Use Caching** | Indicates whether caching is enabled. When reading data from the archiver, some data is saved in the system memory and retrieved using caching. This results in faster retrieval as the data is already stored in the buffer. |

**Table 26. Security**

| Field | Description |
|---|---|
| **Data is Read-Only After (Hours)** | The number of hours for data to be stored in a read/write archive. After the time lapses, that portion of the archive file is automatically made read-only. Incoming data values with timestamps prior to this time are rejected. A single archive file, therefore, may have a portion made read-only, another portion that is read/write containing recently written data, and another that is unused free space. |
| **Generate Message on Data Update** | Indicates whether an audit log entry will be made any time the value of a previously archived data point is overwritten. This log entry will contain both the original and new values. |

4. As needed, modify values in the available fields.
5. In the upper-left corner of the page, select **Save**.

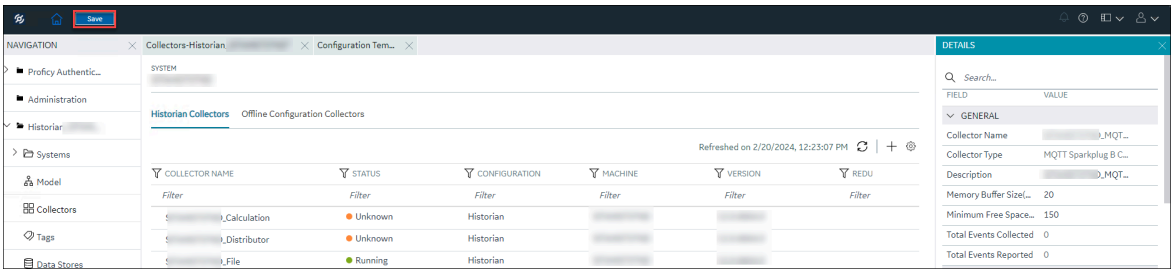



The data store is modified.

## Set a Default Data Store

A default data store is the one that is considered if you do not specify a data store while adding a tag.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Data Stores**. Alternatively, you can select **Systems**, right-click the system in which the data store is available (or select ), and then select **Browse Data Stores**.
3. Select the system whose default data store you want to change.
   The details of the system appear in the **DETAILS** section.
4. Under **System Defaults**, next to **Default Data Store**, select .
   The **Default Data Store: <data store name>** window appears, displaying a list of data stores in the system.
5. Select the data store that you want to set as default, and then select **Set as Default**.
6. In the upper-left corner of the page, select **Save**.




The data store is set as default.

## Access the Archives in a Data Store

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Data Stores**.
   Alternatively, you can select **Systems**, right-click the system in which the data store is available (or select ⋯), and then select **Browse Data Stores**.
   The **Data Stores** section appears.
3. Right-click the data store whose archives you want to access (or select ⋯), and then select **Browse Archives**.
   The archives in the data store appear, indicating the current one and the old ones.

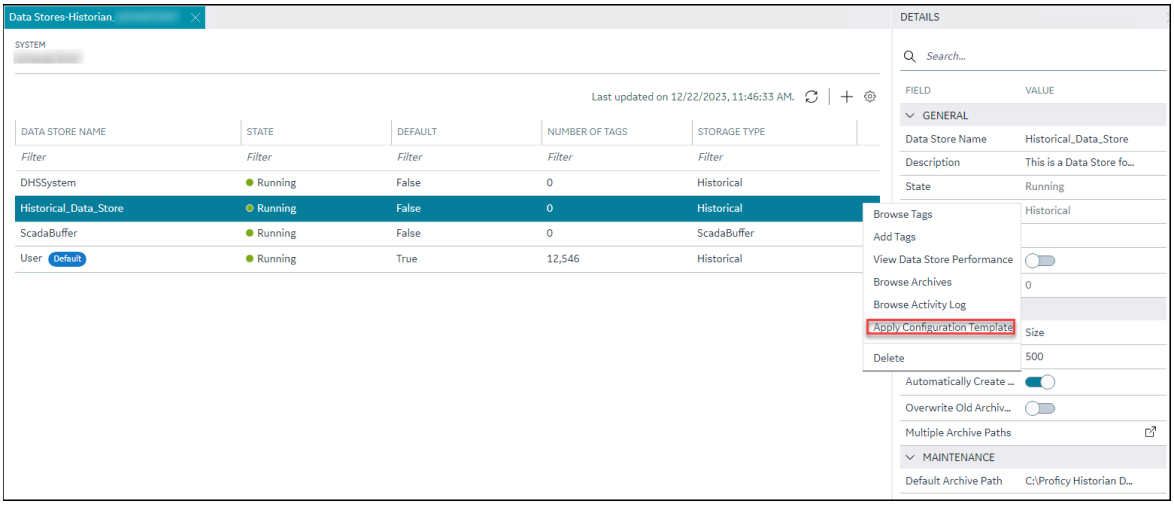
## Apply the Configuration Template to a Data Store

You can apply the created template to user-created data store as needed. You will be prompted to confirm whether you want to overwrite few of the configuration values with the values in the template.

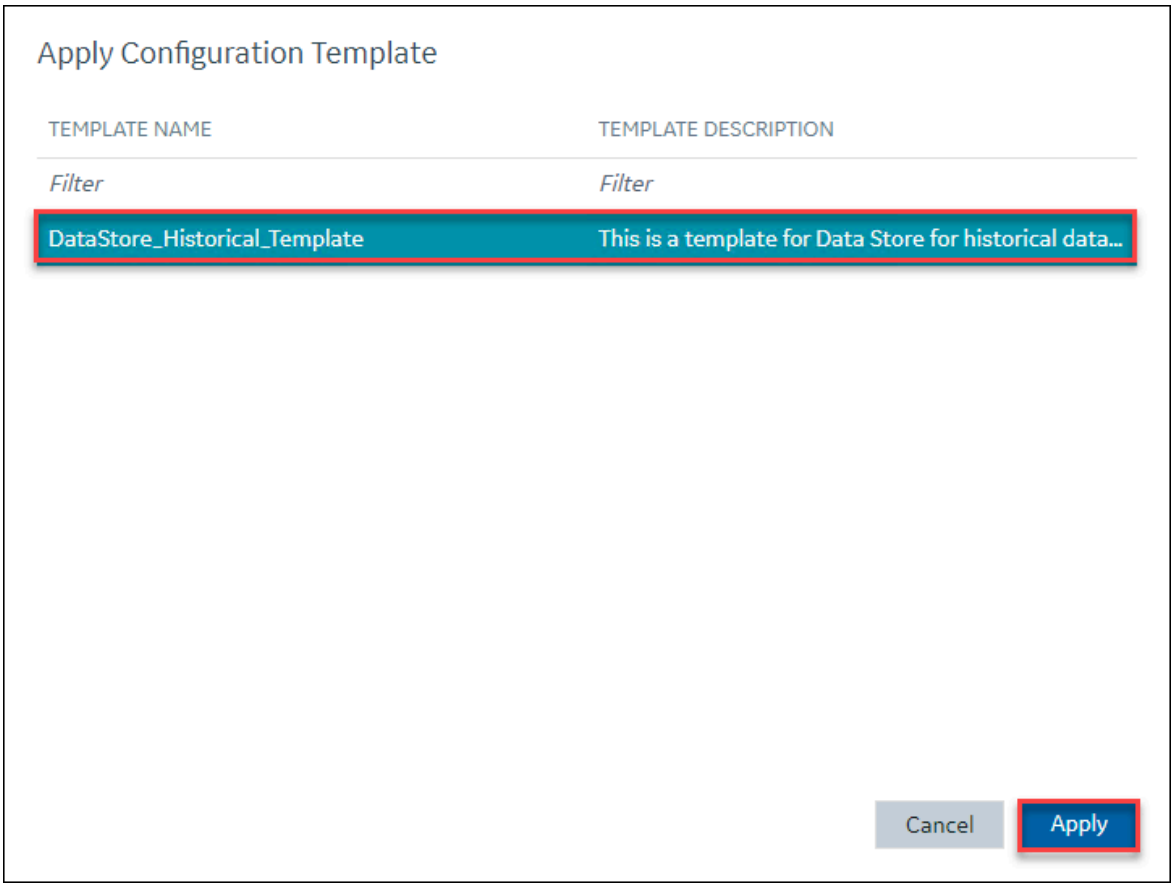
- Ensure that you have a data store created *(on page 370)*.
- Ensure that you have a configuration template for data stores *(on page 767)*.

This topic describes how to apply a data store configuration template to a data store. You can apply a data store configuration template to a user-created data store, provided they are not the default data store.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Data Stores**.
   The **Data Stores** section appears.
3. Right-click the data store (or select ⋯), and then select **Apply Configuration Template**.



The **Apply Configuration Template** window appears, listing the available templates.




**Note:**
You can apply a data store configuration template to a user-created data store, provided they are not the default data store.

4. Select **Apply**.
   A confirmation window appears, prompting you to confirm whether you want to overwrite few of the configuration values with the values in the template.
5. Select **Ok**.
6. In the upper-left corner, select **Save**.
   The configurations in the template are applied to the data store.

# Multiple Archive Paths

## About Multiple Archive Paths

Multiple archive paths feature extends the ability to configure a default archive path for each data store.

This feature enables you to automatically store your archives in different locations based on the age of the archive. It can be useful when splitting the data for a data store across multiple hard drives, primarily for performance and cost optimization. Additionally, it could help optimize disk space constraints.

For instance, in a performance and cost optimization scenario, the most recent data could reside on a fast/expensive NVMe or SSD drive, intermediate data on a slower/cheaper local HDD, and older data on even slower/cheaper long-term storage (for example, NAS or cloud). The requirement for the drive is that it must be mapped to the Historian server node as a drive or folder and accessible by the DataArchiver service. For example, it could be a shared drive such as \\Server\HistorianData mapped to H:\HistorianData. The physical location of the drive does not matter if the DataArchiver service can access it.

**Note:**
There are many sources of drives and various methods to map them. The Historian team has not tested all possibilities and cannot guarantee compatibility with every configuration. However, if the drive can be accessed (with read and write permissions) by the DataArchiver service, it will likely function properly.

**About Multiple Archive Path Configuration**

Configuration for multiple archive paths can be done on a per-data store basis. Additionally, each data store could have different configurations, so you can decide whether you want to configure multiple archive paths for all the data stores or not. When configuring the archive paths, you can specify a path and an age.

Where,

The age defines how far in the past the archive end time must be before it could be moved.

The path is the directory where the archive will be moved to.

Once the end time of the archive becomes older than the current time minus the age, the archive will be moved. Since moving an archive can be a slow process, ensure that the archive is not modified during the move. Therefore, only archives that have become read-only can be moved. Archives become read-only when their end time is older than the **Data is Read Only After (Hours)** property. Therefore, be sure to set your time offsets greater than this property. If you set it smaller than this property, then the value of the property is used instead.

The duration of the move depends on the size of the archives and the speed of the source and destination disks. During the move, the source archive is still available for reading. Once the move has completed successfully, the source archive is deleted.

A move is stopped in the following scenarios:

- If a move failure occurs: The move is stopped, and an appropriate message is logged in the log file. The move retires later.
- If system shuts down during the move: The move is stopped. Since archives are being moved and deleted during this process, it is important to have proper backups of the system in case errors occur.

**Note:**
When configuring archive paths in a Mirror setup, the configured paths must exist on all mirror nodes.

**Example of Multiple Archive Paths**

Consider that your system setup includes the following:

- A fast NVME drive (N:), though not very large.
- A slow HDD (H:), which is larger compared to the fast NVME drive (N;).
- A remote NAS drive mapped to (Z:), the largest of the three drives.

**Note:**
All the three drives use the \Proficy Historian Data\Archives directory.

Your system design requirement is to have your frequently used data on the NVME for the best performance. The NVME drives greatly improve read and write performance than the traditional HDDs. Looking at your data access needs, data storage rate, and the NVME size, you decide to have three months of data there. You do the same for the HDD and decide to have nine months of data there. Data older than that goes to the NAS drive. For a simpler understanding, let us consider that you decide to have daily archives. Finally, you decide that data becomes read-only in a month.

Before you configure multiple archive paths, know the following:

- The multiple archive paths feature is available for any data store other than the System data store.
- While moving archives can be helpful, if not managed well, it may strain disk resources and impact system performance. Although it is ensured that system performance is not impacted by moving

archives, it is recommended that you move archives in smaller sizes and be mindful that your configurations do not affect system performance.

- Archive performance, particularly read performance, is crucial to be considered. When data is moved to slower drives, slower read performance is expected, but this is acceptable only for data on those slower paths. Data reads on faster paths should remain unaffected. To manage this, the Historian treats all paths except the default one as "slower" locations. Determining which archives are needed for queries can be challenging until after the query is executed.

  For instance, queries like "Raw By Number" reading backwards could require access to any archive in the past. However, queries with specific time ranges reveal which archives will be necessary. If the system identifies that archives on these "slower" drives are required, the query will be executed at a lower priority using Low Priority Read Threads. The configured number of read threads is divided into three categories: High, Medium, and Low, each with its specific number of threads. Additionally, each category can utilize lower priority threads if necessary.

  For example, a high priority query initially uses a High priority thread but can use a Medium or Low priority thread if all high priority threads are busy. The same flexibility applies to Medium priority queries. It is important to note that thread priority does not affect Windows thread priority; all threads run at the same Windows priority level, so executing queries on Low Priority threads does not affect their performance. Instead, it operates as a thread reservation system. By identifying and assigning "slow" queries to Low priority threads, High and Medium threads remain available for other queries. Generally, most queries are Medium priority unless programmatically changed. You must determine the appropriate number of read threads to configure to ensure adequate resources are provided.

**Example Configuration procedure**

1. Configure the default path. This can be done in the same way as you do for data stores. In this example, this is set to N:\Proficy Historian Data\Archives.
2. Configure the archive locations *(on page 468)* based on the time.

| Time Offset (Hours) | Path |
|---|---|
| 2232 (3 months) | h:\Proficy Historian Data\Archives |
| 8760 (1 year) | Z:\Proficy Historian Data\Archives |

Based on the configuration, the DataArchiver will periodically look if any archives need to be moved. Only archives that are closed (that is, are time-based or have a fixed end time), and are read-only will be checked. The DataArchiver will check the archive's end time against each configured age, from smallest to

largest. The first age where the archive end time is older will be used to determine the directory to which the archive must move to.

So, based on this example, the following will happen:

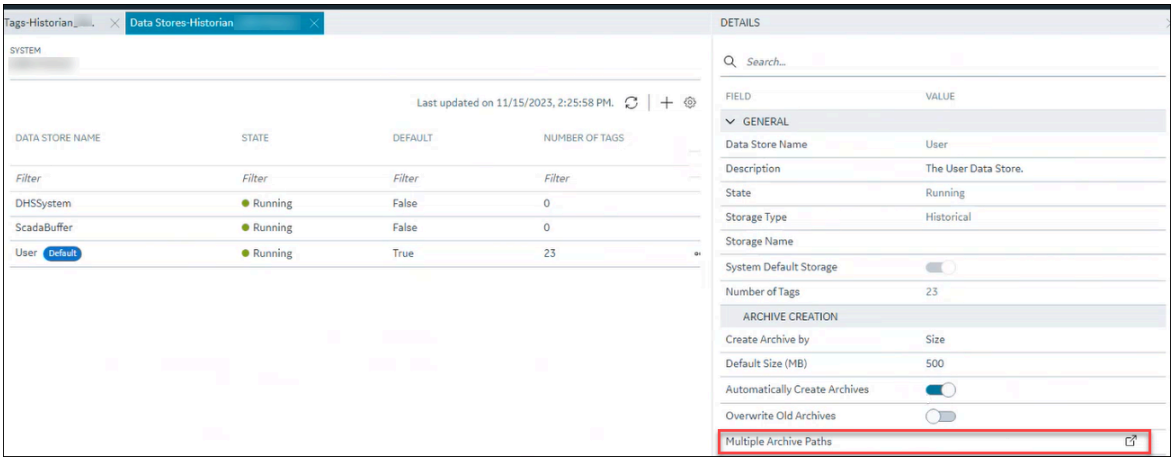
1. The newly created archive will go into the normal default path, that is, N:\Proficy Historian Data \Archives. That archive will exist and be written to until after a month. At that point, it will become read-only because that is what was configured.
2. When the archive is three months (2232 hours) old, the DataArchiver will move the archive to H: \Proficy Historian Data\Archive, as the end time is older than the 2232 hours that you configured and less than the 8760 hours. If the move is successful, the original archive is deleted. This process will repeat when the archive is older than 8760 hours.
3. Then it will be moved to Z:\Proficy Historian Data\Archives. Here it will stay unless the configuration is changed. If, for example, the 8760 was changed to 16520, then an archive that was 10000 hours old would be moved back to H:\Proficy Historian Data\Archives.

## Configure Multiple Archive Paths

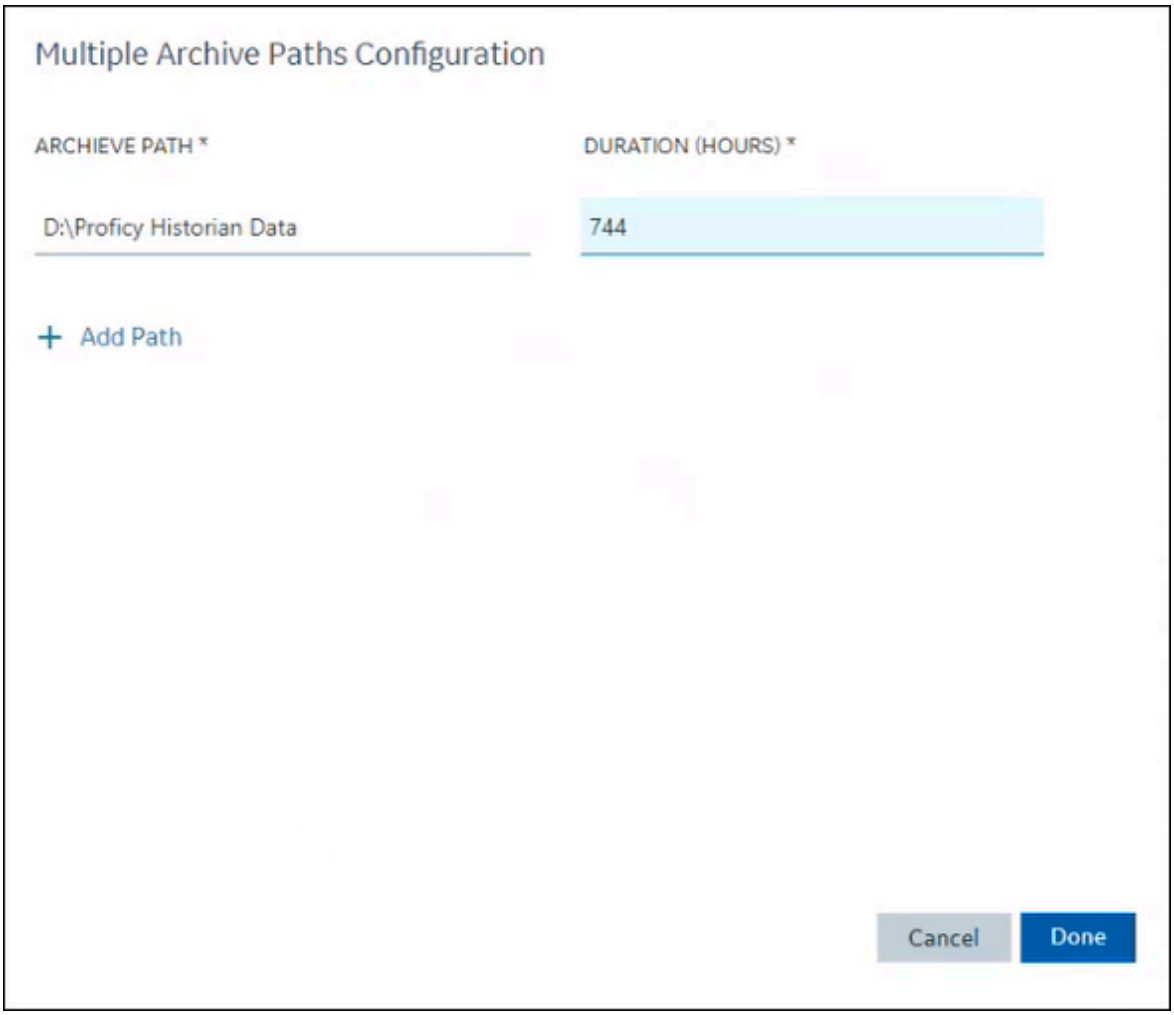
Multiple archive paths feature extends the ability to configure a default archive path for each data store.

- Ensure that you read about multiple archive paths *(on page 464)*.
- You already have a **Default Archive Path** configured for this data store.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Data Stores**. The **Data Stores** section appears.
3. Select the data store as needed.
   The **DETAILS** section displays the details of the data store.
4. In the **ARCHIVE CREATION** group, in the **Multiple Archive Paths** property, select ⧉.



The **Multiple Archive Paths Configuration** window appears.




5. In **ARCHIVE LOCATION**, enter the path as needed.
6. In **DURATION (HOURS)**, enter a duration after which the archive files should be transferred to the specified path.

> **Note:**
> The duration you specify must be greater than the archive hours specified in **Data is Read-Only After (Hours)** property.

7. To add more paths, select **Add Path**.
8. After you add the paths and duration as needed, select **Done**.
   The specified multiple archive paths are added. The archive files in the existing path will be transferred to the new path after the specified duration from their creation or modification.

## Access the Activity Logs of a Data Store

Activity logs are generated when activities are performed on tags and collectors in a data store.

Examples:

- When a tag is created, modified, or deleted
- When a collector instance is created, modified, or deleted
- When data collection for a tag or a collector begins or ends
- When an archive is created or will be closed soon

You can access these logs for each tag/collector or for all the tags and collectors in a system. You can filter these logs based on the start and end dates, priority, topics, and the content in the logs. You can also export all the logs or selected ones.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Data Stores**. Alternatively, you can select **Systems**, right-click the system in which the data store is available (or select ⋯), and then select **Browse Data Stores**.
   The **Data Stores** section appears.
3. Right-click the data store whose archives you want to access (or select ⋯), and then select **Browse Activity Logs**.
   The activity logs of the data store appear. You can filter on the Start Time and End Time along with other fields to search for activity log.

## Access the Tags in a Data Store

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Data Stores**. Alternatively, you can select **Systems**, right-click the system in which the data store is available (or select ⋯), and then select **Browse Data Stores**.
   The **Data Stores** section appears.
3. Right-click the data store whose tags you want to access (or select ⋯), and then select **Browse Tags**.
   The tags in the data store appear, indicating the current status of the data collection for each tag.

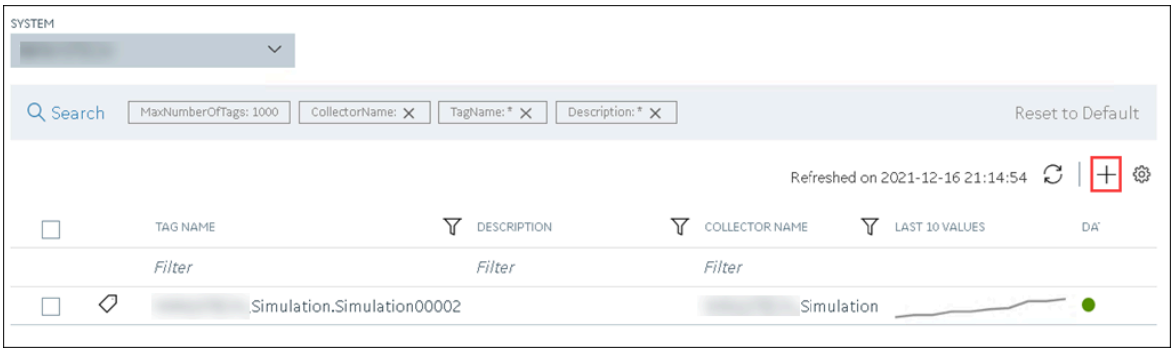
# Add Tags for the Data Store Using Configuration Hub

- Add the collector instance *(on page 357)* using which you want to collect data. Ensure that the collector is running.
- By default, the tag data is stored in the user data store, which is created automatically when you set up Configuration Hub. If, however, you want to store the data in a different data store, create it *(on page 370)*.

This topic describes how to specify the tags for which you want to collect data by browsing through the tags in the data source. For example, for an iFIX collector, if there are 1,00,000 tags in the iFIX server, you must specify the ones for which you want to collect data. Only then data is collected for those tags.

In addition to adding tags from the data source, you can create tags manually *(on page 473)*.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
3. Select +.



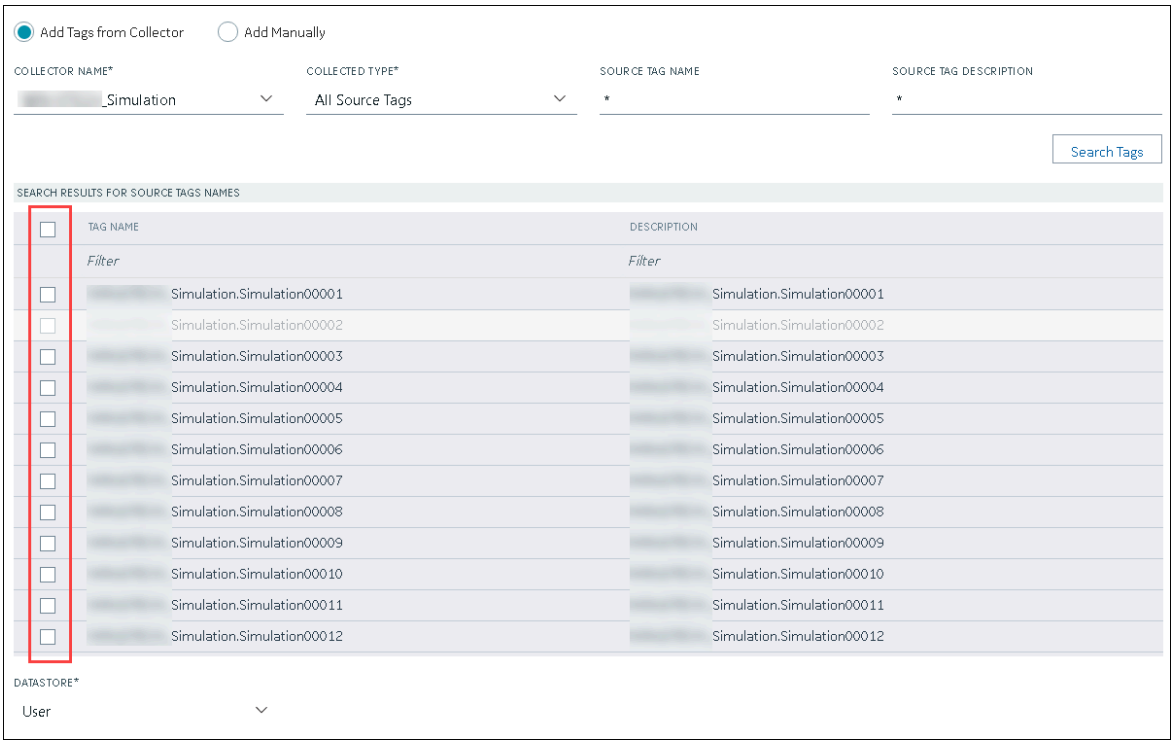

The **Add Tag-<system name>** page appears. The **Add Tags from Collector** option is selected by default.

4. Enter values as described in the following table.

| Field | Description |
|---|---|
| **COLLECTOR NAME** | Select the collector instance that you want to use to collect data. A value is required. |
| **COLLECTED TYPE** | Specify whether you want to browse through all the tags in the data source or only from the tags that you have not added yet. A value is required. |

| Field | Description |
|---|---|
| **SOURCE TAG NAME** | Enter the name of the tag (either completely or partially) to narrow down the search results. |
| **SOURCE TAG DESCRIPTION** | Enter the description of the tag (either completely or partially) to narrow down the search results. |

5. Select **Search Tags**.
   A list of tags that match *all* the criteria that you have specified appears. If a tag is already added, it is disabled.
6. Select the check box corresponding to each tag for which you want to collect data.




7. In the **DATA STORE** field, if you want to store the data in a different data store than the user data store, select the same.
8. Select **Add Tag**.
   Data collection begins for the selected tags.

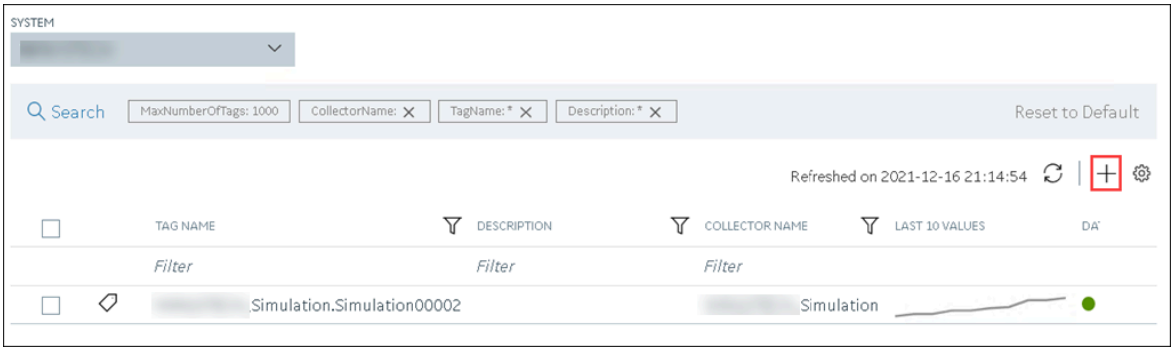
As needed, configure each tag by providing values for the tag properties. For information on the delta query modes, refer to Counter Delta Queries.

# Add a Tag Manually

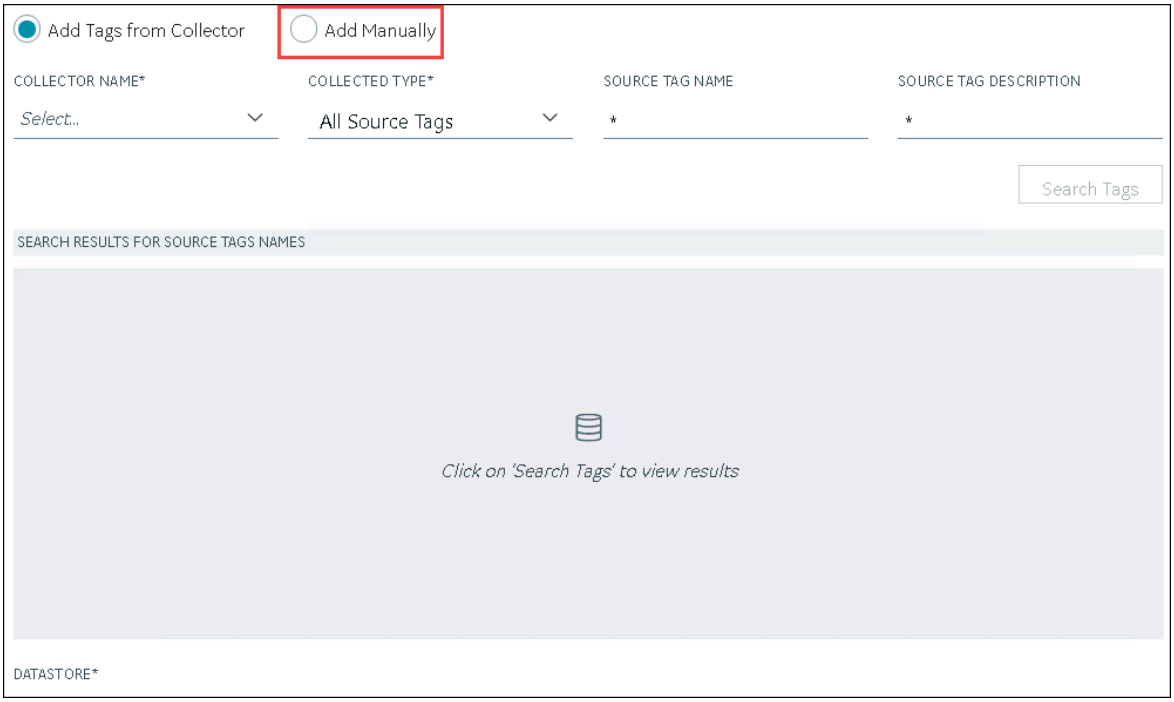
- Add the collector instance *(on page 357)* using which you want to collect data.
- By default, the tag data is stored in the user data store, which is created automatically when you set up Configuration Hub. If, however, you want to store the data in a different data store, create it *(on page 370)*.

After you create a collector instance, you specify which tags from the source must be used for data collection *(on page 357)*. In addition, if you want to use the same tag twice (say, with a different collection interval or collector compression settings), you can add the tag manually. You can also create a calculation tag or a tag to store the values imported using the Excel Add-in.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
3. Select +.




   The **Add Tag-<system name>** page appears. The **Add Tags from Collector** option is selected by default.
4. Select **Add Manually**.




5. Enter values as described in the following table.

| Field | Description |
|---|---|
| **COLLECTOR NAME** | Select the collector instance that you want to use to collect data. If, however, this tag is not associated with a collector, you can leave the field blank (for example, you want to ingest this data manually instead of using a collector). |
| **SOURCE ADDRESS** | Specify the source tag to which you want to map the one you are creating. This field is enabled only if you select a value in the **COLLECTOR NAME** field. When you select ▫▫▫, the **Browse Source Tag: <collector name>** window appears. Provide the search criteria to find the tag that you want to map. |
| **TAG NAME** | Enter a name for the tag. A value is required and must be unique for the Historian server.<br><br>The value that you enter:<br>◦ Must begin with a letter or a number.<br>◦ Can contain up to 256 characters. |

| Field | Description |
|---|---|
| | ◦ Can include any of the following special characters: /!\| #{}%$-_<br>◦ Must not include a space or any of the following characters: ~`+^:;.,?"*=@ |
| **DATA TYPE** | Select the data type of the tag data. To find out the data types supported by a collector, refer to the documentation on the collector that you have created.<br><br>**Important:**<br>If you select an unsupported data type, you may receive incorrect data or even lose data.<br><br>If you select **Multi-Field**, the **USER-DEFINED TYPE NAME** field appears, and the **ENUMERATED SET** and **ARRAY TAG** fields are disabled.<br><br>If you select **Fixed String**, the **STRING LENGTH** field appears. |
| **STRING LENGTH** | Enter the maximum character length allowed for the tag data. This field appears only if the value in the **DATA TYPE** field is **Fixed String**. A value is required.<br><br>You can enter a value between 1 and 255. The default value is 8. |
| **USER-DEFINED TYPE NAME** | Select the user-defined data type (UDT) *(on page 719)* that you want to assign to the tag. This field appears only if the value in the **DATA TYPE** field is **Multi-Field**. A value is required. |
| **ENUMERATED SET** | Select the enumerated set *(on page 707)* that you want to assign to the tag. This field is not applicable for string and multi-field data types (enumerated sets) and for array tags. |
| **ARRAY TAG** | Switch the toggle to indicate whether the tag stores an array of data. This field is disabled if you select a value in the **ENUMERATED SET** field or if the value in the **DATA TYPE** field is **Multi-Field**.<br><br>For information on array tags, refer to About Array Tags *(on page 660)*. |

| Field | Description |
|---|---|
| **TIME RESOLUTION** | Select the time resolution for the tag. A value is required.<br><br>For example, if you select **Seconds**, when you plot the data on a trend chart, the timestamp of the data points will be one second apart. |
| **DATA STORE** | If you want to store the data in a different data store that the user data store, select the same. |

6. Select **Add Tag**.
   Data collection begins for the selected tags.

As needed, configure each tag by providing values for the tag properties. For information on the delta query modes, refer to Counter Delta Queries.

## View the Performance of a Data Store

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Data Stores**.
   Alternatively, you can select **Systems**, right-click the system in which the data store is available (or select ⋯), and then select **Browse Data Stores**.
   The **Data Stores** section appears.
3. Right-click the data store whose performance you want to access (or select ⋯), and then select **View Data Store Performance**.
   The performance of the data store appears, displaying the following information.

| Field | Description |
|---|---|
| **ARCHIVE COMPRESSION** | The current effect of archive data compression. If the value is zero, it indicates that archive compression is either ineffective or disabled. To increase the effect of data compression, increase the value of archive compression deadbands on individual tags.<br><br>In calculating the effect of archive compression, Historian counts internal system tags as well as data source tags. Therefore, when working with a very small number of tags and with compression disabled on data source tags, this field may indicate a value other than zero. If you use a realistic number of |

| Field | Description |
|---|---|
| | tags, however, system tags will constitute a very small percentage of total tags and will therefore not cause a significant error in calculating the effect of archive compression on the total system. |
| **WRITE CACHE HIT** | The hit ratio of the write cache in percentage of total writes. It is a measure of how efficiently the system is collecting data. Typically, this value should range from 95 to 99.99%. If the data is changing rapidly over a wide range, however, the hit percentage drops significantly because current values differ from recently cached values. More regular sampling may increase the hit percentage. Out-of-order data also reduces the hit ratio. |
| **RECEIVE RATE** | Indicates how busy the server is at a given instance and the rate at which the server is receiving data from collectors. |
| **FREE SPACE** | Indicates how much disk space (in MB) is left in the current archive. |
| **CONSUMPTION RATE** | Indicates how fast the archive disk space is consumed. If the value is too high, you can reduce it by slowing the poll rate on selected tags or data points or by increasing the filtering on the data (widening the compression deadband to increase compression). |
| **EST. DAYS TO FULL** | Indicates how much time is left before the archive is full, based on the current consumption rate. This value is dynamically calculated by the server and becomes more accurate as an archive file gets closer to completion. This value is only an estimate and will vary based on a number of factors, including the current compression effectiveness. The system sends messages notifying you at 5, 3, and 1 days until full. After the archive is full, a new archive must be created (can be automatic or manual).<br><br>To increase this value, you must reduce the consumption rate. To ensure that collection is not interrupted, make sure that the **Automatically Create Archives** option is enabled.<br><br>You may also want to enable the **Overwrite Old Archives** option if you have limited disk capacity. Enabling this option, however, |

| Field | Description |
|---|---|
| | means that some old data will be lost when new data overwrites the data in the oldest online archive. Use this feature only when necessary. |
| **FAILED WRITES** | Indicates the number of samples that failed to be written. Since failed writes are a measure of system malfunctions or an indication of offline archive problems, this value should be zero. If you observe a non-zero value, investigate the cause of the problem and take corrective action.<br><br>Historian also generates a message if a write fails. Note that the message only appears once per tag, for a succession of failed writes associated with that tag. For example, if the number displayed in this field is 20, but they all pertain to one Historian tag, you will only receive one message until that Historian tag is functional again. |
| **ALERTS SINCE STARTUP** | Indicates a count of system warnings or alerts generated since the last startup. A high value here may indicate a problem of some kind. You should review the alerts and determine the probable cause. The count resets to zero on restart. The message database, however, may contain more alerts than this value. |
| **MESSAGE SINCE STARTUP** | Displays a count of system messages generated since the last startup. The system resets the value to zero on restart. The message database, however, may contain more messages than this value. |

## Delete a Data Store

The following conditions apply when deleting a data store:

- You cannot delete the system data store.
- You cannot delete a data store if it contains tags. If you remove the tags from the system or permanently delete them, you can delete the data store. However, the archives are not deleted.
- You cannot delete the default data store. You can delete a user data store as long as there is another default data store set for tag addition.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Data Stores**.
   The **Data Stores** section appears.
3. Right-click the data store that you want to delete (or select ⋯), and then select **Delete**.
   A message appears, asking you to confirm that you want to delete the data store.
4. Select **Delete**.
   The data store is deleted. However, the archives in the data store are not deleted.

# Adding a Collector Instance

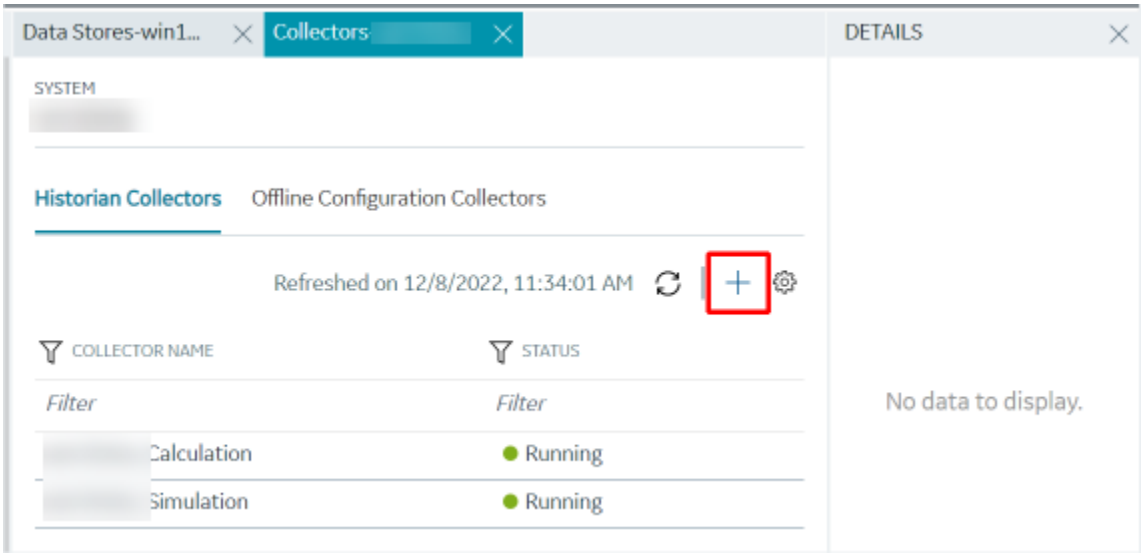
## Add and Configure a Calculation Collector

Using the Calculation collector, you can perform data calculations on values already in the archiver. It retrieves data from tags in the Historian archive, performs the calculation, and then stores the resulting values into new archive tags.

You can create a Calculation collector only for an on-premises Historian server, not for a cloud destination.

This topic describes how to add a collector instance using Configuration Hub. You can also add a collector instance using the RemoteCollectorConfigurator utility *(on page 797)*, which does not require you to install Web-based Clients.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.
   A list of collectors in the default system appears.
3. In the upper-right corner of the main section, select +.




The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.

4. In the **MACHINE NAME** field, select the machine in which you want to add a collector instance.
5. In the **COLLECTOR TYPE** field, select **Calculation Collector**, and then select **Get Details**.
   The **INSTALLATION DRIVE** and **DATA DIRECTORY** fields are disabled and populated.
6. Select **Next**.
   The **Source Configuration** section appears. The **HISTORIAN SERVER** field is disabled and populated.
7. Select **Next**.
   The **Destination Configuration** section appears. Under **CHOOSE DESTINATION**, the **Historian Server** option is selected by default; the other options are disabled. In addition, the **DESTINATION HISTORIAN SERVER** field is disabled and populated with the value you selected in the **MACHINE NAME** field in the **Collector Selection** section.
8. If you created security groups or enabled a strict client/collector authentication, enter the **USERNAME** and **PASSWORD** of the on-premises Historian server that you created during the installation of the collector.
   If you entered the **USERNAME**and **PASSWORD**, select **Test Connection**. This will help you to test if the Historian server that you are trying to connect is valid or if the credentials that you entered are valid.
9. Select **Next**.
   The **Collector Initiation** section appears.
10. If needed, modify the value in the **COLLECTOR NAME** field.
    The value that you enter:

- Must be unique.
- Must not exceed 15 characters.
- Must not contain a space.
- Must not contain special characters except a hyphen, period, and an underscore.

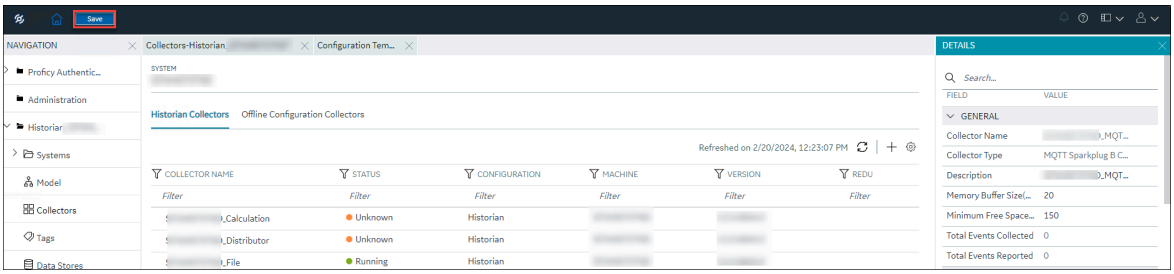
11. In the **RUNNING MODE** field, select one of the following options.
    - **Service - Local System Account**: Select this option if you want to run the collector as a Windows service using the credentials of the local user (that is, the currently logged-in user). If you select this option, the **USERNAME** and **PASSWORD** fields are disabled.
    - **Service Under Specific User Account**: Select this option if you want to run the collector as a Windows service using a specific user account. If you select this option, you must enter values in the **USERNAME** and **PASSWORD** fields. If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:
        - iH Security Admins
        - iH Collector Admins
        - iH Tag Admins

    You can also configure the collector to start automatically when you start the computer.
12. Select **Add**.
    The collector instance is added. The fields specific to the collector section appear in the **DETAILS** section.
13. Under **COLLECTOR SPECIFIC CONFIGURATION**, configure values as described in the following table.

| Field | Description |
|---|---|
| **Calculation Timeout (sec)** | The maximum time a calculation must be performed before being terminated. The default value is 10 seconds. If the calculation takes longer, it is cancelled, and a bad data quality sample is stored in the destination tag with a subquality, calculation error. |
| **Max Recovery Time (hr)** | The maximum time, in hours till now, that the collector will attempt to restore data. This is applicable only to event-based tags. The default value is 4 hours.<br><br>If you want to disable automatic calculation of the tag, set the value of this field to 0. |
| **MTLS Security** | Indicates whether you want to use Mutual TLS (MTLS) protocol to enforce a secure and strong authentication mechanism. |

| Field | Description |
|---|---|
| **MTLS Data Encryption** | Indicates whether you want to encrypt the data that the collector shares to the data archiver (DA). |
| For more information on how to enable MTLS Security, refer to Enable MTLS Security *(on page 632)*. | |

14. If needed, enter values in the other sections *(on page 579)*.
15. In the upper-left corner of the page, select **Save**.




    The changes to the collector instance are saved.
16. If needed, restart the collector.

Specify the tags *(on page 357)* whose data you want to collect using the collector.

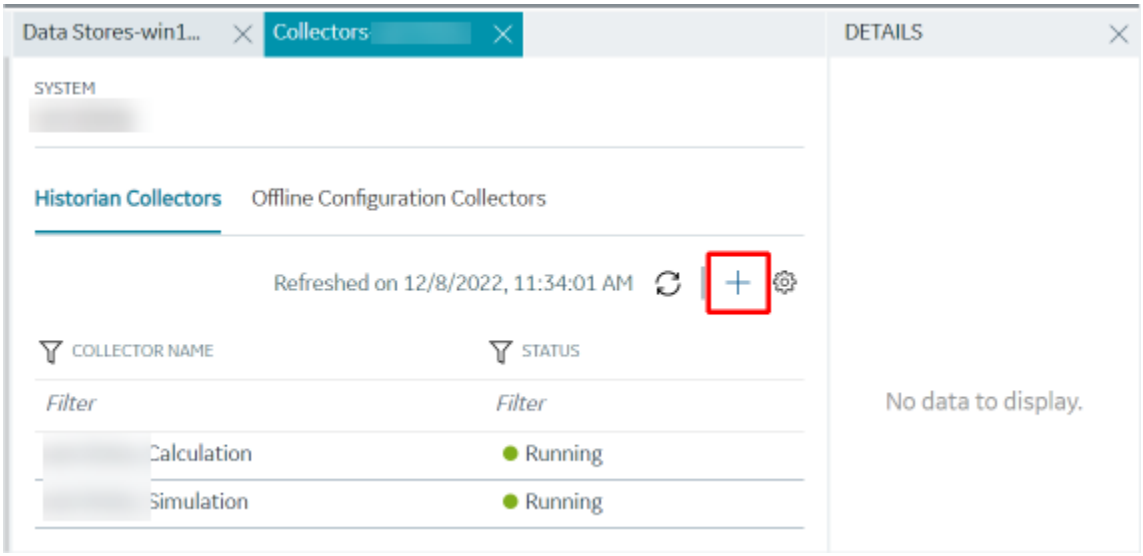
## Add and Configure a CygNet Collector

A CygNet collector collects data from a CygNet server and stores it in the Historian server. For more information, refer to Overview of the CygNet Collector *(on page 2167)*.

> **Note:**
> You cannot send data to a cloud destination using a CygNet collector.

This topic describes how to add a collector instance using Configuration Hub. You can also add a collector instance using the RemoteCollectorConfigurator utility *(on page 797)*, which does not require you to install Web-based Clients.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.
   A list of collectors in the default system appears.
3. In the upper-right corner of the main section, select +.




The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.

4. In the **MACHINE NAME** field, select the machine in which you want to add a collector instance.
5. In the **COLLECTOR TYPE** field, select **Cygnet Collector**, and then select **Get Details**.
   The **INSTALLATION DRIVE** and **DATA DIRECTORY** fields are disabled and populated.
6. Select **Next**.
   The **Source Configuration** section appears.
7. In the **SERVER SITE** field, enter the host name or IP address of the CygNET server from which you want to collect data.
8. Select **Next**.
   The **Destination Configuration** section appears. Under **CHOOSE DESTINATION**, the **Historian Server** option is selected by default; the other options are disabled.
9. If you created security groups or enabled a strict client/collector authentication, enter the **USERNAME** and **PASSWORD** of the on-premises Historian server that you created during the installation of the collector.
   If you entered the **USERNAME**and **PASSWORD**, select **Test Connection**. This will help you to test if the Historian server that you are trying to connect is valid or if the credentials that you entered are valid.
   If the entered credentials are valid, a successful connection message appears.
10. Select **Next**.
    The **Collector Initiation** section appears. The **COLLECTOR NAME** field is populated with a value in the following format: `<Historian server name>_Cygnet`
11. If needed, modify the value in the **COLLECTOR NAME** field.

The value that you enter:

- Must be unique.
- Must contain the string `Cygnet`.
- Must not exceed 15 characters.
- Must not contain a space.
- Must not contain special characters except a hyphen, period, and an underscore.

12. In the **RUNNING MODE** field, select one of the following options.
    - **Service - Local System Account**: Select this option if you want to run the collector as a Windows service using the credentials of the local user (that is, the currently logged-in user). By default, this option is selected, and the **USERNAME** and **PASSWORD** fields are disabled.
    - **Service Under Specific User Account**: Select this option if you want to run the collector as a Windows service using a specific user account. If you select this option, you must enter values in the **USERNAME** and **PASSWORD** fields.

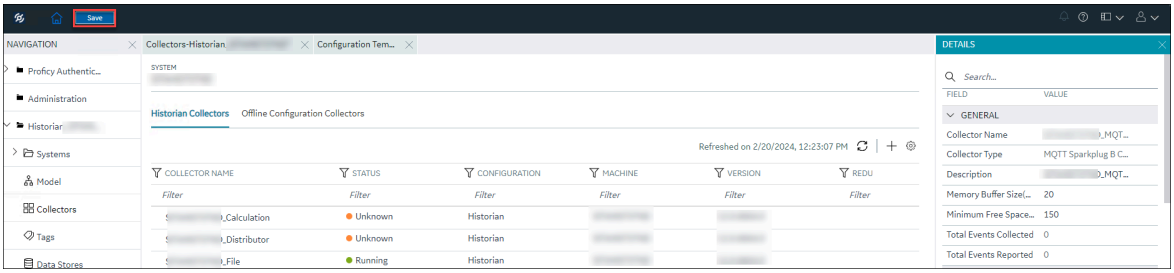
      If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:
      - iH Security Admins
      - iH Collector Admins
      - iH Tag Admins

    You can also configure the collector to start automatically when you start the computer.
13. Select **Add**.
    The collector instance is added. The fields specific to the collector section appear in the **DETAILS** section.
14. Under **COLLECTOR SPECIFIC CONFIGURATION**, configure values as described in the following table.

| Field | Description |
|---|---|
| **Recovery Time** | The maximum time, in hours, for which the collector will attempt to recover data after the collector is started or when connection between the collector and the CygNet server is re-established. This time is calculated as the duration between the current time and the last known write time.<br><br>Continuous data collection is resumed only after the previous data has been recovered.<br><br>By default, this value is set to 0, which means data recovery is not attempted. The maximum value you can provide is 168 hours (that is, 7 days). |

| Field | Description |
|---|---|
| **Thread Count** | The maximum number of threads that you want to collector to use to query data from the CygNet server. |
| **CygNet Debug Mode** | The debug mode for the collector. You can enter a value between 0 and 255, where 0 turns off debugging and 255 enables detailed debugging (with query transactions).<br><br>**Note:**<br>Do not turn on debugging for a long period. If you do so, very large log files are created, which can consume a great deal of disk space. We recommend a maximum of 10 minutes. |
| **General Optimized** | Indicates whether you want to apply optimization on the tag data reads. |
| **Service-Site** | Identifies the CygNet site or data source from which the CygNet collector collects data. A value is required. |
| **MTLS Security** | Indicates whether you want to use Mutual TLS (MTLS) protocol to enforce a secure and strong authentication mechanism. |
| **MTLS Data Encryption** | Indicates whether you want to encrypt the data that the collector shares to the data archiver (DA). |
| For more information on how to enable MTLS Security, refer to Enable MTLS Security *(on page 632)*. | |

15. If needed, enter values in the other sections common to all collectors *(on page 579)*.
16. In the upper-left corner of the page, select **Save**.




The changes to the collector instance are saved.

17. If needed, restart the collector.

Specify the tags *(on page 357)* whose data you want to collect using the collector.

## Add and Configure a File Collector

A File collector is used to send data from one Historian server to another one.

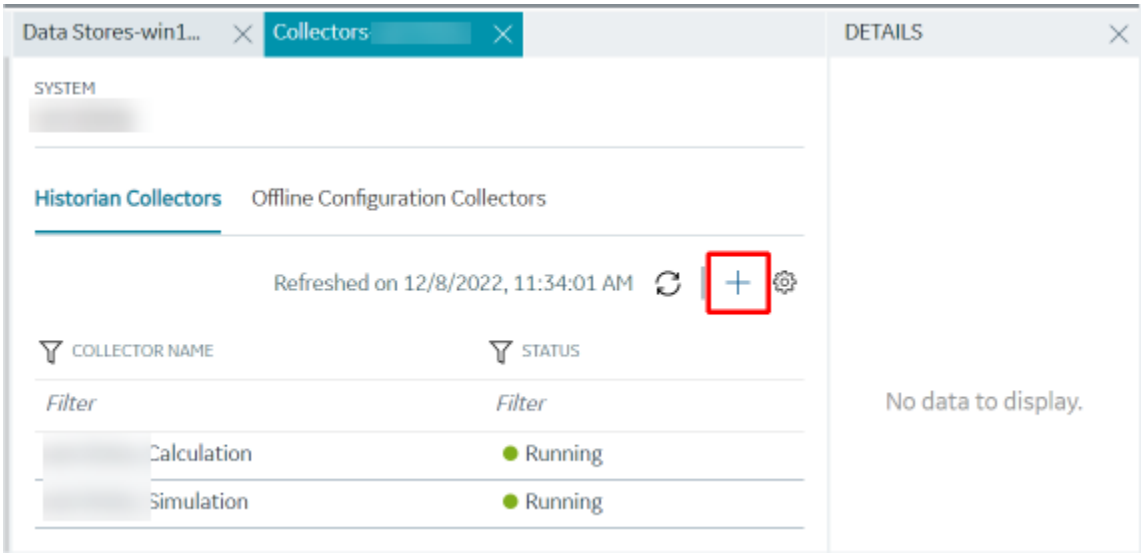
**Note:**

- You cannot send data to a cloud destination using the File collector.
- You can create only one instance of the File collector.

For more information, refer to Overview of the File Collector *(on page 2181)*.

This topic describes how to add a collector instance using Configuration Hub. You can also add a collector instance using the RemoteCollectorConfigurator utility *(on page 797)*, which does not require you to install Web-based Clients.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**. A list of collectors in the default system appears.
3. In the upper-right corner of the main section, select +.




The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.

4. In the **MACHINE NAME** field, select the machine in which you want to add a collector instance. The collector will send data to this machine.
5. In the **COLLECTOR TYPE** field, select **File Collector**, and then select **Get Details**.
   The **INSTALLATION DRIVE** and **DATA DIRECTORY** fields are disabled and populated.
6. Select **Next**.
   The **Destination Configuration** section appears. Under **CHOOSE DESTINATION**, the **Historian Server** option is selected by default; the other options are disabled.
7. If you created security groups or enabled a strict client/collector authentication, enter the **USERNAME** and **PASSWORD** of the on-premises Historian server that you created during the installation of the collector.
   If you entered the **USERNAME**and **PASSWORD**, select **Test Connection**. This will help you to test if the Historian server that you are trying to connect is valid or if the credentials that you entered are valid.
   If the entered credentials are valid, a successful connection message appears.
8. Select **Next**.
   The **Collector Initiation** section appears. The **COLLECTOR NAME** field is populated with a value in the following format: `<Historian server name>_File`
9. If needed, modify the value in the **COLLECTOR NAME** field.
   The value that you enter:
   - Must be unique.
   - Must not exceed 15 characters.

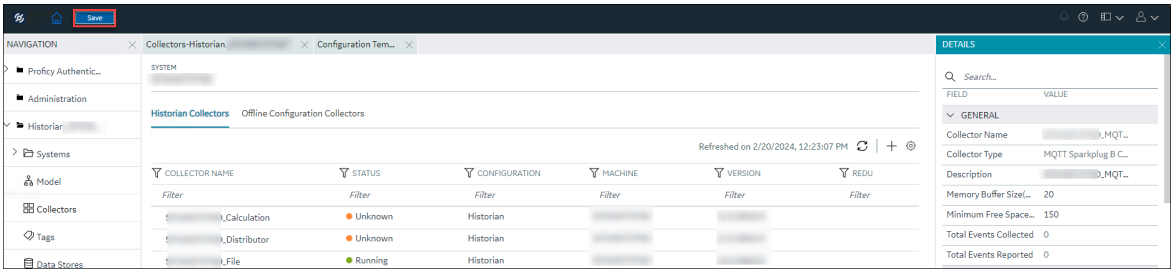
- Must not contain a space.
- Must not contain special characters except a hyphen, period, and an underscore.

10. In the **RUNNING MODE** field, select one of the following options.
    - **Service - Local System Account**: Select this option if you want to run the collector as a Windows service using the credentials of the local user (that is, the currently logged-in user). By default, this option is selected, and the **USERNAME** and **PASSWORD** fields are disabled.
    - **Service Under Specific User Account**: Select this option if you want to run the collector as a Windows service using a specific user account. If you select this option, you must enter values in the **USERNAME** and **PASSWORD** fields. If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:
        - iH Security Admins
        - iH Collector Admins
        - iH Tag Admins

    You can also configure the collector to start automatically when you start the computer.
11. Select **Add**.
    The collector instance is added. The fields specific to the collector section appear in the **DETAILS** section.
12. In the **COLLECTOR SPECIFIC CONFIGURATION** section, configure values as described in the following table.

| Field | Description |
|---|---|
| **Scan Interval** | The interval, in seconds, after which the collector initiates an import operation. The maximum value that you can enter is 65. |
| **CSV File Spec** | The file extension for a CSV file to be imported. You can specify more than one extension type, such as csv, txt, dat. |
| **XML File Spec** | The file extension for an XML file to be imported. |
| **Purge Processed (days)** | The number of days after which you want the contents of the `Processed Files` folder to be automatically purged. |
| **Purge Error (days)** | The number of days after which you want the contents of the `Error Files` folder to be automatically purged. |

13. Ad needed, enter values in the other sections common to all collectors *(on page 579)*.
14. In the upper-left corner of the page, select **Save**.




The changes to the collector instance are saved.

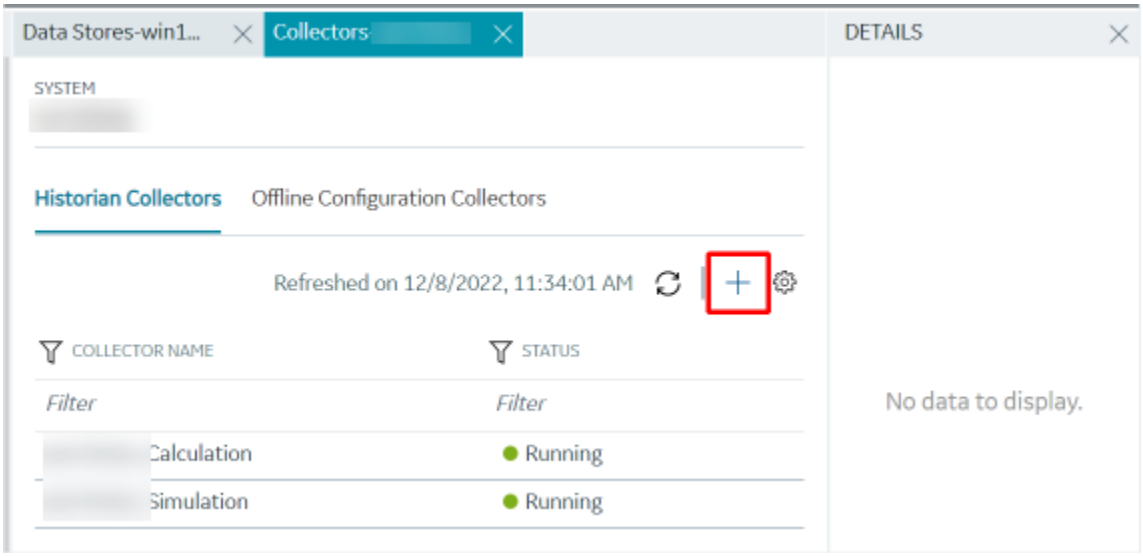
15. If needed, restart the collector.

Import files using the collector *(on page 2188)*.

## Add and Configure a HAB Collector

The HAB collector collects data from Habitat, which is a SCADA application that contains real-time data. The collector interacts with the Habitat Sampler application to fetch data from the Habitat database records and stores the data in a Historian server. For more information, refer to Overview of the HAB Collector *(on page 2208)*.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**. A list of collectors in the default system appears.
3. On the right, next to Settings in the main section, select +.



The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.

4. In the **MACHINE NAME** field, select the machine in which you want to add a collector instance.
5. In the **COLLECTOR TYPE** field, select **Hab Collector**, and then select **Get Details**.
   The **INSTALLATION DRIVE** and **DATA DIRECTORY** fields are disabled and populated.
6. Select **Next**.
   The **Source Configuration** section appears.
7. Enter values as described in the following table.

| Field | Description |
|---|---|
| **SERVER SITE** | Enter name that you want to assign to the site. A value is required and must be unique. It is used by Habitat to identify the collector instance. By default, this field is populated with a value in the following format: `<Historian server name>Hab` |
| **SERVER 1** (under NODE 1) | Enter the host name or IP address of the Habitat server in the primary site from which you want to collect data. This server acts as the primary/active server from which the collector receives data. A value is required. |
| **SERVER 1** (under NODE 2) | Enter the host name or IP address of the Habitat server in the second/backup site from which you want to collect data. This server acts as a standby server in case server 1 under node 1 fails. A value is required. If you do not have a secondary/backup site, enter the same value as **SERVER 1** under node 1. |
| **SERVER 2** (under NODE 1) | Enter the host name or IP address of the Habitat server that you want to use as a standby server in the same site as server 1. This server acts as a standby server in case server 1 under node 2 fails. |
| **SERVER 2** (under NODE 2) | Enter the host name or IP address of the Habitat server in the secondary/backup site from which you want to collect data. This server acts as a standby server in case server 2 under node 1 fails.<br><br>For example, suppose Machine A and Machine B are in node AB, and Machine X and Machine Y are in node XY. Suppose you |

| Field | Description |
|---|---|
| | want to use Machine A as the primary server and the remaining machines as standby servers. In that case, enter values as follows:<br>◦ **SERVER 1** (under Node 1): `Machine A`<br>◦ **SERVER 2** (under Node 1): `Machine B`<br>◦ **SERVER 1** (under Node 2): `Machine X`<br>◦ **SERVER 2** (under Node 2): `Machine Y`<br>If Machine A fails, the Machine B becomes active. If Machine B fails, Machine X becomes active. If Machine X fails, Machine Y becomes active. |
| **SOCKET** | The socket number (port number) used by the Habitat Sampler application to connect. Each collector instance can connect to only one socket. The default value is 8040. |
| **RETRY** | The duration, in seconds, after which the collector tries to communicate with the site. The default value is 5 seconds. |

8. Select **Next**.
   The **Destination Configuration** section appears. Under **CHOOSE DESTINATION**, the **Historian Server** option is selected by default. The other options are disabled because you cannot send data to a cloud destination using the HAB collector.
9. If you created security groups or enabled a strict client/collector authentication, enter the **USERNAME** and **PASSWORD** of the on-premises Historian server that you created during the installation of the collector.
   If you entered the **USERNAME**and **PASSWORD**, select **Test Connection**. This will help you to test if the Historian server that you are trying to connect is valid or if the credentials that you entered are valid.
   If the entered credentials are valid, a successful connection message appears.
10. Select **Next**.
    The **Collector Initiation** section appears. By default, the **COLLECTOR NAME** field is populated with a value in the following format: `<Historian server name>_Hab`
11. If needed, modify the value in the **COLLECTOR NAME** field.
    The value that you enter:
    ◦ Must be unique.
    ◦ Must not exceed 15 characters.

- Must not contain a space.
- Must not contain special characters except a hyphen, period, and an underscore.

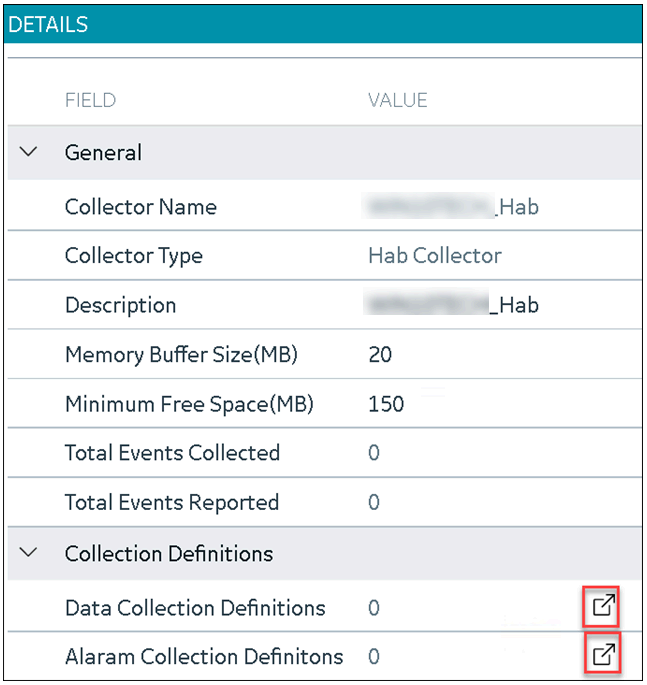
12. In the **SamplerID** field, enter the user ID to connect to Habitat Sampler.

    By default, this field contains the Collector Name. You can first provide the Collector Name and update the Sample ID field as this filed will automatically takes the Collector Name.

13. In the **RUNNING MODE** field, select one of the following options.
    - **Service - Local System Account**: Select this option if you want to run the collector as a Windows service using the credentials of the local user (that is, the currently logged-in user). If you select this option, the **USERNAME** and **PASSWORD** fields are disabled. By default, this option is selected.
    - **Service Under Specific User Account**: Select this option if you want to run the collector as a Windows service using a specific user account. If you select this option, you must enter values in the **USERNAME** and **PASSWORD** fields.

      If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:
      - iH Security Admins
      - iH Collector Admins
      - iH Tag Admins

14. Select **Add**.
    The collector instance is added. The fields specific to the collector appear in the **DETAILS** section.
15. Depending on whether you want to configure tags or alarms, select next to the corresponding field under **Collection Definitions**.



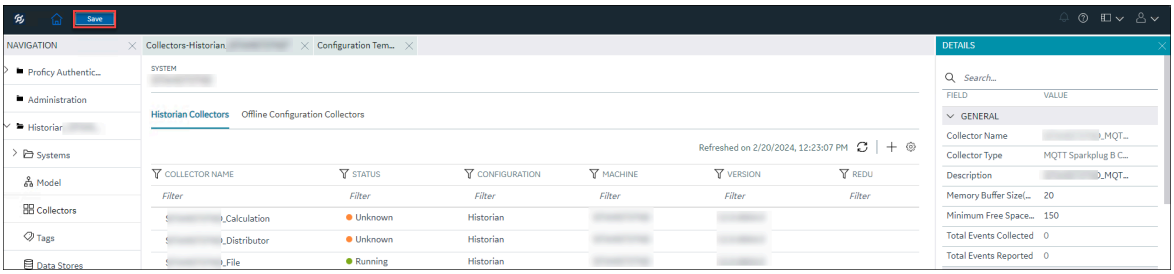

The **Data Collections** or **Alarm Collections** section appears.

16. Select +, and then enter values in the available fields for data collection and/or alarm collection *(on page 495)*. You can also copy a collection definition by right-clicking it and selecting **Duplicate**.
17. Under **COLLECTOR SPECIFIC CONFIGURATION**, configure values as described in the following table.

| Field | Description |
|---|---|
| **Auto Tag Sync** | If you enable this option, the collector creates tags automatically in Historian based on the key value. In addition, any tag changes in Habitat (such as adding, renaming, and deleting tags) will reflect automatically in Historian. No manual steps are required.<br><br>If you disable this option, any tag changes in Habitat will be captured in the `<collector name>_Tag_Unconfirmed.xml` file. Only after you approve these changes *(on page 2243)*, they are reflected in Historian. |
| **Tag Deletion Type** | Specify whether deleted tags in Habitat that you have approved must be deleted or disabled for data collection in Historian. Enter one of the following values: |

| Field | Description |
|---|---|
| | ◦ DISABLE_TAG (this is the default value)<br>◦ DELETE_TAG |
| **MTLS Security** | Indicates whether you want to use Mutual TLS (MTLS) protocol to enforce a secure and strong authentication mechanism. |
| **MTLS Data Encryption** | Indicates whether you want to encrypt the data that the collector shares to the data archiver (DA). |
| For more information on how to enable MTLS Security, refer to Enable MTLS Security *(on page 632)*. | |

18. As needed, provide values in the the other sections common to all collectors *(on page 579)*.
19. In the upper-left corner of the page, select **Save**.




The changes to the collector instance are saved.

20. Start the collector *(on page 637)*.

If you have disabled the automatic tag sync option in the configuration file, tag changes in Habitat (such as adding, renaming, and deleting tags) are captured in the `<collector_name>_tag.xml` file. You must approve the changes so that they are reflected in Historian. To do so:

1. Right-click the HAB collector instance that you have created, and then select **Confirm Queued Tags**.

   A list of tags that have been changed appears.

2. Select the check boxes corresponding to the tags whose changes you want to approve, and then select **Confirm**. You can filter the list using the **TAG TYPE** field.

   The tag changes are approved. The status of the tags is updated in the `<collector_name>_tag.xml` file (that is, the Confirmed parameter is set to true).

## Collection Definitions of a HAB Collector

**Table 27. Tag Data Collection Definition Parameters**

| Parameter | Description |
|---|---|
| **DEFINITION NAME** | The name of the collection definition. A value is required and must be unique.<br><br>**Important:**<br>You must not change the name after the collector starts collecting data. If, however, you want to change the name after the collector has started collecting data, refer to FAQs on HAB Collector *(on page 2246)*. |
| **STATUS** | Indicates whether the collection definition is enabled. If an error occurs while processing the collection definition, the collector automatically disables the collection definition. |
| **FAMILY** | The name of the Habitat family from which you want to collect data (for example, EMS, DTS). A value is required and must be unique. |
| **APPLICATION** | The name of the Habitat application from which you want to collect data (for example, SCADA). A value is required. |
| **DATABASE** | The name of the Habitat database from which you want to collect data (for example, SCADAMOM). A value is required. |
| **RECORD TYPE** | The name of the HDB record from which you want to collect data (for example, ANALOG,POINT,COUNT). A value is required. Enter any record type that contains the composite key and MRID fields (because the collector uses these two fields to create tags). |
| **COLLECTION TYPE** | Indicates whether you want to perform polled or unsolicited data collection.<br><br>• **Polled**: Indicates a periodic data collection, where data is collected at a regular time interval (indicated as PERIODIC in Habitat).<br>• **Unsolicited**: Indicates that data is collected only when values have changed since the last time data was collected (indicated as EXCEPTION in Habitat). |

**Table 27. Tag Data Collection Definition Parameters (continued)**

| Parameter | Description |
|---|---|
| **KEY** | The value that will be used to filter tags for data collection. A value is required. You can use the wildcard character * to get a range of values. For example, if you want to collect data from all tags that begin with SUBSTN.LAKEVIEW, enter: SUBSTN.LAKEVIE*.*.*.*.* |
| **PREFIX FIELD** | The prefix that you want to use for tags. You can provide a different value for each collection definition, which helps you identify tags based on the collection definition. |
| **TAG NAME FIELD** | Determines how the tags created in Historian must be named. A value is required.<br><br>For example, if you want the tags to be named after the value in the LOC_CIRCLG field, enter `LOC_CIRCLG`. When you do so, if the value in the LOC_CIRCLG field in Habitat is DOUGLAS, the tag created in Historian will be named <TagPrefix value>.<AlarmPrefix value>.DOUGLAS.<br><br>You can enter multiple values separated by commas.<br><br>For example, if you want the tags to be named after the values in the LOC_CIRCLG and PRIOR_CIRCLG fields, enter `LOC_CIRCLG,PRIOR_CIRCLG`. When you do so, if the values in the LOC_CIRCLG and PRIOR_CIRCLG fields are Douglas and 1 respectively, the tag created in Historian will be named <TagPrefix value>.<AlarmPrefix value>.DOUGLAS.1. |
| **MRID FIELD** | The MRID of the record. A value is required. If, however, MRID is not available, enter the composite key or any other unique identifier of the record. |
| **DESCRIPTION FIELD** | The description of the tag. A value is required. |
| **VALUE FIELD** | The name of the property that stores the tag value in Habitat. A value is required.<br><br>You can enter multiple values, separated by commas. One Historian tag will be created for each value you enter in this field. |

**Table 27. Tag Data Collection Definition Parameters (continued)**

| Parameter | Description |
|---|---|
| **TIMESTAMP FIELD** | The name of the property that stores the timestamp of a tag in Habitat. You can use a HDB timestamp or custom/alias timestamps.<br><br>**Examples:**<br><br>For analog, you can use:<br><br>• **FIELDTIME**: A combination of FLDTIME_ANALOG and FLDMSEC_ANALOG, in the hour: min:sec: millisec format.<br>• **SCADATIME**: Used to capture SCTIME_ANALOG in the hour: min:sec: millisec format.<br>• **SAMPLETIME**: The time at which the data sample was collected in Habitat in the hour: min:sec format.<br><br>For point, you can use:<br><br>• **FIELDTIME**: A combination of FLDTIME_POINT and FLDMSEC_POINT, in the hour: min:sec: millisec format.<br>• **SCADATIME**: Used to capture SCTIME_POINT in the hour: min:sec: millisec format.<br>• **SAMPLETIME**: The time at which the data sample was collected in Habitat in the hour: min:sec format.<br><br>For count, you can use:<br><br>• **FIELDTIME**: A combination of FLDTIME_COUNT and FLDMSEC_COUNT, in the hour: min:sec: millisec format.<br>• **SAMPLETIME**: The time at which the data sample was collected in Habitat in the hour: min:sec format.<br><br>You can edit the names of these custom/alias timestamps using the registry entries FieldTimeCustomFieldName, ScadaTimeCustomFieldName, and SampleTimeCustomFieldName respectively. |
| **QUALITY FIELD** | The name of the property that stores the tag quality in Habitat. |
| **SAMPLE RATE** | The rate at which you want to collect or poll data. A value is required. |

**Table 27. Tag Data Collection Definition Parameters (continued)**

| Parameter | Description |
|---|---|
| | For example, if the sample rate is 10 seconds:<br>• For a polled collection type, data is written to Data Archiver every 10 seconds.<br>• For an unsolicited collection type, the collector checks for any data changes every 10 seconds. Only if there are changes, the data is written to Data Archiver. |
| **SAMPLE UNIT** | The unit of measurement for the sample rate. A value is required. Valid values:<br>• sec<br>• min<br>• hour<br>• day<br>• week<br>• month |
| **PERMANENT** | Indicates whether you want to store the data in buffer files in Habitat in the event of a connection loss. We strongly recommend that you set this parameter to true to prevent loss of data. |

**Table 28. Alarm Data Collection Definition Parameters**

| Parameter | Description |
|---|---|
| **DEFINITION NAME** | The name of the collection definition. A value is required and must be unique.<br>**Important:**<br>You must not change the name after the collector starts collecting data. If, however, you want to change the name after the collector has started collecting data, refer to FAQs on HAB Collector *(on page 2246)*. |

**Table 28. Alarm Data Collection Definition Parameters (continued)**

| Parameter | Description |
|---|---|
| **STATUS** | Indicates whether the collection definition is enabled. If an error occurs while processing the collection definition, the collector automatically disables the collection definition. |
| **FAMILY** | The name of the Habitat family from which you want to collect data (for example, EMS, DTS). A value is required and must be unique. |
| **APPLICATION** | The name of the Habitat application from which you want to collect data (for example, SCADA). A value is required. |
| **DATABASE** | The name of the Habitat database from which you want to collect data (for example, ALARMLIST). A value is required. |
| **RECORD TYPE** | The name of the HDB record from which you want to collect data (for example, CIRCLG, ALMQ). A value is required. Enter any record type that contains the composite key and MRID fields (because the collector uses these two fields to create tags). |
| **KEY** | This field is not applicable to alarms. Enter *. |
| **TAG NAME FIELD** | Determines how the tags created in Historian must be named. A value is required.<br><br>For example, if you want the tags to be named after the value in the LOC_CIRCLG field, enter `LOC_CIRCLG`. When you do so, if the value in the LOC_CIRCLG field in Habitat is DOUGLAS, the tag created in Historian will be named <TagPrefix value>.<AlarmPrefix value>.DOUGLAS.<br><br>You can enter multiple values separated by commas.<br><br>For example, if you want the tags to be named after the values in the LOC_CIRCLG and PRIOR_CIRCLG fields, enter `LOC_CIRCLG,PRIOR_CIRCLG`. When you do so, if the values in the LOC_CIRCLG and PRIOR_CIRCLG fields are Douglas and 1 respectively, the tag created in Historian will be named <TagPrefix value>.<AlarmPrefix value>.DOUGLAS.1. |

**Table 28. Alarm Data Collection Definition Parameters (continued)**

| Parameter | Description |
|---|---|
| **PREFIX** | The prefix that you want to use for tags. A value is required. You can provide a different value for each collection definition, which helps you identify tags based on the collection definition. |
| **VALUE** | The name of the property that stores the tag value in Habitat. A value is required.<br><br>You can enter multiple values, separated by commas (for example, `TEXT_CIRCLG, PRIOR_CIRCLG, TIME_CIRCLG`). The values will then be concatenated in the corresponding Historian tag. |
| **TIMESTAMP** | This field is not applicable to alarms. Leave the field blank. You can use the **VALUE FIELD** field to provide the timestamp (for example, TIME_CIRCLG). |
| **QUALITY** | This field is not applicable to alarms. Leave the field blank. |
| **DISABLED** | Switch the toggle to enable alarm filtering. You can then collect data only for the filtered alarms. |
| **ALARM LOCATION** | The alarm location based on which you want to filter alarm data. A value is required if alarm filtering is enabled. You can enter multiple values separated by commas (for example, LA,NY). The default value is *, which indicates that data for that parameter is not filtered. |
| **ALARM AREA** | The alarm area based on which you want to filter alarm data. A value is required if alarm filtering is enabled. You can enter multiple values separated by commas (for example, LAKEVIEW,RICHVIEW). The default value is *, which indicates that data for that parameter is not filtered. |
| **ALARM CATEGORY** | The alarm category based on which you want to filter alarm data. A value is required if alarm filtering is enabled. You can enter multiple values separated by commas (for example, PressureSensor,MotionSensor). The default value is *, which indicates that data for that parameter is not filtered. |
| **ALARM PRIORITY** | The alarm priority based on which you want to filter alarm data. A value is required if alarm filtering is enabled. You can enter multiple |

**Table 28. Alarm Data Collection Definition Parameters (continued)**

| Parameter | Description |
|---|---|
| | values separated by commas (for example, 1,2). The default value is *, which indicates that data for that parameter is not filtered. |
| **ALARM EXCEPTION** | The alarm exception based on which you want to filter alarm data. A value is required if alarm filtering is enabled. You can enter multiple values separated by commas. The default value is *, which indicates that data for that parameter is not filtered. |
| **SAMPLE RATE** | The rate at which you want to collect data. A value is required. |
| **SAMPLE UNIT** | The unit of measurement for the sample rate. A vaue is required.<br>Valid values:<br>• sec<br>• min<br>• hour<br>• day<br>• week<br>• month |
| **PERMANENT** | Indicates whether you want to store the data in buffer files in Habitat in the event of a connection loss. We strongly recommend that you set this parameter to true to prevent loss of data. |

## About Adding an iFIX Collector Instance

This topic provides guidelines on how to configure the iFIX collector using Configuration Hub based on the running mode of iFIX. It also describes the collector behaviour and recommended configuration in each case.

| iFIX Running Mode | Recommended Configuration for the iFIX Collector | Collector Behaviour After You Add the Collector Instance |
|---|---|---|
| iFIX is running in service mode and is secured.<br>The iFIX Alarms and Events and the OPC Alarms and Events Servers are running as service. | Configure the iFIX collector services under a user account under which iFIX is running as a service. While adding an instance of the iFIX collector or the iFIX | • The iFIX collector starts running as a service. It appears in the collectors list in Configuration Hub. |

| iFIX Running Mode | Recommended Configuration for the iFIX Collector | Collector Behaviour After You Add the Collector Instance |
|---|---|---|
| | Alarms and Events collector using Configuration Hub, select **Service Under Specific User Account**. | • You can run the collector at a command prompt using the Collector Start action. A shortcut is created in the Windows Start menu so that you can run the collector in the command-line mode.<br>• By default, when not started as an SCU task, the iFIX collector points to the iFIX nodename. You must configure the iFIX node in the **Collector Configuration** section in Historian Administrator. |
| iFIX is running as a service and is not secured.<br><br>The iFIX Alarms and Events and the OPC Alarms and Events servers are running as service. | You can configure the iFIX collector service using a local system account or a specific user account. | • The iFIX collector starts running as a service.<br>• You can run the collector at a command prompt using the Collector Start action. A shortcut is created in the Windows Start menu so that you can run the collector in the command-line mode.<br>• By default, when not started as an SCU task, the iFIX collector points to the iFIX nodename. You must configure the iFIX node in the **Collector Configuration** section in Historian Administrator. |

| iFIX Running Mode | Recommended Configuration for the iFIX Collector | Collector Behaviour After You Add the Collector Instance |
|---|---|---|
| iFIX is not running as a service mode and is secured. | Configure the iFIX collector services under a user account that is added in the IFIXUSERS group. Do not configure as a local system service. While adding an instance of the iFIX collector or the iFIX Alarms and Events collector using Configuration Hub, select **Service Under Specific User Account**. | • Since Remote Collector Manager tries to start the collector as a service, and iFIX is not running as a service, an error message appears while adding a collector instance. However, the instance is configured successfully although it does not appear in the collectors list in Configuration Hub.<br>• A shortcut is created in the Windows Start menu so that you can run the collector in the command-line mode, and the related registry folder is created.<br>• You must start the collector manually for the first time using the shortcut. It will then connect to the Historian server, and it will then appear in the collectors list in Configuration Hub.<br>• Once connected to server, you can start/stop it at a command prompt. |
| iFIX is not running as a service mode, and is not secured. | You can configure the iFIX collector service using a local system account or a specific user account. | • Since Remote Collector Manager tries to start the collector as a service, and iFIX is not running as a |

| iFIX Running Mode | Recommended Configuration for the iFIX Collector | Collector Behaviour After You Add the Collector Instance |
|---|---|---|
| | | service, an error message appears while adding a collector instance. However, the instance is configured successfully.<br>• A shortcut is created in the Windows Start menu so that you can run the collector in the command-line mode, and the related registry folder is created.<br>• You must start the collector manually for the first time using the shortcut. It will then connect to the Historian server.<br>• Once connected to server, you can start/stop it at a command prompt. |
| iFIX is not running. | You can configure the iFIX collector service using a local system account or a specific user account, as per the security configuration of iFIX. | • Since Remote Collector Manager tries to start the collector as a service, and iFIX is not running as a service, an error message appears while adding a collector instance. However, the instance is configured successfully although it does not appear in the collectors list in Configuration Hub.<br>• A shortcut is created in the Windows Start menu so that you can run the |

| iFIX Running Mode | Recommended Configuration for the iFIX Collector | Collector Behaviour After You Add the Collector Instance |
|---|---|---|
| | | collector in the command-line mode, and the related registry folder is created.<br>• After you start iFIX, you must start the collector manually for the first time using the shortcut. It will then connect to the Historian server.<br>• Once connected to server, you can start/stop it at a command prompt. |

## Add and Configure an iFIX Collector

Ensure that iFIX is running in a Windows-service mode. For more information, refer to About Adding an iFIX Collector Instance *(on page 501)*.

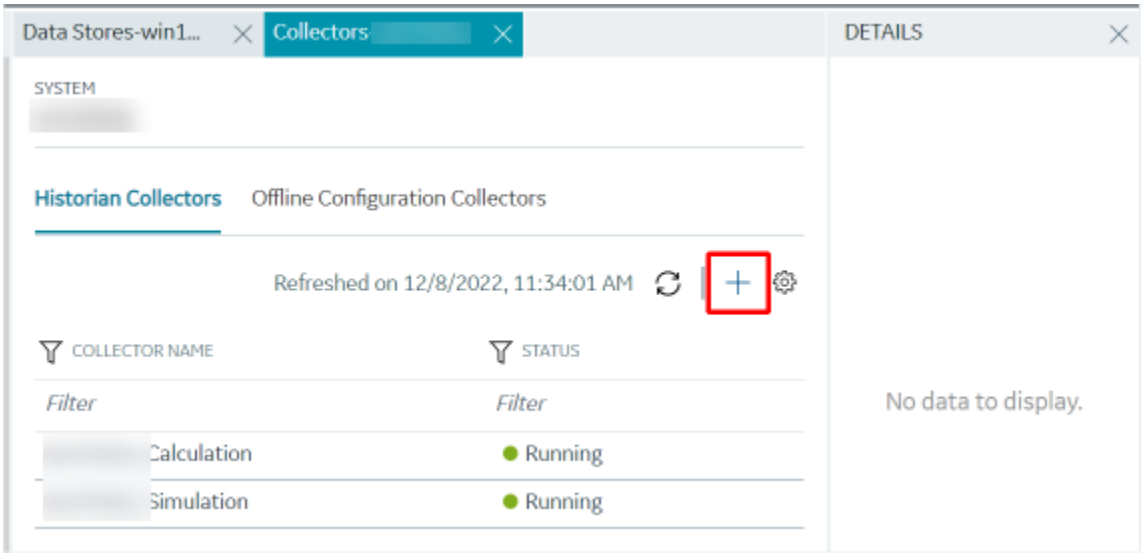
The iFIX collectors collect data from iFIX and store it in the Historian server. They include:

- The iFIX collector
- The iFIX Alarms and Events collector

When you install collectors, if iFIX is installed on the same machine as the collectors, instances of the iFIX collectors are created automatically. This topic describes how to create additional instances if needed.

This topic describes how to add a collector instance using Configuration Hub. You can also add a collector instance using the RemoteCollectorConfigurator utility *(on page 797)*, which does not require you to install Web-based Clients.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**. A list of collectors in the default system appears.
3. In the upper-right corner of the main section, select +.




The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.

4. In the **MACHINE NAME** field, select the machine in which you want to add a collector instance.
5. In the **COLLECTOR TYPE** field, select **iFIX Alarms Events Collector** or **iFIX Collector**, and then select **Get Details**.
   The **INSTALLATION DRIVE** and **DATA DIRECTORY** fields are disabled and populated.
6. Select **Next**.
   The **Source Configuration** section appears. The **iFIX SERVER** field is disabled and populated.
7. Select **Next**.
   The **Destination Configuration** section appears. The **Historian Server** option is selected by default. You cannot select any other option for an iFIX Alarms and Events collector.

   Under **CHOOSE DESTINATION**, the **Historian Server** option is selected by default. In addition, the **DESTINATION HISTORIAN SERVER** field is disabled and populated with the collector machine name.
8. Select the destination to which you want to send data, and then enter the values in the corresponding fields. You can send data to an on-premises Historian server or to a cloud destination.

a. If you need to send data to a cloud destination, select the cloud destinations as needed.
   - **Predix Timeseries**- Select this if you need to send data to Predix cloud. For more information, refer to Predix Cloud *(on page 621)*.
   - **Azure IoT Hub**- Select this if you need to send data to Azure Cloud in KairosDB format. For more information, refer to Azure IoT Hub (KairosDB format) *(on page 608)*.
   - **MQTT**- Select this if you need to send data to any of the following cloud destination.
     - Alibaba cloud. For more information, refer to Alibaba Cloud *(on page 587)*.
     - AWS cloud. For more information, refer to AWS Cloud *(on page 594)*.
     - Google cloud. For more information, refer to Google Cloud *(on page 614)*.

b. If you need to send data to an on-premises Historian server, select **Historian Server**.
   If you created security groups or enabled a strict client/collector authentication, enter the **USERNAME** and **PASSWORD** of the on-premises Historian server that you created during the installation of the collector.
   If you entered the **USERNAME**and **PASSWORD**, select **Test Connection**. This will help you to test if the Historian server that you are trying to connect is valid or if the credentials that you entered are valid.
   If the entered credentials are valid, a successful connection message appears.

9. Select **Next**.
10. In the **RUNNING MODE** field, select one of the following options.
    - **Service - Local System Account**: Select this option if you want to run the collector as a Windows service using the credentials of the local user (that is, the currently logged-in user). If you select this option, the **USERNAME** and **PASSWORD** fields are disabled. By default, this option is selected.
    - **Service Under Specific User Account**: Select this option if you want to run the collector as a Windows service using a specific user account. If you select this option, you must enter values in the **USERNAME** and **PASSWORD** fields.

    If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:
    - iH Security Admins
    - iH Collector Admins
    - iH Tag Admins
11. Select **Next**.
    The **Collector Initiation** section appears.

12. If needed, modify the value in the **COLLECTOR NAME** field.
    The value that you enter:
    - Must be unique.
    - Must not exceed 15 characters.
    - Must not contain a space.
    - Must not contain special characters except a hyphen, period, and an underscore.
13. In the **RUNNING MODE** field, select one of the following options.
    - **Service - Local System Account**: Select this option if you want to run the collector as a Windows service using the credentials of the local user (that is, the currently logged-in user). If you select this option, the **USERNAME** and **PASSWORD** fields are disabled.
    - **Service Under Specific User Account**: Select this option if iFIX is running in a secured mode, or if you want to run the collector as a Windows service using a specific user account. If you select this option, you must enter the credentials of the iFIX user in the **USERNAME** and **PASSWORD** fields.

      If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:
      - iH Security Admins
      - iH Collector Admins
      - iH Tag Admins

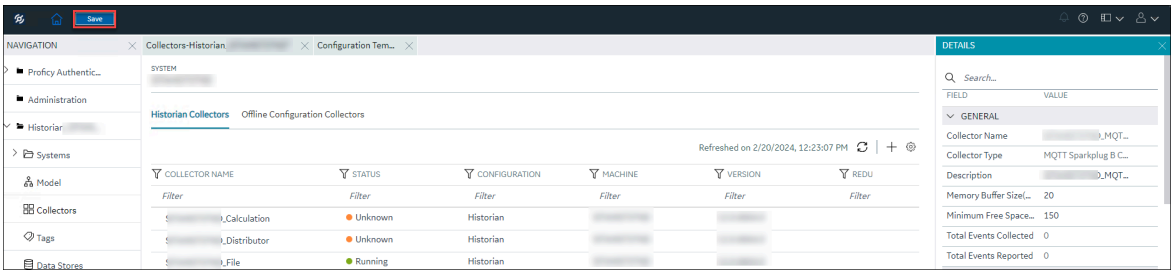
    You can also configure the collector to start automatically when you start iFIX.
14. Select **Add**.
    The collector instance is added, and appears in the Collectors list. A shortcut is created so that you can open it at a command prompt.
    If iFIX is not running in a service mode, an error message may appear. However, the collector instance is created; therefore, you can ignore the error message. Although the collector instance does not appear in the list of collectors in Configuration Hub, a shortcut is created. You can run the collector manually at a command prompt or as a SCU task. For more information, refer to About Adding an iFIX Collector Instance *(on page 501)*.
15. In the **COLLECTOR SPECIFIC CONFIGURATION** section, configure the values as described in the following table.

| Field | Description |
|---|---|
| **Nodes to Browse** | Enter the mask that you want to use to select tags when browsing for tags in the collector. The default value is FIX. |

| Field | Description |
|---|---|
| | If you want to browse for tags on other iFIX nodes via FIX networking, you can enter the other node name(s) here, separated by commas with no spaces. You must have the iFIX system configured for networking. For more information, refer to the iFIX product documentation on iFIX networking.<br><br>**Note:**<br>If you have modified iFIX node name, then you must also update the value in the **Nodes to Browse** field before browsing for tags in the iFIX collector.<br><br>When you browse multiple nodes for tags to add to an iFIX collector, do not use space characters between node names or between the required comma and next node name. All characters after the space are ignored. |
| **Tag Browse Criteria** | Specify the tags for data collection *(on page 357)*.<br><br>**Note:**<br>If you want to add block or field types to the list, edit the `FixTag.dat` file for Historian Administrator you are using. Refer to Editing FixTag.dat File *(on page 2257)* for more information. |
| **MTLS Security** | Indicates whether you want to use Mutual TLS (MTLS) protocol to enforce a secure and strong authentication mechanism. |

| Field | Description |
|---|---|
| **MTLS Data Encryption** | Indicates whether you want to encrypt the data that the collector shares to the data archiver (DA). |
| For more information on how to enable MTLS Security, refer to Enable MTLS Security *(on page 632)*. | |

16. As needed, enter values in the other sections common o all collectors *(on page 579)*.
17. In the upper-left corner of the page, select **Save**.




The changes to the collector instance are saved.

18. If needed, restart the collector.

Specify the tags whose data you want to collect using the collector. In the **CHOOSE CONFIGURATION** field,

- If you have selected **Historian Configuration**, specify the tags using Configuration Hub *(on page 357)*.
- If you have selected **Offline Configuration**, modify the offline configuration file of the collector. By default, the file is available in the following location: `<installation folder of Historian>\GE Digital\<collector name>`. For information, refer to Offline Configuration for Collectors *(on page 2039)*. This option is applicable only if you have selected a cloud destination. This option is not applicable to an iFIX Alarms and Events collector.

## Add and Configure an MQTT Collector

1. Ensure that you have an MQTT broker.

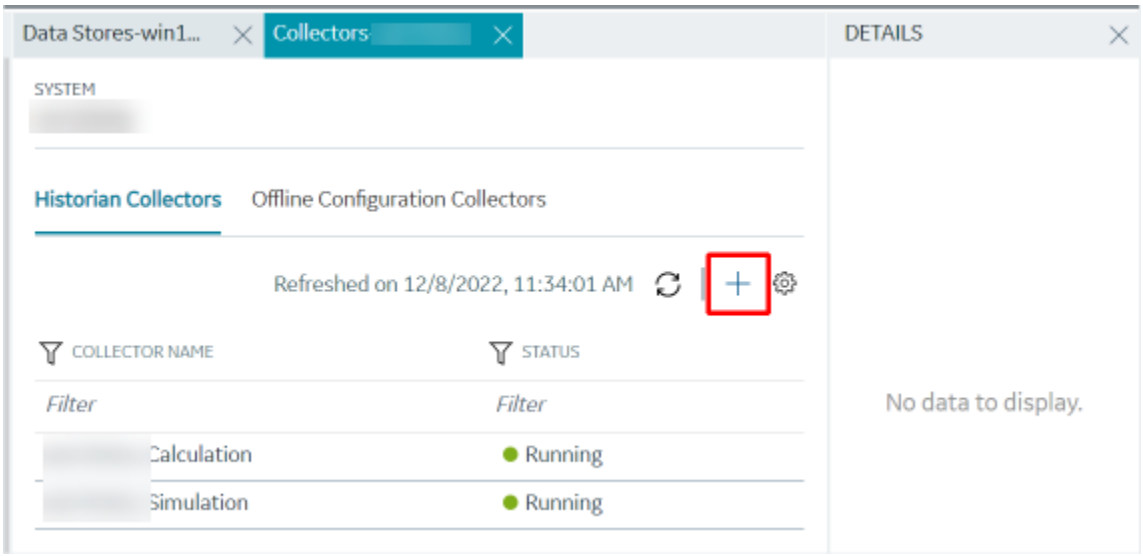
> **Note:**
> We have tested with the MQTT brokers Mosquitto 2.0.15 and HiveMQ-4.2.1. You can, however, use other MQTT brokers as well.

2. If you want to use username/password-based authentication or certificate-based authentication to connect the MQTT broker and the MQTT collector, configure the authentication in the MQTT broker.
3. If you want to use certificate-based authentication, ensure that the following files are available on your collector machine:
    - CA server root file
    - Private key file
    - Client certificate file

The MQTT collector collects data published to a topic using an MQTT broker. For more information, refer to Overview of the MQTT Collector *(on page 2298)*.

This topic describes how to add a collector instance using Configuration Hub. You can also add a collector instance using the RemoteCollectorConfigurator utility *(on page 797)*, which does not require you to install Web-based Clients.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**. A list of collectors in the default system appears.
3. In the upper-right corner of the main section, select +.




The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.

4. In the **MACHINE NAME** field, select the machine in which you want to add a collector instance.
5. In the **COLLECTOR TYPE** field, select **MQTT Collector**, and then select **Get Details**.
   The **INSTALLATION DRIVE** and **DATA DIRECTORY** fields are disabled and populated.

6. Select **Next**.
   The **Source Configuration** section appears.
7. Enter values as described in the following table.

| Field | Description |
|---|---|
| **MQTT BROKER ADDRESS** | Enter the host name of the MQTT broker using which you want to collect data. A value is required. |
| **MQTT BROKER PORT** | Enter the port number of the MQTT broker. A value is required. |
| **TOPIC** | Enter the MQTT topic from which you want to collect data. A value is required. You can enter multiple topics separated by commas.<br><br>If you want to use the Sparkplug B format, enter a value in the following format: `namespace/group_id/message_type/edge_node_id/device_id`<br><br>where:<br>◦ `namespace` is the Sparkplug version. Enter `spBv1.0`.<br>◦ `group_id` is the ID of the group of nodes from which you want to collect data.<br>◦ `message_type` is the message type from which you want to collect data. The collector processes data only from NDATA and DDATA message types.<br>◦ `edge_node_id` is used to identify the MQTT EoN node within the infrastructure.<br>◦ `device_id` a device attached to the MQTT EoN node either physically or logically.<br>You can use the wildcard character # for any of these parameters (except for namespace). |
| **USERNAME** | Enter the username to connect to the MQTT broker. A value is required if you have configured username/password-based authentication in the MQTT broker. |
| **PASSWORD** | Enter the password to connect to the MQTT broker. A value is required if you have configured username/password-based authentication in the MQTT broker. |

| Field | Description |
|---|---|
| **CA SERVER ROOT FILE** | Enter the path to the CA server root file to connect to the MQTT broker. A value is required if you have configured certificate-based authentication in the MQTT broker. |
| **PRIVATE KEY FILE** | Enter the path to the private key file to connect to the MQTT broker. A value is required if you have configured certificate-based authentication in the MQTT broker. |
| **CLIENT CERTIFICATE FILE** | Enter the path to the client certificate file to connect to the MQTT broker. A value is required if you have configured certificate-based authentication in the MQTT broker. |
| **REQUESTED QUALITY OF SERVICE (QOS) LEVEL** | Select the quality of service that you want to use while collecting data from an MQTT broker.<br>◦ **QoS 0**: Indicates that the message is delivered at most once or it is not delivered at all.<br>◦ **QoS 1**: Indicates that the message is always delivered at least once.<br>◦ **QoS 2**: Indicates that the message is delivered once.<br>For more information, refer to https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/. |
| **MQTT VERSION** | Select the version of the MQTT that you want to use. |
| **CLEAN SESSION** | Select one of the following options:<br>◦ **True**: Select this option if you do not want to create a new session when the MQTT broker and the collector are disconnected from each other.<br>◦ **False**: Select this option if you want to retain the session when the MQTT broker and the collector are disconnected from each other. This ensures that there is no loss of data. If you want to choose this option, ensure that you have selected **QoS 1** or **QoS 2** in the **REQUESTED QUALITY OF SERVICE (QOS) LEVEL** field. |
| **SESSION EXPIRY INTERVAL** | Enter the duration, in seconds, after which the data will be discarded when connection between the MQTT broker and collector is re-established. |

| Field | Description |
|---|---|
| | For example, if you enter 100 in this field, and if the MQTT broker and collector are disconnected for 90 seconds, the data is collected. If, however, the MQTT broker and the collector are disconnected for more than 100 seconds, the data will be discarded.<br><br>This field is applicable only for MQTT V5 and only if you set the **CLEAN SESSION** field to **False**. |
| **CONTENT TYPE** | Select the format that you want to use for the payload:<br>◦ **JSON**: Select this option if you want to use the KairosDB format.<br>◦ **SparkPlug B v1.0**: Select this option if you want to use the Sparkplug format. |

8. Select **Next**.
   The **Destination Configuration** section appears. The collector machine name provided by you is selected as the **Source Configuration** by default.

   Under **CHOOSE DESTINATION**, the **Historian Server** option is selected by default. In addition, the **DESTINATION HISTORIAN SERVER** field is disabled and populated with the collector machine name.

9. Select the destination to which you want to send data, and then enter the values in the corresponding fields. You can send data to an on-premises Historian server or to a cloud destination.

   a. If you need to send data to a cloud destination, select the cloud destinations as needed.
      - **Predix Timeseries**- Select this if you need to send data to Predix cloud. For more information, refer to Predix Cloud *(on page 621)*.
      - **Azure IoT Hub**- Select this if you need to send data to Azure Cloud in KairosDB format. For more information, refer to Azure IoT Hub (KairosDB format) *(on page 608)*.
      - **MQTT**- Select this if you need to send data to any of the following cloud destination.
        - Alibaba cloud. For more information, refer to Alibaba Cloud *(on page 587)*.
        - AWS cloud. For more information, refer to AWS Cloud *(on page 594)*.
        - Google cloud. For more information, refer to Google Cloud *(on page 614)*.

   b. If you need to send data to an on-premises Historian server, select **Historian Server**.

If you created security groups or enabled a strict client/collector authentication, enter the **USERNAME** and **PASSWORD** of the on-premises Historian server that you created during the installation of the collector.
If you entered the **USERNAME**and **PASSWORD**, select **Test Connection**. This will help you to test if the Historian server that you are trying to connect is valid or if the credentials that you entered are valid.
If the entered credentials are valid, a successful connection message appears.

10. Select **Next**.
    The **Collector Initiation** section appears.
11. If needed, modify the value in the **COLLECTOR NAME** field. The value must be unique, must contain the string `MQTT`, and must not contain a space.
    The value that you enter:
    - Must be unique.
    - Must contain the string `MQTT`.
    - Must not exceed 15 characters.
    - Must not contain a space.
    - Must not contain special characters except a hyphen, period, and an underscore.
12. In the **RUNNING MODE** field, select one of the following options.
    - **Service - Local System Account**: Select this option if you want to run the collector as a Windows service using the credentials of the local user (that is, the currently logged-in user). If you select this option, the **USERNAME** and **PASSWORD** fields are disabled.
    - **Service Under Specific User Account**: Select this option if you want to run the collector as a Windows service using a specific user account. If you select this option, you must enter values in the **USERNAME** and **PASSWORD** fields.

      If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:
      - iH Security Admins
      - iH Collector Admins
      - iH Tag Admins

    You can also configure the collector to start automatically when you start the computer.
13. Select **Add**.
    The collector instance is added. The fields specific to the collector section appear in the **DETAILS** section.

14. In the **COLLECTOR SPECIFIC CONFIGURATION** and **INSTANCE CONFIGURATION** sections, configure the values as described in the following table.

**COLLECTOR SPECIFIC CONFIGURATION**

| Field | Description |
|---|---|
| **MTLS Security** | Indicates whether you want to use Mutual TLS (MTLS) protocol to enforce a secure and strong authentication mechanism. |
| **MTLS Data Encryption** | Indicates whether you want to encrypt the data that the collector shares to the data archiver (DA). |
| For more information on how to enable MTLS Security, refer to Enable MTLS Security *(on page 632)*. | |

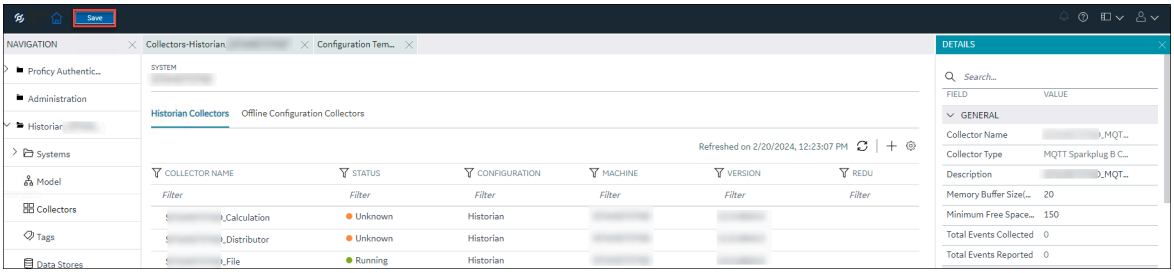
**INSTANCE CONFIGURATION**

| Field | Description |
|---|---|
| **MQTT Broker Address** | Enter the host name of the MQTT broker using which you want to collect data. A value is required. |
| **MQTT Broker Topic** | Enter the MQTT topic from which you want to collect data. A value is required. You can enter multiple topics separated by commas.<br><br>If you want to use the Sparkplug B format, enter a value in the following format: `namespace/group_id/message_type/edge_node_id/device_id`<br><br>where:<br>◦ `namespace` is the Sparkplug version. Enter `spBv1.0`.<br>◦ `group_id` is the ID of the group of nodes from which you want to collect data.<br>◦ `message_type` is the message type from which you want to collect data. The collector processes data only from NDATA and DDATA message types. |

| Field | Description |
|---|---|
| | ◦ `edge_node_id` is used to identify the MQTT EoN node within the infrastructure.<br>◦ `device_id` a device attached to the MQTT EoN node either physically or logically.<br>You can use the wildcard character # for any of these parameters (except for namespace). |
| **MQTT Brker Port** | Enter the port number of the MQTT broker. A value is required. |
| **Username** | Enter the username to connect to the MQTT broker. A value is required if you have configured username/password-based authentication in the MQTT broker. |
| **Password** | Enter the password to connect to the MQTT broker. A value is required if you have configured username/password-based authentication in the MQTT broker. |
| **CA Server Root File** | Enter the path to the CA server root file to connect to the MQTT broker. A value is required if you have configured certificate-based authentication in the MQTT broker. |
| **Private Key File** | Enter the path to the private key file to connect to the MQTT broker. A value is required if you have configured certificate-based authentication in the MQTT broker. |
| **CLIENT Certificate File** | Enter the path to the client certificate file to connect to the MQTT broker. A value is required if you have configured certificate-based authentication in the MQTT broker. |
| **Requested Quality Of Service (QoS) Level** | Select the quality of service that you want to use while collecting data from an MQTT broker.<br>◦ **QoS 0**: Indicates that the message is delivered at most once or it is not delivered at all.<br>◦ **QoS 1**: Indicates that the message is always delivered at least once.<br>◦ **QoS 2**: Indicates that the message is delivered once.<br>For more information, refer to https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/. |
| **MQTT Version** | Select the version of the MQTT that you want to use. |

| Field | Description |
|---|---|
| **CLEAN Session** | Select one of the following options:<br>◦ **True**: Select this option if you do not want to create a new session when the MQTT broker and the collector are disconnected from each other.<br>◦ **False**: Select this option if you want to retain the session when the MQTT broker and the collector are disconnected from each other. This ensures that there is no loss of data. If you want to choose this option, ensure that you have selected **QoS 1** or **QoS 2** in the **REQUESTED QUALITY OF SERVICE (QOS) LEVEL** field. |
| **SESSION Expiry Interval** | Enter the duration, in seconds, after which the data will be discarded when connection between the MQTT broker and collector is re-established.<br><br>For example, if you enter 100 in this field, and if the MQTT broker and collector are disconnected for 90 seconds, the data is collected. If, however, the MQTT broker and the collector are disconnected for more than 100 seconds, the data will be discarded.<br><br>This field is applicable only for MQTT V5 and only if you set the **CLEAN SESSION** field to **False**. |
| **Content Type** | Select the format that you want to use for the payload:<br>◦ **JSON**: Select this option if you want to use the KairosDB format.<br>◦ **SparkPlug B v1.0**: Select this option if you want to use the Sparkplug format. |

15. As needed, enter values in the other sections common to all collectors *(on page 579)*.
16. In the upper-left corner of the page, select **Save**.



The changes to the collector instance are saved.

17. If needed, restart the collector.

Specify the tags whose data you want to collect using the collector. In the **CHOOSE CONFIGURATION** field,

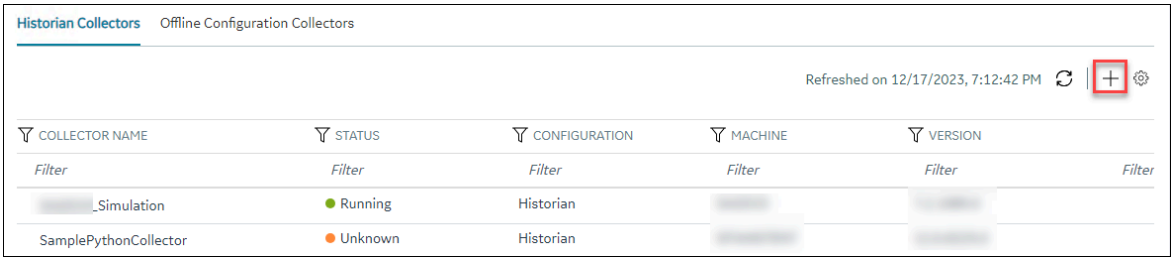
- If you have selected **Historian Configuration**, specify the tags using Configuration Hub *(on page 357)*.
- If you have selected **Offline Configuration**, modify the offline configuration file of the collector. By default, the file is available in the following location: `<installation folder of Historian>\GE Digital\<collector name>`. For information, refer to Offline Configuration for Collectors *(on page 2039)*. This option is applicable only if you have selected a cloud destination.

## Add an MQTT Sparkplug B Collector Instance using Configuration Hub

1. Install the Historian server *(on page 104)* and collectors *(on page 142)*.
2. Ensure that you have an MQTT broker.
3. If you want to use username/password-based authentication or certificate-based authentication to connect the MQTT broker and the MQTT Sparkplug B collector, configure the authentication in the MQTT broker.
4. If you want to use certificate-based authentication, ensure that the following files are available on your collector machine:
    - CA server root file
    - Private key file
    - Client certificate file

This topic describes how to add and configure an MQTT Sparkplug B collector instance using Configuration hub. You can also add and configure an MQTT Sparkplug B collector instance using RemoteCollectorConfigurator.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**. A list of collectors in the default system appears.
3. In the upper-right corner of the main section, select .




The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.

4. In the **MACHINE NAME** field, select the machine in which you want to add a collector instance.
5. In the **COLLECTOR TYPE** field, select **MQTT Sparkplug B Collector**, and then select **Get Details**.

> **Note:**
> The **INSTALLATION DRIVE** and **BASE DATA DIRECTORY** fields cannot be changed. This is the drive location and the data directory folder that you provided during Collectors installation.

The **INSTALLATION DRIVE** and **DATA DIRECTORY** fields are populated with the drive location and the data directory folder.

6. Select **Next**.
   The **Source Configuration** section appears.
7. Enter the values as described in the following table:

| Field | Description |
|---|---|
| **BROKER CONFIGURATION** | |
| **BROKER ADDRESS** | Enter the host name of the MQTT broker using which you want to collect data. A value is required. |
| **BROKER PORT** | Enter the port number of the MQTT broker. A value is required. |
| **CLIENT ID** | Enter the client ID of the MQTT Sparkplug B collector is running. This is required if you want to send the data to a cloud destination. If you do not have a client ID set up, by default, the interface name is taken. |

| Field | Description |
|---|---|
| **PRIMARY HOST ID** | Enter the unique host ID of the Collector. The Collector will publish the STATE message topic using this host ID and then the Publisher will subscribe and start publishing the topics to this host ID. |
| **REORDER TIMEOUT** | Enter the duration for waiting before sending a CMD message if a sequence is skipped. You can enter the duration in milliseconds. |
| **MQTT VERSION** | Select the version of the MQTT that you want to use. The following versions are supported:<br>◦ MQTT_V311<br>◦ MQTT_V5 |

**TOPIC**: The parameters that need to be included in the topic:

```
Namespace/Groupname/<Message Type>/NodeID/<DeviceID>
```

You can also use wildcards in the **GROUP ID**, **EDGE NODE ID**, and **DEVICE ID** fields. The following wildcards are supported:

- **+** (single-level wildcard): Supported for all the three fields.
- **#** (Multi-level wildcard): Supported for the **EDGE NODE ID** and **DEVICE ID**

**+** (Single-level wildcard): Can be used to subscribe to only one topic level. For example, if you subscribe to a topic `<Admin>/+/<ABC-123>`, you will receive messages from all the nodes corresponding to the group and device. That is,

```
<Admin>/Node1/<ABC-123>
<Admin>/Node2/<ABC-123>
<Admin>/Node3/<ABC-123>
...
<Admin>/Noden/<ABC-123>
```

# (Multi-level wildcard): Can be used to subscribe to any number of levels within a topic. For example, if you subscribe to a topic `<Admin>/#`, you will receive messages from all the nodes and devices corresponding to the group,

| Field | Description |
|---|---|
| <Admin>/Node1/<ABC-123><br><Admin>/Node2/<ABC-123><br><Admin>/Node3/<ABC-123><br><Admin>/Node1/<ABC-124<br><Admin>/Node2/<ABC-124><br>...<br><Admin>/Noden/<Devicen> | |
| **GROUP ID** | Enter the Sparkplug B group name to which you want your collector to subscribe. If this is empty along with the other fields below **TOPIC**, the collector will subscribe to all the available groups, nodes, and devices. |
| **EDGE NODE ID** | Enter the Sparkplug B edge node ID to which you want your collector to subscribe. If this is empty, then the Collector will subscribe to all the edge nodes corresponding to the entered **GROUP ID**. If the **GROUP ID** and **DEVICE ID** are also empty, then the collector will subscribe to all the available groups, nodes and devices. |
| **DEVICE ID** | Enter the Sparkplug B device name. If this is empty, the collector subscribes to node messages if a **NODE ID** is entered, otherwise, if a **DEVICE ID** is entered, it subscribes to device messages. |
| **TAG CONFIGURATION** | |
| **TAG NAME PREFIX FORMAT** | |
| **ELEMENT** | Enter a prefix to be included in the tag. By using this field, you can clearly identify a tag. For example, you can clearly differentiate the tags that are collected.<br><br>The following options are available: |

| Field | Description |
|---|---|
| | ◦ **<interfacename>**<br>◦ **<groupid>**<br>◦ **<edgenodeid>**<br>◦ **<deviceid>**<br><br>For example, if all four fields are provided and the interface/collector name is "sparkplug1" and the Topic contains group id = g1 edge node id = n1, device id = d1 then device tags will be created in Historian as “sparkplug1.g1.n1.d1.tag1”. |
| **DELIMITER** | Enter a delimiter you need to be included in the tag. You can use any special characters as delimiter. However, it is recommended that you use a delimiter that is ideal and clear to be identified. For example, "/", ".", "_".<br><br>**Note:**<br>"?" and "*" are not allowed. |
| **PREVIEW** | The preview of how the tags will be created and stored based on the **TAG PREFIX** and the **DELIMTER** that you selected. |
| **TAG MASK** | |
| **TAGS TO ADD** | Provide a mask along with wildcard to collect those tags that include the mask you provided and store in Historian. For example, *Pressure*. This will collect all the tags that begin with "Pressure". If you enter Pressure*, all the tags that end with "Pressure" will be collected. Similarly, if you enter *Pres?, all the tags that contain "pres" at the beginning will be collected. It can be "Pressure", "Press", or "Pres1". |

| Field | Description |
|---|---|
| | **Note:** Whenever a new tag is collected, the collector verifies the tag availability in the Historian and, if not present, adds the tag, then adds the data samples, streaming the data to the Historian server or a cloud destination. |
| **TAGS TO EXCLUDE** | Provide a mask along with wildcard to exclude those tags that include the mask you provided. For example, *Pressure*. This will exclude all the tags that begin with "Pressure". If you enter Pressure*, all the tags that end with "Pressure" will be excluded. Similarly, if you enter *Pres?, all the tags that contain "pres" at the beginning will be excluded. It can be "Pressure", "Press", or "Pres1". |
| **AUTHENTICATION** | |
| **USER CREDENTIALS** | |
| **USERNAME** | Enter the username to connect to the MQTT broker. A value is required if you have configured username/password-based authentication in the MQTT broker. |
| **PASSWORD** | Enter the password to connect to the MQTT broker. A value is required if you have configured username/password-based authentication in the MQTT broker. |
| **SSL/TLS** | |
| **CA SERVER ROOT FILE** | Enter the path to the CA server root file to connect to the MQTT broker. A value is required if you have configured certificate-based authentication in the MQTT broker. |

| Field | Description |
|---|---|
| **PRIVATE KEY FILE** | Enter the path to the private key file to connect to the MQTT broker. A value is required if you have configured certificate-based authentication in the MQTT broker. |
| **CLIENT CERTIFICATE FILE** | Enter the path to the client certificate file to connect to the MQTT broker. A value is required if you have configured certificate-based authentication in the MQTT broker. |

8. Select **Next**.
   The **Destination Configuration** section appears.

   Under **CHOOSE DESTINATION**, the **Historian Server** option is selected by default. In addition, the **DESTINATION HISTORIAN SERVER** field is disabled and populated with the collector machine name.

9. Select the destination to which you want to send data, and then enter the values in the corresponding fields. You can send data to an on-premises Historian server or to a cloud destination.

   a. If you need to send data to a cloud destination, select the cloud destinations as needed.
      - **Predix Timeseries**- Select this if you need to send data to Predix cloud. For more information, refer to Predix Cloud *(on page 621)*.
      - **Azure IoT Hub**- Select this if you need to send data to Azure Cloud in KairosDB format. For more information, refer to Azure IoT Hub (KairosDB format) *(on page 608)*.
      - **MQTT**- Select this if you need to send data to any of the following cloud destination.
        - Alibaba cloud. For more information, refer to Alibaba Cloud *(on page 587)*.
        - AWS cloud. For more information, refer to AWS Cloud *(on page 594)*.
        - Google cloud. For more information, refer to Google Cloud *(on page 614)*.

   b. If you need to send data to an on-premises Historian server, select **Historian Server**.
      If you created security groups or enabled a strict client/collector authentication, enter the **USERNAME** and **PASSWORD** of the on-premises Historian server that you created during the installation of the collector.

If you entered the **USERNAME**and **PASSWORD**, select **Test Connection**. This will help you to test if the Historian server that you are trying to connect is valid or if the credentials that you entered are valid.
If the entered credentials are valid, a successful connection message appears.

10. After you selected the destination, select **Next**.
    The **Collector Initiation** section appears.
11. If needed, modify the value in the **COLLECTOR NAME** field.

    **Note:**
    If you will be using the collector in Historian Administrator, the **COLLECTOR NAME** must include Sparkplug B in it.

    The value that you enter:
    - Must be unique.
    - Must not exceed 15 characters.
    - Must not contain a space.
    - Must not contain special characters except a hyphen, period, and an underscore.
12. In the **RUNNING MODE** field, select one of the following options.
    - **Service - Local System Account**: Select this option if you want to run the collector as a Windows service using the credentials of the local user (that is, the currently logged-in user). If you select this option, the **USERNAME** and **PASSWORD** fields are disabled.
    - **Service Under Specific User Account**: Select this option if you want to run the collector as a Windows service using a specific user account. If you select this option, you must enter values in the **USERNAME** and **PASSWORD** fields.

      If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:
      - iH Security Admins
      - iH Collector Admins
      - iH Tag Admins

    You can also configure the collector to start automatically when you start the computer.
13. Select **Add**.
    The collector instance is added. The fields specific to the collector section appear in the **DETAILS** section.
14. If needed, enter values in available fields *(on page 579)*.

15. In the upper-left corner of the page, select **Save**.
16. If needed, restart the collector.

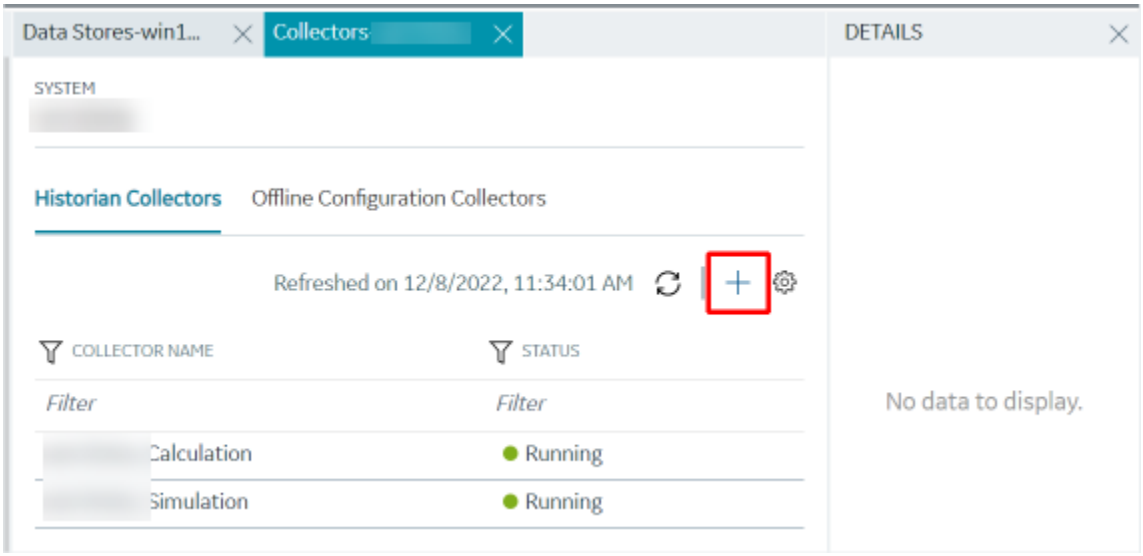
- Specify the tags whose data you want to collect using the collector. In the **CHOOSE CONFIGURATION** field,
  - If you have selected **Historian Configuration**, specify the tags using Configuration Hub *(on page 357)*.
  - If you have selected **Offline Configuration**, modify the offline configuration file of the collector. By default, the file is available in the following location: `<installation folder of Historian>\GE Digital\<collector name>`. For information, refer to Offline Configuration for Collectors *(on page 2039)*. This option is applicable only if you have selected a cloud destination.
- If needed, you can configure the collector instance *(on page 2331)*.

## Add and Configure an ODBC Collector Using Configuration Hub

The ODBC collector collects data from an application based on an ODBC driver. For more information, refer to Overview of the ODBC Collector *(on page 2341)*.

This topic describes how to add a collector instance using Configuration Hub. You can also add a collector instance using the RemoteCollectorConfigurator utility *(on page 797)*, which does not require you to install Web-based Clients.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**. A list of collectors in the default system appears.
3. In the upper-right corner of the main section, select +.




The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.

4. In the **MACHINE NAME** field, select the machine in which you want to add a collector instance.
5. In the **COLLECTOR TYPE** field, select **ODBC Collector**, and then select **Get Details**.
   The **INSTALLATION DRIVE** and **DATA DIRECTORY** fields are disabled and populated.
6. Select **Next**.
   The **Source Configuration** section appears.
7. Enter values as described in the following table.

| Field | Description |
|---|---|
| **ODBC SERVER** | Enter the host name or IP address of the ODBC server from which you want to collect data. A value is required. |
| **USERNAME** | Enter the username to connect to the ODBC server. A value is required. |
| **PASSWORD** | Enter the password to connect to the ODBC server. A value is required. |

8. Select **Next**.
   The **Destination Configuration** section appears. The collector machine name provided by you is selected as the **Source Configuration** by default.

Under **CHOOSE DESTINATION**, the **Historian Server** option is selected by default. In addition, the **DESTINATION HISTORIAN SERVER** field is disabled and populated with the collector machine name.

9. Select the destination to which you want to send data, and then enter the values in the corresponding fields. You can send data to an on-premises Historian server or to a cloud destination.

    a. If you need to send data to a cloud destination, select the cloud destinations as needed.
        - **Predix Timeseries**- Select this if you need to send data to Predix cloud. For more information, refer to Predix Cloud *(on page 621)*.
        - **Azure IoT Hub**- Select this if you need to send data to Azure Cloud in KairosDB format. For more information, refer to Azure IoT Hub (KairosDB format) *(on page 608)*.
        - **MQTT**- Select this if you need to send data to any of the following cloud destination.
            - Alibaba cloud. For more information, refer to Alibaba Cloud *(on page 587)*.
            - AWS cloud. For more information, refer to AWS Cloud *(on page 594)*.
            - Google cloud. For more information, refer to Google Cloud *(on page 614)*.

    b. If you need to send data to an on-premises Historian server, select **Historian Server**.
    If you created security groups or enabled a strict client/collector authentication, enter the **USERNAME** and **PASSWORD** of the on-premises Historian server that you created during the installation of the collector.
    If you entered the **USERNAME**and **PASSWORD**, select **Test Connection**. This will help you to test if the Historian server that you are trying to connect is valid or if the credentials that you entered are valid.
    If the entered credentials are valid, a successful connection message appears.

10. Select **Next**.
    The **Collector Initiation** section appears.
11. If needed, modify the value in the **COLLECTOR NAME** field.
    The value that you enter:
        - Must be unique.
        - Must contain the string `ODBC`.
        - Must not exceed 15 characters.
        - Must not contain a space.
        - Must not contain special characters except a hyphen, period, and an underscore.
12. In the **RUNNING MODE** field, select one of the following options.

- **Service - Local System Account**: Select this option if you want to run the collector as a Windows service using the credentials of the local user (that is, the currently logged-in user). If you select this option, the **USERNAME** and **PASSWORD** fields are disabled.
- **Service Under Specific User Account**: Select this option if you want to run the collector as a Windows service using a specific user account. If you select this option, you must enter values in the **USERNAME** and **PASSWORD** fields.

  If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:
  - iH Security Admins
  - iH Collector Admins
  - iH Tag Admins

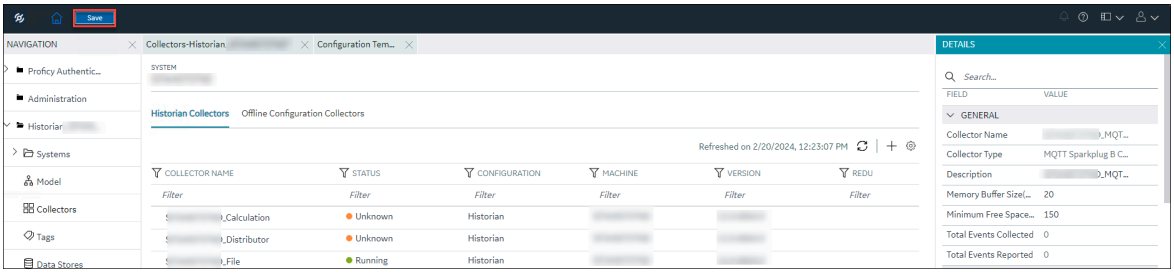
You can also configure the collector to start automatically when you start the computer.

13. Select **Add**.
    The collector instance is added. The fields specific to the collector appear in the **DETAILS** section.
14. In the **COLLECTOR SPECIFIC CONFIGURATION** section, configure values as described in the following table.

| Field | Description |
|---|---|
| **Recovery Time (hours)** | Enter the maximum time, in hours, for which the collector will attempt to recover data after the collector is started or when connection between the collector and the ODBC server is re-established. This time is calculated as the duration between the current time and the last known write time.<br><br>Continuous data collection is resumed only after the previous data has been recovered.<br><br>By default, this value is set to 0, which means data recovery is not attempted. The maximum value you can provide is 168 hours (that is, 7 days). |
| **Throttle (Milliseconds)** | Enter the frequency, in milliseconds, at which you want the ODBC collector to query the ODBC |

| Field | Description |
|---|---|
| | server for tag data. This will minimize the load on the ODBC server. You can enter a value up to 16 hours.<br><br>**Note:**<br>If this field is blank, enter the required minimum value of 100 milliseconds. |
| **MTLS Security** | Indicates whether you want to use Mutual TLS (MTLS) protocol to enforce a secure and strong authentication mechanism. |
| **MTLS Data Encryption** | Indicates whether you want to encrypt the data that the collector shares to the data archiver (DA). |
| For more information on how to enable MTLS Security, refer to Enable MTLS Security *(on page 632)*. | |

15. As needed, enter values in the other sections common to all collectors *(on page 579)*.
16. In the upper-left corner of the page, select **Save**.




The changes to the collector instance are saved.

17. If needed, restart the collector.

Specify the tags whose data you want to collect using the collector. In the **CHOOSE CONFIGURATION** field,

- If you have selected **Historian Configuration**, specify the tags using Configuration Hub *(on page 357)*.
- If you have selected **Offline Configuration**, modify the offline configuration file of the collector. By default, the file is available in the following location: *<installation folder of*

*Historian*>\GE Digital\<*collector name*>. For information, refer to Offline Configuration for Collectors *(on page 2039)*. This option is applicable only if you have selected a cloud destination.

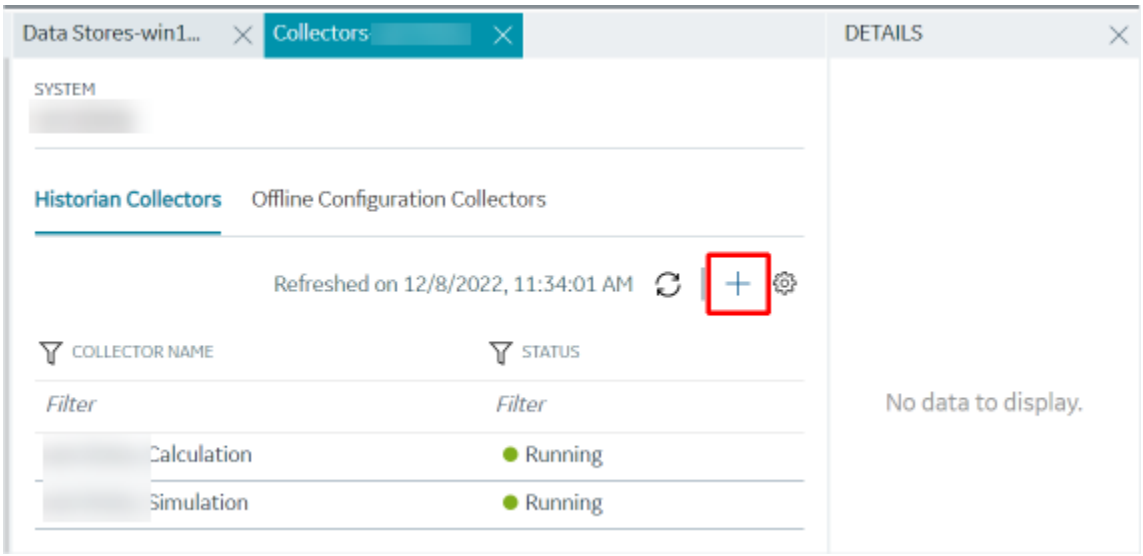
## Add and Configure an OPC Classic Alarms and Events Collector

The OPC Classic Alarms and Events collector collects alarms and events data from any OPC 1.0 or OPC 2.0 compliant OPC server.

You can create an OPC Alarms and Events collector only for an on-premises Historian server, not for a cloud destination.

This topic describes how to add a collector instance using Configuration Hub. You can also add a collector instance using the RemoteCollectorConfigurator utility *(on page 797)*, which does not require you to install Web-based Clients.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**. A list of collectors in the default system appears.
3. In the upper-right corner of the main section, select +.




The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.

4. In the **MACHINE NAME** field, select the machine in which you want to add a collector instance.
5. In the **COLLECTOR TYPE** field, select **OPC Alarms and Events Collector**, and then select **Get Details**.

The **INSTALLATION DRIVE** and **DATA DIRECTORY** fields are disabled and populated.

6. Select **Next**.
   The **Source Configuration** section appears.
7. In the **OPC A&E SERVER** field, enter the host name or IP address of the OPC server from which you want to collect alarms and events data.
8. Select **Next**.
   The **Destination Configuration** section appears. Under **CHOOSE DESTINATION**, the **Historian Server** option is selected by default; the other options are disabled. In addition, the **DESTINATION HISTORIAN SERVER** field is disabled and populated with the value you selected in the **MACHINE NAME** field in the **Collector Selection** section.
9. If you created security groups or enabled a strict client/collector authentication, enter the **USERNAME** and **PASSWORD** of the on-premises Historian server that you created during the installation of the collector.
   If you entered the **USERNAME**and **PASSWORD**, select **Test Connection**. This will help you to test if the Historian server that you are trying to connect is valid or if the credentials that you entered are valid.
10. Select **Next**.
    The **Collector Initiation** section appears.
11. If needed, modify the value in the **COLLECTOR NAME** field.
    The value that you enter:
    - Must be unique.
    - Must not exceed 15 characters.
    - Must not contain a space.
    - Must not contain special characters except a hyphen, period, and an underscore.
12. In the **RUNNING MODE** field, select one of the following options.
    - **Service - Local System Account**: Select this option if you want to run the collector as a Windows service using the credentials of the local user (that is, the currently logged-in user). If you select this option, the **USERNAME** and **PASSWORD** fields are disabled.
    - **Service Under Specific User Account**: Select this option if you want to run the collector as a Windows service using a specific user account. If you select this option, you must enter values in the **USERNAME** and **PASSWORD** fields.

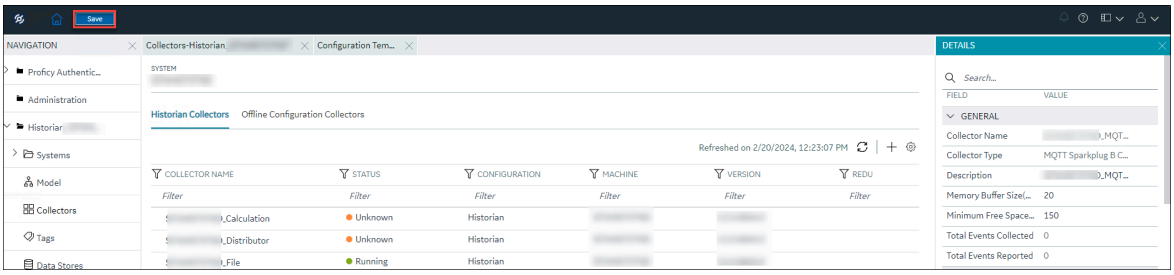
      If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:
      - iH Security Admins
      - iH Collector Admins
      - iH Tag Admins

You can also configure the collector to start automatically when you start the computer.

13. Select **Add**.
    The collector instance is added. The fields specific to the collector section appear in the **DETAILS** section.
14. In the **COLLECTOR SPECIFIC CONFIGURATION** section, configure the values as described in the following table.

| Field | Description |
|---|---|
| **MTLS Security** | Indicates whether you want to use Mutual TLS (MTLS) protocol to enforce a secure and strong authentication mechanism. |
| **MTLS Data Encryption** | Indicates whether you want to encrypt the data that the collector shares to the data archiver (DA). |
| For more information on how to enable MTLS Security, refer to Enable MTLS Security *(on page 632)*. | |

15. As needed, enter values in the other sections common to all collectors *(on page 579)*.
16. In the upper-left corner of the page, select **Save**.




    The changes to the collector instance are saved.
17. If needed, restart the collector.

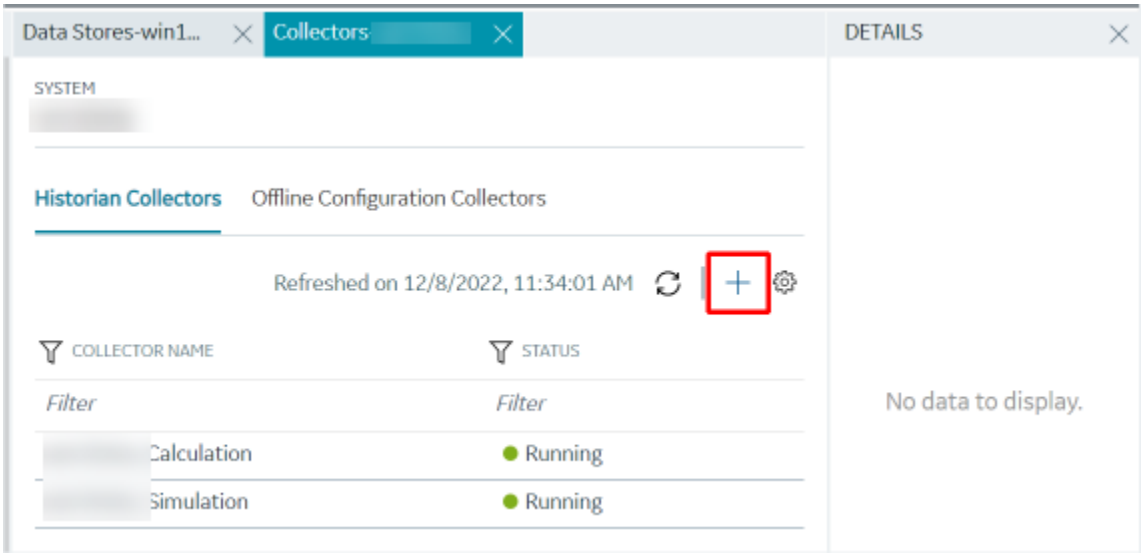
Specify the tags *(on page 357)* whose data you want to collect using the collector.

## Add and Configure an OPC Classic Data Access Collector

The OPC Classic Data Access (DA) collector collects data from any OPC 1.0 or OPC 2.0 compliant OPC Classic server. For more information, refer to Overview of the OPC Classic DA Collector *(on page 2356)*.

This topic describes how to add a collector instance using Configuration Hub. You can also add a collector instance using the RemoteCollectorConfigurator utility *(on page 797)*, which does not require you to install Web-based Clients.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.
   A list of collectors in the default system appears.
3. In the upper-right corner of the main section, select +.




   The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.
4. In the **MACHINE NAME** field, select the machine in which you want to add a collector instance.
5. In the **COLLECTOR TYPE** field, select **OPC Collector**, and then select **Get Details**.
   The **INSTALLATION DRIVE** and **DATA DIRECTORY** fields are disabled and populated.
6. Select **Next**.
   The **Source Configuration** section appears.
7. Enter values as described in the following table.

| Field | Description |
|---|---|
| **OPC SERVER** | Select the machine on which you have installed the OPC Classic DA server from which you want to collect data. |

| Field | Description |
|---|---|
| **MACHINE NAME** | Enter the host name or IP address of the OPC server. This field appears only if you have selected a remote OPC server. A value is required. |
| **OPC DA SERVER PROG ID** | Enter the prog ID of the OPC server. This field appears only if you have selected a remote OPC server. A value is required. |

8. Select **Next**.
   The **Destination Configuration** section appears. The collector machine name provided by you is selected as the **Source Configuration** by default.

   Under **CHOOSE DESTINATION**, the **Historian Server** option is selected by default. In addition, the **DESTINATION HISTORIAN SERVER** field is disabled and populated with the collector machine name.

9. Select the destination to which you want to send data, and then enter the values in the corresponding fields. You can send data to an on-premises Historian server or to a cloud destination.

   a. If you need to send data to a cloud destination, select the cloud destinations as needed.
      - **Predix Timeseries**- Select this if you need to send data to Predix cloud. For more information, refer to Predix Cloud *(on page 621)*.
      - **Azure IoT Hub**- Select this if you need to send data to Azure Cloud in KairosDB format. For more information, refer to Azure IoT Hub (KairosDB format) *(on page 608)*.
      - **MQTT**- Select this if you need to send data to any of the following cloud destination.
        - Alibaba cloud. For more information, refer to Alibaba Cloud *(on page 587)*.
        - AWS cloud. For more information, refer to AWS Cloud *(on page 594)*.
        - Google cloud. For more information, refer to Google Cloud *(on page 614)*.

   b. If you need to send data to an on-premises Historian server, select **Historian Server**.
      If you created security groups or enabled a strict client/collector authentication, enter the **USERNAME** and **PASSWORD** of the on-premises Historian server that you created during the installation of the collector.

If you entered the **USERNAME**and **PASSWORD**, select **Test Connection**. This will help you to test if the Historian server that you are trying to connect is valid or if the credentials that you entered are valid.
If the entered credentials are valid, a successful connection message appears.

10. Select **Next**.
    The **Collector Initiation** section appears. The **COLLECTOR NAME** field is populated with a value in the following format: `<system name>_OPC_<OPC server name>`
11. If needed, modify the value in the **COLLECTOR NAME** field.
    The value that you enter:
    - Must be unique.
    - Must not exceed 15 characters.
    - Must not contain a space.
    - Must not contain special characters except a hyphen, period, and an underscore.
12. In the **RUNNING MODE** field, select one of the following options.
    - **Service - Local System Account**: Select this option if you want to run the collector as a Windows service using the credentials of the local user (that is, the currently logged-in user). By default this option is selected, and the **USERNAME** and **PASSWORD** fields are disabled.
    - **Service Under Specific User Account**: Select this option if you want to run the collector as a Windows service using a specific user account. If you select this option, you must enter values in the **USERNAME** and **PASSWORD** fields.

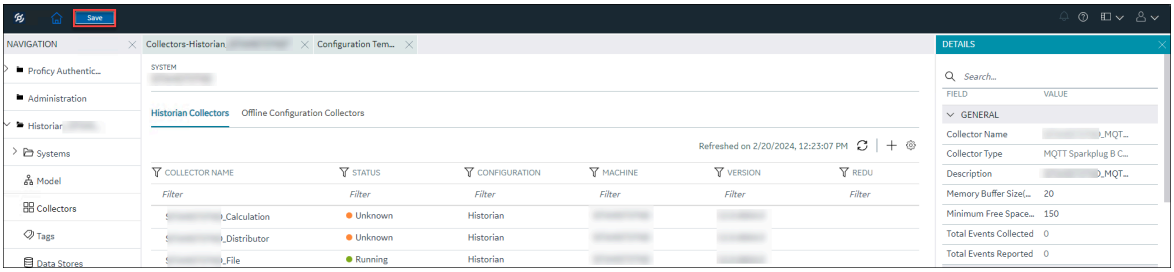
      If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:
      - iH Security Admins
      - iH Collector Admins
      - iH Tag Admins

    You can also configure the collector to start automatically when you start the computer.
13. Select **Add**.
    The collector instance is added. The fields specific to the collector section appear in the **DETAILS** section.
14. Under **COLLECTOR SPECIFIC CONFIGURATION**, configure values as described in the following table.

| Field | Description |
|---|---|
| **OPC Server Prog ID** | The program ID of the OPC server from which you want to collect data. |
| **Read Mode** | The read mode that you want the collector to use. For information, refer to the documentation of the OPC server that you are using or the OPC specification on the OPC Foundation website. |
| **First Browse Criteria** | A comma-separated first-level search criterion for browsing tags from the data source. The top-level and second-level criteria are used together by the AND operation to browse tags.<br><br>For example, if you enter `USGB014` in the **First Browse Criteria** field and `F_CV, B_CUALM` in the **Second Browse Criteria** field, it returns all the tags that contain:<br>◦ USGB014<br><br>-and-<br><br>◦ F_CV or B_CUALM |
| **Second Browse Criteria** | A comma-separated second-level search criterion for browsing tags from the data source. The top-level and second-level criteria are used together by the AND operation to browse tags. |
| **Threading Model** | The type of the threading model selected for the collector. The model selected must match the threading model of the OPC server.<br>◦ **Multithreaded**: Select this option for better performance. We recommend that you configure your collector to use the default multi-threading model.<br>◦ **Apparent**: Select this option for best compatibility. Some OPC servers do not work well with multi-threading. If you experience problems running your collector with multi-threading, use the apartment model. |

| Field | Description |
|---|---|
| | The default setting is multi-threaded. For information, refer to the documentation of the OPC server you are using. |
| **Configuration Changes** | Indicates whether the collector configuration changes are processed in real time or after restarting the collector.<br>◦ **Made On-Line**: Select this option to process any configuration changes immediately (after 30 seconds) after you select the **Update** button.<br>**Note:**<br>▪ Some OPC servers cannot handle processing configuration changes online. If you experience any instability with changes made online, use the next option.<br>◦ **Made After Collector Restart**: Select this option to hold all configuration changes until you manually restart the collector. |
| **MTLS Security** | Indicates whether you want to use Mutual TLS (MTLS) protocol to enforce a secure and strong authentication mechanism. |
| **MTLS Data Encryption** | Indicates whether you want to encrypt the data that the collector shares to the data archiver (DA). |
| For more information on how to enable MTLS Security, refer to Enable MTLS Security *(on page 632)*. | |

15. As needed, enter values in the other sections common to all collectors *(on page 579)*.
16. In the upper-left corner of the page, select **Save**.



The changes to the collector instance are saved.

17. If needed, restart the collector.

Specify the tags whose data you want to collect using the collector. In the **CHOOSE CONFIGURATION** field,

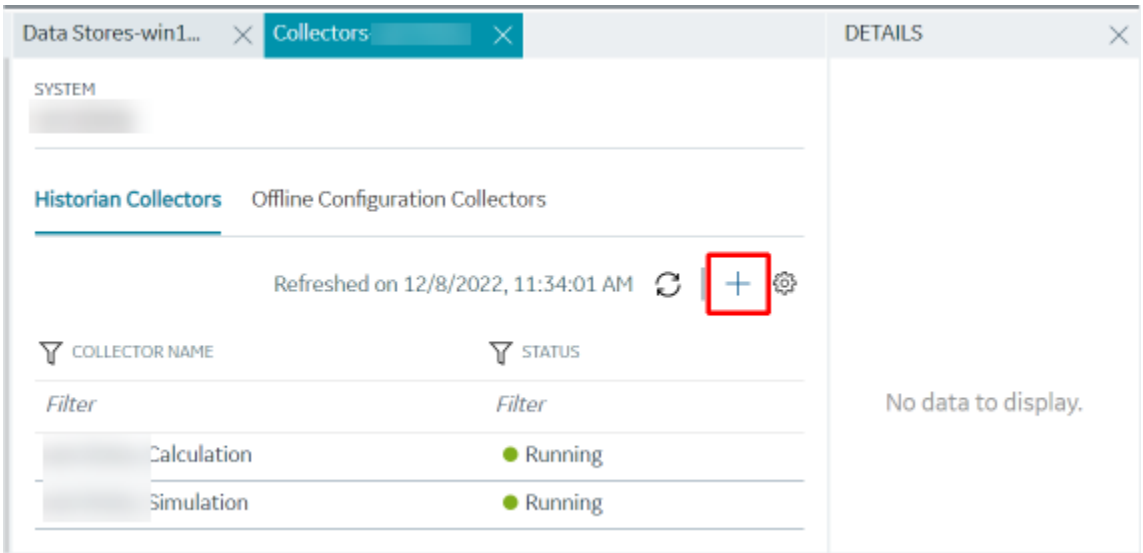
- If you have selected **Historian Configuration**, specify the tags for data collection *(on page 357)*.
- If you have selected **Offline Configuration**, modify the offline configuration file of the collector. By default, the file is available in the following location: `<installation folder of Historian>\GE Digital\<collector name>`. For information, refer to Offline Configuration for Collectors *(on page 2039)*. This option is applicable only if you have selected a cloud destination.

## Add and Configure an OPC Classic HDA Collector

The OPC Classic Historical Data Access (HDA) collector collects data from any OPC HDA 1.2 - compliant OPC Classic HDA server. For more information, refer to Configure the OPC Classic HDA Collector Using Historian Administrator *(on page 2380)*.

This topic describes how to add a collector instance using Configuration Hub. You can also add a collector instance using the RemoteCollectorConfigurator utility *(on page 797)*, which does not require you to install Web-based Clients.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.
3. In the upper-right corner of the main section, select +.



The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.

4. In the **MACHINE NAME** field, select the machine in which you want to add a collector instance.
5. In the **COLLECTOR TYPE** field, select **OPC HDA Collector**, and then select **Get Details**.
   The **INSTALLATION DRIVE** and **DATA DIRECTORY** fields are disabled and populated.
6. Select **Next**.
   The **Source Configuration** section appears.
7. Enter values as described in the following table.

| Field | Description |
|---|---|
| **OPC HDA SERVER** | Select the machine on which you have installed the OPC Classic HDA server from which you want to collect data. |
| **MACHINE NAME** | Enter the host name or IP address of the OPC server. This field appears only if you have selected a remote OPC server. A value is required. |
| **OPC DA SERVER PROG ID** | Enter the prog ID of the OPC server. This field appears only if you have selected a remote OPC server. A value is required. |

8. Select **Next**.
   The **Destination Configuration** section appears. The collector machine name provided by you is selected as the **Source Configuration** by default.

   Under **CHOOSE DESTINATION**, the **Historian Server** option is selected by default. In addition, the **DESTINATION HISTORIAN SERVER** field is disabled and populated with the collector machine name.

9. Select the destination to which you want to send data, and then enter the values in the corresponding fields. You can send data to an on-premises Historian server or to a cloud destination.

a. If you need to send data to a cloud destination, select the cloud destinations as needed.
   - **Predix Timeseries**- Select this if you need to send data to Predix cloud. For more information, refer to Predix Cloud *(on page 621)*.
   - **Azure IoT Hub**- Select this if you need to send data to Azure Cloud in KairosDB format. For more information, refer to Azure IoT Hub (KairosDB format) *(on page 608)*.
   - **MQTT**- Select this if you need to send data to any of the following cloud destination.
     - Alibaba cloud. For more information, refer to Alibaba Cloud *(on page 587)*.
     - AWS cloud. For more information, refer to AWS Cloud *(on page 594)*.
     - Google cloud. For more information, refer to Google Cloud *(on page 614)*.

b. If you need to send data to an on-premises Historian server, select **Historian Server**.
   If you created security groups or enabled a strict client/collector authentication, enter the **USERNAME** and **PASSWORD** of the on-premises Historian server that you created during the installation of the collector.
   If you entered the **USERNAME**and **PASSWORD**, select **Test Connection**. This will help you to test if the Historian server that you are trying to connect is valid or if the credentials that you entered are valid.
   If the entered credentials are valid, a successful connection message appears.

10. Select **Next**.
    The **Collector Initiation** section appears.
11. If needed, modify the value in the **COLLECTOR NAME** field.
    The value that you enter:
    - Must be unique.
    - Must contain the string `OPCHDA`.
    - Must not exceed 15 characters.
    - Must not contain a space.
    - Must not contain special characters except a hyphen, period, and an underscore.
12. In the **RUNNING MODE** field, select one of the following options.
    - **Service - Local System Account**: Select this option if you want to run the collector as a Windows service using the credentials of the local user (that is, the currently logged-in user). By default, this option is selected, and the **USERNAME** and **PASSWORD** fields are disabled.
    - **Service Under Specific User Account**: Select this option if you want to run the collector as a Windows service using a specific user account. If you select this option, you must enter values in the **USERNAME** and **PASSWORD** fields.

If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:

- iH Security Admins
- iH Collector Admins
- iH Tag Admins

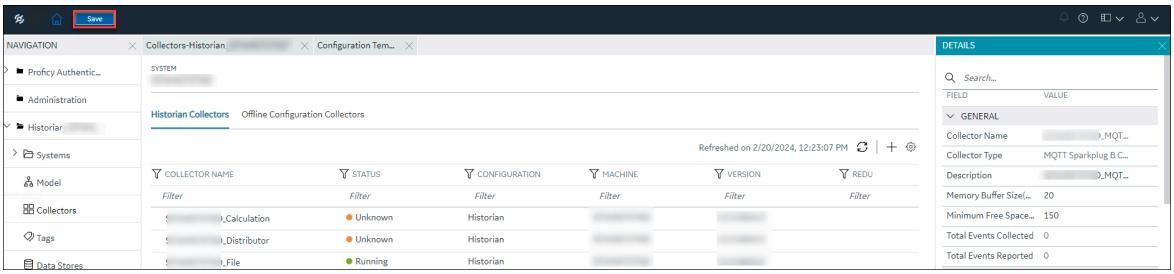
You can also configure the collector to start automatically when you start the computer.

13. Select **Add**.
    The collector instance is added. The fields specific to the collector section appear in the **DETAILS** section.
14. As needed, under **COLLECTOR SPECIFIC CONFIGURATION**, configure values as described in the following table.

| Field | Description |
|---|---|
| **Recovery Time** | It indicates the maximum time, in hours, for which the collector will attempt to recover data after the collector is started or when connection between the collector and the OPC server is re-established. This time is calculated as the duration between the current time and the last known write time.<br><br>Continuous data collection is resumed only after the previous data has been recovered.<br><br>You can enter a value between 1 and 150. |
| **MTLS Security** | Indicates whether you want to use Mutual TLS (MTLS) protocol to enforce a secure and strong authentication mechanism. |
| **MTLS Data Encryption** | Indicates whether you want to encrypt the data that the collector shares to the data archiver (DA). |
| For more information on how to enable MTLS Security, refer to Enable MTLS Security *(on page 632)*. | |

15. As needed, enter values in the other sections common to all collectors *(on page 579)*.

16. In the upper-left corner of the page, select **Save**.




    The changes to the collector instance are saved.
17. If needed, restart the collector.

Specify the tags whose data you want to collect using the collector. In the **CHOOSE CONFIGURATION** field,

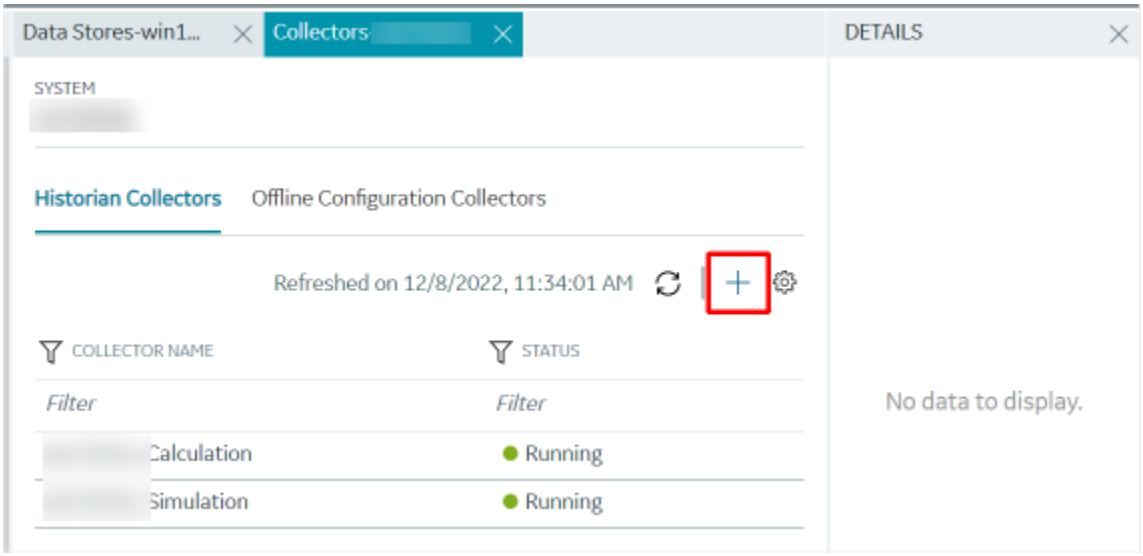
- If you have selected **Historian Configuration**, specify the tags using Configuration Hub *(on page 357)*.
- If you have selected **Offline Configuration**, modify the offline configuration file of the collector. By default, the file is available in the following location: `<installation folder of Historian>\GE Digital\<collector name>`. For information, refer to Offline Configuration for Collectors *(on page 2039)*. This option is applicable only if you have selected a cloud destination.

## Add and Configure an OPC UA Data Access Collector

The OPC UA Data Access (DA) collector gathers and collects data from a OPC UA 1.0-compliant OPC UA DA server. For more information, refer to Configure an OPC UA DA Collector Using Historian Administrator *(on page 2432)*.

This topic describes how to add a collector instance using Configuration Hub. You can also add a collector instance using the RemoteCollectorConfigurator utility *(on page 797)*, which does not require you to install Web-based Clients.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**. A list of collectors in the default system appears.
3. In the upper-right corner of the main section, select +.




The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.

4. In the **MACHINE NAME** field, select the machine in which you want to add a collector instance.
5. In the **COLLECTOR TYPE** field, select **OPC UA DA Collector**, and then select **Get Details**.
   The **INSTALLATION DRIVE** and **DATA DIRECTORY** fields are disabled and populated.
6. Select **Next**.
   The **Source Configuration** section appears.
7. In the **OPC UA SERVER URI** field, enter the URI to connect to the OPC server in the following format:
   `opc.tcp://<host name or IP address of the OPC UA server>:<port number>`
8. Select **Next**.
   The **Destination Configuration** section appears. The collector machine name provided by you is selected as the **Source Configuration** by default.

   Under **CHOOSE DESTINATION**, the **Historian Server** option is selected by default. In addition, the **DESTINATION HISTORIAN SERVER** field is disabled and populated with the collector machine name.
9. Select the destination to which you want to send data, and then enter the values in the corresponding fields. You can send data to an on-premises Historian server or to a cloud destination.

a. If you need to send data to a cloud destination, select the cloud destinations as needed.
   - **Predix Timeseries**- Select this if you need to send data to Predix cloud. For more information, refer to Predix Cloud *(on page 621)*.
   - **Azure IoT Hub**- Select this if you need to send data to Azure Cloud in KairosDB format. For more information, refer to Azure IoT Hub (KairosDB format) *(on page 608)*.
   - **MQTT**- Select this if you need to send data to any of the following cloud destination.
     - Alibaba cloud. For more information, refer to Alibaba Cloud *(on page 587)*.
     - AWS cloud. For more information, refer to AWS Cloud *(on page 594)*.
     - Google cloud. For more information, refer to Google Cloud *(on page 614)*.

b. If you need to send data to an on-premises Historian server, select **Historian Server**.
   If you created security groups or enabled a strict client/collector authentication, enter the **USERNAME** and **PASSWORD** of the on-premises Historian server that you created during the installation of the collector.
   If you entered the **USERNAME**and **PASSWORD**, select **Test Connection**. This will help you to test if the Historian server that you are trying to connect is valid or if the credentials that you entered are valid.
   If the entered credentials are valid, a successful connection message appears.

10. Select **Next**.
    The **Collector Initiation** section appears. The **COLLECTOR NAME** field is populated with a value in the following format: `<Historian server name>_OPCUACollector_<number>`
11. If needed, modify the value in the **COLLECTOR NAME** field.
    The value that you enter:
    - Must be unique.
    - Must contain the string `OPCUACollector`.
    - Must not exceed 15 characters.
    - Must not contain a space.
    - Must not contain special characters except a hyphen, period, and an underscore.
12. In the **RUNNING MODE** field, select one of the following options.
    - **Service - Local System Account**: Select this option if you want to run the collector as a Windows service using the credentials of the local user (that is, the currently logged-in user). If you select this option, the **USERNAME** and **PASSWORD** fields are disabled.
    - **Service Under Specific User Account**: Select this option if you want to run the collector as a Windows service using a specific user account. If you select this option, you must enter values in the **USERNAME** and **PASSWORD** fields.

If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:

- iH Security Admins
- iH Collector Admins
- iH Tag Admins

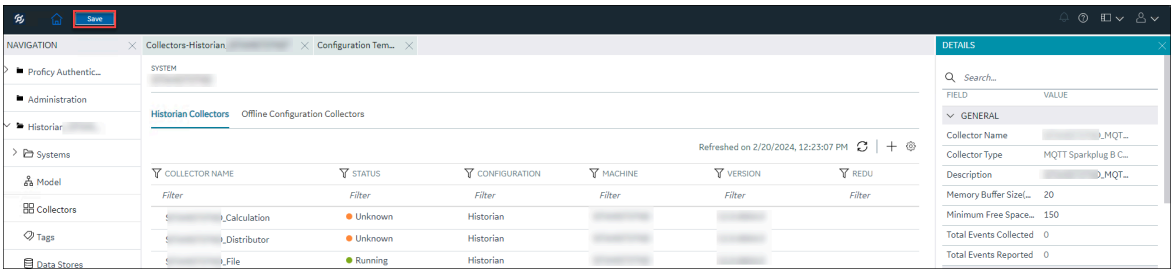
You can also configure the collector to start automatically when you start the computer.

13. Select **Add**.
    The collector instance is added. The fields specific to the collector section appear in the **DETAILS** section.
14. In the **COLLECTOR SPECIFIC CONFIGURATION** section, configure values as described in the following table.

| Field | Description |
|---|---|
| **OPC UA Server URL** | The URI to connect to the OPC UA server. Enter a value in the following format: `opc.tcp://<host name or IP address of the OPC UA server>:<port number>` |
| **Secured Connectivity** | Indicates whether you want a secured connection between the OPC UA server and the collector. By default, this field is set to false.<br><br>You can establish a secured connectivity in one of the following ways:<br>◦ **Using certificates:** To use certificates, switch off the **User Security** toggle.<br>◦ **Using user authentication:** To use user authentication, switch on the **User Security** toggle. |
| **User Security** | This field is enabled only if you have enabled secured connectivity. Switch on this toggle if you want to use user authentication to connect to the OPC server. When you do so, the **User Name** and **Password** fields are enabled. You can either enter the user credentials in these fields, or you can use the values in the `ClientConfig.ini` file. For instructions, refer to Connect with the OPC UA DA Server Securely *(on page 2434)*. |

| Field | Description |
|---|---|
| **Username** | This field is enabled only if you have set the secured connectivity to true and switched on the **User Security** toggle. Enter the username that you want to use to connect to the OPC server. If you do not provide a value, the username from the `ClientConfig.ini` file is considered. |
| **Password** | This field id enabled only if you have set the secured connectivity to true and selected the **Enable User Security** check box. Enter the password that you want to use to connect to the OPC server. If you do not provide a value, the password from the `ClientConfig.ini` file is considered. |
| **MTLS Security** | Indicates whether you want to use Mutual TLS (MTLS) protocol to enforce a secure and strong authentication mechanism. |
| **MTLS Data Encryption** | Indicates whether you want to encrypt the data that the collector shares to the data archiver (DA). |
| For more information on how to enable MTLS Security, refer to Enable MTLS Security *(on page 632)*. | |

15. As needed, enter values in the other sections common to all collectors *(on page 579)*.
16. In the upper-left corner of the page, select **Save**.




The changes to the collector instance are saved.

17. If needed, restart the collector.

If you have enabled secured connection, establish a secured connection between the OPC server and the collector *(on page 2434)*.

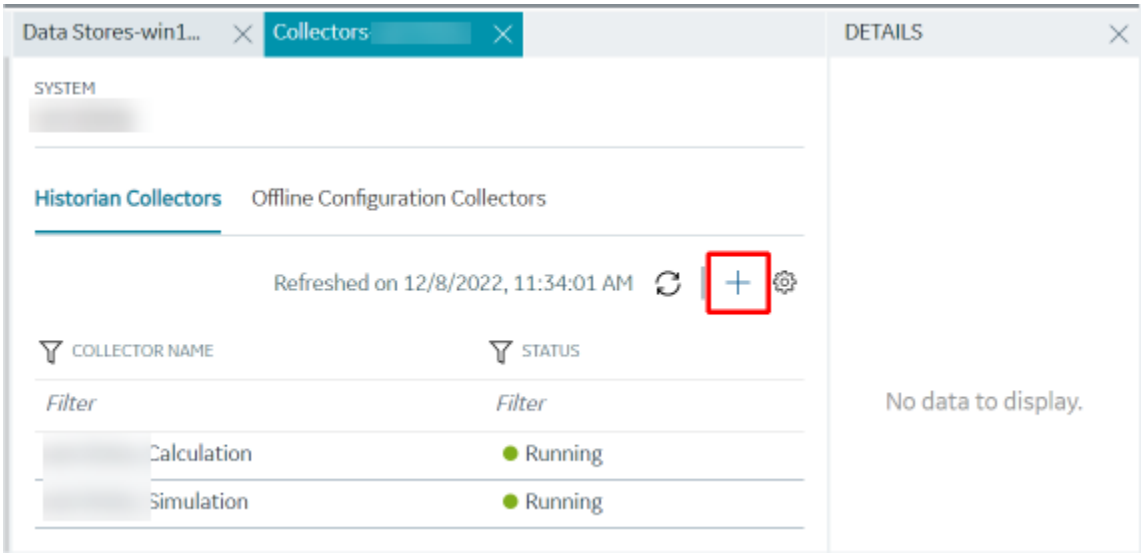
## Add and Configure an OSI PI Collector

Install PI AF SDK version 2.7.5 or later.

The OSI PI collector collects data from an OSI PI server. For more information, refer to Overview of the OSI PI Collector *(on page 2443)*.

This topic describes how to add a collector instance using Configuration Hub. You can also add a collector instance using the RemoteCollectorConfigurator utility *(on page 797)*, which does not require you to install Web-based Clients.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.
   A list of collectors in the default system appears.
3. In the upper-right corner of the main section, select +.




   The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.
4. In the **MACHINE NAME** field, select the machine in which you want to add a collector instance.
5. In the **COLLECTOR TYPE** field, select **OSI PI Collector**, and then select **Get Details**.
   The **INSTALLATION DRIVE** and **DATA DIRECTORY** fields are disabled and populated.
6. Select **Next**.
   The **Source Configuration** section appears.
7. Enter values as described in the following table.

| Field | Description |
|---|---|
| **PI SERVER** | Enter the host name or IP address of the OSI PI server from which you want to collect data. A value is required. |
| **PI USERNAME** | Enter the username to connect to the OSI PI server. |
| **PI PASSWORD** | Enter the password to connect to the OSI PI server. |

8. Select **Next**.
   The **Destination Configuration** section appears. The collector machine name provided by you is selected as the **Source Configuration** by default.

   Under **CHOOSE DESTINATION**, the **Historian Server** option is selected by default. In addition, the **DESTINATION HISTORIAN SERVER** field is disabled and populated with the collector machine name.

9. Select the destination to which you want to send data, and then enter the values in the corresponding fields. You can send data to an on-premises Historian server or to a cloud destination.

   a. If you need to send data to a cloud destination, select the cloud destinations as needed.
      - **Predix Timeseries**- Select this if you need to send data to Predix cloud. For more information, refer to Predix Cloud *(on page 621)*.
      - **Azure IoT Hub**- Select this if you need to send data to Azure Cloud in KairosDB format. For more information, refer to Azure IoT Hub (KairosDB format) *(on page 608)*.
      - **MQTT**- Select this if you need to send data to any of the following cloud destination.
        - Alibaba cloud. For more information, refer to Alibaba Cloud *(on page 587)*.
        - AWS cloud. For more information, refer to AWS Cloud *(on page 594)*.
        - Google cloud. For more information, refer to Google Cloud *(on page 614)*.

   b. If you need to send data to an on-premises Historian server, select **Historian Server**.
      If you created security groups or enabled a strict client/collector authentication, enter the **USERNAME** and **PASSWORD** of the on-premises Historian server that you created during the installation of the collector.

If you entered the **USERNAME**and **PASSWORD**, select **Test Connection**. This will help you to test if the Historian server that you are trying to connect is valid or if the credentials that you entered are valid.
If the entered credentials are valid, a successful connection message appears.

10. Select **Next**.
    The **Collector Initiation** section appears. The **COLLECTOR NAME** field is populated with a value in the following format: `<host name or IP address of the PI server>_PICollector`
11. If needed, modify the value in the **COLLECTOR NAME** field.
    The value that you enter:
    - Must be unique.
    - Must not exceed 15 characters.
    - Must not contain a space.
    - Must not contain special characters except a hyphen, period, and an underscore.
12. In the **RUNNING MODE** field, select one of the following options.
    - **Service - Local System Account**: Select this option if you want to run the collector as a Windows service using the credentials of the local user (that is, the currently logged-in user). By default, this option is selected, and the **USERNAME** and **PASSWORD** fields are disabled.
    - **Service Under Specific User Account**: Select this option if you want to run the collector as a Windows service using a specific user account. If you select this option, you must enter values in the **USERNAME** and **PASSWORD** fields.

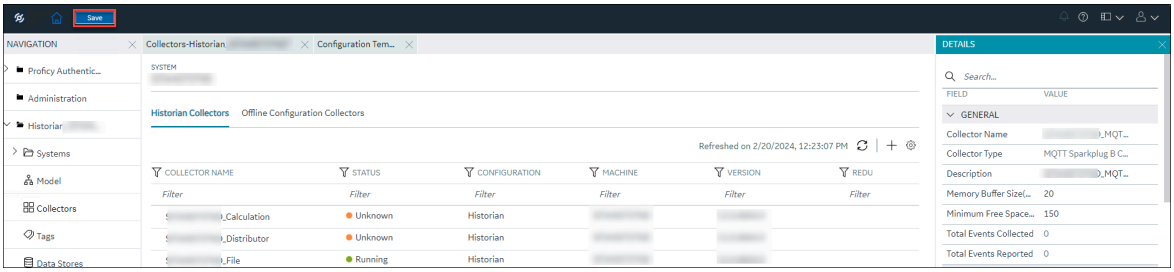
      If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:
      - iH Security Admins
      - iH Collector Admins
      - iH Tag Admins

    You can also configure the collector to start automatically when you start the computer.
13. Select **Add**.
    The collector instance is added. The fields specific to the collector appear in the **DETAILS** section.
14. In the **COLLECTOR SPECIFIC CONFIGURATION** sections, configure values in the following table.

| Field | Description |
|---|---|
| **Max Recovery Time (hours)** | Enter the maximum time, in hours, for which the collector will attempt to recover data after the collector is started or when connection be- |

| Field | Description |
|---|---|
| | tween the collector and the OSI PI server is re-established. This time is calculated as the duration between the current time and the last known write time.<br><br>Continuous data collection is resumed only after the previous data has been recovered.<br><br>The default value is 4 hours. |
| **Data Source** | Specify whether you want to collect data from PI archive or PI snapshot:<br>◦ **Archive**: Stores timeseries-based data.<br>◦ **Snapshot**: Stores the most recent values of tags. |
| **MTLS Security** | Indicates whether you want to use Mutual TLS (MTLS) protocol to enforce a secure and strong authentication mechanism. |
| **MTLS Data Encryption** | Indicates whether you want to encrypt the data that the collector shares to the data archiver (DA). |
| For more information on how to enable MTLS Security, refer to Enable MTLS Security *(on page 632)*. | |

15. As needed, enter values in the other sections common to all collectors *(on page 579)*.
16. In the upper-left corner of the page, select **Save**.




The changes to the collector instance are saved.

17. If needed, restart the collector.

Specify the tags whose data you want to collect using the collector. In the **CHOOSE CONFIGURATION** field,

- If you have selected **Historian Configuration**, specify the tags using Configuration Hub *(on page 357)*.
- If you have selected **Offline Configuration**, modify the offline configuration file of the collector. By default, the file is available in the following location: `<installation folder of Historian>\GE Digital\<collector name>`. For information, refer to Offline Configuration for Collectors *(on page 2039)*. This option is applicable only if you have selected a cloud destination.

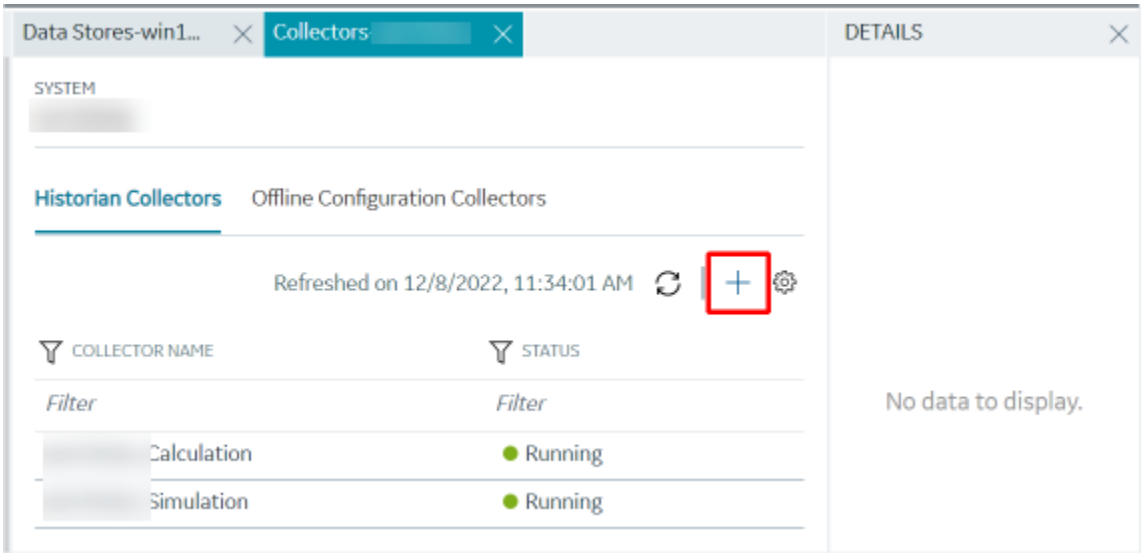
## Add and Configure an OSI PI Distributor

Install PI AF SDK version 2.7.5 or later.

An OSI PI distributor collects data from a Historian server and sends it to an OSI PI server. For more information, refer to Overview of the OSI PI Distributor *(on page 2457)*.

This topic describes how to add a collector instance using Configuration Hub. You can also add a collector instance using the RemoteCollectorConfigurator utility *(on page 797)*, which does not require you to install Web-based Clients.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**. A list of collectors in the default system appears.
3. In the upper-right corner of the main section, select +.



The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.

4. In the **MACHINE NAME** field, select the machine in which you want to add a collector instance.
5. In the **COLLECTOR TYPE** field, select **OSI PI Distributor Collector**, and then select **Get Details**.
   The **INSTALLATION DRIVE** and **DATA DIRECTORY** fields are disabled and populated.
6. Select **Next**.
   The **Source Configuration** section appears.
7. Enter values as described in the following table.

| Field | Description |
|---|---|
| **HISTORIAN SOURCE SERVER** | Enter the host name or IP address of the Historian server from which you want to collect data. A value is required. |
| **USERNAME** | Enter the username to connect to the Historian server. A value is required. |
| **PASSWORD** | Enter the password to connect to the Historian server. A value is required. |

8. Select **Next**.
   The **Destination Configuration** section appears. Under **CHOOSE DESTINATION**, the **PI Server** option is selected by default; the other options are disabled.
9. Enter values as described in the following table.

| Field | Description |
|---|---|
| **PI SERVER** | Enter the host name or IP address of the OSI PI server to which you want to send data. A value is required. |
| **PI USERNAME** | Enter the username to connect to the OSI PI server. |
| **PI PASSWORD** | Enter the password to connect to the OSI PI server. |

10. Select **Next**.
    The **Collector Initiation** section appears.

11. If needed, modify the value in the **COLLECTOR NAME** field.
    The value that you enter:
    - Must be unique.
    - Must not exceed 15 characters.
    - Must not contain a space.
    - Must not contain special characters except a hyphen, period, and an underscore.
12. In the **RUNNING MODE** field, select one of the following options.
    - **Service - Local System Account**: Select this option if you want to run the collector as a Windows service using the credentials of the local user (that is, the currently logged-in user). By default, this option is selected, and the **USERNAME** and **PASSWORD** fields are disabled.
    - **Service Under Specific User Account**: Select this option if you want to run the collector as a Windows service using a specific user account. If you select this option, you must enter values in the **USERNAME** and **PASSWORD** fields.

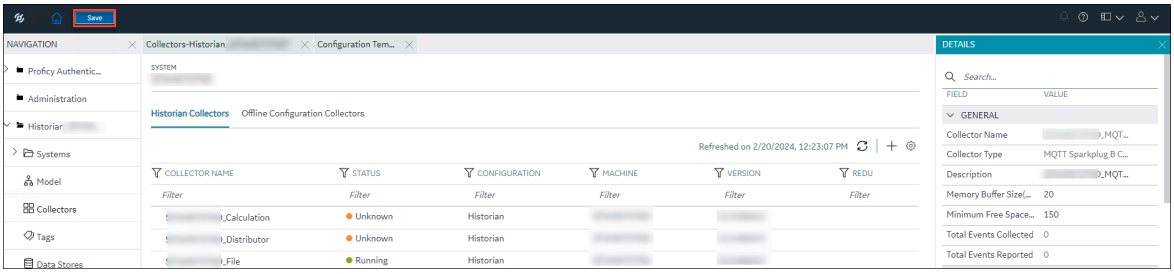
      If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:
      - iH Security Admins
      - iH Collector Admins
      - iH Tag Admins

    You can also configure the collector to start automatically when you start the computer.
13. Select **Add**.
    The collector instance is added. The fields specific to the collector instance appear in the **DETAILS** section.
14. In the **COLLECTOR SPECIFIC CONFIGURATION** section, configure values in the following table.

| Field | Description |
|---|---|
| **Max Recovery Time (hours)** | Enter the maximum time, in hours, for which the collector will attempt to recover data after the collector is started or when connection between the collector and the OSI PI server is re-established. This time is calculated as the duration between the current time and the last known write time.<br><br>Continuous data collection is resumed only after the previous data has been recovered. |

| Field | Description |
|---|---|
| | The default value is 4 hours. |
| **Data Source** | Specify whether you want to collect data from PI archive or PI snapshot:<br>◦ **Archive**: Stores timeseries-based data.<br>◦ **Snapshot**: Stores the most recent values of tags. |
| **MTLS Security** | Indicates whether you want to use Mutual TLS (MTLS) protocol to enforce a secure and strong authentication mechanism. |
| **MTLS Data Encryption** | Indicates whether you want to encrypt the data that the collector shares to the data archiver (DA). |
| For more information on how to enable MTLS Security, refer to Enable MTLS Security *(on page 632)*. | |

15. As needed, enter values in the other sections common to all collectors *(on page 579)*.
16. In the upper-left corner of the page, select **Save**.




The changes to the collector instance are saved.

17. If needed, restart the collector.

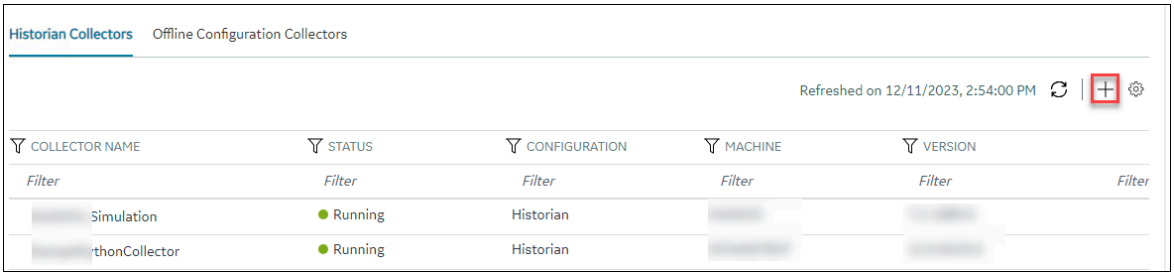
Specify the tags *(on page 357)* whose data you want to collect using the collector.

## Add a Python Collector Instance using Configuration Hub

- Install Python Collector *(on page 2464)*.

This topic describes how to add a Python collector instance using Configuration hub.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.
   A list of collectors in the default system appears.
3. In the upper-right corner of the main section, select +.




   The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.
4. In the **MACHINE NAME** field, select the machine in which you want to add a collector instance.
5. In the **COLLECTOR TYPE** field, select **Python Collector**, and then select **Get Details**.

> **Note:**
> The **INSTALLATION DRIVE** and **BASE DATA DIRECTORY** fields cannot be changed. This is the drive location and the data directory folder that you provided during Collectors installation.

   The **INSTALLATION DRIVE** and **DATA DIRECTORY** fields are populated with the drive location and the data directory folder.
6. Select **Next**.
   The **Destination Configuration** section appears. The collector machine name provided by you is selected as the **Source Configuration** by default.

   Under **CHOOSE DESTINATION**, the **Historian Server** option is selected by default. In addition, the **DESTINATION HISTORIAN SERVER** field is disabled and populated with the collector machine name.
7. Select the destination to which you want to send data, and then enter the values in the corresponding fields. You can send data to an on-premises Historian server or to a cloud destination.

a. If you need to send data to a cloud destination, select the cloud destinations as needed.
   - **Predix Timeseries**- Select this if you need to send data to Predix cloud. For more information, refer to Predix Cloud *(on page 621)*.
   - **Azure IoT Hub**- Select this if you need to send data to Azure Cloud in KairosDB format. For more information, refer to Azure IoT Hub (KairosDB format) *(on page 608)*.
   - **MQTT**- Select this if you need to send data to any of the following cloud destination.
     - Alibaba cloud. For more information, refer to Alibaba Cloud *(on page 587)*.
     - AWS cloud. For more information, refer to AWS Cloud *(on page 594)*.
     - Google cloud. For more information, refer to Google Cloud *(on page 614)*.

b. If you need to send data to an on-premises Historian server, select **Historian Server**.
   If you created security groups or enabled a strict client/collector authentication, enter the **USERNAME** and **PASSWORD** of the on-premises Historian server that you created during the installation of the collector.
   If you entered the **USERNAME**and **PASSWORD**, select **Test Connection**. This will help you to test if the Historian server that you are trying to connect is valid or if the credentials that you entered are valid.
   If the entered credentials are valid, a successful connection message appears.

8. After you selected the destination, select **Next**.
   The **Collector Initiation** section appears.
9. If needed, modify the value in the **COLLECTOR NAME** field.
   The value that you enter:
   - Must be unique.
   - Must not exceed 15 characters.
   - Must not contain a space.
   - Must not contain special characters except a hyphen, period, and an underscore.
10. In the **RUNNING MODE** field, select one of the following options.
    - **Service - Local System Account**: Select this option if you want to run the collector as a Windows service using the credentials of the local user (that is, the currently logged-in user). If you select this option, the **USERNAME** and **PASSWORD** fields are disabled.
    - **Service Under Specific User Account**: Select this option if you want to run the collector as a Windows service using a specific user account. If you select this option, you must enter values in the **USERNAME** and **PASSWORD** fields.

If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:

- iH Security Admins
- iH Collector Admins
- iH Tag Admins

You can also configure the collector to start automatically when you start the computer.

11. Select **Add**.
    The collector instance is added. The fields specific to the collector section appear in the **DETAILS** section.

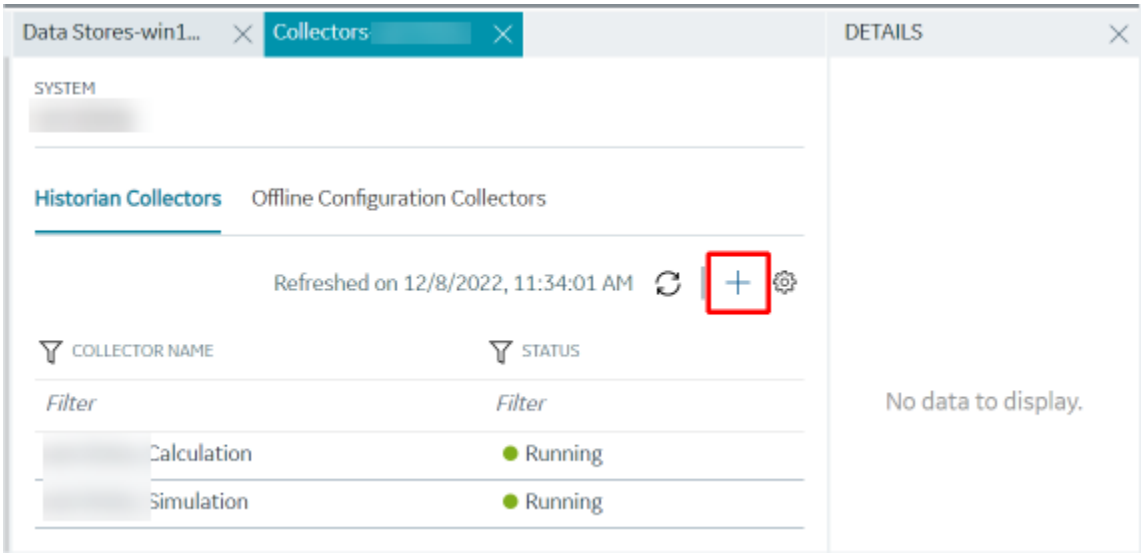
Specify the tags *(on page 357)* whose data you want to collect using the collector.

## Add and Configure a Server-to-Server Collector

The Server-to-Server collector collects data and messages from a source Historian server to a destination Historian server or a cloud destination. For more information, refer to Overview of the Server-to-Server Collector *(on page 2499)*.

This topic describes how to add a collector instance using Configuration Hub. You can also add a collector instance using the RemoteCollectorConfigurator utility *(on page 797)*, which does not require you to install Web-based Clients.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.
   A list of collectors in the default system appears.
3. In the upper-right corner of the main section, select +.




The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.

4. In the **MACHINE NAME** field, select the machine in which you want to add a collector instance.
5. In the **COLLECTOR TYPE** field, select **Server to Server Collector**, and then select **Get Details**.
   The **INSTALLATION DRIVE** and **DATA DIRECTORY** fields are disabled and populated.
6. Select **Next**.
   The **Source Configuration** section appears.
7. In the **HISTORIAN SOURCE SERVER** field, enter the host name or IP address of the Historian server from which you want to collect data. By default, the local host name appears.
8. If you created security groups or enabled a strict client/collector authentication, enter the **USERNAME** and **PASSWORD** of the on-premises Historian server that you created during the installation of the collector.
   If you entered the **USERNAME**and **PASSWORD**, select **Test Connection**. This will help you to test if the Historian server that you are trying to connect is valid or if the credentials that you entered are valid.
   If the entered credentials are valid, a successful connection message appears.
9. Select **Next**.
   The **Destination Configuration** section appears. The collector machine name provided by you is selected as the **Source Configuration** by default.

   Under **CHOOSE DESTINATION**, the **Historian Server** option is selected by default. In addition, the **DESTINATION HISTORIAN SERVER** field is populated with the collector machine name, you can change it in case of a remote Historian server.

10. Select the destination to which you want to send data, and then enter the values in the corresponding fields. You can send data to an on-premises Historian server or to a cloud destination.

    a. If you need to send data to a cloud destination, select the cloud destinations as needed.
        - **Predix Timeseries**- Select this if you need to send data to Predix cloud. For more information, refer to Predix Cloud *(on page 621)*.
        - **Azure IoT Hub**- Select this if you need to send data to Azure Cloud in KairosDB format. For more information, refer to Azure IoT Hub (KairosDB format) *(on page 608)*.
        - **MQTT**- Select this if you need to send data to any of the following cloud destination.
            - Alibaba cloud. For more information, refer to Alibaba Cloud *(on page 587)*.
            - AWS cloud. For more information, refer to AWS Cloud *(on page 594)*.
            - Google cloud. For more information, refer to Google Cloud *(on page 614)*.

    b. If you need to send data to an on-premises Historian server, select **Historian Server**.
    If you created security groups or enabled a strict client/collector authentication, enter the **USERNAME** and **PASSWORD**.
    If you entered the **USERNAME**and **PASSWORD**, select **Test Connection**.
    If the entered credentials are valid, a successful connection message appears.

11. Select **Next**.
    The **Collector Initiation** section appears.
12. If needed, modify the value in the **COLLECTOR NAME** field.
    The value that you enter:
    - Must be unique.
    - Must not exceed 15 characters.
    - Must not contain a space.
    - Must not contain special characters except a hyphen, period, and an underscore.
13. In the **RUNNING MODE** field, select one of the following options.
    - **Service - Local System Account**: Select this option if you want to run the collector as a Windows service using the credentials of the local user (that is, the currently logged-in user). If you select this option, the **USERNAME** and **PASSWORD** fields are disabled.
    - **Service Under Specific User Account**: Select this option if you want to run the collector as a Windows service using a specific user account. If you select this option, you must enter values in the **USERNAME** and **PASSWORD** fields.

If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:

- iH Security Admins
- iH Collector Admins
- iH Tag Admins

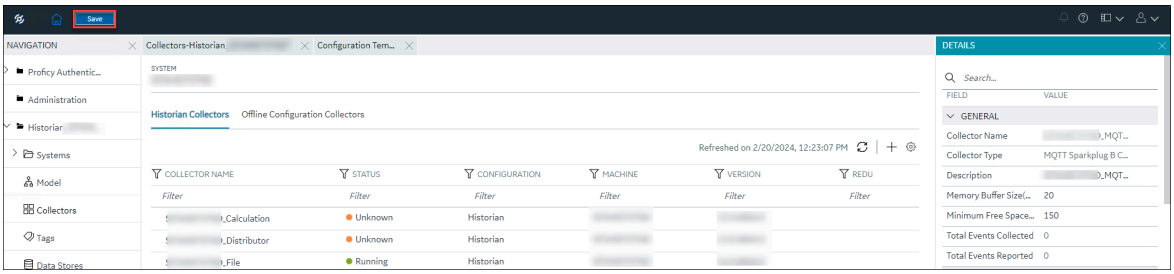
You can also configure the collector to start automatically when you start the computer.

14. Select **Add**.
    The collector instance is added. The fields specific to the collector section appear in the **DETAILS** section.
15. In the **COLLECTOR SPECIFIC CONFIGURATION** section, configure values as described in the following table.

| Field | Description |
|---|---|
| **Alarm Replication** | Indicates whether you want to enable or disable alarm replication. If you enable alarm replication, all collected alarm data will be transferred from the source server to the destination server. If you enable alarm replication, you also enable alarm recovery. However, if you set the **Max Recovery Time** value to zero, alarm recovery does not happen. |
| **Message Replication** | Indicates whether you to want to enable or disable message replication. If you enable message replication, messages will be transferred from the source server to the destination server. You can use this data for audits. If you enable message replication, you also enable message recovery. However, if you set the **Max Recovery Time** value to zero, message recovery does not happen. |
| **Calculation Timeout (sec)** | The maximum time allowed for a tag's calculation formula to execute before being terminated. The default value is 10 seconds. |
| **Max Recovery Time (hr)** | The maximum duration, in hours, for which the collector will attempt to restore data during recovery logic. The default value is 4 hours. |
| **Add Prefix to Messages** | The prefix to identify replicated messages on the destination.<br><br>Alarms and events data will automatically have a prefix added to it with the following syntax: `MachineName_Datasource` |

| Field | Description |
|---|---|
| | For example, if your alarm is forwarded from the server `Almserver12` with a data source named `OPCAE`, the prefix will be `Almserver12_OPCAE`. |
| **MTLS Security** | Indicates whether you want to use Mutual TLS (MTLS) protocol to enforce a secure and strong authentication mechanism. |
| **MTLS Data Encryption** | Indicates whether you want to encrypt the data that the collector shares to the data archiver (DA). |
| For more information on how to enable MTLS Security, refer to Enable MTLS Security *(on page 632)*. | |

16. As needed, enter values in the other sections common to all collectors *(on page 579)*.
17. In the upper-left corner of the page, select **Save**.




The changes to the collector instance are saved.

18. If needed, restart the collector.

Specify the tags whose data you want to collect using the collector. In the **CHOOSE CONFIGURATION** field,

- If you have selected **Historian Configuration**, specify the tags using Configuration Hub *(on page 357)*.
- If you have selected **Offline Configuration**, modify the offline configuration file of the collector. By default, the file is available in the following location: `<installation folder of Historian>\GE Digital\<collector name>`. For information, refer to Offline Configuration for Collectors *(on page 2039)*. This option is applicable only if you have selected a cloud destination.

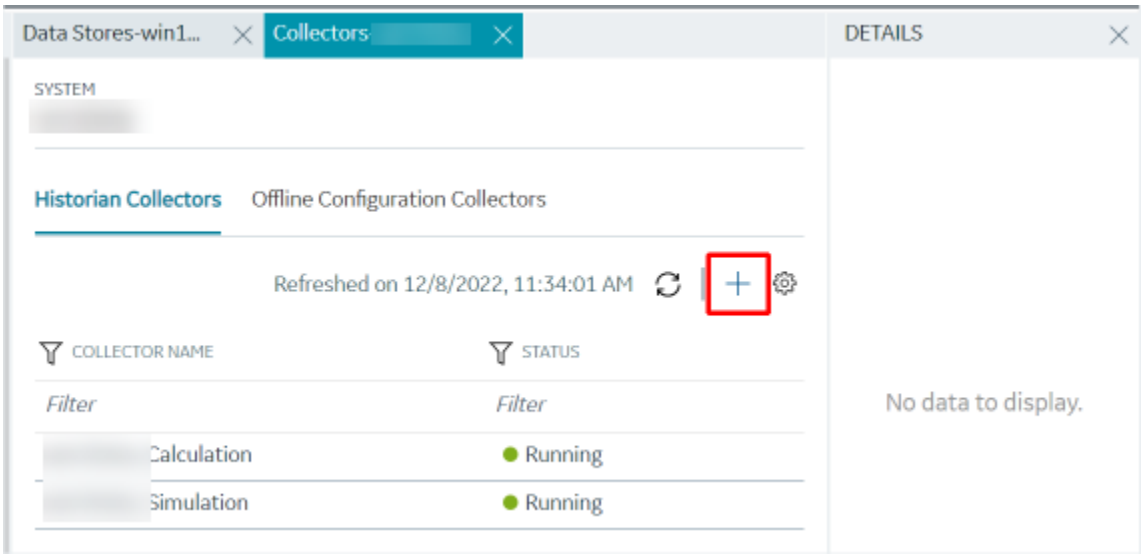
## Add and Configure a Server-to-Server Distributor

The Server-to-Server distributor is used to send data from a smaller Historian server to a larger, centralized on-premises Historian server. For more information, refer to Overview of the Server-to-Server

Distributor *(on page 2554)*. If you want to send data to a cloud destination, use the Server-to-Server collector *(on page 559)* instead.

This topic describes how to add a collector instance using Configuration Hub. You can also add a collector instance using the RemoteCollectorConfigurator utility *(on page 797)*, which does not require you to install Web-based Clients.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.
   A list of collectors in the default system appears.
3. In the upper-right corner of the main section, select +.




   The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.
4. In the **MACHINE NAME** field, select the machine in which you want to add a collector instance.
5. In the **COLLECTOR TYPE** field, select **Server to Server Distributor Collector**, and then select **Get Details**.
   The **INSTALLATION DRIVE** and **DATA DIRECTORY** fields are disabled and populated.
6. Select **Next**.
   The **Source Configuration** section appears.
7. In the **HISTORIAN SOURCE SERVER** field, enter the host name or IP address of the Historian server from which you want to collect data. By default, the local host name appears.

8. If you created security groups or enabled a strict client/collector authentication, enter the **USERNAME** and **PASSWORD** of the on-premises Historian server that you created during the installation of the collector.
   If you entered the **USERNAME**and **PASSWORD**, select **Test Connection**. This will help you to test if the Historian server that you are trying to connect is valid or if the credentials that you entered are valid.
   If the entered credentials are valid, a successful connection message appears.
9. Select **Next**.
   The **Destination Configuration** section appears. Under **CHOOSE DESTINATION**, the **Historian Server** option is selected by default; the other options are disabled. In addition, the **DESTINATION HISTORIAN SERVER** field is populated with the collector machine name, you can change it in case of a remote Historian server.
10. In the **USERNAME** and **PASSWORD** fields, enter the credentials to access the destination Historian server. Values are required only for a remote Historian server.
    If you entered the **USERNAME**and **PASSWORD**, select **Test Connection**. This will help you to test if the Historian server that you are trying to connect is valid or if the credentials that you entered is valid.
11. Select **Next**.
    The **Collector Initiation** section appears.
12. If needed, modify the value in the **COLLECTOR NAME** field.
    The value that you enter:
    - Must be unique.
    - Must not exceed 15 characters.
    - Must not contain a space.
    - Must not contain special characters except a hyphen, period, and an underscore.
13. In the **RUNNING MODE** field, select one of the following options.
    - **Service - Local System Account**: Select this option if you want to run the collector as a Windows service using the credentials of the local user (that is, the currently logged-in user). If you select this option, the **USERNAME** and **PASSWORD** fields are disabled.
    - **Service Under Specific User Account**: Select this option if you want to run the collector as a Windows service using a specific user account. If you select this option, you must enter values in the **USERNAME** and **PASSWORD** fields.

      If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:

- iH Security Admins
- iH Collector Admins
- iH Tag Admins

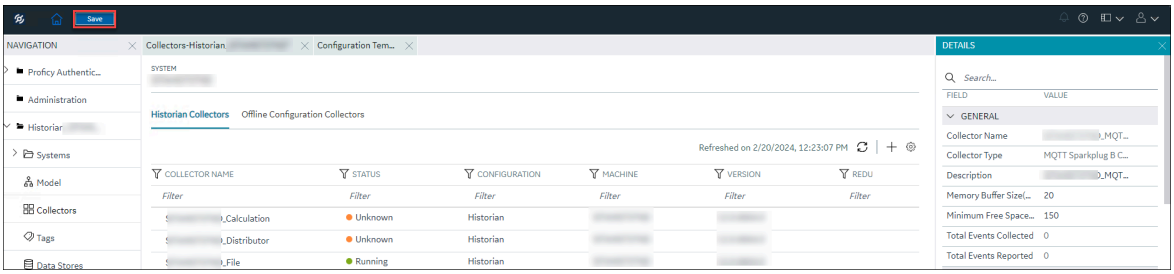
You can also configure the collector to start automatically when you start the computer.

14. Select **Add**.
    The collector instance is added. The fields specific to the collector section appear in the **DETAILS** section.
15. In the **COLLECTOR SPECIFIC CONFIGURATION** section, configure values as described in the following table.

| Field | Description |
|---|---|
| **Alarm Replication** | Indicates whether you want to enable or disable alarm replication. If you enable alarm replication, all collected alarm data will be transferred from the source server to the destination server. If you enable alarm replication, you also enable alarm recovery. However, if you set the **Max Recovery Time** value to zero, alarm recovery does not happen. |
| **Message Replication** | Indicates whether you to want to enable or disable message replication. If you enable message replication, you also enable message recovery. However, if you set the **Max Recovery Time** value to zero, message recovery does not happen. |
| **Calculation Timeout (sec)** | The maximum time allowed for a tag's calculation formula to execute before being terminated. The default value is 10 seconds. |
| **Max Recovery Time (hr)** | The maximum duration, in hours, for which the collector will attempt to restore data during recovery logic. The default value is 4 hours. |
| **Add Prefix to Messages** | The prefix to identify replicated messages on the destination.<br>Alarms and events data will automatically have a prefix added to it with the following syntax:<br>`MachineName_Datasource`<br>For example, if your alarm is forwarded from the server `Almserver12` with a data source named `OPCAE`, the prefix will be `Almserver12_OPCAE`. |
| **MTLS Security** | Indicates whether you want to use Mutual TLS (MTLS) protocol to enforce a secure and strong authentication mechanism. |

| Field | Description |
|---|---|
| **MTLS Data Encryption** | Indicates whether you want to encrypt the data that the collector shares to the data archiver (DA). |
| For more information on how to enable MTLS Security, refer to Enable MTLS Security *(on page 632)*. | |

16. As needed, enter values in the other sections common to all collectors *(on page 579)*.
17. In the upper-left corner of the page, select **Save**.




The changes to the collector instance are saved.

18. If needed, restart the collector.

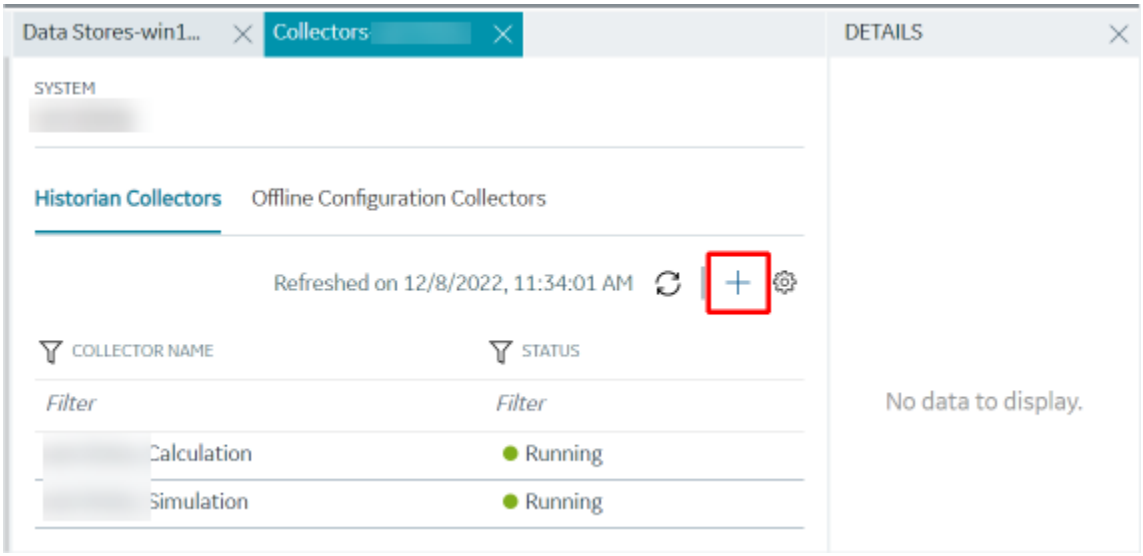
Specify the tags *(on page 357)* whose data you want to collect using the collector.

## Add and Configure a Simulation Collector

The Simulation collector generates random numbers and string patterns for demonstration purposes. For more information, refer to Overview of the Simulation Collector *(on page 2558)*.

This topic describes how to add a collector instance using Configuration Hub. You can also add a collector instance using the RemoteCollectorConfigurator utility *(on page 797)*, which does not require you to install Web-based Clients.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**. A list of collectors in the default system appears.
3. In the upper-right corner of the main section, select +.




The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.

4. In the **MACHINE NAME** field, select the machine in which you want to add a collector instance.
5. In the **COLLECTOR TYPE** field, select **Simulation Collector**, and then select **Get Details**.
   The **INSTALLATION DRIVE** and **DATA DIRECTORY** fields are disabled and populated.
6. Select **Next**.
   The **Source Configuration** section appears. The **COLLECTOR MACHINE NAME** field is disabled and populated.
7. Select **Next**.
   The **Destination Configuration** section appears. The collector machine name provided by you is selected as the **Source Configuration** by default.

   Under **CHOOSE DESTINATION**, the **Historian Server** option is selected by default. In addition, the **DESTINATION HISTORIAN SERVER** field is disabled and populated with the collector machine name.
8. Select the destination to which you want to send data, and then enter the values in the corresponding fields. You can send data to an on-premises Historian server or to a cloud destination.

a. If you need to send data to a cloud destination, select the cloud destinations as needed.
   - **Predix Timeseries**- Select this if you need to send data to Predix cloud. For more information, refer to Predix Cloud *(on page 621)*.
   - **Azure IoT Hub**- Select this if you need to send data to Azure Cloud in KairosDB format. For more information, refer to Azure IoT Hub (KairosDB format) *(on page 608)*.
   - **MQTT**- Select this if you need to send data to any of the following cloud destination.
     - Alibaba cloud. For more information, refer to Alibaba Cloud *(on page 587)*.
     - AWS cloud. For more information, refer to AWS Cloud *(on page 594)*.
     - Google cloud. For more information, refer to Google Cloud *(on page 614)*.

b. If you need to send data to an on-premises Historian server, select **Historian Server**.
   If you created security groups or enabled a strict client/collector authentication, enter the **USERNAME** and **PASSWORD** of the on-premises Historian server that you created during the installation of the collector.
   If you entered the **USERNAME**and **PASSWORD**, select **Test Connection**. This will help you to test if the Historian server that you are trying to connect is valid or if the credentials that you entered are valid.
   If the entered credentials are valid, a successful connection message appears.

9. Select **Next**.
   The **Collector Initiation** section appears.
10. If needed, modify the value in the **COLLECTOR NAME** field.
    The value that you enter:
    - Must be unique.
    - Must not exceed 15 characters.
    - Must not contain a space.
    - Must not contain special characters except a hyphen, period, and an underscore.
11. In the **RUNNING MODE** field, select one of the following options.
    - **Service - Local System Account**: Select this option if you want to run the collector as a Windows service using the credentials of the local user (that is, the currently logged-in user). If you select this option, the **USERNAME** and **PASSWORD** fields are disabled.
    - **Service Under Specific User Account**: Select this option if you want to run the collector as a Windows service using a specific user account. If you select this option, you must enter values in the **USERNAME** and **PASSWORD** fields.

If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:

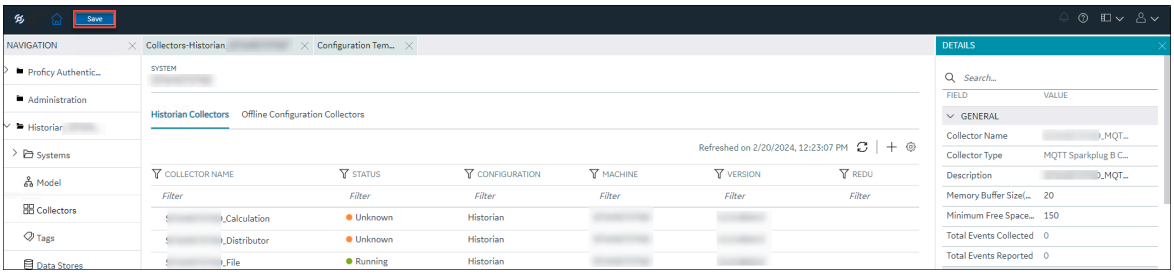
- iH Security Admins
- iH Collector Admins
- iH Tag Admins

You can also configure the collector to start automatically when you start the computer.

12. Select **Add**.
    The collector instance is added. The fields specific to the collector section appear in the **DETAILS** section.
13. In the **COLLECTOR SPECIFIC CONFIGURATION** section, configure values as described in the following table.

| Field | Description |
|---|---|
| **Number of Tags** | The number of Historian tags that you want the create for the collector. |
| **Function Period (seconds)** | The period, in seconds, of the `SIN`,`STEP`, and `RAMP` functions implemented in the collector. |
| **MTLS Security** | Indicates whether you want to use Mutual TLS (MTLS) protocol to enforce a secure and strong authentication mechanism. |
| **MTLS Data Encryption** | Indicates whether you want to encrypt the data that the collector shares to the data archiver (DA). |
| For more information on how to enable MTLS Security, refer to Enable MTLS Security *(on page 632)*. | |

14. As needed, enter values in the other sections common to all collectors *(on page 579)*.
15. In the upper-left corner of the page, select **Save**.



The changes to the collector instance are saved.

16. If needed, restart the collector.

Specify the tags whose data you want to collect using the collector. In the **CHOOSE CONFIGURATION** field,

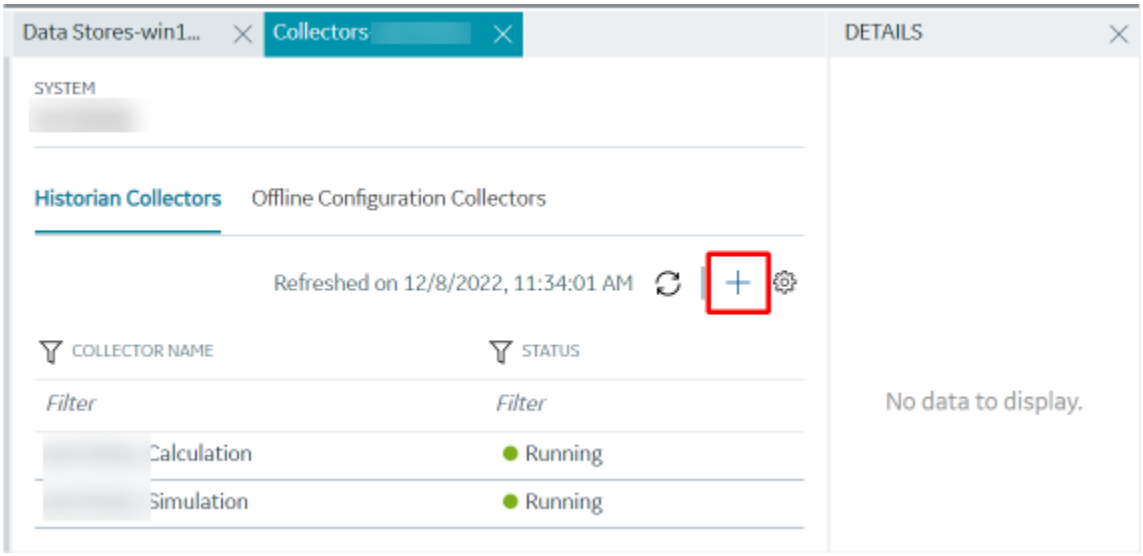
- If you have selected **Historian Configuration**, specify the tags using Configuration Hub *(on page 357)*.
- If you have selected **Offline Configuration**, modify the offline configuration file of the collector. By default, the file is available in the following location: `<installation folder of Historian>\GE Digital\<collector name>`. For information, refer to Offline Configuration for Collectors *(on page 2039)*. This option is applicable only if you have selected a cloud destination.

## Add and Configure Windows Performance Collector

The Windows Performance collector collects Windows performance counter data. For more information, refer to Overview of the Windows Performance Collector *(on page 2562)*.

This topic describes how to add a collector instance using Configuration Hub. You can also add a collector instance using the RemoteCollectorConfigurator utility *(on page 797)*, which does not require you to install Web-based Clients.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**. A list of collectors in the default system appears.
3. In the upper-right corner of the main section, select +.




The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.

4. In the **MACHINE NAME** field, select the machine in which you want to add a collector instance.
5. In the **COLLECTOR TYPE** field, select **Windows Performance Collector**, and then select **Get Details**.
   The **INSTALLATION DRIVE** and **DATA DIRECTORY** fields are disabled and populated.
6. Select **Next**.
   The **Source Configuration** section appears. The **WINDOWS MACHINE NAME** field is disabled and populated.
7. Select **Next**.
   The **Destination Configuration** section appears. The collector machine name provided by you is selected as the **Source Configuration** by default.

   Under **CHOOSE DESTINATION**, the **Historian Server** option is selected by default. In addition, the **DESTINATION HISTORIAN SERVER** field is disabled and populated with the collector machine name.
8. Select the destination to which you want to send data, and then enter the values in the corresponding fields. You can send data to an on-premises Historian server or to a cloud destination.

a. If you need to send data to a cloud destination, select the cloud destinations as needed.
   - **Predix Timeseries**- Select this if you need to send data to Predix cloud. For more information, refer to Predix Cloud *(on page 621)*.
   - **Azure IoT Hub**- Select this if you need to send data to Azure Cloud in KairosDB format. For more information, refer to Azure IoT Hub (KairosDB format) *(on page 608)*.
   - **MQTT**- Select this if you need to send data to any of the following cloud destination.
     - Alibaba cloud. For more information, refer to Alibaba Cloud *(on page 587)*.
     - AWS cloud. For more information, refer to AWS Cloud *(on page 594)*.
     - Google cloud. For more information, refer to Google Cloud *(on page 614)*.

b. If you need to send data to an on-premises Historian server, select **Historian Server**.
   If you created security groups or enabled a strict client/collector authentication, enter the **USERNAME** and **PASSWORD** of the on-premises Historian server that you created during the installation of the collector.
   If you entered the **USERNAME**and **PASSWORD**, select **Test Connection**. This will help you to test if the Historian server that you are trying to connect is valid or if the credentials that you entered are valid.
   If the entered credentials are valid, a successful connection message appears.

9. Select **Next**.
   The **Collector Initiation** section appears. The **COLLECTOR NAME** field is populated with a value in the following format: `<Historian server name>_WindowsPerfMon`
10. If needed, modify the value in the **COLLECTOR NAME** field.
    The value that you enter:
    - Must be unique.
    - Must not exceed 15 characters.
    - Must not contain a space.
    - Must not contain special characters except a hyphen, period, and an underscore.
11. In the **RUNNING MODE** field, select one of the following options.
    - **Service - Local System Account**: Select this option if you want to run the collector as a Windows service using the credentials of the local user (that is, the currently logged-in user). By default, this option is selected, and the **USERNAME** and **PASSWORD** fields are disabled.
    - **Service Under Specific User Account**: Select this option if you want to run the collector as a Windows service using a specific user account. If you select this option, you must enter values in the **USERNAME** and **PASSWORD** fields.

If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:

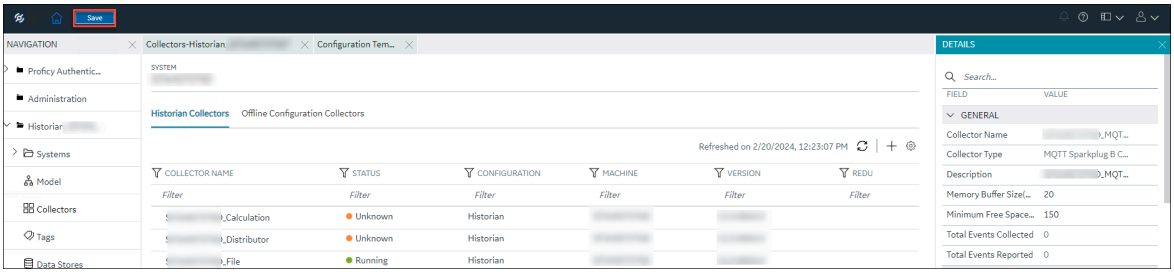
- iH Security Admins
- iH Collector Admins
- iH Tag Admins

You can also configure the collector to start automatically when you start the computer.

12. Select **Add**.
    The collector instance is added.
13. Select the collector instance.
    The fields specific to the collector section appear in the **DETAILS** section.
14. In the **COLLECTOR SPECIFIC CONFIGURATION** section, configure values as described in the following table.

| Field | Description |
|---|---|
| **MTLS Security** | Indicates whether you want to use Mutual TLS (MTLS) protocol to enforce a secure and strong authentication mechanism. |
| **MTLS Data Encryption** | Indicates whether you want to encrypt the data that the collector shares to the data archiver (DA). |
| For more information on how to enable MTLS Security, refer to Enable MTLS Security *(on page 632)*. | |

15. As needed, enter values in the the other sections common to all collectors *(on page 579)*.
16. In the upper-left corner of the page, select **Save**.




    The changes to the collector instance are saved.
17. If needed, restart the collector.

Specify the tags whose data you want to collect using the collector. In the **CHOOSE CONFIGURATION** field,

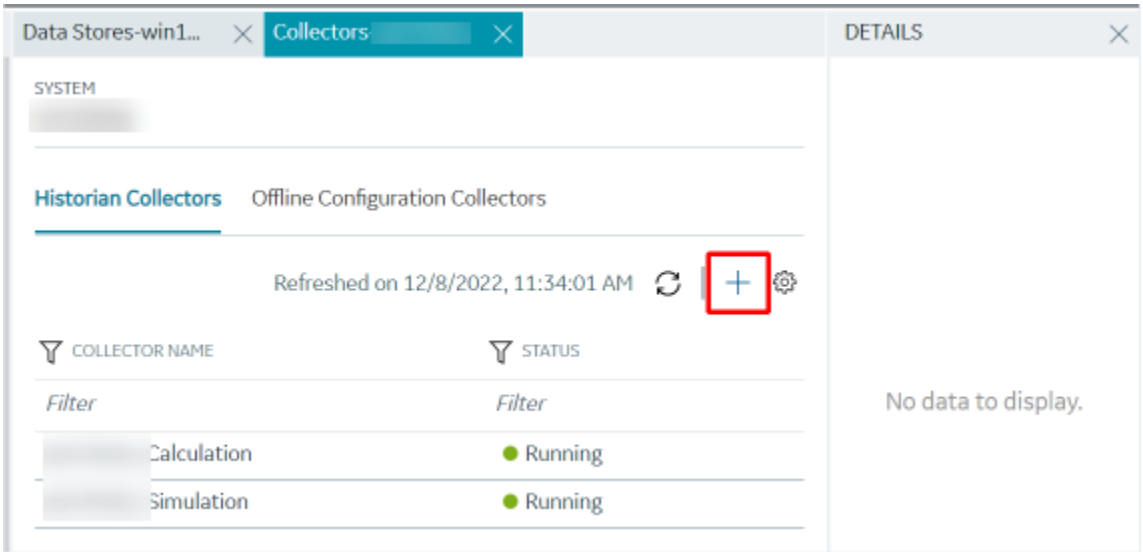
- If you have selected **Historian Configuration**, specify the tags using Configuration Hub *(on page 357)*.
- If you have selected **Offline Configuration**, modify the offline configuration file of the collector. By default, the file is available in the following location: `<installation folder of Historian>\GE Digital\<collector name>`. For information, refer to Offline Configuration for Collectors *(on page 2039)*. This option is applicable only if you have selected a cloud destination.

## Add and Configure a Wonderware Collector

The Wonderware Collector gathers data samples from a Wonderware Historian 2014 R2 server and stores it in the Proficy Historian server. For more information, refer to Overview of the Wonderware Collector *(on page 2566)*.

This topic describes how to add a collector instance using Configuration Hub. You can also add a collector instance using the RemoteCollectorConfigurator utility *(on page 797)*, which does not require you to install Web-based Clients.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**. A list of collectors in the default system appears.
3. In the upper-right corner of the main section, select +.



The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.

4. In the **MACHINE NAME** field, select the machine in which you want to add a collector instance.
5. In the **COLLECTOR TYPE** field, select **Wonderware Collector**, and then select **Get Details**.
   The **INSTALLATION DRIVE** and **DATA DIRECTORY** fields are disabled and populated.
6. Select **Next**.
   The **Source Configuration** section appears. The field is disabled and populated.
7. In the **WONDERWARE SERVER** field, enter the host name or IP address of the Wonderware Historian server from which you want to collect data.
8. Enter values in the **USERNAME** and **PASSWORD** fields to connect to the Wonderware Historian server.
9. Select **Next**.
   The **Destination Configuration** section appears. The collector machine name provided by you is selected as the **Source Configuration** by default.

   Under **CHOOSE DESTINATION**, the **Historian Server** option is selected by default. In addition, the **DESTINATION HISTORIAN SERVER** field is disabled and populated with the collector machine name.

10. Select the destination to which you want to send data, and then enter the values in the corresponding fields. You can send data to an on-premises Historian server or to a cloud destination.

    a. If you need to send data to a cloud destination, select the cloud destinations as needed.
       - **Predix Timeseries**- Select this if you need to send data to Predix cloud. For more information, refer to Predix Cloud *(on page 621)*.
       - **Azure IoT Hub**- Select this if you need to send data to Azure Cloud in KairosDB format. For more information, refer to Azure IoT Hub (KairosDB format) *(on page 608)*.
       - **MQTT**- Select this if you need to send data to any of the following cloud destination.
         - Alibaba cloud. For more information, refer to Alibaba Cloud *(on page 587)*.
         - AWS cloud. For more information, refer to AWS Cloud *(on page 594)*.
         - Google cloud. For more information, refer to Google Cloud *(on page 614)*.

    b. If you need to send data to an on-premises Historian server, select **Historian Server**.
       If you created security groups or enabled a strict client/collector authentication, enter the **USERNAME** and **PASSWORD** of the on-premises Historian server that you created during the installation of the collector.

If you entered the **USERNAME**and **PASSWORD**, select **Test Connection**. This will help you to test if the Historian server that you are trying to connect is valid or if the credentials that you entered are valid.
If the entered credentials are valid, a successful connection message appears.

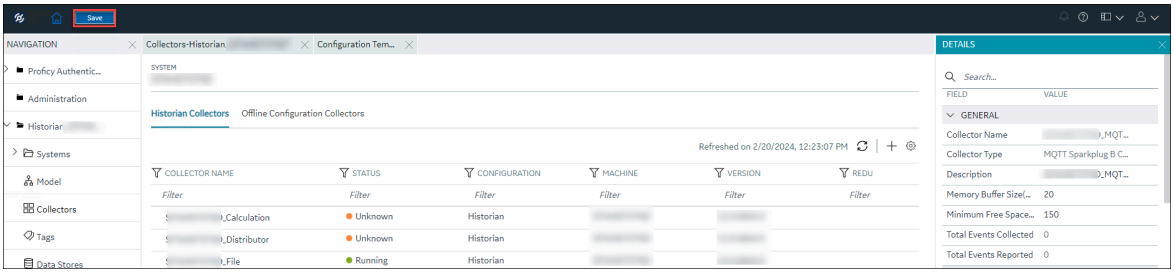
11. Select **Next**.
    The **Collector Initiation** section appears. The **COLLECTOR NAME** field is populated with a value in the following format: `<Historian server name>_Wonderware`
12. If needed, modify the value in the **COLLECTOR NAME** field.
    The value that you enter:
    - Must be unique.
    - Must contain the string `Wonderware`.
    - Must not exceed 15 characters.
    - Must not contain a space.
    - Must not contain special characters except a hyphen, period, and an underscore.
13. In the **RUNNING MODE** field, select one of the following options.
    - **Service - Local System Account**: Select this option if you want to run the collector as a Windows service using the credentials of the local user (that is, the currently logged-in user). By default, this option is selected, and the **USERNAME** and **PASSWORD** fields are disabled.
    - **Service Under Specific User Account**: Select this option if you want to run the collector as a Windows service using a specific user account. If you select this option, you must enter values in the **USERNAME** and **PASSWORD** fields.

      If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:
      - iH Security Admins
      - iH Collector Admins
      - iH Tag Admins

    You can also configure the collector to start automatically when you start the computer.
14. Select **Add**.
    The collector instance is added.
15. Select the collector instance.
    The fields specific to the collector section appear in the **DETAILS** section.
16. In the **COLLECTOR SPECIFIC CONFIGURATION** section, configure values as described in the following table.

| Field | Description |
|---|---|
| **Recovery Time (hours)** | Enter the maximum time, in hours, for which the collector will attempt to recover data after the collector is started or when connection between the collector and the Wonderware Historian server is re-established. This time is calculated as the duration between the current time and the last known write time.<br><br>Continuous data collection is resumed only after the previous data has been recovered.<br><br>The default value is 0 hours. |
| **Throttle (Milliseconds)** | Enter the frequency of Wonderware data polling. This is to minimize the load on the Wonderware Historian server. By default, Wonderware Collector tries to query the tag data every 100 milliseconds based on the collection interval time. You can change this value to any time between 100 milliseconds to 16 hours. |
| **MTLS Security** | Indicates whether you want to use Mutual TLS (MTLS) protocol to enforce a secure and strong authentication mechanism. |
| **MTLS Data Encryption** | Indicates whether you want to encrypt the data that the collector shares to the data archiver (DA). |
| For more information on how to enable MTLS Security, refer to Enable MTLS Security *(on page 632)*. | |

17. As needed, enter values in the the other sections common to all collectors *(on page 579)*.
18. In the upper-left corner of the page, select **Save**.




The changes to the collector instance are saved.

19. If needed, restart the collector.

Specify the tags whose data you want to collect using the collector. In the **CHOOSE CONFIGURATION** field,

- If you have selected **Historian Configuration**, specify the tags using Configuration Hub *(on page 357)*.
- If you have selected **Offline Configuration**, modify the offline configuration file of the collector. By default, the file is available in the following location: `<installation folder of Historian>\GE Digital\<collector name>`. For information, refer to Offline Configuration for Collectors *(on page 2039)*. This option is applicable only if you have selected a cloud destination.

## Collector Configuration - Common Fields

This topic provides a list of general fields that you can configure for any collector instance. These fields appear in the **DETAILS** section when you select a collector instance. For fields specific to a cloud destination, refer to Alibaba Cloud *(on page 587)*, AWS Cloud *(on page 594)*, Azure Cloud *(on page 601)*, Google Could *(on page 614)*, and Predix Cloud *(on page 621)*. For fields specific to the collector type, refer to the topics on adding and configuring each individual collector.

After you enter/modify a value in these fields, the changes are saved automatically after you place the cursor in a different field. Until the changes are saved, the values appear in bold formatting.

**Table 29. The General Section**

| Field | Description |
|---|---|
| **Collector Name** | The name of the collector instance. This field is disabled. |
| **Collector Type** | The type of the collector instance. This field is disabled. |
| **Description** | The description of the collector instance. This field is disabled. |
| **Memory Buffer Size (MB)** | The size of the memory buffer currently assigned to the store-and-forward function. The memory buffer stores data during short-term or momentary interruptions of the server connection; the disk buffer handles long duration outages. To estimate the size you need for this buffer, you need to know how fast the collector is trying to send data to the |

**Table 29. The General Section (continued)**

| Field | Description |
|---|---|
| | server and how long the server connection is likely to be down. With those values and a safety margin, you can compute the required size of the buffer.<br><br>The default value is 20. |
| **Minimum Free Space (MB)** | The minimum free disk space that must be available on the computer. If the minimum space required is not available when the collector starts, the collector will shut down. |
| **Total Events Collected** | This field is disabled. |
| **Total Events Reported** | This field is disabled. |

**Table 30. The Tags section**

| Field | Description |
|---|---|
| **Tag Prefix** | The prefix that will be added to each tag that you configure for the collector instance. This field is disabled and populated with the name of the collector instance.<br><br>This field applies to all collectors except File and Calculation collectors. |
| **Collection Interval Value** | The interval at which the collector collects data for all the tags configured in the collector instance.<br><br>• For polled data collection, this value represents the time required to complete a poll of tags in the collector.<br>• For unsolicited data collection, it represents the frequency at which data is retrieved from tags in the collector. The collection interval can be individually configured for each tag.<br><br>You can set this value for each tag as well. |

**Table 30. The Tags section (continued)**

| Field | Description |
|---|---|
| | **Important:**<br>For an OPC collector, to avoid collecting redundant values when using device time-stamps, specify a collection interval that is greater than the OPC server update rate. |
| **Collection Interval** | The units of measure for the collection interval value. |
| **Collection Type** | The type of the data collection:<br>• **Polled:** Data is collected based on a scheduled time interval. This type of data collection is supported only for:<br>◦ The Calculation collector<br>◦ The HAB collector<br>◦ The iFIX collector<br>◦ The OPC Classic DA collector<br>◦ The OPC UA DA collector<br>◦ The Python collector<br>◦ The Simulation collector<br>◦ The Windows Performance collector<br>• **Unsolicited:** Data is collected based on an event. This type of data collection is supported only for:<br>◦ The Calculation collector<br>◦ The HAB collector<br>◦ The MQTT collector<br>◦ The MQTT Sparkplug B collector<br>◦ The ODBC collector<br>◦ The OPC Classic DA collector<br>◦ The OPC Classic HDA collector<br>◦ The OPC UA DA collector<br>◦ The OSI PI collector<br>◦ The OSI PI distributor |

**Table 30. The Tags section (continued)**

| Field | Description |
|---|---|
| | ◦ The Python collector<br>◦ The Server-to-Server collector<br>◦ The Server-to-Server distributor<br>◦ The Wonderware Collector |
| **Time Assigned By** | Indicates whether the timestamp for the collected data is set based on the data source or the collector. For example, for an OSI PI collector, if you select **Source**, the timestamp of the OSI PI server is considered for the values collected by the collector. If you select **Collector**, the timestamp of the collector is considered. |

**Table 31. The Collector Compression Section**

| Field | Description |
|---|---|
| **Collector Compression** | Indicates whether you want to apply collector compression, which is a smoothing filter to data retrieved from the data source. By ignoring small changes in values that fall within a deadband centered around the last reported value, only significant changes are stored in Historian, thus consuming less archive storage space.<br><br>For more information, refer to About Collector and Archive Compression *(on page 660)*. |
| **Deadband** | Indicates whether you want to apply a deadband based on the percentage of values or on absolute values.<br><br>For example, if you set the deadband to 20% for a range of 0 to 500 engineering units, the deadband value is 100 units, which is 50 units on each side. Therefore, only if the difference between two values is greater than 50, they are stored in Historian. |

**Table 31. The Collector Compression Section (continued)**

| Field | Description |
|---|---|
| | **Note:**<br>If the data quality changes from good to bad or vice versa, the values are stored in Historian regardless of the deadband value. |
| **Deadband Value** | The deadband value that you want to use for values collected by the collector. Depending on whether you have selected percent or absolute, the deadband value is determined.<br><br>For example, if you want to set a deadband of 5 units on either side of a value (that is, value +/- 5), enter 10 in the **Deadband Value** field, and select **Absolute** in the **Deadband** field. Similarly, if you want to set a deadband of 5% on either side of a value, enter 10 in the **Deadband Value** field, and select **Percent** in the **Deadband** field.<br><br>For more information, refer to About Collector and Archive Compression *(on page 660)*. |
| **Compression Timeout** | The time for one poll cycle for which collector compression is not used, thus sending all the samples to Historian.<br><br>This is used for a Calculation collector or Server-to-Server collector, when calculations fail, you may possibly observe collector compression (even if it is not enabled), thus producing no results or bad quality data. In such cases, you can use compression timeout, thus sending all the samples to Historian.<br><br>For more information, refer to About Collector and Archive Compression *(on page 660)*. |

**Table 31. The Collector Compression Section (continued)**

| Field | Description |
|---|---|
| **Compression Timeout Interval** | The units of measure for compression timeout. |
| **Spike Logic Control** | Indicates whether you want to apply spike logic to tag values. When you apply spike logic, in the event of a sudden change in tag values, a data sample is inserted just before the spike. The timestamp of the inserted sample is determined by your polling interval. If samples are collected at 1 second intervals, the inserted sample's timestamp will be 1 second before the spike. This helps clearly identify the spike, and retains a more accurate picture of the data leading up to it.<br><br>For more information, refer to Enable Spike Logic *(on page 970)*. |
| **Multiplier** | Specifies how much larger a spike value must be than the deadband range before the spike logic is invoked.<br><br>For example, if you enter 3 in the **Multiplier** field, and the deadband is set to 5%, the spike logic will not be invoked until the difference between the spike value and the previously archived data point is 15% of the EGU range. |
| **Interval** | Specifies how many samples must have been compressed before the spike logic is invoked. For example, if you enter 4 in the **Interval** field, and 6 values have been compressed since the last archived data sample, the spike logic will be invoked. |

**Table 32. The Collector Options Section**

| Field | Description |
|---|---|
| **Online Tag Configuration Changes** | Indicates whether you want tag configuration changes to reflect immediately. If you disable this option, any tag configuration changes will reflect only after you restart the collector instance. |
| **Browse Source Address Space** | Indicates whether you want to allow browsing for tags in the source. You may sometimes want to disable this option to reduce processing load on the collector. |
| **Synchronize Timestamps to Server** | Indicates whether you want to adjust the time-stamp of data to align with the time setting in the Historian server. Note that this does not change the time setting in the collector machine; it only calculates the timestamp based on the difference between the time settings in the server machine and the collector machine, independent of time zone or daylight saving differences.<br><br>**Note:**<br>• This option is applicable only if the timestamp of the collector is considered (instead of that of the data source - as specified in the **Time Assigned By** field).<br>• If this option is disabled, and if the time in the collector machine is more than 15 minutes ahead of the time in the server machine, data will not be stored in Historian. |
| **Source/Device Timestamp in** | The source/device timestamp format. Either UTC or Local. |

**Table 32. The Collector Options Section (continued)**

| Field | Description |
|---|---|
| **Delay Collection at Startup (sec)** | The duration, in seconds, after which you want the data collection to begin post tag configuration. |

**Table 33. The Advanced Section**

| Field | Description |
|---|---|
| **Debug Mode** | The debug mode for collector logs. 0 indicates normal log level, whereas 255 indicates that debugging is enabled.<br><br>**Note:**<br>Leaving the debug mode enabled for a long time consumes disk space. |
| **Message Compression** | Indicates whether you want to apply message compression. |

**Table 34. The Collector Status Output Section**

| Field | Description |
|---|---|
| **Rate Output Address** | The address in the source database into which the collector writes the output of events/minute. This will help an operator (or a HMI/SCADA application) learn the performance of the collector. Values are captured once a minute.<br><br>You must enter the address of a writable analog field.<br><br>For example, for an iFIX collector, enter the address of an iFIX tag in the following format: `<node name>.<tag name>.<field name>` (for example, `MyNode.MySIM_AO.F_CV`). |
| **Status Output Address** | The address in the source database into which the collector writes the current value of its status (for example, running, stopped). This will help an opera- |

**Table 34. The Collector Status Output Section (continued)**

| Field | Description |
|---|---|
| | tor (or a HMI/SCADA application) learn the current status of the collector. The value is updated only if the status of the collector changes.<br><br>You must enter the address of a writable text field of at least eight characters.<br><br>For an iFIX collector, use TX tag for the output address. Enter the address in the following format: `<node name>.<tag name>.<field name>` (for example, `MyNode.MyCollector_TX.A_CV`). |
| **Heartbeat Output Address** | The address in the source database into which the collector writes the heartbeat signal output. Values are captured once a minute.<br><br>You must enter the address of a writable analog field.<br><br>For an iFIX data collector, use an iFIX tag for the output address. Enter the address in the following format: `<node name>.<tag name>.<field name>` (for example, `MyNode.MyCollector_TX.A_CV`).<br><br>You can program the iFIX database to generate an alarm if values are not written every minute, notifying you that the collector has stopped. |

# Sending Data to Cloud

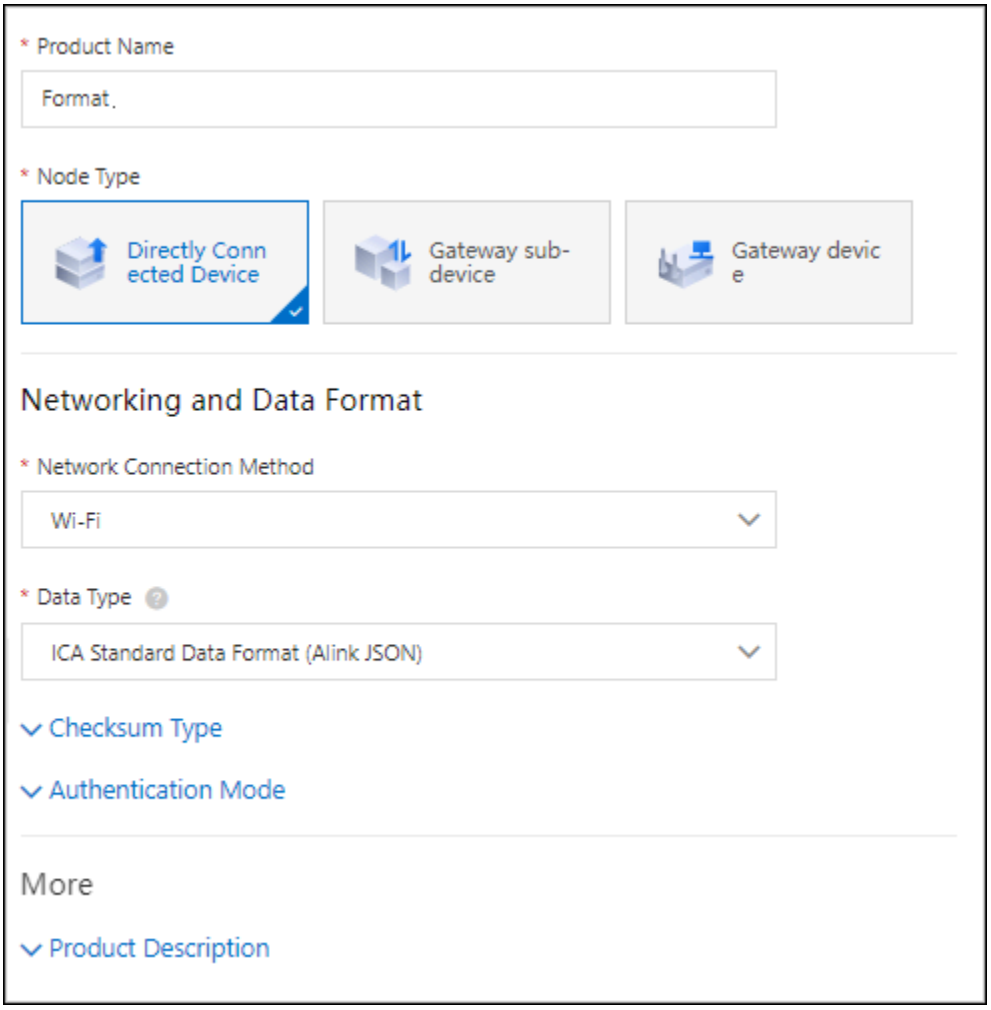
## Send Data to Alibaba Cloud

Generate a password using the utility. While generating the password, use the same algorithm that you will use to connect to Alibaba Cloud.

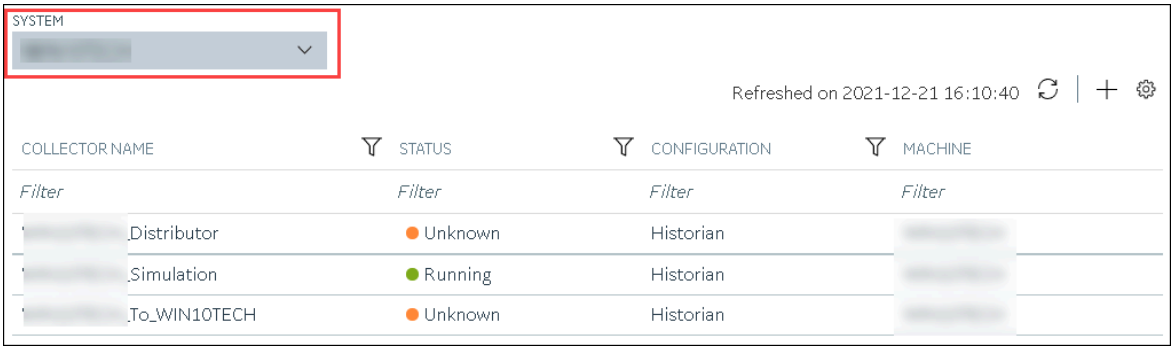
To send data to Alibaba Cloud, you can choose any of the following collectors:

- The iFIX collector
- The MQTT collector
- The ODBC collector
- The OPC Classic DA collector
- The OPC Classic HDA collector
- The OPC UA DA collector
- The OSI PI collector
- The Python Collector
- The Server-to-Server collector
- The Simulation collector
- The Windows Performance collector
- The Wonderware collector

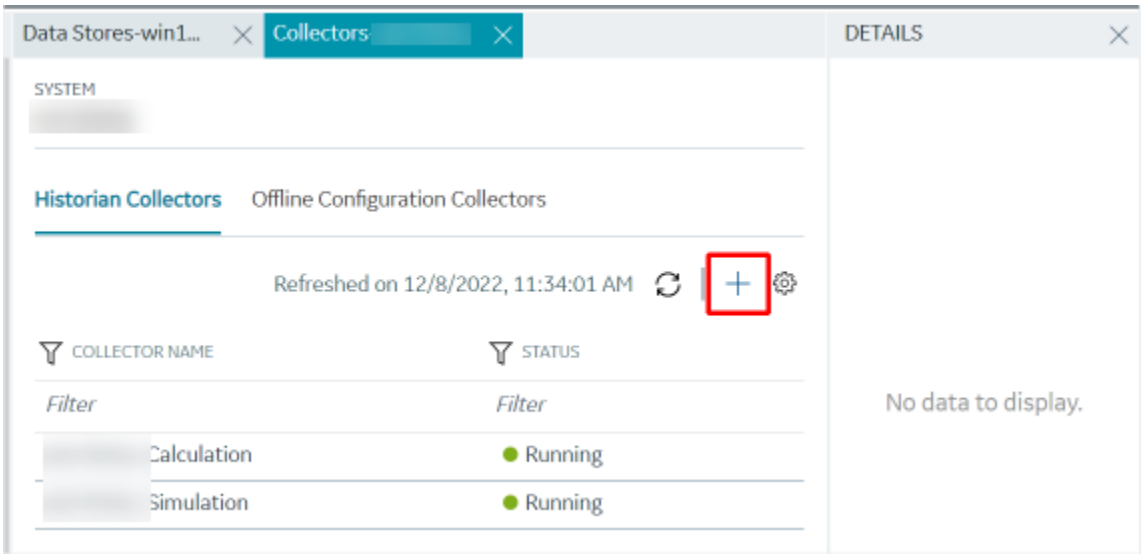
1. Access Alibaba IoT Platform console.
2. Create a product. When you do so:
    - In the **Node Type** field, select **Directly Connected Device**.
    - In the **Network Connection Method** field, select **Wi-Fi**.
    - In the **Data Type** field, select **ICA Standard Data Format**.




3. Note down the region ID for the region you have selected. For a list of region IDs, refer to https://www.alibabacloud.com/help/doc-detail/40654.htm.
4. Access the product certificate, and note down the product secret and product key values.
5. Create a device.
6. Access Configuration Hub *(on page 336)*.
7. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**. A list of collectors in the system appears.
8. If needed, select the system in which you want to add a collector instance.

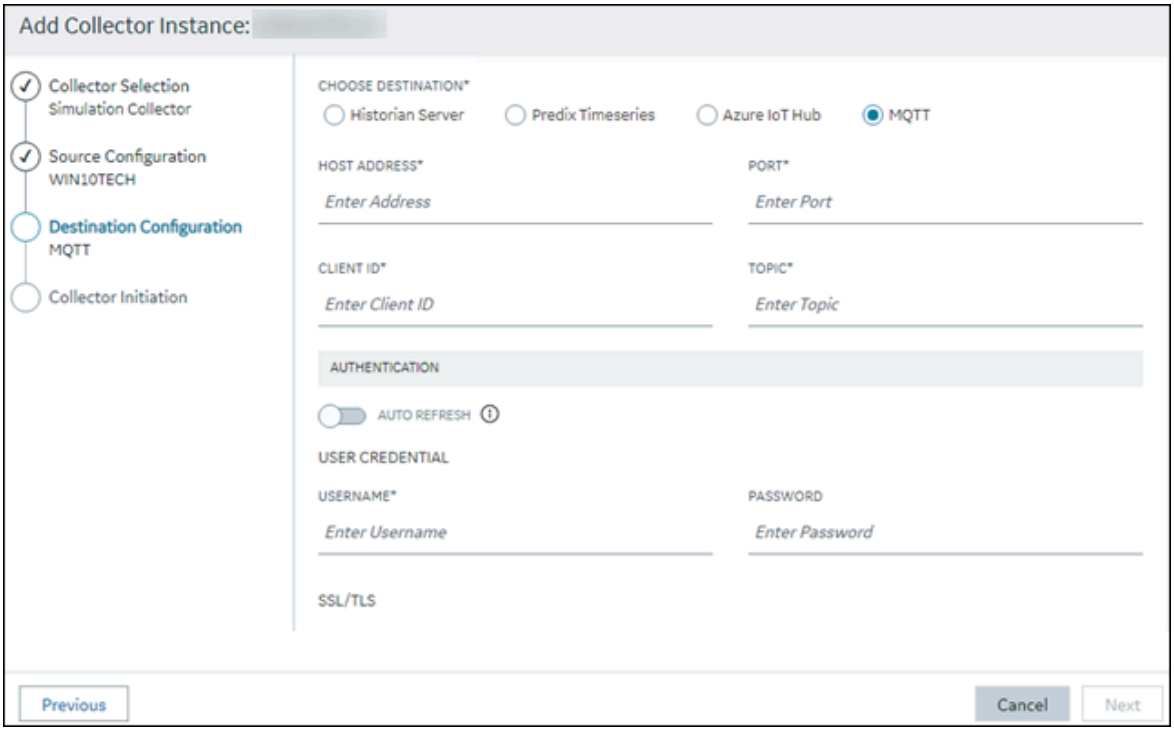



9. In the upper-right corner of the main section, select +.




   The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.

10. In the **COLLECTOR TYPE** field, select a collector type, and then select **Get Details**.
    The **INSTALLATION DRIVE** and **DATA DIRECTORY** fields are disabled and populated.
11. Select **Next**.
    The **Source Configuration** section appears, populating the hostname of the collector machine.
12. As needed, enter values in the available fields, and then select **Next**.
    The **Destination Configuration** section appears.

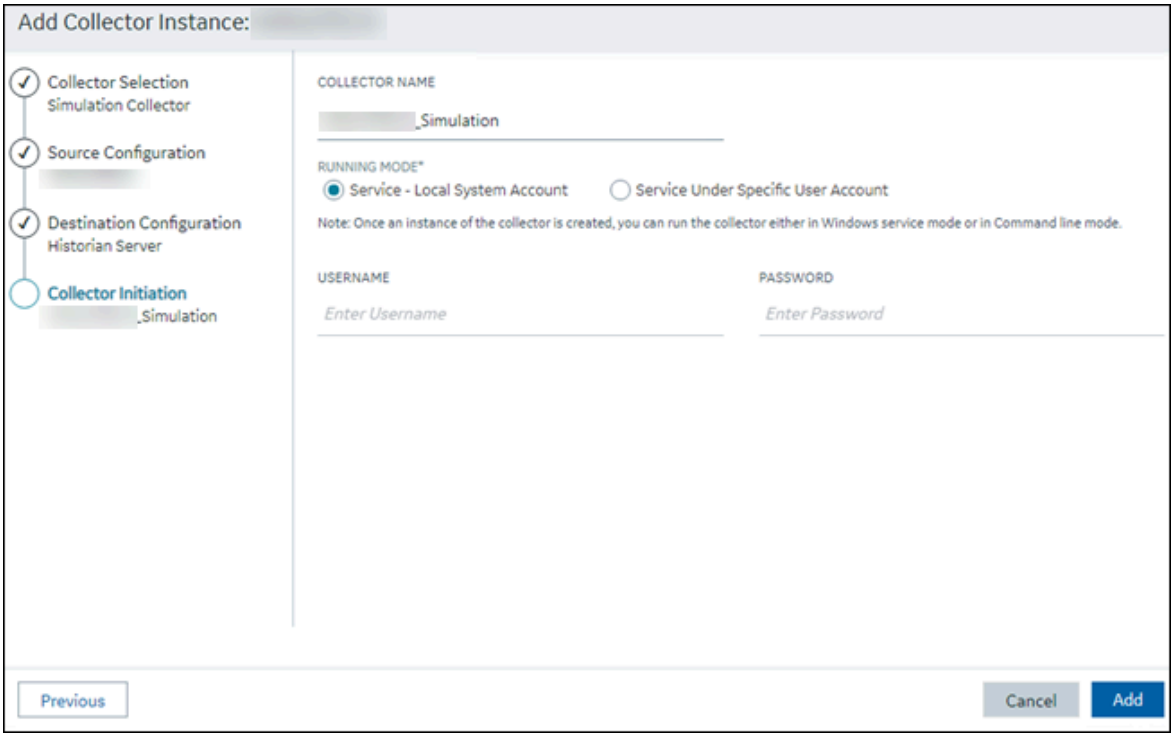



13. In the **CHOOSE DESTINATION** field, select **MQTT**, and then provide values as described in the following table.

| Field | Description |
|---|---|
| HOST ADDRESS | Enter a value in the following format: `<product name>.iot-as-mqtt.<region ID>.aliyuncs.com`. A value is required.<br><br>For example: `a23dr53dwrt.iot-as-mqtt.cn-shanghai.aliyuncs-.com` |
| PORT | Enter `1883`. A value is required. |
| CLIENT ID | Enter a value in the following format: `<device name>\|securemode=<value>,signmethod=<algorithm name>`. A value is required.<br>◦ For securemode, enter `2` for direct TLS connection, or enter `3` for direct TCP connection.<br>◦ For signmethod, specify the signature algorithm that you want to use. Valid values are hmacmd5, hmacsha1, hmacsha256, and sha256. You must use the same algorithm to generate the password.<br><br>For example: `MyDevice\|securemode=3,signmethod=hmacsha1` |

| Field | Description |
|---|---|
| TOPIC | Enter a value in the following format: `/sys/<product name>/<device name>/thing/event/property/post`. A value is required.<br><br>For example: `/sys/a23dr53dwrt/MyDevice/thing/event/property/post` |
| USERNAME | Enter a value in the following format: `<device name><product name>`. A value is required.<br><br>For example: `MyDevicea23dr53dwrt` |
| PASSWORD | Enter the password that you have generated. A value is required. |
| CHOOSE CONFIGURATION | Select the type of the configuration to specify the tags whose data you want to collect. Select one of the following options:<br>◦ **Historian Configuration**: Select this option if you want to add the tags manually using Historian Administrator *(on page 659)*. If you select this option, the **CONFIGURATION HISTORIAN SERVER** field appears.<br>◦ **Offline Configuration**: Select this option if you want to provide the tag names using the offline configuration *(on page 2039)* file instead of adding tags manually. By default, this file is located in the following location: `<installation folder of Historian>\GE Digital\<collector name>` |

14. Select **Next**.
    The **Collector Initiation** section appears.




15. Enter a collector name.

    The value that you enter:

    - Must be unique.
    - Must not exceed 15 characters.
    - Must not contain a space.
    - Must not contain special characters except a hyphen, period, and an underscore.

16. In the **RUNNING MODE** field, select one of the following options.

    - **Service - Local System Account**: Select this option if you want to run the collector as a Windows service using the credentials of the local user (that is, the currently logged-in user). If you select this option, the **USERNAME** and **PASSWORD** fields are disabled.
    - **Service Under Specific User Account**: Select this option if you want to run the collector as a Windows service using a specific user account. If you select this option, you must enter values in the **USERNAME** and **PASSWORD** fields. If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:
        - iH Security Admins
        - iH Collector Admins
        - iH Tag Admins

    If you choose the first option, you can also configure the collector to start automatically when you start the computer, or, in the case of iFIX collectors, whenever you start iFIX.

17. Select **Add**.
    The collector instance is created.
18. Specify the tags for which you want to collect data.
    - If you have selected **Historian Configuration** in the **CHOOSE CONFIGURATION** field, specify the tags manually *(on page 357)*.
    - If you have selected **Offline Configuration** in the **CHOOSE CONFIGURATION** field, specify the tags using the offline configuration file *(on page 2039)*.

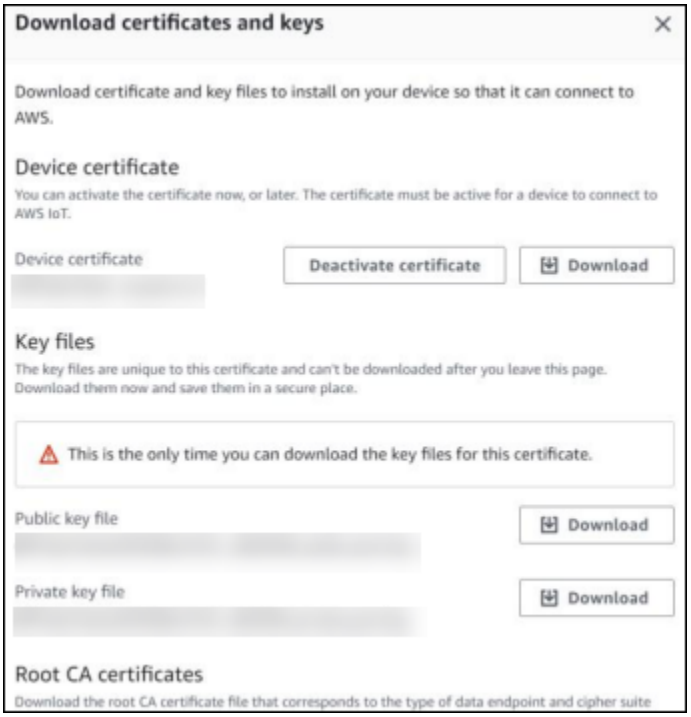
    The collector begins sending Historian data to the device that you have created.

## Send Data to AWS IoT Core

To send data to an AWS IoT Code, you can choose any of the following collectors:

- The iFIX collector
- The MQTT collector
- The ODBC collector
- The OPC Classic DA collector
- The OPC Classic HDA collector
- The OPC UA DA collector
- The OSI PI collector
- The Python Collector
- The Server-to-Server collector
- The Simulation collector
- The Windows Performance collector
- The Wonderware collector

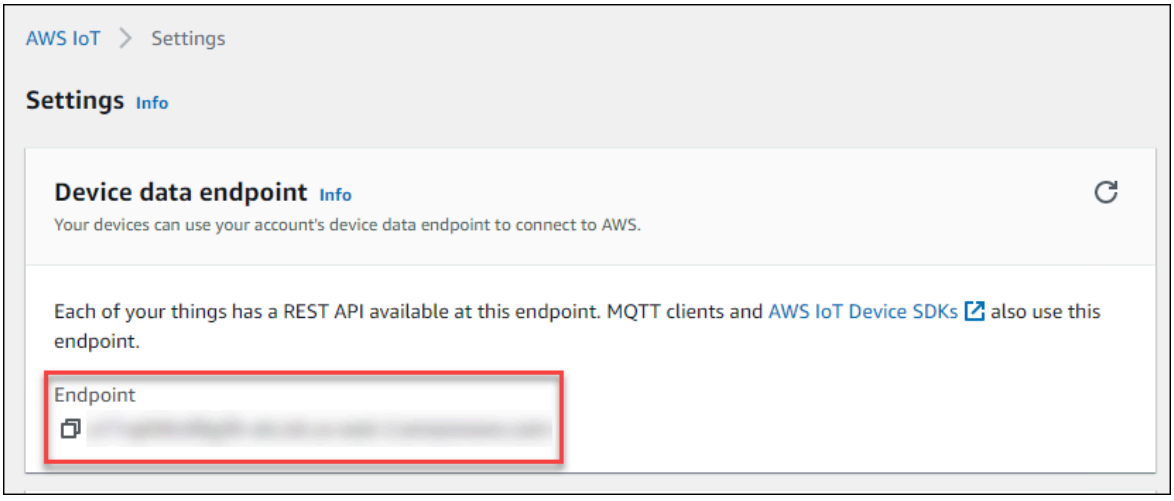
1. Access the **AWS Management Console** page.
2. Search and select **IoT Core**.
   The **AWS IoT** page appears.
3. Create a policy allowing the permissions that you want to grant on your device (for example, iot:Connect, iot:Publish, iot:Subscribe, iot:Receive). For the resource, provide the topic name. If, however, you want to use all topics, enter *.
4. Create a thing, linking it with the policy that you have created.
5. Download the certificates and key files for the device to communicate. In addition, download the root CA certificate.




**Important:**

This is mandatory, and it is the only time you can download the certificates.

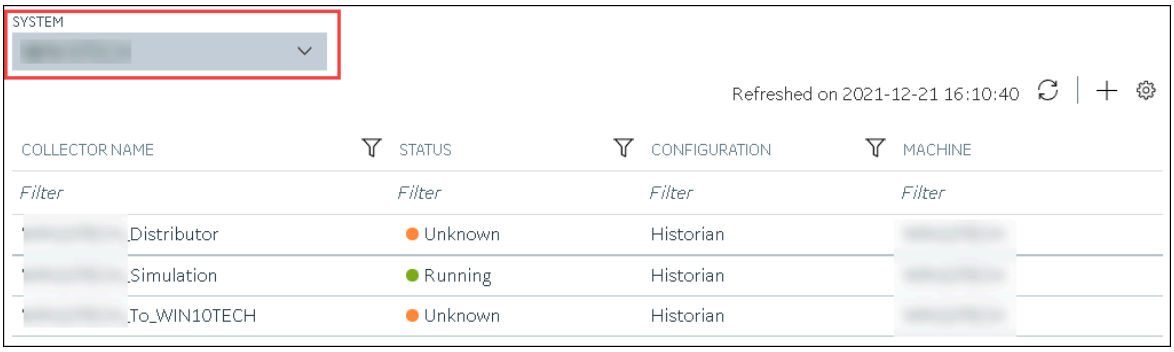
6. In the left navigation pane, select **Settings**.
7. Make a note of the endpoint that appears.



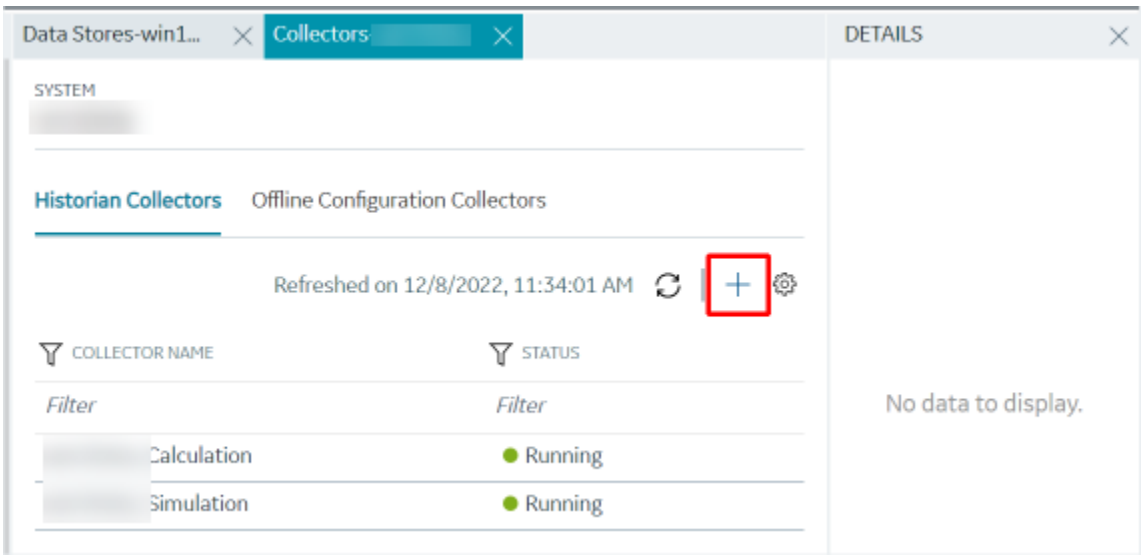

8. Access Configuration Hub *(on page 336)*.
9. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.

   A list of collectors in the system appears.

10. If needed, select the system in which you want to add a collector instance.

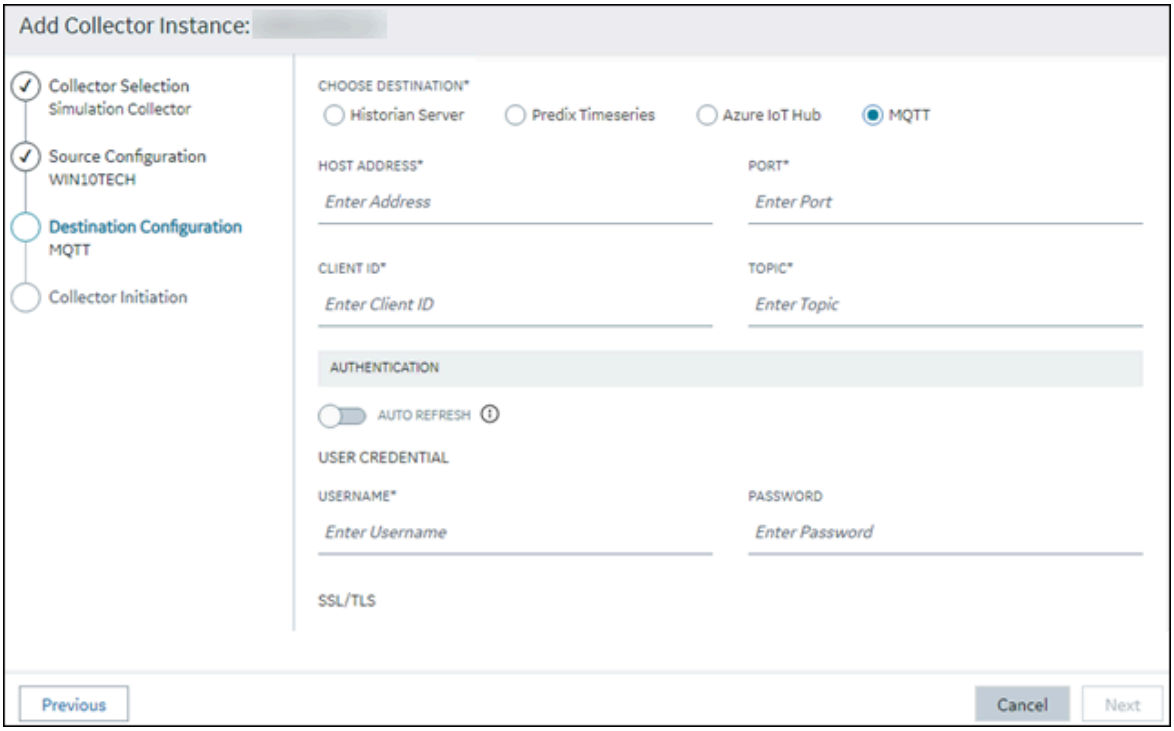



11. In the upper-right corner of the main section, select +.




The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.

12. In the **COLLECTOR TYPE** field, select a collector type (except the File collector and the Server-to-Server collector), and then select **Get Details**.
    The **INSTALLATION DRIVE** and **DATA DIRECTORY** fields are disabled and populated.
13. Select **Next**.
    The **Source Configuration** section appears, populating the hostname of the collector machine.
14. As needed, enter values in the available fields, and then select **Next**.
    The **Destination Configuration** section appears.



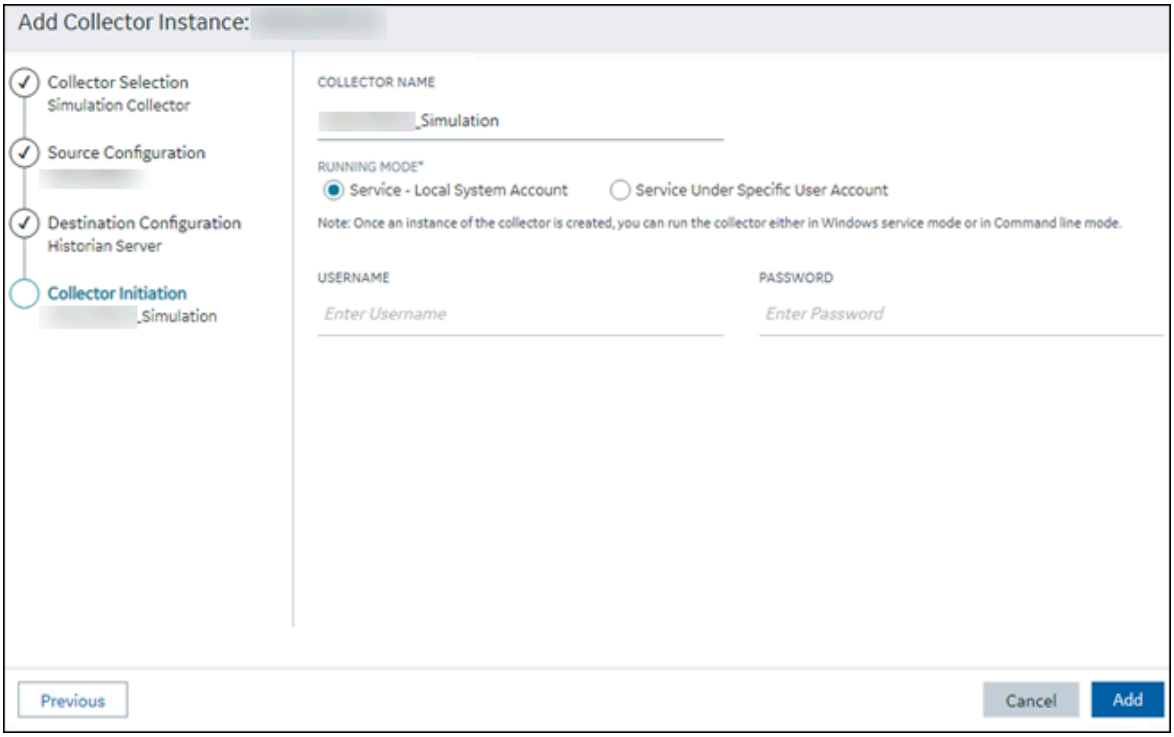

15. In the **CHOOSE DESTINATION** field, select **MQTT**, and then provide values as described in the following table.

| Field | Description |
|---|---|
| HOST ADDRESS | Enter the endpoint that you have noted down. A value is required. |
| PORT | Enter 8883. A value is required. |
| CLIENT ID | Enter the thing name. A value is required. |
| TOPIC | Enter the MQTT topic to which you want the collector to publish data. A value is required. For information on topic names, refer to https://docs.aws.amazon.com/iot/latest/developer-guide/topics.html. |
| USERNAME | Enter any value. Since we will use a certificate-based authentication, username and password will not be used; however, you must still enter a value. |

| Field | Description |
|---|---|
| PASSWORD | Enter any value. Since we will use a certificate-based authentication, username and password will not be used; however, you must still enter a value. |
| CA SERVER ROOT FILE | Enter the path of the root CA certificate file that you have downloaded. |
| CLIENT CERTIFICATE | Enter the path of the device certificate that you have downloaded. |
| PRIVATE KEY FILE | Enter the path of the private key file that you have downloaded. |
| PUBLIC KEY FILE | Enter the path of the public key file that you have downloaded. |
| CHOOSE CONFIGURATION | Select the type of the configuration to specify the tags whose data you want to collect. Select one of the following options:<br>◦ **Historian Configuration**: Select this option if you want to add the tags manually using Historian Administrator *(on page 659)*. If you select this option, the **CONFIGURATION HISTORIAN SERVER** field appears.<br>◦ **Offline Configuration**: Select this option if you want to provide the tag names using the offline configuration *(on page 2039)* file instead of adding tags manually. By default, this file is located in the following location: `<installation folder of Historian>\GE Digital\<collector name>` |

16. Select **Next**.

    The **Collector Initiation** section appears.




17. Enter a collector name.

    The value that you enter:

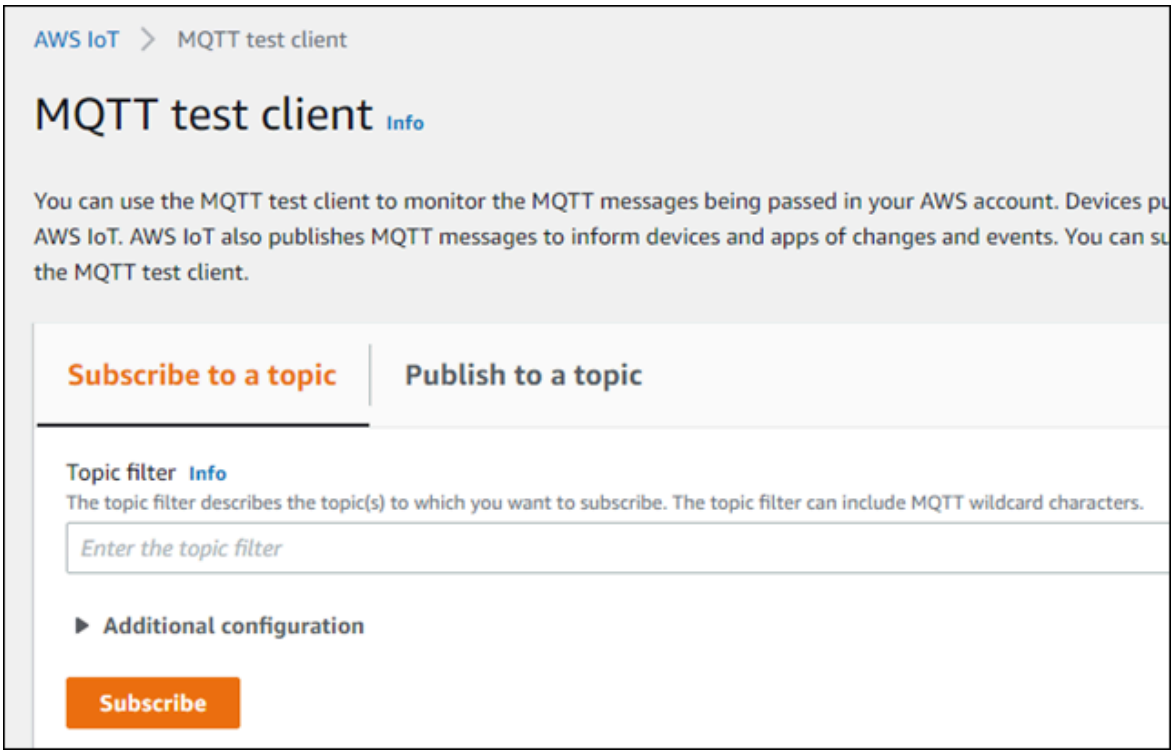
    - Must be unique.
    - Must not exceed 15 characters.
    - Must not contain a space.
    - Must not contain special characters except a hyphen, period, and an underscore.

18. In the **RUNNING MODE** field, select one of the following options.
    - **Service - Local System Account**: Select this option if you want to run the collector as a Windows service using the credentials of the local user (that is, the currently logged-in user). If you select this option, the **USERNAME** and **PASSWORD** fields are disabled.
    - **Service Under Specific User Account**: Select this option if you want to run the collector as a Windows service using a specific user account. If you select this option, you must enter values in the **USERNAME** and **PASSWORD** fields. If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:
        - iH Security Admins
        - iH Collector Admins
        - iH Tag Admins

    If you choose the first option, you can also configure the collector to start automatically when you start the computer, or, in the case of iFIX collectors, whenever you start iFIX.

19. Select **Add**.
    The collector instance is created.
20. Specify the tags for which you want to collect data.
    - If you have selected **Historian Configuration** in the **CHOOSE CONFIGURATION** field, specify the tags manually *(on page 357)*.
    - If you have selected **Offline Configuration** in the **CHOOSE CONFIGURATION** field, specify the tags using the offline configuration file *(on page 2039)*.

    The collector begins sending Historian data to the thing that you have created.
21. Access AWS IoT Core, and in the left pane, select **Test**.
    The **MQTT test client** page appears.




22. Subscribe to the topic to which the collector is publishing data, and then select **Subscribe**.
    The messages received from the topic appear, indicating that the collector is sending data to the AWS IoT device.
    AWS supports a payload of maximum 128 KB. Therefore, if the message size is greater than 128 KB, create a registry key named CloudMaxSamplesPerMsg for the collector instance, and decrease the value to 700 or less. If, however, you want to send more data in a message, we recommend that you create another collector instance and send data to another thing resource in AWS.

> **Tip:**
> To find out the message size, modify the collector instance *(on page 634)* and set the log level to 3 or more.

23. Create a VPC destination or an HTTP destination for the messages.
24. Monitor the data that you have collected.

## Send Data to Azure Cloud in the Key-Value Format

To send data to an Azure IoT Hub device, you can choose any of the following collectors:

- The iFIX collector
- The MQTT collector
- The ODBC collector
- The OPC Classic DA collector
- The OPC Classic HDA collector
- The OPC UA DA collector
- The OSI PI collector
- The Python Collector
- The Server-to-Server collector
- The Simulation collector
- The Windows Performance collector
- The Wonderware collector

This topic describes how to send data in the key-value format. In this format, the message size is bigger because names of the tag properties are repeated. However, it provides clarity to novice users. For example: `{"body":`

```
[{"tagname":"Azure_Iot_simulation_tag_1","epochtime":1629730936000,"tagvalue":7129.124023438,"quality":3},
{"tagname":"Azure_Iot_simulation_tag_2","epochtime":1629730936000,"tagvalue":123.3738924567,"quality":3}] ,"messa
```

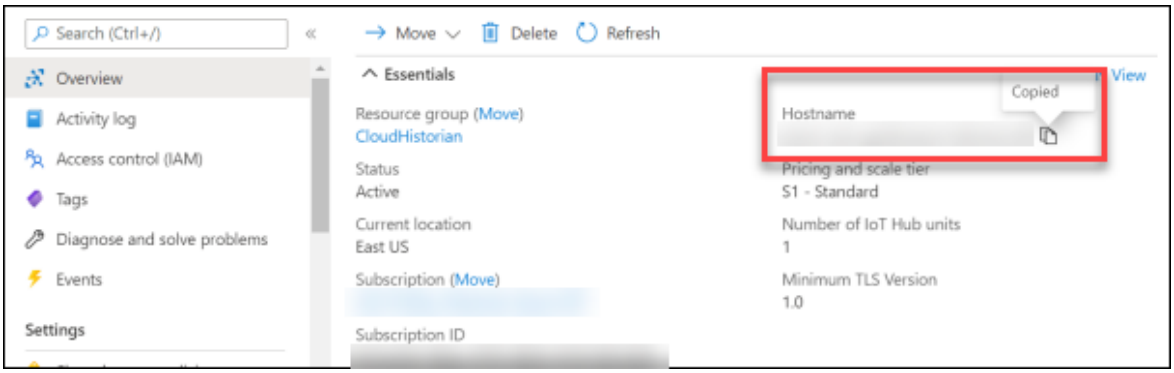
You can also send data in the KairosDB format *(on page 608)*.

> **Note:**
> Data in Azure IoT Hub is stored for maximum seven days, after which it is deleted from the hub. Therefore, you must consume the data within seven days. Based on your requirement, you can store it in a relevant Azure storage. You can then use Azure functions or streaming analytics to analyse the data.

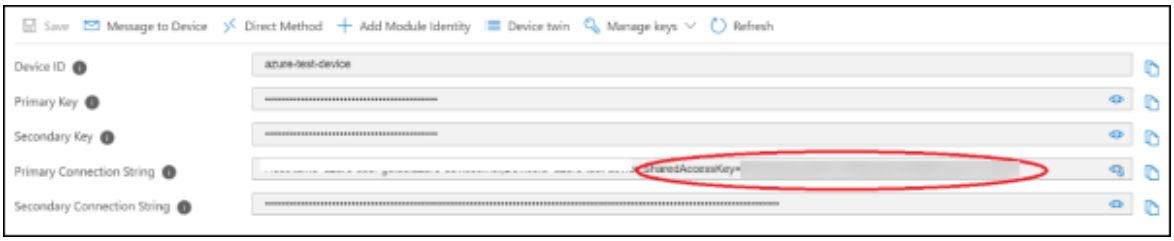
1. Create Azure IoT Hub.

> **Tip:**
> To choose the correct Azure IoT Hub tier based on your data throughput, refer to https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-devguide-quotas-throttling. For guidance on choosing the appropriate subscription, refer to https://azure.microsoft.com/en-us/pricing/details/iot-hub/

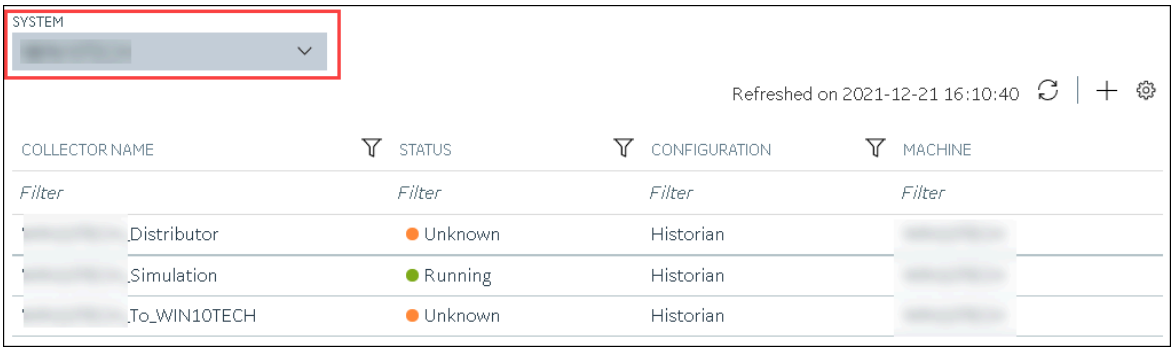
2. After you create Azure IoT Hub, select **Go to resource**, and then note down the hostname:



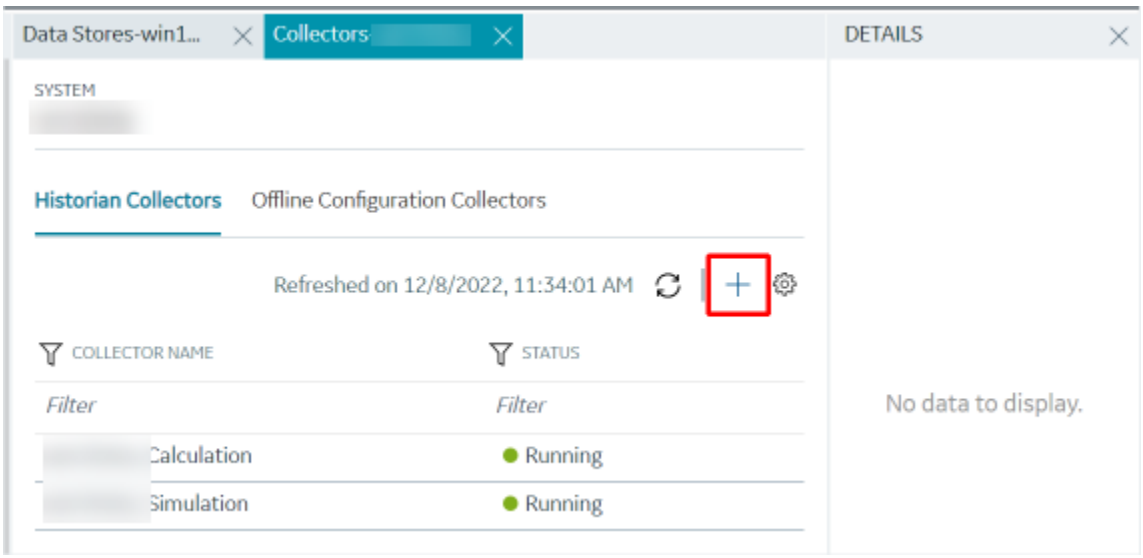

3. Create devices in Azure IoT Hub to group related tag information; thus mapping a collector instance to a device. We recommend that you create one device per collector instance. Ensure that the device is running.
   When you create a device, use the following guidelines to choose the authentication type:
   - **Symmetric Key**: Select this option if you want to use a Shared Access Signature (SAS) authentication.
   - **X.509 Self-Signed**: Select this option if you want to create self-signed certificates using OpenSSL. We recommend that you use these certificates only for testing purposes. For instructions, refer to https://docs.microsoft.com/en-us/azure/iot-hub/tutorial-x509-self-sign.
   - **X.509 CA Signed**: Select this option if you want to use CA-signed certificates
4. If you have selected **Symmetric Key** in the previous step, select the link in the **Device ID** column, and note down the shared access key value.



5. Access Configuration Hub *(on page 336)*.
6. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.
   A list of collectors in the default system appears.
7. If needed, select the system in which you want to add a collector instance.
8. If needed, select the system in which you want to add a collector instance.

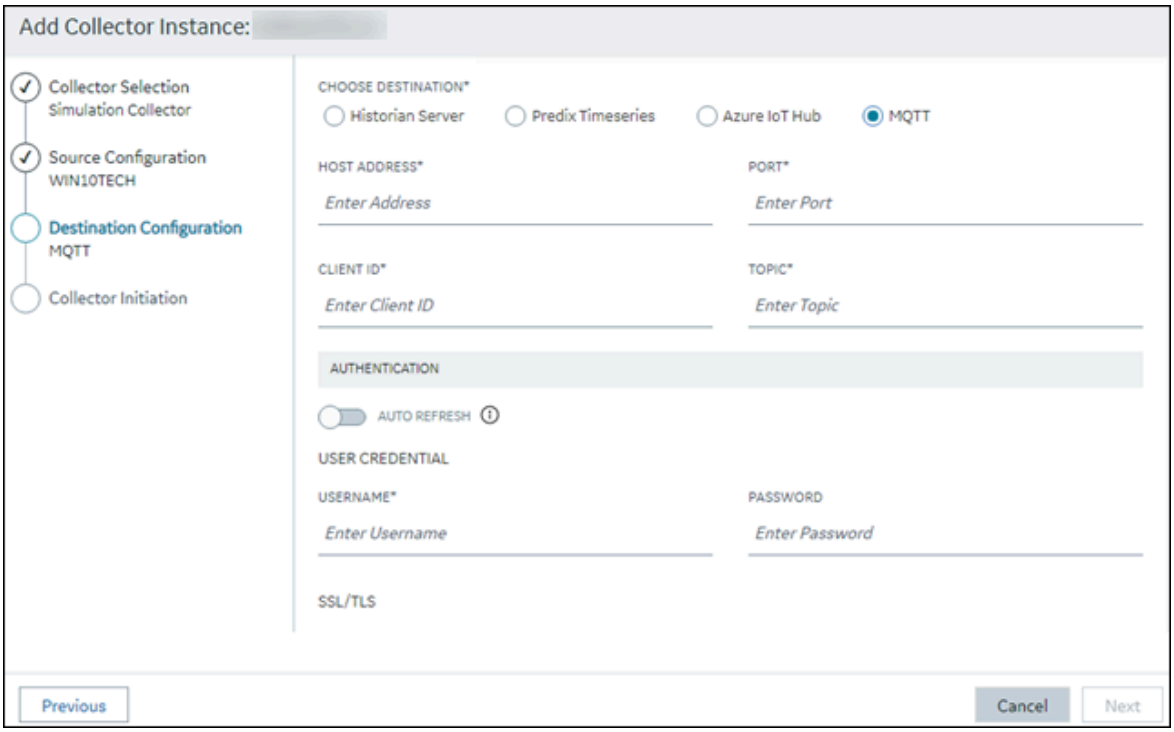



9. In the upper-right corner of the main section, select +.




   The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.
10. In the **COLLECTOR TYPE** field, select a collector type, and then select **Get Details**.
    The **INSTALLATION DRIVE** and **DATA DIRECTORY** fields are disabled and populated.
11. Select **Next**.
    The **Source Configuration** section appears.
12. As needed, enter values in the available fields.
13. Select **Next**.

The **Destination Configuration** section appears.




14. Select **MQTT**, and provide values as described in the following table.

| Field | Description |
|---|---|
| **HOST ADDRESS** | Enter the host name of the resource that you have noted down in step 2. A value is required and must be in the following format: `<Azure IoT Hub name>.azure-devices.net` |
| **PORT** | Enter `8883`. |
| **CLIENT ID** | Enter the ID of the device that you created in step 3. A value is required and must be unique for an MQTT broker. |
| **TOPIC** | Enter `devices/<device ID>/messages/events`. |
| **AUTO REFRESH** | Indicates whether you want to automatically create/refresh the SAS authentication token when it expires. |

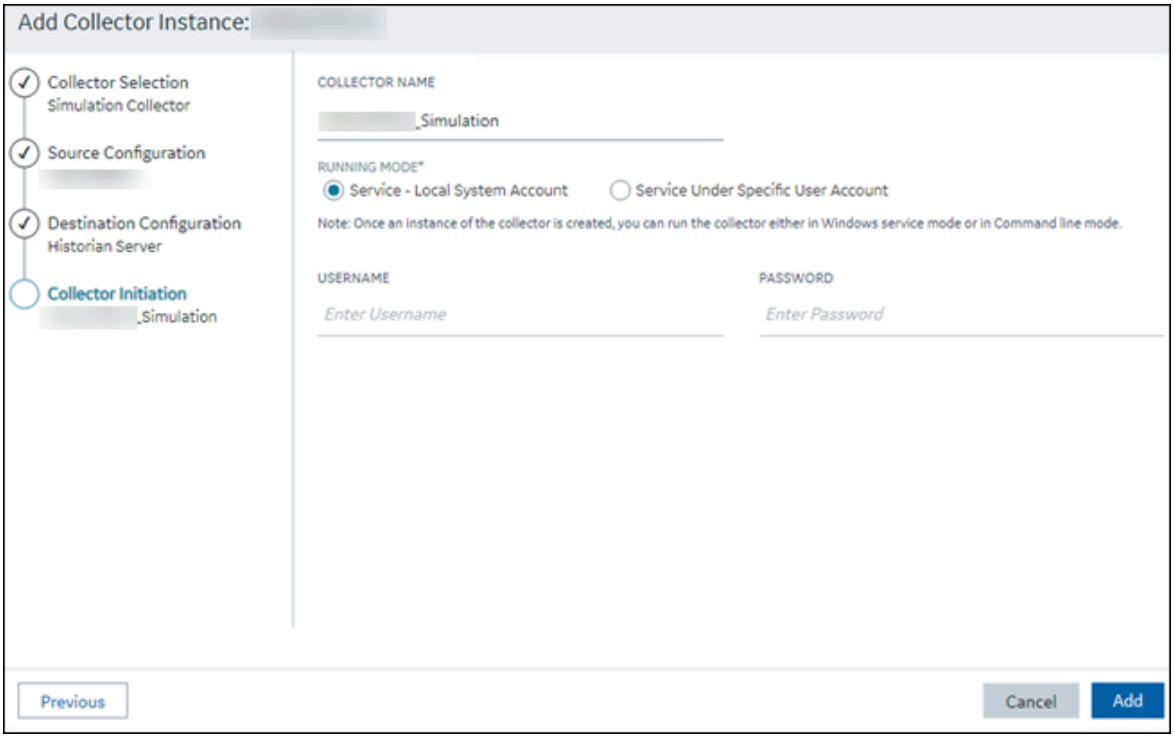
| Field | Description |
|---|---|
| | ◦ If you switch the toggle off, you must manually provide the token as soon as it expires.<br>◦ If you switch the toggle on, you must provide the shared access key that you have noted down in step 4. And, you can leave the **PASSWORD** field blank.<br>This is applicable only if you have selected **Symmetric Key** in step 3. |
| **USERNAME** | Enter a value in the following format: `<host name or IP address>/<device ID>/?api-version=2018-06-30` |
| **PASSWORD** | Enter the SAS token. This is applicable only if you have selected **Symmetric Key** in step 3 and if you have switched off the **AUTO REFRESH** toggle.<br><br>For instructions on generating a SAS token, refer to https://docs.microsoft.com/en-us/azure/cognitive-services/translator/document-translation/create-sas-tokens?tabs=Containers. |
| **DEVICE SHARED KEY** | Enter the shared access key value that you noted down in step 4. A value is required. This is applicable only if you have selected **Symmetric Key** in step 3 and if you have switched the **AUTO REFRESH** toggle on. |
| **CA SERVER ROOT FILE** | Enter the path of the CA server root file that you want to use. You can find the file here: https://github.com/Azure-Samples/IoTMQTTSample/blob/master/IoTHubRootCA_Baltimore.pem. |
| **CLIENT CERTIFICATE** | Enter the path to the client certificate. A value is required. This is applicable only if you have selected one of these options in step 3: |

| Field | Description |
|---|---|
| | ◦ **X.509 Self-Signed**: If you have selected this option, you can generate the certificate using OpenSSL.<br>◦ **X.509 CA Signed**: If you have selected this option, you would receive the certificate from CA. |
| **PRIVATE KEY FILE** | Enter the complete path to the private key file. A value is required. This is applicable only if you have selected one of these options in step 3:<br>◦ **X.509 Self-Signed**: If you have selected this option, you can generate the key file using OpenSSL.<br>◦ **X.509 CA Signed**: If you have selected this option, you would receive the key file from CA. |
| **PUBLIC KEY FILE** | Enter the path to the public key file. This is applicable only if you have selected one of these options in step 3:<br>◦ **X.509 Self-Signed**: If you have selected this option, you can generate the key file using OpenSSL.<br>◦ **X.509 CA Signed**: If you have selected this option, you would receive the key file from CA. |
| **CHOOSE CONFIGURATION** | The type of the configuration to specify the tags whose data you want to collect. Select one of the following options:<br>◦ **Historian Configuration**: Select this option if you want to add the tags manually *(on page 357)*. If you select this option, the **CONFIGURATION HISTORIAN SERVER** field appears.<br>◦ **Offline Configuration**: Select this option if you want to provide the tag names using the offline configuration *(on page* |

| Field | Description |
|---|---|
| | *2039)* file instead of adding tags manually. By default, this file is located in the following location: <*installation folder of Historian*>\GE Digital\<*collector name*> |
| **CONFIGURATION HISTORIAN SERVER** | The host name of the machine from which you want to access Historian Administrator to add the tags manually for the collector. This field appears only if you have selected **Historian Configuration** in the **CHOOSE CONFIGURATION** field. |

15. Select **Next**.

    The **Collector Initiation** section appears.




16. Enter a collector name.

    The value that you enter:

    - Must be unique.
    - Must not exceed 15 characters.

- Must not contain a space.
- Must not contain special characters except a hyphen, period, and an underscore.

17. In the **RUNNING MODE** field, select one of the following options.
    - **Service - Local System Account**: Select this option if you want to run the collector as a Windows service using the credentials of the local user (that is, the currently logged-in user). If you select this option, the **USERNAME** and **PASSWORD** fields are disabled.
    - **Service Under Specific User Account**: Select this option if you want to run the collector as a Windows service using a specific user account. If you select this option, you must enter values in the **USERNAME** and **PASSWORD** fields. If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:
        - iH Security Admins
        - iH Collector Admins
        - iH Tag Admins

    If you choose the first option, you can also configure the collector to start automatically when you start the computer, or, in the case of iFIX collectors, whenever you start iFIX.
18. Select **Add**.
    The collector instance is created.
19. Specify the tags for which you want to collect data.
    - If you have selected **Historian Configuration** in the **CHOOSE CONFIGURATION** field, specify the tags manually *(on page 357)*.
    - If you have selected **Offline Configuration** in the **CHOOSE CONFIGURATION** field, specify the tags using the offline configuration file *(on page 2039)*.

    The collector begins sending Historian data to the Azure IoT Hub device that you have created.

## Send Data to Azure Cloud in the KairosDB Format

To send data to an Azure IoT Hub device, you can choose any of the following collectors:

- The iFIX collector
- The MQTT collector
- The ODBC collector
- The OPC Classic DA collector
- The OPC Classic HDA collector
- The OPC UA DA collector
- The OSI PI collector
- The Server-to-Server collector

- The Simulation collector
- The Windows Performance collector
- The Wonderware collector

This topic describes how to send data in the KairosDB format. In this format, the message size is less because names of the tag properties are not repeated. For example: `[{"<tag name>":"Cloud_GCYSS3X2E.Simulation00001","<timestamp, tag value, and quality>":[[1586260104000,132560.203125000,3]]}`. If you use this format, you can only use SAS-based authentication; you cannot use certificate-based authentication.

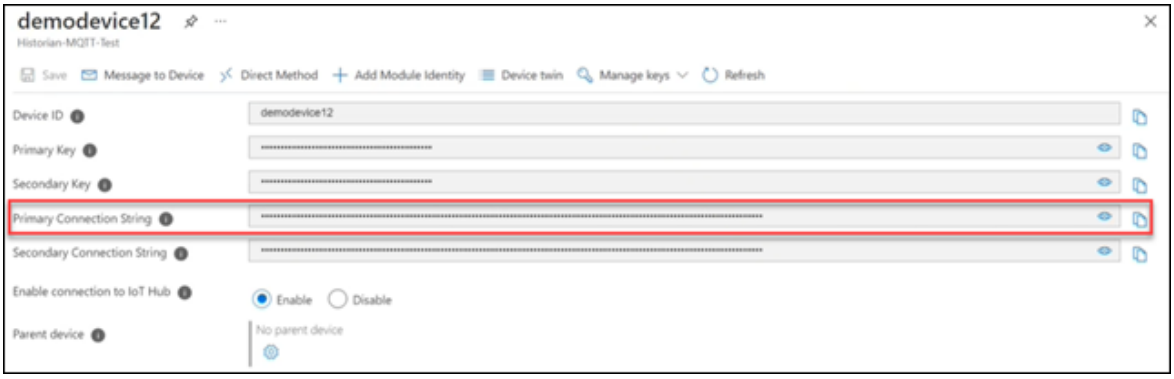
You can also send data in the key-value format *(on page 601)*.

**Note:**
Data in Azure IoT Hub is stored for maximum seven days, after which it is deleted from the hub. Therefore, you must consume the data within seven days. Based on your requirement, you can store it in a relevant Azure storage. You can then use Azure functions or streaming analytics to analyse the data.

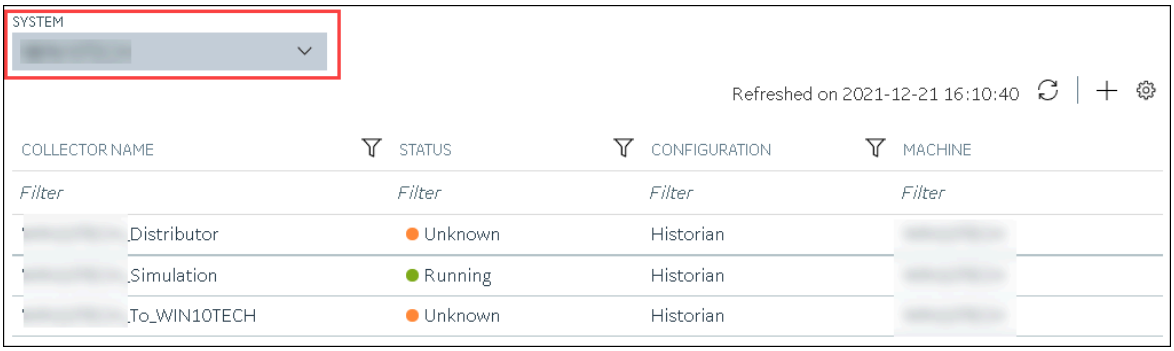
1. Create Azure IoT Hub.

   **Tip:**
   To choose the correct Azure IoT Hub tier based on your data throughput, refer to https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-devguide-quotas-throttling. For guidance on choosing the appropriate subscription, refer to https://azure.microsoft.com/en-us/pricing/details/iot-hub/

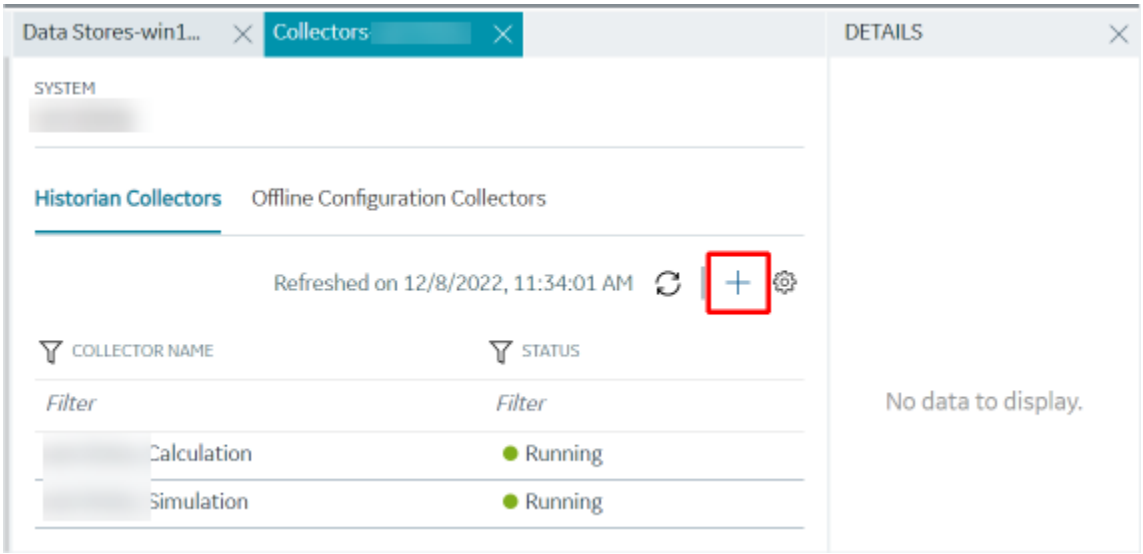
2. Create devices in Azure IoT Hub to group related tag information; thus mapping a collector instance to a device. We recommend that you create one device per collector instance. Ensure that the device is running.
   When you create a device, use only in the Symmetric Key authentication.
3. Select the link in the **Device ID** column, and note down the primary connection string value.




4. Access Configuration Hub *(on page 336)*.
5. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.
   A list of collectors in the default system appears.
6. If needed, select the system in which you want to add a collector instance.
7. If needed, select the system in which you want to add a collector instance.

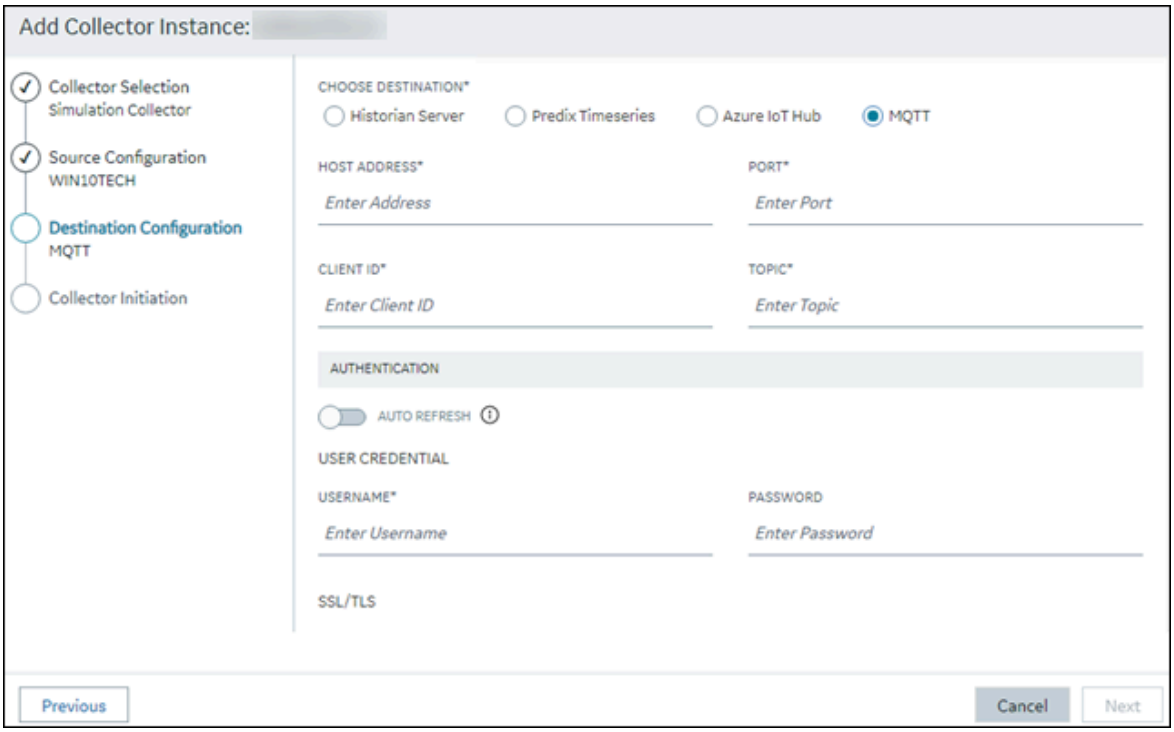



8. In the upper-right corner of the main section, select +.



The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.

9. In the **COLLECTOR TYPE** field, select a collector type, and then select **Get Details**.
   The **INSTALLATION DRIVE** and **DATA DIRECTORY** fields are disabled and populated.
10. Select **Next**.
    The **Source Configuration** section appears.
11. As needed, enter values in the available fields.
12. Select **Next**.
    The **Destination Configuration** section appears.




13. Select **Azure IoT Hub**, and provide values as described in the following table.

| Field | Description |
|---|---|
| **DEVICE CONNECTION STRING** | Identifies the Azure IoT device to which you want to send data. Enter the primary connection string value that you have noted down in step 3. |

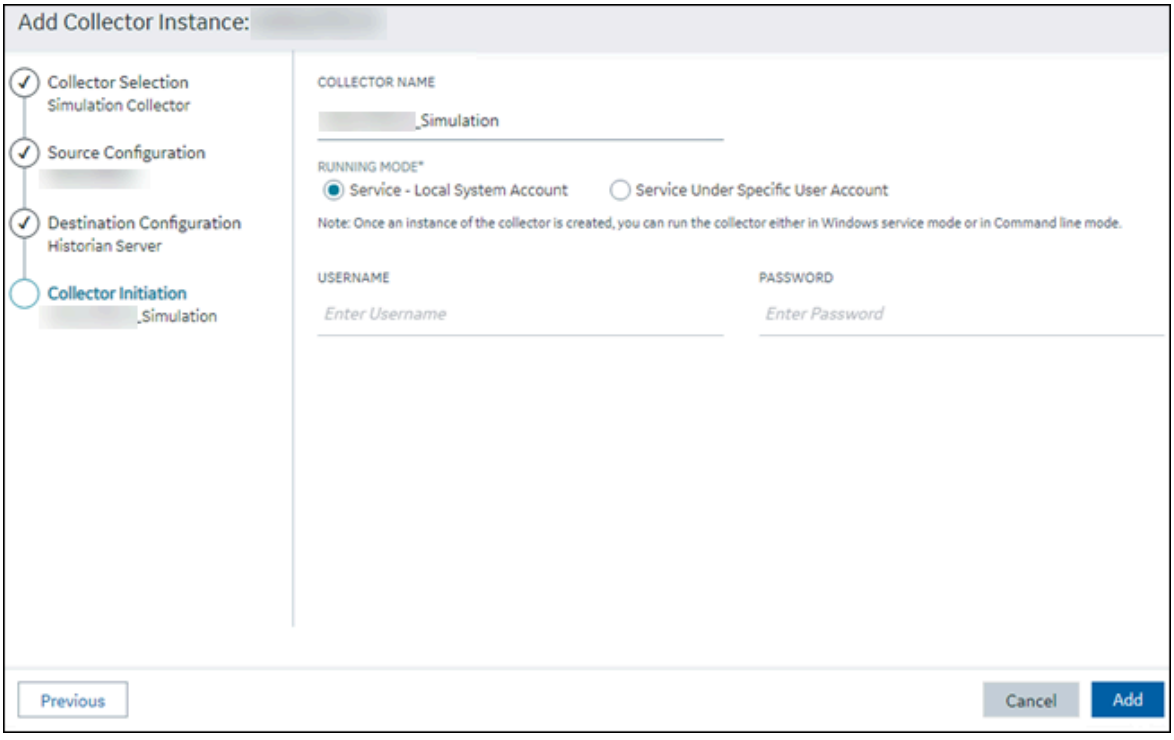
| Field | Description |
|---|---|
| **TRANSPORT PROTOCOL** | The protocol that you want to use to send data to Azure IoT Hub. Select one of the following options:<br>◦ HTTP<br>◦ MQTT<br>◦ AMQP<br>◦ MQTT_OVER_WEBSOCKETS<br>◦ AMQP_OVER_WEBSOCKETS<br>For information on which protocol to use, refer to Protocols and Port Numbers *(on page 626)*. |
| **PROXY** | Identifies the URL of the proxy server to be used for both the authentication process and for sending data. If the collector is running on a network where proxy servers are used to access web resources outside of the network, then you must provide the proxy server settings. However, it does not affect the proxy server used by Windows when establishing secure connections. As a result, you must still configure the proxy settings for the Windows user account under which the collector service runs. |
| **PROXY USERNAME** | The username to connect to the proxy server. |
| **PROXY PASSWORD** | The password to connect to the proxy server. |
| **CHOOSE CONFIGURATION** | The type of the configuration to specify the tags whose data you want to collect. Select one of the following options:<br>◦ **Historian Configuration**: Select this option if you want to add the tags manually *(on page 357)*. If you select this option, the **CONFIGURATION HISTORIAN SERVER** field appears. |

| Field | Description |
|---|---|
| | ◦ **Offline Configuration**: Select this option if you want to provide the tag names using the offline configuration *(on page 2039)* file instead of adding tags manually. By default, this file is located in the following location: <*installation folder of Historian*>\GE Digital\<*collector name*> |
| **CONFIGURATION HISTORIAN SERVER** | The host name of the machine from which you want to access Historian Administrator to add the tags manually for the collector. This field appears only if you have selected **Historian Configuration** in the **CHOOSE CONFIGURATION** field. |

The collector instance is created and connected to the Azure IoT Hub device.

14. Select **Next**.

    The **Collector Initiation** section appears.




15. Enter a collector name.

The value that you enter:
- Must be unique.
- Must not exceed 15 characters.
- Must not contain a space.
- Must not contain special characters except a hyphen, period, and an underscore.

16. In the **RUNNING MODE** field, select one of the following options.
    - **Service - Local System Account**: Select this option if you want to run the collector as a Windows service using the credentials of the local user (that is, the currently logged-in user). If you select this option, the **USERNAME** and **PASSWORD** fields are disabled.
    - **Service Under Specific User Account**: Select this option if you want to run the collector as a Windows service using a specific user account. If you select this option, you must enter values in the **USERNAME** and **PASSWORD** fields. If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:
        - iH Security Admins
        - iH Collector Admins
        - iH Tag Admins

    If you choose the first option, you can also configure the collector to start automatically when you start the computer, or, in the case of iFIX collectors, whenever you start iFIX.
17. Select **Add**.
    The collector instance is created.
18. Specify the tags for which you want to collect data.
    - If you have selected **Historian Configuration** in the **CHOOSE CONFIGURATION** field, specify the tags manually *(on page 357)*.
    - If you have selected **Offline Configuration** in the **CHOOSE CONFIGURATION** field, specify the tags using the offline configuration file *(on page 2039)*.

    The collector begins sending Historian data to the Azure IoT Hub device that you have created.

## Send Data to Google Cloud

1. Download the Google root CA certificate from https://pki.google.com/roots.pem.
2. Create public/private key pairs. Use OpenSSL only for testing purposes.

To send data to a Google Cloud device, you can choose any of the following collectors:

- The iFIX collector
- The MQTT collector

- The ODBC collector
- The OPC Classic DA collector
- The OPC Classic HDA collector
- The OPC UA DA collector
- The OSI PI collector
- The Python Collector
- The Server-to-Server collector
- The Simulation collector
- The Windows Performance collector
- The Wonderware collector

1. Access Google Cloud Platform.
2. Create a project. Note down the project ID.
3. Create a registry.

   When you create the registry:

   - Use the MQTT protocol.
   - You can choose to provide a CA certificate.

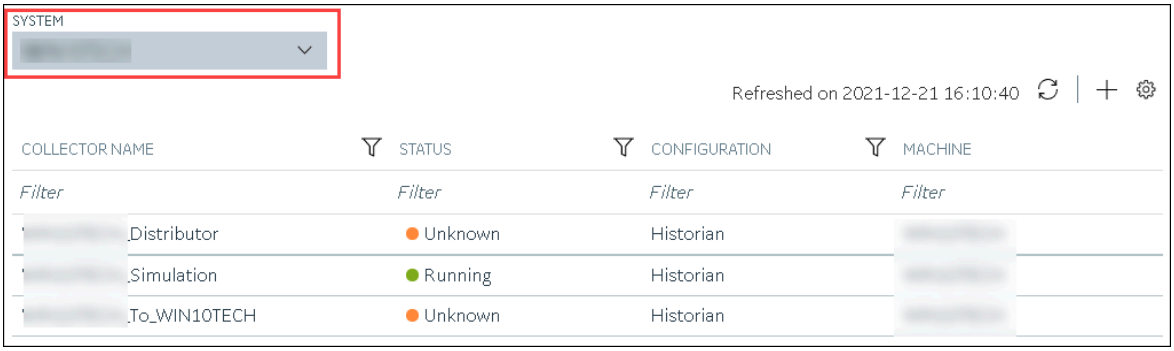
   Note down the registry ID and the region values.
4. Add a device to the registry.

   When you add the device:

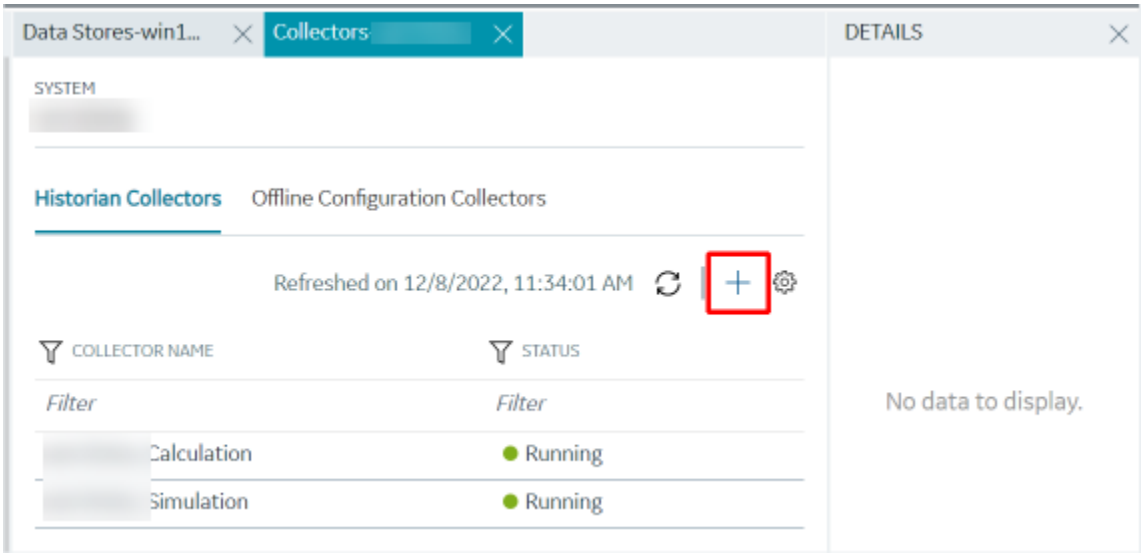
   - Allow device communication.
   - Upload the public key or enter the details manually.

   Note down the device ID.
5. Access Configuration Hub *(on page 336)*.
6. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.

   A list of collectors in the default system appears.
7. If needed, select the system in which you want to add a collector instance.
8. If needed, select the system in which you want to add a collector instance.

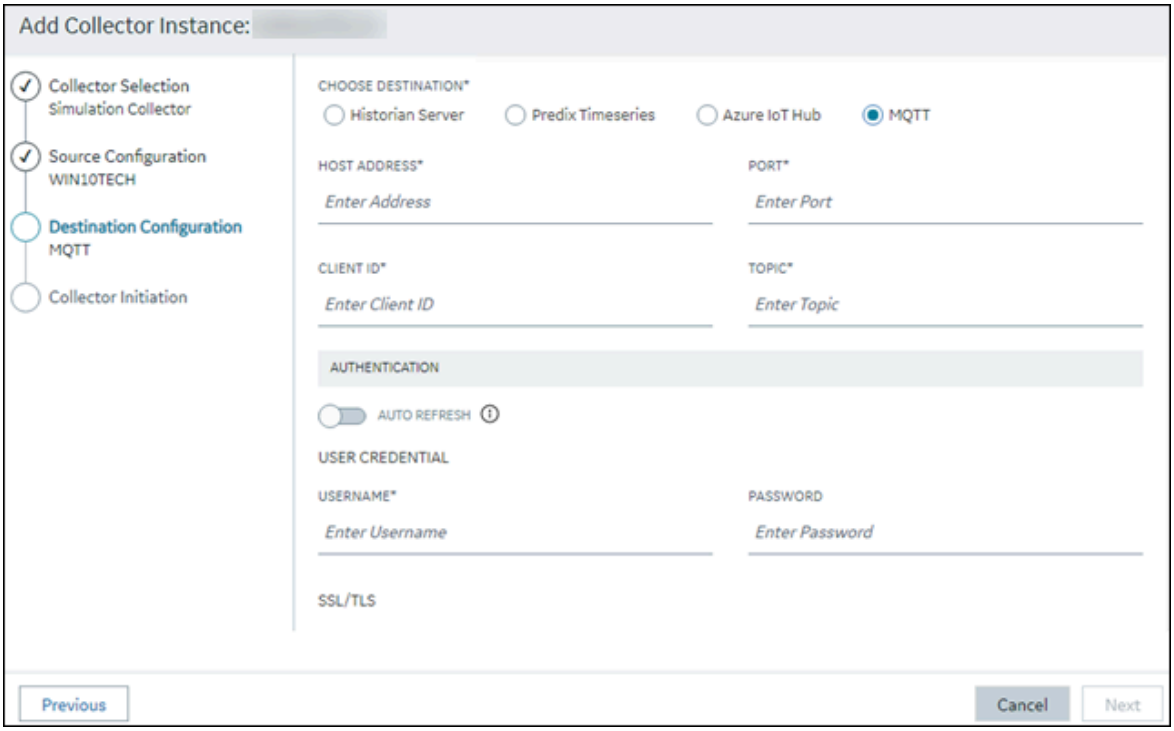



9. In the upper-right corner of the main section, select +.




The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.

10. In the **COLLECTOR TYPE** field, select a collector type, and then select **Get Details**.
    The **INSTALLATION DRIVE** and **DATA DIRECTORY** fields are disabled and populated.
11. Select **Next**.
    The **Source Configuration** section appears.
12. As needed, enter values in the available fields.
13. Select **Next**.
    The **Destination Configuration** section appears.




14. Select **MQTT**, and provide values as described in the following table.

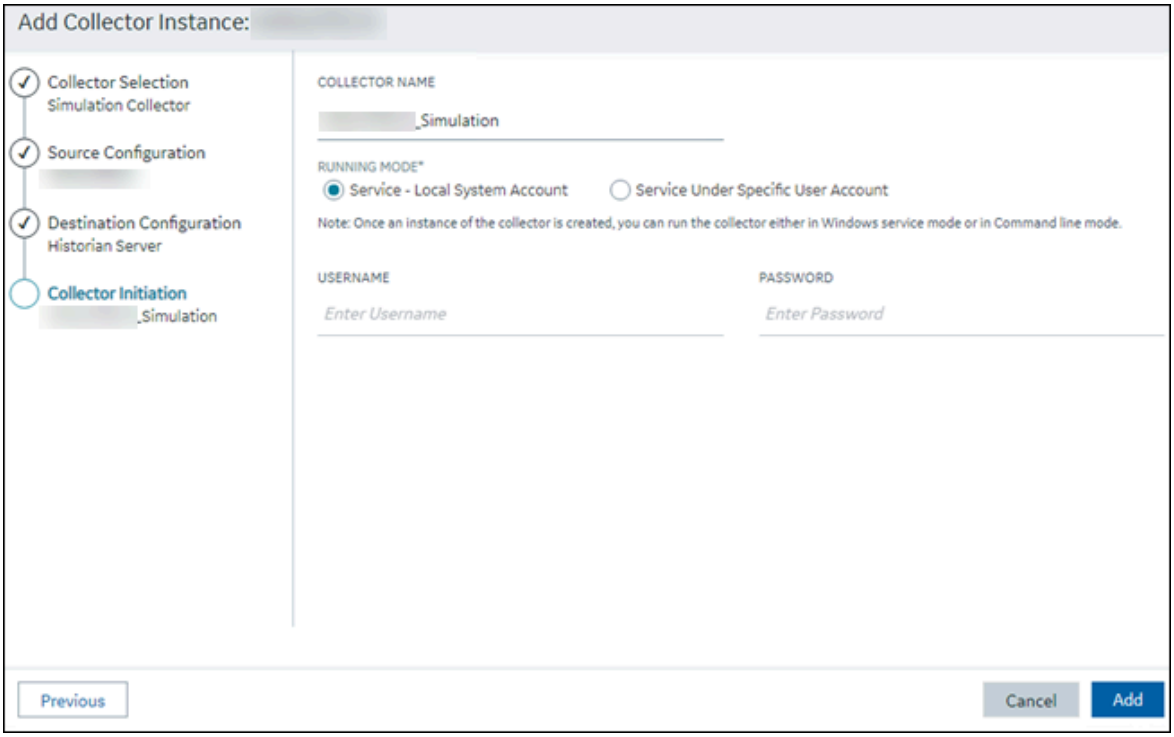
| Field | Description |
|---|---|
| **HOST ADDRESS** | Enter `mqtt.googleapis.com`. A value is required. |
| **PORT** | Enter `8883` or `443`. |
| **CLIENT ID** | Enter the ID of the device that you created in the following format: `projects/<project ID>/locations/<cloud region>/registries/<registry ID>/devices/<device ID>`. For example: `projects/mygcpproject/locations/asia-east1/registries/testmqttgcpiot/devices/gcptesting`<br><br>A value is required and must be unique for an MQTT broker. |
| **TOPIC** | Enter `devices/<device ID>/events`. |
| **AUTO REFRESH** | Indicates whether you want to automatically refresh the authentication token when it expires. If you switch the toggle off, you must manually |

| Field | Description |
|---|---|
| | provide the token as soon as it expires. Google Cloud accepts only those tokens that expire in 24 hours or less; therefore, we recommend that you switch the toggle on. |
| **USERNAME** | Enter any value. This value is not used, but only if you enter a value, you can proceed. |
| **PASSWORD** | If you have switched the **AUTO REFRESH** toggle on, leave this field blank. Historian generates a JSON Web Token (JWT) and uses it automatically. |
| **CA SERVER ROOT FILE** | Enter the path of the Google root CA certificate that you have downloaded. |
| **CLIENT CERTIFICATE** | Enter the path to the client certificate. |
| **PRIVATE KEY FILE** | Enter the complete path to the private key file. A value is required. |
| **PUBLIC KEY FILE** | Enter the path to the public key file. A value is required. |
| **CHOOSE CONFIGURATION** | The type of the configuration to specify the tags whose data you want to collect. Select one of the following options:<br>◦ **Historian Configuration**: Select this option if you want to add the tags manually *(on page 357)*. If you select this option, the **CONFIGURATION HISTORIAN SERVER** field appears.<br>◦ **Offline Configuration**: Select this option if you want to provide the tag names using the offline configuration *(on page 2039)* file instead of adding tags manually. By default, this file is located in the following location: `<installation folder of Historian>\GE Digital\<collector name>` |

| Field | Description |
|---|---|
| **CONFIGURATION HISTORIAN SERVER** | The host name of the machine from which you want to access Historian Administrator to add the tags manually for the collector. This field appears only if you have selected **Historian Configuration** in the **CHOOSE CONFIGURATION** field. |

15. Select **Next**.

    The **Collector Initiation** section appears.




16. Enter a collector name.

    The value that you enter:

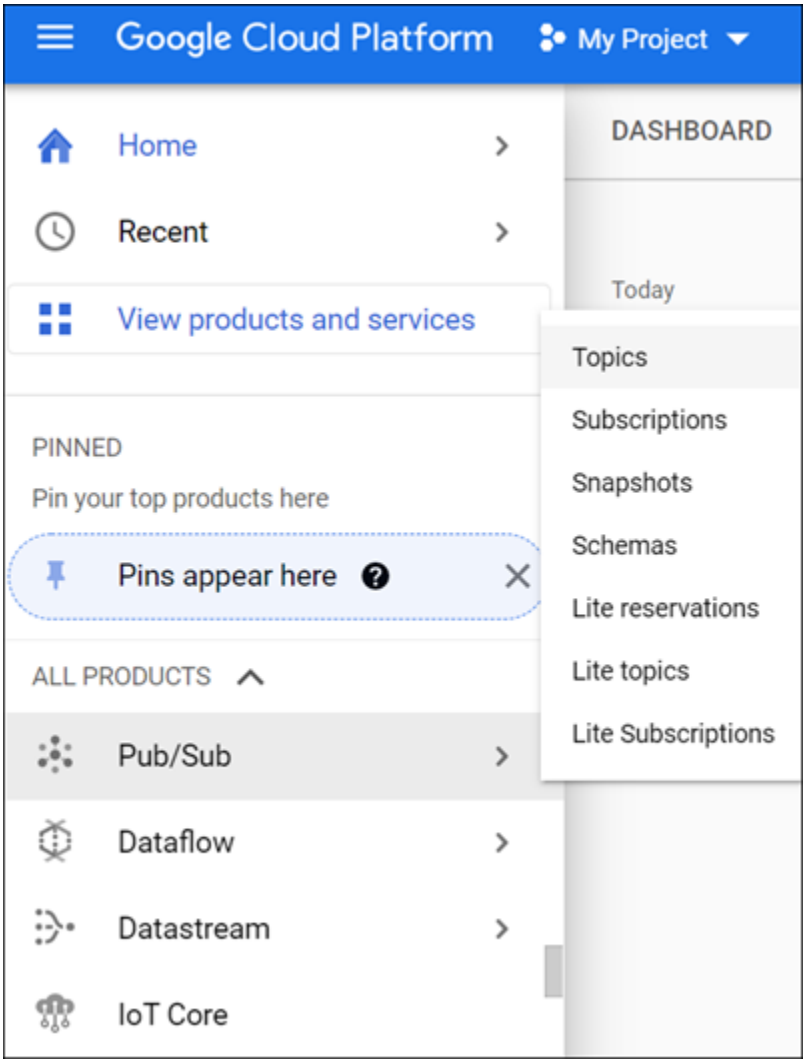
    - Must be unique.
    - Must not exceed 15 characters.
    - Must not contain a space.
    - Must not contain special characters except a hyphen, period, and an underscore.

17. In the **RUNNING MODE** field, select one of the following options.

- **Service - Local System Account**: Select this option if you want to run the collector as a Windows service using the credentials of the local user (that is, the currently logged-in user). If you select this option, the **USERNAME** and **PASSWORD** fields are disabled.
- **Service Under Specific User Account**: Select this option if you want to run the collector as a Windows service using a specific user account. If you select this option, you must enter values in the **USERNAME** and **PASSWORD** fields. If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:
  - iH Security Admins
  - iH Collector Admins
  - iH Tag Admins

If you choose the first option, you can also configure the collector to start automatically when you start the computer, or, in the case of iFIX collectors, whenever you start iFIX.

18. Select **Add**.
    The collector instance is created.
19. Access Google Cloud Platform, and select **Pub/Sub > Topics**.




20. Select **Messages > PULL**.
    Messages published to the topic that you have created appear. These messages contain the data sent by the collector instance. You can verify that the message content is correct by selecting **Message body**.

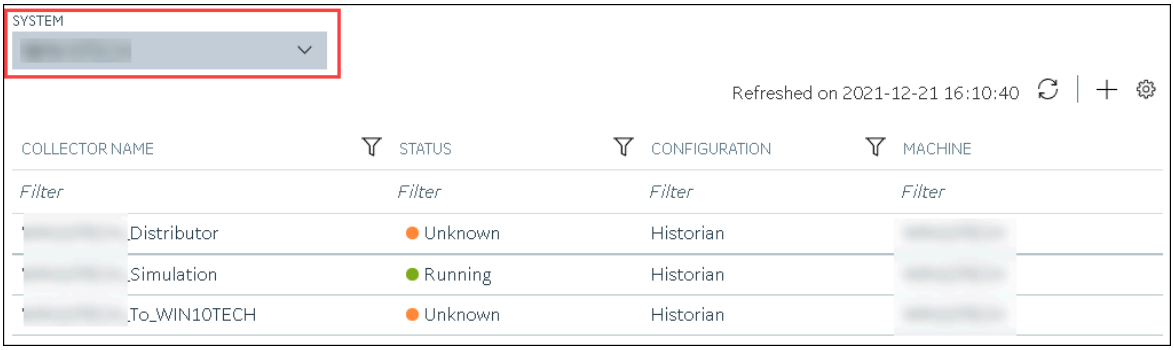
## Send Data to Predix Cloud

To send data to Predix Cloud, you can choose any of the following collectors:

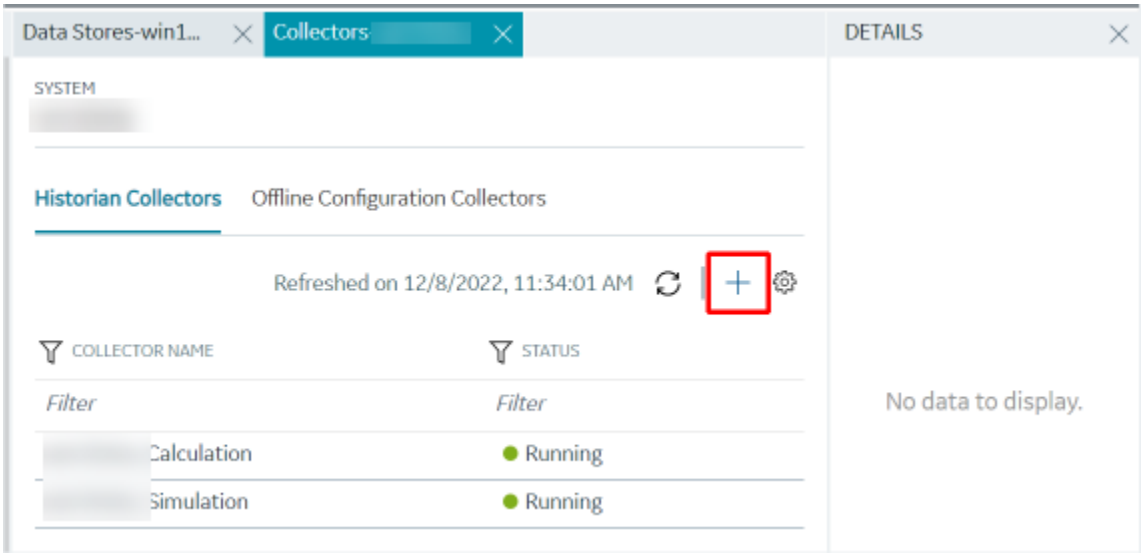
- The iFIX collector
- The MQTT collector
- The ODBC collector
- The OPC Classic DA collector
- The OPC Classic HDA collector

- The OPC UA DA collector
- The OSI PI collector
- The Python Collector
- The Server-to-Server collector
- The Simulation collector
- The Windows Performance collector
- The Wonderware collector

1. Register with the Timeseries service or any UAA service that you want to use. Note down the destination address, URI, client ID, client secret, and the zone ID that you have provided.
2. Access Configuration Hub *(on page 336)*.
3. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**. A list of collectors in the system appears.
4. If needed, select the system in which you want to add a collector instance.

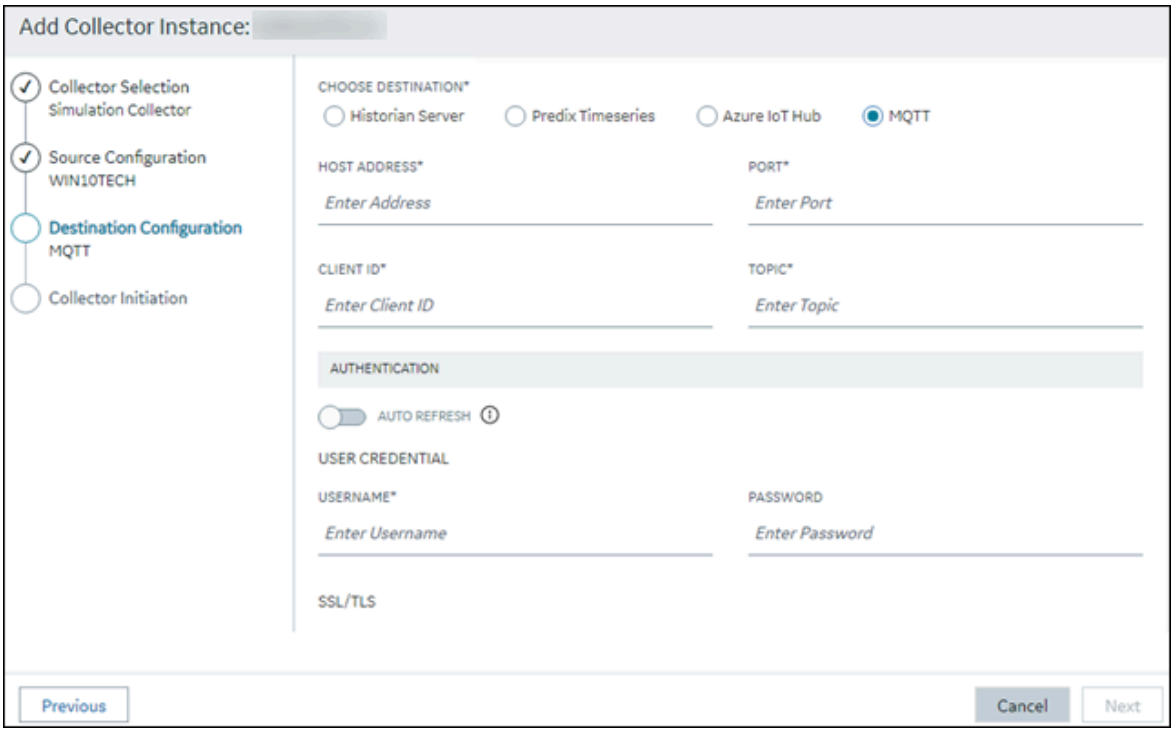



5. In the upper-right corner of the main section, select +.



The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.

6. In the **COLLECTOR TYPE** field, select a collector type (except the File collector and the Server-to-Server collector), and then select **Get Details**.
   The **INSTALLATION DRIVE** and **DATA DIRECTORY** fields are disabled and populated.
7. Select **Next**.
   The **Source Configuration** section appears, populating the hostname of the collector machine.
8. As needed, enter values in the available fields, and then select **Next**.
   The **Destination Configuration** section appears.




9. In the **CHOOSE DESTINATION** field, select **Predix Timeseries**, and then provide values as described in the following table.

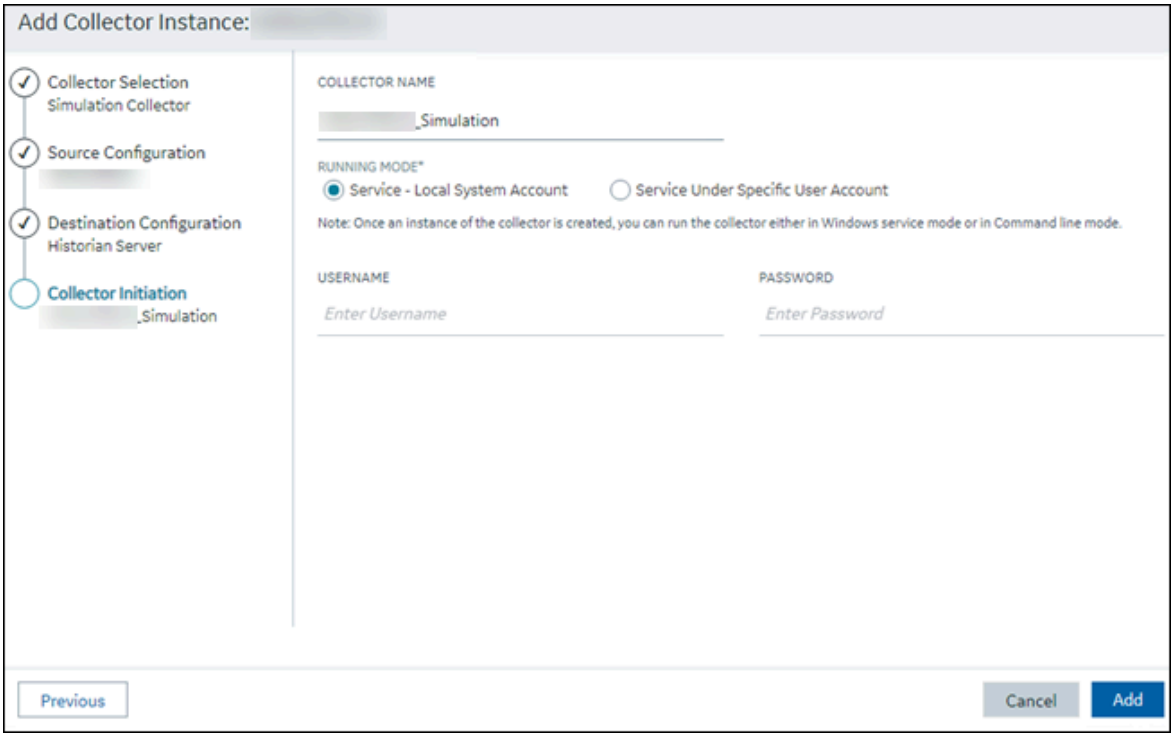
| Field | Description |
|---|---|
| **CLOUD DESTINATION ADDRESS** | The URL of a data streaming endpoint exposed by the Predix Time Series instance to which you want to send data. Typically, it starts with “wss://”. This value is used as part of the interface name and default tag prefix of the collector. Your Predix Time Series administrator can provide this URL. |

| Field | Description |
|---|---|
| **IDENTITY ISSUER** | The URL of an authentication endpoint for the collector to authenticate itself and acquire necessary credentials to stream to the Predix Time Series. In other words, this is the issuer ID of the Proficy Authentication instance that you want to use to connect to Predix Time Series. Typically, it starts with https:// and ends with "/oauth/token". |
| **CLIENT ID** | Identifies the collector when interacting with Predix Time Series. This is equivalent to the username in many authentication schemes. The client must exist in the Proficy Authentication instance identified by the identity issuer, and the system requires that the `timeseries.zones. {ZoneId}.ingest` and `timeseries.zones.{ZoneId}.query` authorities are granted access to the client for the Predix Zone ID specified. Your Predix Time Series administrator can provide this information. |
| **CLIENT SECRET** | The secret to authenticate the collector. This is equivalent to the password in many authentication schemes. |
| **ZONE ID** | Unique identifier of the instance to which the collector will send data. |
| **PROXY** | Identifies the URL of the proxy server to be used for both the authentication process and for sending data. If the collector is running on a network where proxy servers are used to access web resources outside of the network, then you must provide the proxy server settings. However, it does not affect the proxy server used by Windows when establishing secure connections. As a result, you must still configure the proxy settings for the Windows user account under which the collector service runs. |
| **PROXY USERNAME** | The username to connect to the proxy server. |
| **PROXY PASSWORD** | The password to connect to the proxy server. |
| **DATAPOINT ATTRIBUTES** | The attributes or parameters related to a datapoint that you want the collector to collect. Select **Add Attributes** to specify the attributes. You can add maximum five attributes for each collector instance. |

| Field | Description |
|---|---|
| **CHOOSE CONFIGURATION** | The type of the configuration to specify the tags whose data you want to collect. Select one of the following options:<br>◦ **Historian Configuration**: Select this option if you want to add the tags manually *(on page 357)*. If you select this option, the **CONFIGURATION HISTORIAN SERVER** field appears.<br>◦ **Offline Configuration**: Select this option if you want to provide the tag names using the offline configuration *(on page 2039)* file instead of adding tags manually. By default, this file is located in the following location: `<installation folder of Historian>\GE Digital\<collector name>` |
| **CONFIGURATION HISTORIAN SERVER** | The host name of the machine from which you want to access Historian Administrator to add the tags manually for the collector. This field appears only if you have selected **Historian Configuration** in the **CHOOSE CONFIGURATION** field. |

10. Select **Next**.

    The **Collector Initiation** section appears.



11. Enter a collector name.
    The value that you enter:
    - Must be unique.
    - Must not exceed 15 characters.
    - Must not contain a space.
    - Must not contain special characters except a hyphen, period, and an underscore.
12. In the **RUNNING MODE** field, select one of the following options.
    - **Service - Local System Account**: Select this option if you want to run the collector as a Windows service using the credentials of the local user (that is, the currently logged-in user). If you select this option, the **USERNAME** and **PASSWORD** fields are disabled.
    - **Service Under Specific User Account**: Select this option if you want to run the collector as a Windows service using a specific user account. If you select this option, you must enter values in the **USERNAME** and **PASSWORD** fields. If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:
      - iH Security Admins
      - iH Collector Admins
      - iH Tag Admins

    If you choose the first option, you can also configure the collector to start automatically when you start the computer, or, in the case of iFIX collectors, whenever you start iFIX.
13. Select **Add**.
    The collector instance is created.
14. Specify the tags for which you want to collect data.
    - If you have selected **Historian Configuration** in the **CHOOSE CONFIGURATION** field, specify the tags manually *(on page 357)*.
    - If you have selected **Offline Configuration** in the **CHOOSE CONFIGURATION** field, specify the tags using the offline configuration file *(on page 2039)*.

    The collector begins sending Historian data to Predix Timeseries.

## Protocols and Port Numbers

The following table provides a list of protocols that are available to send data to Azure IoT Hub, guidelines on which protocol to choose, and the port number that each protocol uses.

| Protocol | When to Use | Port Number |
|---|---|---|
| HTTP | Use this protocol if the data that you want to send is not large and/or the default ports for the other protocols are not available. | 80 |
| MQTT | MQTT is lightweight compared to AMQP, and is widely used. Use this protocol if you want to send data using low bandwidth and/or you do not want to connect to multiple devices using the same connection. | 8883 |
| AMQP | AMQP is more reliable compared to other protocols. It sends data in batches, and hence, the network traffic is less compared to that of MQTT. Use this protocol if you want to send a large amount of data from multiple collectors frequently. | 5671 |
| MQTT over web sockets | MQTT is lightweight compared to AMQP, and is widely used. In addition, communication using web sockets is more reliable and secure. Use this protocol if you want to send data using low bandwidth and securely. | 443 |
| AMQP over web sockets | AMQP is more reliable compared to other protocols. It sends data in batches, and hence, the network traffic is less compared to that of MQTT. In addition, communication using web sockets is more reliable and secure. Use this protocol if you want to send a large amount of data from multiple collectors frequently and securely. | 443 |

# Managing Collector Instances

## About Managing Collectors Using Configuration Hub

Collectors are used to collect data from various sources and send it to Historian. For a list of collectors and their usage, refer to About Historian Data Collectors *(on page 1990)*.

After you install collectors and Remote Management Agent, the following artefacts will be available:

- **Executable files:** These files are required to add a collector instance.
- **Instances of the following collectors:**
  - The iFIX collector
  - The iFIX Alarms & Events collector
  - The OPC Classic Data Access collector for CIMPLICITY
  - The OPC Classic Alarms and Events collector for CIMPLICITY

  These instances will be created only if iFIX and/or CIMPLICITY are installed on the same machine as the collectors.
- **The Remote Collector Management agent:** Provides the ability to manage collectors remotely.

You can then add a collector instance. This section describes how to add a collector instance using Configuration Hub *(on page 357)*. You can also add a collector instance using the RemoteCollectorConfigurator utility *(on page 797)*, which does not require you to install Web-based Clients.

## Access a Collector Instance

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.
   A list of collectors appears, displaying the following columns:

| Column | Description |
|---|---|
| **COLLECTOR NAME** | The name of the collector instance. If you select the link in this column, the details of the collector instance appears. |
| **STATUS** | The status of the collector. Contains one of the following values:<br>◦ **Started**<br>◦ **Stopped**<br>◦ **Running**<br>◦ **Paused** |
| **CONFIGURATION** | The source of the tag configuration for the collector. Contains one of the following values:<br>◦ **HISTORIAN**: Indicates that tags are configured using Historian Administrator.<br>◦ **OFFLINE**: Indicates that tags are configured using an offline configuration *(on page 2039)* file. |

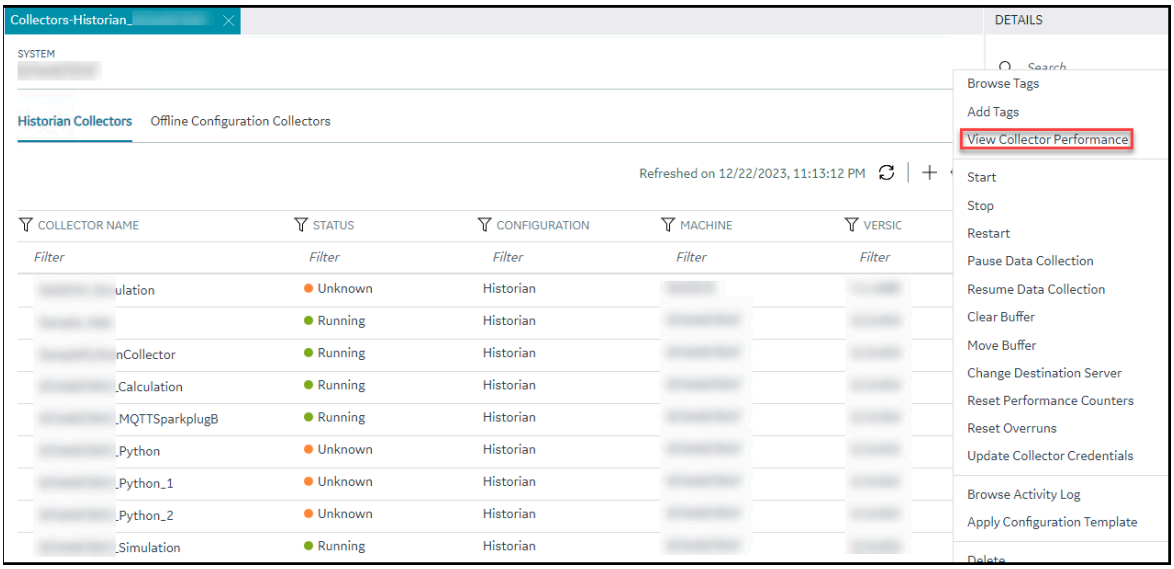
| Column | Description |
|---|---|
| **MACHINE** | The name of the machine on which the collector is installed. |
| **VERSION** | The version number of the collector. |
| **REPORT RATE** | The average rate at which the collector is sending data. This is a general indicator of load on the collector. |
| **OVERRUNS** | The total number of data events not collected. In normal operation and under normal conditions, this value should always be zero. If the value is not zero, which indicates that data is being lost, you must take steps to reduce peak load on the system by increasing the collection interval. |
| **COMPRESSION** | The effectiveness of collector compression. If the value is low, you can increase the compression deadbands to pass fewer values and thus increase the effect of compression. |
| **OUT OF ORDER** | The total number of out-of-order samples for the collector. |
| **REDUNDANCY** | Indicates whether collector redundancy is enabled, which decreases the likelihood of lost data due to soft- ware or hardware failures. For information, refer to About Collector Redundancy *(on page 975)*. |
| **TAG COUNT** | The number of tags for which the collector collects data. |
| **COMMENTS** | The comments that you have entered for the collector. |

**Tip:**
You can also add, reorder, and remove columns from the table. For instructions, refer to Common Tasks in Configuration Hub *(on page 353)*.

3. Select the row containing the collector whose details you want to access.
   The details of the collector appear in the **DETAILS** section.

**Note:**
If the **DETAILS** section does not appear, in the upper-right corner of the page, select , and then select **Details**.

4. If you want to access the collector performance, right-click the collector (or select ⋯), and then select **View Collector Performance**.




The **Collector Performance** section appears, displaying the following graphs:

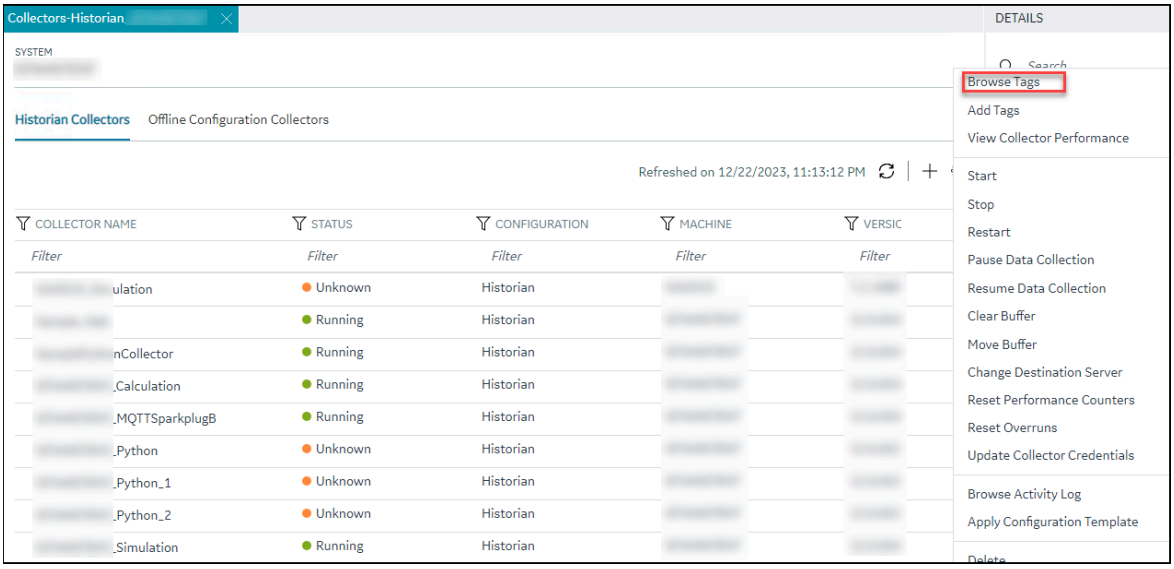
- **REPORT RATE**: The rate at which the collector collects data.
- **TOTAL EVENTS REPORTED**: The total number of events reported to the Historian archive from the collector. This number may not match the total events collected due to collector compression.
- **STATUS**: The status of the collector plotted at regular intervals.
- **COLLECTOR COMPRESSION**: The collector compression (in percentage) applied to tag values plotted at regular intervals.
- **OUT OF ORDER**: The number of samples that have been received out of sequence. Even though data is still stored, a steadily increasing number of out-of-order events indicates a problem with data transmission that you should investigate. For example, a steadily increasing number of out-of-order events when you are using the OPC Collector means that there is an out-of-order between the OPC server and the OPC collector. This may also cause an out-of-order between the OPC collector and the data archiver but that is not what this graph indicates.
- **OVERRUNS**: The number of overruns in relation to the total events collected. Overruns are a count of the total number of data events not collected on their scheduled polling cycle. An overrun occurs when the data source is changing tag values faster than the collector collecting values, which causes it to consistently remain behind the archiver updates. It implies that the collector is running against the hardware and/or network limits and you may consider partitioning the tags into two or more sets, each with separate collector instances.

In a normal operation, this value should be zero. You may be able to reduce the number of overruns on the collector by increasing the tag collection intervals (per tag).

- **MINIMUM EVENT RATE**: Specifies the minimum number of data samples per minute sent to the archiver from all the collector instance.
- **TOTAL EVENTS COLLECTED**: The total number of events collected from the data source by the collector instance.
- **MAXIMUM EVENT RATE**: Specifies the maximum number of data samples per minute sent to the archiver from all the collector instance.

## Access the Tags in a Collector Instance

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.
   A list of collectors appears.
3. Right-click the collector instance whose tags you want to access (or select ⋯), and then select **Browse Tags**.




   A list of tags for which the collector instance collects data appears.
4. To narrow down your search results, select **Search**, enter the search criteria, and then enter **Search**.
   You can add more search criteria by selecting **Add Attribute**.
   You can enter a name or a value partially or use the wildcard character asterisk (*).
   The list of tags are filtered based on the search criteria.

## Add a Collector Instance

Before you begin using a collector, you must add an instance of the collector. You can add multiple instances of the same collector or instances of multiple collectors. To add multiple instances of a collector, perform the steps once again.

You can add and configure the following types of collector instances:

- The Calculation collector *(on page 479)*
- The CygNet collector *(on page 482)*
- The File collector *(on page 486)*
- The HAB collector *(on page 489)*
- The iFIX collector *(on page 505)*
- The MQTT collector *(on page 510)*
- The MQTT Sparkplug B collector *(on page 519)*
- The ODBC collector *(on page 527)*
- The OPC Classic Alarms and Events collector *(on page 532)*
- The OPC Classic DA collector *(on page 534)*
- The OPC Classic HDA collector *(on page 540)*
- The OPC UA DA collector *(on page 544)*
- The OSI PI collector *(on page 549)*
- The OSI PI distributor *(on page 553)*
- The Server-to-Server collector *(on page 559)*
- The Server-to-Server distributor *(on page 563)*
- The Simulation collector *(on page 567)*
- The Windows Performance collector *(on page 571)*
- The Wonderware collector *(on page 575)*

## Enable MTLS Security for Collectors

Mutual TLS (MTLS) protocol is a secure and strong authentication mechanism. Using this protocol, you can also encrypt the data between Historian Server and Collectors.

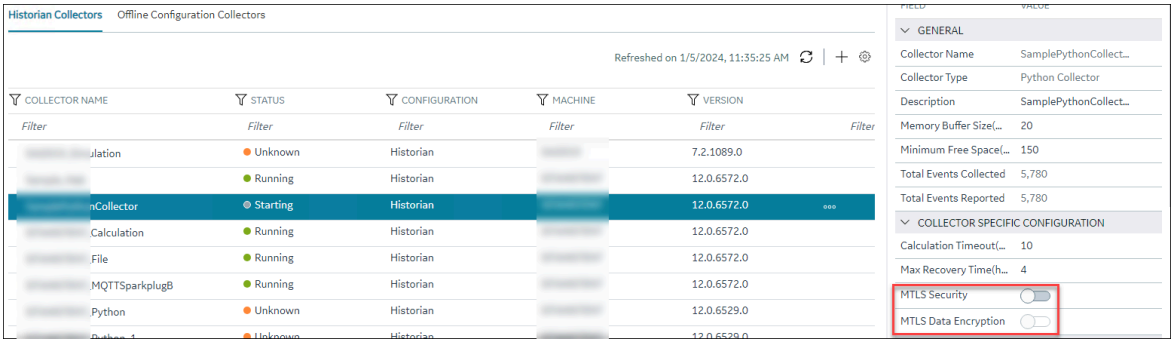
Ensure that you have configured certificate-based security.

This topic describes how to enable MTLS security and encrypt data for collectors.

**Note:**
**MTLS Security** and **MTLS Data Encryption** are not applicable to the File collector.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.
   A list of collectors in the default system appears.
3. Select the row containing the collector for which you want to enable **MTLS Security** and **MTLS Data Encryption**.
   The details of the collector appear in the **DETAILS** section.

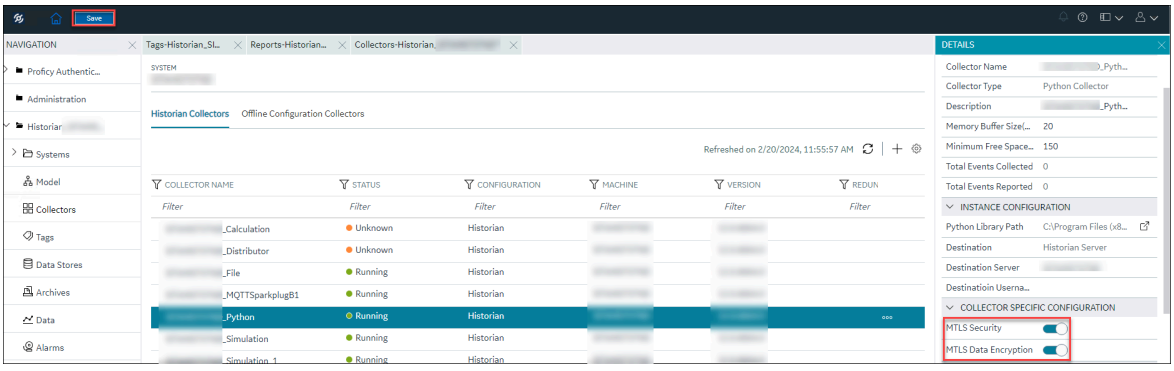
> **Note:**
> If the **DETAILS** section does not appear, in the upper-right corner of the page, select , and then select **Details**.




4. In the **COLLECTOR SPECIFIC CONFIGURATION** section, enable **MTLS Security**.

> **Note:**
> You can enable **MTLS Security** and **MTLS Data Encryption** only if the collector is running.

5. After you enable **MTLS Security**, enable **MTLS Data Encryption**.




6. In the upper-right corner, select **Save**.
   You will be prompted to restart the collector.

7. To apply the changes, select **Restart Now**.

   If you select **Restart Later**, a notification appears in the **DETAILS** section, stating that the selected collector instance needs a restart.

## Modify a Collector Instance

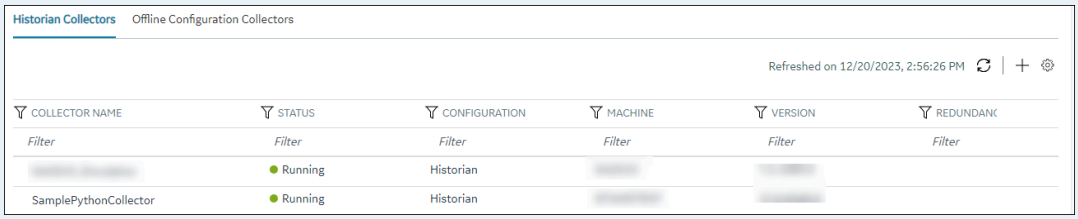
This topic describes how to modify a collector instance using Configuration Hub. You can also modify a collector instance using the RemoteCollectorConfigurator utility *(on page 799)*, which does not require you to install Web-based Clients.

**Note:**

- If the status of a collector instance in unknown, you cannot modify it.
- You cannot modify the instance of an offline collector.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.
   A list of collectors appears.

   **Tip:**
   You can filter the collectors by the system name.

3. Select the collector instance that you want to modify.
   The details of the collector appear in the **DETAILS** section.

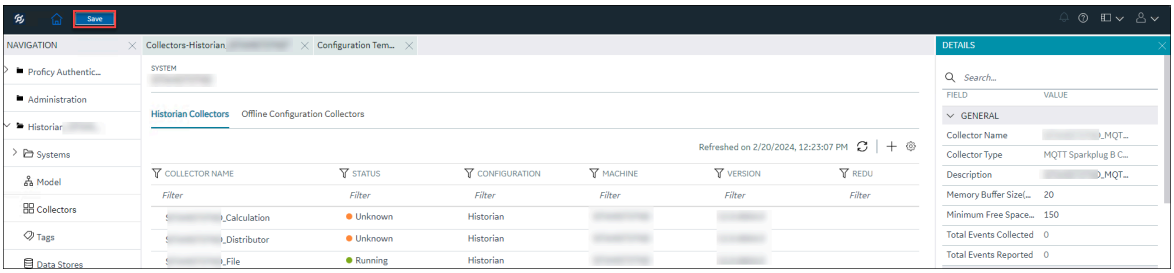
   **Tip:**
   If the **DETAILS** section does not appear, in the upper-right corner of the page, select , and then select **Details**.

4. As needed, modify values in the available fields *(on page 579)*.

**Note:**

- You cannot modify the destination of a collector.
- For collectors earlier than version 9.0:
  - You cannot modify the details in the **INSTANCE CONFIGURATION** section.
  - Some of the details, such as the collector type, do not appear.

5. After you modify the values, in the upper-left corner of the page, select **Save**.




Based on the values that you modified, you might be prompted to restart the collector. For the modifications to take effect, you must restart the collector.

6. Select **Restart Now**.
   The collector instance restarts and the modifications take effect.

   If you select **Restart Later**, a notification appears in the **DETAILS** section, stating that the selected collector instance needs a restart.

## Add a Comment to a Collector Instance

This topic describes how to add a comment to a collector instance.

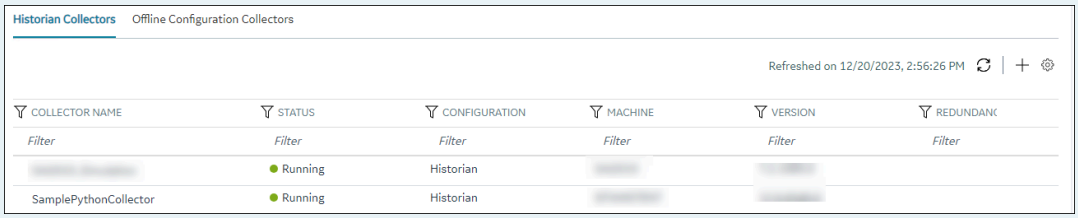
**Note:**

- You cannot modify or delete comments.
- You cannot add comments to offline collectors.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.
   A list of collectors appears.

**Tip:**

You can filter the collectors by the system name.




3. Select the collector instance to which you want to add a comment.
   The details of the collector appear in the **DETAILS** section, along with a list of comments at the end.

**Tip:**

If the **DETAILS** section does not appear, in the upper-right corner of the page, select , and then select **Details**.

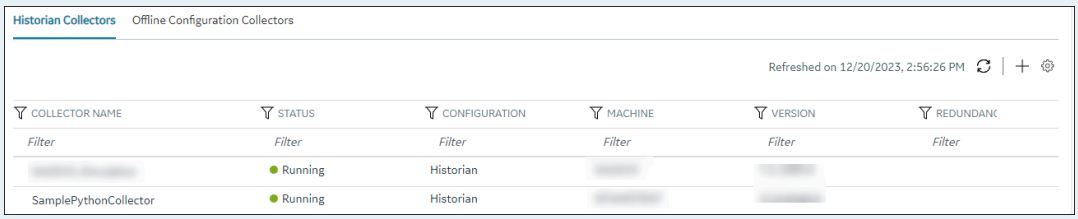
4. In the **DETAILS** section, in the text box below **Comments**, enter your comment, and then select **Add Comment**.
   The comment is added to the collector instance.

## Access a Comment on a Collector Instance

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.
   A list of collectors appears.

**Tip:**

You can filter the collectors by the system name.




3. Select the collector instance whose comments you want to access.

The details of the collector appear in the **DETAILS** section, along with a list of comments at the end.

> **Tip:**
>
> If the **DETAILS** section does not appear, in the upper-right corner of the page, select , and then select **Details**.

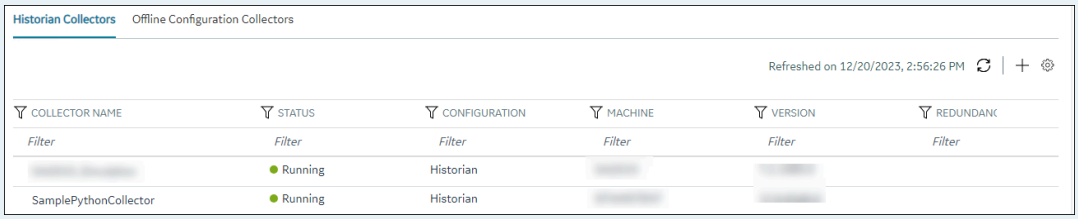
4. To access comments in full screen, select . If you want to search for a comment, enter the search criteria in the **Search** field. You can also filter the comments based on a date and time range by selecting the values in the **FROM** and **TO** fields.
   The comments are filtered based on the search criteria.

## Start a Collector

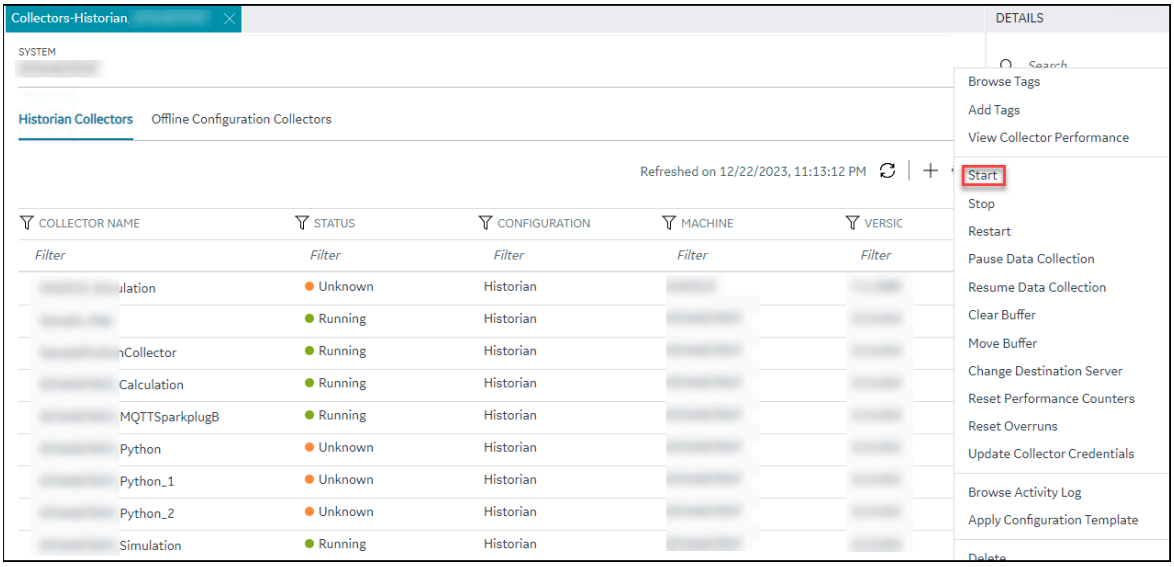
You can start a collector using one of the following options:

- **Service**: Select this option if you want to start the collector as a Windows service using the credentials of the local user (that is, the currently logged-in user). If you select this option, the **USERNAME** and **PASSWORD** fields are disabled.
- **Command Line**: Select this option if you want to start the collector at a command prompt using a specific user account. If you select this option, you must enter values in the **USERNAME** and **PASSWORD** fields.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.
   A list of collectors appears.

> **Tip:**
>
> You can filter the collectors by the system name.
>
> 
> 

3. Right-click the collector instance that you want to start (or select ), and then select **Start**.




The **Start: <collector name>** window appears.

4. Under **RUNNING MODE**, select one of the following options:
   - **Service**: Select this option if you want to start the collector as a Windows service using the credentials of the local user (that is, the currently logged-in user). If you select this option, the **USERNAME** and **PASSWORD** fields are disabled.
   - **Command Line**: Select this option if you want to start the collector at a command prompt using a specific user account. If you select this option, you must enter values in the **USERNAME** and **PASSWORD** fields.
5. Select **Start**.
   The collector is started, and the data collection begins. The status of the collector in the **Collectors** section changes to Starting and then to Running. If, however, the connection fails, the status changes to Unknown.

> **Note:**
> If auto-refresh is not enabled, refresh the collector manually.

## Stop a Collector

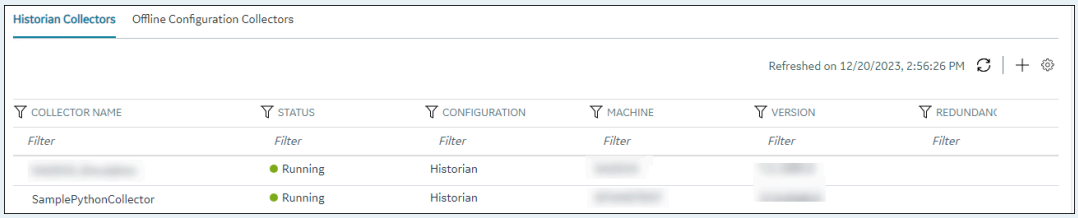
When you stop a collector, the collector stops collecting data, and it is disconnected from the destination. If, however, you want the collector to remain connected to the destination, you can instead pause data collection *(on page 641)*.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.

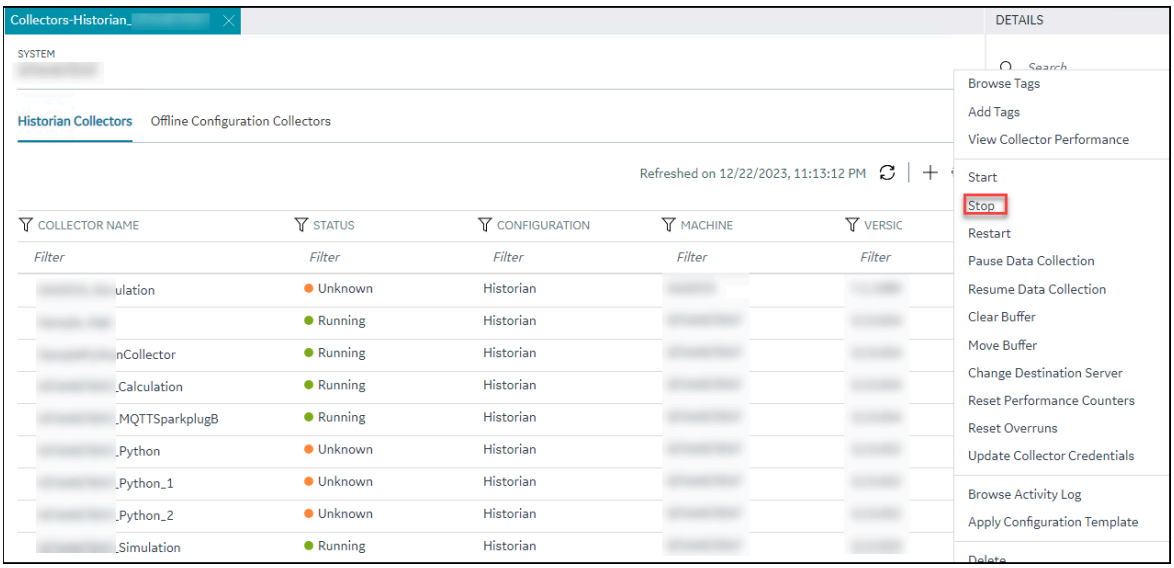
A list of collectors appears.

**Tip:**

You can filter the collectors by the system name.




3. Right-click the collector instance that you want to stop (or select ), and then select **Stop**.




The **Stop: <collector name>** window appears. The **COLLECTOR MACHINE** and **CURRENT RUNNING MODE** fields are populated and disabled.

4. If the collector is running in the command-line mode, enter values in the **USERNAME** and **PASSWORD** fields.
5. Select **Stop**.

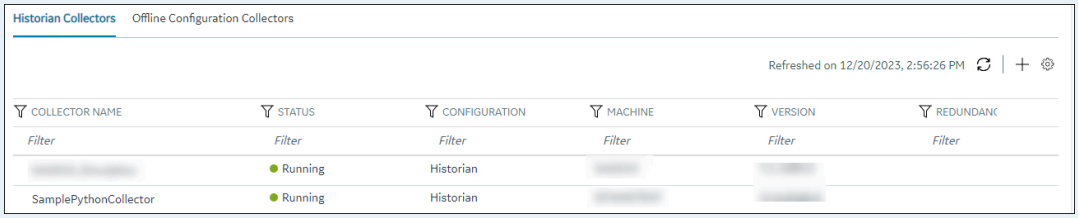
   The collector is stopped, and the data collection is paused. The status of the collector in the **Collectors** section changes to Stopped.

## Restart a Collector

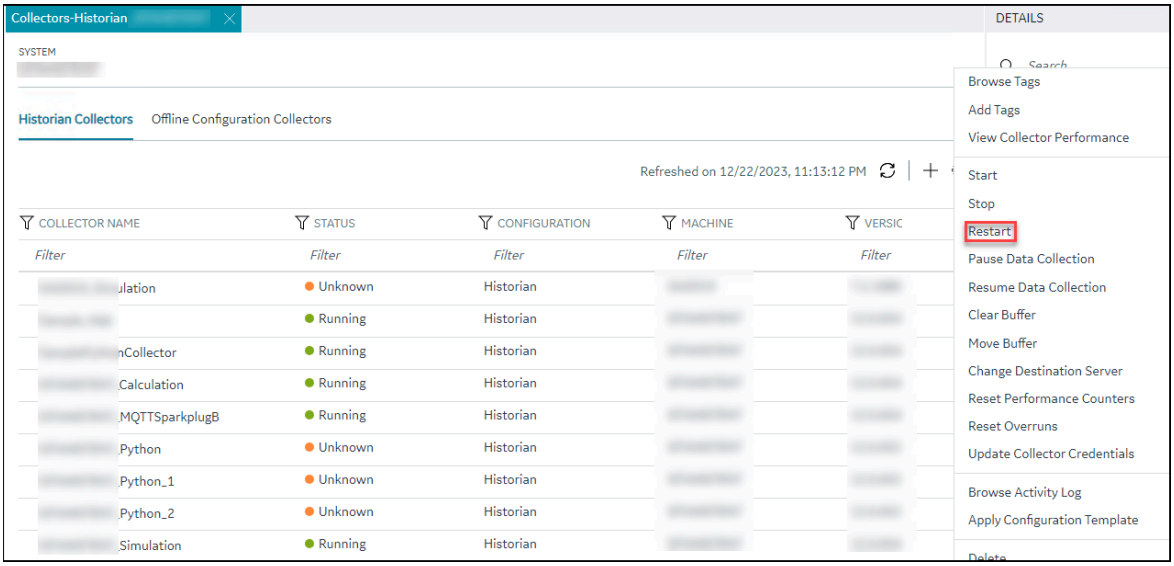
You can restart a collector to stop and start it again. You can restart a collector only if it is running. However, when you modify a collector instance, based on the values you modify, you will be prompted

and given the option to restart the collector before you save the modifications. Without restarting, the modifications will not take effect.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.
   A list of collectors appears.

> **Tip:**
> You can filter the collectors by the system name.
>
> 
> 

3. Right-click the collector instance that you want to restart (or select ⋯), and then select **Restart**.




   The **Restart: <collector name>** window appears. The **COLLECTOR MACHINE** and **CURRENT RUNNING MODE** fields are populated and disabled.
4. If the collector is running in the command-line mode, enter values in the **USERNAME** and **PASSWORD** fields.
5. Select **Restart**.
   The collector is restarted, and the data collection is resumed.

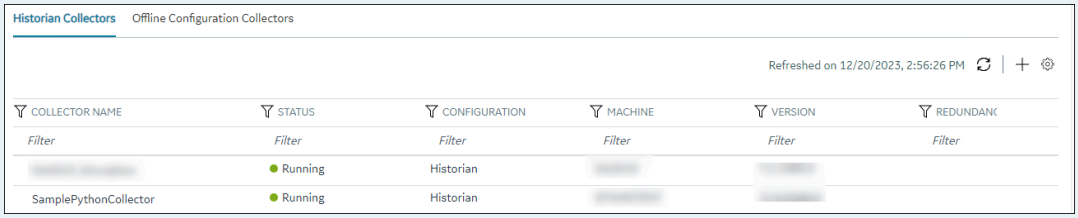
# Pause the Data Collection of a Collector

When you pause data collection, the collector stops collecting the data. However, the collector is still connected to the destination. If you want to disconnect the collector from the destination, stop the collector *(on page 638)*.

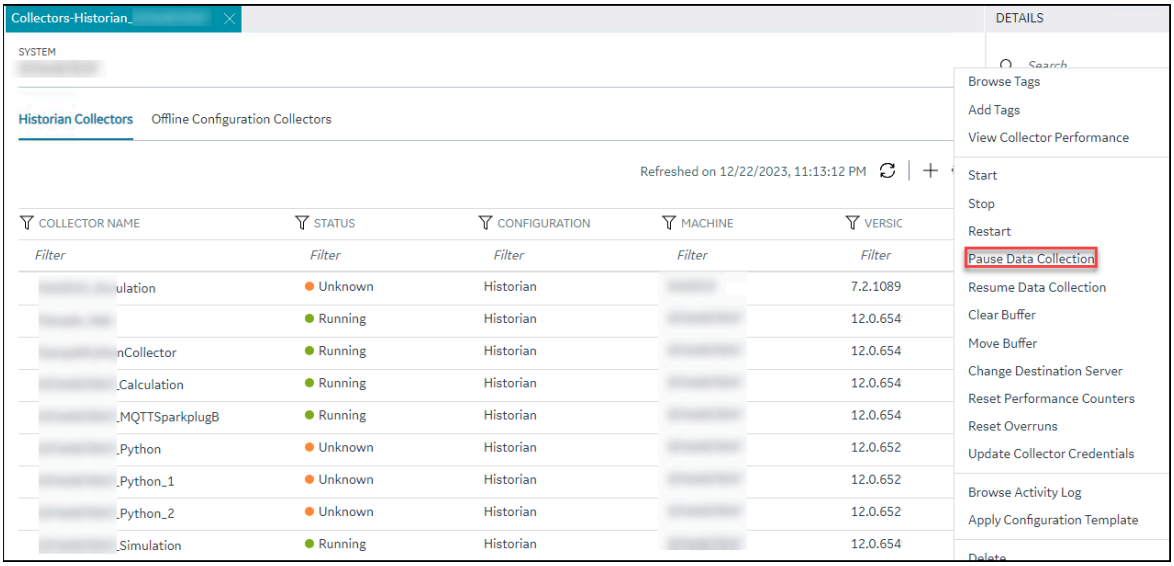
**Note:**
You cannot pause the data collection of an offline collector.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**. A list of collectors appears.

**Tip:**
You can filter the collectors by the system name.




3. Right-click the collector instance for which you want to pause data collection (or select ⋯), and then select **Pause Data Collection**.



A message appears, asking you to confirm whether you want to pause data collection.

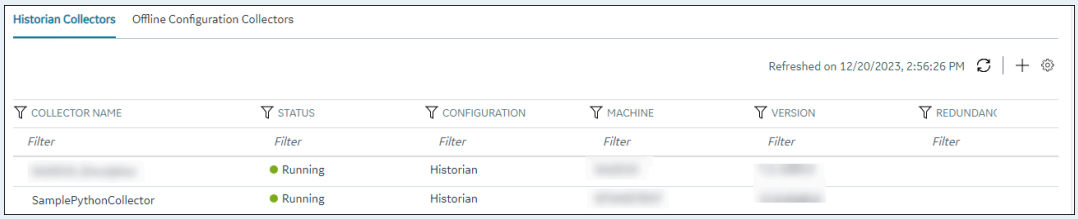
4. Select **Pause**.

   The data collection is paused, and the collector is stopped.

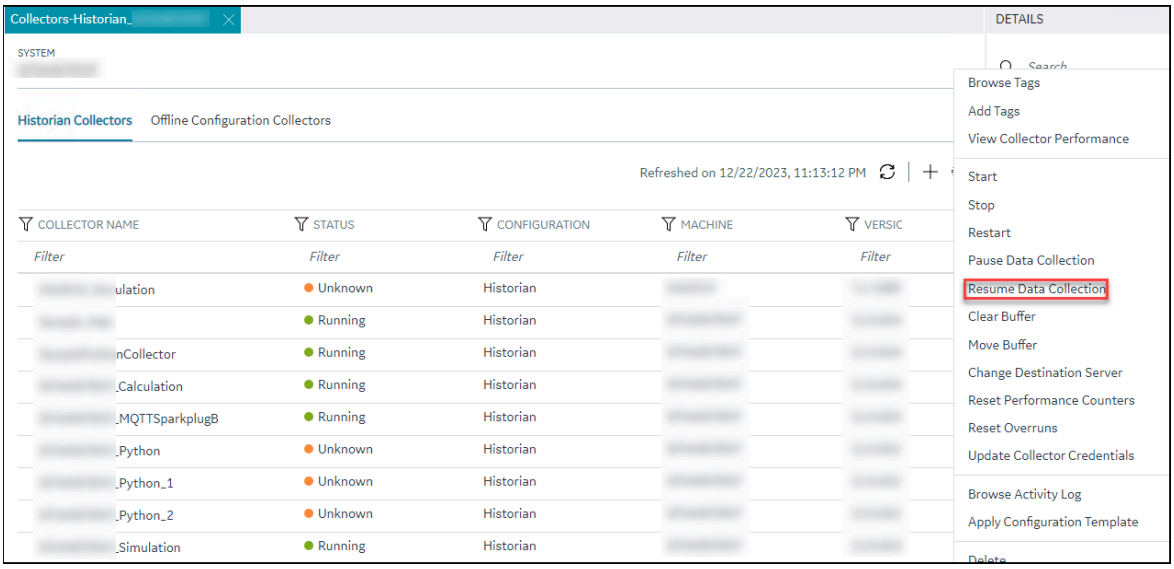
## Resume the Data Collection of a Collector

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.

   A list of collectors appears.

> **Tip:**
>
> You can filter the collectors by the system name.
>
> 
> 

3. Right-click the collector instance for which you want to resume data collection (or select ⋯), and then select **Resume Data Collection**.




A message appears, asking you to confirm whether you want to resume data collection.

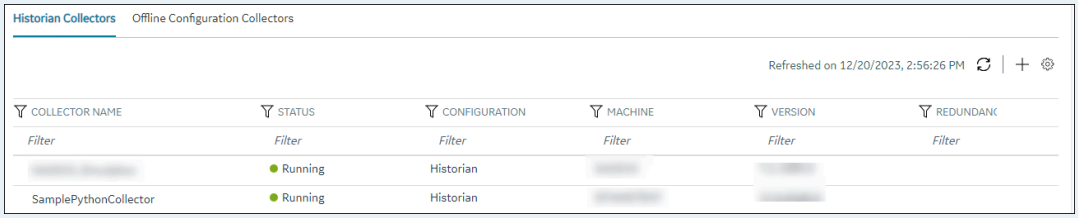
4. Select **Resume**.

   The collector is started, and the data collection is resumed.

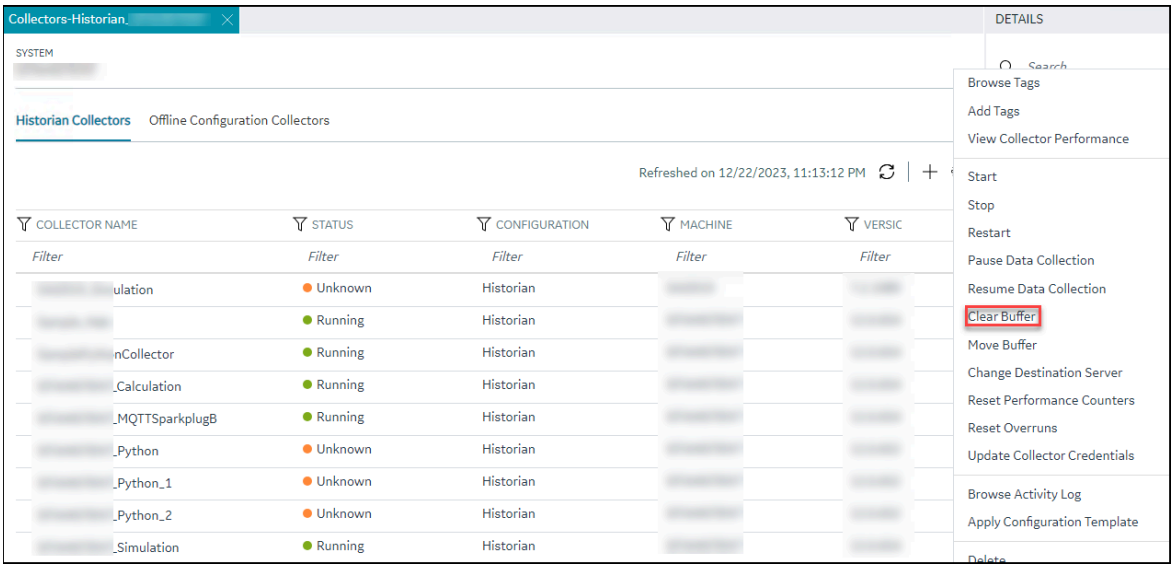
# Delete the Buffer Files of a Collector

When you delete buffer files, the collector is stopped, and after the buffer files are deleted, it is restarted.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.
   A list of collectors appears.

> **Tip:**
> You can filter the collectors by the system name.
>
> 
> 

3. Right-click the collector instance whose buffer files you want to delete (or select ), and then select **Clear Buffer**.




   A message appears, asking you to confirm that you want to clear the buffer files.
4. Select **Clear**.
   The **Clear Buffer: <collector name>** window appears.
5. If the collector is running in the command-line mode with a specific user account, enter values in the **USERNAME** and **PASSWORD** fields.

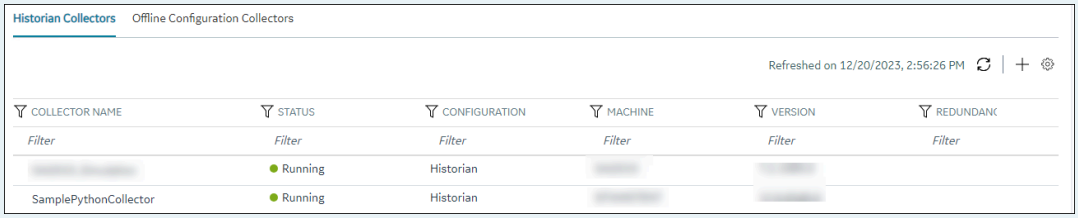
6. Select **Clear**.
   The buffer files of the collector are deleted.

## Move the Buffer Files of the Collector

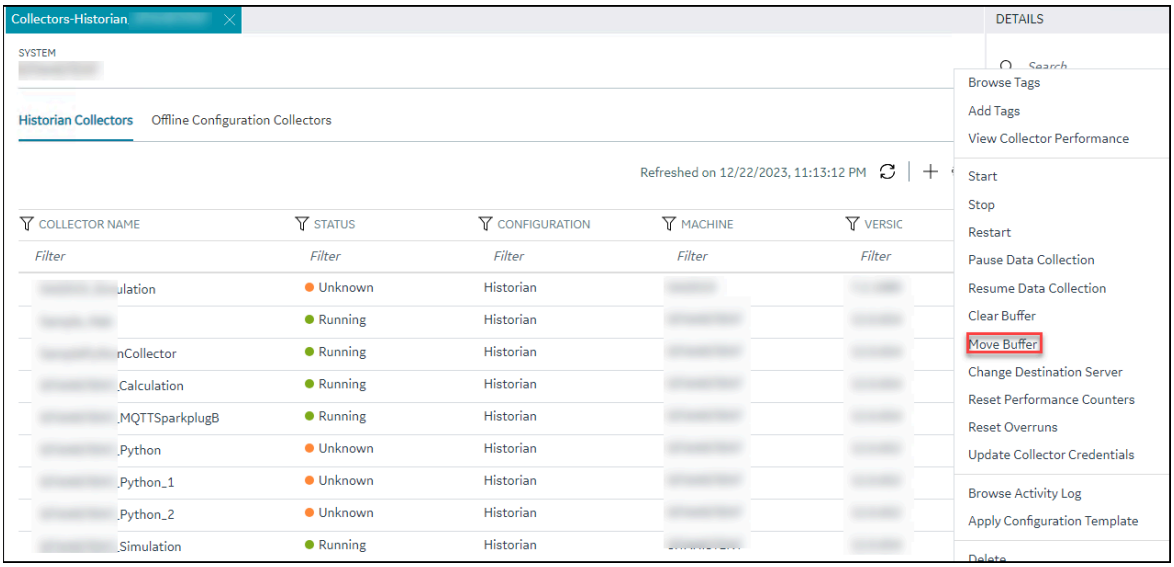
We recommend that you move the buffer files to a new folder within the same drive. You cannot move files to a folder on a network shared drive.

When you move buffer files, the collector is stopped, and after the buffer files are moved, it is restarted.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.
   A list of collectors appears.

> **Tip:**
> You can filter the collectors by the system name.
>
> 
> 

3. Right-click the collector instance whose buffer files you want to move (or select ⋯), and then select **Move Buffer**.



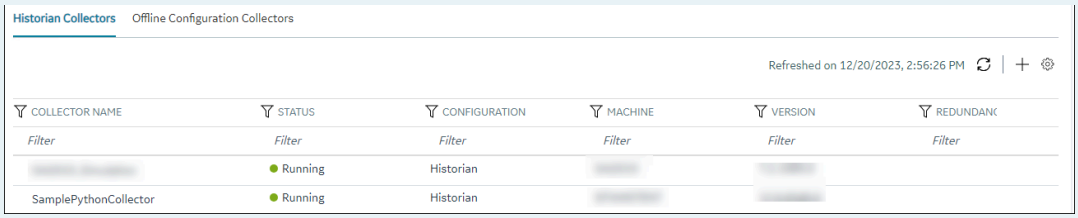
The **Move Buffer: <collector name>** window appears. The **CURRENT LOCATION**, **COLLECTOR MACHINE**, and **RUNNING MODE** fields are populated and disabled.

4. In the **TARGET LOCATION** field, enter the path of the folder to which you want to move the buffer files.
5. If the collector is running in the Windows service mode, select **Move Buffer**. If the collector is running in the command-line mode, enter values in the **USERNAME** and **PASSWORD** fields, and then select **Move Buffer**.
   The buffer files are moved, and the collector is started.

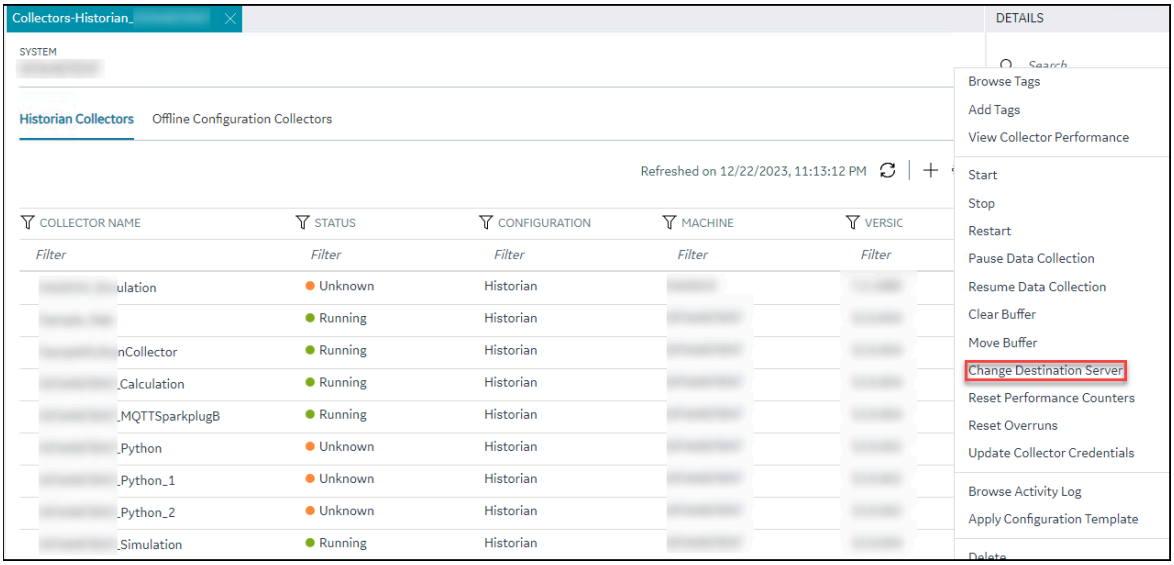
## Change the Destination Server of a Collector

1. Ensure that Historian is installed on the new destination server to which you want the collector to send data.
2. Ensure that the collector whose destination server you want to change is running.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.
   A list of collectors appears.

> **Tip:**
> You can filter the collectors by the system name.
>
> 
> 

3. Right-click the collector instance whose destination server you want to change (or select ⋯), and then select **Change Destination Server**.




The **Change Destination Server: <collector name>** window appears. The **COLLECTOR MACHINE**, **CURRENT RUNNING MODE**, and **CURRENT DESTINATION SERVER** fields are populated and disabled.

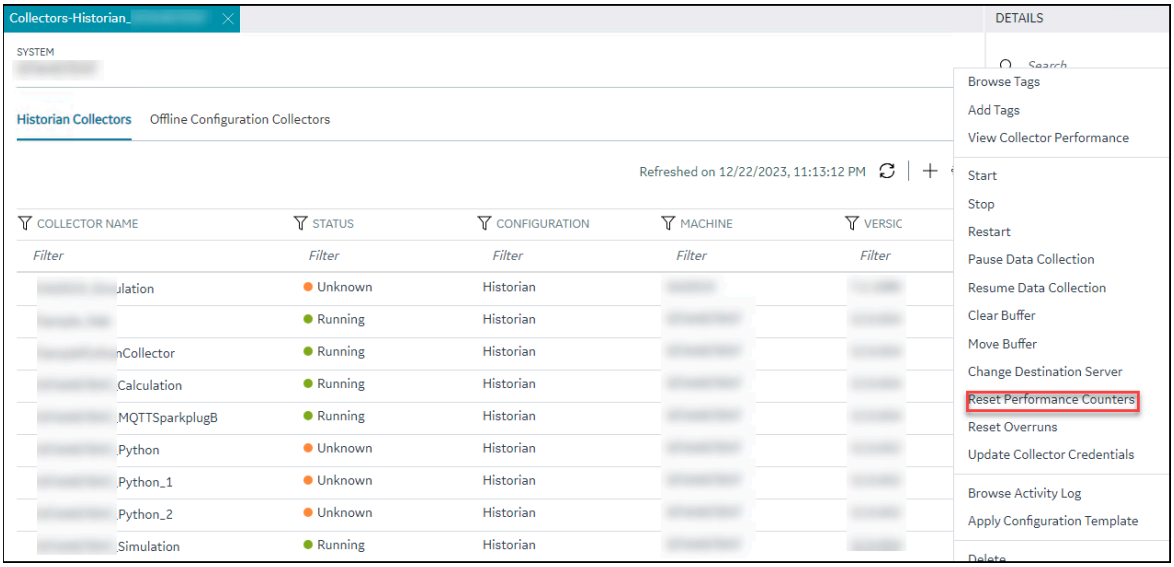
4. In the **NEW RUNNING MODE** field, select one of the following options:
   - **Service**: Select this option if you want to start the collector as a Windows service using the credentials of the local user (that is, the currently logged-in user). If you select this option, the **USERNAME** and **PASSWORD** fields are disabled.
   - **Command Line**: Select this option if you want to start the collector in the command-line mode. If you select this option, you must enter values in the **USERNAME** and **PASSWORD** fields.
5. In the **NEW DESTINATION SERVER** field, enter the host name or IP address of the new destination server to which you want the collector to send data.
6. In the **USERNAME** and **PASSWORD** fields, enter the credentials to access the new destination server.
7. Select **Change Server**.
   The destination server of the collector is changed, and the collector is stopped.

1. Update the network message compression of the collector by modifying the collector instance using Configuration Hub.
2. Reconfigure the collector properties using Historian Administrator.
3. Restart the collector *(on page 639)*.

## Reset Performance Counters

This topic describes how to reset performance counters for a collector. Performance counters are used to monitor the performance of Historian components. You can reset it to zero if you want to reset the performance counters.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.
   A list of collectors appears, displaying the details.
3. Select the row containing the collector whose details you want to access.
4. Right-click the collector (or select ), and then select **Reset Performance Counters**.



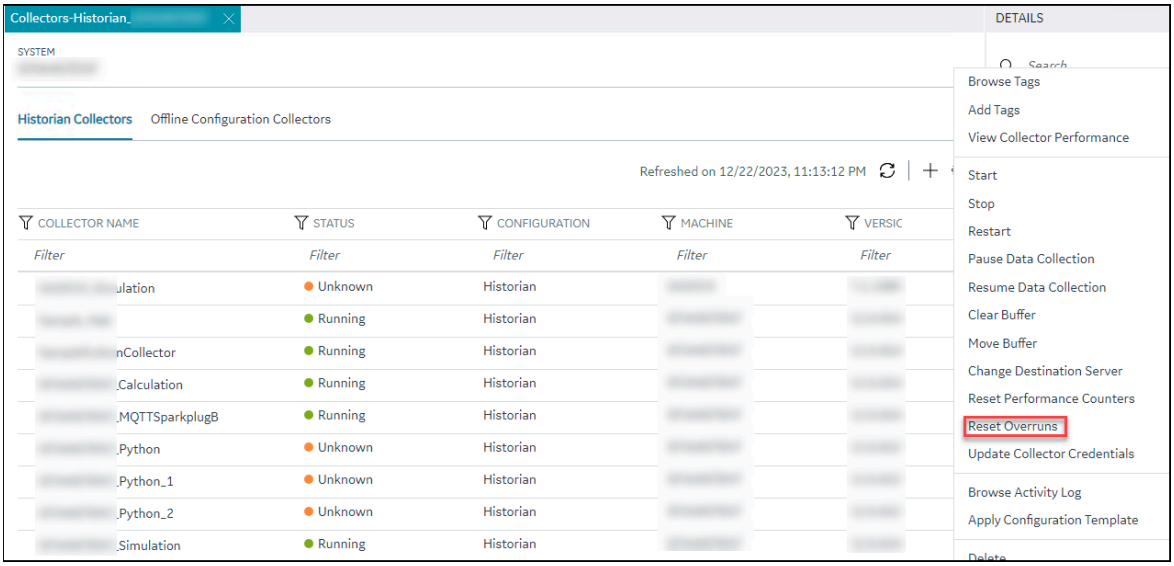

   A confirmation window appears, prompting you whether or not to reset the performance counters to zero.
5. Select **Reset**.
   All the performance counters are reset, including overruns.

## Reset Overruns

This topic describes how to reset overruns count. Overruns are a count of the total number of data events not collected on their scheduled polling cycle. In normal operation, this value should be zero, if not, you can reset it to zero.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.
   A list of collectors appears, displaying the details.
3. Select the row containing the collector whose details you want to access.

4. Right-click the collector (or select ⋯), and then select **Reset Overruns**.

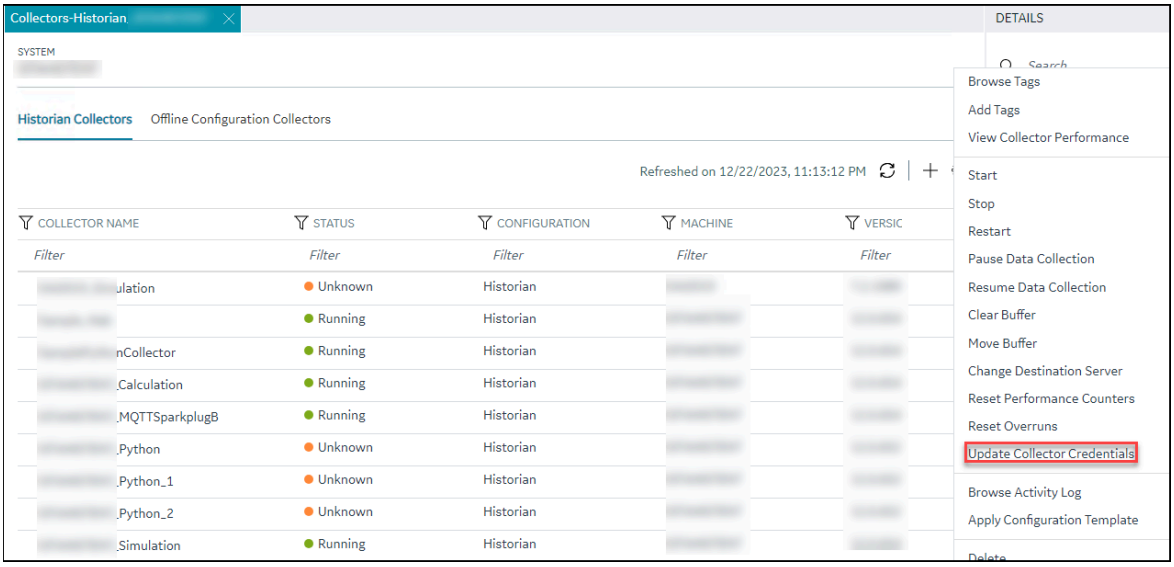



   A confirmation window appears, prompting you whether or not to reset the performance counter to zero.
5. Select **Reset**.
   The Overruns are reset.

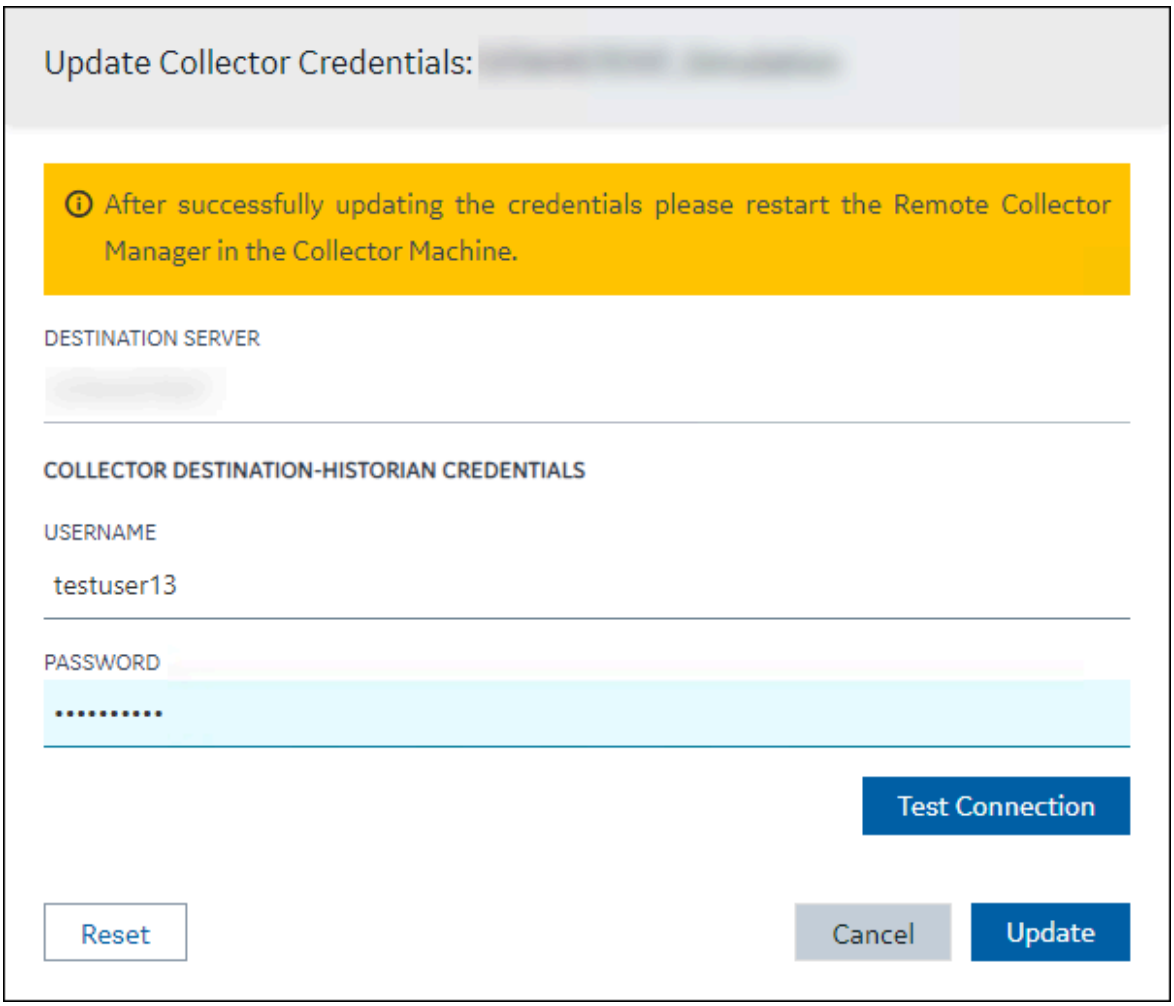
## Update Collector Credentials

Whenever there are some changes in the credentials of the destination Historian server, you can update the credentials for the collectors using that server. This topic describes how to update the collector credentials.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.
   A list of collectors appears, displaying the details.
3. Select the row containing the collector whose details you want to access.
4. Right-click the collector (or select ⋯), and then select **Update Collector Credentials**.




The **Update Collector Credentials: <Collector name>** window appears.



5. Update the **USERNAME** and **PASSWORD**, and select **Test Connection**.
   Testing connection will help you to validate if the destination server credentials that you entered are valid or not.
6. Select **Update**.
   The updated credentials are saved.
7. Restart the Remote Collector Manager in the collector machine.

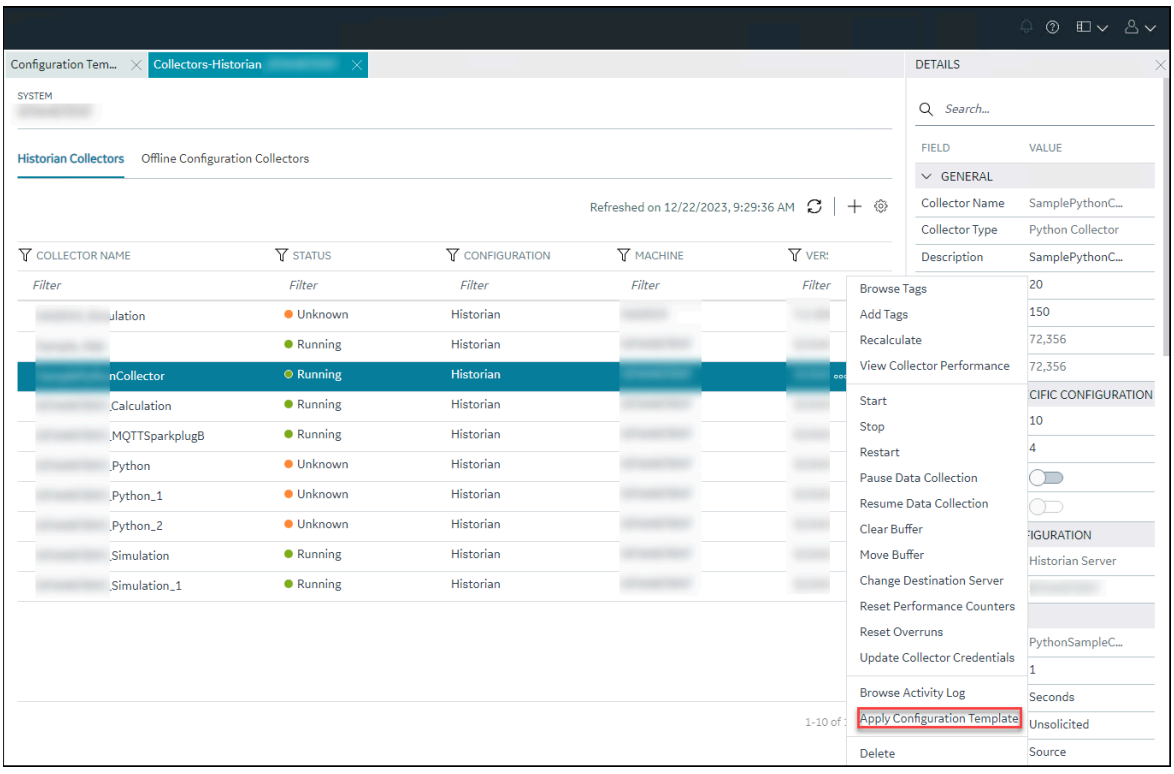
## Apply Configuration Template to a Collector

You can apply the created template to collectors as needed. You will be prompted to confirm whether you want to overwrite few of the configuration values with the values in the template.

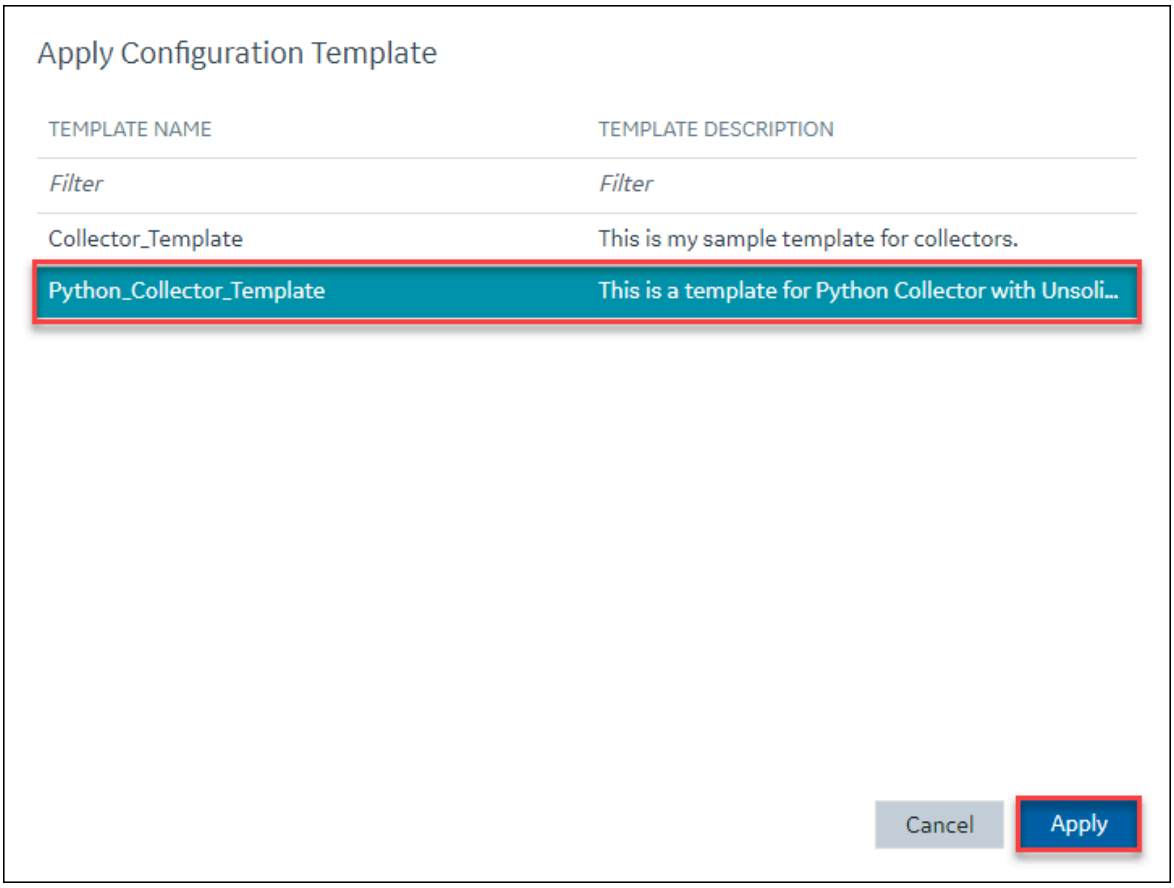
- Ensure that you have a collector instance added *(on page 357)*.
- Ensure that you created a configuration template for collectors *(on page 757)*.

This topic describes how to apply a configuration template to a collector.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.
   A list of collectors appears.
3. Right-click the collector (or select ⋯), and then select **Apply Configuration Template**.



The **Apply Configuration Template** window appears, listing the available templates.




4. Select **Apply**.
   A confirmation window appears, prompting you to confirm whether you want to overwrite few of the configuration values with the values in the template.
5. Select **Ok**.
6. In the upper-left corner, select **Save**.
   The configurations in the template are applied to the collector.

## Configure Collector Redundancy

1. Create the collector instances *(on page 357)* that you want to use for collector redundancy. All these collectors must be of the same type.

> **Note:**
> Collector redundancy is not available and supported for the following collectors:
> - File Collector.
> - Calculation Collector.

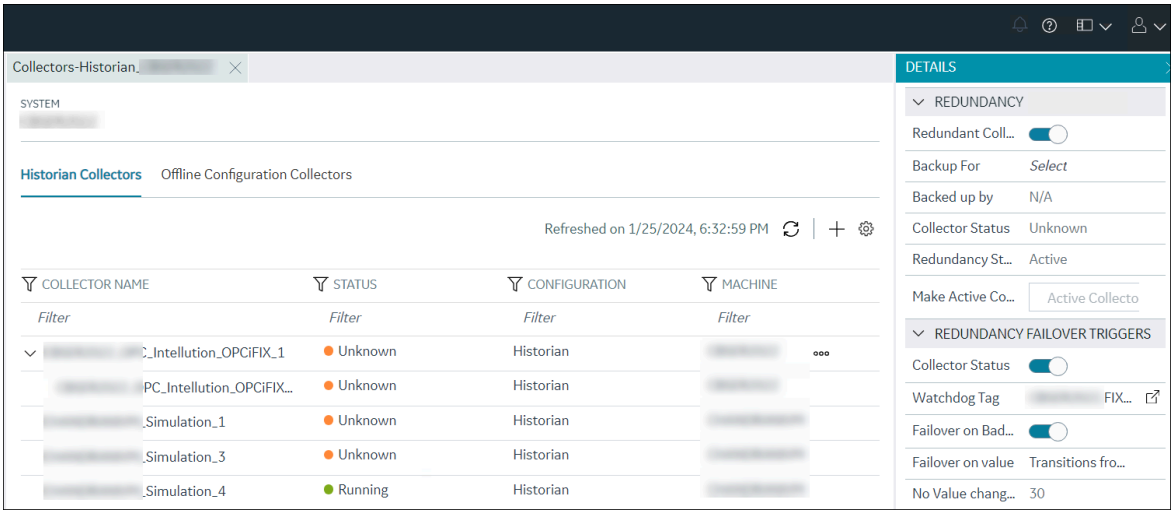
- Python Collector.
- Server-to-Server Collector.
- Server-to-Server Distributor.
- OSI PI Collector.
- OSI PI Distributor.

2. Create tags *(on page 909)* in the primary collector.
3. If you want to use the values of a watchdog tag as a failover trigger, create the watchdog tag *(on page 909)* in the primary collector.

Collector redundancy ensures that collection of your data remains uninterrupted. It uses two or more collectors that gather data from a single source. For more information, refer to About Collector Redundancy *(on page 975)*.

This topic describes how to set up redundancy between two collectors - one primary and the other secondary. You can, however, set up multiple secondary collectors.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.
   A list of collectors appears.




3. For the primary collector, in the **DETAILS** section, under **REDUNDANCY**, switch the **Redundant Collector** toggle. And then, in the upper-left corner of the page, select **Save**.
   Redundancy is enabled for the primary collector.
4. Repeat the previous step for the secondary collector.
   Redundancy is enabled for the secondary collector.

5. For the secondary collector, in the **DETAILS** section, under **REDUNDANCY**, in the **Backup For** field, select the primary collector, and then select **Save**.
   The secondary collector is configured as a backup for the primary collector.

   > **Note:**
   > If you want to add another collector in the redundancy group, in the **Backup For** field, select the secondary collector.

6. If you want to manually trigger a failover, perform the following steps. Or, if you want to configure automatic triggers, skip to the next step.
   a. Select the collector that you want to make active.
   b. In the **DETAILS** section, under **REDUNDANCY**, in the **Make Active Collector** field, select **Active Collector**.
7. For the secondary collector, in the **DETAILS** section, under **REDUNDANCY FAILOVER TRIGGERS**, enter values as described in the following table, and then select **Save**.

| Field | Description |
|---|---|
| **Collector Status** | Switch the toggle if you want to trigger failover when the primary collector stops collecting data. |
| **Watchdog Tag** | Select , and then select the tag whose values you want to use as a trigger.<br>**Note:**<br>The watchdog tag must be in the primary collector. |
| **Failover on Bad Quality** | Switch the toggle if you want to trigger failover when bad data is collected for the watchdog tag. |
| **Failover on value** | Specify whether you want to trigger a failover if the value of the watchdog tag is non-zero. You can use this option only if you have not used the **No Value change interval** field. |
| **No Value change interval** | Enter the duration in seconds after which you want to trigger a failover if the values of the watchdog tag do not change for the |

| Field | Description |
|---|---|
| | specified duration. You can use this option only if you have not used the **Failover on value** field. |

The failover triggers are configured. As soon as any of these conditions are satisfied, the secondary collector becomes active and collects data.

## Delete a Collector Instance

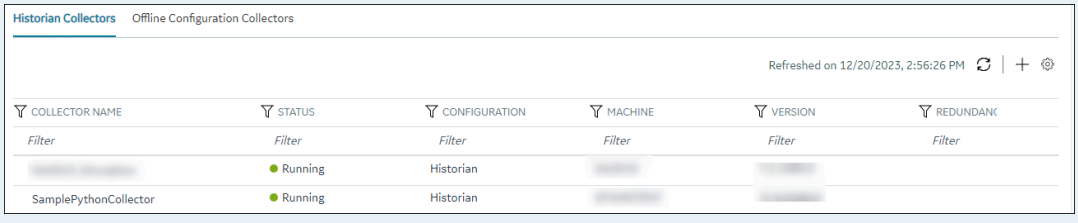
If you no longer want to use a collector instance to collect data, you can delete it. When you delete a collector instance, the Windows service for the collector, the Registry folder, and the buffer files are deleted as well.

This topic describes how to delete a collector instance using Configuration Hub. You can also delete a collector instance using the RemoteCollectorConfigurator utility *(on page 817)*, which does not require you to install Web-based Clients.

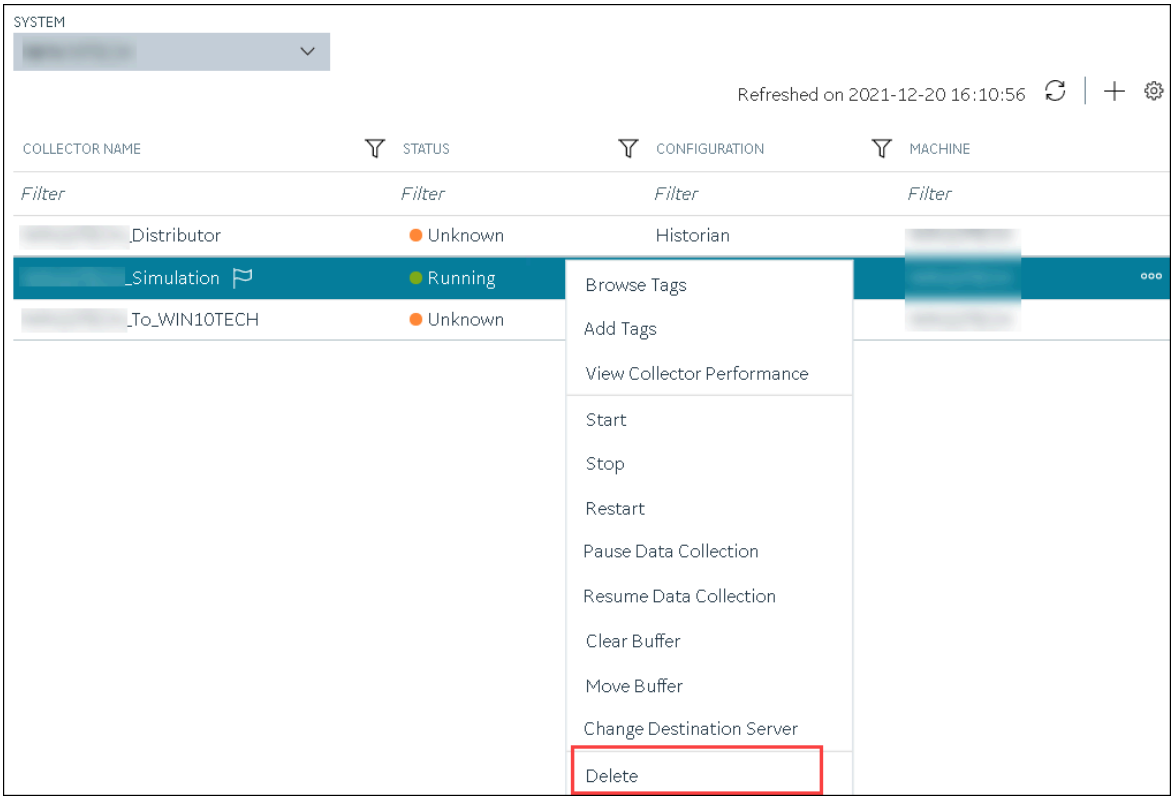
**Note:**
When you delete an offline collector instance, the corresponding configuration file is not deleted. However, if another collector instance of the same interface name is created, the existing configuration file is replaced by a template configuration file.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**. A list of collectors appears.

**Tip:**
You can filter the collectors by the system name.




3. Right-click the collector instance that you want to delete (or select ), and then select **Delete**.




A message appears, asking you to confirm that you want to delete the collector instance.

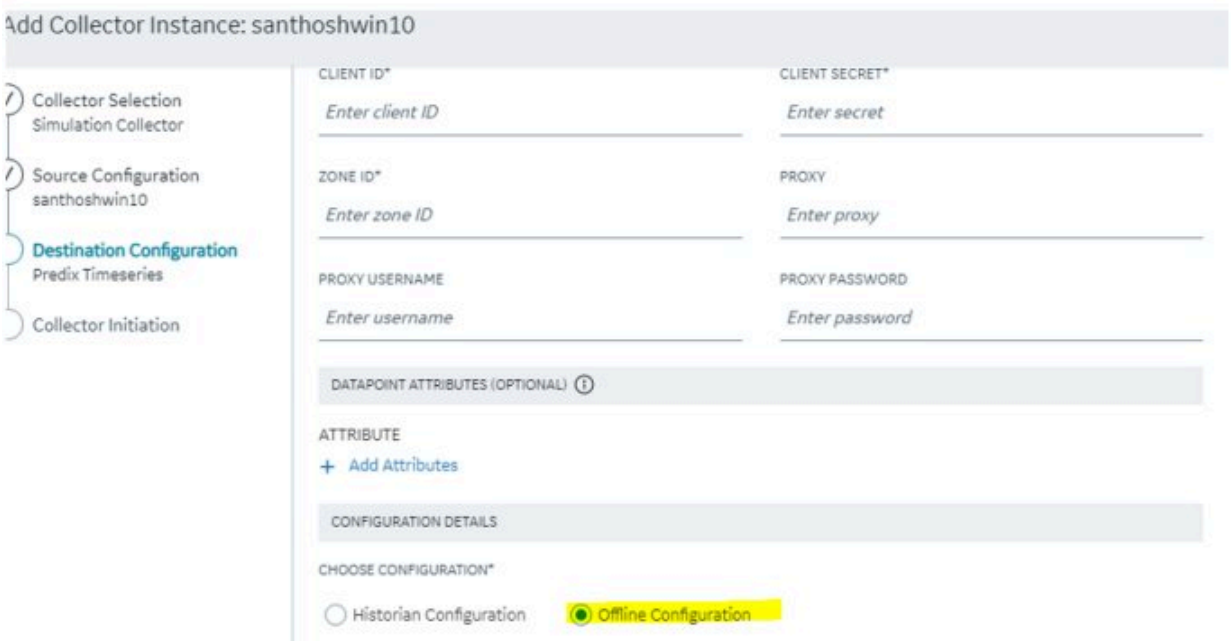
4. If you want to delete the tags as well, select the **Delete associated tags** check box.
5. Select **Delete**.

   The collector instance is deleted.

# Managing Offline Configuration Collector Instances

## Access Offline Configuration Collectors

Offline Configuration Collectors are the instances of collectors whose destination is the cloud and display the configuration details as Offline Configuration.

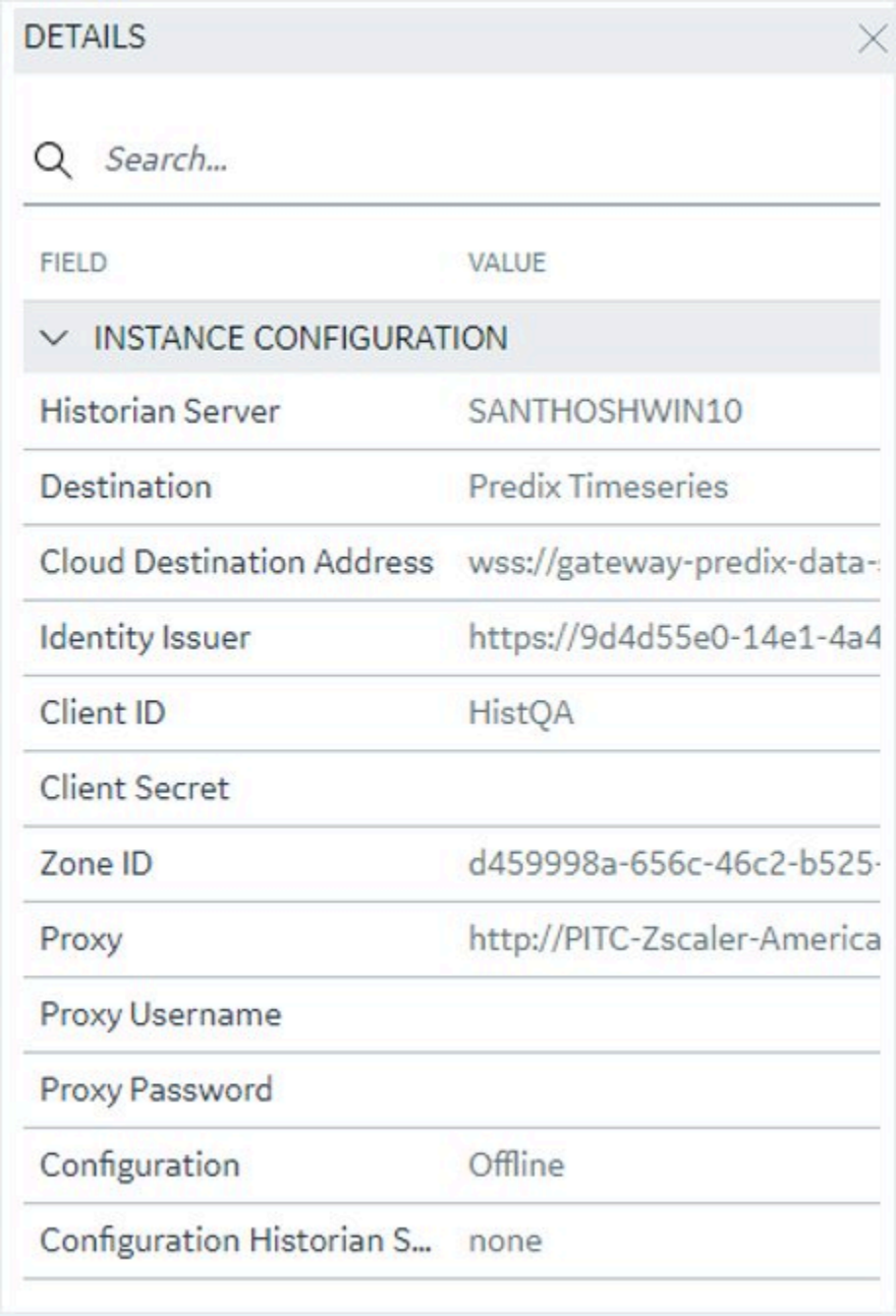



1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, expand **Systems**, and then select the system whose collectors you want to access.
   The system appears in the main section.
3. Right-click the system whose collectors you want to access (or select ⋯), and then select **Browse Collectors**.
   This displays the list of Historian collectors and Offline configuration collectors. By default, the Historian collector instances added to the system appear, displaying the following information.

   **Tip:**
   - To access the details of a collector, select the row containing the collector instance. The details appear in the **DETAILS** section.
   - You can show/hide/reorder columns in the table. For instructions, refer to Common Tasks in Configuration Hub *(on page 353)*.

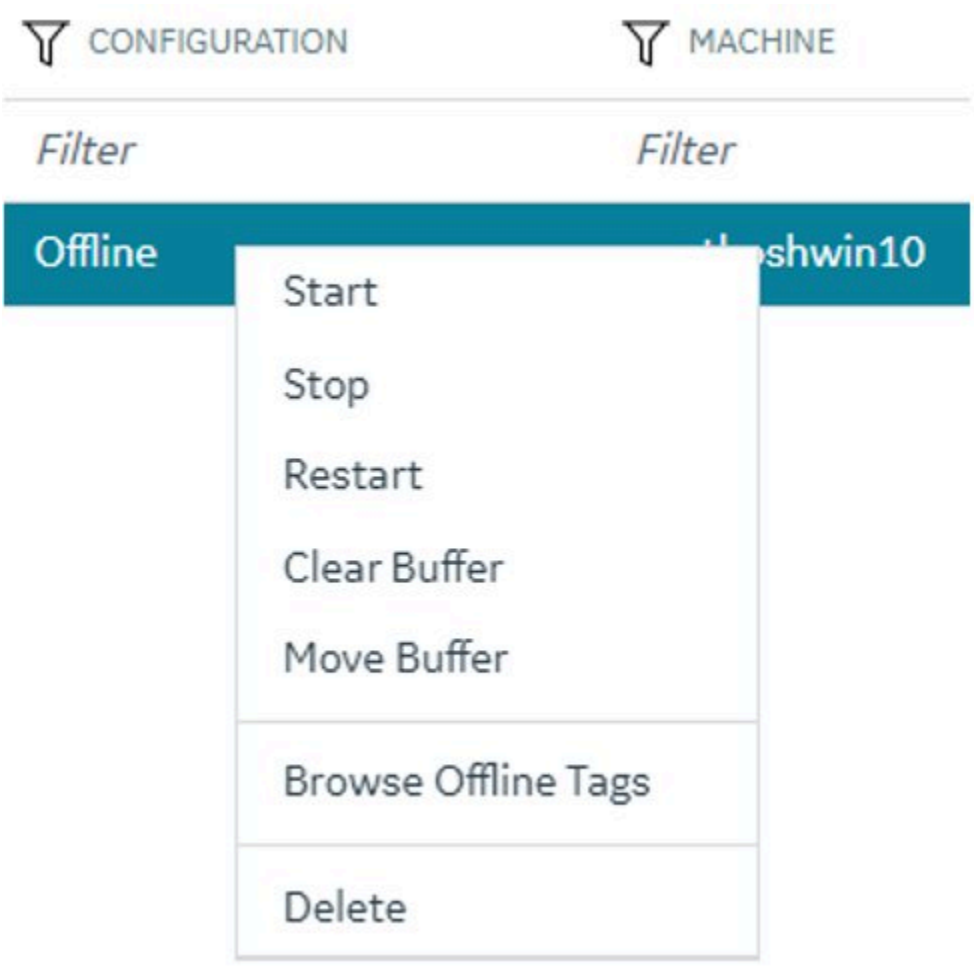
   The Details panel for the Offline Configuration Collector Interface contains information regarding the instance configuration provided for the destination while installing software

like Predix, MQTT, or Azure. For example, here is an example of the Details panel for the Offline Configuration Collector:




## Manage Offline Configuration Collectors

Like other Historian collectors, Offline Configuration collectors can also be managed by selecting different options:



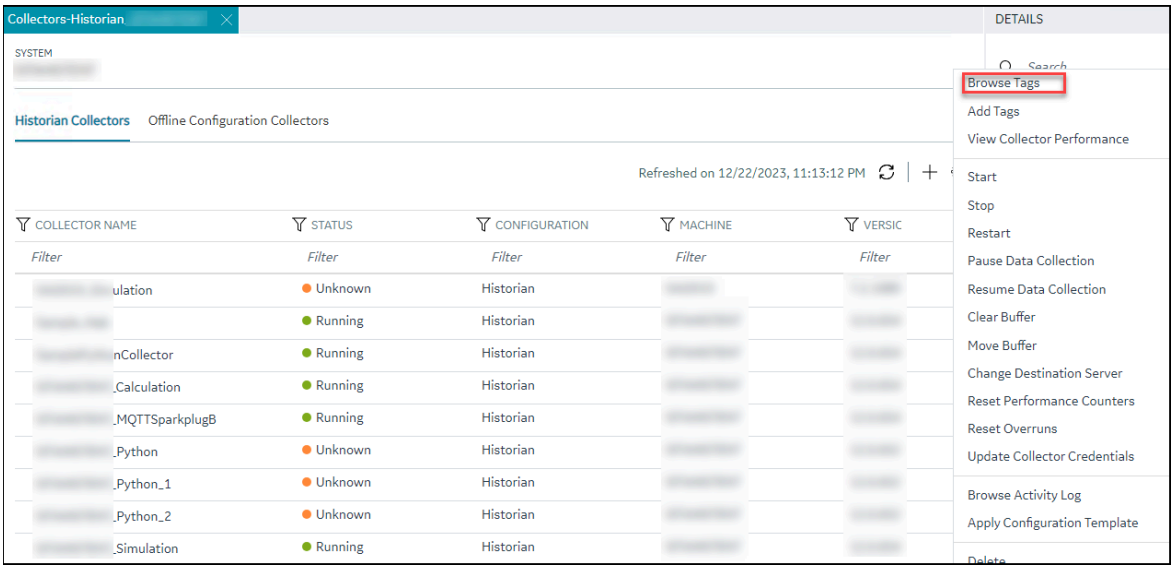

Refer to the following sections on how to use these options:

- Start a Collector *(on page 637)*
- Stop a Collector *(on page 638)*
- Restart a Collector *(on page 639)*
- Delete the Buffer Files of a Collector *(on page 643)*
- Move the Buffer Files of the Collector *(on page 644)*
- Delete a Collector Instance *(on page 654)*

## Access the Tags in the Offline Configuration Collector Instance

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**. A list of collectors appears.

3. Right-click the Offline Configuration collectors you want to access (or select ⋯), and then select **Browse Tags**.




A list of tags for which the Offline Configuration collector instance collects data appears.

4. To narrow down your search results, select **Search**, enter the search criteria, and then enter **Search**.
   You can add more search criteria by selecting **Add Attribute**.
   You can enter a name or a value partially or use the wildcard character asterisk (*).
   The list of tags are filtered based on the search criteria.

# Managing Tags

## About Tags

A Historian tag is used to store data related to a property.

For example, if you want to store the pressure, temperature, and other operating conditions of a boiler, a tag will be created for each one in Historian.

When you collect data using a collector, tags are created automatically in Historian to store these values. These tags are mapped with the corresponding properties in the source.

For example, suppose you want to store OSI PI data in Historian. You will specify the OSI PI tags for which you want to collect data. The OSI PI collector creates the corresponding tags in Historian, and it stores the values in those tags.

You can also choose to create tags manually (for example, to store the result of a calculation performed by the Calculation collector).

## About Array Tags

You can store a set of values with a single timestamp and single quality and then read the elements individually or as an array.

The following conditions apply when using an array tag:

- You need not specify the size of an array tag. Data Archiver will store the number of elements that were written.
- You can change a tag to an array tag later as well. However, when you do so, only the latest data is retrieved. If you want to get the old data, you must change the tag back to its previous type.
- The maximum number of elements that an array tag can store is 10,000.
- You cannot associate an enumerated set or a user-defined data type (UDT) with an array tag.
- Fixed String and Scaled data types are not supported.
- Scaling, collector compression, and archive compression do not apply to an array tag.
- You cannot use an array element as a calculation trigger.
- You cannot plot a trend chart for an array tag.
- TagStats calculation mode is not supported.

## About Collector and Archive Compression

### Collector Compression

Collector compression applies a smoothing filter to data retrieved from the data source. By ignoring small changes in values that fall within a deadband centered around the last reported value, only significant changes are reported to the archiver. Fewer samples reported yields less work for the archiver and less archive storage space used.

You can specify the deadband value. For convenience, if you enter a deadband percentage, Historian Administrator shows the deadband in engineering units. For example, if you specify a 20% deadband on 0 to 500 EGU span, it is calculated and shown as 100 engineering units. If you later change the limits to 100 and 200, the 20% deadband is now calculated as 20 engineering units.

The deadband is centered around the last reported sample, not simply added to it or subtracted. If your intent is to have a deadband of 1 unit between reported samples, you must enter a compression deadband of 2 so that it is one to each side of the last reported sample. In the previous example of 0 to 500 EGU range, with a deadband of 20%, the deadband is 100 units; This means that only if the value changes by more than 50 units, it is reported.

Changes in data quality from good to bad, or bad to good, automatically exceed collector compression and are reported to the archiver. Any data that comes to the collector out of time order will also automatically exceed collector compression.

It is possible for collected tags with no compression to appear in Historian as if the collector or archive compression options are enabled. If collector compression occurs, you will notice an increase in the percentage of the compression value in the **Collectors** section of the **System Statistics** page in Historian Administrator. When archive compression occurs, you will notice the archive compression value and status bar change on the **System Statistics** page.

For instructions on setting collector compression, refer to Access/Modify a Tag *(on page 894)*.

Even if collector compression is not enabled, you may notice it in the following scenarios:

- When a succession of bad data quality samples appears, Historian collects only the first sample in the series. No new samples are collected until the data quality changes. Historian does not collect the redundant bad data quality samples, and this is reflected in the collector compression percentage.
- For a Calculation or Server-to-Server collector, when calculations fail, producing no results or bad quality data, collector compression is used. The effect of Collector Compression Timeout is to behave, for one poll cycle, as if the collector compression feature is not being used. The sample collected from the data source is sent to the archiver. Then the compression is turned back on, as configured, for the next poll cycle with new samples being compared to the value sent to the archiver.

**Note:**
Array tags do not support archive and collector compression. If the tag is an array tag, then the **Compression** tab is disabled.

### Handling Value Step Changes with Collector Data Compression

If you enable collector compression, the collector does not send values to the archiver any new input values if the value remains within its compression deadband. Occasionally, after several sample intervals inside the deadband, an input makes a rapid step change in value during a single sample interval. Since there have been no new data points recorded for several intervals, an additional sample is stored one interval before the step change with the last reported value to prevent this step change from being viewed as a slow ramp in value. This value marks the end of the steady-state, non-changing value period, and provides a data point from which to begin the step change in value.

**Note:**
You can configure individual tags can be configured to retrieve step value changes.

The collector uses an algorithm that views the size of the step change and the number of intervals since the last reported value to determine if a marker value is needed. The following is an example of the algorithm:

```
BigDiff=abs(HI_EGU-LO_EGU)*(CompressionDeadbandPercent/(100.0*2.0))*4.0

If ( Collector Compression is Enabled )

If ( Elapsed time since LastReportedValue>=( SampleInterval * 5 ) )

If ( abs(CurrentValue-LastReportedValue) > BigDiff )

Write LastReportedValue,Timestamp=(CurrentTime-SampleInterval)
```

In the example above, if a new value was not reported for at least the last 4 sample intervals, and the new input value is at least 4 deltas away from the old value (where a single delta is equal to half of the compression deadband), then a marker value is written.

**Note:**
These settings are also adjustable from the Registry. Please contact technical support for more information.

### Value Spike with Collector Compression

For example, a collector reads a value X once per second, with a compression deadband of 1.0. If the value of X is 10.0 for a number of seconds starting at 0:00:00 and jumps to 20.0 at 0:00:10, the data samples read would be:

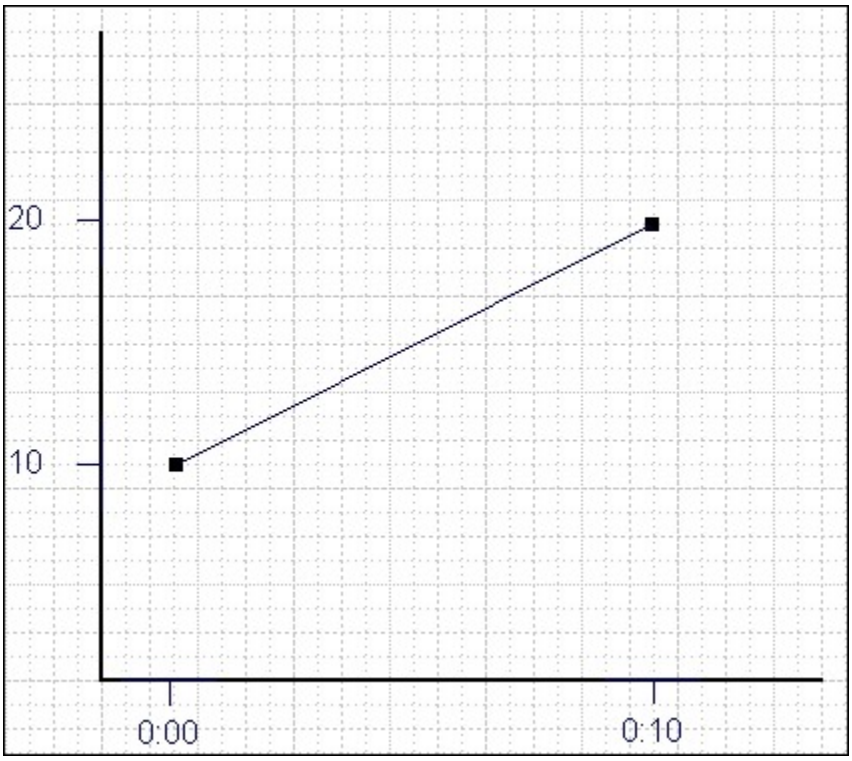
| Time | X Value |
|---|---|
| 0:00:00 | 10.0 (steady state value) |
| 0:00:01 | 10.0 |
| 0:00:02 | 10.0 |
| 0:00:03 | 10.0 |
| 0:00:04 | 10.0 |
| 0:00:05 | 10.0 |
| 0:00:06 | 10.0 |

| Time | X Value |
|---|---|
| 0:00:07 | 10.0 |
| 0:00:08 | 10.0 |
| 0:00:09 | 10.0 |
| 0:00:10 | 20.0 (new value after step change) |

To increase efficiency, the straightforward compression would store only 2 of these 11 samples.

| Time | X Value |
|---|---|
| 0:00:00 | 10.0 (steady state value) |
| 0:00:10 | 20.0 (new value after step change) |

However, without the marker value, if this data were to be put into a chart, it would look like the data value **ramped** over 10 seconds from a value of 10.0 to 20.0, as shown in the following chart.




The addition of a **marker value** to the data being stored results in the following data values:

| Time | X Value |
|---|---|
| 0:00:00 | 10.0 (steady state value) |
| 0:00:09 | 10.0 (inserted Marker value) |
| 0:00:10 | 20.0 (new value after step change) |

If you chart this data, the resulting trend accurately reflects the raw data and likely real world values during the time period as shown in the following chart.

## Evaluating and Controlling Data Compression

You can achieve optimum performance in Historian by carefully controlling the volume of dynamic data it collects and archives. You need enough information to tell you how the process is running, but you do not need to collect and store redundant or non-varying data values that provide no useful information.

### Control Data Flow

You can control the amount of online or dynamic data the system handles at a given time by adjusting certain system parameters. The general principle is to control the flow of data into the archive either by adjusting the rate at which the collectors gather data or by adjusting the degree of filtering (compression) the system applies to the data collected.

Adjust the following parameters to *reduce* the rate of data flow into the server.

- Reduce the polling rate by increasing the collection interval for unsolicited and polled collection.
- Enable collector compression and optionally use compression timeout.
- Set the compression deadband on the collectors to a wider value.
- Use the collector compression timeout.

Adjust the following parameters to *increase the filtering* applied by the archiver in the server.

- Enable archive (trend) compression.
- Set the archive compression deadband to a wider value.
- Where possible, use the scaled data type and enable input scaling on selected tags.
- Where possible, select milliseconds or microseconds rather than seconds for time resolution. Seconds is optimum for most common devices. This affects disk space.

**Evaluate Data Compression Performance**

You can determine how effectively data compression is functioning at any given time by examining the system statistics displayed on the **System Statistics** page of Historian Administrator.

The compression field at the top of the page shows the current effect of archive compression. Values for this parameter should typically range from 0 to 9%. If the value is zero, it indicates that compression is either ineffective or turned off. If it shows a value other than zero, it indicates that archive compression is operating and effective. The value itself indicates how well it is functioning. To increase the effect of data compression, increase the value of archive compression deadband so that compression becomes more active.

## Archive Compression

Archive compression is used to reduce the number of samples stored when data values for a tag form a straight line in any direction. For a horizontal line (non-changing value), the behavior is similar to collector compression. But, in archive compression, it is not the *values* that are being compared to a deadband, but the *slope of line* those values produce when plotted value against time. Archive compression logic is executed in the data archiver and, therefore, can be applied to tags populated by methods other than collectors.

You can use archive compression on tags where data is being added to a tag by migration, or by the File collector, or by an SDK program for instance. Each time the archiver receives a new value for a tag, the archiver computes a line between this incoming data point and the last archived value.

The deadband is calculated as a tolerance centered about the slope of this line. The slope is tested to see if it falls within the deadband tolerance calculated for the previous point. If the new point does not exceed the tolerance, it is not stored in the archive. This process repeats with subsequent points. When an incoming value exceeds the tolerance, the value held by the archiver is written to disk and the incoming sample is withheld.

The effect of the archive compression timeout is that the incoming sample is automatically considered to have exceeded compression. The withheld sample is archived to disk and the incoming sample becomes the new withheld sample. If the Archive Compression value on the System Statistics page indicates that archive compression is occurring, and you did not enable archive compression for the tags, the reason could be because of internal statistics tags with archive compression enabled.

For instructions on setting archive compression, refer to Access/Modify a Tag *(on page 894)*.

## About Scaling

Scaling converts a data value from a raw value expressed in an arbitrary range of units, such as a number of counts, to one in engineering units, such as gallons per minute or pounds per square inch. The scaled data type can serve as a third form of data compression, in addition to collector compression and archive compression, if it converts a data value from a data type that uses a large number of bytes to one that uses fewer bytes.

## About Condition-Based Collection

Condition based collection is a method to control the storage of data for data tags by assigning a condition. Data is always collected but it is only written to the Data Archiver if the condition is true; otherwise, the collected data is discarded.

This condition is driven by a trigger tag; a tag collected by the collector evaluating the condition. Ideally, Condition based Collection should be used only with tags that are updating faster than the trigger tag. Condition based collection can be used to archive only the specific data which is required for analysis, rather than archiving data at all times, as the collector is running.

For example, if a collector has tags for multiple pieces of equipment, you can stop collection of tags for one piece of equipment during its maintenance. It is typically used on tags that use fast polled collection but you don't want to use collector compression. While the equipment is running, you want all the data but when the equipment is stopped, you don't want any data stored. The trigger tag would also typically use polled collection. But, either tag could use unsolicited collection.

The condition is evaluated every time data is collected for the data tag. When a data sample is collected, the condition is evaluated and data is either queued for sending to archiver, or discarded. If the condition

cannot be evaluated as true or false, like if the trigger tag contains a bad data quality or the collector is not collecting the trigger tag, the condition is considered true and the data is queued for sending.

No specific processing occurs when the condition becomes true or false. If the condition becomes true, no sample is stored to the data tag using that condition, but the data tag will store a sample next time it collects. When the condition becomes false, no end of the collection marker is stored until the data tag is collected.

For example, if the condition becomes false at 1:15 and the data tag gets collected at 1:20, the end of collection marker will be created at 1:20 and have a timestamp of 1:20, not 1:15.

Condition based collection is supported by only archiver and collectors of Historian version 4.5 and above. Condition based collection does not apply to alarm collectors. This condition based collection is applicable to the following collectors only:

- Simulation Collector
- OPC Collector
- iFIX Collector
- PI Collector

For instructions on setting the condition-based collection, refer to Access/Modify a Tag *(on page 894)*.

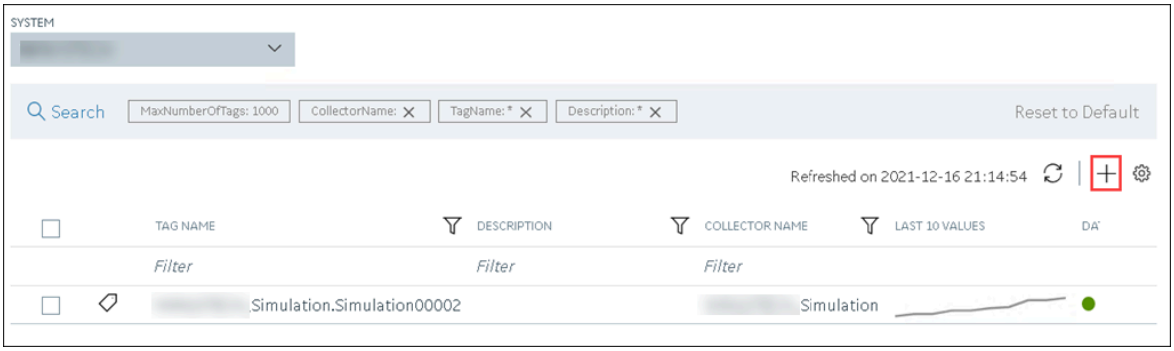
## Add Tags for the Data Store Using Configuration Hub

- Add the collector instance *(on page 357)* using which you want to collect data. Ensure that the collector is running.
- By default, the tag data is stored in the user data store, which is created automatically when you set up Configuration Hub. If, however, you want to store the data in a different data store, create it *(on page 370)*.

This topic describes how to specify the tags for which you want to collect data by browsing through the tags in the data source. For example, for an iFIX collector, if there are 1,00,000 tags in the iFIX server, you must specify the ones for which you want to collect data. Only then data is collected for those tags.

In addition to adding tags from the data source, you can create tags manually *(on page 473)*.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
3. Select +.

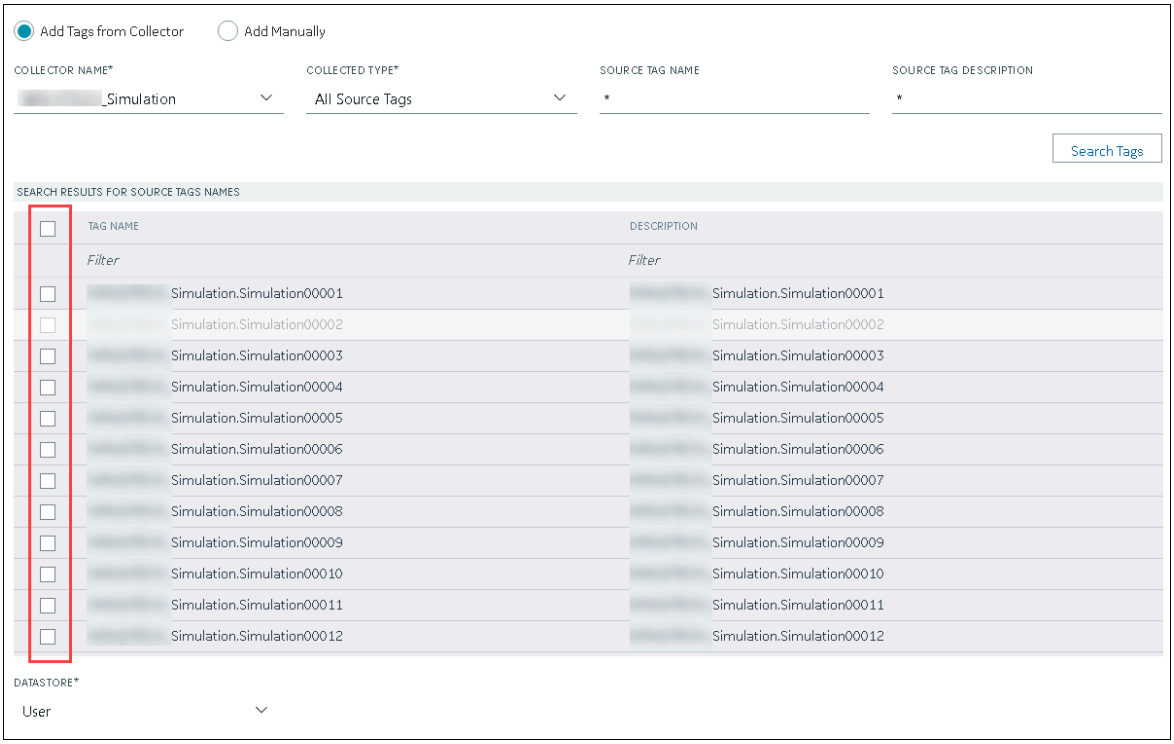



The **Add Tag-<system name>** page appears. The **Add Tags from Collector** option is selected by default.

4. Enter values as described in the following table.

| Field | Description |
|---|---|
| **COLLECTOR NAME** | Select the collector instance that you want to use to collect data. A value is required. |
| **COLLECTED TYPE** | Specify whether you want to browse through all the tags in the data source or only from the tags that you have not added yet. A value is required. |
| **SOURCE TAG NAME** | Enter the name of the tag (either completely or partially) to narrow down the search results. |
| **SOURCE TAG DESCRIPTION** | Enter the description of the tag (either completely or partially) to narrow down the search results. |

5. Select **Search Tags**.
   A list of tags that match *all* the criteria that you have specified appears. If a tag is already added, it is disabled.
6. Select the check box corresponding to each tag for which you want to collect data.




7. In the **DATA STORE** field, if you want to store the data in a different data store than the user data store, select the same.
8. Select **Add Tag**.
   Data collection begins for the selected tags.

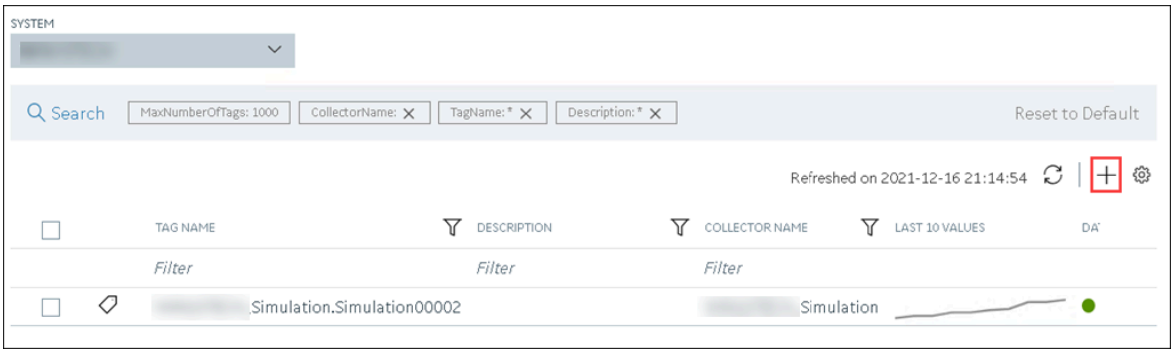
As needed, configure each tag by providing values for the tag properties. For information on the delta query modes, refer to Counter Delta Queries.

## Add a Tag Manually

- Add the collector instance *(on page 357)* using which you want to collect data.
- By default, the tag data is stored in the user data store, which is created automatically when you set up Configuration Hub. If, however, you want to store the data in a different data store, create it *(on page 370)*.

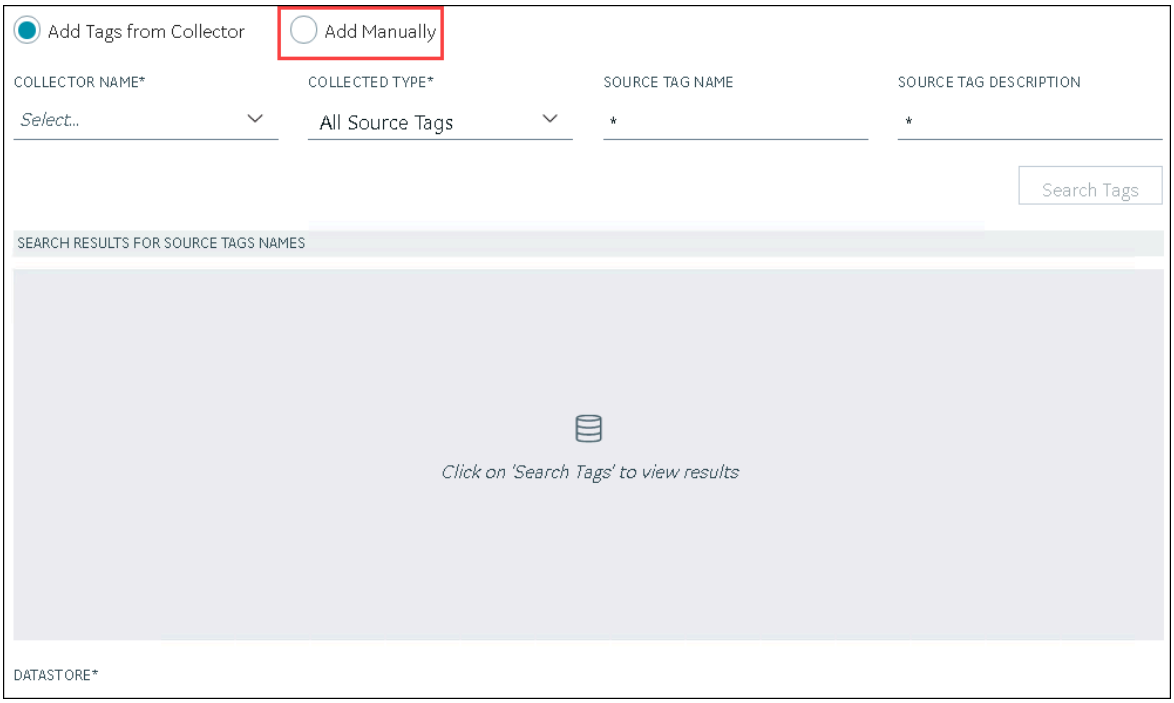
After you create a collector instance, you specify which tags from the source must be used for data collection *(on page 357)*. In addition, if you want to use the same tag twice (say, with a different collection interval or collector compression settings), you can add the tag manually. You can also create a calculation tag or a tag to store the values imported using the Excel Add-in.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
3. Select +.




The **Add Tag-<system name>** page appears. The **Add Tags from Collector** option is selected by default.

4. Select **Add Manually**.




5. Enter values as described in the following table.

| Field | Description |
|---|---|
| **COLLECTOR NAME** | Select the collector instance that you want to use to collect data. If, however, this tag is not associated with a collector, you |

| Field | Description |
|---|---|
| | can leave the field blank (for example, you want to ingest this data manually instead of using a collector). |
| **SOURCE ADDRESS** | Specify the source tag to which you want to map the one you are creating. This field is enabled only if you select a value in the **COLLECTOR NAME** field. When you select ▫▫▫, the **Browse Source Tag: <collector name>** window appears. Provide the search criteria to find the tag that you want to map. |
| **TAG NAME** | Enter a name for the tag. A value is required and must be unique for the Historian server.<br><br>The value that you enter:<br>◦ Must begin with a letter or a number.<br>◦ Can contain up to 256 characters.<br>◦ Can include any of the following special characters: /!\| #{}%$-_<br>◦ Must not include a space or any of the following characters: ~`+^:;.,?"*=@ |
| **DATA TYPE** | Select the data type of the tag data. To find out the data types supported by a collector, refer to the documentation on the collector that you have created.<br><br>**Important:**<br>If you select an unsupported data type, you may receive incorrect data or even lose data.<br><br>If you select **Multi-Field**, the **USER-DEFINED TYPE NAME** field appears, and the **ENUMERATED SET** and **ARRAY TAG** fields are disabled.<br><br>If you select **Fixed String**, the **STRING LENGTH** field appears. |
| **STRING LENGTH** | Enter the maximum character length allowed for the tag data. This field appears only if the value in the **DATA TYPE** field is **Fixed String**. A value is required.<br><br>You can enter a value between 1 and 255. The default value is 8. |

| Field | Description |
|---|---|
| **USER-DEFINED TYPE NAME** | Select the user-defined data type (UDT) *(on page 719)* that you want to assign to the tag. This field appears only if the value in the **DATA TYPE** field is **Multi-Field**. A value is required. |
| **ENUMERATED SET** | Select the enumerated set *(on page 707)* that you want to assign to the tag. This field is not applicable for string and multi-field data types (enumerated sets) and for array tags. |
| **ARRAY TAG** | Switch the toggle to indicate whether the tag stores an array of data. This field is disabled if you select a value in the **ENUMERATED SET** field or if the value in the **DATA TYPE** field is **Multi-Field**.<br><br>For information on array tags, refer to About Array Tags *(on page 660)*. |
| **TIME RESOLUTION** | Select the time resolution for the tag. A value is required.<br><br>For example, if you select **Seconds**, when you plot the data on a trend chart, the timestamp of the data points will be one second apart. |
| **DATA STORE** | If you want to store the data in a different data store that the user data store, select the same. |

6. Select **Add Tag**.
   Data collection begins for the selected tags.

As needed, configure each tag by providing values for the tag properties. For information on the delta query modes, refer to Counter Delta Queries.

## Access a Tag

To search for a tag, you can choose from the following options:

- Access all the tags in all the systems available in the Historian server.
- Access the tags added to a specific collector instance.
- Access the tags added to all the collector instances in a specific Historian system.

You can narrow down the search results further by performing a search.

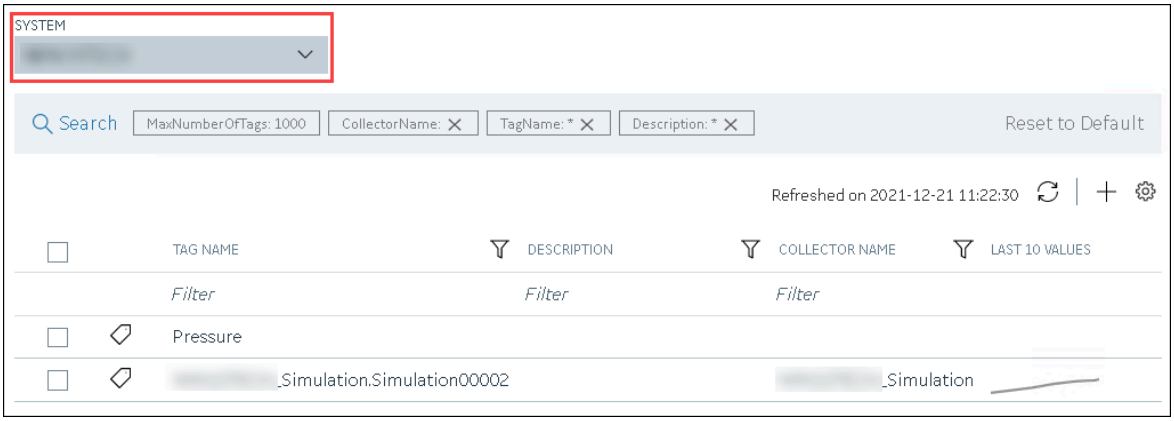
**Note:**
By default, maximum one million tags are retrieved. If the Historian clients are configured to retrieve more than a million tags, to retrieve all of them, add the `MaxTagsToRetrieve` registry key under `HKEY_LOCAL_MACHINE\SOFTWARE\Intellution, Inc.\iHistorian\Services\DataArchiver\`, and then set the maximum number of tags that you want to retrieve. Restart the Historian Data Archiver service for the change to reflect.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
   A list of all the tags appears, displaying the following information.

| Column | Description |
|---|---|
| **TAG NAME** | The name of the tag. |
| **DESCRIPTION** | The description of the tag. |
| **COLLECTOR NAME** | The name of the collector instance to which you have added the tag. |
| **LAST 10 VALUES** | The last 10 values collected for the tag, plotted as a trend chart. If you pause over the chart, the minimum, maximum, first, and last values among the 10 values appear. |
| **TAG ALIAS** | Indicates whether the tag contains aliases, which are created when you rename the tag using an alias *(on page 698)*. |

**Tip:**
You can show/hide/reorder columns in the table. For instructions, refer to Common Tasks in Configuration Hub *(on page 353)*.

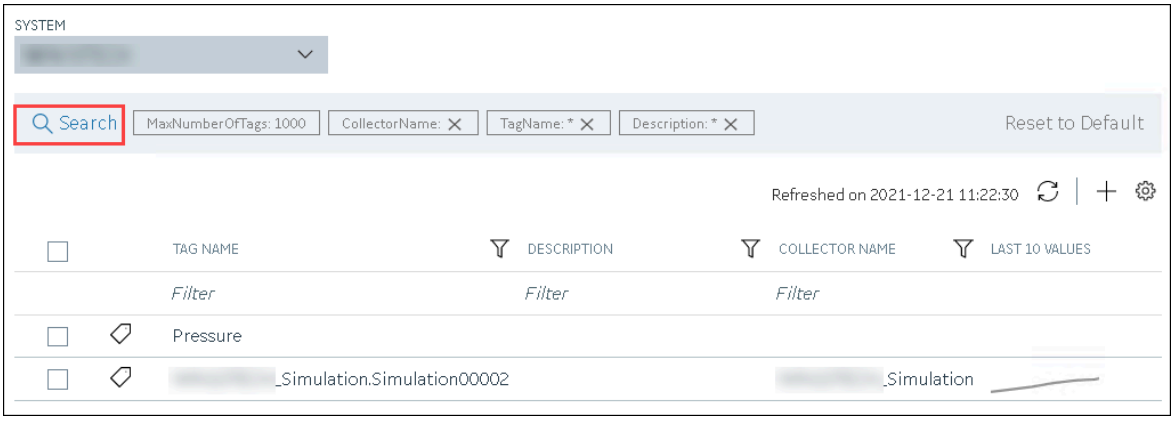
3. If you want to access the tags specific to a Historian system, in the drop-down list box in the upper-left corner of the main section, select the system.




Alternatively, you can access the system from the **NAVIGATION** section, right-click the system (or select ), and then select **Browse Tags**.

The list of tags is filtered to display only the tags specific to the system.

4. If you want to narrow down the search results, select **Search**.




Enter the search criteria, and then select **Search**. You can add more attributes by selecting **Add Attribute**. You can enter a name or a value partially or use the wildcard character asterisk (*).

The tags are filtered based on the search criteria.

5. Select the row containing the tag that you want to access.

The tag details appear in the **DETAILS** section.

**Table 35. The General Section**

| Field | Description |
|---|---|
| **Tag Name** | The name of the tag. This field is disabled and populated. |
| **Description** | The description of the tag. |
| **Comment** | Comments that apply to the tag. |

| Field | Description |
|---|---|
| **StepValue** | Indicates that the actual measured value changes in a sharp step instead of a smooth linear interpolation. This option is applicable only for numeric data. Enabling this option only affects data retrieval; it has no effect on data collection or storage. |
| **Last Modified Time** | The date the last tag parameter modification was made. This field is disabled and populated. |
| **Last Modified User** | The name of the person who last modified the tag configuration parameters. This field is disabled and populated. |

**Table 36. The Collection Section**

| Field | Description |
|---|---|
| **Collector** | The name of the collector that collects data for the selected tag. |
| **Source Address** | The address for the tag in the data source. Leave this field blank for tags associated with the Calculation or Server-to-Server collector.<br><br>For Python Expression tags, this field contains the full applicable JSON configuration, which includes an indication of the source address.<br><br>**Note:**<br>When exporting or importing tags using the Excel Add-in for Historian, the Calculation column, not the SourceAddress column, holds the formulas for tags associated with the Calculation or Server-to-Server collector. |
| **Data Type** | The data type of the tag.<br><br>The main use of the scaled data type is to save space, but this results in a loss of precision. Instead of using 4 bytes of data, it only uses 2 bytes by storing the data as a percentage of the EGU limit. Changing the EGU limits will result in a change in the values that are displayed. For example, if the original EGU values were 0 to 100 and a value of 20 was stored using the scaled data type and if the EGUs are changed to 0 to 200, the original value of 20 will be represented as 40. |

| Field | Description |
|---|---|
| | **Note:**<br>If you change the data type of an existing tag between a numeric and a string or binary data type (and vice versa), the tag's compression and scaling settings will be lost. |
| **Value** | The number of bytes for a fixed string data type. This field is enabled only for fixed string data types. |
| **Enumerated Set Name** | The name of the enumerated set *(on page 707)* that you want to assign to the tag. |
| **Array Tag** | Indicates that the tag is an array tag *(on page 660)*. |
| **Location** | The distributed location of the system in which the tag data is stored (applicable only for a horizontally scalable system). |
| **Data Store** | The data store in which the tag data is stored. |
| **Collection** | Indicates whether data collection is enabled or disabled for the tag. If you disable collection for the tag, Historian stops collecting data for the tag, but does not delete the tag or its data. |
| **Collection Type** | The type of data collection used for this tag, which can be polled or unsolicited. Polled means that the data collector requests data from the data source at the collection interval specified in the polling schedule. Unsolicited means that the data source sends data to the collector whenever necessary (independent of the data collector polling schedule). |
| **Collection Interval** | The time interval between readings of data from this tag. With Unsolicited Collection Type, this field defines the minimum interval at which unsolicited data should be sent by the data source. |
| **Collection Offset** | Used with the collection interval to schedule collection of data from a tag. For example, to collect a value for a tag every hour at thirty minutes past the hour (12:30, 1:30, 2:30, and so on), enter a collection interval of 1 hour and an offset of 30 minutes. Similarly, to collect a value each day at 8am, enter a collection interval of 1 day and an offset of 8 hours. |

| Field | Description |
|---|---|
| | **Note:**<br>If you enter a value in milliseconds, the value must be in intervals of 1000 ms. For example, 1000, 2000, and 3000 ms are valid values, but 500 and 1500 ms are invalid. The minimum value is 1000 ms. |
| **Time Resolution** | The precision for timestamps, which can be either seconds, milliseconds or microseconds. |
| **Condition-Based** | Indicates whether condition-based data collection *(on page 666)* is enabled. |
| **Trigger Tag** | The name of the trigger tag used in the condition. |
| **Comparison** | The comparison operator that you want to use in the condition. Select one of the following options:<br>◦ **Undefined**: Collection will resume only when the value of the triggered tag changes. This is considered an incomplete configuration, so condition-based collection is turned off and all the collected data is sent to archiver.<br>◦ **< =**: Setting condition as trigger tag value less than or equal to the compare value.<br>◦ **> =** Setting condition as trigger tag value greater than or equal to the compare value.<br>◦ **<**: Setting condition as trigger tag value less than the compare value.<br>◦ **>**: Setting condition as trigger tag value greater than the compare value.<br>◦ **=**: Setting condition as trigger tag value equals compare value.<br>◦ **!=**: Setting condition as trigger tag value not the same as compare value. |
| **Compare Value** | A target value that you want to compare with the value of the trigger tag. If using = and != comparison parameters, ensure that the format of the compared value and triggered tag are the same. For example, for a float type trigger tag, the compare value must be a float value; otherwise, the condition result is an invalid configuration. When the config- |

| Field | Description |
|---|---|
| | uration is invalid, condition-based collection is disabled and all data is sent to archiver. |
| **End of Collection Markers** | Indicates whether end-of-collection markers are enabled. This will mark all the tag's values as bad, and sub-quality as ConditionCollectionHalted when the condition becomes false. Trending and reporting applications can use this information to indicate that the real-world value was unknown after this time until the condition becomes true and a new sample is collected. If disabled, a bad data marker is not inserted when the condition becomes false. |

**Table 37. The Collection Options Section**

| Field | Description |
|---|---|
| **Data Collection** | Indicates whether data collection is enabled or disabled for the tag. If you disable collection for the tag, Historian stops collecting data for the tag, but does not delete the tag or its data. |
| **Collection Type** | The type of data collection used for this tag:<br>◦ **Polled**: The data collector requests data from the data source at the collection interval specified in the polling schedule.<br>◦ **Unsolicited**: The data source sends data to the collector whenever necessary (independent of the data collector polling schedule). |
| **Collection Interval** | The time interval between readings of data from this tag. For unsolicited collection type, this field defines the minimum interval at which unsolicited data should be sent by the data source. |
| **Collection Offset Value** and **Collection Offset** | Used with the collection interval to schedule collection of data from a tag. For example, to collect a value for a tag every hour at thirty minutes past the hour (12:30, 1:30, 2:30, and so on), enter a collection interval of 1 hour and an offset of 30 minutes. Similarly, to collect a value each day at 8am, enter a collection interval of 1 day and an offset of 8 hours.<br> **Note:**<br>If you enter a value in milliseconds, the value must be in intervals of 1000 ms. For example, 1000, 2000, and 3000 ms are |

| Field | Description |
|---|---|
| | valid values, but 500 and 1500 ms are invalid. The minimum value is 1000 ms. |
| **Time Resolution** | The precision for timestamps, which can be either seconds, milliseconds or microseconds. |
| **Condition based collection** | Indicates whether condition-based data collection *(on page 666)* is enabled. |
| **Trigger Tag** | The name of the trigger tag used in the condition. |
| **Comparison** | The comparison operator that you want to use in the condition. This field is enabled only if you have enabled condition-based collection. Select one of the following options:<br>◦ **Undefined**: Collection will resume only when the value of the triggered tag changes. This is considered an incomplete configuration, so condition-based collection is turned off and all the collected data is sent to archiver.<br>◦ **< =**: Setting condition as trigger tag value less than or equal to the compare value.<br>◦ **> =** Setting condition as trigger tag value greater than or equal to the compare value.<br>◦ **<**: Setting condition as trigger tag value less than the compare value.<br>◦ **>**: Setting condition as trigger tag value greater than the compare value.<br>◦ **=**: Setting condition as trigger tag value equals compare value.<br>◦ **!=**: Setting condition as trigger tag value not the same as compare value. |
| **Compare Value** | A target value that you want to compare with the value of the trigger tag. If using = and != comparison parameters, ensure that the format of the compared value and triggered tag are the same. For example, for a float type trigger tag, the compare value must be a float value; otherwise, the condition result is an invalid configuration. When the configuration is invalid, condition-based collection is disabled and all data is sent to archiver. |

| Field | Description |
|---|---|
| **End of Collection Markers** | Indicates whether end-of-collection markers are enabled. This will mark all the tag's values as bad, and sub-quality as ConditionCollectionHalted when the condition becomes false. Trending and reporting applications can use this information to indicate that the real-world value was unknown after this time until the condition becomes true and a new sample is collected. If disabled, a bad data marker is not inserted when the condition becomes false. |

**Table 38. The Scaling Section**

| Field | Description |
|---|---|
| **EGU Description** | The Engineering Units (EGU) provide context to a tag's value by providing an accurate representation of the tag values through their corresponding units. This will help you to know the units of data that you pull.<br><br>For example, you can enter "Temperature" or "degree Celsius" in **EGU Description** for a single tag or multiple tags associated with temperature values.<br><br>When you view the last 10 values of the tag *(on page 690)*, or when you generate a query *(on page 734)*/write *(on page 738)* report in the **Data** page, EGU you enter will be displayed under the **Engineering Units** column in the table view and as a tool tip in the trend view. |
| **Hi Engineering Units** | The current value of the upper range limit of the span for this tag.<br><br>Engineering Hi and Lo are retrieved automatically for F_CV fields for iFIX tags; all others are left at default settings. When adding tags from the server using an OPC Collector, the OPC Collector queries the server for the EGU units and EGU Hi/Lo limits. However, not all OPC Servers make this information available. Therefore, if the server does not provide the limits when requested to do so, the collector automatically assigns an EGU range of 0 to 10,000. |

| Field | Description |
|---|---|
| **Lo Engineering Units** | The current value of the lower range limit of the span for this tag. |
| **Input Scaling** | Indicates whether input scaling is enabled, which converts an input data point to an engineering units value.<br><br>For example, to rescale and save a 0 - 4096 input value to a scaled range of 0 - 100, enter 0 and 4096 as the low and high input scale values and 0 and 100 as the low and high engineering units values, respectively.<br><br>If a data point exceeds the high or low end of the input scaling range, Historian logs a bad data quality point with a ScaledOutOfRange subquality. In the previous example, if your input data is less than 0, or greater than 4096, Historian records a bad data quality for the data point.<br><br>**OPC Servers and TRUE Values:** Some OPC servers return a TRUE value as -1. If your OPC server is returning TRUE values as -1, modify the following scaling settings:<br>◦ **Hi Engineering Units**: Enter `0`.<br>◦ **Lo Engineering Units**: Enter `1`.<br>◦ **Hi Scaling Value**: Enter `0`.<br>◦ **Lo Scaling Value**: Enter `-1`.<br>◦ **Input Scaling**: Enable this option. |
| **Hi Scaling Value** | The upper limit of the span of the input value. |
| **Lo Scaling Value** | The lower limit of the span of the input value. |

**Table 39. The Collector Compression Section**

| Field | Description |
|---|---|
| **Collector Compression** | Indicates whether collector compression *(on page 660)* is enabled. |
| **Collector Deadband** and **Deadband value** | The current value of the compression deadband. This value can be computed as a percent of the span, centered around the data value or given as an absolute range around the data value. |

| Field | Description |
|---|---|
| | **Note:**<br>Some OPC servers add and subtract the whole deadband value from the last data value. This effectively doubles the magnitude of the deadband compared to other OPC servers. To determine how your specific server handles deadband, refer to the documentation of your OPC server.<br><br>**Example:**<br><br>Suppose the engineering units are 0 to 200. Suppose the deadband value is 10%, which is 20 units. If the deadband value is 10% and the last reported value is 50, the value will be reported when the current value exceeds 50 + 10 = 60 or is less than 50 - 10 = 40. Note that the deadband (20 units) is split around the last data value (10 on either side.)<br><br>Alternatively, you could specify an absolute deadband of 5. In this case, if the last value was 50, a new data sample will be reported when the current value exceeds 55 or drops below 45.<br><br>If compression is enabled and the deadband is set to zero, the collector ignores data values that do not change and records any that do change. If you set the deadband to a non-zero value, the collector records any value that lies outside the deadband. If the value changes drastically, a pre-spike point may be inserted. For information, refer to Enable Spike Logic *(on page 970)*. |
| **Engineering Unit** | Converts the deadband percentage into engineering units and displays the result. This value establishes the deadband range that is centered around the new value.<br><br>This field represents a calculated number created to give an idea of how large a deadband you are creating in engineering units. The deadband is entered in percentage and Historian converts the percentage in to engineering units. |
| **Compression Timeout** and **Compression Timeout Interval** | Indicates the maximum amount of time the collector will wait between sending samples for a tag to the archiver. This time is maintained per tag, as different tags report to the archiver at different times. |

| Field | Description |
|---|---|
| | For polled tags, this value should be in multiples of your collection interval. After the timeout value is exceeded, the tag stores a value at the next scheduled collection interval, and not when the timeout occurred. For example, if you have a 10-second collection interval, a 1-minute compression timeout, and a collection that started at 2:14:00, if the value has not changed, the value is logged at 2:15:10 and not at 2:15:00.<br><br>For unsolicited tags, a value is guaranteed in, at most, twice the compression timeout interval.<br><br>A non-changing value is logged on each compression timeout. For example, an unsolicited tag with a 1-second collection interval and a 30-second compression timeout is stored every 30 seconds.<br><br>A changing value for the same tag may have up to 60 seconds between raw samples. In this case, if the value changes after 10 seconds, then that value is stored, but the value at 30 seconds (if unchanged) will not be stored. The value at 60 seconds will be stored. This leaves a gap of 50 seconds between raw samples which is less than 60 seconds.<br><br>Compression timeout is supported in all collectors except the PI collector. |

**Table 40. The Archive Compression Section**

| Field | Description |
|---|---|
| **Archive Compression** | Indicates whether archive compression *(on page 660)* is enabled. If enabled, Historian applies the archive deadband settings against all reported data from the collector. |
| **Archive Deadband** and **Deadband value** | The current value of the archive deadband, expressed as a percent of span or an absolute number. |
| **Engineering Unit** | Converts the deadband percentage into engineering units and displays the result. This value establishes the deadband range that is centered around the new value. |

| Field | Description |
|---|---|
| **Compression Timeout** and **Compression Timeout Interval** | The maximum amount of time from the last stored point before another point is stored, if the value does not exceed the archive compression deadband.<br><br>The data archiver treats the incoming sample after the timeout occurs as if it exceeded compression. It then stores the pending sample. |

**Table 41. The Advanced Section**

| Field | Description |
|---|---|
| **Time Assigned By** | The source of the timestamp for a data value is either the collector or the data source.<br><br>All tags, by default, have their time assigned by the collector. When you configure a tag for a polled collection rate, the tag is updated based on the collection interval. For example, if you set the collection interval to 5 seconds with no compression, then the archive will be updated with a new data point and timestamp every 5 seconds, even if the value is not changing.<br><br>However, if you set the **Time Assigned By** field to **Source** for the same tag, the archive only updates when the device timestamp changes. For example, if the poll time is still 5 seconds, but if the timestamp on the device does not change for 10 minutes, no new data will be added to the archive for 10 minutes.<br><br>**Note:**<br>This field is disabled for Calculation and Server-to-Server tags. |
| **Time Zone Bias** | The number of minutes from GMT that should be used to translate timestamps when retrieving data from this tag. For example, the time zone bias for Eastern Standard time is -300 minutes (GMT-5).<br><br>This field is not used during collection. Use this option if a particular tag requires a time zone adjustment during retrieval other than the client or server time zone. For example, you could retrieve data for two tags with different time zones by using the tag time zone selection in the iFIX chart. |

| Field | Description |
|---|---|
| **Time Adjustment** | If the Server-to-Server collector is not running on the source computer, select the **Adjust for Source Time Difference** option to compensate for the time difference between the source archiver computer and the collector computer.<br><br>**Note:**<br>This field only applies to tags associated with the Server-to-Server collector that use a polled collection type. |

**Table 42. The Security Section**

| Field | Description |
|---|---|
| **Read Group** | The Windows security group that can retrieve the tag data and plot it in a trend chart.<br><br>For example, if you select a group with power users, in addition to members of the iH Security Admins group, only a member of the power users group will be able to read data for that tag. Even a member of the iH Readers group will not be able to access data for that tag, unless they are also defined as a member of the power users group. |
| **Write Group** | The Windows security group that can write tag data (for example, using the Excel Add-in for Historian). |
| **Administer Group** | The Windows security group that can create, modify, and delete the tag. |
| For more information, refer to implementing tag-level security *(on page 291)*.<br><br>**Note:**<br>When it comes to the group security, the security settings applied at the tag level, if any, take the precedence over those at the data store level. | |

**Note:**
If you are using domain groups (instead of local groups), the **Read Group**, **Write Group**, and **Administer Group** fields contain only the groups whose names begin with iH<space> (case-sensitive). Therefore, ensure that the group that you want to use begins with iH<space>.

**Table 43. The Delta Query Section**

| Field | Description |
|---|---|
| **Delta Max Value** | The maximum value that a tag can have. It also called the rollover point of the counter or totalizer. If the tag values exceed MaxValue, the counter is reset to the minimum value. If you do not provide MaxValue, the delta query cannot check for a positive counter wrap. |
| **Delta Min Value** | The minimum value that a tag can have. If the tag values are less than MinValue (and the counter is going in the negative direction), the tag values are reset to MaxValue. If you do not provide MinValue, 0 is considered. |
| **Delta Max Positive RPH** | The maximum rate per hour between two consecutive data points in the positive direction. If two consecutive data points exceed this value, they are not considered in a delta query. |
| **Delta Max Negative RPH** | The maximum rate per hour between two consecutive data points in the negative direction. If two consecutive data points exceed this value, they are not considered in a delta query.<br><br>The Delta Max Positive RPH and Delta Max Negative RPH values are used to determine if a counter wrap has occurred or if the counter has been manually reset. They help ignore data points whose values increase or decrease drastically. |

For information on how these values are calculated, refer to Counter Delta Queries *(on page 1113)*.

**The Spare Fields section**

Spare configuration enables you to add additional configuration to the tag using the **Spare Field 1** to **Spare Field 5** fields in all the collectors except in a Server-to-Server collector, Server-to-Server distributor, and an OSI PI distributor.

- In case of an OSI PI distributor, data is read from the Historian tag displayed in the **Source Address** field and sent to the OSI PI tag name displayed in the **Spare Field 1** field. To control the source and destination tags, change the **Source Address** and **Spare Field 1** values. You can add or update values in the remaining spare fields.
- In case of Server-to-Server collector and Server-to-Server distributor, you can update the **Spare Field 1** to **Spare Field 4** values, but the **Spare Field 5** field is used only for internal purposes. Therefore, do not update the **Spare Field 5** field.

**The TAG ALIAS Section**

Contains a list of the old names of the tag that you have renamed using an alias. For more information, refer to Rename a Tag *(on page 698)*.

## Configure Multiple Tags

This topic describes how to select multiple tags and configure their properties. This allows you to select two or more tags and configure their relevant properties at once. By default, 100 tags are listed at once in the grid. If you need to configure more than 100 tags simultaneously, you can use the page numbers at the bottom-right corner of the grid to navigate to the next available tags and select them as needed. Depending on the data types of the tags you select, all the properties that are relevant to the selected tags will be available in the **DETAILS** section for configuration.

Before you begin the configuration of multiple tags, know the following:

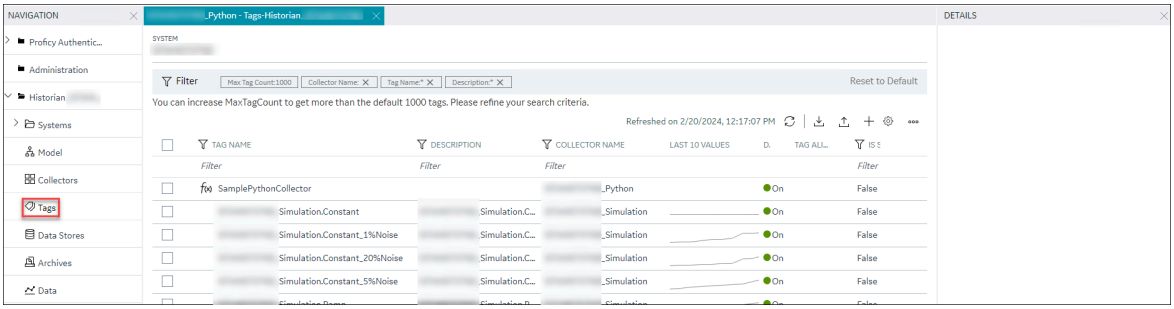
- When you select multiple tags, only the common properties will be displayed in the **DETAILS** section.
- The configuration values displayed in the **DETAILS** section will be of the last tag that is selected. However, the properties you update will be applied to all the other selected tags.
- When you configure multiple tags, only the configured properties are updated, and all the other properties remain unchanged.
- When you select multiple tags, aliases are disabled.

The table below provides the properties that will be disabled based on the selection of different tag data types.

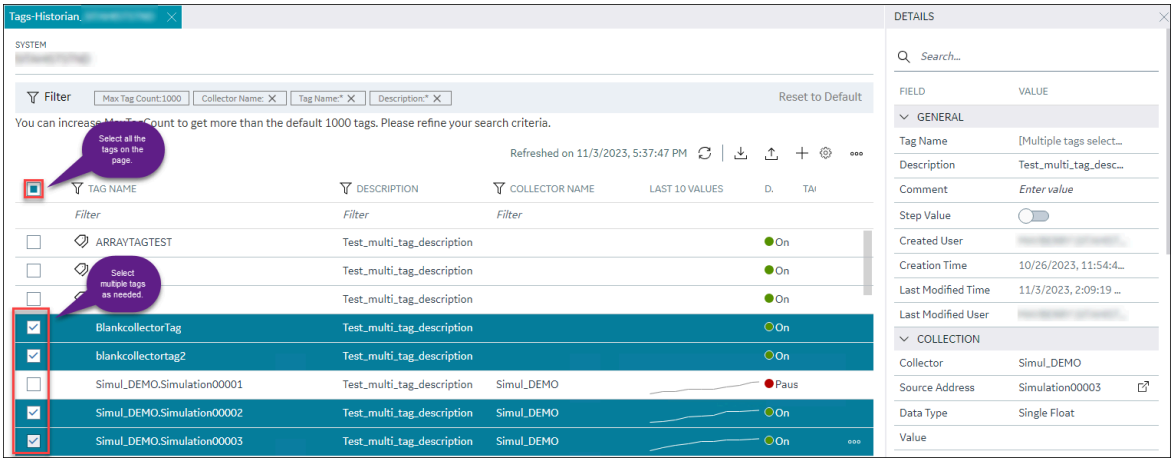
| Tag Data Type/ Tag Type | Properties Disabled |
|---|---|
| Array, Blob, Multifield | Compression, Scaling, Calculation, and Delta mode. |

| Tag Data Type/ Tag Type | Properties Disabled |
|---|---|
| Fixed string, Variable string | Compression, Scaling, Calculation, and Delta mode. |
| Calculation | None. |
| Calculation combined with other tag data types | Condition based collection, Time assigned by, and other relevant properties.<br><br>For example, if you select Calculation, Array, and Blob, Condition based collection, Time assigned by, Compression, Scaling, Calculation, and Delta mode are disabled. |
| Non-calculation, Array, Multifield | All the properties other than General, Collection, Scaling, Compression, Advanced, Security, Delta, and Spare. |
| Multifield | Array tag. |
| UDT, Enum tags | Enumerated set. |
| Array, Blob, Scaled, Fixed string, Variable string | Enumerated set. |

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.




3. To select multiple tags, select the check boxes corresponding to the tags as needed. Alternatively, to select all the available tags, select the check box in the upper-left corner in the grid header.



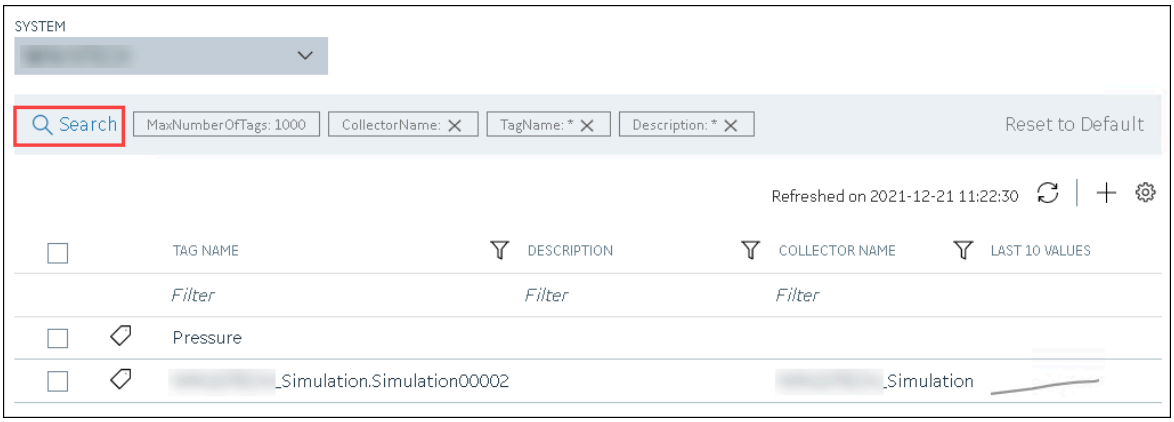

4. In the **DETAILS** section, edit the available properties as needed.
5. After you edit, in the top-left corner of the main ribbon bar, select **Save**.

## Access the Trend Chart of Tag Values

This topic describes how to access the values of a tag in a trend chart. The difference in the timestamp of consecutive values depends on the time resolution of the tag. For example, if the time resolution is seconds, the timestamp of consecutive values of the tag will be one second apart.

You can plot the trend chart for up to 10 tags at a time.

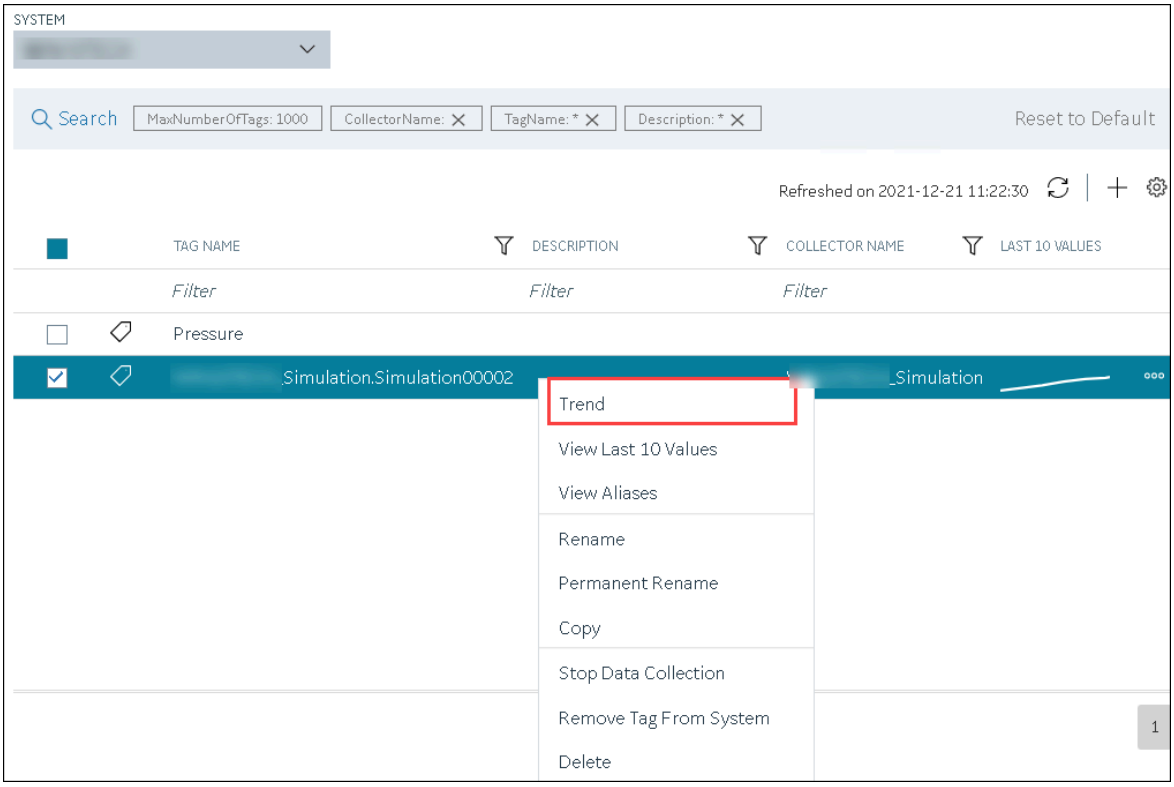
1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
   A list of all the tags appears.
3. If you want to narrow down the search results, select **Search**.




Enter the search criteria, and then select **Search**. You can add more attributes by selecting **Add Attribute**. You can enter a name or a value partially or use the wildcard character asterisk (*).

The list of tags are filtered based on the search criteria.

4. If you want to access the trend chart of a single tag, right-click the tag (or select ⋯), and then select **Trend**.




If you want to access the trend chart for multiple tags, select the check boxes corresponding to the tags, right-click (or select ⋯), and then select **Trend**. You can select up to 10 tags.
The tag values are plotted on a trend chart. You can switch between the trend view and the table view.

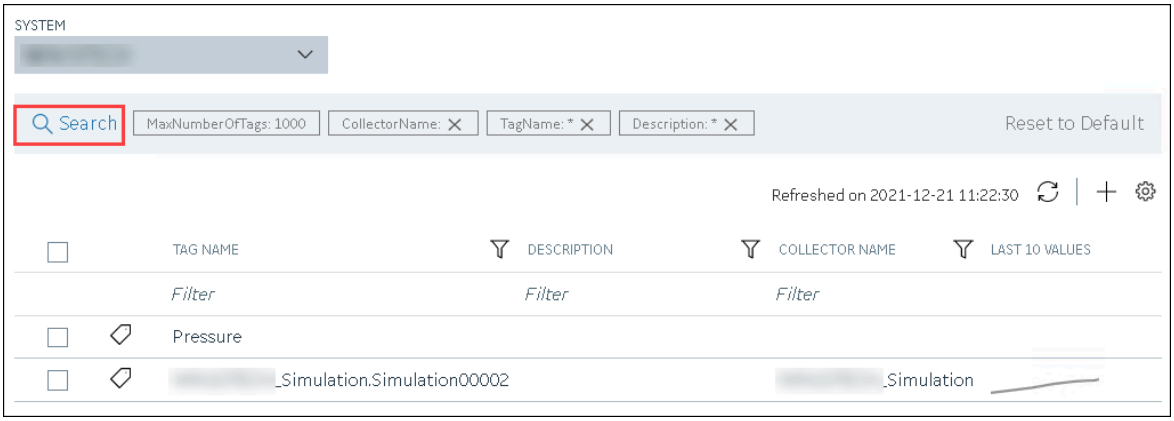
> **Tip:**
> You can also filter the chart by changing the duration, sampling type, time interval, and so on by selecting ▽.

## Access the Last 10 Values of a Tag

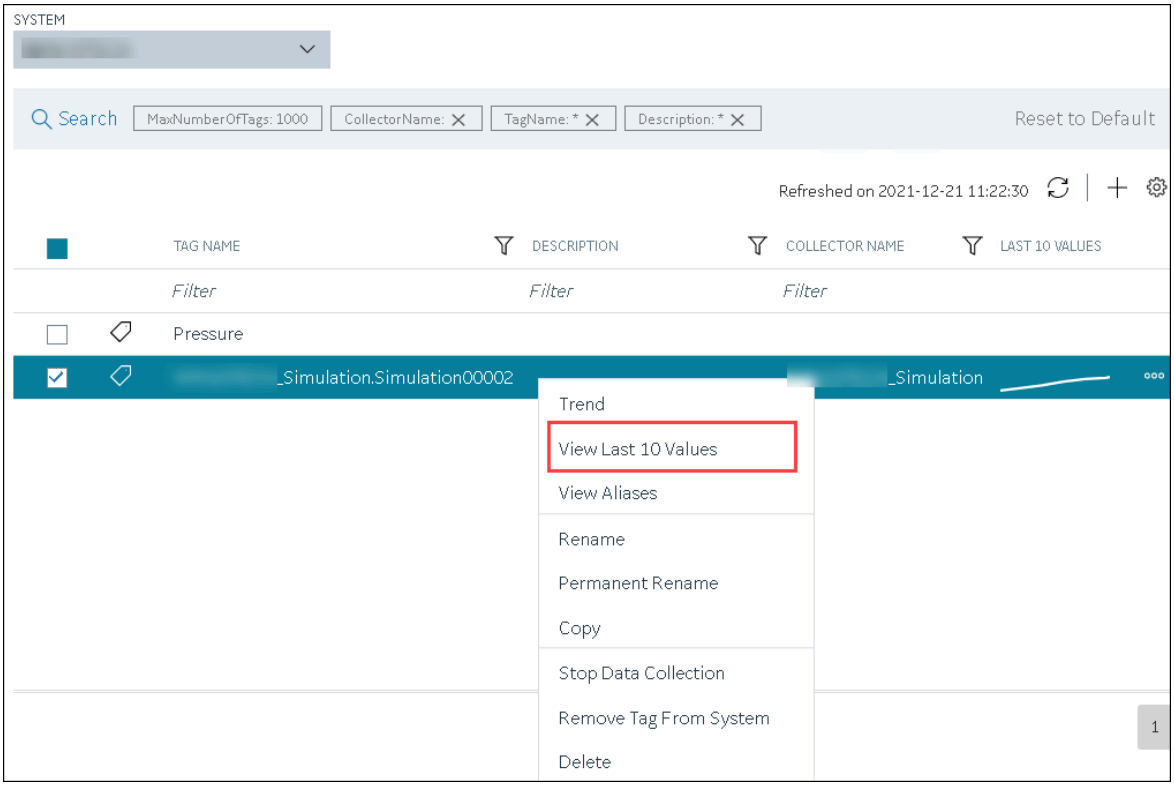
This topic describes how to access the last 10 values of a tag up to the current time. The difference in the timestamp of consecutive values depends on the time resolution of the tag. For example, if the time resolution is seconds, the timestamp of consecutive values of the tag will be one second apart.

**Note:**
If a tag name contains a comma or a semicolon, you cannot view the last 10 values of the tag.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
   A list of all the tags appears.
3. If you want to narrow down the search results, select **Search**.




   Enter the search criteria, and then select **Search**. You can add more attributes by selecting **Add Attribute**. You can enter a name or a value partially or use the wildcard character asterisk (*).
   The list of tags are filtered based on the search criteria.
4. If you want to access the last 10 values of a single tag, right-click the tag (or select ⋯), and then select **View Last 10 Values**.




The last 10 values of the tag appear, along with the timestamp and quality of each value. You can switch between the trend view and the table view.

5. If you want to access the last 10 values of multiple tags, select the check boxes corresponding to the tags, right-click (or select ⋯), and then select **Trend**. You can select up to 10 tags.

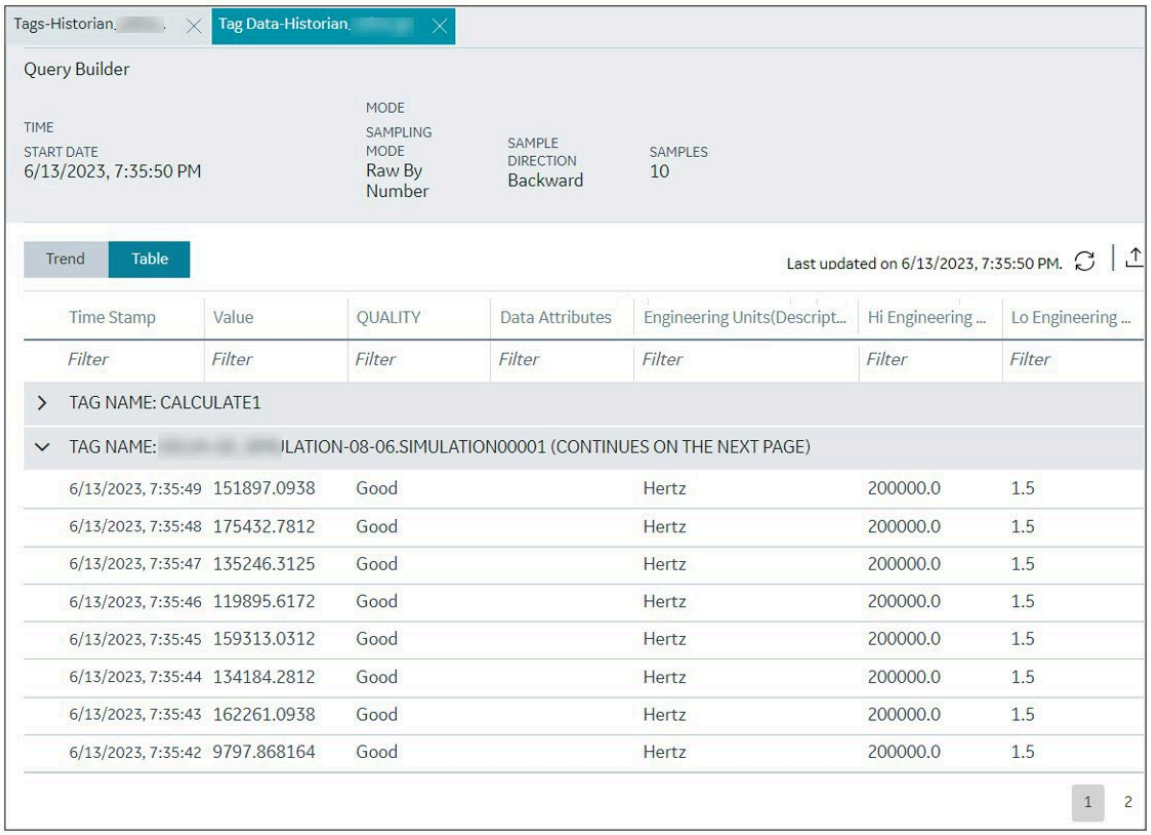
   In table view you can view data attributes which are introduced to store the 128-bit subquality for every sample.

> **Note:**
> SCADA applications such as Habitat support data samples with several quality types. To support such SCADA applications, Historian is now enhanced to store up to 128-bit quality types, which are stored in the data attributes. These attributes are extended qualities that you can store more than the regular qualities and subqualities (such as good and bad). In addition to regular qualities, the HAB collector collects extends qualities such as replaced, suspect, garbage, old, and so on.

The Data Attributes are displayed in the table view along with Timestamp, Value, and Quality. Data Attributes are supported for Current Value, Raw by Number, Raw by Time, Lab, and Lab to Raw. In addition:

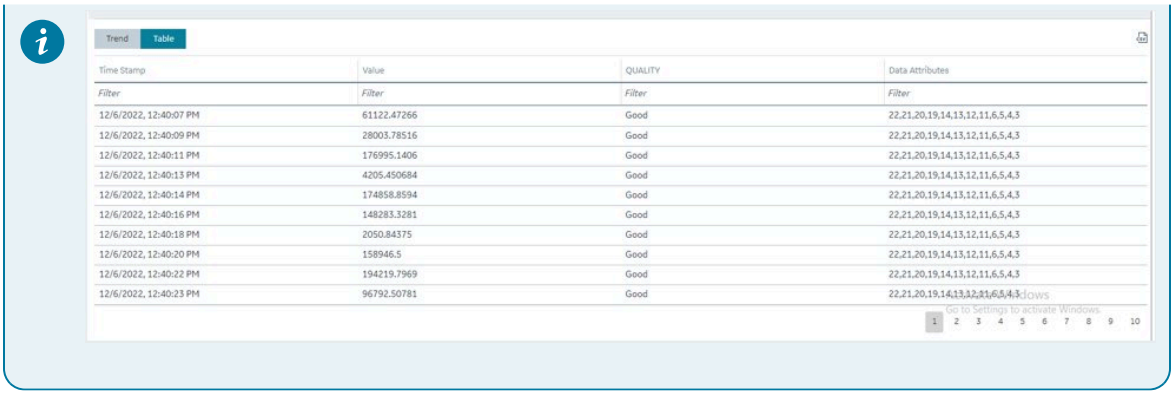
- For an array tag, all the values in the array appear for each timestamp.
- For a tag using an enumerated set, values of all the states appear for each timestamp.
- For a tag using a user-defined data type (UDT), values for all the fields appear for each timestamp.

6. For the selected tag, to view the results in a table, select **Table**.

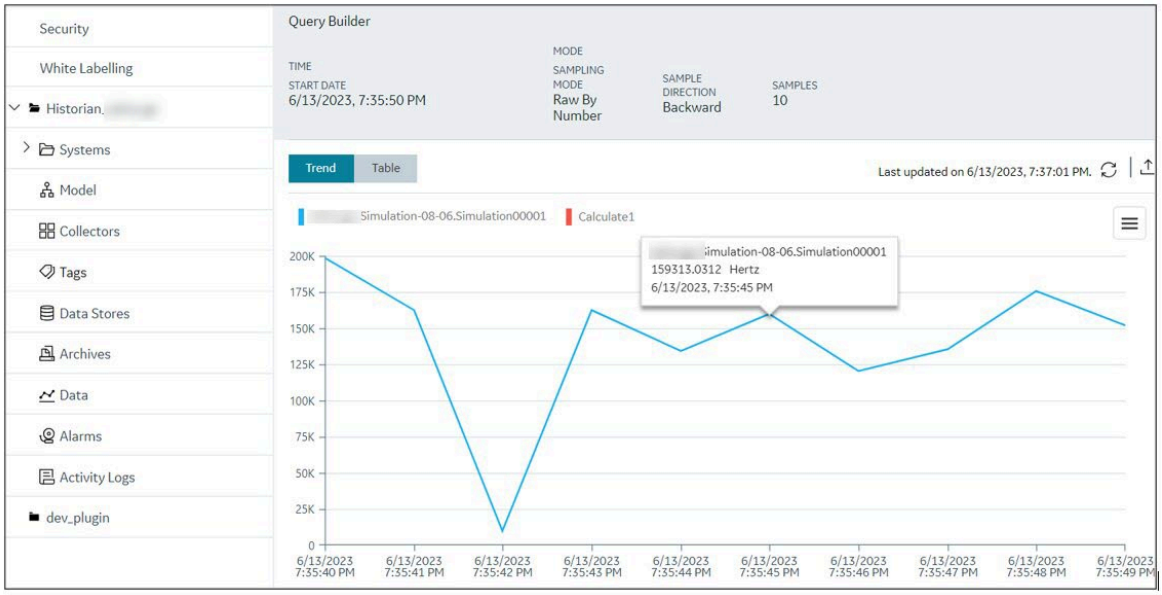



**Tip:**

You can also filter the values by changing the duration, sampling type, time interval, engineering units, and so on by selecting .




7. For the selected tag, to view the results in a trend chart, select **Trend**.

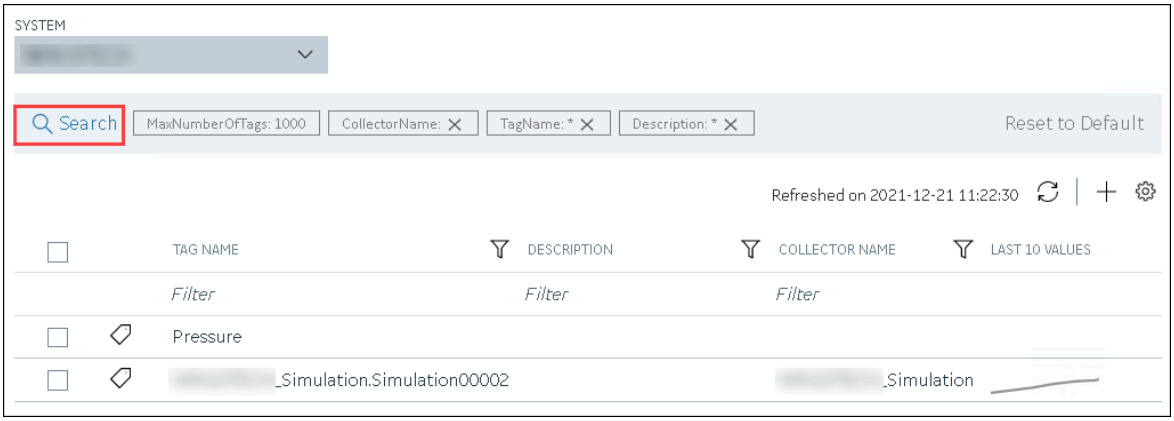



## Access a Tag Alias

After you rename a tag, the old name is called the tag alias. You can retrieve tag data using the tag alias as well. When you copy a tag, the tag alias is also captured to aid in an audit trail.

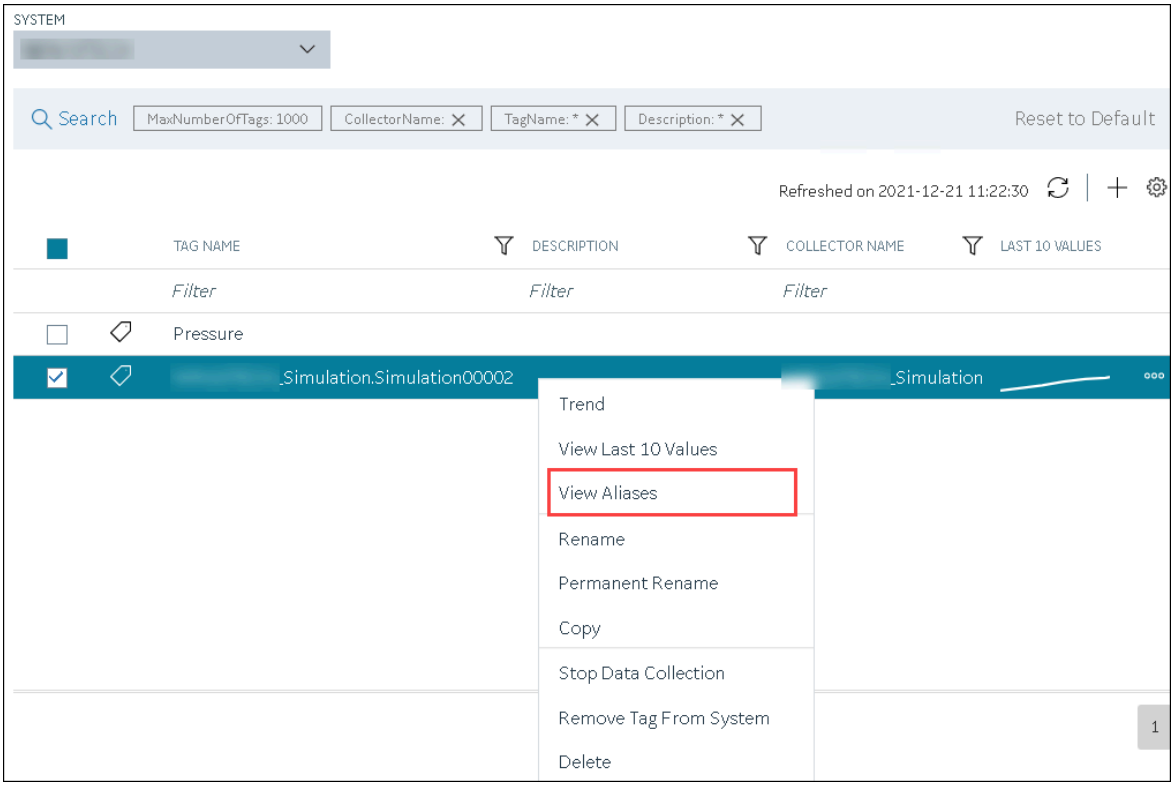
**Note:**
If a tag name contains a comma or a semicolon, you cannot view the tag alias.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
   A list of all the tags appears.
3. If you want to narrow down the search results, select **Search**.




Enter the search criteria, and then select **Search**. You can add more attributes by selecting **Add Attribute**. You can enter a name or a value partially or use the wildcard character asterisk (*). The list of tags are filtered based on the search criteria.

4. Right-click the tag whose alias you want to access (or select ⋯), and then select **View Aliases**.




Alternatively, you can select in the **TAG ALIAS** column.

All the tag aliases of the selected tag appear.

## Export Tags as a CSV File

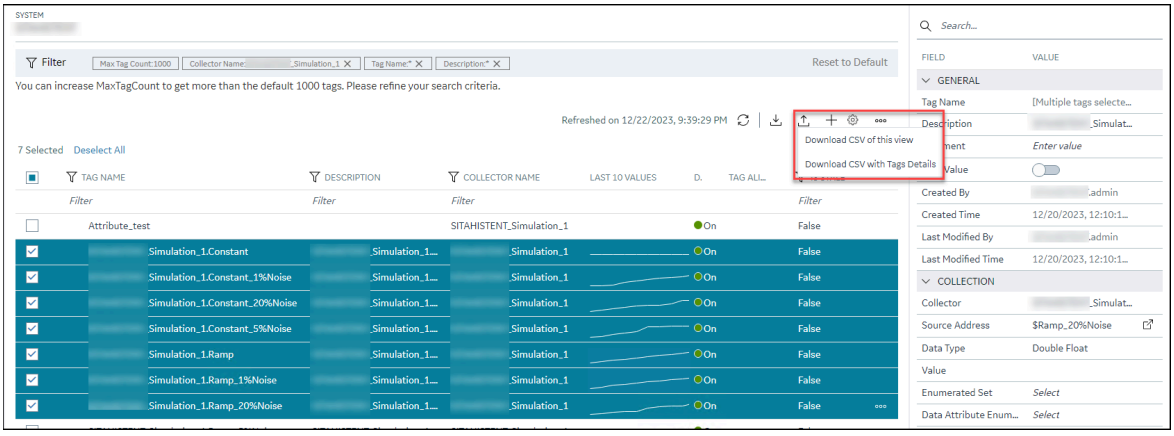
This topic describes how to export tags as a CSV file. You can export tags from Configuration Hub as a CSV file and add/modify tags in bulk and then import them. You can also import the CSV file that was exported using Excel Add-in.

**Note:**
Similarly, you can also export enumerated sets, and user-defined types from the **Manage Enumerated Sets** and **Manage User-Defined Types** windows.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
   A list of all the tags appears, displaying the corresponding details.
3. Select and then select any one of the following options:

| Option | Description |
|---|---|
| **Download CSV of this View** | Select this option if you want to download the selected tags and their corresponding details as listed in the tags grid. |
| **Download CSV with Tag Details** | Select this option if you want to download the selected tags and all their details, including configurations, from the **DETAILS** section. |

**Note:**
Irrespective of the number of tags you select, all the available tags will be exported.



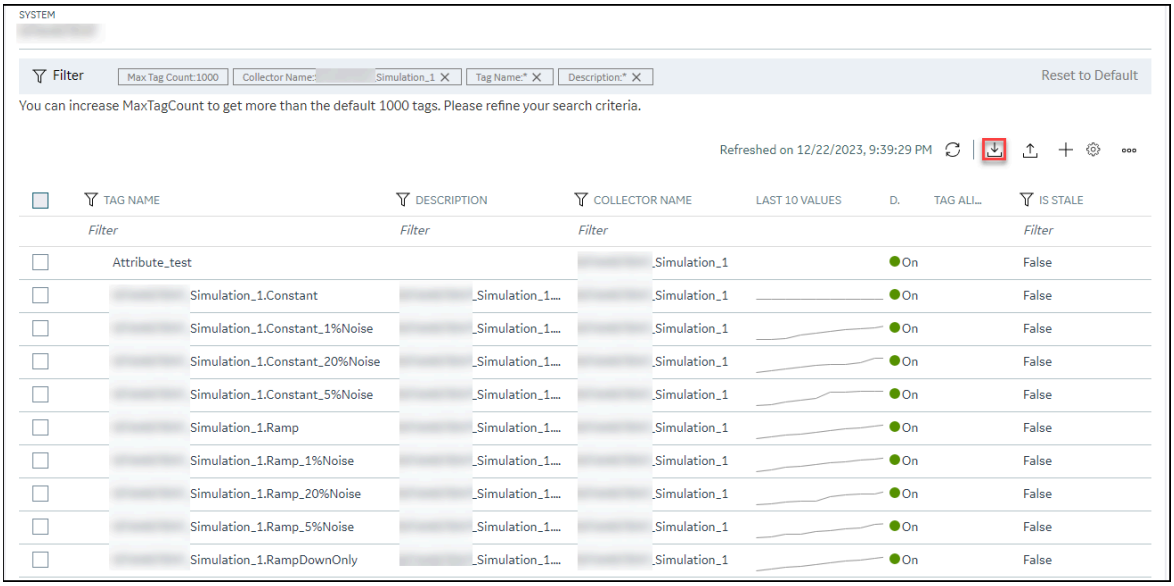

4. Enter a name for the file and save it in a location as needed.

# Import Tags from a CSV File

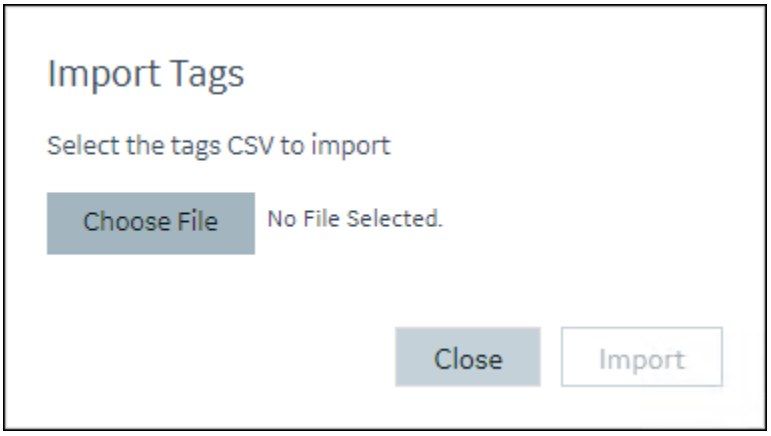
This topic describes how to import tags into Configuration Hub from a CSV file. You can either export tags and their details as a CSV file and edit them, and then import them back, or you can import tags and their details from another CSV file that you created externally, like Excel Add-in.

**Note:**
Similarly, you can also import enumerated sets, and user-defined types from the **Manage Enumerated Sets** and **Manage User-Defined Types** windows.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
   A list of all the tags appears, displaying the corresponding details.
3. In upper-right corner of the grid, select .




The **Import Tags** page appears.




4. Select **Choose File**, and then select the CSV file as needed.

> **Note:**
> If you import any irrelevant CSV file, the import will fail and you will be notified.

5. Select **Import**.
   The tags and their details are imported.

## Rename a Tag

To rename a tag, you must be a member of the administrator's group with tag-level security.

When you rename a tag, you can choose between the following options:

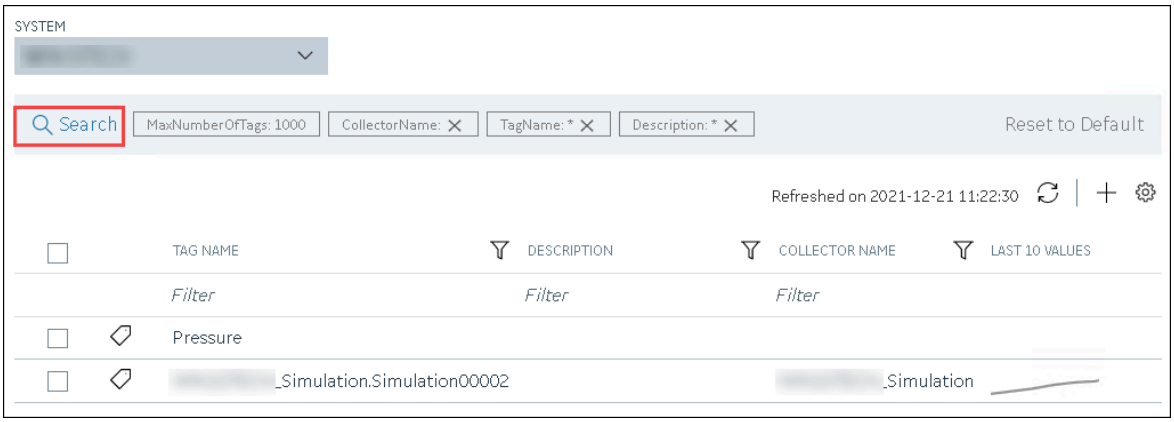
- **Rename using an alias:** In this case, the old name is called the tag alias. You can retrieve tag data using the tag alias as well. When you copy a tag, the tag alias is captured as well to aid in an audit trail.

> **Note:**
> If a tag name contains a comma or a semicolon, you cannot view the tag alias.

- **Rename permanently:** In this case, the old name is no longer captured. Therefore, you can create another tag with this old name. You cannot store and forward data using the old name. This implies that data for the tag is collected separately for the new name.

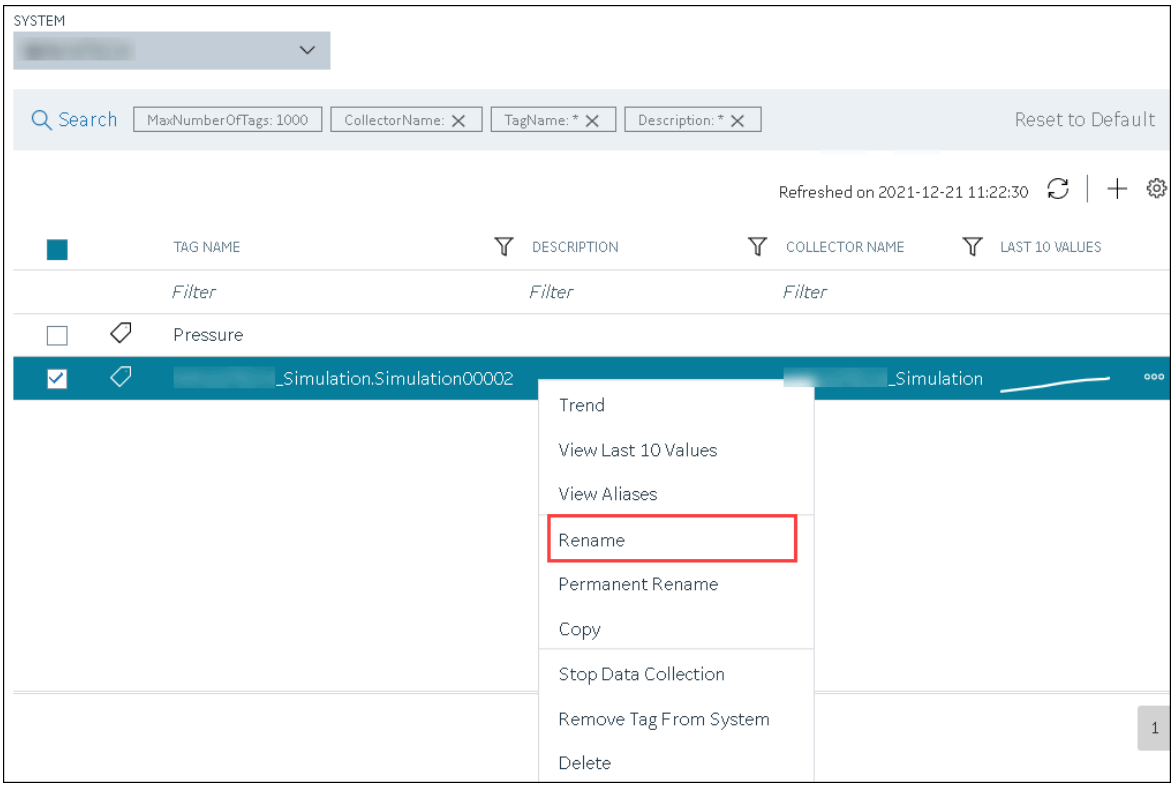
1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
   A list of all the tags appears.

3. If you want to narrow down the search results, select **Search**.

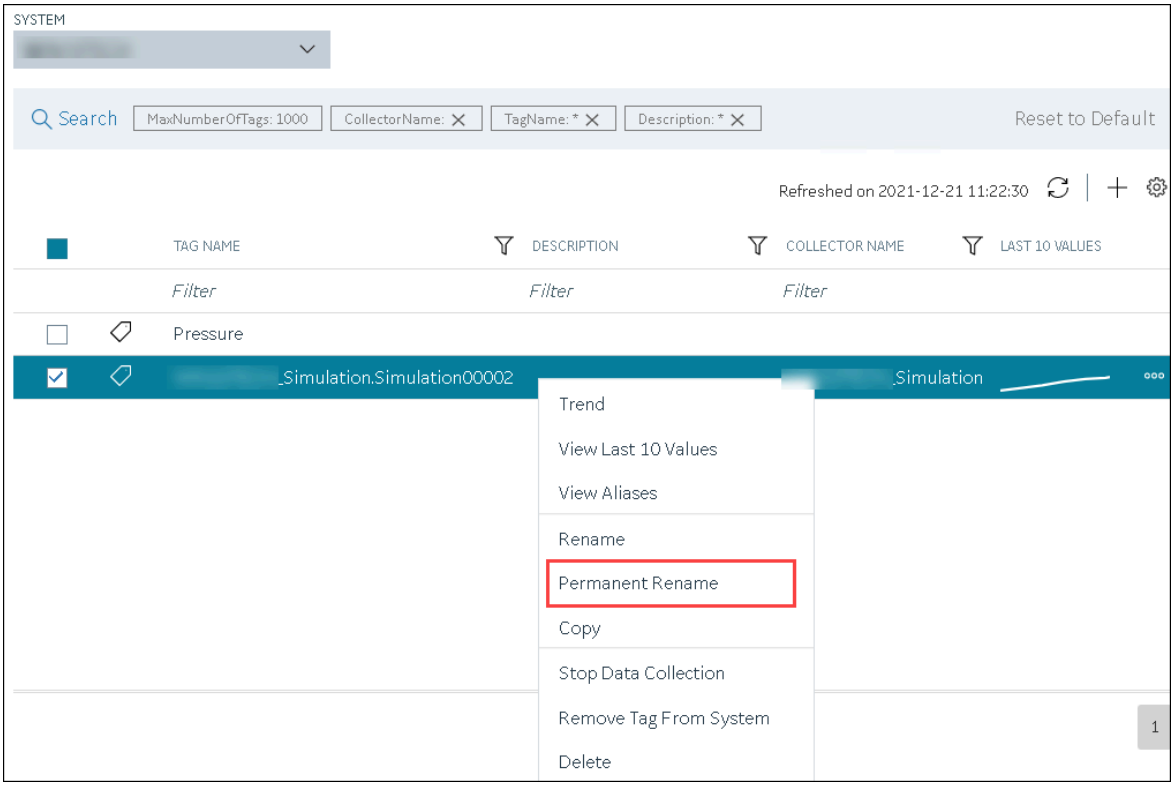



Enter the search criteria, and then select **Search**. You can add more attributes by selecting **Add Attribute**. You can enter a name or a value partially or use the wildcard character asterisk (*). The list of tags are filtered based on the search criteria.

4. Right-click the tag that you want to rename (or select ⋯).
5. If you want to rename the tag using an alias, select **Rename**.




If you want to rename the tag permanently, select **Permanent Rename**.




The **Rename Tag: <tag name>** window or the **Permanent Rename Tag: <tag name>** window appears.

6. Enter the new name of the tag, and then select **Rename**. The tag name must be unique in the Historian server.

   The tag is renamed. If you have renamed using an alias, the **TAG ALIAS** column displays , indicating that the tag now has an alias.

If you have renamed the tag permanently:

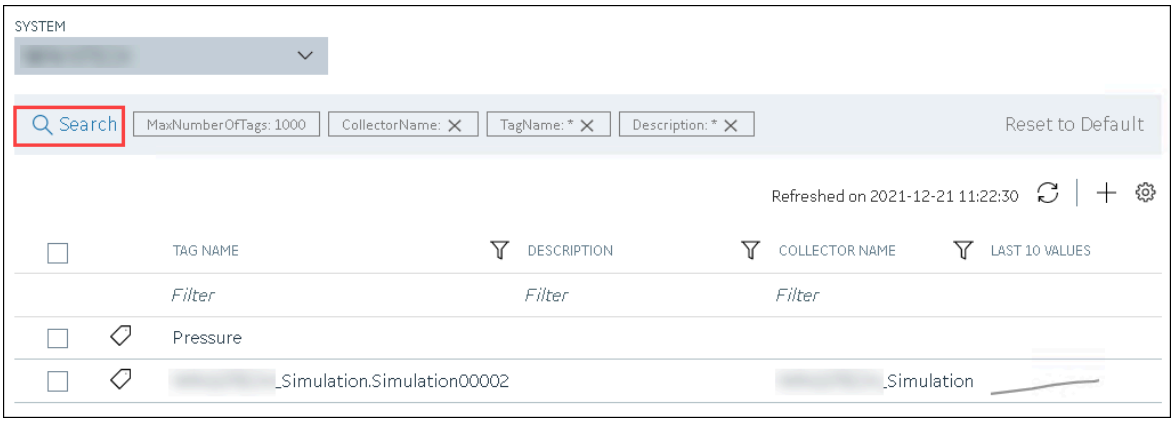
- If the tag is used as a trigger, reassign the trigger.
- Restart the collector instance *(on page 639)*.

## Copy a Tag

If you want to create a tag with the same properties as another one, you can copy it, and then modify the properties as needed. When you copy a tag, the tag alias is captured as well to aid in an audit trail.

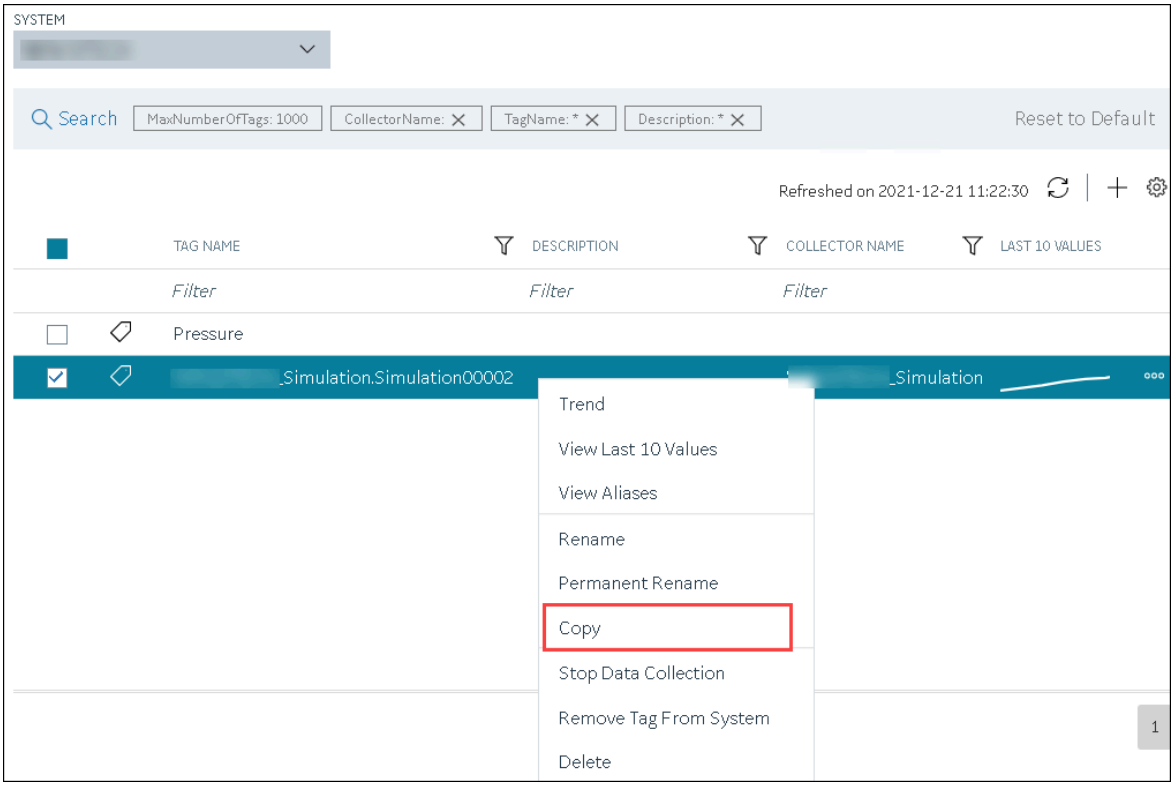
1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
   A list of all the tags appears.

3. If you want to narrow down the search results, select **Search**.




Enter the search criteria, and then select **Search**. You can add more attributes by selecting **Add Attribute**. You can enter a name or a value partially or use the wildcard character asterisk (*). The list of tags are filtered based on the search criteria.

4. Right-click the tag that you want to copy (or select ⋯), and then select **Copy**.




The **Copy Tag: <tag name>** window appears.

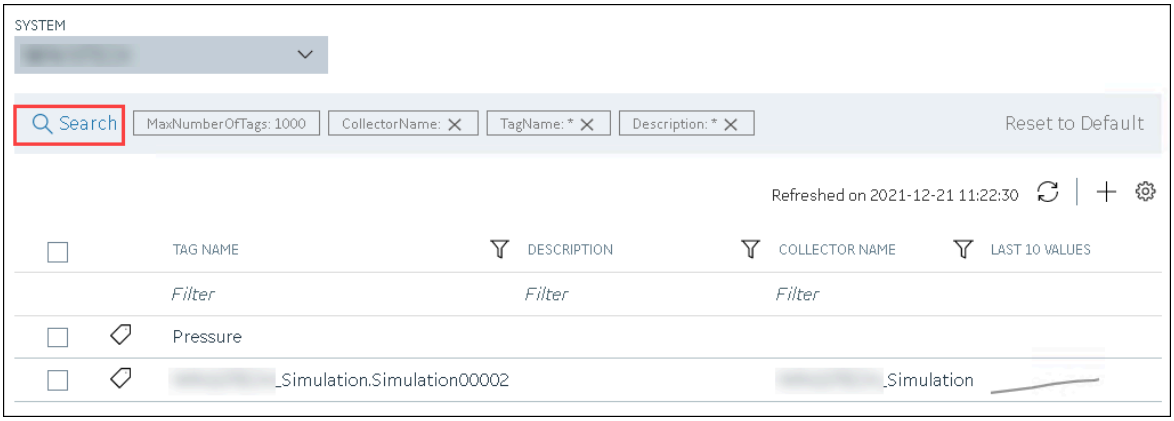
5. In the **NEW TAG NAME** field, provide a name for the tag. A value is required and must be unique for the Historian system.

6. Select **Copy**.
   The tag is copied, inheriting the properties of the original tag. In addition, data collection begins for the tag.

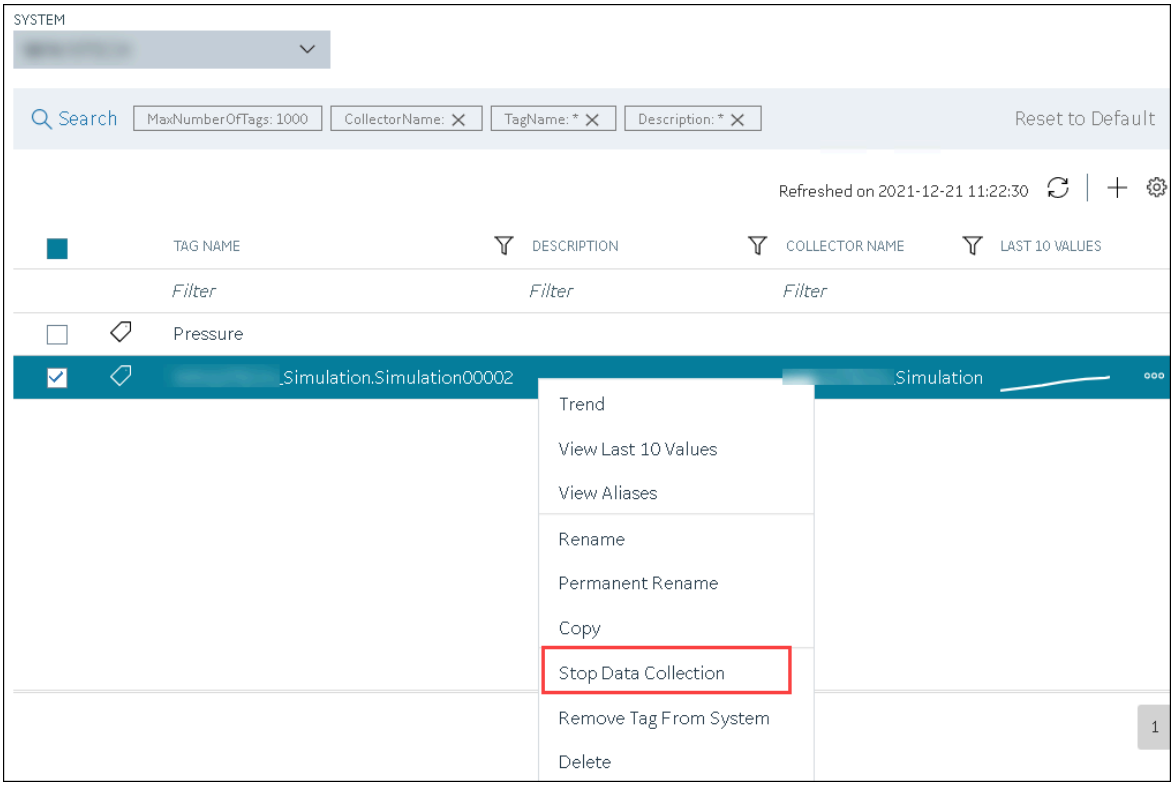
## Stop the Data Collection of a Tag

If you want to stop using a tag for a while, you can stop the data collection of the tag, which allows you to resume the data collection later. If, however, you no longer want to use the tag or its data, you can remove it from the system *(on page 704)* or delete it *(on page 705)*.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
   A list of all the tags appears.
3. If you want to narrow down the search results, select **Search**.




   Enter the search criteria, and then select **Search**. You can add more attributes by selecting **Add Attribute**. You can enter a name or a value partially or use the wildcard character asterisk (*).
   The list of tags are filtered based on the search criteria.
4. Right-click the tag for which you want to stop data collection (or select ), and then select **Stop Data Collection**.




A message appears, asking you to confirm that you want to stop the data collection for the tag.

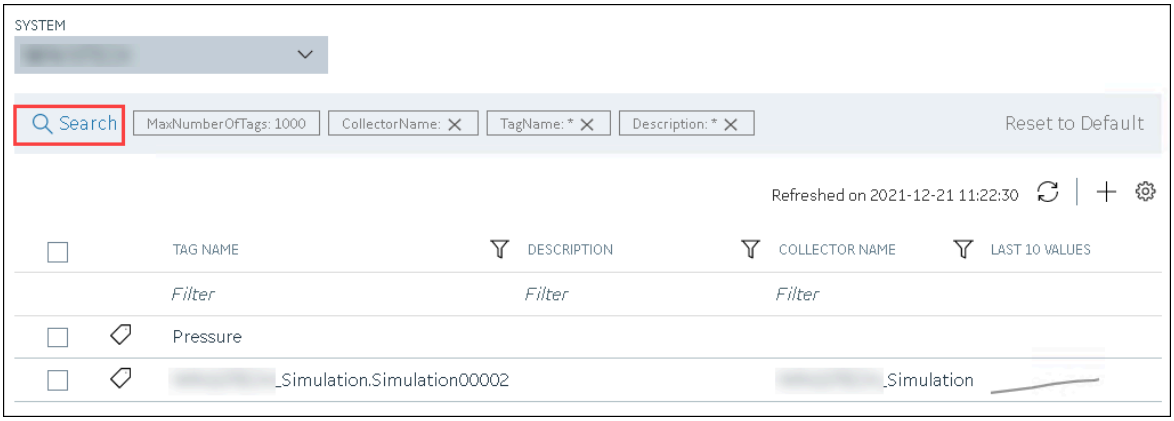
5. Select **Stop**.

   Data collection is stopped for the tag.

## Start the Data Collection of a Tag

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.

   A list of all the tags appears.
3. If you want to narrow down the search results, select **Search**.



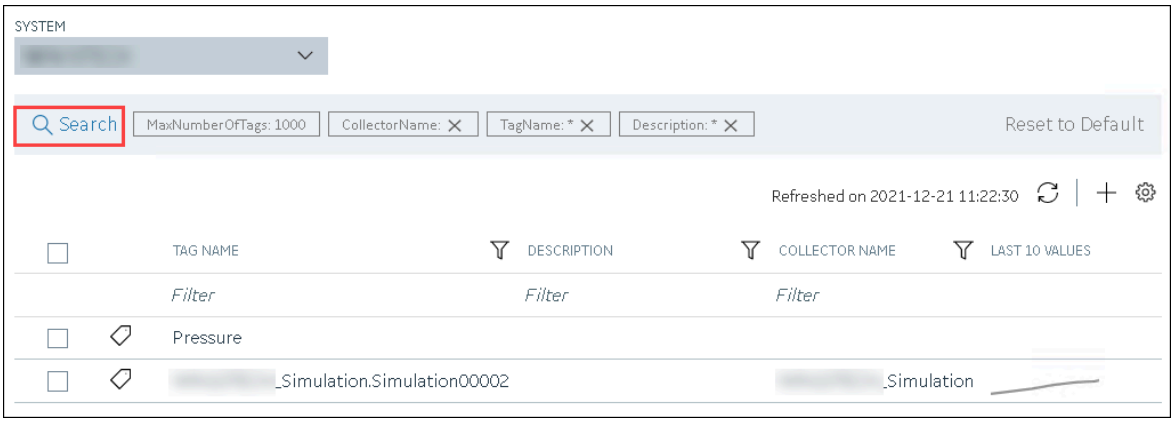
Enter the search criteria, and then select **Search**. You can add more attributes by selecting **Add Attribute**. You can enter a name or a value partially or use the wildcard character asterisk (*). The list of tags are filtered based on the search criteria.

4. Right-click the tag for which you want to start data collection (or select ⋯), and then select **Start Data Collection**.
   A message appears, asking you to confirm that you want to start the data collection for the tag.
5. Select **Start**.
   Data collection is started for the tag.

## Remove a Tag from a System

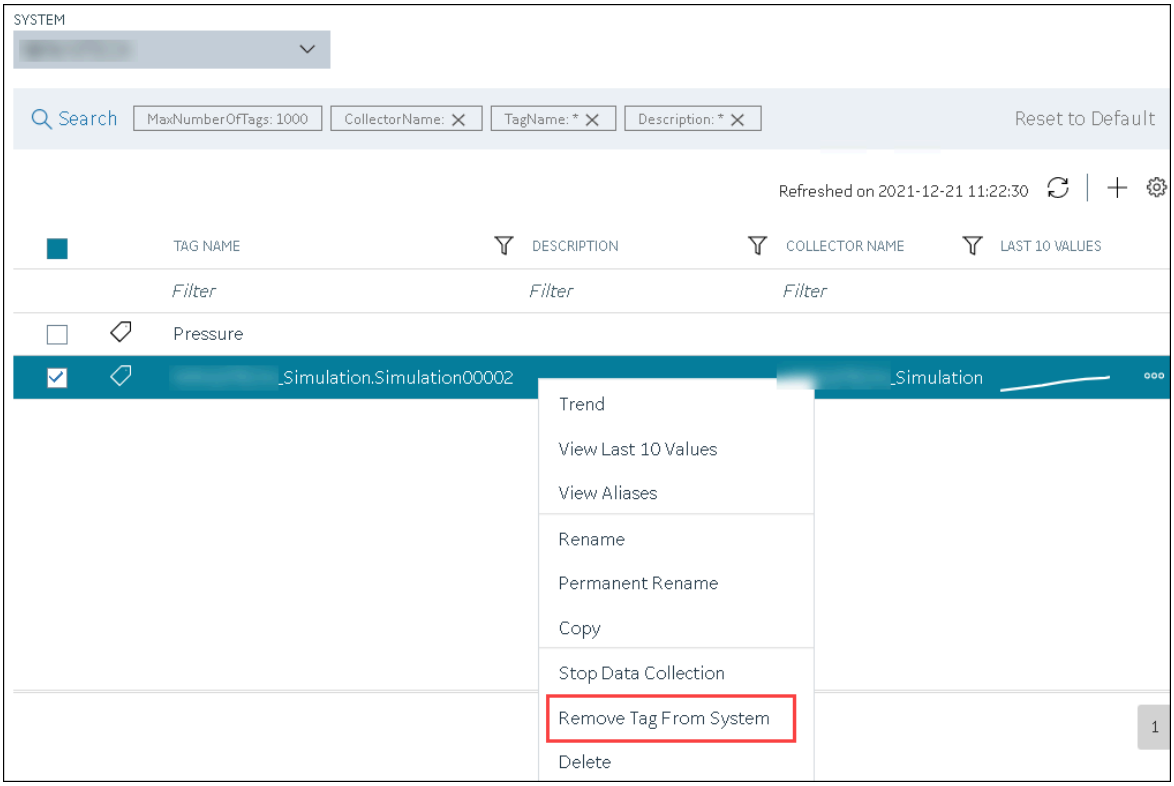
When you remove a tag from a system, the tag and its data will still be available. Therefore, you cannot create a tag with the same name. If, however, you no longer need the tag or its data, you can delete it *(on page 705)*. Or, you can choose to stop the data collection *(on page 702)* for the tag, which allows you to resume the data collection *(on page 703)* later.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
   A list of all the tags appears.
3. If you want to narrow down the search results, select **Search**.




Enter the search criteria, and then select **Search**. You can add more attributes by selecting **Add Attribute**. You can enter a name or a value partially or use the wildcard character asterisk (*). The list of tags are filtered based on the search criteria.

4. Right-click the tag that you want to remove (or select ⋯), and then select **Remove Tag From System**.




A message appears, asking you to confirm that you want to remove the tag from the system.

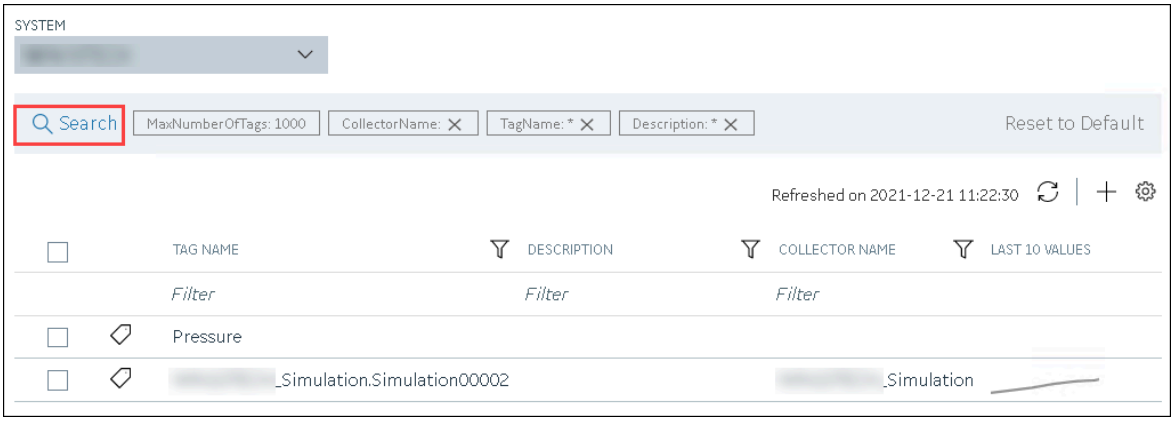
5. Select **Remove**.
   The tag is removed from the system.

## Delete a Tag

If the tag that you want to delete is associated with a variable in a Historian model, remove the mapping between the variable and the tag.

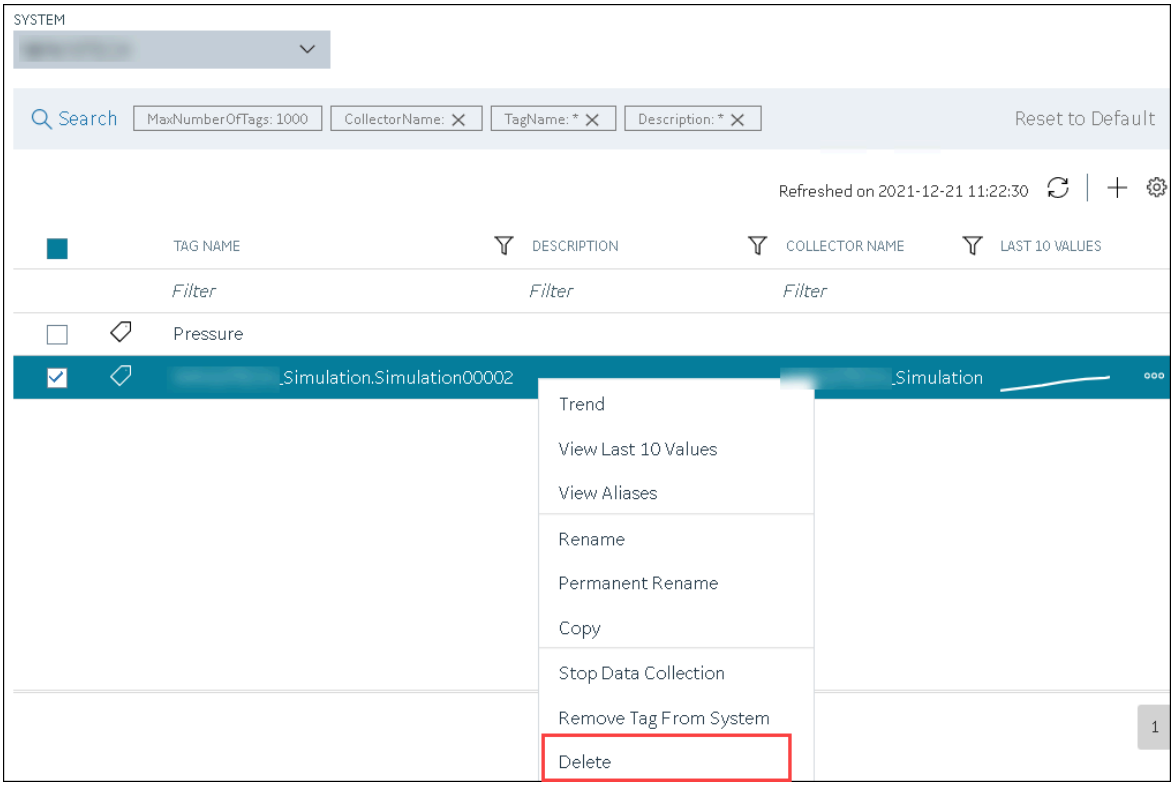
When you delete a tag, it is deleted from Historian, and the tag data will no longer be available. If, however, you want the tag data to be available, instead of deleting the tag, remove it from the system *(on page 704)*. Or, you can choose to stop the data collection *(on page 702)* for the tag, which allows you to resume the data collection *(on page 703)* later.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
   A list of all the tags appears.
3. If you want to narrow down the search results, select **Search**.




Enter the search criteria, and then select **Search**. You can add more attributes by selecting **Add Attribute**. You can enter a name or a value partially or use the wildcard character asterisk (*). The list of tags are filtered based on the search criteria.

4. Right-click the tag that you want to delete (or select ⋯), and then select **Delete**.




A message appears, asking you to confirm that you want to delete the tag.

5. Select **Delete**.

The tag is deleted.

# Managing Enumerated Sets

## About Enumerated Sets

An enumerated data set provides an enhanced way of displaying data. It enables you to retrieve numeric data as string state values. You can use the string values in reports, charts, etc.

An enumerated set contains several states. A state is the number-string value pair in a set. It contains a set of numeric values and their corresponding string values. You can define an enumerated set for a single value or a range of values. These state values are defined for data states stored in Data Archiver. Data is retrieved using the value of the state. You have to define state values within a set to assign enumerated values.

**Table 44. Example of a Single-Value Enumerated Set**

| State Name | State Value |
|---|---|
| Manual | 0 |
| Automatic | 1 |

**Table 45. Example of a Range-of-Values Enumerated Set:**

| State Name | State Value |
|---|---|
| ON | 0 to 100 |
| OFF | 101 to 200 |

State names can be duplicated. If duplicated states exist, take precautions to avoid unpredictable results. For example, a tag is associated with an enumerated set defined as follows.

| State Name | State Value |
|---|---|
| 0 | Open |
| 1 | Close |
| 2 | Close |
| 2 | Open |

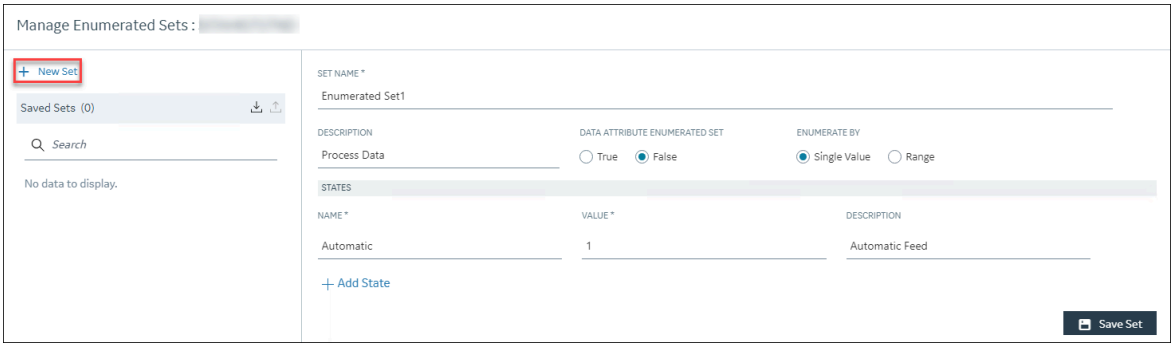
The server will return unpredictable results due to the State Name duplication for an input of 2.

**Note:**
You cannot assign an enumerated set to an array tag.

## Create an Enumerated Set

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
3. Select , and then select **Manage Enumerated Sets**.
   The **Manage Enumerated Sets: <system name>** window appears.

4. Select **New Set**, and enter values as described in the following table.
   The fields to create enumerated sets appear.

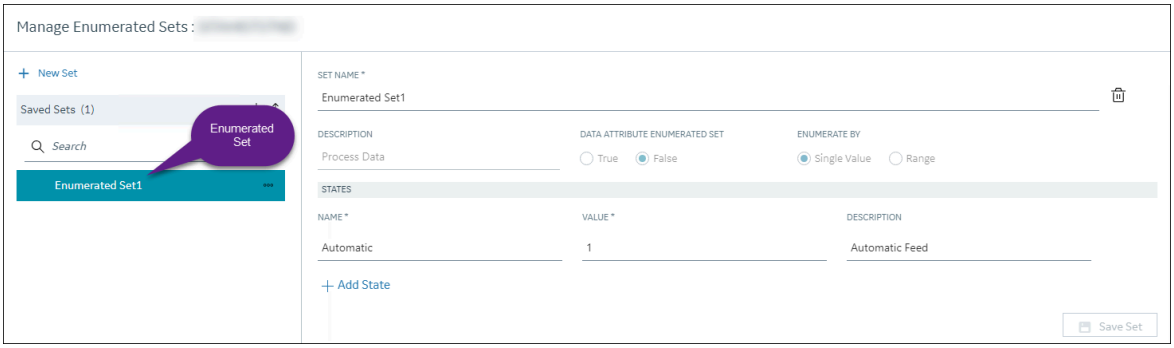



| Field | Description |
|---|---|
| **SET NAME** | Enter a name for the set. |
| **DESCRIPTION** | Enter a description for the set. |
| **DATA ATTRIBUTE ENUMERATED SET** | This is not applicable if you are creating an enumerated set. Set this to **False**. |

| Field | Description |
|---|---|
| **ENUMERATE BY** | Specify whether you want to define a single value or a range of values.<br><br>A single value is best used with integer values because they match exactly. A range of values can be used with floating point values because they may not match exactly due to rounding. |
| **NAME** | Enter a name for the state. |
| **VALUE** | Enter a numeric value for the state; string values such as on/off are not supported. This field appears only if you select **Single Value** in the **ENUMERATE BY** field. |
| **START RANGE** and **END RANGE** | Enter the start and end values of the range. Enter only numeric values; string values are not supported. These fields appear only if you select **Range** in the **ENUMERATE BY** field.<br><br>If you want to assign ON for values 0-100, enter ON in the **NAME** field, and enter 0 and 100 in the **START RANGE** and **END RANGE** fields. |
| **DESCRIPTION** (under **STATES**) | Enter a description for the state. |

5. For each state that you want to create, select **Add State**, and repeat the previous steps.
6. Select **Save Set**.
7. If needed, add more sets, and then select **Done**.

   The enumerated set is created.




Assign the enumerated set to a tag *(on page 710)*.

# Assign an Enumerated Set to a Tag

- Create the tag *(on page 357)* for which you want to use an enumerated set.
- Create the enumerated set *(on page 708)* that you want to assign to a tag.

If you assign an enumerated set to a tag, when you retrieve the tag data, instead of the actual values, the corresponding state names appear.

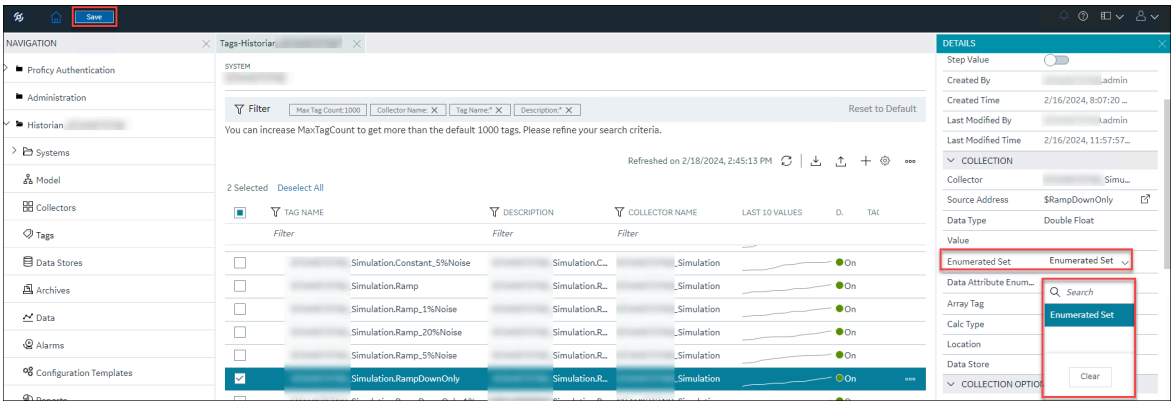
**Example of an Enumerated Set**

Suppose you have created the following states in an enumerated set:

| State Name | State Range |
|---|---|
| ON | 0 to 100 |
| OFF | 101 to 200 |

Suppose you assign the enumerated set to a tag. If the tag values are, say, 50, 100, 75, and 104, when you retrieve the tag values, ON, ON, ON, and OFF, respectively, are retrieved instead of the actual tag values.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
3. Select the tag to which you want to assign an enumerated set.
   The details of the tag appear in the **DETAILS** section.
4. Under **Collection**, select **Enumerated Set**.




5. In the upper-left corner of the page, select **Save**.
   The selected enumerated set is assigned to the tag.

# Export an Enumerated Set

This topics describes how to export enumerated sets as a CSV file. You can export enumerated sets from Configuration Hub as a CSV file add/modify sets in bulk and then import them. You can also import the CSV file that was exported using Excel Add-in.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
3. Select , and then select **Manage Enumerated Sets**.
   The **Manage Enumerated Sets: <system name>** window appears.
4. In **Saved Sets**, select .
5. Enter a name for the file and save it in a location as needed.

## Import an Enumerated Set

This topic describes how to import enumerated sets into Configuration Hub from a CSV file. You can either export enumerated sets and their details as a CSV file and edit them, and then import them back, or you can import enumerated sets and their details from another CSV file that you created externally, like excel add-in.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
3. Select , and then select **Manage Enumerated Sets**.
   The **Manage Enumerated Sets: <system name>** window appears.
4. In **Saved Sets**, select .
   The **Import Saved Sets** page appears.
5. Select **Choose File**, and then select the CSV file as need.

   **Note:**
   If you import any irrelevant CSV file, the import will fail and you will be notified.

6. Select **Import**.
   The enumerated sets and their details are imported.
7. In the **Manage Enumerated Sets: <system name>**, select **Done**.

## Rename Enumerated Set

You can rename an enumerated set if needed.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
3. Select ⋯, and then select **Manage Enumerated Sets**.
   The **Manage Enumerated Sets: <system name>** window appears.
4. Select the enumerated set that you want to rename.
   Details corresponding to the selected data set are displayed.
5. In **SET NAME**, rename the name of the set.
6. Select **Save Set**.
   The set name is renamed.
7. Click **Done**.
   The changes are saved.

## Delete an Enumerated Set

This topic describes how to delete an enumerated set.

> **Note:**
> You cannot delete an enumerated set if it is assigned to a tag.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
3. Select ⋯, and then select **Manage Enumerated Sets**.
   The **Manage Enumerated Sets: <system name>** window appears.
4. Select the enumerated set that you want to delete, or select ⋯, and then select **Delete Set**. Or, select 🗑 next to the Set name.
   A message appears, asking you to confirm that you want to delete the set.
5. Select **Delete**.
   The enumerated set is deleted.

# Managing Data Attribute Enumerated Set

## About Data Attribute Enumerated Set

A data attribute enumerated set enables you to specify whether you want to define a custom text for the sub-quality status and view it in the **Attributes** column instead of the value when you retrieve the tag data. For example, if your tag is receiving bit positions indicating the sub-quality status (that are, 1 for Old data, 2 for Bad data, and 30 for Good data), you can map the bit position to a custom text and assign it to a

set. So, when you view the actual values of the tag, the **Attributes** column will display the mapped custom text instead of the integer (the bit positions). For example, if you mapped a custom text name "good-16" to a value of 16, and the bit position of an incoming value is 16, it will display "good-16" in the **Attributes** column instead of 16.

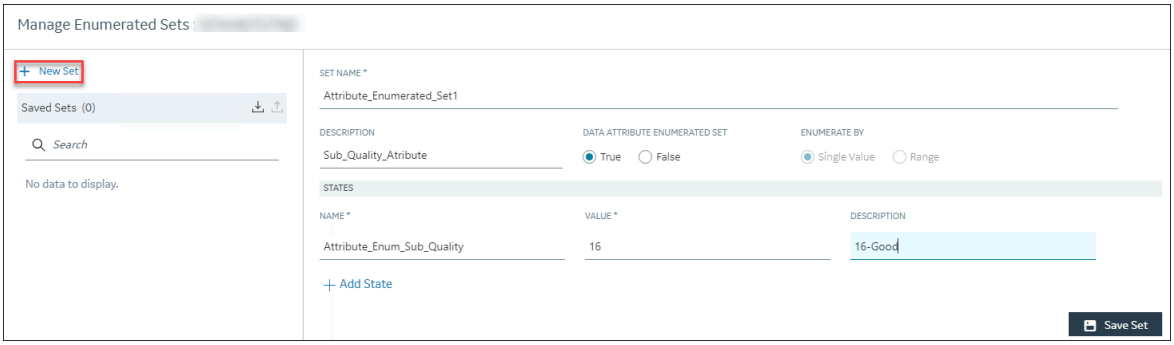
**Table 46. Example of an Attribute Enumerated Set**

| State Name | State Value |
|---|---|
| Uninitiated | 0 |
| Old | 1 |
| Bad | 2 |
| Not in Service | 10 |
| Good | 30 |

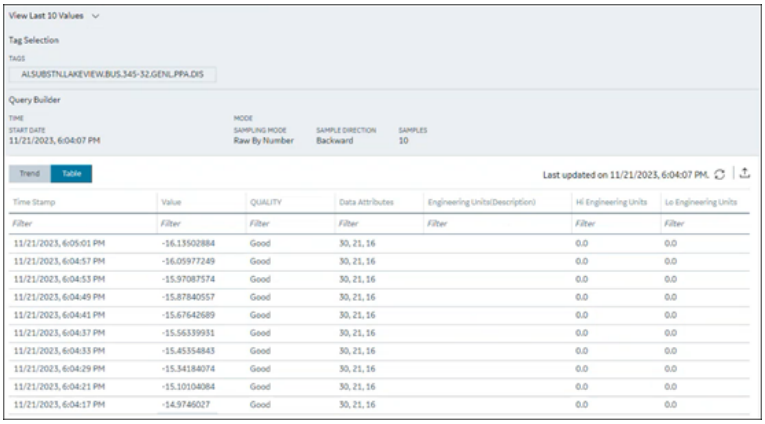
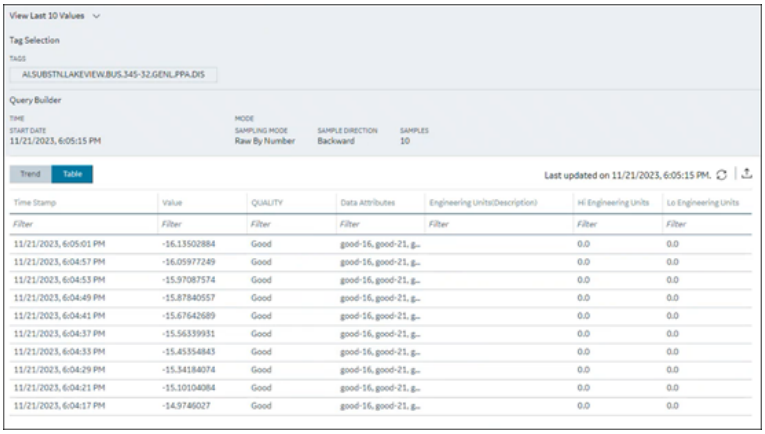
## Create a Data Attribute Enumerated Set

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
3. Select ⋯, and then select **Manage Enumerated Sets**.
   The **Manage Enumerated Sets: <system name>** window appears.

4. Select **New Set**, and enter values as described in the following table.
   The fields to create enumerated sets appear.

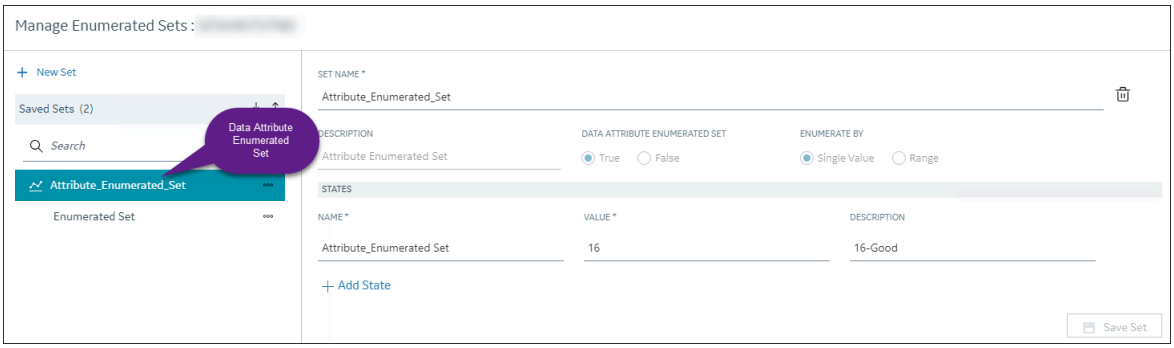



| Field | Description |
|---|---|
| **SET NAME** | Enter a name for the set. |
| **DESCRIPTION** | Enter a description for the set. |
| **DATA ATTRIBUTE ENUMERATED SET** | Specify whether you want to define a custom text for the sub-quality status and view it in the **Attributes** column instead of the value when you retrieve the tag data. For example, if your tag is receiving bit positions indicating the sub-quality status (that are, 1 for Old data, 2 for Bad data, and 30 for Good data), you can map the bit position to a custom text and assign it to a set. So, when you view the actual values of the tag, the **Attributes** column will display the mapped custom text instead of the integer (the bit positions). For example, if you mapped a custom text name "good-16" to a value of 16, and the bit position of an incoming value is 16, it will display "good-16" in the **Attributes** column instead of 16.<br><br>To create an attribute enumerated set, you must set **DATA ATTRIBUTES ENUMERATED SET** to **True**. If you choose this option, the **ENUMERATED BY** field is set to **Single Value** and becomes read-only.<br><br>**Note:**<br>It is not recommended to modify the created Data Attribute Enumerated set in Historian Administrator. If you modify it using Historian Administrator, the **DATA ATTRIBUTE ENUMERATED SET** field will be changed to **False** in Configuration Hub. |

| Field | Description |
|---|---|
| | Example- Before applying the Data Attribute Enumerated Set<br>  <br>Example- After applying the Data Attribute Enumerated Set<br>  |
| **ENUMERATE BY** | Specify whether you want to define a single value or a range of values.<br><br>For data attribute enumerated set, single value is selected by default and disabled. |
| **NAME** | Enter a name for the state. |
| **VALUE** | Enter a numeric value for the state; string values such as on/off are not supported. This field appears only if you select **Single Value** in the **ENUMERATE BY** field. |
| **DESCRIPTION** (under **STATES**) | Enter a description for the state. |

5. For each state that you want to create, select **Add State**, and repeat the previous steps.

6. Select **Save Set**.
7. If needed, add more sets, and then select **Done**.
   The data attribute enumerated set is created.




## Assign a Data Attribute Enumerated Set to a Tag

- Create the tag *(on page 357)* for which you want to use an enumerated set.
- Create a data attribute enumerated set that you want to assign to a tag.

If you assign a data attribute enumerated set to a tag, when you retrieve the tag data, the corresponding custom text that was mapped appears.

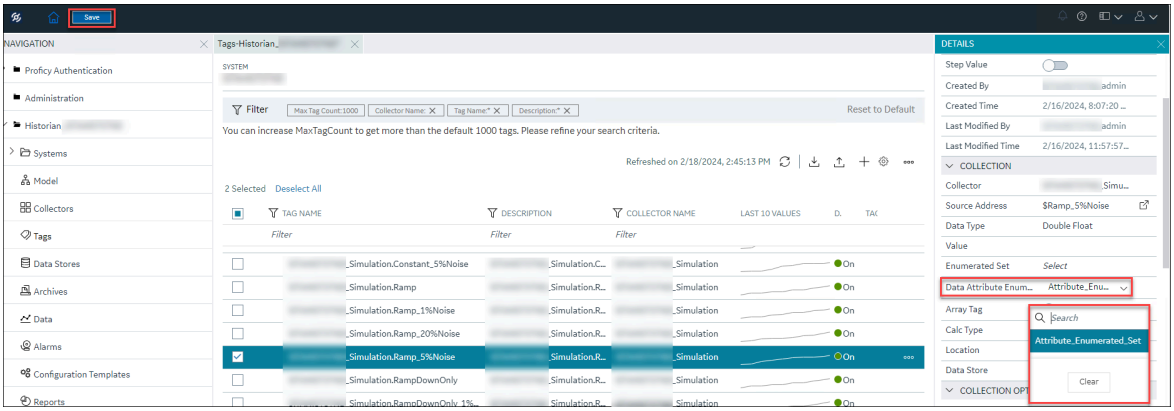
**Example of a Data Attribute Enumerated Set**

Suppose you have created the following states in a data attribute enumerated set:

| State Name | State Value |
|---|---|
| Old | 1 |
| Bad | 2 |
| Good | 30 |

Suppose you assign the data attribute enumerated set to a tag. If the tag's quality status are, say, 30, 30, 1, and 2, when you retrieve the tag values, Good, Good, Old, and Bad, respectively, are retrieved instead of the actual quality status values in the **Attributes** column.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
3. Select the tag to which you want to assign an enumerated set.
   The details of the tag appear in the **DETAILS** section.
4. Under **Collection**, select **Data Attribute Enumerated Set**.




5. In the upper-left corner of the page, select **Save**.
   The selected data attribute enumerated set is assigned to the tag.

## Export Data Attribute Enumerated Sets

This topics describes how to export data attribute enumerated sets as a CSV file. You can export data attribute enumerated sets from Configuration Hub as a CSV file add/modify sets in bulk and then import them. You can also import the CSV file that was exported using Excel Add-in.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
3. Select ⋯, and then select **Manage Enumerated Sets**.
   The **Manage Enumerated Sets: <system name>** window appears.
4. In **Saved Sets**, select ⬆.
5. Enter a name for the file and save it in a location as needed.

## Import Data Attribute Enumerated Sets

This topic describes how to import data attribute enumerated sets into Configuration Hub from a CSV file. You can either export data attribute enumerated sets and their details as a CSV file and edit them, and then import them back, or you can import data attribute enumerated sets and their details from another CSV file that you created externally, like Excel Add-in.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
3. Select ⋯, and then select **Manage Enumerated Sets**.
   The **Manage Enumerated Sets: <system name>** window appears.

4. In **Saved Sets**, select .
   The **Import Saved Sets** page appears.
5. Select **Choose File**, and then select the CSV file as need.

   **Note:**
   If you import any irrelevant CSV file, the import will fail and you will be notified.

6. Select **Import**.
   The data attribute enumerated sets and their details are imported.
7. In the **Manage Enumerated Sets: <system name>**, select **Done**.

## Rename a Data Attribute Enumerated Set

You can rename a data attribute enumerated set if needed.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
3. Select , and then select **Manage Enumerated Sets**.
   The **Manage Enumerated Sets: <system name>** window appears.
4. Select the data attribute enumerated set that you need to rename.
   Details corresponding to the selected data set are displayed.
5. In **SET NAME**, rename the name of the set.
6. Select **Save Set**.
   The set name is updated.
7. Click **Done**.
   The changes are saved.

## Delete a Data Attribute Enumerated Set

This topic describes how to delete a data attribute enumerated set.

**Note:**
You cannot delete a data attribute enumerated set if it is assigned to a tag.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
3. Select , and then select **Manage Enumerated Sets**.
   The **Manage Enumerated Sets: <system name>** window appears.

4. Select the data attribute enumerated set that you want to delete, or select ⋯, and then select **Delete Set**. Or, select 🗑 next to the Set name.
   A message appears, asking you to confirm that you want to delete the set.
5. Select **Delete**.
   The data attribute enumerated set is deleted.

# Managing User-Defined Data Types

## About User-Defined Data Types

Sometimes, a single tag cannot store all the required details of a parameter. For example, if you want to store the name, address, and phone number of the manufacturer of a machine, it may not be feasible to store all these details in a single tag, which uses a single data type. In such cases, you can create a user-defined data type (UDT), which includes one or more fields, and then apply that type to Historian tags. Each of these fields in a UDT can contain a different data type based on your requirement.

The following conditions apply when working with a UDT:

- You must have appropriate security permissions to create, modify, and delete a UDT. The type can have its own Administrator security group.
- You cannot create an array tag that uses a UDT.
- UDTs cannot have fields of Scaled or FixedString data types.
- Scaling, collector compression, and archive compression do not apply to UDT tags.
- You cannot associate an enumerated set with a UDT tag.
- A UDT supports maximum 100 fields.

## Create User-Defined Data Types

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
3. Select ⋯, and then select **Manage User-Defined Types**.
   The **Manage User-Defined Types: <system name>** window appears.
4. Select **New Type**, and enter values as described in the following table.

| Field | Description |
|---|---|
| **TYPE NAME** | Enter a name for the user-defined data type (UDT). |
| **DESCRIPTION** | Enter a description for the UDT. |

| Field | Description |
|---|---|
| **ADMINISTER GROUP** | Specify the Windows Security Group that you want to assign to the UDT. |
| **DATA QUALITY** | Select one of the following options:<br>◦ **Store Individual Quality**: Select this option to store field-level quality. Storing individual qualities consumes more disk space.<br>◦ **Store Master Field Quality**: Select this option if you want to assign the same quality as the master field to all the other fields in the UDT. If you select this option, the data sample will have a single quality similar to how an array tag works. |
| **MASTER** | Select the radio button corresponding to the field that you want to set as master. When you do so, the data type of this field is used for all the remaining fields in the UDT as well. Only one field can be the master in a UDT. The **MASTER** column appears only if you select **Store Master Field Quality** under **DATA QUALITY**. |
| **NAME** | Enter a name for the field. |
| **DESCRIPTION** (under **FIELD**) | Enter a description for the field. |
| **DATA TYPE** | Select the data type for the field. If you select **Store Master Field** under **DATA QUALITY**, and if you set this field as master, the data type of this field will be applied to the remaining fields in the UDT as well. |

5. For each field that you want to create, select **Add Field**, and provide the required details.
6. Select **Save Type**.
7. If needed, add more UDTs, and then select **Done**.
   The UDT is created.

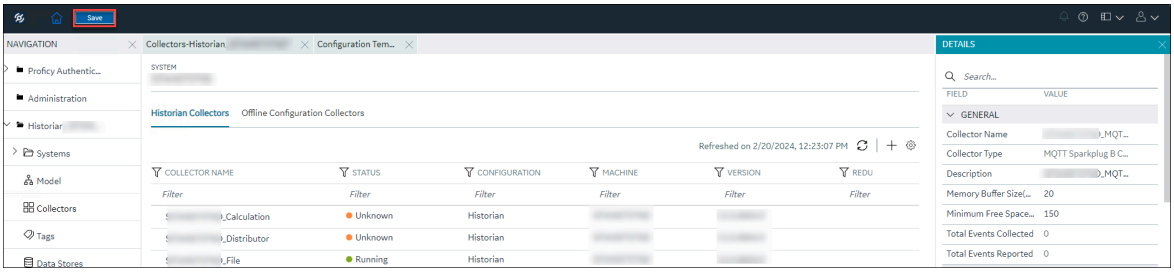
Assign the UDT to a tag *(on page 720)*.

## Assign a User-Defined Data Type to a Tag

- Create the tag *(on page 357)* for which you want to assign a user-defined data type (UDT).
- Create the UDT *(on page 719)* that you want to assign to the tag.

When you assign a UDT to a tag, the tag can collect data of various data types.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
3. Select the tag for which you want to assign a UDT.
   The details of the tag appear in the **DETAILS** section.
4. Under **Collection**, in the **User Defined Type Name** field, select the UDT that you want to assign.
5. In the upper-left corner of the page, select **Save**.




   The UDT is assigned to the tag.

## Export User-defined Types

This topics describes how to export user-defined types as a CSV file. You can add/modify them in bulk, and then import them. You can also import the CSV file that was exported using Excel Add-in.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
3. Select , and then select **Manage User-Defined Types**.
   The **Manage User-Defined Types: <system name>** window appears.
4. In **Saved Types**, select .
5. Enter a name for the file and save it in a location as needed.

## Import User-defined Types

This topic describes how to import user-defined types into Configuration Hub from a CSV file. You can either export user-defined types and their details as a CSV file and edit them, and then import them back, or you can import user-defined types and their details from another CSV file that you created externally, like excel add-in.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.

3. Select , and then select **Manage User-Defined Types**.
   The **Manage User-Defined Types: <system name>** window appears.
4. In **Saved Types**, select .
   The **Import Saved Types** window appears.
5. Select **Choose File**, and then select the CSV file as need.

   **Note:**
   If you import any irrelevant CSV file, the import will fail and you will be notified.

6. Select **Import**.
   The user-defined types and their details are imported.
7. In the **Manage User-Defined Types: <system name>**, select **Done**.

## Rename User-defined Types

You can rename a user-defined type as needed.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
3. Select , and then select **Manage User-Defined Types**.
   The **Manage User-Defined Types: <system name>** window appears.
4. In **Saved Types**, select the saved type as needed.
   Details corresponding to the selected user-defined type are displayed.
5. Change the user-defined type name as needed.
6. Select **Save Type**.
   The user-defined type name is renamed.
7. Click **Done**.
   The changes are saved.

## Delete a User-Defined Data Type

You cannot delete a user-defined data type (UDT) if it is assigned to a tag.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
3. Select , and then select **Manage User-Defined Types**.
   The **Manage User-Defined Types: <system name>** window appears.

4. Select the UDT that you want to delete, select ooo, and then select **Delete**. Or, select next to the UDT name.
   A message appears, asking you to confirm that you want to delete the UDT.
5. Select **Delete**.
   The UDT is deleted.

# Managing Archives

## About Archives

Historian archives are data files, each of which contains data gathered from all data sources during a specific period of time.

### Types of Archive Files:

- ***machine name*_Config.ihc:** Contains information about the archiver, tag configuration, and collector configuration.
- ***machine name*_ArchiveXXX.iha:** Contains tag data, where x is a number indicating the place of the file in a time-based sequence.

### Creation of Archive Files Automatically

Archive files grow to a user-configured maximum size as data is recorded by the server. When data starts loading into an archive file, Historian will automatically create a new blank archive file. When the current archive file becomes full, Historian will immediately serve data to the newly created archive file. This significantly reduces archive creation and transition time.

If, however, the option to automatically create archive files is not enabled, you must create an archive file manually *(on page 868)*.

**Note:**
If the option to automatically create an archive is not enabled and you do not create a new archive manually, or if the available disk space is less than the required amount of free disk space, a new archive file will not be created.

### Overriding Old Archive Files

If you enable the **Overwrite Old Archives** option, the system replaces the oldest archived data with new data when the latest archive default size has been reached. Since this action deletes historical data,

exercise caution in using this feature. Be sure that you have a backup of the archive so that you can restore it later. Best practice is to create an additional archive to prevent premature loss of data due to overwriting. For example, if you want to save 12 months of data into 12 archives, create 13 archives.

During archiver startup and every 60 seconds while the server is running, Historian verifies that you have configured enough free disk space to save the archives, buffer files, and log files. If there is insufficient disk space, the Data Archiver shuts down and a message is logged into the log file. By default, you can view the Historian archiver log file in `C:\Historian Data\LogFiles`.

```
[03/03/10 15:28:41.398] Insufficient space available in [d:\Historian\Archives\]
        [03/03/10 15:28:41.399] The server requires a minimum of [5000 MB] to continue
        [03/03/10 15:28:41.679] USER: DataArchiver TOPIC: ServiceControl MSG: DataArchiver(DataArchiver)
        Archiver shutdown at 03/03/10 15:28:41.653
        [03/03/10 15:28:41.807] DataArchiver Service Stopped.
        [03/03/10 15:28:41.809] [d:\Historian\LogFiles\DataArchiver-34.log] Closed.
```

## Guidelines for Setting Archive Size

Since archived data files can become quite large, you must adjust system parameters carefully to limit data collection to meaningful data only and thus minimize the required size of system storage. You can allocate up to 256 GB per archive.

For each archive, you need approximately 1MB of archive space for every 1000 tags to store tag information. Archive size is a function of the rate at which you archive data and the time period you want the archive to cover. A typical user wants the archive to cover a time period of, say, 30 days.

The following factors affect the rate at which you archive data:

- Number of tags
- Polling frequency of each tag
- Compression settings
- Data types

Based on these parameters, the archive size is calculated as follows:

$$\#Tags \times \frac{Values}{Tag} \times \frac{Tags}{Second} \times \%PassComp \times \frac{Bytes}{Value} \times \frac{Seconds}{Hour} \times \frac{Hours}{Day} \times \frac{MB}{Bytes} = \frac{MB}{Day}$$

### Calculating Archive Size

Suppose you want to store data, and you have the following parameters:

- Number of tags: 5000
- Polling rate: 1 value/5 seconds
- Pass compression: 5%.

  Pass compression is the number of data values archived relative to the number of values read.

- Bytes/value: 4
- Duration: 30 days

Based on the preceding formula, for the given parameters, the archive size is calculated as follows:

$$5000 \times \frac{1}{1} \times \frac{1}{5} \times \frac{5}{100} \times \frac{4}{1} \times \frac{3600}{1} \times \frac{24}{1} \times \frac{1}{1024 \times 1024} \times 30 = 494 \frac{MB}{Month}$$

The calculation shows that a file size of 500 MB is adequate for archiving one month of data for this application.

Therefore, we recommend that you set the default archive size to 500 MB for systems with 1000 tags or more. If you believe the computed size is too large for your application, you can modify parameters as follows:

- Decrease the polling frequency.
- Increase compression deadband, reducing the pass percentage.
- Reduce the number of tags.
- Add more disk capacity to your computer.

### Archive Size Calculator

An archive size calculator tool is available to estimate archive size and collector compression based on a tag that has already been configured or based on your inputs. Log on to http://digitalsupport.ge.com to download this tool and other GE Intelligent Platforms freeware product solutions.

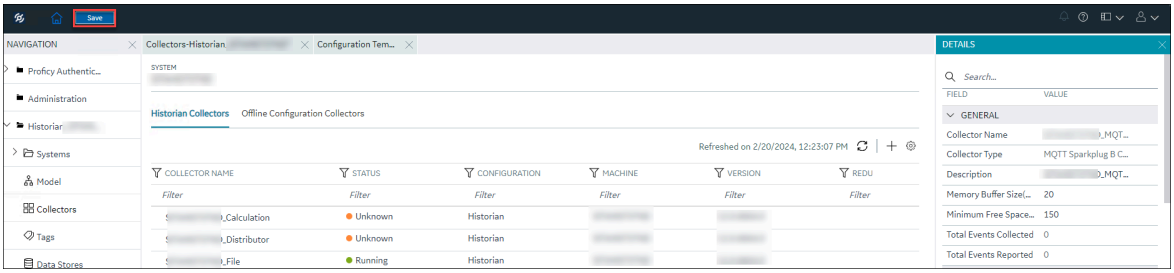
## Access an Archive

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Archives**. The **Archives** section appears. By default, only the archives of the default data store appear. If you want to access archives from another data store, select it in the **DATA STORE** field.

3. Select the archive whose details you want to access.
   The **DETAILS** section displays the following information of the archive.

| Field | Description |
|---|---|
| **Status** | Identifies the status of the archive. Contains one of the following values:<br>◦ **Current**: The archive is actively accepting data.<br>◦ **Active**: The archive contains data but is not currently accepting data.<br>◦ **Empty**: The archive was created but has never accepted data. |
| **Start Time** | The time at which writing data to the archive has begun. |
| **End Time** | The time at which writing data to the archive has ended. |
| **Last Backup** | The time at which the archive has been backed up last. |
| **Backup User** | The user who created the last backup of the archive. |
| **File Name** | The name and folder path of the archive file. |
| **File Size (MB)** | The size of the archive file. |
| **File Attribute** | Indicates whether the archive file is read-only or read/write. |

4. If needed, change the values in the **Filename** and **File Attribute** fields. You cannot, however, change the file attribute for the current archive.
5. In the upper-left corner of the page, select **Save**.




The archive is modified.

## Create Archives Automatically

Historian can automatically create archives for you if the current archive reaches a specific size or after a specific duration. This topic describes how to set these options. You can, however, choose to create archives manually *(on page 728)*.

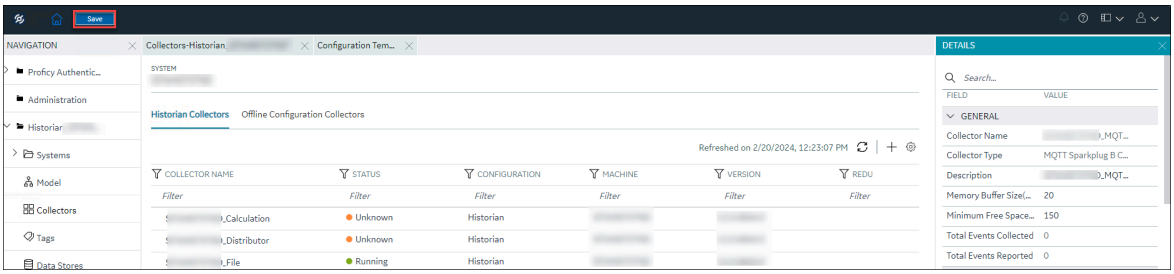
1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Archives**.
   The **Data Stores** section appears.
3. Select the data store in which you want to create archives automatically.
   The **DETAILS** section displays the details of the data store.
4. Under **Archive Creation**, enter values as described in the following table.

| Field | Description |
|---|---|
| **Automatically Create Archives** | Switch the toggle on to indicate that you want Historian to create archive files automatically when the current one is full. |
| **Create Archive By** | Select whether you want to create a new archive automatically after the current one reaches a specific *size* or after a specific *duration*. This field is enabled only if you switch the **Automatically Create Archives** toggle on.<br><br>Select one of the following options:<br>◦ **Size**: Select this option if you want to create a new archive when the current one reaches a specific size. Specify the size in the **Default Size (MB)** field (which appears only if you select **Size**).<br>◦ **Days** or **Hours**: Select one of these options if you want to create a new archive after a specific duration. Specify the duration in the **Archive Duration** field (which appears only if you select **Days** or **Hours**). |
| **Default Size (MB)** | The default size of an archive after which a new one will be automatically created. This field |

| Field | Description |
|---|---|
| | appears only if you select **Size** in the **Create Archive By** field. |
| **Archive Duration** | The duration after which a new archive will be automatically created. This field appears only if you select **Days** or **Hours** in the **Create Archive By** field. |

If needed, you can switch the **Overwrite Old Archives** toggle on. If you enable this option, the oldest archived data is replaced with the latest one when the latest archive default size is reached. Since this action deletes historical data, exercise caution in using this feature. Be sure that you have a backup of the archive so that you can restore it later. Best practice is to create an additional archive to prevent premature loss of data due to overwriting. For example, if you want to save 12 months of data into 12 archives, create 13 archives.

5. In the upper-left corner of the page, select **Save**.




Archives will now be created automatically based on the criteria you specified.

## Create Archives Manually

This topic describes how to create an archive file manually. You can create multiple archives at the same time. You can, however, choose to create archives automatically *(on page 727)*.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Archives**.
   The **Archives** section appears.
3. In the upper-right corner of the section, select +.
   The **Add Archives** window appears.
4. Enter values as described in the following table.

| Field | Description |
|---|---|
| **ARCHIVE NAME** | Identifies the name of the archive. A value is required. By default, it is in the following format: *<data store name>_<system name>_*-Archive*<number>*. The number is used to name the archives sequentially. You can only add a suffix to the archive name. |
| **DATA STORE** | Select the data store in which you want to create the archives. A value is required. |
| **FILE LOCATION** | Enter the folder path in which you want to store the archives. A value is required. By default, it is `C:\Proficy Historian Data\Archives` |
| **EACH ARCHIVE SIZE** | Enter the size, in MB, that you want to allocate to each archive. A value is required. This field is populated with the value in the **Default Size (MB)** field in the data store (if applicable). |
| **NUMBER OF ARCHIVES** | Enter the number of archives that you want to create. A value is required. The default value is 1. |
| **ALLOCATE SPACE** | Specify the disk space that you want to allocate for archives. A value is required.<br><br>**Important:**<br>If there is insufficient disk space, Data Archiver is shut down and a message is logged into the log file. By default, you can view the log file in `C:\Historian Data\LogFiles`. |

5. Select **Add**.

   The archives are created.

## Back up an Archive

Ensure you have enough disk space.

You must back up archive files periodically to ensure that your data is protected. These backup files contain tag data as well as alarms and events data. You can send these files to a shared network location or to physical media.

Always back up archives before a planned Historian software product upgrade. Use Microsoft® Volume Shadow Copy Service when backing up archive files that are more than 2 GB in size or when backing up more than the last two archives. For more information, refer to Back Up Archives with Volume Shadow Copy Service *(on page 731)*.

This topic describes how to back up an archive manually. You can also back up an archive automatically *(on page 874)*.

**Important points to remember:**

- The .IHC file is automatically backed up when, and only when, you back up the current archive .IHA file. By default, the .IHC backup path is the same as the archives path.
- The .IHC backup file uses the following naming convention: `<system name>_Config-Backup.ihc`. If the default backup path is different from the archives counterpart, the .IHC file is copied to the backup folder with the standard .IHC naming convention: `ComputerName_Config.ihc`.
- In the mirroring system, Client Manager sends a backup message to Data Archiver located on the Client Manager node to which you are connected. The back up, then, happens in the specified location on that node. If that Data Archiver is not running, a NOT_CONNECTED error message appears, and the backup will not happen.
- If you back up an archive more than once, the backup tool will (by default) attempt to use the same name for the backup file and will detect that an archive with the same name already exists. Rename the backup archive file or move the original backup archive file to a different folder.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Archives**. The **Archives** section appears.
3. Right-click the archive that you want to back up (or select ⋯), and then select **Backup Archive**. The **Backup Archives** window appears. The **ARCHIVE NAME** field is disabled and populated. The **BACKUP FILE PATH** field is disabled and populated with the value in the **DEFAULT BACKUP PATH** field of the data store.

4. Select **OK**.
   The archive file is backed up.

## Back up Archives with Volume Shadow Copy Service

Use the Microsoft® Volume Shadow Copy Service to back up large archive files, or if you want to back up more than the last two archives, as it allows you to backup and restore archives reliably and in a short period of time without affecting the data collection.

The Volume Shadow Copy feature is provided by Windows Operating System, and the instructions to use backup and restore vary depending on the backup application that is used in the Windows operating system.

VSS provides fast volume capture of the state of a disk which is called a snapshot or shadow copy. When the snapshot is taken, disk writes are suspended for a brief period of time, typically on the order of milliseconds. After the snapshot, disk writes can resume, but the original state of the files are maintained by a difference file. The difference file allows the state of the original file at the time of the snapshot to be reconstructed. This behavior allows files to be backed up while new data is being written to files.

If you are using `ihArchiveBackup.exe` before the upgrade, your backup will continue to work in the same or similar manner as it did before the upgrade. There is no change in the backup procedure and the Auto Recovery Backup Files option remains unchanged.

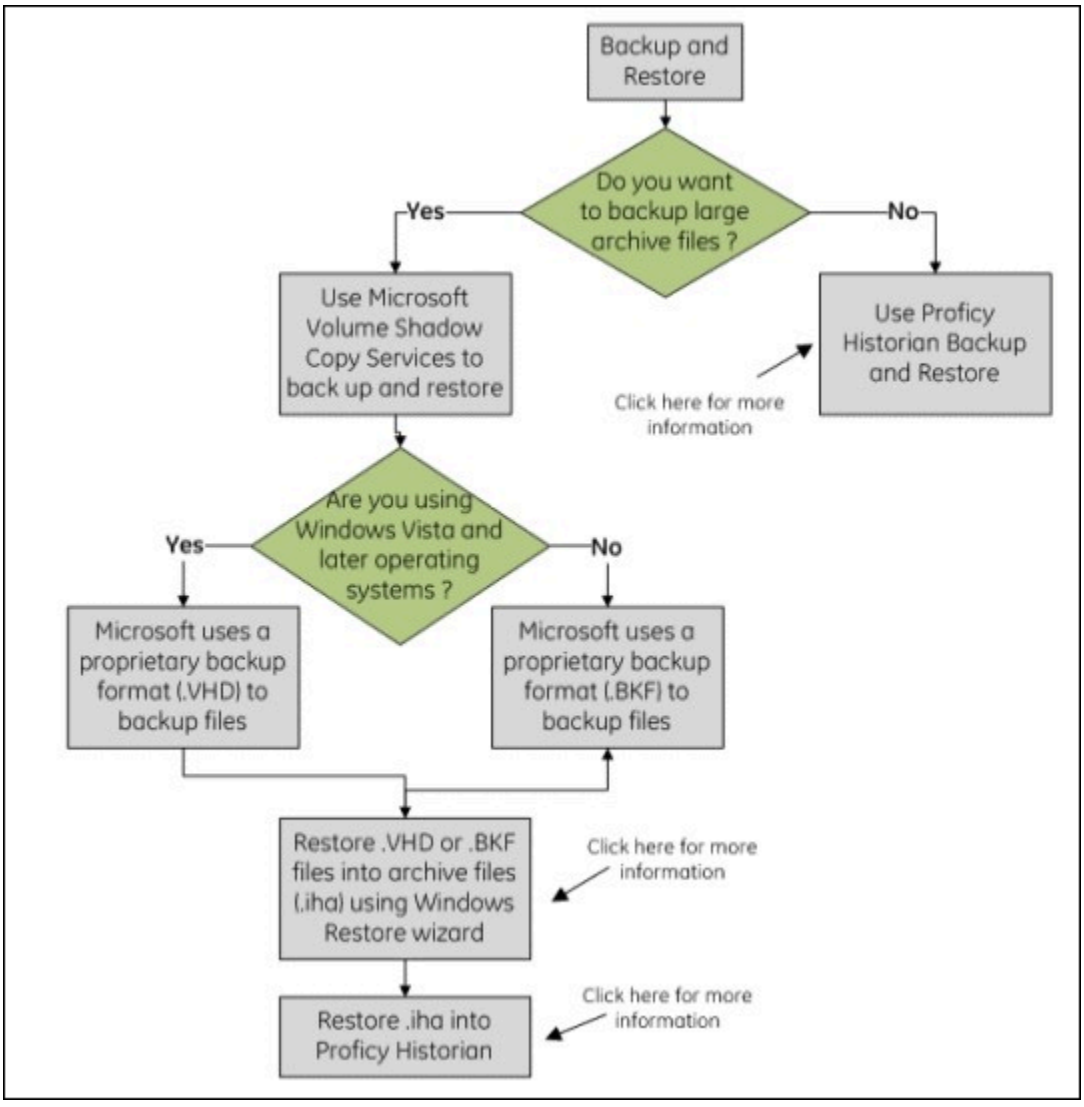
**Note:**
Though you could use either ihArchiveBackup.exe or VSS for backup, VSS is a better choice for both larger archives or if you are backing up more than the last two archives to reduce the load on the Data Archiver service.

Microsoft uses a backup format called Virtual Hard Disk (VHD) to back up files.

When you create archives backup using Microsoft® Volume Shadow Copy Service, you must first restore the archives files (that is, .bkf or .vhd into .iha) using the Windows Restore wizard, and then restore the archives (.iha) into Historian. For more information on restoring an archive (.iha) into Historian, refer to the Restoring an Archive topic.

**Note:**
It is recommended that you:

- Use Microsoft® Volume Shadow Copy Service when you want to back up archive files that are more than 2 GB in size, or if you are backing up more than your last two archives.
- Ensure you have enough hard drive space on your default backup location before backing up your archives.




**Important:**

For optimum performance, it is recommended you to save paging file of the operating system, Historian archives, and scheduled backup directory on separate drives.

## Restore an Archive

- Before restoring an archive from a removable disk, copy the archive file to the normal archive path and then restore the archive from that location. Leave the original backup file in the backup file folder.
- Ensure that the current archive is online.

Under certain circumstances, you may want to restore tag and alarms and events data to Historian. This may be after an unplanned shutdown, or you may need to retrieve data from an old, inactive archive.

**Important:**
Restoring an archive is a resource-intensive operation and should be scheduled for non-peak usage times.

Archives that have been previously removed from Historian can be found in the `\Archives\Offline` directory.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Archives**.
   The **Archives** section appears.
3. In the upper-right corner of the section, select .
   The **Restore Archives** window appears.
4. Enter values as described in the following table.

| Field | Description |
|---|---|
| **FILE LOCATION** | This field is populated with the value in the **DEFAULT BACKUP PATH** field of the data store. Append the name of the .zip file of the archive that you want to restore. The **ARCHIVE NAME** field is populated with the .zip file name that you enter. |
| **DATA STORE** | Select the data store in which you want to restore the archive. |

5. Select **Restore**.
   The archive is restored; it is moved to the `Archive` folder and is made available for querying.

## Close an Archive

By default, an archive is closed when it is full. However, you can manually close an archive before it is full.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Archives**.
   The **Archives** section appears.
3. Right-click the archive that you want to back up (or select ⋯), and then select **Close Archive**.
   A message appears, asking you to confirm that you want to close the archive.
4. Select **Yes**.
   The archive file is closed, and another one is used for writing data.

## Remove an Archive

When you remove an archive, it no longer appears in Configuration Hub. However, it exists in the Archives folder (by default, `C:\Proficy Historian Data\Archives`).

You cannot remove the current archive. If you want to remove it, you must first close it. When you do so, another archive is used for writing data.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Archives**.
   The **Archives** section appears.
3. Right-click the archive that you want to remove (or select ⋯), and then select **Remove Archive**.
   A message appears, asking you to confirm that you want to remove the archive.
4. Select **Yes**.
   The archive file is removed.

# Reading/Writing Data

## Query Data

Using Configuration Hub, you can query the data of selected tags. This data is then plotted in a trend chart. You can also export this data into a PDF, SVG, PNG, or a JPEG file, or save it as favorites.

Querying data involves the following steps:

1. **Selecting the tags:** You can select tags from all the tags in the system. While selecting tags, you can view them in a hierarchical object model view or in a flat list.
2. **Applying conditions and filters:** You can select the sampling mode, size, query modifiers, and so on. You can also select the calculation modes for the calculated sampling mode.
3. **Generating the report:** You can plot the query results in a trend chart or view them in a table. You can also export the data.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Data**.
   The **Data** section appears, displaying a list of object instances and the underlying variables and contained types hierarchically. If you want to view a flat list of all the tags, select .
3. Select the tags for which you want to query data, select Read data operation, and then select **Next**. You can select up to 10 tags.
4. Enter values as described in the following table.

| Field | Description |
|---|---|
| **START DATE/TIME** | Enter the start date and time for the query. |
| **END DATE/TIME** | Enter the end date and time for the query. |
| **SAMPLING MODES** | Select the sampling mode for the query:<br>◦ **Calculated** *(on page 1073)*: Returns the result of a calculation on tag values (for example, count, maximum, delta).<br>◦ **Current Value** *(on page 1045)*: Returns a single sample containing the current value of the tag. Retrieves the timestamp, value, and quality.<br>◦ **Interpolated** *(on page 1048)*: Returns the interpolated tag values.<br>◦ **Interpolated to Raw**<br>◦ **Lab** *(on page 1047)*: Returns only collected values. Each collected value is repeated until the next collected value, resulting in a jagged step plot instead of a smooth curve.<br>◦ **Lab to Raw**<br>◦ **Raw By Number** *(on page 1057)*: Returns the specified number of raw samples of all qualities beginning with the start time and moving in the specified direction. |

| Field | Description |
|---|---|
| | ◦ **Raw By Time** *(on page 1057)*: Returns all raw samples of all qualities with a timestamp greater than the start time and less than or equal to the end time. It will not return a raw sample with same time stamp as the start time.<br>◦ **Trend** *(on page 1065)*: Returns significant points, which are raw samples. These points are determined by finding the raw minimum and raw maximum values within each interval.<br>◦ **Trend to Raw**<br>◦ **Trend to Raw2**<br>◦ **Trend2** *(on page 1069)*: Splits up a given time period into a number of intervals (using either a specified number of samples or specified interval length), and returns the minimum and maximum data values that occur within the range of each interval, together with the timestamps of the raw values. This sampling mode is suitable for analysis of minimum and maximum values and for plotting programs that can handle unevenly spaced data. |
| **CALCULATION MODE** | Select the calculation mode that you want to use. This field appears only if you select **Calculated** in the **SAMPLING MODES** field. For information on calculation modes, refer to Calculation Modes *(on page 1073)*. |
| **SAMPLING DIRECTION** | The direction in which you want to retrieve the query results:<br>◦ **Forward**: Returns query results starting from the start data to the end date.<br>◦ **Backward**: Returns query results starting from the end date to the start data. |
| **SAMPLE INCREMENT** | Identifies the amount of data samples in each sample:<br>◦ **By Size**: Select this option if you want each sample to contain a specific number of data points, and then enter the number of data points.<br>◦ **By Time**: Select this option if you want each sample to contain data points collected for a specified duration, and then enter the duration. |

| Field | Description |
|---|---|
| **QUERY MODIFIERS** | Used for retrieving a specific set of data. For information, refer to Query Modifiers *(on page 1159)*. |
| **Filters** | You can enter your filter conditions using the Filter tag, the Filter Condition, and the Filter Comparison Value.<br><br>In the Filter Tag Name field, select the tag you want to enable filtering with.<br><br>In the Condition field, select your comparison condition.<br><br>In the Value field, enter your filter condition value. |

5. Select **Generate Report**.

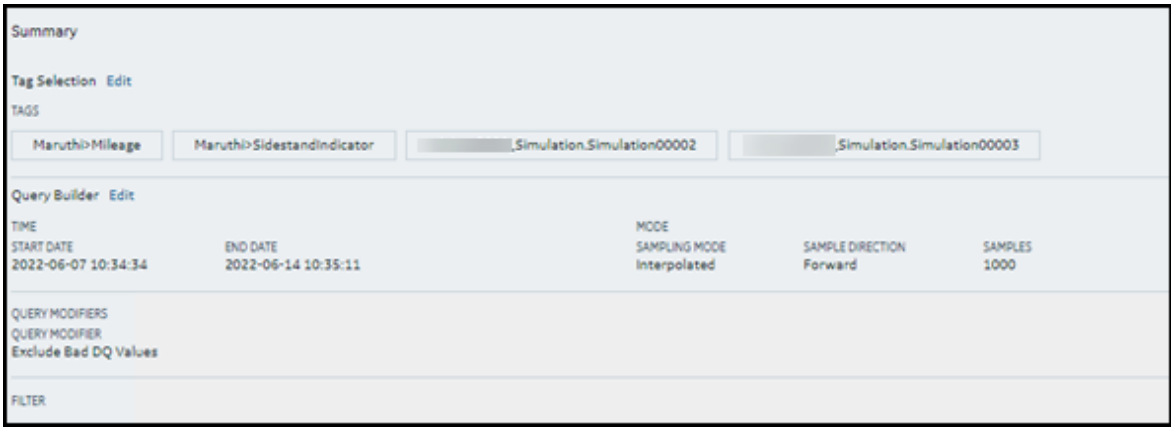
   The query results for the selected tags are plotted on a trend chart.

You can narrow down the start and end dates by dragging the timeline below the trend chart.

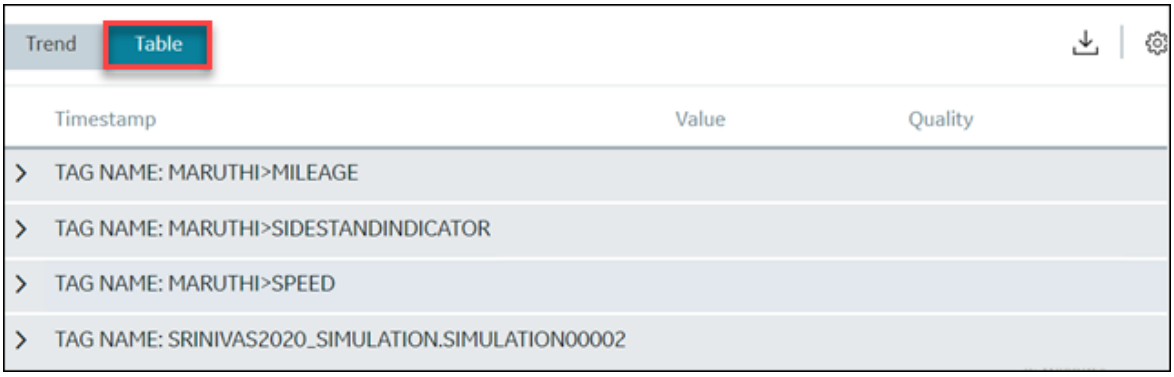
Query_Results.mp4

The Tags and query criteria appear in the **Summary** section; you can edit them if needed.

Select the Edit beside Tag Selection and Query Builder to modify tags and query criteria.




If you want to view the results in a table, select **Table**.




You can save the query if you may frequently use to read specific tags data with a specific query criteria. For more information, refer to Save a query *(on page 741)*.

If you want to export the results into a .csv file, select CSV. The results are exported.

If you want to print the trend, select ☰, and then select **Print**.

To export into other formats (such as a .pdf, .svg, .png, or .jpeg), select ☰, and then select the format.

## Write Data

Using Configuration Hub, you can write data for selected tags. This helps you verify that data is being sent to Data Archiver.

Writing data involves the following steps:

1. **Selecting the tags:** You can select tags from all the tags in the system. While selecting tags, you can view them in a hierarchical object model view or in a flat list.
2. **Entering the data:** You can enter data for each selected tag, along with the data type and quality.
3. **Generating the report:** You can plot the query results in a trend chart or view them in a table. You can also export the data.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Data**.
   The **Data** section appears, displaying a list of object instances and the underlying variables and contained types hierarchically. If you want to view a flat list of all the tags, select .
3. Under **DATA OPERATION**, select **Write Data**.
4. Select the tags for which you want to write data, and then select **Next**.
5. For each tag that you have selected, enter values as described in the following table.

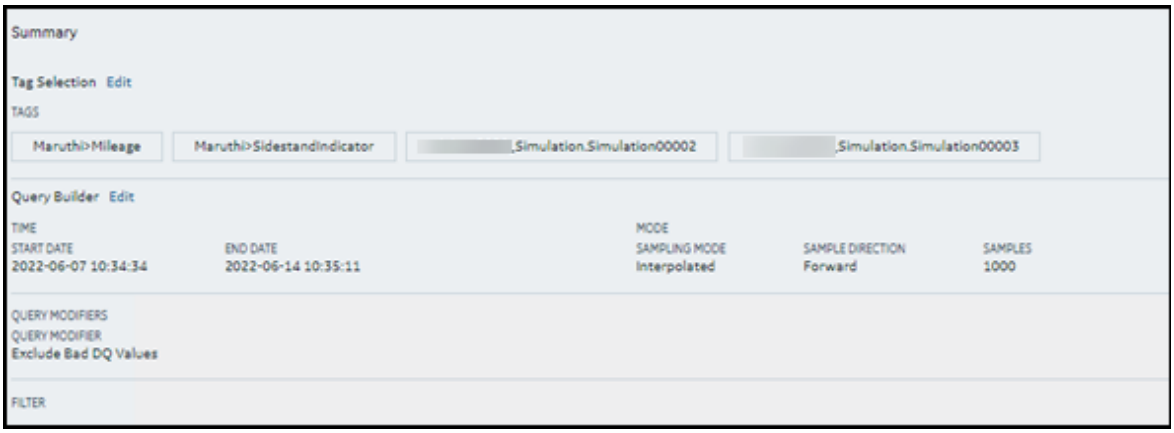
| Column | Description |
|---|---|
| **VALUE** | Enter the value of the tag. A value is required. |
| **DATA TYPE** | If Hierarchical tags are selected, select the data type of the tag value. A value is required.<br><br>If normal tags are selected, the data type will be automatically populated. |
| **DATA QUALITY** | Select the data quality of the tag value from Good or Bad from the list. A value is required. |

6. Select **Write Data**, and then select **Next**.
7. Select **Generate Report**.
   The query results for the selected tags are plotted on a trend chart.

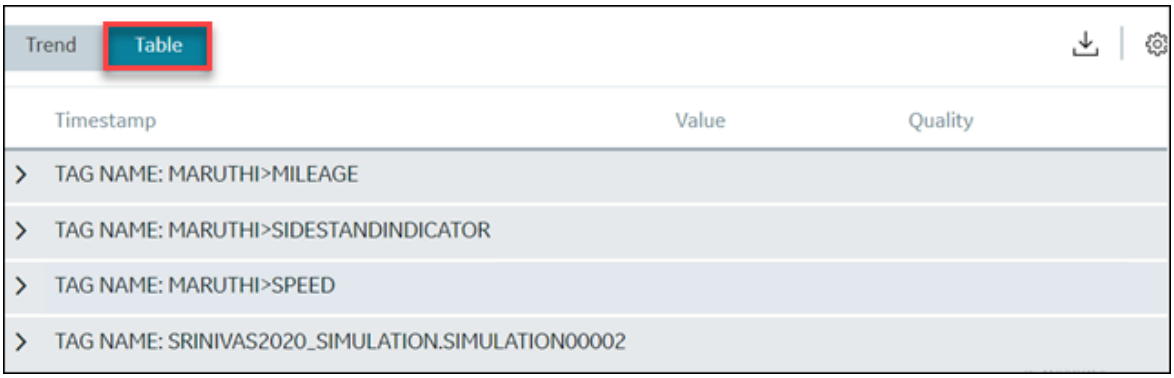
You can narrow down the start and end dates by dragging the timeline below the trend chart.

Query_Results.mp4

The query criteria appear in the **Summary** section; you can edit them if needed.



If you want to view the results in a table, select **Table**.




## About Saved Query

## About Saved Query

You can save queries that you use to read tags data. You can use this option to save queries that you may frequently use to read specific tags data with a specific query criteria. This will help you in the following:

- Quickly gather insights of specific tags.
- Create and save a predefined query only once and use it as needed.
- You can use the **Shared** option to make the query visible to the other users in the same network.

> **Note:**
> Once you share a query, all the other users will also be able to edit, and delete it. Be mindful about making a query as a shared query.

- Automate a monotonous query process through the saved queries.

**What to do Next**: You can Save a query *(on page 741)*.

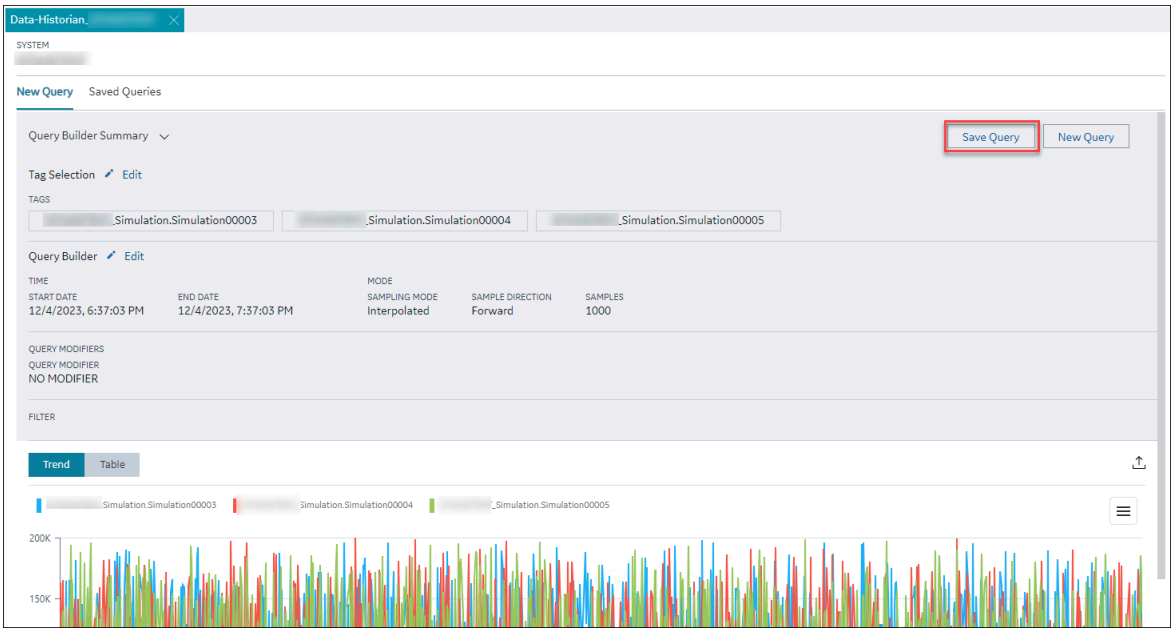
## Save a Query

This topic describes how to save a Query to read tags data.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Data**.
   The **Data** section appears, displaying a list of object instances and the underlying variables and contained types hierarchically. If you want to view a flat list of all the tags, select .

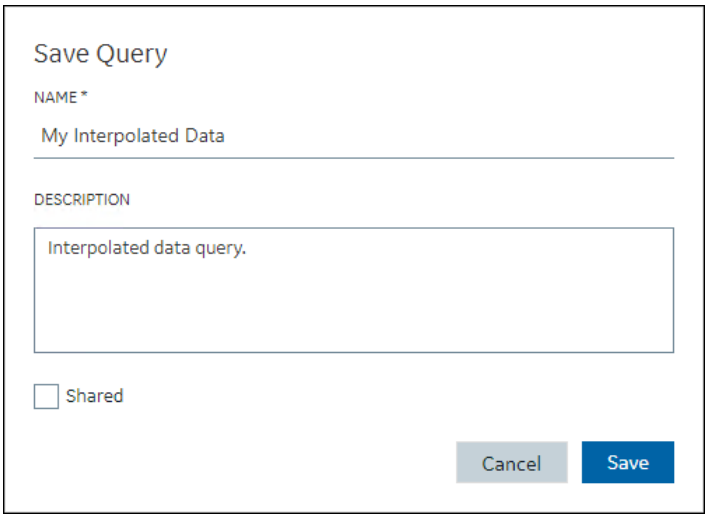
3. Select the tags for which you want to query data, select **Read Data** operation, and then select **Next**.
4. Enter the values as needed. For more information on the values, refer to Query Data *(on page 738)*.
5. Select **Generate Report**.

   The query results are plotted on a trend chart.




6. Select **Save Query**.

   The **Save Query** window appears.




7. Enter values as described in the following table.

| Field | Description |
|---|---|
| **NAME** | Enter a meaningful name for your query. You can enter alphanumeric values and also use special characters. |
| **DESCRIPTION** | This is optional. You can enter a description about the query you are creating. The description can help other users to get a quick overview of what this query is about. |
| **Shared** | Select this check box to share this query with others.<br><br>**Note:**<br>Once you share a query, all the other users will also be able to edit, and delete it. Be mindful about making a query as a shared query. |

8. Select **Save**.
   The query is saved.
9. Select the **Saved Queries** tab.
   The query you saved will be listed.

You can run or edit *(on page 744)* the query.

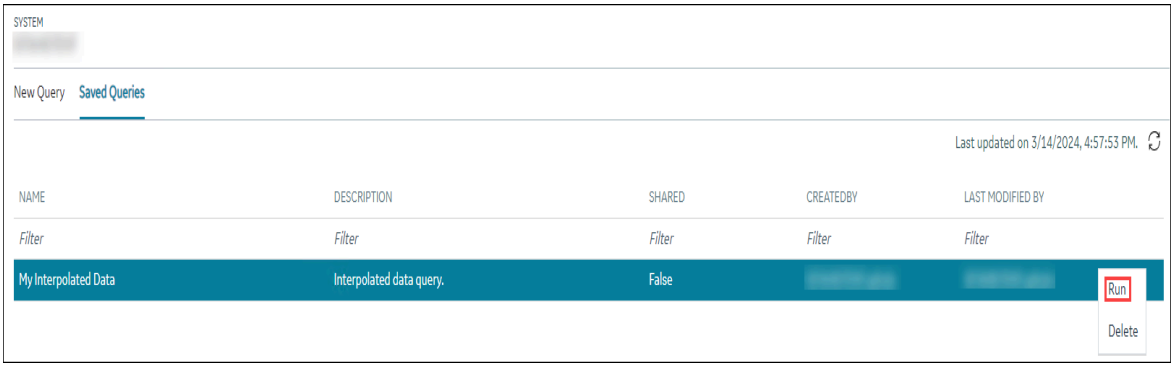
## Run a Saved Query

After you create and save a query, you can run the query and see the query results.

This topic describes how to run a saved query.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Data**.
   The **Data** section appears, displaying a list of object instances and the underlying variables and contained types hierarchically.
3. Select the **Saved Queries** tab.
   The list of all the saved queries appears.
4. Select a query as needed.

5. Select , and then select **Run**.




The query results are plotted on a trend chart.

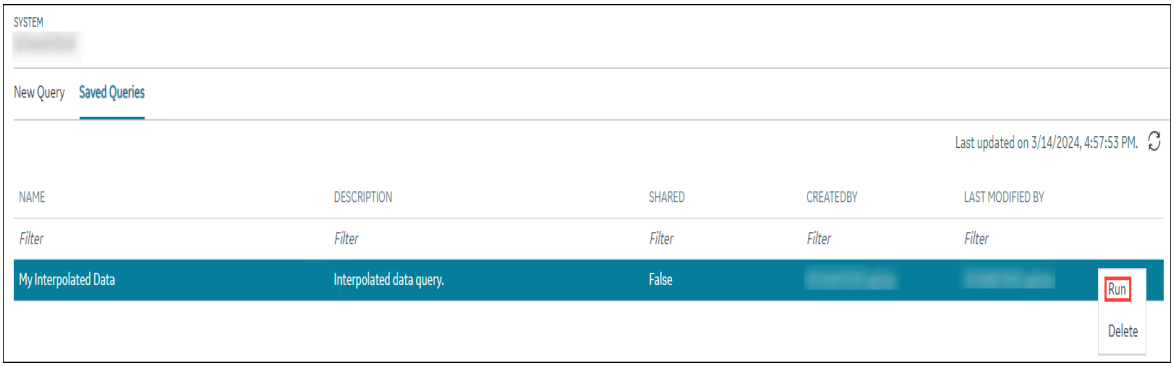
## Update a Saved Query

There can be changes in a process and so in the values. To update your saved queries, you can use the update option.

This topic describes how to edit and update a saved query.

**Note:**
If you are accessing a shared query, and if you did not create it, be mindful before you edit it. If possible, avoid editing the query.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Data**.
   The **Data** section appears, displaying a list of object instances and the underlying variables and contained types hierarchically.
3. Select the **Saved Queries** tab.
   The list of all the saved queries appears.
4. Select a query as needed.
5. Select , and then select **Run**.




The query results are plotted on a trend chart.

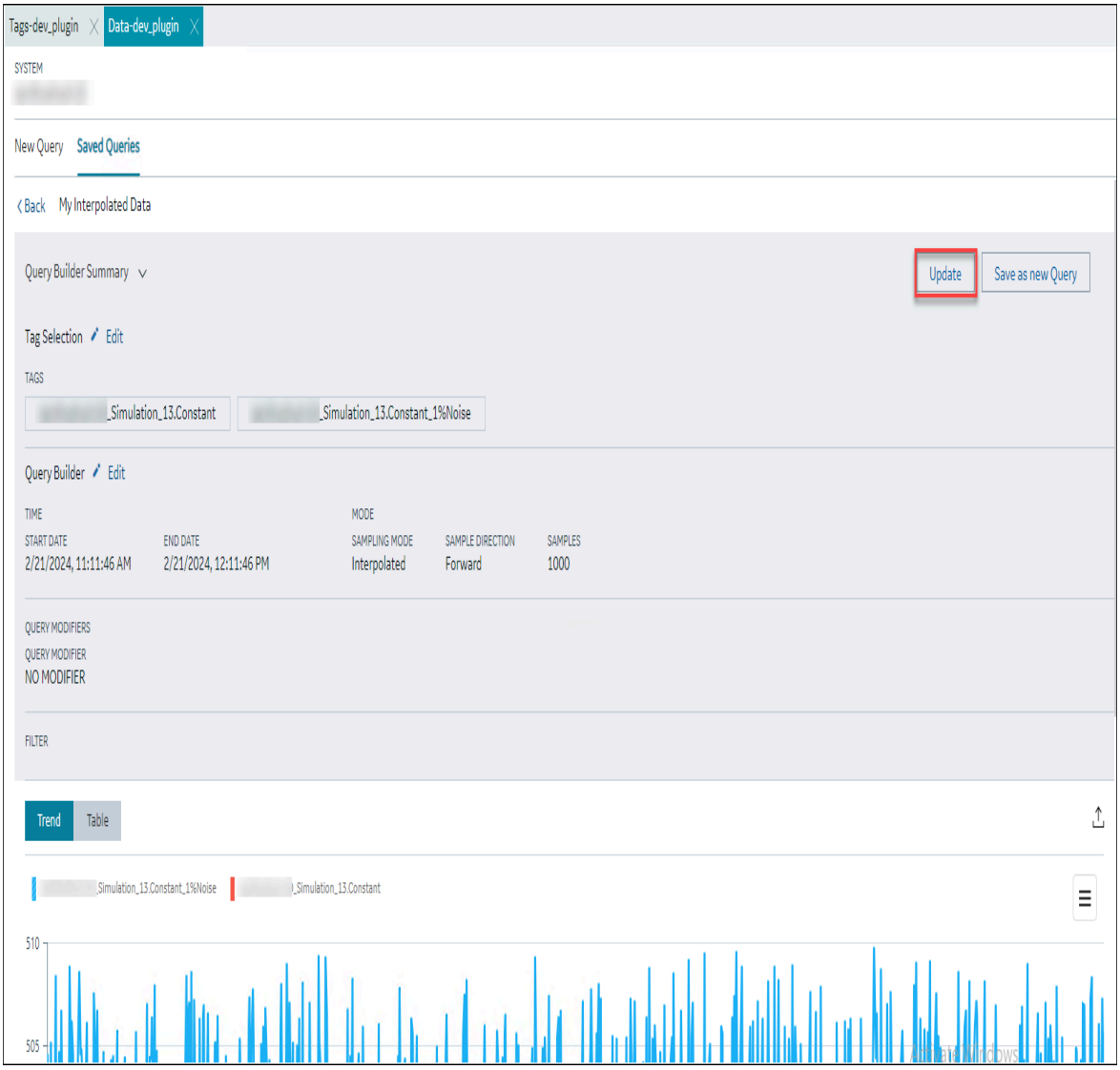
6. Select the **Edit** beside **Tag Selection** and **Query Builder** to edit tags and query criteria.

    a. Edit the tag selections or query criteria as needed.

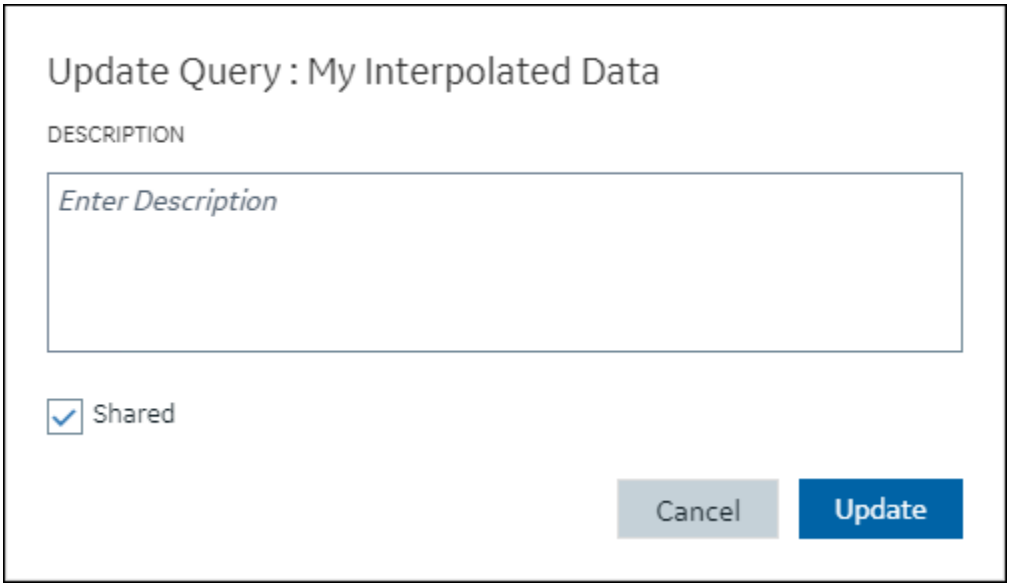
    b. Select**Generate Report**.

    The **Query Builder Summary** appears.

7. Select **Update**.




The **Update Query:<Query Name>** window appears.

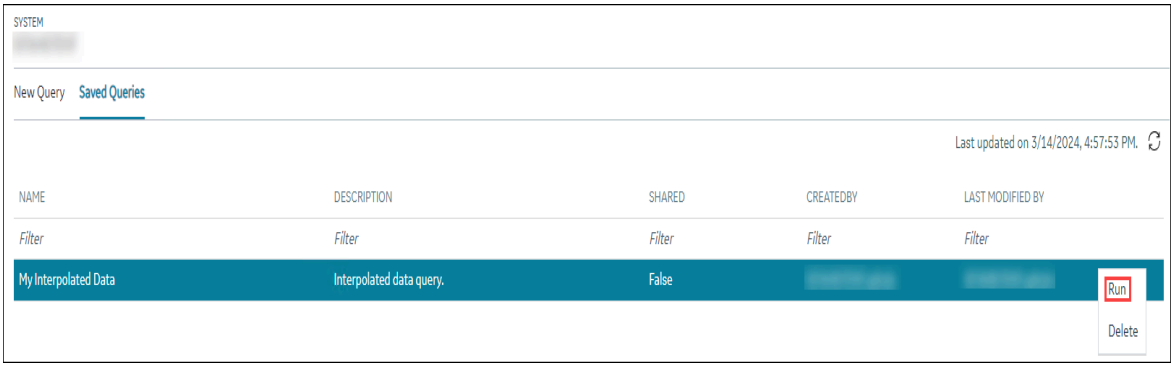



8. Update the description if needed
9. Select **Update**.
   The query is updated.
10. To go back to the saved queries tab, in the top-left corner, select **<Back**

## Update a Saved Query and Save it as a New Query

This topic describes how to update a saved query and save it as a new query.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Data**.
   The **Data** section appears, displaying a list of object instances and the underlying variables and contained types hierarchically.
3. Select the **Saved Queries** tab.
   The list of all the saved queries appears.
4. Select a query as needed.
5. Select , and then select **Run**.




The query results are plotted on a trend chart.

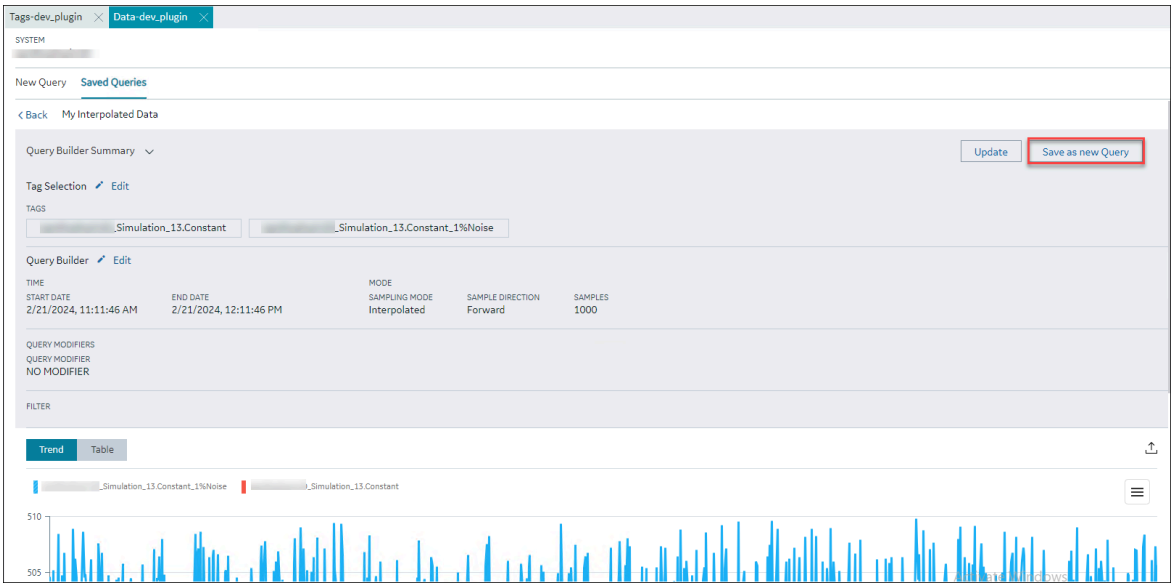
6. Select the **Edit** beside **Tag Selection** and **Query Builder** to edit tags and query criteria.

    a. Edit the tag selections or query criteria as needed.

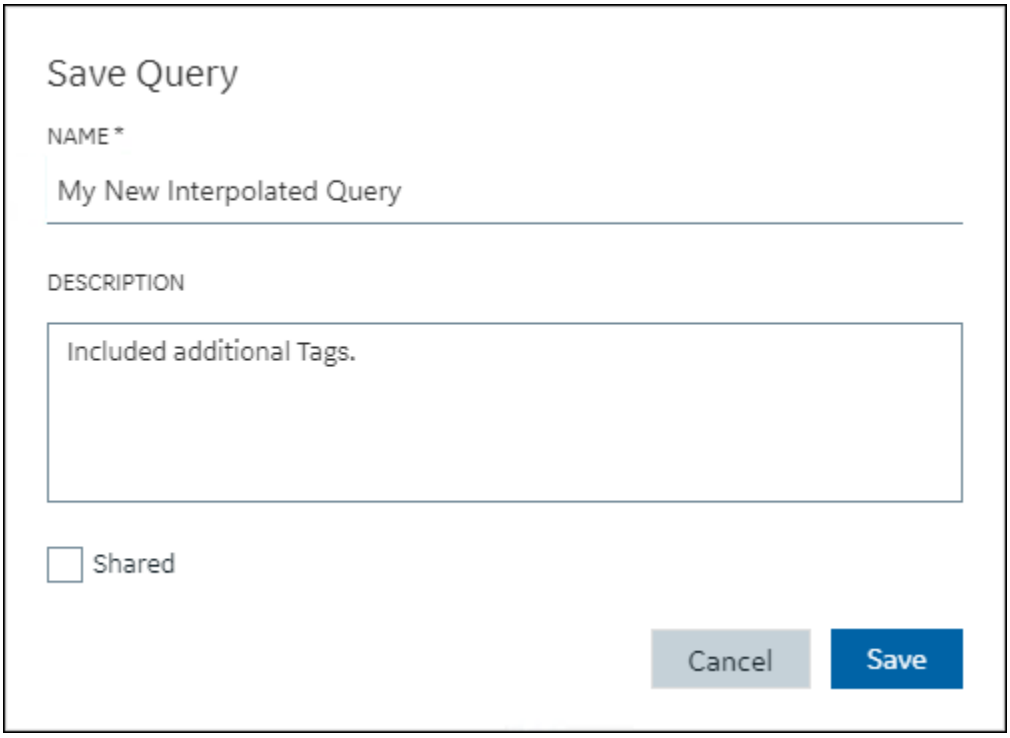
    b. Select**Generate Report**.

    The **Query Builder Summary** appears.

7. Select **Save as new Query**.




The **Save Query** window appears.




8. Enter a new name and description for the query.
9. Select **Save**.
   The updated query is saves as a new query.
10. To go back to the saved queries tab, in the top-left corner, select **<Back**

## Delete a Saved Query

This topic describes how to delete a saved query.

**Note:**
If you are accessing a shared query, and if you did not create it, be mindful before you delete it. Instead, kindly try to create a new query as needed.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Data**.
   The **Data** section appears, displaying a list of object instances and the underlying variables and contained types hierarchically.
3. Select the **Saved Queries** tab.
   The list of all the saved queries appears.
4. Select a query as needed.

5. Select , and then select **Delete**.
   The **Delete Query** window appears, prompting you whether to delete the saved query or not.
6. Select **Yes**.
   The saved query is deleted.

# Managing Alarms and Events

## About Alarms and Events

Using Configuration Hub, you can manage alarms and events data from any OPC-compliant alarms and events server.

> **Note:**
> The Alarm Archiver must be available and running to access this feature.

**Alarms:** Alarms are generally defined as tags going into an abnormal condition. For example, you can set an alarm on a boiler when it reaches a specified temperature. Alarms usually have a well-defined life cycle, which is defined by the individual data sources from which the alarms data is collected. They enter an alarm state, are generally acknowledged, then return to normal.

To collect alarms and events data, you can use any of the following collectors:

- iFIX Alarms and Events collector
- OPC Classic Alarms and Events collector

Historian handles alarm data in two ways. You can view the entire Alarm as a single record that contains all information about the alarm, or you can view the Alarm History, which shows the transitions of the alarm as individual records.

**Events:** Events are generally defined as activities in a system that occur once only. For example, a user logging on to a device is an event. When viewing this data in Historian, each event is returned as a record.

## Alarms and Events Requirements

To generate alarms and events data, you can use any OPC-compliant alarms and events server such as:

- CIMPLICITY
- iFIX

To collect alarms and events data, install collectors *(on page 142)*, and then add any of the following collector instances:

- iFIX Alarms and Events collector *(on page 505)*
- OPC Classic Alarms and Events collector *(on page 532)*

To store alarms and events data, install Historian Alarms and Events *(on page 140)*. During the installation, provide the credentials of Microsoft SQL server 2012 or later.

To view and analyze alarms and events data, you can use any of the following applications:

- Configuration Hub
- Historian Administrator
- Crystal Reports 11 or later
- The OLEDB Provider
- REST APIs

**Note:**
Before starting the Alarm Archive service, ensure that "`NT AUTHORITY/SYSTEM`" has "`SysAdmin`" privileges.

## Create an Alarm

This topic describes how to create an alarm manually. You can use it to verify that Historian is collecting alarm data. You can then use the iFIX Alarms and Events collector or the OPC Classic Alarms and Events collector to collect alarm data and store it in Historian.

1. Access Configuration Hub *(on page 336)*
2. In the **NAVIGATION** section, under the Configuration Hub for Historian, select **Alarms**.
   The **Alarms** section appears.
3. In the upper-right corner of the page, select ooo.

   **Note:**
   To select hierarchical model tags you must need to have the valid license.

   The **Add Alarm** window appears, displaying a list of tags in a hierarchical view. If you want to view a flat list of all the tags, select .

4. Select the check boxes corresponding to the tags for which you want to create an alarm, and then select **Next**.
   The **Alarm Attributes** section appears, displaying the selected tags. The **TAG NAME** and **TIMESTAMP** columns are automatically populated.
5. For each tag, enter values as described in the following table.

| Column | Description |
|---|---|
| **SEVERITY** | Specify whether you want to record high or low severity for the alarm. |
| **DATA QUALITY** | Specify whether you want to record good or bad quality for the alarm. |
| **MESSAGE** | Enter a message for the alarm. |

6. Select **Write Alarm**.
   The alarm is created.

## Access/Filter Alarms

1. Access Configuration Hub *(on page 336)*
2. In the **NAVIGATION** section, under the Configuration Hub for Historian, select **Alarms**.
   The **Alarms** section appears.
3. To filter for alarms, enter values as described in the following table.

| Field | Description |
|---|---|
| **GENERATED FROM** | Enter the start date and time for which you want to filter alarms. By default, ten minutes earlier than the current time is considered. A value is required. |
| **GENERATED TO** | Enter the end date and time for which you want to filter alarms. By default, the current time is considered. A value is required. |
| **COLLETOR NAME** | Select the collector that collects the alarms and events data. |

4. Select **Apply**.

The alarms and events data is filtered based on the criteria. To access the events data, select **Events**. The list of events is also filtered based on the criteria.

> **Tip:**
> You can show/hide/reorder columns in the table. For instructions, refer to Common Tasks in Configuration Hub *(on page 353)*.

## Back up Alarms Using Configuration Hub

1. Access Configuration Hub *(on page 336)*
2. In the **NAVIGATION** section, under the Configuration Hub for Historian, select **Alarms**.
   The **Alarms** section appears.
3. Select ⋯, and then select **Backup Alarms**.
   The **Backup Alarms** window appears.
4. Enter values as described in the following table.

| Field | Description |
|---|---|
| **GENERATED FROM** | Enter the start time for which you want to back up alarms. |
| **GENERATED TO** | Enter the end time for which you want to back up alarms. |
| **PROVIDE FILE NAME** | Enter the file name and location of the back up file that you want to create. By default, the backup location of the default data store is considered. And, the file name is in the following format: `mm_dd_yyyy_hh_mm_ss`. For example: 11_25_2022_11_23_26.zip. The time format followed is the 24-hour time notation.<br><br>**Note:**<br>This file naming convention for alarm backup is standard to Historian. Even if you use other date settings such as `dd-mm`, the file name will still be saved in the mm-dd format.<br><br>The end time stamp in the file name indicates the time at which the alarms have been backed up but not the time till when the alarms are backed up. For example, suppose, at 8.00 am, you back up alarms for the past two hours, the backup will contain alarms from 6:00 am to 8:00 am but may not contain an alarm |

| Field | Description |
|---|---|
| | generated at 8:00 am. The last alarm may have been at 7:50 am. But the backup file name will have the time stamp of 8:00:00 along with the date. |

5. Select **Backup**.
   The alarms data is backed up.

## Restore Alarms Using Configuration Hub

Restoring alarms to a running system makes them available for query and analysis. You can restore alarms that have been backed up or deleted previously.

1. Access Configuration Hub *(on page 336)*
2. In the **NAVIGATION** section, under the Configuration Hub for Historian, select **Alarms**.
   The **Alarms** section appears.
3. Select , and then select **Restore Alarms**.
   The **Restore Alarms** window appears.
4. In the **PROVIDE FILE NAME** field, provide the backup file in the .zip format that you want to restore.

   **Note:**
   Remember to include .zip at the end of the file name.

5. Select **Restore**.
   The alarm data is restored.

## About Purging Alarms

Purging alarm data involves deleting the data from the database.

**Note:**

- Even after purging, the data is not lost; a backup is created to maintain an audit trail. You can restore the data if needed.
- When using circular archives (that is, archives that roll over), alarms are purged automatically.

You can choose to purge alarm data for any of the following reasons:

- To maintain alarm data efficiently
- The data is outdated or redundant
- The disk space is limited

Data in the following tables is purged:

- Alarm Attribute Values
- Alarm Attribute Value History
- Delete from Alarm History
- Delete from Alarm Table esignatures
- comments

You can purge data using one of the following methods:

- Purge data within a specified duration. You can do this using Configuration Hub *(on page 756)*, using the Proficy Historian Alarm and Event Data Migration utility *(on page 1221)* or at a command prompt *(on page 1223)*.
- Purge data related to a specific alarm ID. You can do this using `Alarms.PurgeAlarmsById` *(on page 1604)* to develop an SDK program.

Purging is performed in batches. You can check the log data in the `Proficy.Historian.AandE.Migration.log` file. By default, this file is located in the `C:\Program Files(x86)\Proficy` folder.

In the case of a failure:

- The batch size is changed to 10. That is, the collector receives an acknowledgement after sending 10 messages, thus reducing the load on the server.
- The waiting time for receiving an acknowledgement is automatically incremented after each failure per batch, starting from 90 seconds to 270 seconds. This gives more time for the server to respond.

**Note:**
After the acknowledgement is received, the batch size and the waiting time are reset for the subsequent batches.

If the time taken to purge exceeds the timeout limit, instead of reverting the entire purging operation, only the current batch, which is still under processing, is purged.

**Best Practices:**

- Restart the Alarms and Events services before purging data.

## Purge Alarms Using Configuration Hub

1. Access Configuration Hub *(on page 336)*
2. In the **NAVIGATION** section, under the Configuration Hub for Historian, select **Alarms**.
   The **Alarms** section appears.
3. Select ooo, and then select **Purge Alarms**.
   The **Purge Alarms** window appears.
4. In the **PROVIDE FILE NAME** field, pro
5. Enter values as described in the following table.

| Field | Description |
|---|---|
| **GENERATED FROM** | Enter the start time for which you want to purge alarms. |
| **GENERATED TO** | Enter the end time for which you want to purge alarms. |
| **PROVIDE FILE NAME** | Enter the file name and location of the back up file that you want to create. This field is enabled only if you select the **BACKUP ALARMS BEFORE PURGE?** check box. By default, the backup location of the default data store is considered. And, the file name is in the following format: `mm_dd_yyyy_hh_mm_ss`. For example: 11_25_2022_11_23_26.zip. The time format followed is the 24-hour time notation.<br><br>**Note:**<br>This file naming convention for alarm backup is standard to Historian. Even if you use other date settings such as `dd-mm`, the file name will still be saved in the mm-dd format.<br><br>The end time stamp in the file name indicates the time at which the alarms have been backed up but not the time till when the alarms are backed up. For example, suppose, at 8.00 am, you |

| Field | Description |
|---|---|
| | back up alarms for the past two hours, the backup will contain alarms from 6:00 am to 8:00 am but may not contain an alarm generated at 8:00 am. The last alarm may have been at 7:50 am. But the backup file name will have the time stamp of 8:00:00 along with the date. |
| **BACKUP ALARMS BEFORE PURGE?** | Select the check box if you want to back up alarms before purging. |

6. Select **Purge**.
   The alarms data is purged. A backup file is created if you have selected the **BACKUP ALARMS BEFORE PURGE?** check box.

# Managing Configuration Templates

## About Configuration Templates

A configuration template is a predefined set of common configuration options. You can create a configuration template for data stores and collectors separately and apply it to data stores and collectors as needed. This will help you save time by eliminating the need to configure common parameters manually and reduce monotonous tasks.

**Note:**
You can apply a data store configuration template to a user-created data store, provided they are not set as the default data store.

The following are some of the advantages of creating a configuration template:

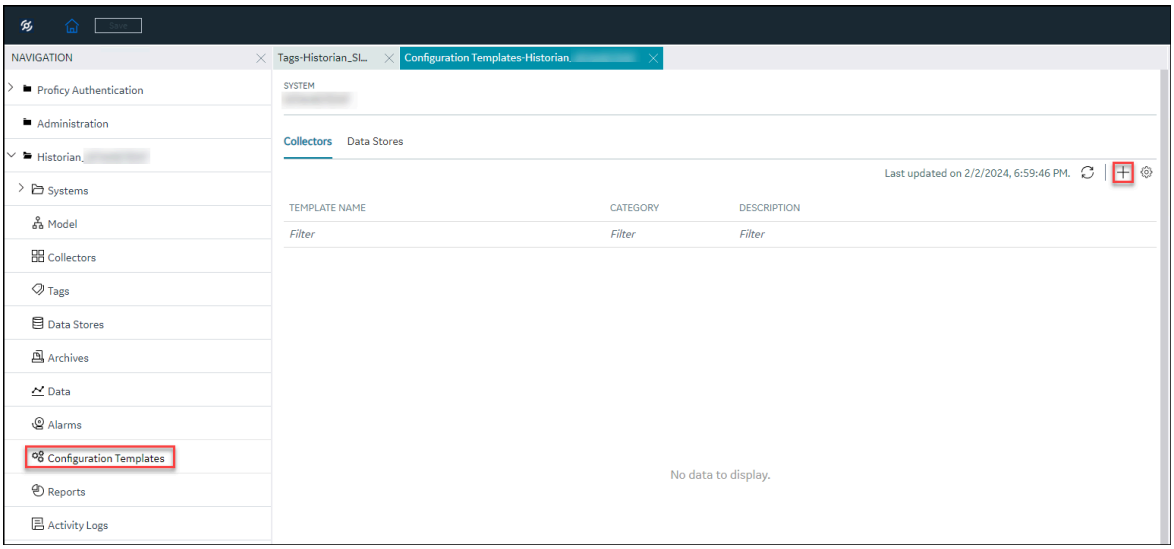
- Update some common configurations of a data store or collector.
- Save time from configuring common parameters.

## Create a Configuration Template for Collectors

This topic describes how to create a configuration template for collectors.

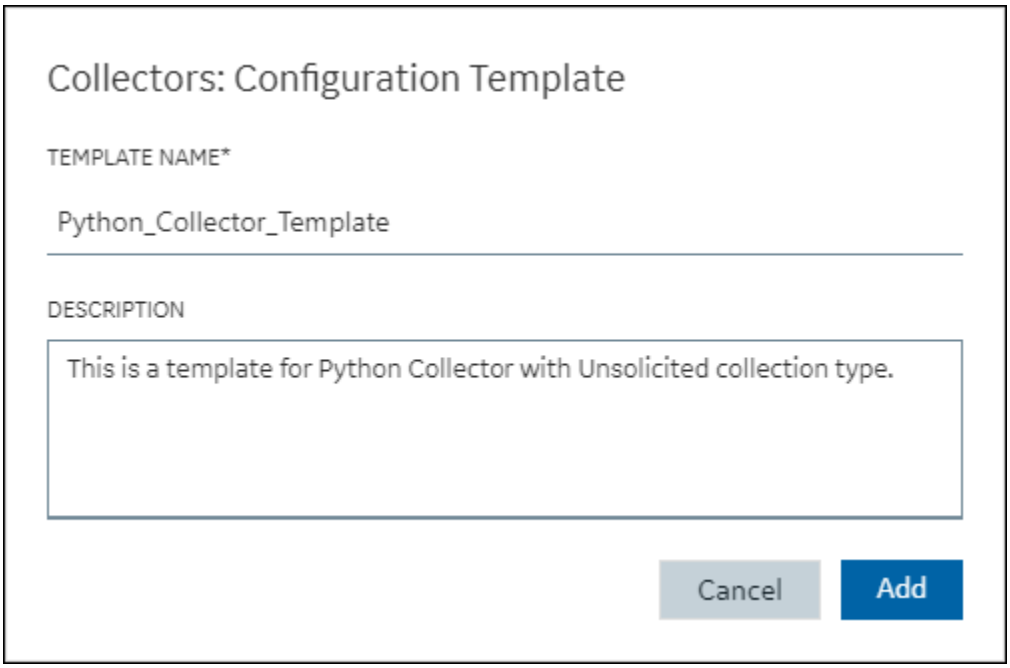
1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select**Configuration Templates**.

   The **Collectors** tab appears, providing you to option to create template.




3. Select +.

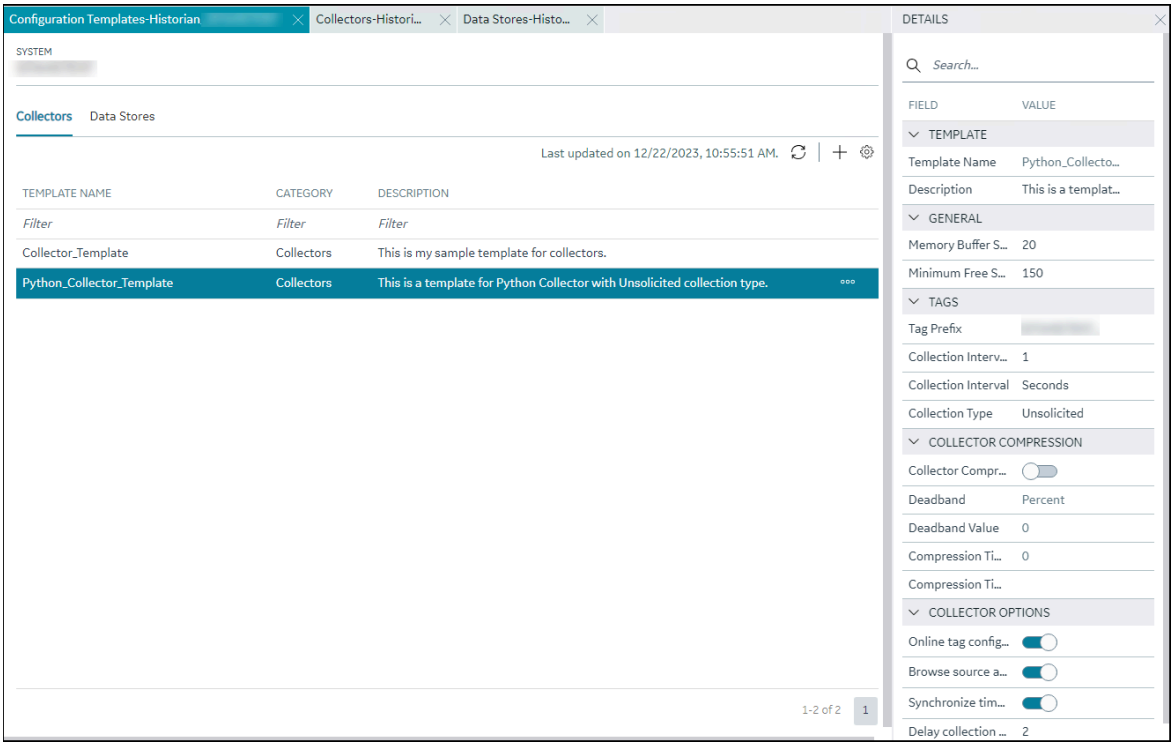
   The **Collectors:Configuration Template** window appears.




4. In **TEMPLATE NAME**, enter a name for the template. You can enter any alphanumeric names and also use special characters.
5. In **DESCRIPTION**, enter a description for your template.

6. Select **Add**.

   The configuration template is created.




7. Select the row that contains the created template.

   The configuration details appear in the **DETAILS** section.

**Table 47. The Template Section**

| Field | Description |
|---|---|
| **Template Name** | The name of the template. This field is read-only. |
| **Description** | The description of the template. You can edit the description if needed. |

**Table 48. The General Section**

| Field | Description |
|---|---|
| **Memory Buffer Size (MB)** | The size of the memory buffer currently assigned to the store-and-forward function. The memory buffer stores data during short-term or momentary interruptions of the server connection; the disk buffer handles long duration |

| Field | Description |
|---|---|
| | outages. To estimate the size you need for this buffer, you need to know how fast the collector is trying to send data to the server and how long the server connection is likely to be down. With those values and a safety margin, you can compute the required size of the buffer.<br><br>The default value is 20. |
| **Minimum Free Space (MB)** | The minimum free disk space that must be available on the computer. If the minimum space required is not available when the collector starts, the collector will shut down. |

**Table 49. The Tags section**

| Field | Description |
|---|---|
| **Tag Prefix** | The prefix that will be added to each tag that you configure for the collector instance. This field is disabled and populated with the name of the collector instance.<br><br>This field applies to all collectors except File and Calculation collectors. |
| **Collection Interval Value** | The interval at which the collector collects data for all the tags configured in the collector instance.<br>◦ For polled data collection, this value represents the time required to complete a poll of tags in the collector.<br>◦ For unsolicited data collection, it represents the frequency at which data is retrieved from tags in the collector. The collection interval can be individually configured for each tag.<br>You can set this value for each tag as well. |

| Field | Description |
|---|---|
| | **Important:**<br>For an OPC collector, to avoid collecting redundant values when using device timestamps, specify a collection interval that is greater than the OPC server update rate. |
| **Collection Interval** | The units of measure for the collection interval value. |
| **Collection Type** | The type of the data collection:<br>◦ **Polled:** Data is collected based on a scheduled time interval. This type of data collection is supported only for:<br>▪ The Calculation collector<br>▪ The HAB collector<br>▪ The iFIX collector<br>▪ The OPC Classic DA collector<br>▪ The OPC UA DA collector<br>▪ The Python collector<br>▪ The Simulation collector<br>▪ The Windows Performance collector<br>◦ **Unsolicited:** Data is collected based on an event. This type of data collection is supported only for:<br>▪ The Calculation collector<br>▪ The HAB collector<br>▪ The MQTT collector<br>▪ The MQTT Sparkplug B collector<br>▪ The ODBC collector<br>▪ The OPC Classic DA collector<br>▪ The OPC Classic HDA collector<br>▪ The OPC UA DA collector<br>▪ The OSI PI collector<br>▪ The OSI PI distributor |

| Field | Description |
|---|---|
| | ▪ The Python collector<br>▪ The Server-to-Server collector<br>▪ The Server-to-Server distributor<br>▪ The Wonderware Collector |

**Table 50. The Collector Compression Section**

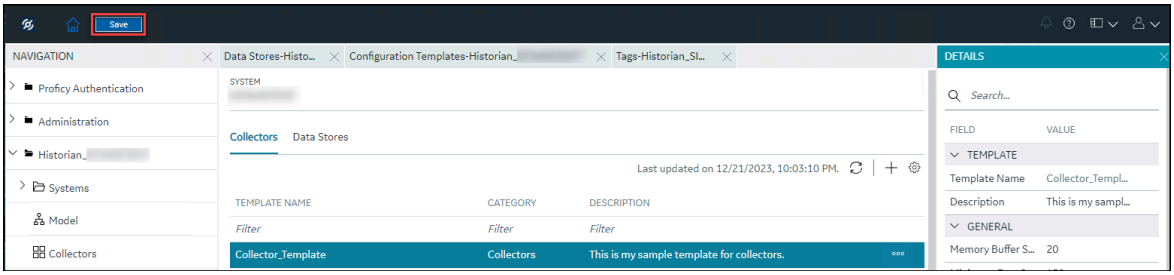
| Field | Description |
|---|---|
| **Collector Compression** | Indicates whether you want to apply collector compression, which is a smoothing filter to data retrieved from the data source. By ignoring small changes in values that fall within a deadband centered around the last reported value, only significant changes are stored in Historian, thus consuming less archive storage space.<br><br>For more information, refer to About Collector and Archive Compression *(on page 660)*. |
| **Deadband** | Indicates whether you want to apply a deadband based on the percentage of values or on absolute values.<br><br>For example, if you set the deadband to 20% for a range of 0 to 500 engineering units, the deadband value is 100 units, which is 50 units on each side. Therefore, only if the difference between two values is greater than 50, they are stored in Historian.<br><br>**Note:**<br>If the data quality changes from good to bad or vice versa, the values are stored in Historian regardless of the deadband value. |
| **Deadband Value** | The deadband value that you want to use for values collected by the collector. Depending on |

| Field | Description |
|---|---|
| | whether you have selected percent or absolute, the deadband value is determined.<br><br>For example, if you want to set a deadband of 5 units on either side of a value (that is, value +/- 5), enter 10 in the **Deadband Value** field, and select **Absolute** in the **Deadband** field. Similarly, if you want to set a deadband of 5% on either side of a value, enter 10 in the **Deadband Value** field, and select **Percent** in the **Deadband** field.<br><br>For more information, refer to About Collector and Archive Compression *(on page 660)*. |
| **Compression Timeout** | The time for one poll cycle for which collector compression is not used, thus sending all the samples to Historian.<br><br>This is used for a Calculation collector or Server-to-Server collector, when calculations fail, you may possibly observe collector compression (even if it is not enabled), thus producing no results or bad quality data. In such cases, you can use compression timeout, thus sending all the samples to Historian.<br><br>For more information, refer to About Collector and Archive Compression *(on page 660)*. |
| **Compression Timeout Interval** | The units of measure for compression timeout. |

**Table 51. The Collector Options Section**

| Field | Description |
|---|---|
| **Online Tag Configuration Changes** | Indicates whether you want tag configuration changes to reflect immediately. If you disable this option, any tag configuration changes will reflect only after you restart the collector instance. |

| Field | Description |
|---|---|
| **Browse Source Address Space** | Indicates whether you want to allow browsing for tags in the source. You may sometimes want to disable this option to reduce processing load on the collector. |
| **Synchronize Timestamps to Server** | Indicates whether you want to adjust the timestamp of data to align with the time setting in the Historian server. Note that this does not change the time setting in the collector machine; it only calculates the timestamp based on the difference between the time settings in the server machine and the collector machine, independent of time zone or daylight saving differences.<br><br>**Note:**<br>◦ This option is applicable only if the timestamp of the collector is considered (instead of that of the data source - as specified in the **Time Assigned By** field).<br>◦ If this option is disabled, and if the time in the collector machine is more than 15 minutes ahead of the time in the server machine, data will not be stored in Historian. |
| **Delay Collection at Startup (sec)** | The duration, in seconds, after which you want the data collection to begin post tag configuration. |




8. After you edit the details as needed, in the upper-left corner, select **Save**.
   The configuration details are saved.

   You can select ••• and duplicate this template, edit, or delete it.

You can apply the configuration template to a collector instance *(on page 650)*.

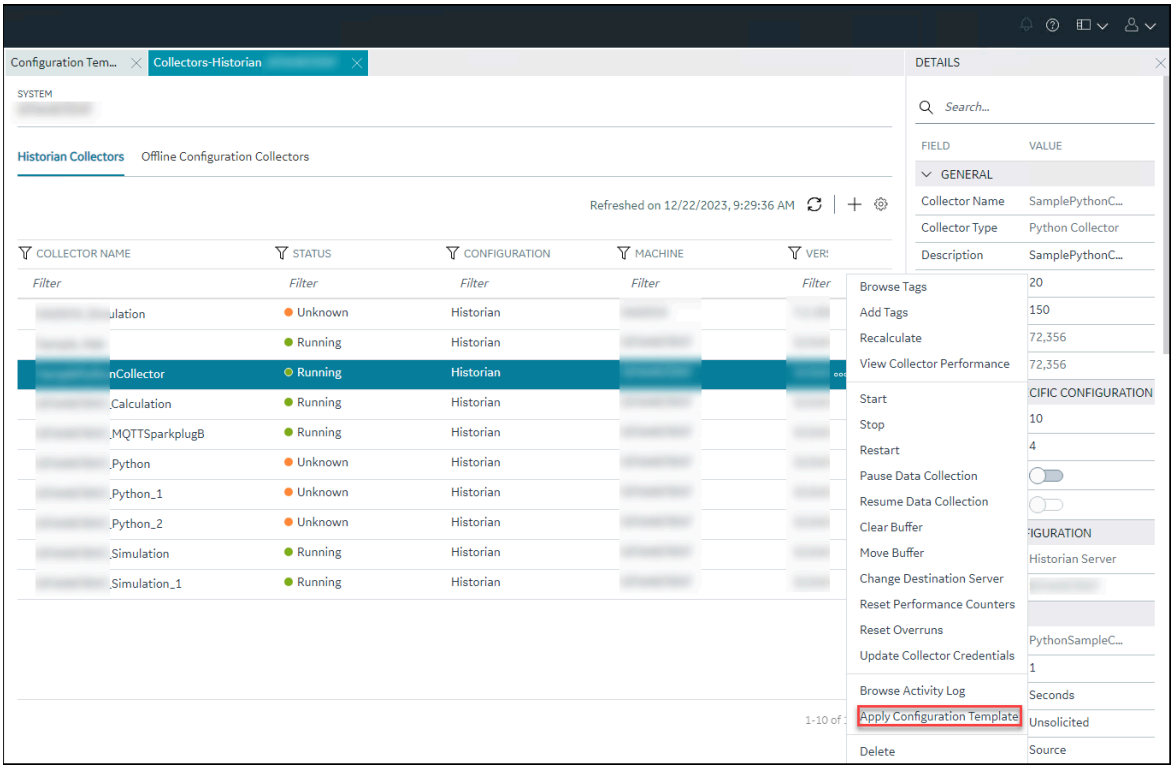
## Apply Configuration Template to a Collector

You can apply the created template to collectors as needed. You will be prompted to confirm whether you want to overwrite few of the configuration values with the values in the template.

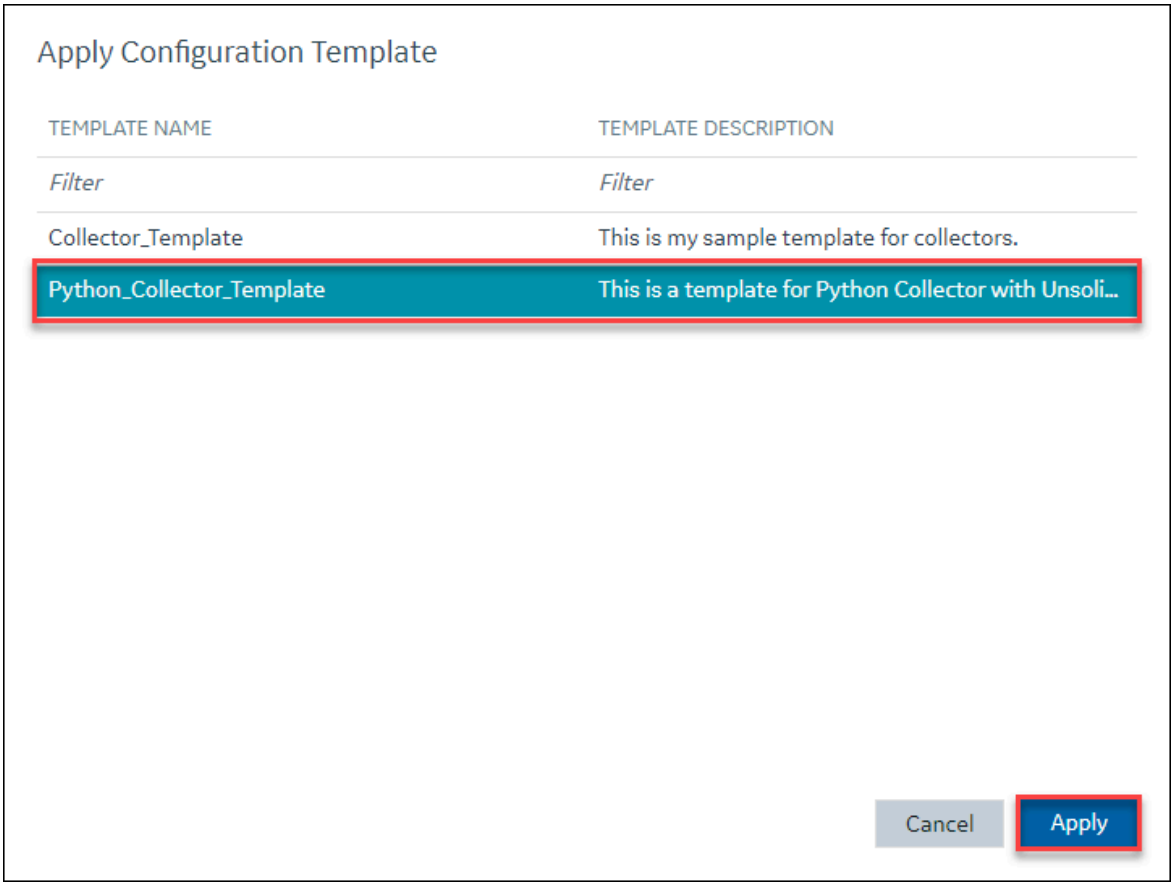
- Ensure that you have a collector instance added *(on page 357)*.
- Ensure that you created a configuration template for collectors *(on page 757)*.

This topic describes how to apply a configuration template to a collector.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.
   A list of collectors appears.
3. Right-click the collector (or select •••), and then select **Apply Configuration Template**.




The **Apply Configuration Template** window appears, listing the available templates.

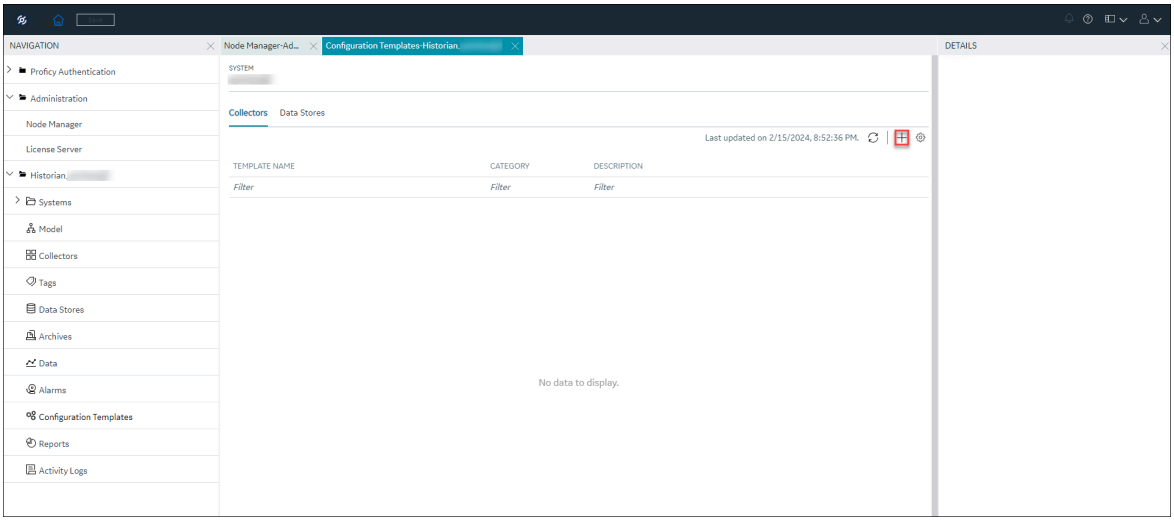



4. Select **Apply**.
   A confirmation window appears, prompting you to confirm whether you want to overwrite few of the configuration values with the values in the template.
5. Select **Ok**.
6. In the upper-left corner, select **Save**.
   The configurations in the template are applied to the collector.

## Create a Configuration Template for Data Stores

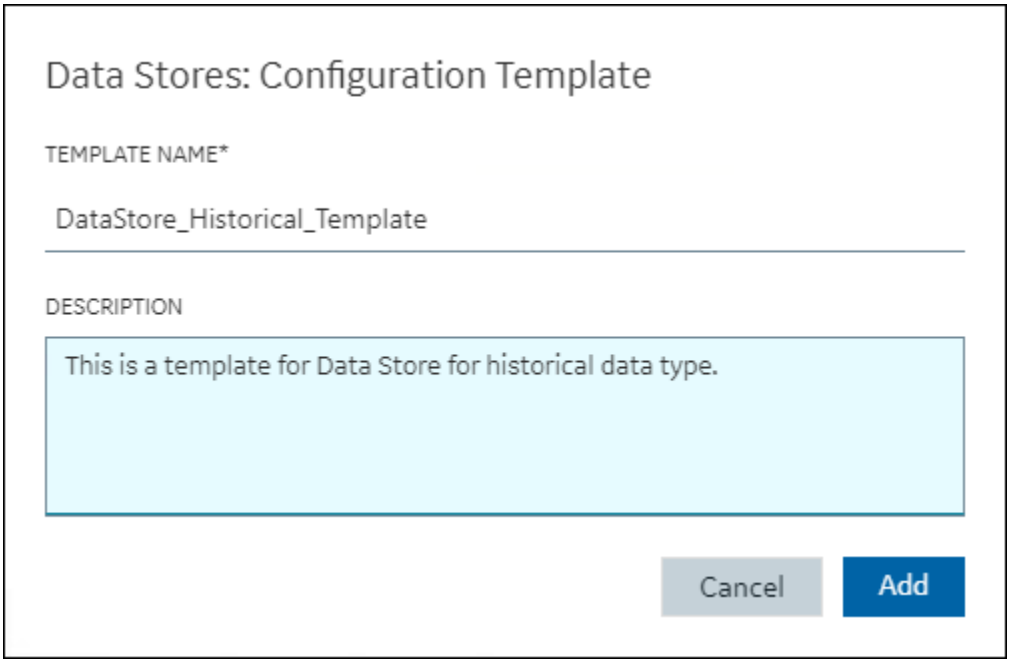
This topic describes how to create a configuration template for data stores.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Configuration Templates**.
3. Select the **Data Stores** tab.



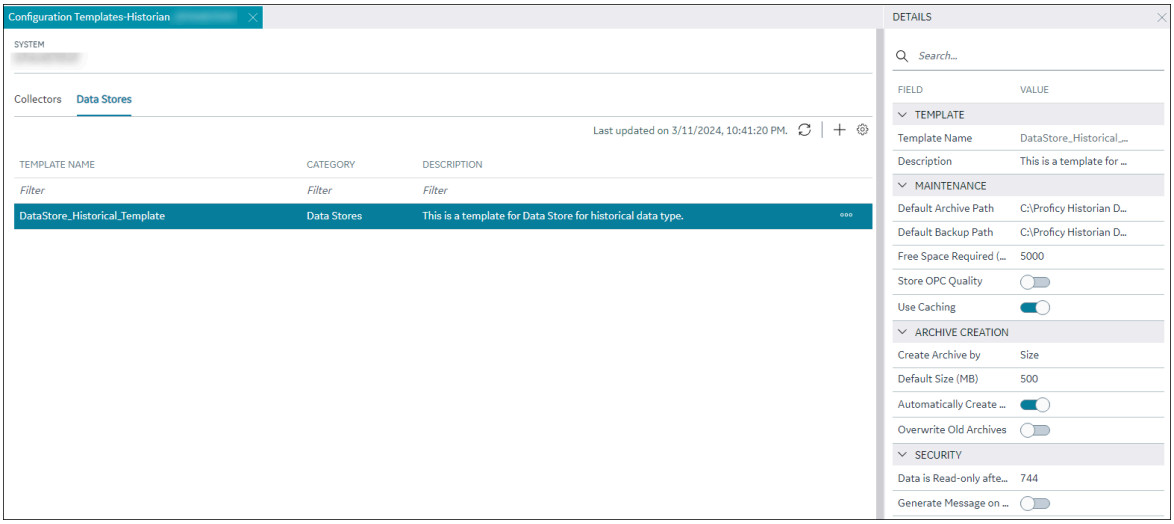

4. Select +.

   The **Data Stores:Configuration Template** window appears.




5. In **TEMPLATE NAME**, enter a name for the template. You can enter any alphanumeric names and also use special characters.
6. In **DESCRIPTION**, enter a description for your template. The description box can do a spell check while you enter the description.
7. Select **Add**.

   The configuration template is created.




8. Select the row that contains the created template.
   The data store details appear in the **DETAILS** section.

**Table 52. The Template Section**

| Field | Description |
|---|---|
| **Template Name** | The name of the template. This field is read-only. |
| **Description** | The description of the template. You can edit the description if needed. |

**Table 53. Maintenance**

| Field | Description |
|---|---|
| **Default Archive Path** | The default folder in which you want to create archives. |
| **Default Backup Path** | The default folder in which you want to place the backup archives. |
| **Free Space Required (MB)** | Indicates the remaining disk space required after a new archive is created. If the available space is less than the requirement, a new archive is not created. The default value is 5000 MB.<br><br>This field is not applicable to alarms and events archives. The alarms and events archiver will continue writing to the alarms and events archive until the drive is full. If this occurs, the alarms and events archiver will buffer incoming alarms and |

| Field | Description |
|---|---|
| | events data until the drive has free space. An error message is logged in the Historian message log. |
| **Store OPC Quality** | Indicates whether OPC data quality is stored. |
| **Use Caching** | Indicates whether caching is enabled. When reading data from the archiver, some data is saved in the system memory and retrieved using caching. This results in faster retrieval as the data is already stored in the buffer. |

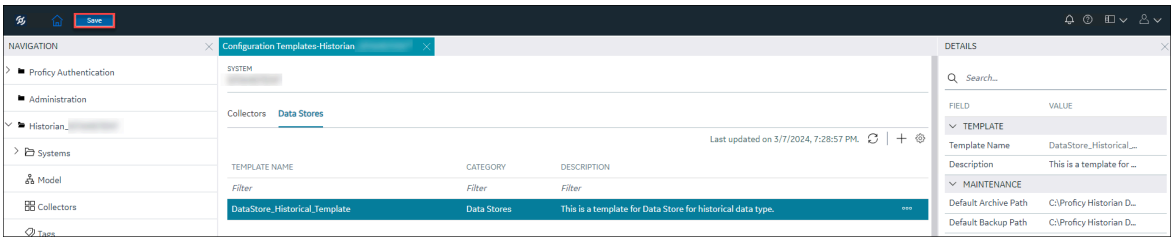
**Table 54. Archive Creation**

| Field | Description |
|---|---|
| **Create Archive by** | Indicates whether you want to create a new archive automatically after the current one reaches a specific size or after a specific duration. This field is enabled only if you switch the **Automatically Create Archives** toggle on.<br><br>Select one of the following options:<br>◦ **Size**: Select this option if you want to create a new archive when the current one reaches a specific size. Specify the size in the **Default Size (MB)** field (which appears only if you select **Size**).<br>◦ **Days** or **Hours**: Select one of these options if you want to create a new archive after a specific duration. Specify the duration in the **Archive Duration** field (which appears only if you select **Days** or **Hours**). |
| **Default Size (MB)** | The default size of an archive after which a new one will be automatically created if you switch the **Automatically Create Archives** toggle on. The **Default Size (MB)** field appears only if you select **Size** in the **Create Archive By** field. |
| **Automatically Create Archives** | Indicates whether you want to create an archive automatically *(on page 727)* after the current one is full. An archive file is considered full based on the size or duration you specify in the **Create Archive By** and the **Archive Duration** or **Default Size** fields. |

| Field | Description |
|---|---|
| **Overwrite Old Archives** | Indicates whether you want to overwrite an old archive file when a new one is created.<br><br>If you enable this option, the oldest archived data is replaced with the latest one when the latest archive default size is reached. Since this action deletes historical data, exercise caution in using this feature. Be sure that you have a backup of the archive so that you can restore it later. Best practice is to create an additional archive to prevent premature loss of data due to overwriting. For example, if you want to save 12 months of data into 12 archives, create 13 archives. |

**Table 55. Security**

| Field | Description |
|---|---|
| **Data is Read-Only After (Hours)** | The number of hours for data to be stored in a read/write archive. After the time lapses, that portion of the archive file is automatically made read-only. Incoming data values with time-stamps prior to this time are rejected. A single archive file, therefore, may have a portion made read-only, another portion that is read/write containing recently written data, and another that is unused free space. |
| **Generate Message on Data Update** | Indicates whether an audit log entry will be made any time the value of a previously archived data point is overwritten. This log entry will contain both the original and new values. |




9. After you edit the details as needed, in the upper-left corner, select **Save**.
   The configuration details are saved.

   You can select ••• and duplicate this template, edit, or delete it.

You can apply the configuration template to a data store *(on page 463)*.

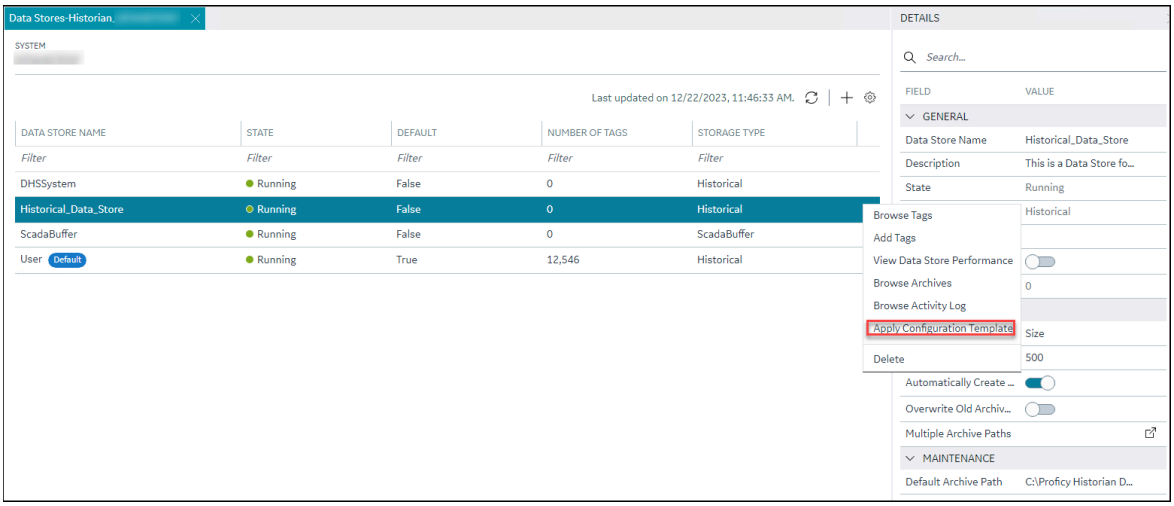
## Apply the Configuration Template to a Data Store

You can apply the created template to user-created data store as needed. You will be prompted to confirm whether you want to overwrite few of the configuration values with the values in the template.

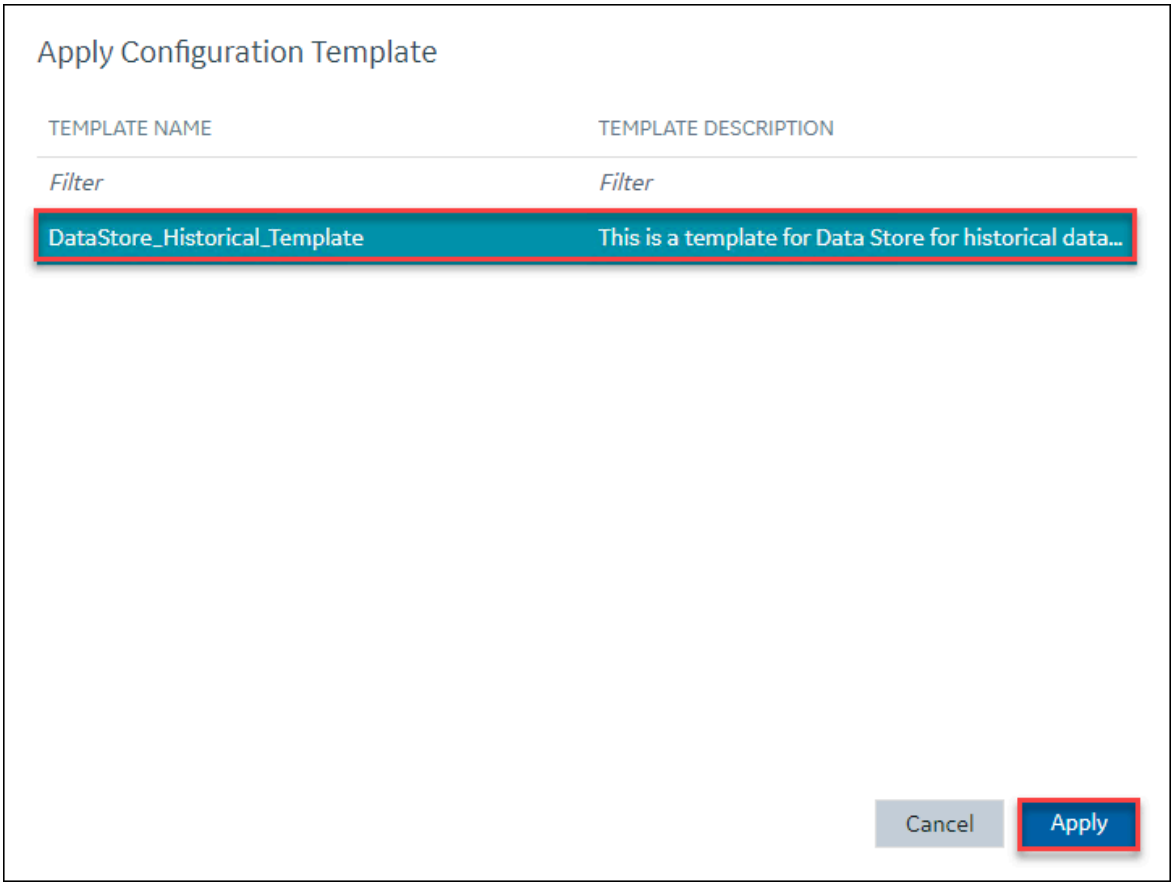
- Ensure that you have a data store created *(on page 370)*.
- Ensure that you have a configuration template for data stores *(on page 767)*.

This topic describes how to apply a data store configuration template to a data store. You can apply a data store configuration template to a user-created data store, provided they are not the default data store.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Data Stores**. The **Data Stores** section appears.
3. Right-click the data store (or select ), and then select **Apply Configuration Template**.




The **Apply Configuration Template** window appears, listing the available templates.




**Note:**
You can apply a data store configuration template to a user-created data store, provided they are not the default data store.

4. Select **Apply**.
   A confirmation window appears, prompting you to confirm whether you want to overwrite few of the configuration values with the values in the template.
5. Select **Ok**.
6. In the upper-left corner, select **Save**.
   The configurations in the template are applied to the data store.

# Managing Reports

## About Reports

Reports provide you with a concise summary of tag-specific data of a specific collector.

You can select a report from the available predefined templates and generate a report to view categorized data. For example, you can generate a report that lists tags with aliases and view the needed information. You can also export the generated report as a CSV file or save it as a PDF file.

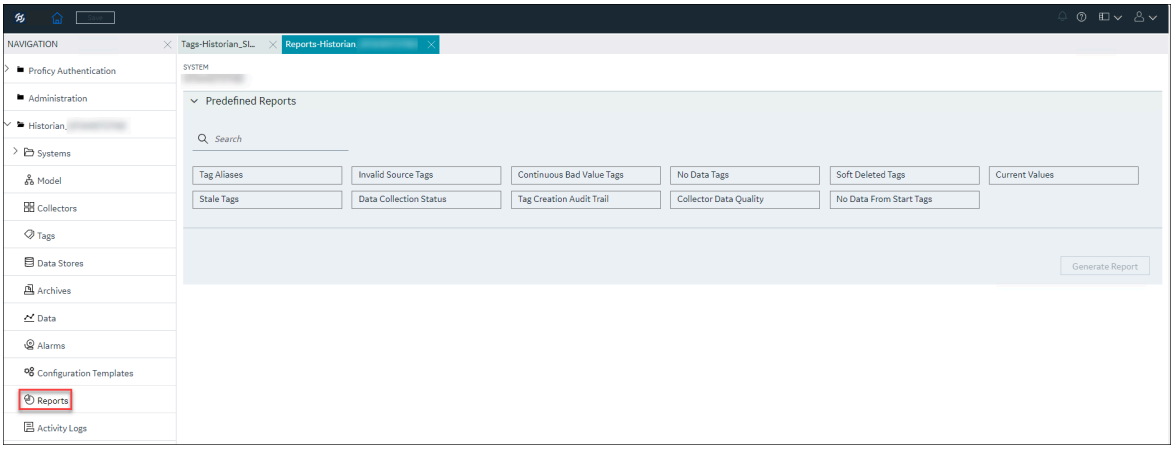
**Note:**
The reports are applicable only to the Historian destination collectors.

For more information on generating reports, refer to generate reports *(on page 774)*.

## Generate Reports

This topic describes how to generate report using a predefined template. You can select template from the available template options and generate a report.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Reports**.
   A list of predefined templates appear.




3. Select a template as needed.
   The corresponding input options appear, enabling you to select the collector and other options as needed. Based on your selections, you can generate the report.
4. Select the collector and other corresponding options.

   **Note:**
   To generate a report, you must select a collector.

   The below table lists different reports and their corresponding input options.

| Report | Description | Input Options | Data Available in the Report |
|---|---|---|---|
| Tag Aliases | This report displays tags that have aliases, along with the list of aliases for the selected collector instance. | ◦ COLLECTOR NAME- Name of the collector instance based on which the report must be generated. | ◦ TAG NAME |
| Invalid Source Tags | This report displays tags with a collector name but no configured source address for the selected collector instance.<br><br>**Note:** This report is not applicable for a File collector. | ◦ COLLECTOR NAME- Name of the collector instance based on which the report must be generated. | ◦ TAG NAME<br>◦ DESCRIPTION |
| Continuous Bad Value Tags | This report displays tags that are continuously returning bad data for the selected Time Interval for the collector instance. You can carefully analyze those tags and make decisions. | ◦ COLLECTOR NAME- Name of the collector instance based on which the report must be generated.<br>◦ TIME INTERVAL- The duration based on which the report must be generated. The available options are,<br>▪ 1 hour<br>▪ 8 hours<br>▪ 1 day<br>▪ 1 week<br>▪ 1 month | ◦ TAG NAME<br>◦ DESCRIPTION |

| Report | Description | Input Options | Data Available in the Report |
|---|---|---|---|
| | | ▪ Custom- Selecting this option enables a custom start and end date/time selector. You can select or enter a custom start and end date/ time based on which the report must be generated. | |
| No Data Tags | This report displays tags that are not writing any data for the selected collector instance. | ◦ COLLECTOR NAME- Name of the collector instance based on which the report must be generated.<br>◦ TIME INTERVAL- The duration based on which the report must be generated. The available options are,<br>▪ 1 hour<br>▪ 8 hours<br>▪ 1 day<br>▪ 1 week<br>▪ 1 month<br>▪ Custom- Selecting this option enables a cus- | ◦ TAG NAME<br>◦ DESCRIPTION<br>◦ VALUE<br>◦ TIME STAMP<br>◦ DATA COLLECTION |

| Report | Description | Input Options | Data Available in the Report |
|---|---|---|---|
| | | tom start and end date/time selector. You can select or enter a custom start and end date/time based on which the report must be generated. | |
| Soft Deleted Tags | This report displays tags that were removed from the system but still exist in Historian and are not being used for the selected collector instance. | ◦ COLLECTOR NAME- Name of the collector instance based on which the report must be generated.<br>◦ TIME INTERVAL- The duration based on which the report must be generated. The available options are,<br>▪ 1 hour<br>▪ 8 hours<br>▪ 1 day<br>▪ 1 week<br>▪ 1 month<br>▪ Custom- Selecting this option enables a custom start and end date/time selector. You can select or | ◦ TAG NAME<br>◦ DESCRIPTION<br>◦ DELETION TIME<br>◦ USER NAME |

| Report | Description | Input Options | Data Available in the Report |
|---|---|---|---|
| | | enter a custom start and end date/ time based on which the report must be generated.<br>◦ USERNAME- The name of the user based on which the report must be generated. | |
| Current Values | This report displays the current values of tags for the selected collector instance. | ◦ COLLECTOR NAME- Name of the collector instance based on which the report must be generated. | ◦ TAG<br>◦ ENGINEERING UNITS (DESCRIPTION)<br>◦ QUALITY<br>◦ VALUE<br>◦ TIME STAMP |
| Stale Tags | This report displays all the stale tags for the selected collector instance. | ◦ COLLECTOR NAME- Name of the collector instance based on which the report must be generated. | ◦ TAG NAME<br>◦ DESCRIPTION |
| Data Collection Status | This report displays the data collection status of tags for the selected collector instance. | ◦ COLLECTOR NAME- Name of the collector instance based on which the report must be generated.<br>◦ DATA COLLECTION- The status of da- | ◦ TAG NAME<br>◦ DESCRIPTION<br>◦ DATA COLLECTION |

| Report | Description | Input Options | Data Available in the Report |
|---|---|---|---|
| | | ta collection based on which the report must be generated. The available options are,<br>▪ On<br>▪ Paused. | |
| Tag Creation Audit Trail | This report displays information about tag creation time and created user. | ◦ COLLECTOR NAME- Name of the collector instance based on which the report must be generated.<br>◦ TIME INTERVAL- The duration based on which the report must be generated. The available options are,<br>▪ 1 hour<br>▪ 8 hours<br>▪ 1 day<br>▪ 1 week<br>▪ 1 month<br>▪ Custom- Selecting this option enables a custom start and end date/time selector. You can select or enter a custom start and end date/time based | ◦ TAG NAME<br>◦ DESCRIPTION<br>◦ CREATION TIME<br>◦ CREATED BY |

| Report | Description | Input Options | Data Available in the Report |
|---|---|---|---|
| | | on which the report must be generated.<br>◦ CREATION TIME- The time of creation based on which the report must be generated.<br>◦ CREATED USER- The name of the user based on which the report must be generated. | |
| Collector Data Quality | This report displays the data quality of tags for the selected collector instance. | ◦ COLLECTOR NAME- Name of the collector instance based on which the report must be generated. | ◦ COLLECTOR NAME<br>◦ DESCRIPTION<br>◦ TOTAL TAG COUNT<br>◦ BAD TAGS COUNT<br>◦ BAD TAGS PERCENT<br>◦ GOOD TAGS COUNT<br>◦ GOOD TAGS PERCENT |
| No Data From Start Tags | This report displays tags that do not have data from the time they were created for the selected collector instance. | ◦ COLLECTOR NAME- Name of the collector instance based on which the report must be generated. | ◦ TAG NAME<br>◦ DESCRIPTION |

5. Click **Generate Report**.
   The data corresponding to the selected report appears.

You can export the generated report as a CSV file *(on page 781)*, or save the generated report as a PDF file *(on page 781)*.

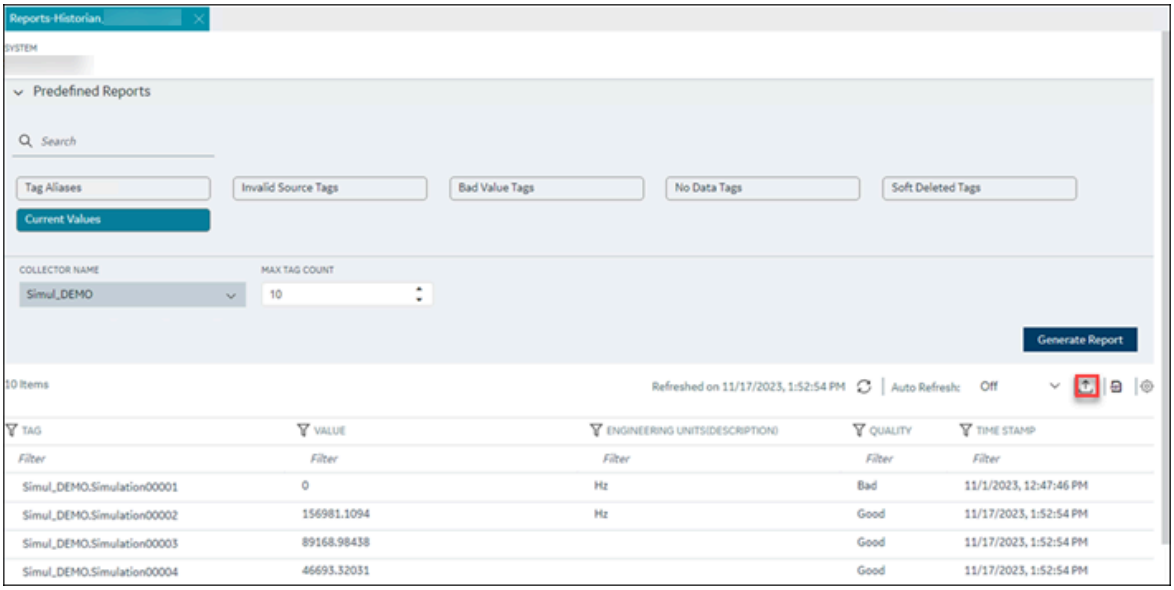
## Export the Generated Report as a CSV File

This topic describes how to export and save a report as CSV file.

Ensure that you have the following:

- Microsoft Excel or any text editor is needed to view the exported CSV file.

1. Generate the report *(on page 774)* as needed.
2. To export the report as a CSV file, in the upper-right corner of the grid, select .
   The report is exported as a CSV file.




   The file will be saved in the <System Name>_Reports_Export_Details_with_<ReportName> format. For example, **SYSTEMADMIN_Reports_Export_Details_with_CurrentValues**.

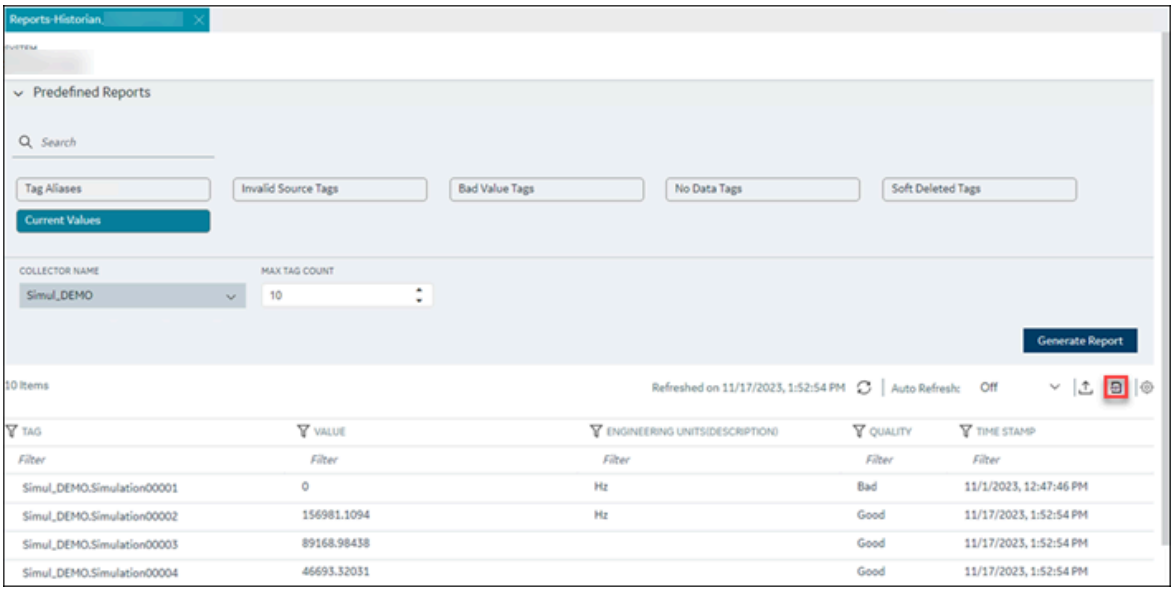
## Save the Generated Report as a PDF File

This topic describes how to save the report as a PDF file.

Ensure that you have the following:

- A PDF reader is required to open and view the report saved as a PDF file.

1. Generate the report *(on page 774)* as needed.
2. To save the report as a PDF file, in the upper-right corner of the grid, select .
   The report is saved as a PDF file.




The file will be saved in the Reports_<System Name>_<ReportName>_<MM_DD_YYYY, hh_mm_ss AM/PM> format. For example, **Reports_SYSTEMADMIN_CurrentValues_10_17_2023, 1_30_20 AM**.

# Access Activity Logs

Activity logs are generated when activities are performed in the Historian system.

Examples:

- When a tag is created, modified, or deleted
- When a collector instance is created, modified, or deleted
- When data collection for a tag or a collector begins or ends
- When an archive is created or will be closed soon

You can access these logs for each tag/collector or for all the tags and collectors in a system. You can filter these logs based on the start and end dates, priority, topics, and the content in the logs. You can also export all the logs or selected ones.

1. Access Configuration Hub *(on page 336)*.
2. If you want to access activity logs for all the tags and collectors in a system, in the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Activity Logs**.
   A list of activity logs appears.
3. If you want to access the activity logs of a collector instance:

   a. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.
   A list of collectors in the system appears.

   b. Right-click the collector whose activity log you want to access (or select ⋯), and then select **Browse Activity Log**.
   A list of activity logs for the collector appears.

4. If you want to access the activity logs of a tag:

   a. In the **NAVIGATION** section, select **Historian_<system_name> > Tags**.
   A list of tags in the system appears.

   b. Right-click the tag whose activity log you want to access (or select ⋯), and then select **Browse Activity Log**.
   A list of activity logs for the tag appears.

5. If you want to filter the list of activity logs:

   a. In the upper-right corner of the main section, select .

   b. Enter values as described in the following table.

| Field | Description |
|---|---|
| **START DATE/TIME** | Select the start date and time for the activity logs. |
| **END DATE/TIME** | Select the end date and time for the activity logs. |
| **PRIORITY** | Specify whether you want to see only alerts or messages or both. |
| **TOPIC** | Select the topic based on which you want to filter the logs:<br>▪ **Configuration**: Includes modifying a collector instance or a tag configuration.<br>▪ **Connections**: Includes system connections and creating a collector interface. |

| Field | Description |
|---|---|
| | ▪ **General**: Includes pausing or resuming data collection.<br>▪ **Performance**: Includes creating or closing an archive and moving buffer files.<br>▪ **ServiceControl**: Includes starting or stopping a collector interface.<br>This list is not comprehensive. |
| **ACTIVITY CONTAINS** | Enter the content of the logs based on which you want to filter them. |

c. Select **Apply**.
The logs are filtered based on the criteria.

If you want to export the logs, select . The logs are exported into a CSV file.

# Troubleshooting Configuration Hub

This topic contains solutions/workarounds to some of the common issues encountered with Configuration Hub. This list is not comprehensive. If the issue you are facing is not listed on this page, refer to Troubleshooting Web-based Clients *(on page 300)* and Troubleshooting the Historian Server *(on page 298)*.

### Unable to Access Configuration Hub After Upgrading Web-based Clients

**Workaround:** Clear your browser cache.

### Even after installing Web-based Clients, you cannot access Configuration Hub.

**Workaround:** Start the Proficy Operations Hub Httpd Reverse Proxy and the Data Archiver services.

### Unable to Access External Configuration Hub if Public Https Port is Different

**Issue:** During Web-based Clients installation, if you provide an existing Proficy Authentication and Configuration Hub details, and if the public https port numbers of these two machines do not match, you cannot access the external Configuration Hub from the current machine.

For example, suppose you have installed Web-based Clients on machine A, which points to the Proficy Authentication and Configuration Hub installed on machine B. If the public https port numbers of

machines A and B do not match, you cannot access Configuration Hub of machine B from machine A (although you can access it locally from machine B).

**Workaround:** Perform the following steps on the machine on which you have installed Configuration Hub (machine B):

1. Access the following folder: `C:\Program Files (x86)\GE\ConfigurationHub\Web\conf\confighub`
2. Access the file that contains the details of the machine from which you want to access external Configuration Hub (machine A). The file name begins with the host name of machine A.
3. In the line that contains the details of the Proficy Authentication server (for example, proxy_pass https://machine_B.Domain.com:Port/uaa/), change the port number to match the public https port number of machine B.
4. Save and close the file.
5. Restart the following services:
    - ConfigHubContainerService
    - ConfigHubNGINXService
    - ConfigHubStorageService

### Error Occurs When Historian Plugin is Registered with an External Configuration Hub

**Description:** If you install Configuration Hub using iFIX, then install Web-based Clients on another machine with local Proficy Authentication, and then register the Historian plugin with Configuration Hub, testing the connection to Configuration Hub fails. Even after you add the IP addresses of both the machines to the hosts file, the issue is not resolved.

**Error Message:** Error while getting token in ConfigAuth App

**Workaround:** Register the Historian plugin *(on page 186)* on the machine on which you have installed Web-based Clients, install the Proficy Authentication certificate *(on page 97)*, and then restart the browser.

### Cannot Access the Collectors Section or Add a Collector Instance

**Possible Cause: User credentials not provided while installing collectors or Remote Collector Manager:**

If installing collectors and the Historian server on the same machine, the collectors installer does not mandate the entry of username and password because it not required for Remote Collector Management Agent to connect to a local Historian server. But if Historian security or strict authentication is enabled, it is mandatory to enter the username and password.

**Workaround:**

- **Option 1:** Disable the strict collector authentication using Historian Administrator, and then restart the Historian Remote Collector Management Agent service.
- **Option 2:** Reinstall Remote Management Agents, providing the user credentials.

## Cannot Access or Add a System in Configuration Hub

**Possible Causes:**

- **User does not have security privileges:** During installation of the Historian server, if you have allowed the installer to create security groups, you must create a user with the name in the following format: <Proficy Authentication host name>.admin. Verify that this user has been created and added to the ihSecurityAdmins group.

  If the Proficy Authentication server hostname is long, resulting in a username longer than 20 characters, Windows does not allow you to create the user. In that case, you can create a Proficy Authentication user, and then create the corresponding Windows user, using the Proficy Authentication Configuration utility:

  1. Access the Proficy Authentication Configuration utility. By default, it is located at `C:\Program Files\GE Digital\Historian Config\uaa_config_tool`.
  2. Run the following command: `uaa_config_tool add_user -u <username> -p <password> -s <the client secret you provided while installing the Historian server> -c`

     Example: `uaa_config_tool add_user -u adminuser -p Password123 -s pwd@123 -c`
- **Incorrect Proficy Authentication details:** You must provide the host name of the Proficy Authentication server while installing the Historian server. In the `Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Intellution, Inc.\iHistorian\SecurityProviders\OAuth2` registry path, verify that the URI value is in the following format: `https://<Proficy Authentication host name>:<port number>/uaa/check_token`.

  If needed, modify the value, and then restart the Data Archiver service.

## Error Appears When Creating a Collector Instance

**Description:** When you add a collector instance, the following error message appears: `The server encountered an error processing the request. Please try again.` The collector instance is added successfully, but it does not appear in the **Collectors** section.

**Possible causes:** When you add a collector instance, the collector service is started and stopped so that it is connected to the Historian server. The collector then appears in the table in the **Collectors** section. Sometimes, however, the collector service does not respond to these commands on time, resulting in the

error message. If you attempt to add the same collector instance again, a message appears, stating that it exists.

**Workaround:** Select **Cancel**, and refresh the **Collectors** section. If the collector instance still does not appear, access the collector machine, and start the collector manually.

### Data Archiver is Shut Down

**Possible causes:** If there is insufficient disk space, the Data Archiver shuts down and a message is logged into the log file. By default, you can view the Historian archiver log file in C:\Historian Data \LogFiles.

**Workaround:** Allocate more disk space for archives, or remove old archives. You can also configure Historian to overwrite old archives automatically by switching the **Overwrite Old Archives** toggle on. If you enable this option, the oldest archived data is replaced with the latest one when the latest archive default size is reached. Since this action deletes historical data, exercise caution in using this feature. Be sure that you have a backup of the archive so that you can restore it later. Best practice is to create an additional archive to prevent premature loss of data due to overwriting. For example, if you want to save 12 months of data into 12 archives, create 13 archives.

For instructions, refer to Access a Data Store *(on page 453)*.

### Proficy Authentication and other Configuration Hub Plugins are not Visible in Configuration Hub, but Historian Plugin is Visible

**Description:**Consider that you used the Proficy Installer and installed common components such as Proficy Authentication and Configuration Hub. Then, you installed Historian Web components and registered Historian Plugin during the installation process. When you try to access Configuration Hub, the Historian Plugin is visible. However, the Plugins like Proficy Authentication, Administrator, and others are not visible. To resolve this issue, perform the below workaround.

**Workaround:**

1. From your desktop, select **Setup Authentication**.

   The **Configuration Hub Login** window appears.

2. In the bottom-right corner, select **Configure Confighub Authentication**.

   The **Configuration Hub Administrator Credentials** window appears.

3. Enter the Configuration Hub Client ID and Secret that were created while installing Configuration Hub.

4. Select **Verify**.

   On a successful verification, the **Register with Proficy Authentication** window appears.

5. In **SERVER NAME (FULLY QUALIFIED NAME)**, enter the server name in a Fully Qualified Domain (FQDN) format.
6. In **SERVER PORT**, enter the Proficy Authentication Server port, by default, it is 443.
7. Enter the Configuration Hub Client ID and Secret that were created while installing Configuration Hub.
8. Select **Register**.
9. Log in to Configuration Hub and see if the other Plugins are also displayed.

   On a successful registration, you will see all the other Plugins that were registered.

# Chapter 4. Remote Collector Management

## Overview of Remote Collector Management

Many Historian users use collectors to collect data from data sources or servers. Typically, these collectors are distributed geographically, and so, accessing them can be challenging and not cost-effective. To overcome this challenge, Historian provides the Remote Collector Management agent, using which you can manage collectors remotely.

**Advantages of using the Remote Collector Management agent:**

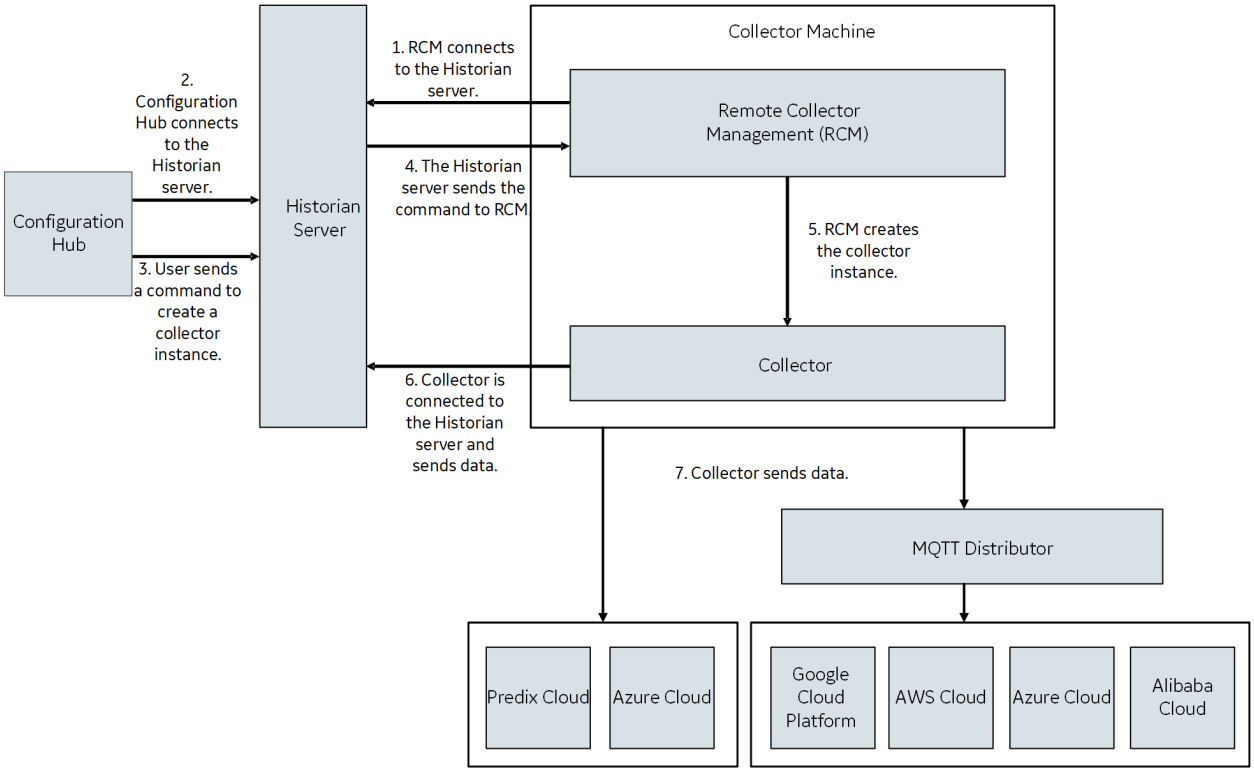
- Accessing a collector machine physically to manage the collector is no longer required.
- Security is enabled. That is, only members of the iH Security Admins, iH Tag Admins, and the iH Collector Admins security groups can manage the collectors remotely.
- Works with the older versions of collectors as well (V5.5 and later).

**Features**

- Add *(on page 797)*, modify *(on page 799)*, or delete *(on page 817)* a collector instance.
- Start *(on page 637)*, stop *(on page 638)*, or restart *(on page 639)* a collector.
- Pause *(on page 641)* or resume *(on page 642)* the data collection of a collector.
- Delete *(on page 643)* or move *(on page 644)* the buffer files of a collector.
- Change the destination server of a collector *(on page 645)*.

**Workflow**

The following diagram provides the workflow of Remote Collector Management when creating a collector instance. After the collector instance is created, the collector sends data to the configured destination. The green lines indicate the initial, one-time steps. The red lines indicate the steps performed every time you want to manage the collector remotely.




**Limitations**

- After installing Remote Management Agent, if you install a new collector, you must manually start it for the first time. This is to establish a connection between the collector and the Remote Collector Management agent. From the next time, you can manage the collector remotely.

# Installing Remote Management Agents

## Install Remote Management Agent Using the Installer

Ensure that all the collectors that you want to manage remotely are in the running state.

If the collectors that you have installed are earlier than version 9.0, you must install Remote Management Agent on each machine on which the collectors that you want to manage are installed. For collectors version 9.0 or later, Remote Management Agent are automatically installed when you install collectors.

This topic describes how to install Remote Management Agent using the installer. You can also install them at a command prompt *(on page 190)*.

1. Run the `InstallLauncher.exe` file.
2. Select **Install Remote Management Agents**.

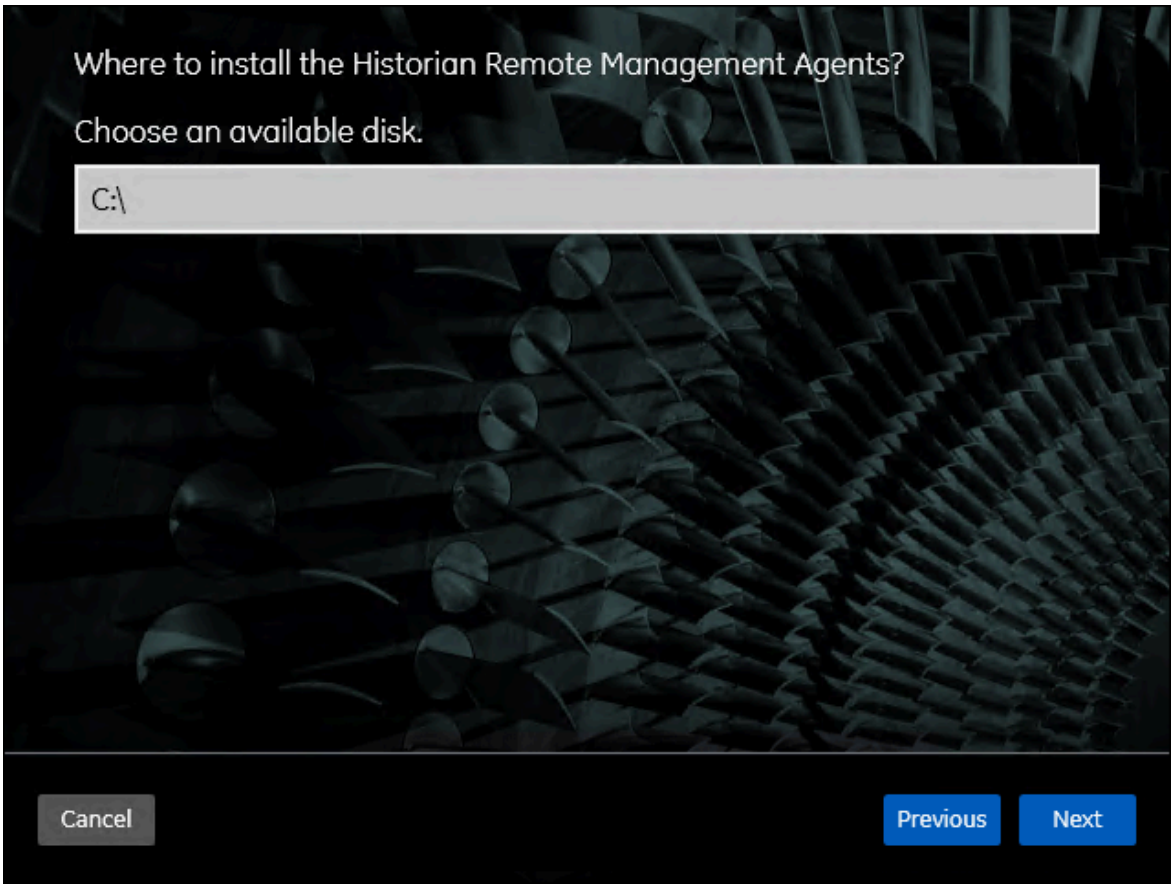
The welcome page appears.

3. Select **Next**.

   The license agreement appears.

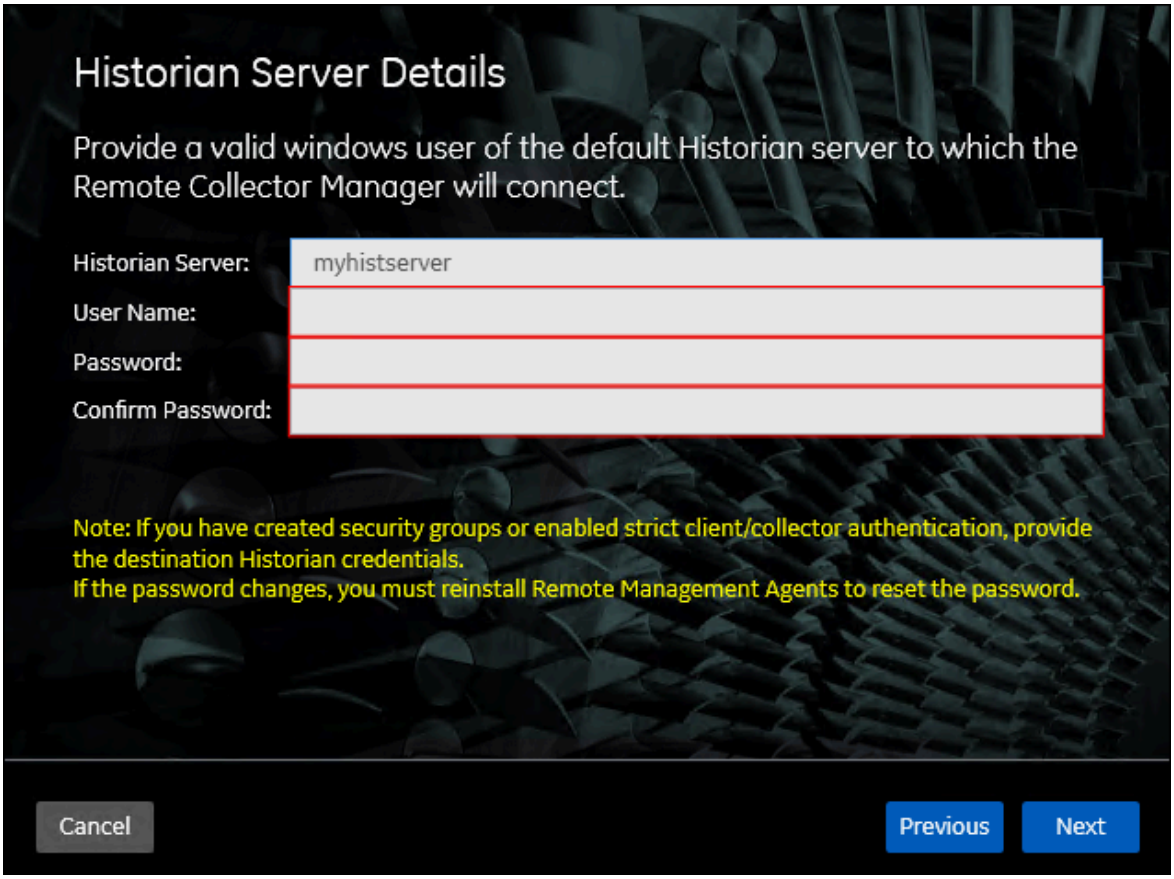
4. Select the **Accept** check box, and then select **Next**.

   The default installation drive appears.

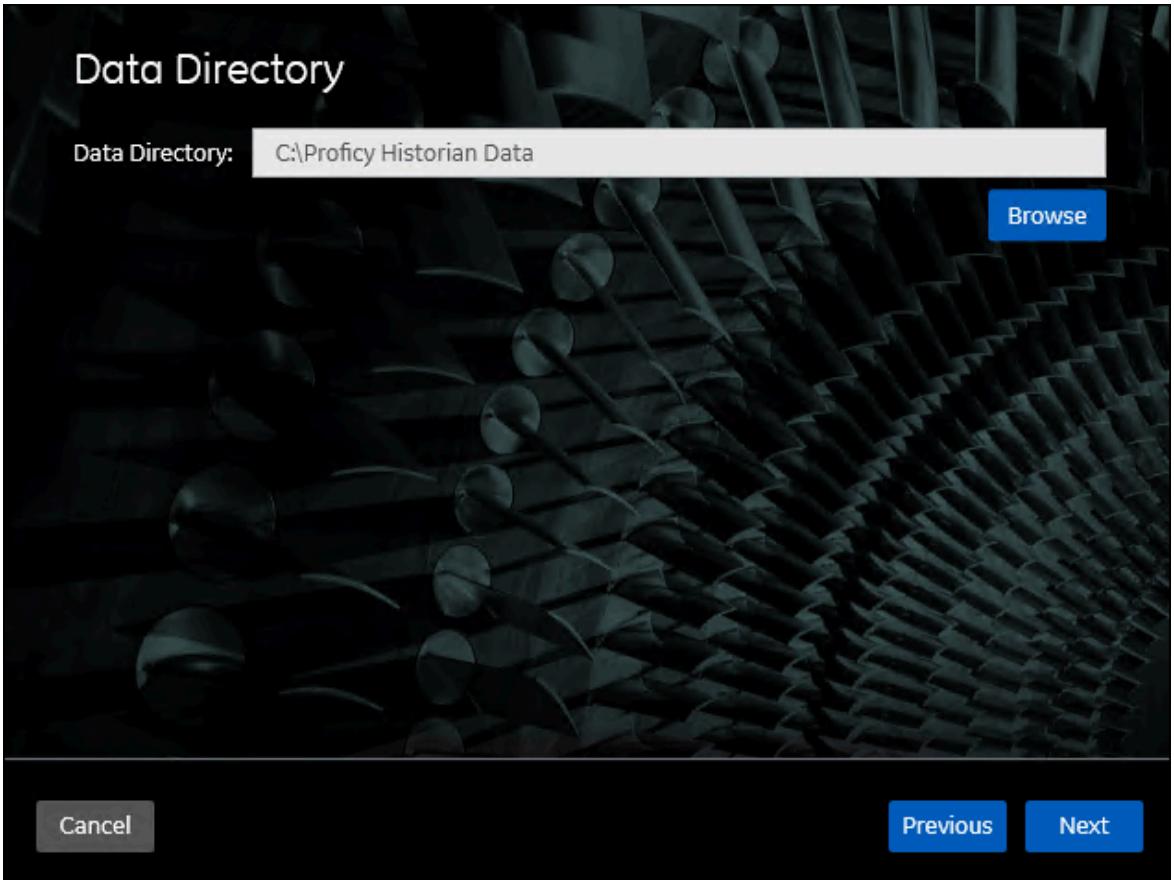



5. If needed, modify the installation drive, and then select **Next**.

   The destination Historian server page appears.




6. Enter the details of the Historian server to which Remote Management Agent will connect, and then select **Next**.
   The **Data Directory** page appears.




7. As needed, modify the location of the data directory, or leave the default value, and then select **Next**.
   The **You are ready to install** page appears.
8. Select **Install**.

- Remote Collector Management is installed on your machine.
- A folder named `Historian Remote Management Agents` is created in the `GE Digital` folder in the installation location that you specified.
- Remote Collector Management is running, and a .shw file is created in the log folder. This file contains the details of the collectors that are running on the machine.
- For each collector that you manage using Remote Collector Management, a new entry named ServiceName is created in the collector registry. If the ServiceName key is not created or updated incorrectly, refer to Troubleshooting Remote Collector Management Issues *(on page 820)*.

## Install Remote Management Agent at a Command Prompt

Ensure that all the collectors that you want to manage remotely are in the running state.

If the collectors that you have installed are earlier than version 9.0, you must install Remote Management Agent on each machine on which the collectors that you want to manage are installed. For collectors version 9.0 or later, Remote Management Agent are automatically installed when you install collectors.

This topic describes how to install Remote Management Agent at a command prompt. You can also install them using the installer *(on page 187)*.

1. Access the command prompt, and navigate to the `RMA` folder in the install media.
2. Run the following command, replacing the values in angular brackets with the appropriate values:

```
HistorianRMA_Install.exe -s RootDrive=<installation drive> DestinationServerName=<Destination Historian server name> UserName1=<Windows username> Password=<Windows password> DataPath="C:\Proficy Historian Data\LogFiles"
```

```
HistorianRMA_Install.exe -s RootDrive=C:\ UserName1=Administrator Password=AdminPassword DestinationServerName=VMHISTWEBAUTO DataPath="C:\Proficy Historian Data\LogFiles"
```

- Remote Collector Management is installed on your machine.
- A folder named `Historian Remote Management Agents` is created in the `GE Digital` folder in the installation location that you specified.
- Remote Collector Management is running, and a .shw file is created in the log folder. This file contains the details of the collectors that are running on the machine.
- For each collector that you manage using Remote Collector Management, a new entry named ServiceName is created in the collector registry. If the ServiceName key is not created or updated incorrectly, refer to Troubleshooting Remote Collector Management Issues *(on page 820)*.

# About Managing Collector Instances Using the RemoteCollectorConfigurator Utility

After you install Historian, you must install the collectors. These collectors are used to collect data from various sources and send it to Historian. For a list of collectors and their usage, refer to About Historian Data Collectors *(on page 1990)*.

After you install collectors and Remote Management Agent, the following artefacts will be available:

- **Executable files:** These files are required to add a collector instance.
- **Instances of the following collectors:**
    - The iFIX collector
    - The iFIX Alarms & Events collector
    - The OPC Classic Data Access collector for CIMPLICITY
    - The OPC Classic Alarms and Events collector for CIMPLICITY

These instances will be created only if iFIX and/or CIMPLICITY are installed on the same machine as the collectors.

- **The Remote Collector Management agent:** Provides the ability to manage collectors remotely.

You can then add a collector instance. This section describes how to add, modify, or delete a collector instance using the RemoteCollectorConfigurator utility. It is a System-API-based tool, which connects to the destination Historian server, and allows you to add, modify, and delete a collector instance without the need to install Web-based Clients. You can also perform these tasks using Configuration Hub *(on page 627)*.

You can use the RemoteCollectorConfigurator utility in one of the following ways:

- **Using Command Prompt:** In this method, you will enter a single command at a command prompt to run the RemoteCollectorConfigurator utility and provide values to all the required parameters.
- **Using the interactive UI of the RemoteCollectorConfigurator utility:** In this method, you will run the RemoteCollectorConfigurator utility, and use the on-screen instructions to manually provide values to all the required parameters. Interactive UI mode does not apply when Strict Authentication is enabled for Historian. Use the RemoteCollectorConfigurator.exe from the command-line in such cases. See the following example.

In both these methods, you can either enter the installation parameters and their values manually or provide a JSON file that contains them. The RemoteCollectorConfigurator utility can also create sample JSON files *(on page 796)*, which you can use to create or modify collector instances. In addition, you can run the RemoteCollectorConfigurator utility without connecting to Historian or a data archiver. This section describes how to use each of these methods.

**Tip:**
You can access the Help for the RemoteCollectorConfigurator utility by running the following command:

```
RemoteCollectorConfigurator.exe --help
```

**Example: RemoteCollectorConfigurator.exe from the Command line**

```
RemoteCollectorConfigurator.exe "HistorianServerName" "Userid" "Password" InterfaceCreateViaCmd
"{\"CollectorDestination\":\"Historian\",\"CollectorSystemName\":\"HistorianServerName\",
\"DestinationHistorian\":\"Historianservername\",\"DestinationHistorianUserName\":\"userid\",
\"DestinationHistorianPassword\":\"Gei321itc\",\"DataPathDirectory\":\"<datadrive>:\\Proficy\",
\"InterfaceName\": \"collector_name\",\"Type\":\"2\",\"mode\":\"1\"}"
```

Or:

```
<installation drive>:\Program Files\GE Digital
\NonWebCollectorInstantiationTool>RemoteCollectorConfigurator.exe "<<Historian Server>>"
"<<Userid>>" "<<Userpassword>>" InterfaceCreateViaFile "Inputjsonfile.json"
```

## Create a Sample JSON File

To add or modify a collector instance, you can provide a JSON file with the required details. Instead of manually creating the file, you can use the RemoteCollectorConfigurator utility to generate a sample JSON file. You can then modify the file as needed, and then use it to add or modify the collector instance.

1. Run the `RemoteCollectorConfigurator.exe` file. By default, it is located in the following folder: `C:\Program Files\GE Digital\NonWebCollectorInstantiationTool \RemoteCollectorConfigurator.exe`.
   A list of options to manage collector instances appears.
2. Enter `7`.
   A list of collector types appears, along with a number assigned to each of them. You are prompted to enter the collector type.
3. Specify the collector type by entering the corresponding number. For example, if you want to add an instance of the Calculation collector, enter `1`.
   A list of destinations appears, along with a number assigned to each of them.
4. Specify the destination by entering the corresponding number. For example, if the destination is a Historian server, enter `1`.
5. If needed, enter the folder path where you want the sample JSON file to be created. By default, the file will be created in the same folder in which the RemoteCollectorConfigurator utility is located.
   A sample JSON file is created.
6. As needed, update the sample JSON file with the required collector instance parameters *(on page 801)*. You can also specify values for the general parameters *(on page 814)*.

Add *(on page 797)* or modify *(on page 799)* the collector instance using the sample JSON file that you have created.

# Add a Collector Instance

- Install collectors *(on page 142)*.
- For an iFIX collector, if iFIX is not running in a Windows-service mode, an error occurs when you add the collector instance. Refer to About Adding an iFIX Collector Instance *(on page 501)* for expected behaviour and configuration recommendations.
- If the destination of a collector is an Azure IoT Hub device, ensure that the device is running.

Before you begin using a collector, you must add an instance of the collector. You can add multiple instances of the same collector or instances of multiple collectors where you have installed the collectors.

For an ODBC collector, a single mapping file is used by multiple instances.

**Note:**
When you install collectors, if iFIX and/or CIMPLICITY is installed on the same machine as the collectors, instances of the following collectors are created automatically:

- The iFIX collector
- The iFIX Alarms & Events collector
- The OPC Classic Data Access collector for CIMPLICITY
- The OPC Classic Alarms and Events collector for CIMPLICITY

You can begin using these collectors, or create more instances as needed.

This topic describes how to add a collector instance using the RemoteCollectorConfigurator utility. You can also add a collector instance using Configuration Hub *(on page 357)*. If you want to add an offline collector instance, refer to Add an Offline Collector Instance *(on page 819)*.

1. If you want to use an interactive UI:

   a. Run the `RemoteCollectorConfigurator.exe` file. By default, it is located in the following folder: `C:\Program Files\GE Digital \NonWebCollectorInstantiationTool`.
   A list of options to manage collector instances appears.

   b. Connect to the collector machine by entering `1` or `2`, depending on whether collectors are installed locally or on a remote machine.

   c. Enter `4`.

You are prompted to choose between entering the installation parameters manually and providing a JSON file.

d. If you want to manually enter the parameters and values, enter `1`, and then run the following command:

```
{"<parameter>":"<value>","<parameter>":"<value>"}
```

If you want to use a JSON file containing the installation parameters and values, enter `2`, and then enter the path to the JSON file that you have created. Instead of manually creating the JSON file, you can use the RemoteCollectorConfigurator utility to generate it automatically *(on page 796)*.

You can leave the Historian username and password blank if there are no Historian security user groups.

For information on the parameters, refer to Collector Instance Parameters *(on page 801)*.

2. If you want to use the Command Prompt window:

a. Access the installation folder of the RemoteCollectorConfigurator utility. By default, it is `C:\Program Files\GE Digital\NonWebCollectorInstantiationTool`.

b. Run Command Prompt in this location.

c. If you want to manually enter the installation parameters and values, run the following command:

```
RemoteCollectorConfigurator.exe "<Destination Historian>" "<Destination Historian username>"
"<Destination Historian password>" InterfaceCreateViaCmd "{\"<parameter>\":\"<value>\",
\"<parameter>\":\"<value>\"}"
```

If you want to use a JSON file containing the installation parameters and values, run the following command:

```
RemoteCollectorConfigurator.exe "<Destination Historian>" "<Destination Historian username>"
"<Destination Historian password>" InterfaceCreateViaFile "<path to the JSON file>"
```

Instead of manually creating the JSON file, you can use the RemoteCollectorConfigurator utility to generate it automatically *(on page 796)*.

If ih security groups are available, you must enter the Windows username and password of the destination Historian. If you have enabled the **Enforce Strict Collector Authentication**

option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:

- iH Security Admins
- iH Collector Admins
- iH Tag Admins

For information on the parameters, refer to Collector Instance Parameters *(on page 801)*.

The collector instance is added.

Specify the tags whose data you want to collect using the collector. For the CollectorDestination parameter:

- If you have entered Historian, access Historian Administrator, and manage the tag configuration. For information, refer to About Tags *(on page 659)*.
- If you not entered a value, modify the offline configuration file of the collector. By default, this file is available in the following location: `<installation folder of Historian>\GE Digital \<collector name>`. For information, refer to Creating Offline Configuration XML file *(on page 2040)*.

## Modify a Collector Instance

Stop the collector *(on page 1340)* whose instance you want to modify.

This topic describes how to modify the destination details of a collector instance using the RemoteCollectorConfigurator utility. You can also modify a collector instance using Configuration Hub *(on page 634)*.

1. If you want to use the Command Prompt window:

   a. Run the `RemoteCollectorConfigurator.exe` file. By default, it is located in the following folder: `C:\Program Files\GE Digital \NonWebCollectorInstantiationTool`.

   b. If you want to manually enter the installation parameters and values, run the following command:

```
RemoteCollectorConfigurator.exe "<Destination Historian>" "<Destination Historian username>"
"<Destination Historian password>" InterfaceEditDestinationViaCmd "{\"<parameter>\":\"<value>\",
\"<parameter>\":\"<value>\"}"
```

If you want to use a JSON file containing the installation parameters and values, run the following command:

```
RemoteCollectorConfigurator.exe "<Destination Historian>" "<Destination Historian username>"
"<Destination Historian password>" InterfaceEditDestinationViaFile "<path to the JSON file>"
```

Instead of manually creating the JSON file, you can use the RemoteCollectorConfigurator utility to generate it automatically *(on page 796)*.

You can leave the Historian username and password blank if there are no Historian security user groups.

For information on the parameters, refer to Collector Instance Parameters *(on page 801)*.

2. If you want to use an interactive UI:

   a. Run the `RemoteCollectorConfigurator.exe` file. By default, it is located in the following folder: `C:\Program Files\GE Digital \NonWebCollectorInstantiationTool`.
   A list of options to manage collector instances appears.

   b. Connect to the collector machine by entering `1` or `2`, depending on whether collectors are installed locally or on a remote machine.

   c. Enter `6`.
   You are prompted to choose between entering the installation parameters manually and providing a JSON file.

   d. If you want to manually enter the parameters and values, enter `1`, and then run the following command:

   ```
   {"<parameter>":"<value>","<parameter>":"<value>"}
   ```

   If you want to use a JSON file containing the installation parameters and values, enter `2`, and then enter the path to the JSON file that you have created. Instead of manually creating the JSON file, you can use the RemoteCollectorConfigurator utility to generate it automatically *(on page 796)*.

   You can leave the Historian username and password blank if there are no Historian security user groups.

   For information on the parameters, refer to Collector Instance Parameters *(on page 801)*.

The destination of the collector instance is modified.

## Collector Instance Parameters

This topic provides a list of the parameters that you must provide when you add or modify a collector instance.

**Table 56. Destination: Historian**

| Parameter | Description |
|---|---|
| CollectorDestination | The type of the configuration to specify the tags whose data you want to collect.<br>• If you want to use Historian Administrator to specify the tags for data collection, enter `Historian`. For information, refer to About Tags *(on page 659)*.<br>• If you want to specify the tags using an offline tag configuration file, do not enter a value. By default, this file is available in the following location: `<installation folder of Historian>\GE Digital\<collector name>`. For information, refer to Creating Offline Configuration XML file *(on page 2040)*. |
| winUserName | The username to connect to the machine on which Historian Administrator is installed. A value is required only if:<br>• You want to use Historian Administrator to specify the tags for data collection.<br>• Historian security groups are used.<br>• The mode is set to 2. |
| winPassword | The password to connect to the machine on which Historian Administrator is installed. A value is required only if: |

**Table 56. Destination: Historian (continued)**

| Parameter | Description |
|---|---|
| | • You want to use Historian Administrator to specify the tags for data collection.<br>• Historian security groups are used.<br>• The mode is set to 2. |
| CollectorSystemName | The name of the machine on which you have installed the collectors. A value is required. |
| InterfaceName | The interface name of the collector instance. A value is required and must be unique per destination. The following characters are not allowed in the interface name:<br><br>• < (less than)<br>• > (greater than)<br>• : (colon)<br>• " (double quote)<br>• / (forward slash)<br>• \ (backslash)<br>• \| (vertical bar or pipe)<br>• ? (question mark)<br>• * (asterisk) |
| InterfaceDescription | The description of the collector instance. |
| InterfaceSubType | The subtype of the collector. For information, refer to Collector Type and Subtype *(on page 1339)*. |
| Type | The type of the collector. For information, refer to Collector Type and Subtype *(on page 1339)*. A value is required. |
| DataPathDirectory | The folder in which you want to store the collector log files. If you do not enter a value, by default, `C:\\Proficy Historian Data` is considered. |
| mode | Identifies whether creating the collector instance requires a specific user account credentials. Enter one of the following values: |

**Table 56. Destination: Historian (continued)**

| Parameter | Description |
|---|---|
| | • 1: Creates the collector instance with the credentials of the local user.<br>• 2: Creates the collector instance with the credentials of a specific user. If you choose this mode, you must enter values for the winUserName and winPassword parameters. |

## Installation parameters for an iFIX collector to send data to Historian

```
{
                "mode": 2,
                "CollectorSystemName": "<host name>",
                "InterfaceDescription": "iFIX collector for unit 1",
                "DataPathDirectory": "C:\\Proficy Historian Data",
                "CollectorDestination": "Historian",
                "winUserName": "<host name>\\<user name>",
                "winPassword": "<password>",
                "InterfaceSubType": "",
               "DestinationHistorianUserName": "<user name>",
                "DestinationHistorianPassword": "<password>",
                "DestinationHistorian": "<host name>",
                "General1": "",
                "General2": "",
                "General3": "FIX",
                "General4": "",
                "General5": "",
                "Type": 1,
                "InterfaceName": "collector_unique_name"
}
```

**Table 57. Destination: Predix TimeSeries**

| Parameter | Description |
|---|---|
| ClientID | The collector when interacting with Predix Time Series. This is equivalent to the username in many authentication schemes. The client must exist in |

**Table 57. Destination: Predix TimeSeries (continued)**

| Parameter | Description |
|---|---|
| | the Proficy Authentication instance identified by the identity issuer, and the system requires that the `timeseries.zones. {ZoneId}.ingest` and `time-series.zones.{ZoneId}.query` authorities are granted access to the client for the Predix Zone ID specified. Your Predix Time Series administrator can provide this information. |
| ClientSecret | The secret to authenticate the collector. This is equivalent to the password in many authentication schemes. |
| CloudDestinationAddress | The URL of a data streaming endpoint exposed by the Predix Time Series instance to which you want to send data. Typically, it starts with “wss://”. This value is used as part of the interface name and default tag prefix of the collector. Your Predix Time Series administrator can provide this URL. |
| CollectorDestination | The type of the cloud destination. For Predix TimeSeries, enter `Predix`. |
| CollectorSystemName | The name of the machine on which you have installed the collectors. A value is required. |
| DataPathDirectory | The folder in which you want to store the collector log files. If you do not enter a value, by default, `C:\ \Proficy Historian Data` is considered. |
| DatapointAttribute<*number*> | The attributes for each data point whose values you want the collector to collect. You can specify maximum five attributes. |
| IdentityIssuer | The URL of an authentication endpoint for the collector to authenticate itself and acquire necessary credentials to stream to the Predix Time Series. In other words, this is the issuer ID of the Proficy Authentication instance that you want to use to con- |

**Table 57. Destination: Predix TimeSeries (continued)**

| Parameter | Description |
|---|---|
| | nect to Predix Time Series. Typically, it starts with https:// and ends with “/oauth/token”. |
| InterfaceName | The interface name of the collector instance. A value is required and must be unique per destination. The following characters are not allowed in the interface name:<br>• < (less than)<br>• > (greater than)<br>• : (colon)<br>• " (double quote)<br>• / (forward slash)<br>• \ (backslash)<br>• \| (vertical bar or pipe)<br>• ? (question mark)<br>• * (asterisk) |
| InterfaceDescription | The description of the collector instance. |
| InterfaceSubType | The subtype of the collector. For information, refer to Collector Type and Subtype *(on page 1339)*. |
| Proxy | Identifies the URL of the proxy server to be used for both the authentication process and for sending data. If the collector is running on a network where proxy servers are used to access web resources outside of the network, then you must provide the proxy server settings. However, it does not affect the proxy server used by Windows when establishing secure connections. As a result, you must still configure the proxy settings for the Windows user account under which the collector service runs. |
| ProxyUserName | The username to connect to the proxy server. |
| ProxyPassword | The password to connect to the proxy server. |

**Table 57. Destination: Predix TimeSeries (continued)**

| Parameter | Description |
|---|---|
| Type | The type of the collector. For information, refer to Collector Type and Subtype *(on page 1339)*. A value is required. |
| ZoneID | Unique identifier of the instance to which the collector will send data. |
| winUserName | The username to connect to the machine on which Historian Administrator is installed. A value is required only if:<br>• You want to use Historian Administrator to specify the tags for data collection.<br>• Historian security groups are used.<br>• The mode is set to 2. |
| winPassword | The password to connect to the machine on which Historian Administrator is installed. A value is required only if:<br>• You want to use Historian Administrator to specify the tags for data collection.<br>• Historian security groups are used.<br>• The mode is set to 2. |
| mode | Identifies whether creating the collector instance requires a specific user account credentials. Enter one of the following values:<br>• 1: Creates the collector instance with the credentials of the local user.<br>• 2: Creates the collector instance with the credentials of a specific user. If you choose this mode, you must enter values for the winUserName and winPassword parameters. |

## Installation parameters for an iFIX collector to send data to Predix TimeSeries

```
{
                "ClientID": "HistQA",
                "ClientSecret": "1234",
                "CloudDestinationAddress": "wss://abcd.run.123.predix.io/v1/stream/messages",
                "CollectorDestination": "Predix",
                "CollectorSystemName": "<host name>",
                "DataPathDirectory": "C:\\Proficy Historian Data",
                "DatapointAttribute1":"\"site\":\"site_1\"",
                "DatapointAttribute2": "",
                "DatapointAttribute3": "",
                "DatapointAttribute4": "",
                "DatapointAttribute5": "",
                "DestinationHistorian": "<host name>",
                "General1": "",
                "General2": "",
                "General3": "abc",
                "General4": "",
                "General5": "",
                "IdentityIssuer": "https://1234567.predix-uaa.run.aws-usw02-pr.ice.predix.io/oauth/token",
                "InterfaceDescription": "1234",
                "InterfaceName": "123",
                "InterfaceSubType": "",
                "Proxy": "http://<host name>:<port number>",
                "ProxyPassword": "",
                "ProxyUserName": "",
                "Type": 1,
                "ZoneID": "123-456-789de-rft",
                "winPassword": "",
                "winUserName": "",
                "mode": 1
}
```

**Table 58. Destination: MQTT**

| Parameter | Description |
|---|---|
| ClientID | The name of the MQTT client. A value is required and must be unique for an MQTT broker. |
| CollectorDestination | The type of the cloud destination. For MQTT, enter `MQTT`. |
| CollectorSystemName | The name of the machine on which you have installed the collectors. A value is required. |
| DataPathDirectory | The folder in which you want to store the collector log files. If you do not enter a value, by default, `C:\ \Proficy Historian Data` is considered. |
| DeviceSharedKey | The device shared key of the MQTT broker. |
| HostAddress | The host name of the MQTT broker to which you want the collector to send data. A value is required. |
| HostPort | The port number to connect to the MQTT broker to which you want the collector to send data. |
| InterfaceName | The interface name of the collector instance. A value is required and must be unique per destination. The following characters are not allowed in the interface name:<br>• < (less than)<br>• > (greater than)<br>• : (colon)<br>• " (double quote)<br>• / (forward slash)<br>• \ (backslash)<br>• \| (vertical bar or pipe)<br>• ? (question mark)<br>• * (asterisk) |
| InterfaceDescription | The description of the collector instance. |
| InterfaceSubType | The subtype of the collector. For information, refer to Collector Type and Subtype *(on page 1339)*. |

**Table 58. Destination: MQTT (continued)**

| Parameter | Description |
|---|---|
| MQTTAutoRefresh | Indicates that the password is automatically generated on expiry; you are not required to provide the password. |
| MQTTCloudSubtype | The subtype of the MQTT broker. |
| MQTTUserName | Enter the username to connect to the MQTT broker. |
| MQTTPassword | Enter the password to connect to the MQTT broker. |
| Topic | The MQTT topic to which you want the collector to publish data. |
| Type | The type of the collector. For information, refer to Collector Type and Subtype *(on page 1339)*. A value is required. |
| winUserName | The username to connect to the machine on which Historian Administrator is installed. A value is required only if:<br>• You want to use Historian Administrator to specify the tags for data collection.<br>• Historian security groups are used.<br>• The mode is set to 2. |
| winPassword | The password to connect to the machine on which Historian Administrator is installed. A value is required only if:<br>• You want to use Historian Administrator to specify the tags for data collection.<br>• Historian security groups are used.<br>• The mode is set to 2. |
| mode | Identifies whether creating the collector instance requires a specific user account credentials. Enter one of the following values: |

**Table 58. Destination: MQTT (continued)**

| Parameter | Description |
|---|---|
| | • 1: Creates the collector instance with the credentials of the local user.<br>• 2: Creates the collector instance with the credentials of a specific user. If you choose this mode, you must enter values for the winUserName and winPassword parameters. |

**Tip:**
To establish an MQTT connection with Alibaba Cloud, refer to https://www.alibabacloud.com/help/doc-detail/73742.htm. To generate a password to connect to Alibaba Cloud, use the utility located here.

## Installation parameters for an iFIX collector to send data to Google Cloud

```
{
                "InterfaceName": "<unique collector name>",
                "InterfaceDescription": "collector for unit 3",
                "Type": 1,
                "mode": 2,
                "CollectorSystemName": "<host name>",
                "DataPathDirectory": "C:\\Proficy Historian Data",
                "CollectorDestination": "MQTT",
                "HostAddress": "mqtt.googleapis.com",
                "HostPort": "8883",
                "ClientID": "projects/mygcpproject/locations/asia-east1/registries/testmqttgcpiot/devices/testdevice",
                "Topic": "/devices/gcptesting/events",
                "DeviceSharedKey": "",
                "MQTTCloudSubtype": "GOOGLE",
                "MQTTUserName": "testusername",
                "MQTTPassword": "testGoogleConnectiionstringPassword",
                "MQTTAutoRefresh": "NO",
                "MQTTCAFile": "",
                "MQTTCertificateFile": "",
                "MQTTPrivateKeyFile": "",
                "MQTTPublicKeyFile": "",
```

```
                "winUserName": "<host name>\\Admin",
                "winPassword": "<password>",
                "InterfaceSubType": "",
                "DestinationHistorianUserName": "<Windows user name of the destination>",
                "DestinationHistorianPassword": "<Windows password of the destination>",
                "DestinationHistorian": "<host name>",
                "General1": "",
                "General2": "",
                "General3": "FIX",
                "General4": "",
                "General5": ""
}
```

**Table 59. Destination: Azure IoT Hub**

| Parameter | Description |
|---|---|
| CollectorDestination | The type of the cloud destination. For Azure IoT Hub, enter `Azure`. |
| CollectorSystemName | The name of the machine on which you have installed the collectors. A value is required. |
| DataPathDirectory | The folder in which you want to store the collector log files. If you do not enter a value, by default, `C:\\Proficy Historian Data` is considered. |
| DeviceConnectionString | Identifies the Azure IoT device to which you want to send data. Enter a value in the following format: `HostName=<value>;DeviceId=<value>;SharedAccessKey=<value>` |
| DeviceId | The ID of the Azure IoT device. |
| SharedAccessKey | The shared access key of the device. |
| InterfaceName | The interface name of the collector instance. A value is required and must be unique per destination. The following characters are not allowed in the interface name:<br>• < (less than)<br>• > (greater than) |

**Table 59. Destination: Azure IoT Hub (continued)**

| Parameter | Description |
|---|---|
| | • : (colon)<br>• " (double quote)<br>• / (forward slash)<br>• \ (backslash)<br>• \| (vertical bar or pipe)<br>• ? (question mark)<br>• * (asterisk) |
| InterfaceDescription | The description of the interface. |
| InterfaceSubType | The subtype of the collector. For information, refer to Collector Type and Subtype *(on page 1339)*. |
| Proxy | Identifies the URL of the proxy server to be used for both the authentication process and for sending data. If the collector is running on a network where proxy servers are used to access web resources outside of the network, then you must provide the proxy server settings. However, it does not affect the proxy server used by Windows when establishing secure connections. As a result, you must still configure the proxy settings for the Windows user account under which the collector service runs. |
| ProxyUserName | The username to connect to the proxy server. |
| ProxyPassword | The password to connect to the proxy server. |
| TrasportProtocol | The protocol that you want to use to send data to Azure IoT Hub. Enter one of the following values:<br><br>• HTTP<br>• MQTT<br>• AMQP<br>• MQTT_OVER_WEBSOCKETS<br>• AMQP_OVER_WEBSOCKETS |

**Table 59. Destination: Azure IoT Hub (continued)**

| Parameter | Description |
|---|---|
| | For information on which protocol to use, refer to Protocols and Port Numbers *(on page 626)*. |
| Type | The type of the collector. For information, refer to Collector Type and Subtype *(on page 1339)*. A value is required. |
| winUserName | The username to connect to the machine on which Historian Administrator is installed. A value is required only if:<br>• You want to use Historian Administrator to specify the tags for data collection.<br>• Historian security groups are used.<br>• The mode is set to 2. |
| winPassword | The password to connect to the machine on which Historian Administrator is installed. A value is required only if:<br>• You want to use Historian Administrator to specify the tags for data collection.<br>• Historian security groups are used.<br>• The mode is set to 2. |
| mode | Identifies whether creating the collector instance requires a specific user account credentials. Enter one of the following values:<br>• 1: Creates the collector instance with the credentials of the local user.<br>• 2: Creates the collector instance with the credentials of a specific user. If you choose this mode, you must enter values for the winUserName and winPassword parameters. |

### Installation parameters for an iFIX collector to send data to Azure IoT Hub

```
{
                "InterfaceName": "collector_unique_name",
                "InterfaceDescription": "iFIX collector for unit 2",
                "Type": 1,
                "mode": 2,
                "CollectorSystemName": "<host name>",
                "DataPathDirectory": "C:\\Proficy Historian Data",
                "CollectorDestination": "Azure",
                "DeviceConnectionString": "HostName=abc.azure-devices.net;DeviceId=Device1;SharedAccessKey=xxxxxxxxxx",
                "TransportProtocol": "AMQP_OVER_WEBSOCKETS",
                "Proxy": "<host name>:<port number>",
                "ProxyUserName": "",
                "ProxyPassword": "",
                "winUserName": "<host name>\\<user name>",
                "winPassword": "<password>",
                "InterfaceSubType": "",
               "DestinationHistorianUserName": "<user name>",
                "DestinationHistorianPassword": "<password>",
                "DestinationHistorian": "<host name>",
                "General1": "",
                "General2": "",
                "General3": "FIX",
                "General4": "",
                "General5": ""
}
```

## General Parameters of a Collector

This topic provides a list of general parameters that are applicable to each type of collector.

| Collector Type | Applicable General Parameters |
|---|---|
| The Calculation collector | • General1 - optional. Used for calculation timeout (sec). Default value: 10.<br>• General2 - optional. Used for Max Recovery Time (hr). Default value: 4 |

| Collector Type | Applicable General Parameters |
|---|---|
| The CygNet collector | • General2 - optional. Used for recovery time (hr). Default value: 0<br>• General3 - optional. Used for thread count. Default value: 5<br>• General4 - optional. Used for the Cygnet debug mode. Default value: 0<br>• General5 - optional. Used for optimization. Default value: 1 |
| The iFIX Alarms and Events collector | General1 - optional. Used for ProgId. Default value: Proficy.OPCiFIXAE.1 |
| The iFIX collector | • General3 - optional<br>• General4 - optional. Used for blocks and fields for blocks. Default value: AI:F_CV,B_CUALM |
| The MQTT collector | • General1 - required. Used for the source host name.<br>General2 - required. Used for the source topic.<br>General3 - required. Used for the source port. |
| The ODBC collector | • General1 - required. Used for the ODBC server.<br>• General2 - required. Used for the ODBC username.<br>• General3 - required. Used for the ODBC password.<br>• General4 - optional. Used for recovery time (hr). Default value: 0.<br>• General5 - optional. Used for throttle (milliseconds). Default value: 100 |
| The OPC Classic Alarms and Events collector | General1 - required. Used for the OPC source server progID. |

| Collector Type | Applicable General Parameters |
|---|---|
| The OPC Classic DA collector | General1 - required. Used for the OPC source server progID. |
| The OPC Classic HDA collector | • General1 - required. Used for the OPC HDA server.<br>• General2 - optional. Used for recovery time (hr). Default value: 24 |
| The OPC UA DA collector | • General1 - required. Used for the OPC UA server URI.<br>• General2 - optional. Used for secured connectivity. Default value: false<br>• General3 - optional. Used to enable user security. Default value: false<br>• General4 - optional. Used for username when security is enabled.<br><br>General5 - optional. Used for password when security is enabled. |
| The OSI PI collector | • General1 - required. Used for the OSI PI server name.<br>• General2 - optional. Used for the OSI PI username. Default value: piadmin<br>• General3 - optional. Used for the OSI PI password.<br>• General4 - optional. Used for max recovery time (hr). Default value: 4<br>• General5 - optional. Used for the OSI PI source (archive or snapshot). Default value: Archive. |
| The OSI PI distributor | • General1 - optional. Used for the OSI PI server.<br><br>General2 - optional. Used for the OSI PI username. Default value: piadmin<br>• General3 - optional. Used for the OSI PI password. |

| Collector Type | Applicable General Parameters |
|---|---|
| | • General4 - optional. Used for max recovery time (hr). Default value: 4<br>• General5 - optional. Used for the OSI PI source (archive or snapshot). Default value: Archive. |
| The Python Collector | • General1 - optional. Used for calculation timeout (sec). Default value: 10<br>• General2 - optional. Used for max recovery time (hr). Default value: 4 |
| The Server-to-Server distributor | • General1 - optional. Used for calculation timeout (sec). Default value: 10<br>• General2 - optional. Used for max recovery time (hr). Default value: 4<br>• General3 - required. Used for the source server name<br>• General4 - optional. Used for message replication (0 or 1) and alarm replication (0 or 1). Enter 0 or 1 only for alarm replication.<br>• General5 - optional. Used for prefix to messages. |
| The Windows Performance collector | None |
| The Wonderware collector | • General1 - required. Used for the Wonderware server.<br>• General2 - required. Used for the Wonderware username.<br>• General3 - required. Used for the Wonderware password.<br>• General4 - optional. Used for the recovery time (hr). Default value: 0<br>• General5 - optional. Used for Throttle (milliseconds). Default value: 100 |

## Delete a Collector Instance

Stop the collector *(on page 1340)* whose instance you want to delete.

If you no longer want to use a collector instance to collect data, you can delete it. When you delete a collector instance, the Windows service for the collector, the Registry folder, and the buffer files are deleted as well.

This topic describes how to delete a collector instance using the RemoteCollectorConfigurator utility. You can also delete a collector instance using Configuration Hub *(on page 654)*. If you want to delete an offline collector, refer to Delete an Offline Collector Instance *(on page 819)*.

1. If you want to use the Command Prompt window:

    a. Access the installation location of the RemoteCollectorConfigurator utility. By default, it is `C:\Program Files\GE Digital\NonWebCollectorInstantiationTool`.

    b. Run the following command:

    ```
    RemoteCollectorConfigurator.exe "<Destination Historian>" "<Destination Historian username>"
    "<Destination Historian password>" InterfaceDelete <interface name> ShouldDeleteTags[<0 or 1>]
    ```

    You can leave the Historian username and password blank if there are no Historian security user groups.

2. If you want to use an interactive UI:

    a. Run the `RemoteCollectorConfigurator.exe` file. By default, it is located in the following folder: `C:\Program Files\GE Digital \NonWebCollectorInstantiationTool`.
    A list of options to manage collector instances appears.

    b. Connect to the collector machine by entering `1` or `2`, depending on whether collectors are installed locally or on a remote machine.

    c. Enter `5`.
    You are prompted to enter the interface name of the collector whose instance you want to delete.

    d. Enter the interface name of the collector that you want to delete.
    You are prompted to specify whether you want to delete the tag data as well.

    e. Enter 1 if you want to delete the tag data as well, or enter 0.

The collector instance is deleted.

# Add an Offline Collector Instance

You can use an offline collector to send data directly to a cloud destination (without using a Historian server).

1. Access the installation folder of ihCollectorManager_x64.exe. By default, it is `C:\Program Files \GE Digital\Historian Remote Management Agents\Collector Manager`.
2. If you want to manually enter the installation parameters and values, run the following command:

   ```
   ihCollectorManager_x64.exe InterfaceCreateViaCmd "{\"<parameter>\":\"<value>\",
   \"<parameter>\":\"<value>\"}"
   ```

   If you want to use a JSON file containing the installation parameters and values, run the following command:

   ```
   ihCollectorManager_x64.exe InterfaceCreateViaFile "<path to the JSON file>"
   ```

   Instead of manually creating the JSON file, you can use the RemoteCollectorConfigurator utility to generate it automatically *(on page 796)*.
   For information on the parameters, refer to Collector Instance Parameters *(on page 801)*.

# Delete an Offline Collector Instance

This topic describes how to delete an offline collector instance using the RemoteCollectorConfigurator utility.

**Note:**
When you delete an offline collector instance, the corresponding configuration file is not deleted. However, if another collector instance of the same interface name is created, the existing configuration file is replaced by a template configuration file.

1. Access the installation folder of ihCollectorManager_x64.exe. By default, it is `C:\Program Files \GE Digital\Historian Remote Management Agents\Collector Manager`.
2. Run the `ihCollectorManager_x64.exe` file.
3. Run the following command:

   ```
   ihCollectorManager_x64.exe InterfaceDelete <interface name>
   ```

# Manage a Collector Remotely

1. Ensure that the Historian server connected to the collectors that you want to manage is upgraded to Historian 8.1.
2. Install Remote Management Agents *(on page 187)*.

   **Note:**
   Remote Collector Management will be installed as part of this installation.

3. Ensure that the Windows Task Scheduler service is running. This service is required to manage collectors in the command line mode. You can check the status of this service in the Microsoft Services Management console.
4. If you want to manage the iFIX collectors remotely, access the **SCU - FIX** window, and modify the task configuration such that the value in the **Command Line** field is **NOSERVICE**.

Perform any of the following tasks using Configuration Hub:

- Start *(on page 637)*, stop *(on page 638)*, or restart *(on page 639)* a collector.
- Pause *(on page 641)* or resume *(on page 642)* the data collection of a collector.
- Delete *(on page 643)* or move *(on page 644)* the buffer files of a collector.
- Change the destination server of a collector *(on page 645)*.

**Note:**
You can also perform these tasks using REST APIs *(on page 1254)*.

# Troubleshooting Remote Collector Management Issues

### Remote Collector Management does not work

**Issue:** If an iFIX collector instance created in version 9.0 exists, after you upgrade collectors, another instance of the iFIX collector is created. Because of this, the Remote Collector Manager (RCM) will not work correctly. This is applicable to the iFIX Alarms and Events collector as well.

**Workaround:** If you want to use RCM, you must delete one of the instances. If needed, you can manually create another instance of the iFIX collector using Configuration Hub or the RemoteCollectorConfigurator utility.

## The ServiceName Registry Key is not Updated

**Issue:** When you attempt to manage collectors, sometimes, an error message appears in the `CollectorManager.shw` file.

**Error message:** Below are collectors in the registry without a service name. The String Registry value 'ServiceName' exists, but is blank. Collector Manager will not try to determine what the service name is. This will need to be manually configured.

**Cause:** When Remote Collector Management is started for the first time, the collector that you want to manage is not running. When this happens, the ServiceName registry key is not updated.

**Workaround:**

1. Stop the Remote Collector Management service.
2. Start the collector.
3. Start the Remote Collector Management service.

## The ServiceName Registry key is Updated Incorrectly

**Workaround:**

1. Stop the Remote Collector Management service.
2. Access the registry folder of the collector.
3. Delete the ServiceName key.
4. Start the collector.
5. Start the Remote Collector Management service.
6. Access the .shw file to verify that the ServiceName key has been updated.

# Chapter 5. Using Historian Administrator

## Historian Administrator

### Overview of Historian Administrator

Historian Administrator is a Windows-based application, which allows you to access administrative functions. Using Historian Administrator, you can monitor, supervise, archive, retrieve, and control data gathering functions from the server, a client, or one or more remote non-web-based nodes.

> **Note:**
> You can install multiple instances of Historian Administrator. Changes that you make to parameters on one instance are not automatically updated in other instances.

Historian Administrator communicates with the Historian server using the Historian API. You can install Historian Administrator on a local or a remote machine that has a TCP/IP connection to the Historian server.

#### Intended Audience

This guide is intended for people who need to:

- Retrieve and analyze archived information.
- Monitor Historian system performance.
- Set up and maintain configuration and other parameters for tags, collectors, and archives.
- Perform specific supervisory and security tasks for Historian.
- Maintain and troubleshoot Historian.

#### About Historian Administrator

Using Historian Administrator, you can:

- Examine key operating statistics for archives and collectors, and display or search system alerts and messages.
- Perform archive maintenance, including:
    - Setting archive size.
    - Selecting options and parameters.
    - Accessing security parameters.

    - Adding and restoring archives.
    - Performing backup and restoration tasks
- Perform tag maintenance, including:
    - Adding, deleting, and copying tags.
    - Searching for tags in a data source or in the Historian database.
    - Starting and stopping data collection for a tag.
    - Configuring, displaying, and editing tag parameters and options.
    - Displaying trend data for selected tags.
- Perform collector maintenance, including:
    - Adding or deleting collectors.
    - Configuring, displaying, and editing parameters for all types of collectors.
    - Creating calculation formulas.
    - Displaying performance trends for selected collectors.

### Limitations

If the number of archives is large (that is, more than 5,000), Historian Administrator takes a long time to start.

## Access Historian Administrator

- Install Historian Administrator (by installing Client Tools).
- Create a Windows user on the Historian server.
- Use a page with a resolution of 1024 x 768 or above.

From the Start menu, select **Historian Administrator**.

**Note:**
By default, The system attempts to connect to the default server using the username and password of the currently logged-in user. If you want to use a different server or user account:

a. Select **Main**.

   A login window appears.

b. Provide the server name, username, password, and domain information, and then select **OK**.

The **Proficy Historian Administrator** window appears, displaying the following pages.

- **System Statistics**: Contains system status indicators, data collector performance indicators, system alerts and messages, with links to the data store maintenance, collector maintenance, tag maintenance, message search, and help pages.
- **Main**: Allows yo to log in to a different Historian Server.
- **Tags**: Contains tag names, parameters, and controls.
- **Collectors**: Contains collector names, parameters, and controls.
- **DataStores**: Contains archive names, parameters, alarms, security, and controls.
- **Messages**: Contains alerts and messages selected by user-defined search parameters.

## Historian in a Regulated Environment

Many FDA-regulated industries are required by the United States government to be compliant with regulations such as the 21CFR Part 11 regulation. If your industry is one of them, you will need software that allows you to build a compliant application or process. The flexibility and versatility of Historian lets you create a compliant process by:

- Providing limited system access to authorized individuals.
- Adding a timestamp to annotations, and displaying timestamps from the Historian archive.
- Requiring electronic signatures on annotations.

> **Note:**
> This option is available if the **Electronic Signatures/Records** option is enabled in the Historian Server.

- Supporting human-readable printouts and computer-readable format for audits (by exporting data to a CSV file that can be imported to a Microsoft Excel document or an SQL database).

### Disabling Guest Accounts for a 21 CFR Part 11-Compliant Environment

If you want to use Historian in a 21 CFR Part 11-compliant environment, make sure that you disable guest accounts on your computer. This action applies whether or not you use Historian security.

### Compliant Parameter Settings

You can set the following 21 CFR 11-compliant parameters in the **Security** section of the **Data Store** page.

| Field | Description |
|---|---|
| Require Point Verification | Indicates whether you must enter identifying information whenever you attempt a restricted action. Whenever you attempt to change the system con- |

| Field | Description |
|---|---|
| | figuration (for the tag, archive, or collector), a tag value, or another record, you must electronically sign the action with a username and password. If the user is authorized to make this change, the identity of the person, the action performed, and the time it was performed, are all recorded in the audit trail.<br><br>**Note:**<br>• The audit features are not dependent on this feature being enabled. Historian audits all user actions regardless of whether this option is enabled.<br>• If you plan to create multiple archives at the same time, select the **Disabled** option.<br><br>Enabling electronic signatures and electronic records also requires you to reverify your identity when you use the Historian Excel Add-in, modify or create a tag, or import data or messages.<br><br>**Note:**<br>This feature is available only if you have purchased the Electronic Signatures and Electronic Records option. |
| Verification Message | When point verification is enabled, you are prompted to enter the username and password whenever you attempt to perform an action specified as requiring point verification. |

# High Availability

## About High Availability

Historian supports high availability of the Historian server, archive files and configuration files. To facilitate this support, Historian uses Microsoft Cluster Server (MSCS) as well as an automated backup strategy. High availability decreases the likelihood of archive file corruption due to software or hardware failures. Implementing high availability ensures that collection of your data remains uninterrupted.

### How it Works

If the primary Historian server fails due to software or hardware issues, MSCS replaces it with another Historian server. In addition, through the use of automated backups to a shared location, at least one known good Historian archive is maintained at all times. Older archives are replaced once per hour only if there is a more recent archive in place.

To provide high availability of Historian server for Client Manager, Configuration Manager, and Diagnostics Manager components of the distributed Historian service using Microsoft Failover Cluster Service, you must configure them as generic services in Failover Cluster. For information, refer to Microsoft documentation.

High availability is achieved by following these steps:

1. Every 5 seconds, MSCS monitors the health of all services and applications it is managing by performing a quick check of whether each service is in a running state.
2. Every 60 seconds, MSCS performs a more thorough test of the applications' health.
3. Server high availability logs are added to the server and queried.
4. If these steps fail, Historian assumes the server is no longer running, which causes the cluster to initiate failover of the application.

**Note:**
By default, MSCS monitors the health of all the services and applications every 5 seconds and 60 seconds. You can change this frequency using the cluster administrator.

For most failures, the cluster will detect a problem within 5 seconds of an application or service becoming unresponsive. In the case of a server hang, where the server process is still running but is otherwise unresponsive, it may take as many as 60 seconds to detect. The time to switch to another Historian server includes the time required for the cluster to re-start the Historian server. Server start-up time depends on the size and number of online archives.

### Limitations

- Only a single Historian resource instance is supported per cluster.
- The Historian installation automatically registers Historian and AlarmArchiver resources with MSCS.
- Configuring an AlarmArchiver resource on MSCS requires a dependency on a Historian resource.

- Running collectors on a cluster is not recommended, as they are not supported on the cluster server. Therefore, failover cannot be performed if a cluster node goes offline. Historian offers a separate collector high availability feature.
- Historian local security with local users is not supported in cluster nodes.

### High Availability of Archive and Configuration Files

Historian supports high availability of the latest archive (`.iha`) and configuration (`.ihc`) files. A copy of the latest `.iha` and `.ihc` file is created once every hour. If the latest archive files become corrupt, Historian will restore the files.

The following conditions apply when using this feature:

- Archive files are backed up only for tag data, not for alarms and events data.
- When using a cluster, this feature is automatically enabled; you cannot disable it.
- By default, this feature is disabled on a 64-bit Windows operating system.
- When you enable this feature, backup of all the current archive files is created. This will increase your storage overhead to twice that of the current archive. Therefore, for better performance, we recommend that you do not use this feature for a large-scale system.

## Enable High Availability of Archive and Configuration Files

1. Install Historian on each node of the cluster. A hardware license key is required for each node of the cluster.
2. Configure a Historian resource on MSCS as follows:
    - Allow client access to the IP address and network name of the Historian server.
    - Ensure that a shared storage device or location is available to all the Historian cluster nodes in order to store the Historian archive files.
3. After you configure the components of the failover cluster, when configuring Client Manager, Configuration Manager and Diagnostics Manager, enable the **Use Network Name for Computer Name** option on the **General** section of the resource's properties.

1. Access Historian Administrator.
2. Select **Archives > Global Options**.
3. In the **Archive Maintenance** field, select the **Maintain Auto Recovery Backup Files** option.

## Register Historian with a Windows 2012 Cluster

Before using later versions of Historian on Windows 2012 Cluster, you must register the `Historian.dll` file using PowerShell.

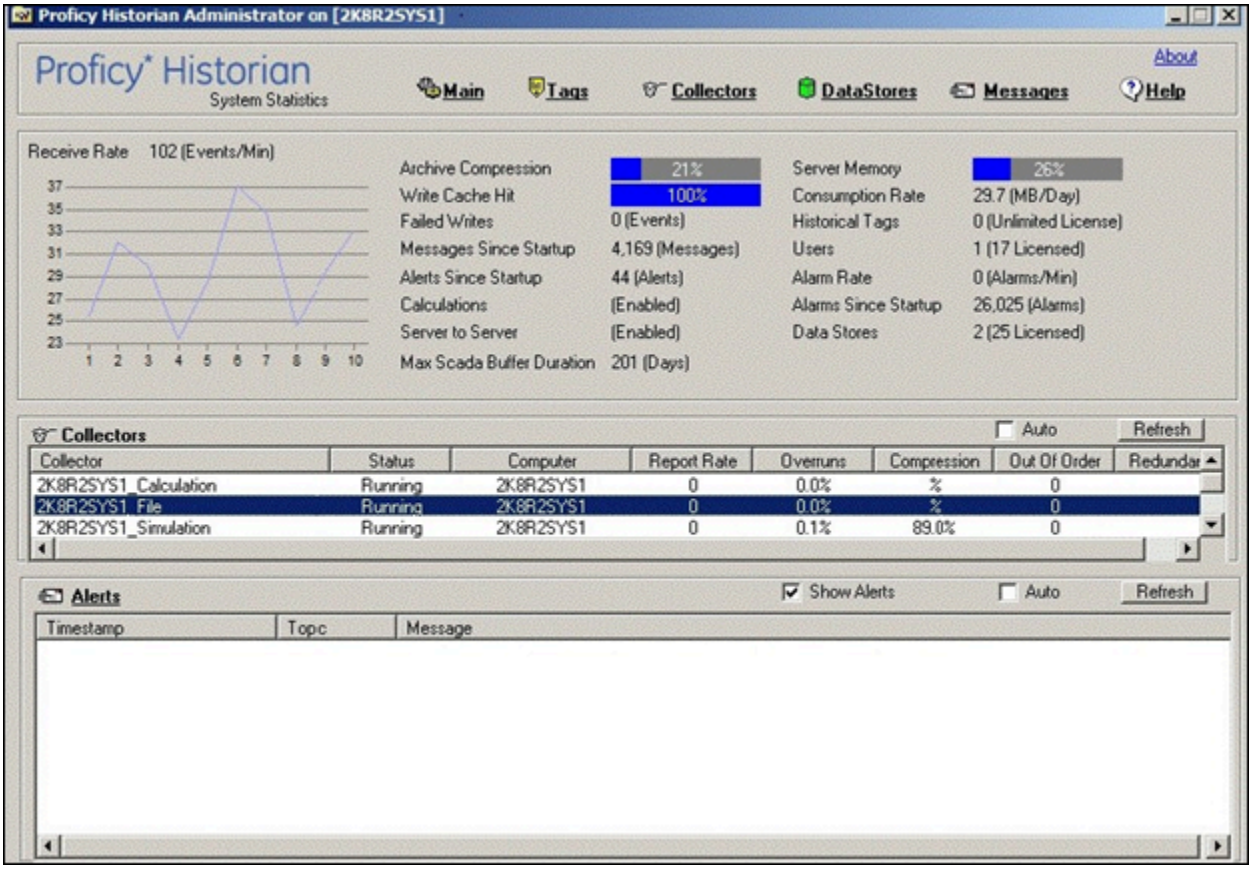
1. Open PowerShell in a Command prompt.
2. Enter the following command to register Data Archiver with the Cluster:
   ```
   Add-clusterresourcetype Historian "C:\ProgramFiles\Proficy\Historian\x64\Server
   \Historian.dll" -DisplayName
   ```
3. Enter the following command to register AlarmArchiver with the Cluster:
   ```
   Add-clusterresourcetype AlarmArchiver "C:\Program Files\Proficy\Historian\x64\Server
   \Historian.dll" -DisplayName
   ```

# Historian Administrator - Pages

## The Main Page

The **Main** page of Historian Administrator displays the system statistics, which contains the current system status and performance statistics. It provides an overall view of the system health. The page has the following sections:

- The **System Statistics** section *(on page 829)*
- The **Collectors** section *(on page 833)*
- The **Alerts** section *(on page 835)*




## The **System Statistics** Section

The following table describes the fields in the **System Statistics** section.

**Note:**
The statistics displayed in this section are calculated independently on various time scales and schedules. As a result, they may be updated at different times.

| Field | Description |
|---|---|
| Receive Rate (a time-based chart in events/minute) | Displays how busy the server is at a given instance and the rate at which the server is receiving data from collectors. |
| Archive Compression (% compression) | Displays the current effect of archive data compression. If the value is zero, it indicates that archive compression is either ineffective or turned off. To increase the effect of data compression, increase the value of archive compression deadbands on individual tags in the Tag Maintenance section to activate compression. |

| Field | Description |
|---|---|
| | In calculating the effect of archive compression, Historian counts internal system tags as well as data source tags. Therefore, when working with a very small number of tags and with compression disabled on data source tags, this field may indicate a value other than zero. If you use a realistic number of tags, however, system tags will constitute a very small percentage of total tags and will therefore not cause a significant error in calculating the effect of archive compression on the total system. |
| Write Cache Hit | Displays the hit ratio of the write cache in percentage of total writes. It is a measure of how efficiently the system is collecting data. Typically, this value should range from 95 to 99.99%. If the data is changing rapidly over a wide range, however, the hit percentage drops significantly because current values differ from recently cached values. More regular sampling may increase the hit percentage. Out-of-order data also reduces the hit ratio. |
| Failed Writes | Displays the number of samples that failed to be written. Since failed writes are a measure of system malfunctions or an indication of offline archive problems, this value should be zero. If you observe a non-zero value, investigate the cause of the problem and take corrective action.<br><br>Historian also generates a message if a write fails. Note that the message only appears once per tag, for a succession of failed writes associated with that tag. For example, if the number displayed in this field is 20, but they all pertain to one Historian tag, you will only receive one message until that Historian tag is functional again. |
| Messages Since Startup | Displays a count of system messages generated since the last startup. The system resets the value to zero on restart. The message database, however, may contain more messages than this value. |
| Alerts Since Startup | Displays a count of system warnings or alerts generated since the last startup. A high value here may indicate a problem of some kind. You should review the alerts and determine the probable cause. The count resets to zero on restart. The message database, however, may contain more alerts than this value. |

| Field | Description |
|---|---|
| Calculations | Displays the value **Enabled** if the Calculation collector is licensed on the software key. |
| Server-to-Server | Displays the value **Enabled** if the Server-to-Server collector is licensed on the software key. |
| Alarms since Startup | Displays a count of alarms received by the data archiver since starting up. |
| Server Memory | Displays how much of the server memory the data archiver consumes. |
| Free Space (MB) | Displays how much disk space (in MB) is left in the current archive. |
| Consumption Rate (MB/day) | Displays how fast the archive disk space is consumed. If the value is too high, you can reduce it by slowing the poll rate on selected tags or data points or by increasing the filtering on the data (widening the compression deadband to increase compression). |
| Est. Days to Full (Days) | Displays how much time is left before the archive is full, based on the current consumption rate. This value is dynamically calculated by the server and becomes more accurate as an archive file gets closer to completion. This value is only an estimate and will vary based on a number of factors, including the current compression effectiveness. The System sends messages notifying you at 5, 3, and 1 days until full. After the archive is full, a new archive must be created (could be automatic).<br><br>To increase this value, you must reduce the consumption rate. To ensure that collection is not interrupted, make sure that the **Automatically Create Archives** option is enabled in the **Data Store Maintenance** section (under **Global Options**). You may also want to enable the **Overwrite Old Archives** option if you have limited disk capacity. Enabling overwrite, however, means that some old data will be lost when new data overwrites the data in the oldest online archive. Use this feature only when necessary. |
| Active Tags | Displays number of tags in your configuration. |
| Licensed Tags | Displays the number of tags authorized for this Historian installation by the software key and license. |

| Field | Description |
|---|---|
| | **Note:**<br>If this field displays 100 tags and the licensed users field displays 1 client, you are likely running in demonstration mode and may have incorrectly installed your hardware key. |
| Active Users | Displays the number of users currently accessing the Historian system. |
| Licensed Users | Displays the number of users authorized to access Historian using the software key and license.<br><br>The number of users that are authorized to access Historian is strictly based on the software key and license. However, if you have utilized your available Client Access Licenses (CAL) and need an additional one to administer the system in an emergency, you have an option to reserve a CAL. This reserved CAL allows you to access the server. To do so, provide the reserved CAL to the system administrators and add them to the `ih Security Admins` group. A system administrator can then connect to Historian in an emergency.<br><br>This facility is optional and does not provide a guaranteed connection. It only eliminates the emergency situations when a CAL is preventing you from accessing the system and may not work if there are other conditions. For example, if the Historian server is busy, you will not be able to connect using this feature.<br><br>**Note:**<br>If this field displays 1 client and the Licensed Tags field displays 100 tags, you are likely running in demonstration mode and you may have incorrectly installed your hardware key. |
| Alarm Rate | Displays the rate at which Historian is receiving alarms and events data. |
| SCADA Tags | Displays the number of CIMPLICITY or iFIX tags. |
| Tags Consumed by Arrays | Indicates the total number of Array tags consumed by Historian. |

## The **Collectors** Section

The **Collectors** section shows current statistics on the operation of all the connected collectors in the system. In this section, you can:

- Access the **Collector Maintenance** page of a collector by selecting the collector name. You can also access the **Collector Maintenance** page by selecting the collector link at the beginning of the **System Statistics** section.
- Automatically refresh the data every 45 seconds by selecting the **Auto** check box.
- Manually refresh the data by selecting **Refresh**.

The following table describes the fields in the **Collectors** section.

| Field | Description |
|---|---|
| Collector | Displays the collector ID, which is used to identify the collector in Historian. |
| Status | Displays the current status of collection. This field contains one of the following values:<br>• **Running**: Indicates that the collector is running.<br>• **Stopped**: Indicates that the collector is not collecting data.<br>• **Unknown**: Indicates that status information about the collector is unavailable, perhaps as a result of a lost connection between the collector and the server. |
| Computer | Displays the name of the computer on which the collector is running. |
| Report Rate | Displays the number of samples per minute that the server is receiving data from the collector. It is a measure of the collection rate and data compression. If the collector compression percent is zero, and if the value in this field is equal to the data acquisition rate, it indicates that every data point received from the collector is being reported to the server. This means that the collector is not performing any data compression. You can lower the report rate, and make the system more efficient, by increasing the data compression at the collector. To do this, widen the collection compression deadbands for selected tags. |
| Overruns | Displays the overruns in relation to the total events collected since startup. This value is calculated by using the following equation: `OVERRUN_PCT = OVERRUNS / ( OVERRUNS + TOTAL_EVENTS_COLLECTED )` Overruns are a count of the |

| Field | Description |
|---|---|
| | total number of data events not collected on their scheduled polling cycle. In normal operation, this value should be zero.<br><br>You may be able to reduce the number of overruns on the collector by increasing the tag collection intervals (per tag). |
| Compression % | Displays the percentage of how effective compression is at present for the specific collector since collector startup. A value of zero indicates that compression is either turned off or not effective. To increase the value, enable compression on the collector's associated tags and increase the width of the compression deadband on selected tags.<br><br>The collector keeps track of how many samples it collected from the data source (for example, the OPC server) and keeps track of how many samples it reported to the Data Archiver (after collector compression is complete).<br><br>A low number or zero means almost everything coming from the data source is being sent to the data archiver. The reason for the low number or zero is that too many samples are exceeding compression or you are not using collector compression.<br><br>A high number or 100 means you are collecting a lot of samples, but they are not exceeding collector compression and therefore are not being sent to server. |
| Out of Order | Displays the number of samples within a series of timestamped data values normally transmitted in sequence that have been received out of sequence since collector startup. This field applies to all collectors. Even though events are still stored, a steadily increasing number of out-of-order events indicates a problem with data transmission that you must investigate.<br><br>For instance, a steadily increasing number of out-of-order events when you are using the OPC collector means that there is an out-of-order connection between OPC Server and the OPC collector. This may also cause out-of-order connection between the OPC collector and the data archiver although that is not what this statistic indicates. |
| Redundancy | Displays the current redundancy status of the collector. This field contains one of the following values: |

| Field | Description |
|---|---|
| | • **Active**: Indicates that the collector is currently collecting data.<br>• **Standby**: Indicates that the collector is in the standby mode. This value appears only if the Collector Redundancy property of the collector is enabled. |

## The **Alerts** Section

The **Alerts** section displays all the alerts and warnings received or generated by the system. A total of up to 250 of the most recent messages appear in this section. In this section, you can:

- Stop automatic updating of the data by clearing the **Show Alerts** check box. However, the check box will be selected automatically when you restart Historian Administrator.
- Automatically refresh the data every 25 seconds by selecting the **Auto** check box.
- Manually refresh the data by selecting **Refresh**.

The following table describes the fields in the **Alerts** section.

| Field | Description |
|---|---|
| Timestamp | The timestamp associated with the message or alert. |
| Topic | The type of alert message. Only the services and performance alerts appear in this field. |
| Message | The content of the message or alert. |

## The Data Store Page

Using the **Data Store** page, you can read and modify the parameters of archives, data stores, global options, security, and alarms.

## The **Archive Details** Section

In the **Archive Details** section, a list of all the archives in your system appears. To access an archive, select it. In this section, you can:

- Close an archive by selecting **Close Archive**.
- Back up an archive by selecting **Backup**, and then providing the file name and path for the backup.
- Remove an archive by selecting **Remove**. The archive is then placed in the `\Archives\Offline` folder. However, the archive is not deleted.

This topic describes the fields in each subsection in the **Archive Details** section.

### The Status Subsection

| Field | Description |
|---|---|
| Status | The current operating state of the archive. This field contains one of the following values:<br>• **Current**: Indicates that the archive is actively accepting data.<br>• **Active**: Indicates that the archive contains data but is not currently accepting data.<br>• **Empty**: Indicates that the archive has never accepted data. |
| Start Time | The time of the oldest sample in the archive. |
| End Time | The time the archive is closed (automatically or manually). |

**Table 60. The Backup Subsection**

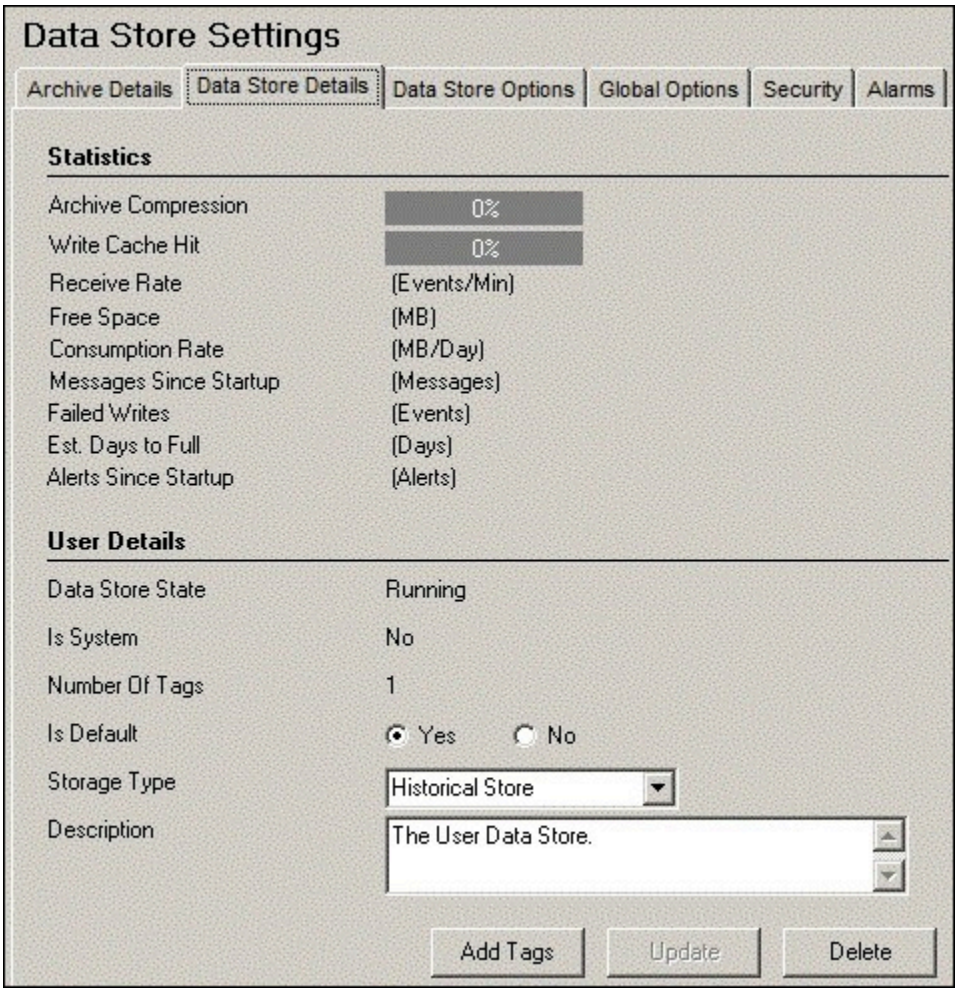
| Field | Description |
|---|---|
| Last Backup | The date and time the last backup was performed on the archive. |
| Backup User | The username of the user who performed the last backup of the archive. |

**Table 61. Resources**

| Field | Description |
|---|---|
| File Location | The path and name of the archive file. |
| File Size | The size (in MB) of the archive file.<br>**Note:**<br>Historian supports a maximum archive size of 256 GB per archive. |
| File Attribute | The attribute to set a closed archive to read-only or read/write.<br>**Note:**<br>To create multiple archives at the same time, set the value of this field to Read/Write. |

# The **Data Store Details** Section

This topic describes the fields in each subsection in the **Data Store Details** section.




## The Statistics Subsection

| Field | Description |
|---|---|
| Archive Compression (% compression) | Displays the current effect of archive data compression. If the value is zero, it indicates that archive compression is either ineffective or turned off. To increase the effect of data compression, increase the value of archive compression deadbands on individual tags in the Tag Maintenance section to activate compression.<br><br>In calculating the effect of archive compression, Historian counts internal system tags as well as data source tags. Therefore, when working with a very small number of tags and with compression dis- |

| Field | Description |
|---|---|
| | abled on data source tags, this field may indicate a value other than zero. If you use a realistic number of tags, however, system tags will constitute a very small percentage of total tags and will therefore not cause a significant error in calculating the effect of archive compression on the total system. |
| Write Cache Hit | Displays the hit ratio of the write cache in percentage of total writes. It is a measure of how efficiently the system is collecting data. Typically, this value should range from 95 to 99.99%. If the data is changing rapidly over a wide range, however, the hit percentage drops significantly because current values differ from recently cached values. More regular sampling may increase the hit percentage. Out-of-order data also reduces the hit ratio. |
| Receive Rate | Displays how busy the server is at a given instance and the rate at which the server is receiving data from collectors. |
| Free Space (MB) | Displays how much disk space (in MB) is left in the current archive. |
| Consumption Rate (MB/day) | Displays how fast the archive disk space is consumed. If the value is too high, you can reduce it by slowing the poll rate on selected tags or data points or by increasing the filtering on the data (widening the compression deadband to increase compression). |
| Messages Since Startup | Displays a count of system messages generated since the last start-up. The system resets the value to zero on restart. The message database, however, may contain more messages than this value. |
| Failed Writes | Displays the number of samples that failed to be written. Since failed writes are a measure of system malfunctions or an indication of offline archive problems, this value should be zero. If you observe a non-zero value, investigate the cause of the problem and take corrective action.<br><br>Historian also generates a message if a write fails. Note that the message only appears once per tag, for a succession of failed writes associated with that tag. For example, if the number displayed in this field is 20, but they all pertain to one Historian tag, you will only receive one message until that Historian tag is functional again. |

| Field | Description |
|---|---|
| Est Days to Full (Days) | Displays how much time is left before the archive is full, based on the current consumption rate. This value is dynamically calculated by the server and becomes more accurate as an archive file gets closer to completion. This value is only an estimate and will vary based on a number of factors, including the current compression effectiveness. The System sends messages notifying you at 5, 3, and 1 days until full. After the archive is full, a new archive must be created (could be automatic).<br><br>To increase this value, you must reduce the consumption rate. To ensure that collection is not interrupted, make sure that the **Automatically Create Archives** option is enabled in the **Data Store Maintenance** section (under **Global Options**). You may also want to enable the **Overwrite Old Archives** option if you have limited disk capacity. Enabling overwrite, however, means that some old data will be lost when new data overwrites the data in the oldest online archive. Use this feature only when necessary. |
| Alerts Since Startup | Displays a count of system warnings or alerts generated since the last startup. A high value here may indicate a problem of some kind. You should review the alerts and determine the probable cause. The count resets to zero on restart. The message database, however, may contain more alerts than this value. |

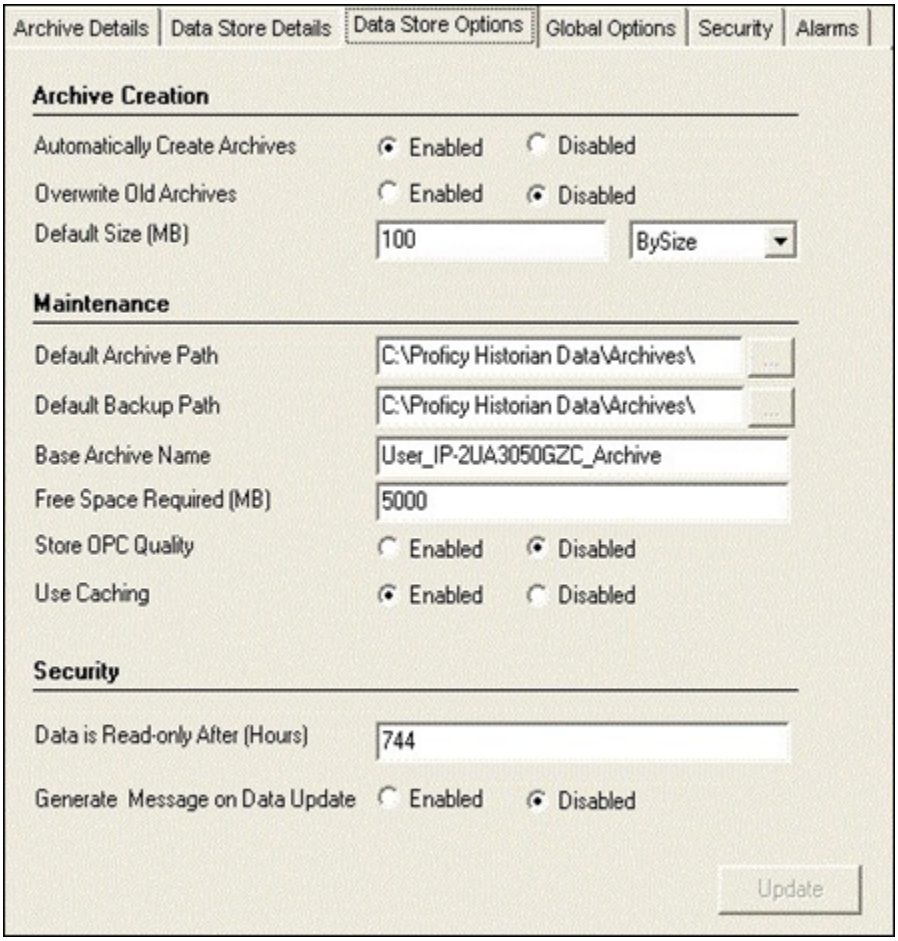
## The User Settings Subsection

| Field | Description |
|---|---|
| Data Store State | The current state of the data store. The value in this field is **Running** until you delete the data store. |
| Is System | Indicates whether this data store is the system data store.<br><br>**Note:**<br>By default, the **Is System** value of the system data store is set to **yes**. You cannot set the **Is System** value of any historical data store to **yes**. |

| Field | Description |
|---|---|
| Number of Tags | Displays the number of tags the data store contains. |
| Is Default (Yes/No) | Indicates whether the data store is the default store. Select **Yes** to set this data store as default one. |
| Storage Type | Indicates whether the storage type is historical or SCADA buffer. |
| Description | The description of the data store. |

## The **Data Store Options** Section

This topic describes the fields in each subsection in the **Data Store Options** section.




### The Archive Creation or the SCADA BufferSubsection

The **Archive Creation** subsection appears only if the data store type is historical. The **SCADA Buffer** subsection appears only if the data store type is SCADA buffer.

| Field | Description |
|---|---|
| Automatically Create Archives | Identifies whether the server must automatically create an archive file whenever the current archive file is full. The archive files are created in the default path directory.<br><br>**Note:**<br>If you plan to create multiple archives at the same time, select the **Disabled** option. |
| Overwrite Old Archives | When enabled, the system replaces the oldest archived data with new data when the default size has been reached. Since this action deletes historical data, exercise caution in using this feature. We recommend that you back up the archive so that you can restore it later.<br><br>**Note:**<br>To create multiple archives at the same time, select the **Disabled** option. If both the **Automatically Create Archives** and **Overwrite Old Archives** are enabled, set the ihArchiveFreeSpaceHardLimit parameter to TRUE using the Historian APIs. |
| Default Size (MB) | The default size of a newly created archive or the duration of a newly created archive in days or hours. Select one of the following options:<br><br>• **BySize**: A new archive file is created after the current archive reaches the default size. The recommended default archive size is at least 500 MB for systems with 1000 tags or more.<br>• **Days**: A new archive file is created after the number of days that you specify in the **Archive Duration** field that will appear.<br>• **Hours**: A new archive file is created after the number of hours that you specify in the **Archive Duration** field that will appear. |
| SCADA Buffer Duration (Days) | Indicates the maximum number of days you want to store the trend data. The maximum number of days is equal to the value of the **Rotational Buffer Duration** shown in the GE Licence Client. |
| Archive Duration (Days/Hours) | Indicates the days or hours for which the duration of the archive is set. |

### The Maintenance Subsection

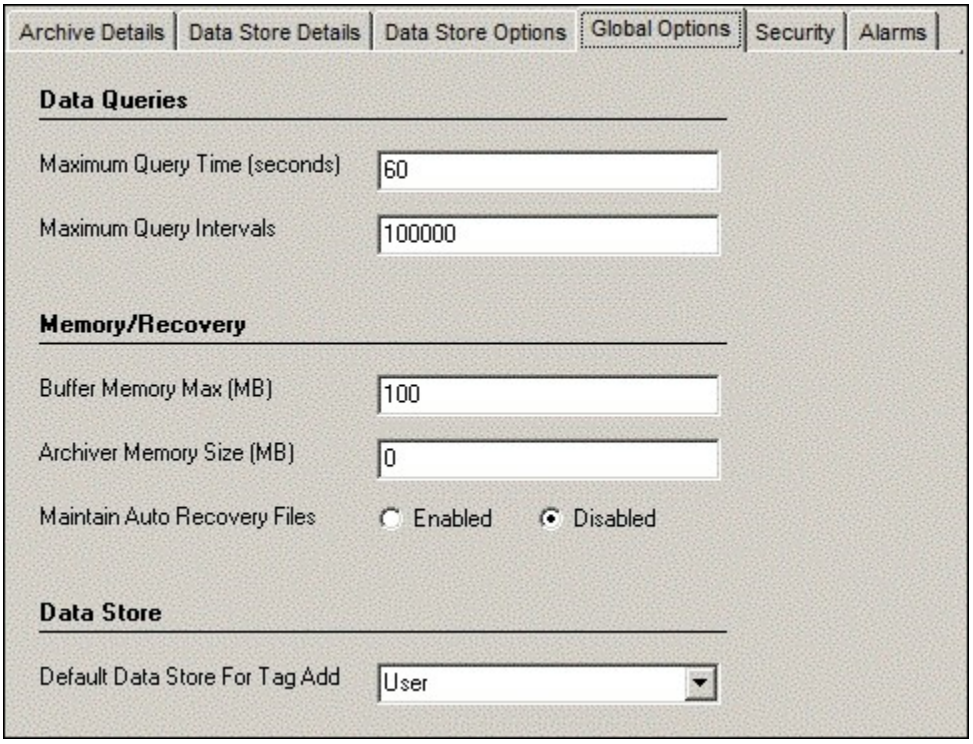
| Field | Description |
|---|---|
| Default Archive Path | The folder path to store newly created archives.<br><br>**Note:**<br>We recommend not to use a period in the default archive path field. If you do so, you will not be able to specify a default archive name. |
| Default Backup Path | The location to which the backup file will be saved. |
| Base Archive Name | A prefix that you want to add to all the archive files. |
| Free Space Required (MB) | Indicates the remaining disk space required after a new archive is created. If the available space is less than the requirement, a new archive is not created. The default value is 5000 MB.<br><br>This field is not applicable to alarms and events archives. The alarms and events archiver will continue writing to the alarms and events archive until the drive is full. If this occurs, the alarms and events archiver will buffer incoming alarms and events data until the drive has free space. An error message is logged in the Historian message log. |
| Store OPC Quality | Indicates whether to store the OPC data quality.<br><br>**Note:**<br>To create multiple archives at the same time, select the **Disabled** option. |
| Use Caching | Indicates whether caching must be enabled. When reading data from the archiver, some data is saved in the system memory and retrieved using caching. This results in faster retrieval as the data is already stored in the buffer.<br><br>**Note:**<br>This option is not available for SCADA Buffer data stores. |

### The Security Subsection

| Field | Description |
|---|---|
| Data is Read-only After (Hours) | The number of hours for data to be stored in a read/write archive. After the time lapses, that portion of the archive file is automatically made read-only. Incoming data values with timestamps prior to this time are rejected. A single archive file, therefore, may have a portion made read-only, another portion that is read/write containing recently written data, and another that is unused free space.<br><br>**Note:**<br>If an archive file is read-only, you cannot move the file in Windows File Explorer. To be able to move a read-only archive file, you must first remove the archive by selecting the file and selecting **Remove** in the **Archive Maintenance** page. |
| Generate Message on Data Update | Indicates whether an audit log entry will be made any time the value of a previously archived data point is overwritten. This log entry will contain both the original and new values.<br><br>**Note:**<br>To create multiple archives at the same time, select the **Disabled** option. This option is not available for SCADA Buffer data store. |

## The **Global Options** Section

This topic describes the fields in each subsection in the **Global Options** section.




## The Data Queries Subsection

| Field | Description |
|---|---|
| Maximum Query Time (seconds) | Specifies the maximum time that a data point or message query can take before it is terminated. Use this setting to limit query time and provide balanced read access to the archiver. This is applicable to all query types. |
| Maximum Query Intervals | Specifies the maximum number of samples per tag that Historian can return from a non-raw data query. Use this setting to throttle query results for non-raw data queries. This setting is not applicable to filtered data queries or raw data queries.<br><br>If the number of returned samples exceeds the value in this field, the query fails and no data is returned. |

## The Memory/Recovery Subsection

| Field | Description |
|---|---|
| Buffer Memory Max (MB) | The maximum memory buffer size that an archiver queue will use before starting to use disk buffering. The default value is 100 MB. |

| Field | Description |
|---|---|
| | **Note:**<br>• You can monitor your collector data write queue using the Perftag_CollectorDataWriteQueueSize tag. If you find that the queue is exceeding 10,000 items, such as during a store and forward flush, change the value of this field to 500 or more to maintain Historian performance.<br>• If you are upgrading from a previous version of Historian, the value in this field remains the same. You can change the value as needed. |
| Archiver Memory Size (MB) | The target memory usage of the archive. The default value is 0, which indicates the system will manage the memory usage. If the archiver is running on a 32-bit operating system and you want to keep more data in memory, you can enter a value up to 1800 MB. If the archiver is running on a 64-bit operating system, we recommend that you use the default value. |
| Maintain Auto Recovery Files | Indicates whether high availability of the latest archive (.iha) and Historian configuration (.ihc) files must be enabled. When enabled, a copy of the latest .iha and .ihc file is made once every hour. These recovery files do not include alarms and events data.<br>**Note:**<br>These files are managed internally by Historian, and should not be used as backup files. To create multiple archives at the same time, select the **Disabled** option. By |

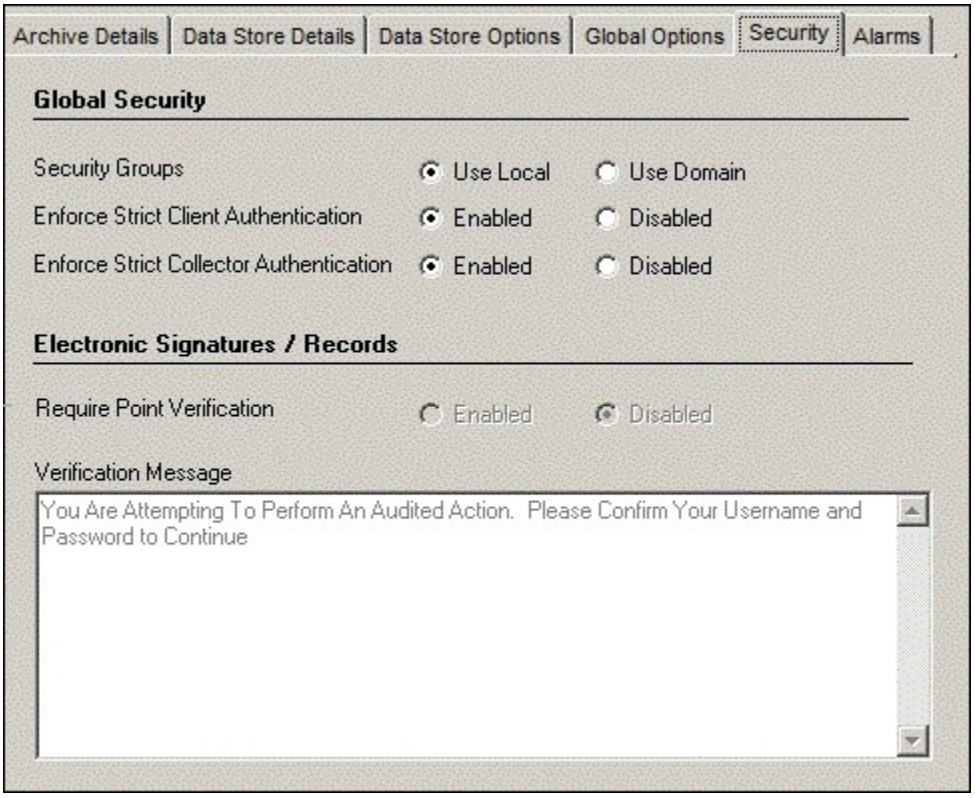
| Field | Description |
|---|---|
| | default, this field is set to **Disabled** on a 64-bit operating system. On a large-scale system, we recommend that you disable this option for better performance. |

### The Data Store Subsection

| Field | Description |
|---|---|
| Default Data Store For Tag Add | The name of the default data store to which you want to add tags. |

## The Security Section

This topic describes the fields in each subsection in the **Security** section




.

### The Global Security Subsection

| Field | Description |
|---|---|
| Security Groups | Indicates whether to use the local security groups or the domain security groups.<br><br>**Note:**<br>To ensure a secure environment when using Historian, do not create any local user accounts unless Historian is set up on a standalone computer and the guest account is disabled. |
| Enforce Strict Client Authentication | Indicates whether to use strict client authentication. If you enable this option, only clients using the security-token-based authentication protocol can connect. Clients using Historian versions prior to 6.0 and other Proficy software they connect to may not be able to connect unless they have the latest updates for that version. If you disable this option, clients of any version can connect if they use a valid user name and password. |
| Enforce Strict Collector Authentication | Indicates whether to use strict collector authentication. If you enable this option, only collectors using the security-token-based authentication protocol can connect. Collectors using Historian versions prior to 6.0 and the other Proficy software they connect to may not be able to connect unless they have the latest updates for that version. If you disable this option, collectors of any version can connect. |

### The Electronic Signatures / Records Subsection

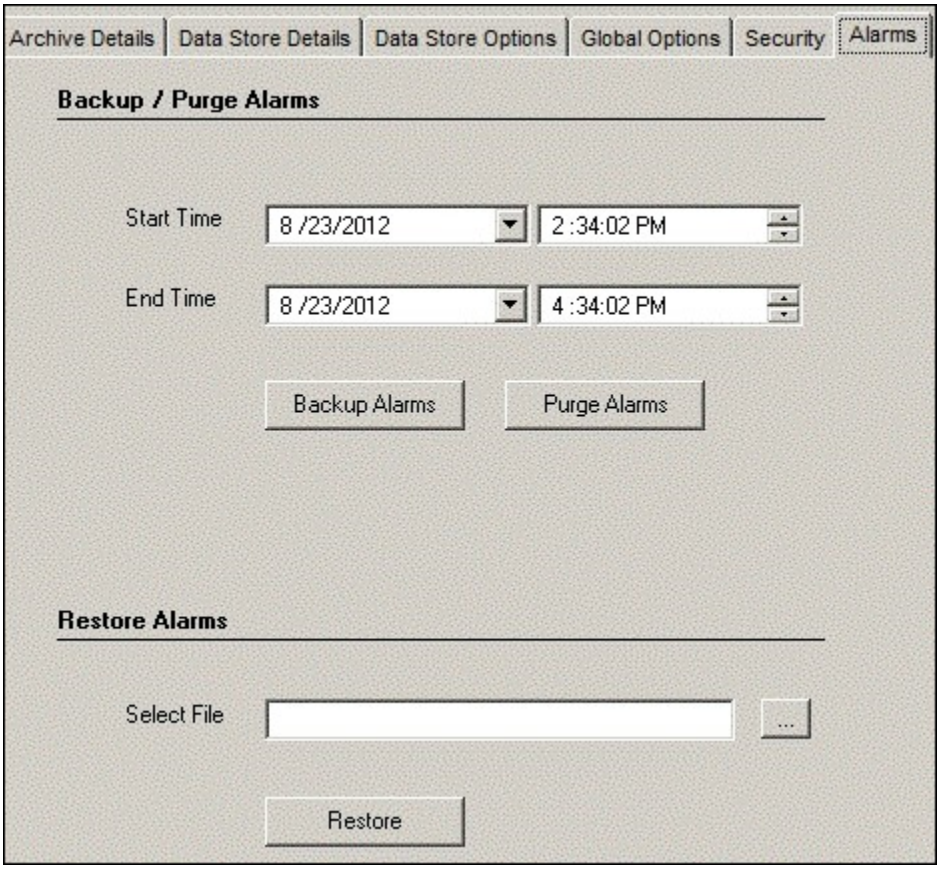
The electronic signatures/records option assists users with government regulations such as the United States Food and Drug Administration's (FDA) 21 CFR Part 11 regulation or any site interested in added security by providing the ability to require a signature and password every time a change in data or configuration is requested. If you did not purchase the Electronic Signatures and Electronic Records option, the Electronic Signatures/Records field is disabled. For more information on Electronic Signatures and Records, refer to the Using Historian in a Regulated Environment *(on page 824)* section.

| Field | Description |
|---|---|
| Require Point Verification | Indicates whether you must enter identifying information whenever you attempt a restricted action. Whenever you attempt to change the system configuration (for the tag, archive, or collector), a tag value, or another record, you |

| Field | Description |
|---|---|
| | must electronically sign the action with a username and password. If the user is authorized to make this change, the identity of the person, the action performed, and the time it was performed, are all recorded in the audit trail.<br><br>**Note:**<br>• The audit features are not dependent on this feature being enabled. Historian audits all user actions regardless of whether this option is enabled.<br>• If you plan to create multiple archives at the same time, select the **Disabled** option.<br><br>Enabling electronic signatures and electronic records also requires you to reverify your identity when you use the Historian Excel Add-in, modify or create a tag, or import data or messages.<br><br>**Note:**<br>This feature is available only if you have purchased the Electronic Signatures and Electronic Records option. |
| Verification Message | When point verification is enabled, you are prompted to enter the username and password whenever you attempt to perform an action specified as requiring point verification. |

## The **Alarms** Section

In the **Alarms** section, you can back up, purge, and restore alarms. This topic describes the fields in each subsection in the **Alarms** section.




The Alarms tab contains the following action buttons. Select a button to perform the action indicated by the name.

In the **Alarms**, you can set the following parameters:

## The Backup / Purge Alarms Subsection

**Table 62. Alarm Parameters**

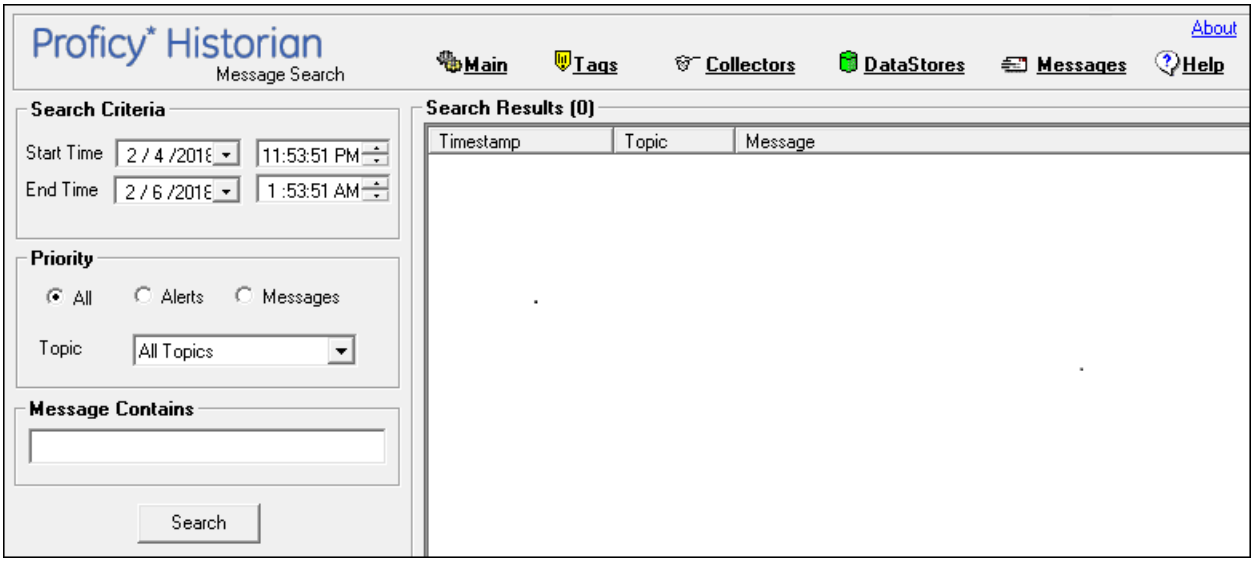
| Field | Description |
|---|---|
| Start Time | The start time for backing up, purging, or restoring alarms. Enter a value in the following format: mm-dd-yyyy hh:mm:ss |
| End Time | The end time for backing up, purging, or restoring alarms. Enter a value in the following format: mm-dd-yyyy hh:mm:ss |

### The Restore Alarms Subsection

| Field | Description |
|---|---|
| Select File | The backup file that you want to restore. |

## Searching in Message Panel

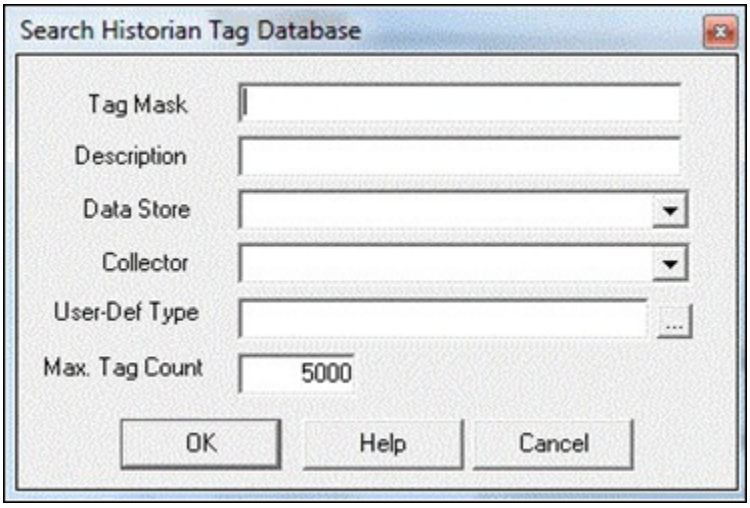
The **Message Search** page, shown in the following figure, lets you enter search parameters, such as start and end times, and to limit the search to alerts only or messages only. It further refines the search by topic and a text mask.




1. Enter a start time and end time (required).
   If your start date and end date are identical, you must enter a timestamp with the date.
2. Select **All/ Alerts/Messages**.
3. Select a **Topic** (optional).
4. Enter a text mask in the **Message Contains** field. (Optional).
   If you do not specify a text mask, all items for the associated alert or message are returned. Use a text substring for a mask. The **Message Contains** field does not accept wildcard characters.
5. Select **Search**.
   The search results are displayed in the **Search Results** panel.

## Searching for Tags

1. Select the **Search Historian Tag Database** link in the **Tag Maintenance** page
   The **Search Historian Tag Database** window in the following figure appears.




2. Enter a tag mask or a description mask in the appropriate fields, using standard Windows wildcard characters.
3. To limit the search to a specific collector, select a collector from the drop-down list in the **Collector** field.
4. Enter the maximum number of tags the search should return.
   Entering 0 (zero) will return ALL tags available in the Historian Tag Database.
5. Select **OK** to execute the search.

# Managing Data Stores

## About Data Stores

A data store is a logical collection of tags. It is used to store, organize, and manage tags according to the data source and storage requirements. A data store can have multiple archive files (*.IHA), and includes both logical and physical storage definitions.

Tags can be segregated into separate archives through the use of data stores. The primary use of data stores is to segregate tags by data collection intervals. For example, you can put a name plate or static tags where the value rarely changes into one data store, and your process tags into another data store. This can improve query performance.

Historian data stores are stored as archive files that contain data gathered from all data sources during a specific period of time. You can write and read data from the archive files.

You can define two types of data stores:

- **Historical Data Store**: Tags stored under historical data store will store data as long as the disk space is available. Depending on your license, you may be able to create multiple historical data stores. The maximum number of historical data stores supported depends on the license.
- **SCADA Buffer Data Store**: Tags stored under the SCADA buffer data store will store data for a specific duration of time based on license.

When you install the Historian server, two historical data stores are installed by default.

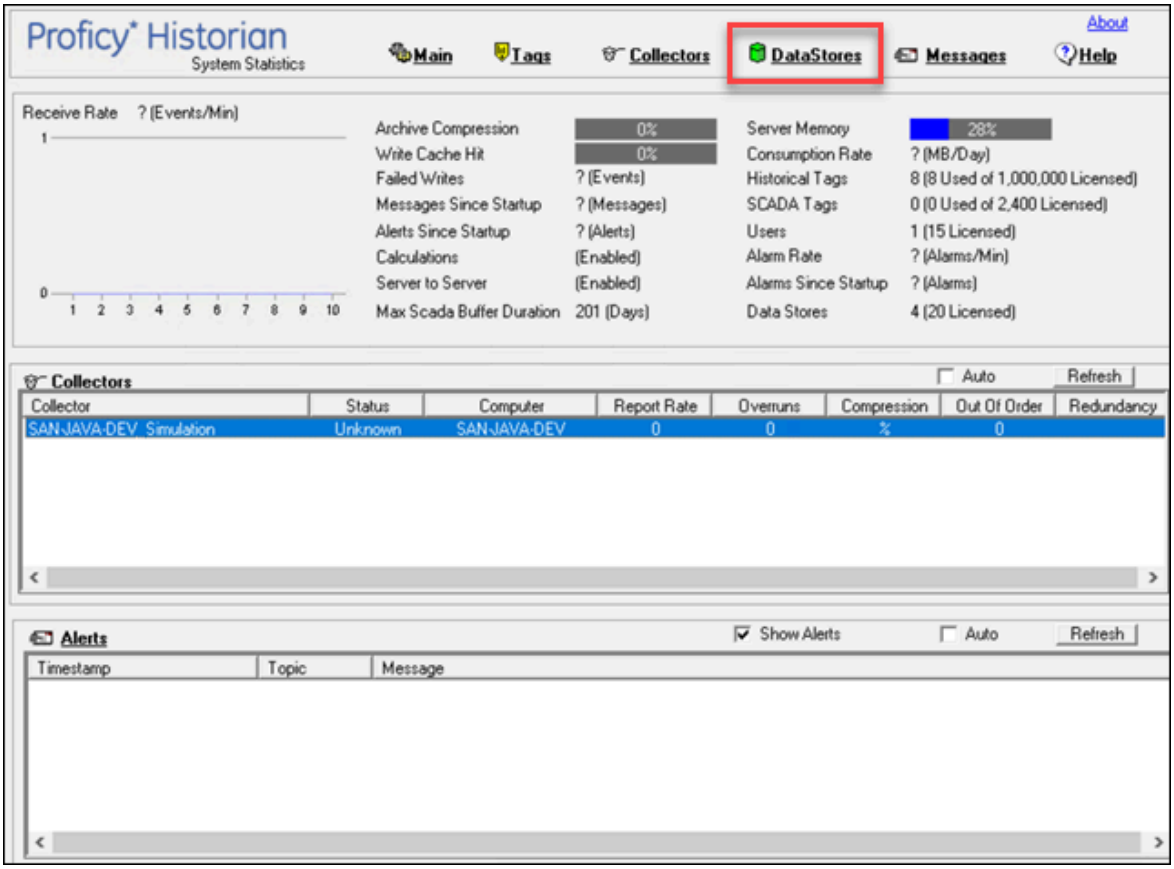
- **System**: Stores Historian messages and performance tags. This is only for internal usage within Historian, and you cannot add tags to this data store. You cannot rename or delete the system data store.
- **User**: Stores tag data. This is a default data store. You can rename and delete a user data store as long as there is another default data store set for tag addition.

Based on your license, a SCADA Buffer data store may also be installed. It stores short-term tags and data.

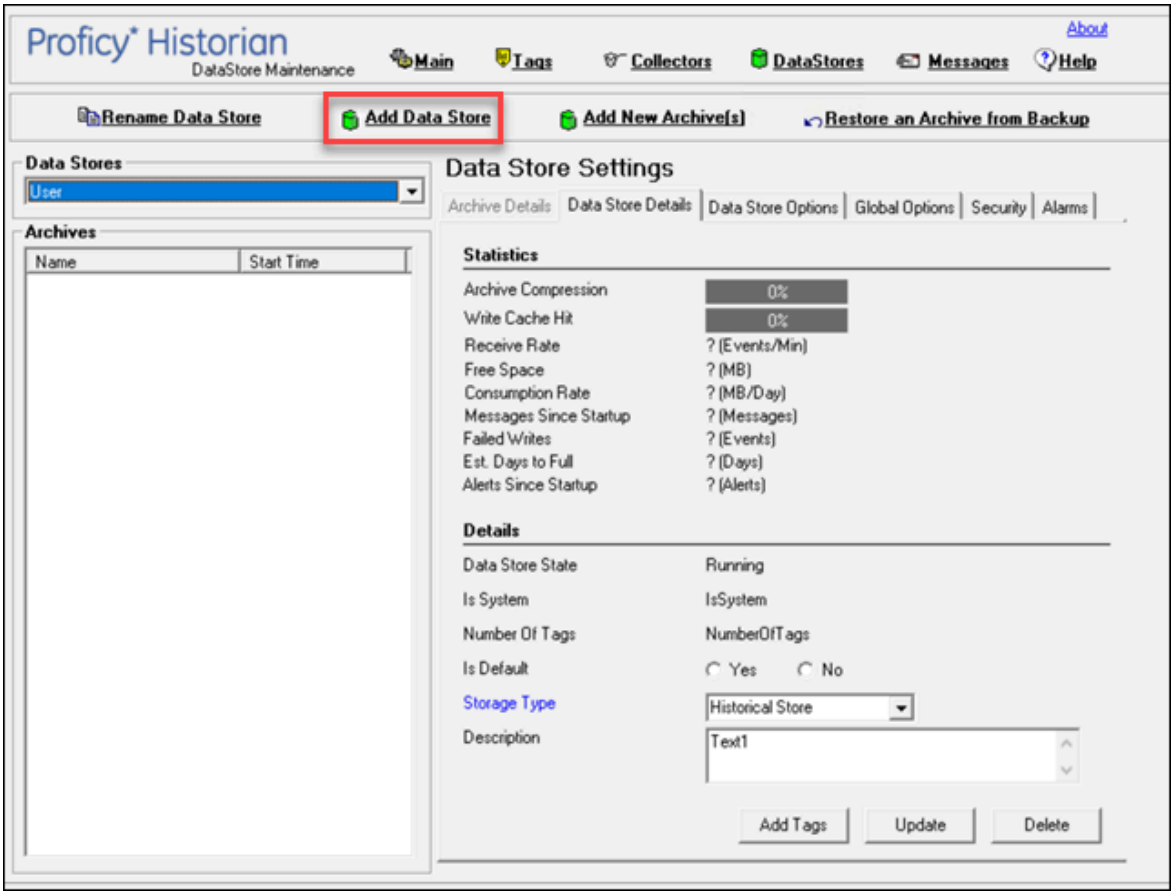
## Create a Data Store

Depending on your license, you can create or add multiple data stores.

1. Access Historian Administrator *(on page 823)*.
2. Select **DataStores**.




3. Select **Add Data Store**.




The **Add New Data Store** window appears.

4. Enter values as described in the following table.

| Field | Description |
|---|---|
| **Data Store Name** | Enter a unique name for the data store. The following characters are not allowed: `/\ \ * ? < > \|` |
| **Default Data Store** | Select this check box to set this data store as the default one for adding tags. A default data store is the one that is considered if you do not specify a data store while adding a tag. You can set only one data store as default. |
| **Description** | Enter a description for the data store. |

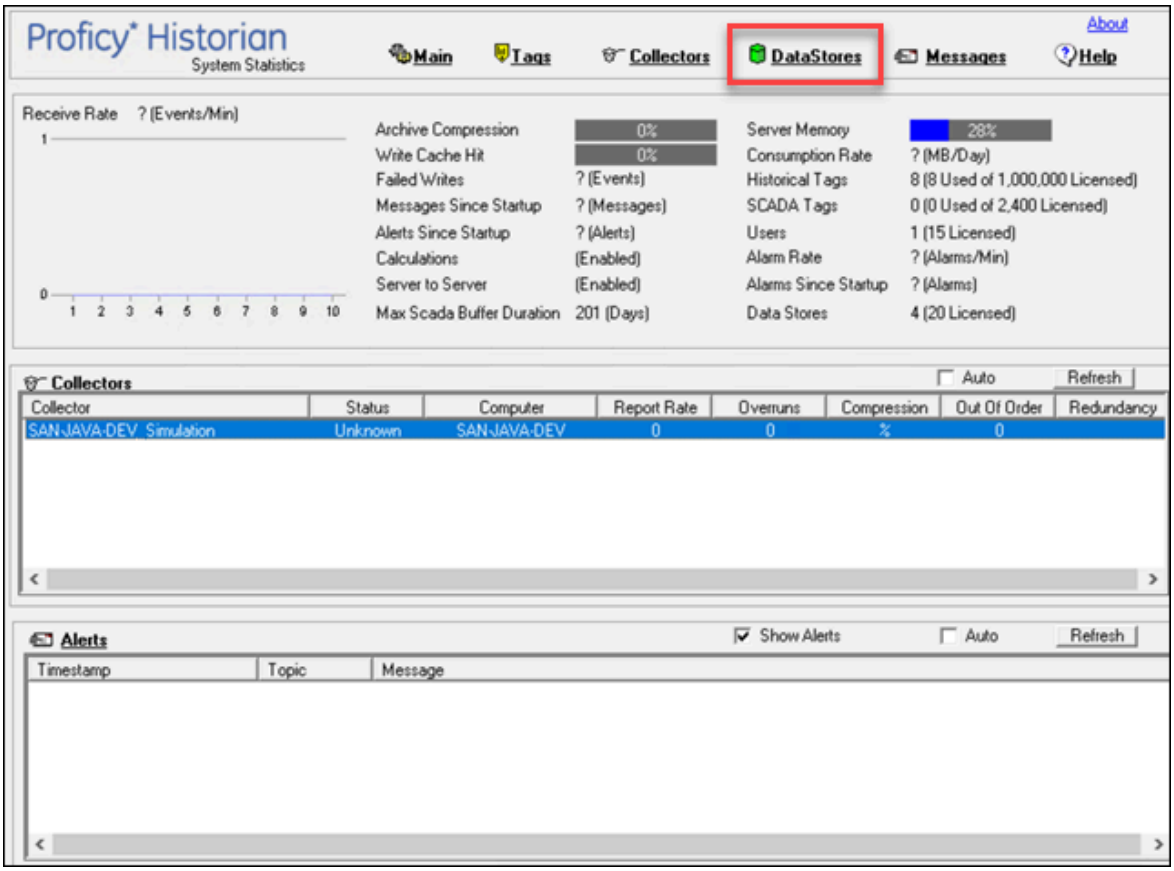
5. Select **OK**.

The data store is created.

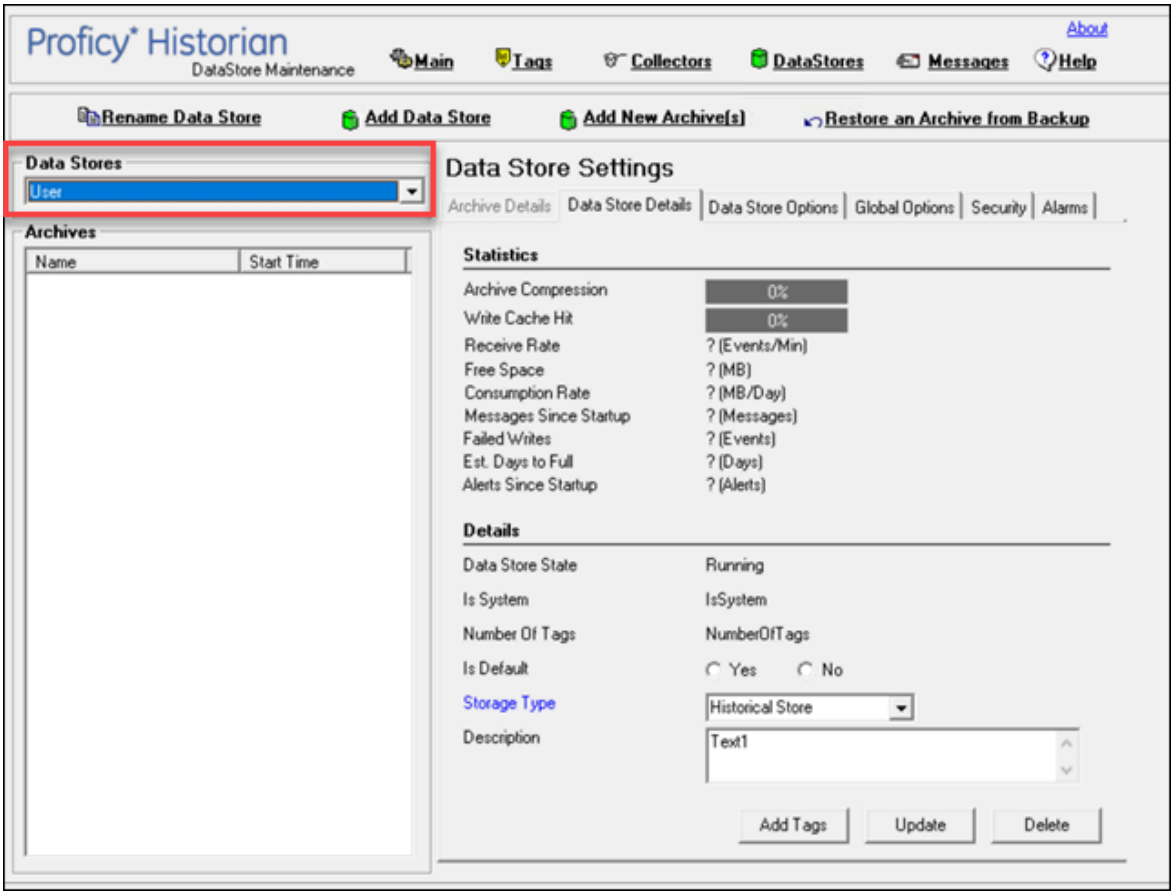
When you add tags to the data store, it will have its own set of .IHA (iHistorian Archive) files. Ensure that you back up the new data store archives periodically.

## Rename a Data Store

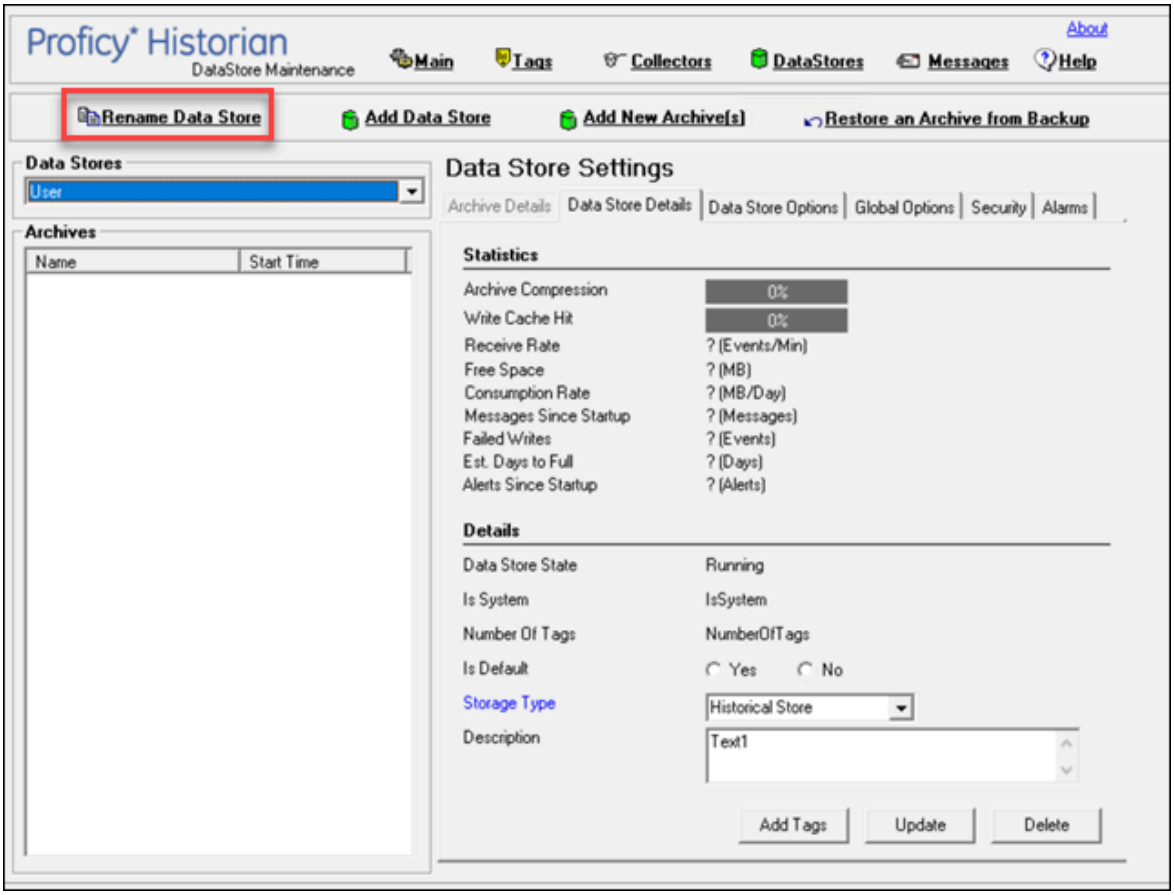
1. Access Historian Administrator *(on page 823)*.
2. Select **DataStores**.




3. In the **Data Stores** field, select the data store that you want to rename.




4. Select **Rename Data Store**.




The **Rename New Data Store** window appears.

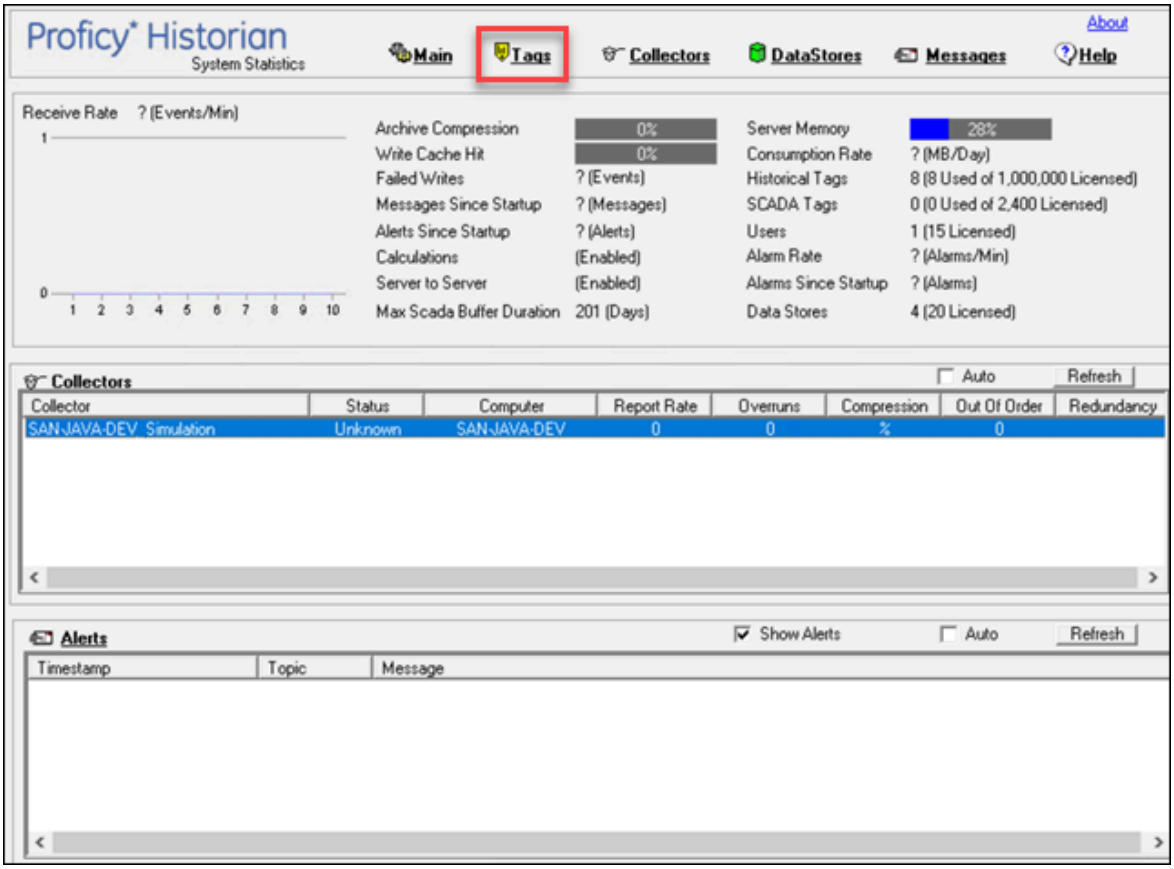
5. In the **New Data Store Name** field, enter the new name. The following special characters cannot be used in data store names: `/\ \ * ? < > |`
6. Select **Rename**.

   The data store is renamed.

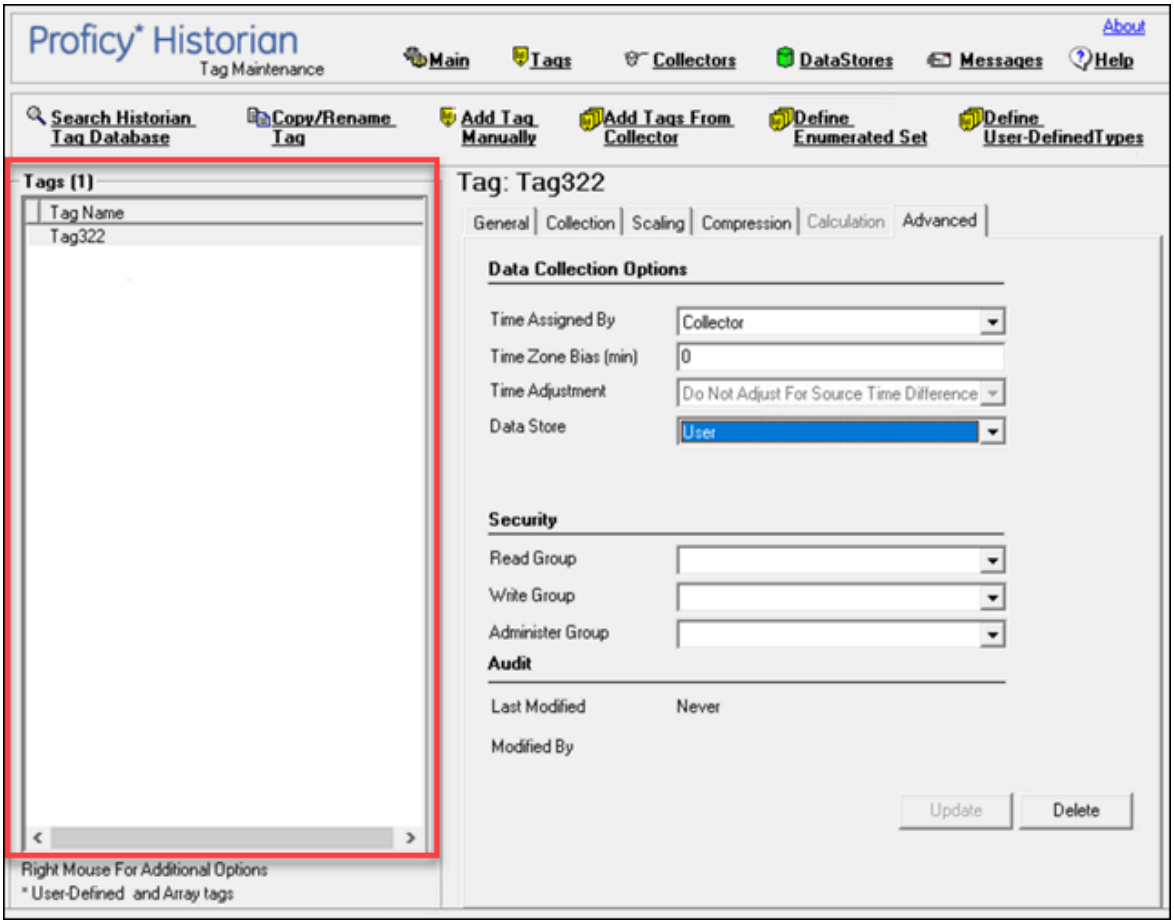
## Move a Tag to Another Data Store

You can move tags from one data store to another. However, moving a tag does not automatically move the data associated with it. If you want to retrieve the data stored before the tag was moved, you have to move the data manually using the migration utility tool.

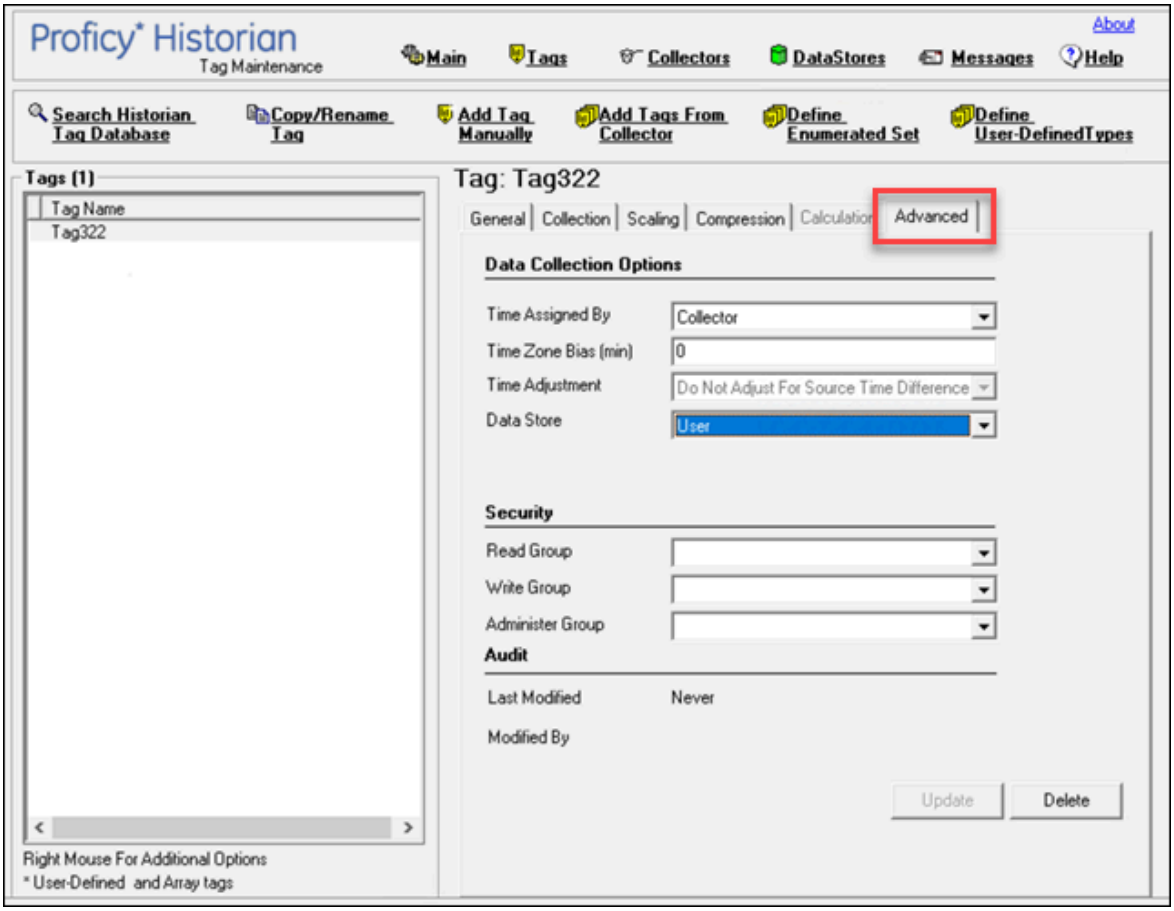
1. Access Historian Administrator *(on page 823)*.
2. Select **Tags**.




3. Select the tag that you want to move to a different data store.




4. Select **Advanced**.




5. In the **Data Store** field, select the data store to which you want to move the tag.
   A message appears, asking you to confirm that the you want to move the tag.
6. Select **Yes**, and then select **Update**.
   The tag has been moved. The new data for the tag will be stored in the new data store. However, if you want to store the old data as well in the new data store, you must manually migrate the tag data.

## Delete a Data Store

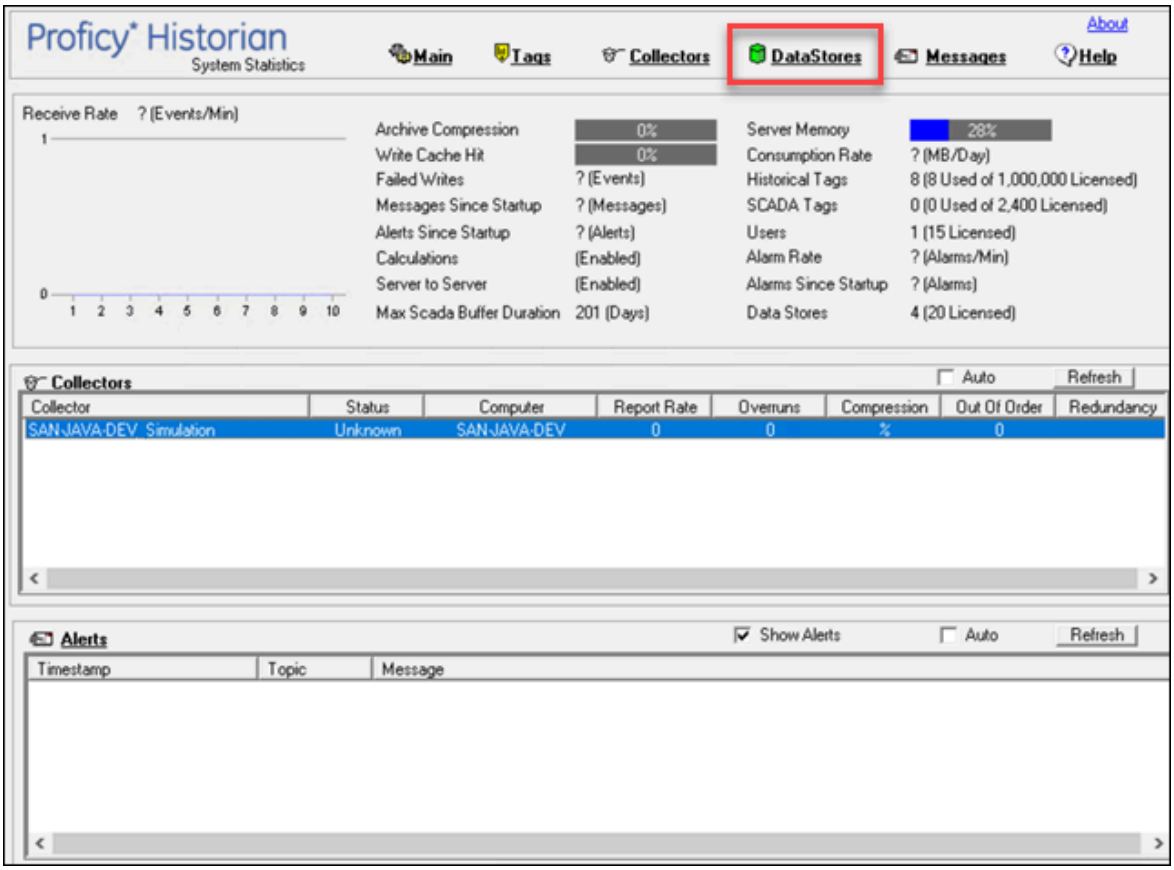
You can delete a data store when it is no longer needed.

**Note:**

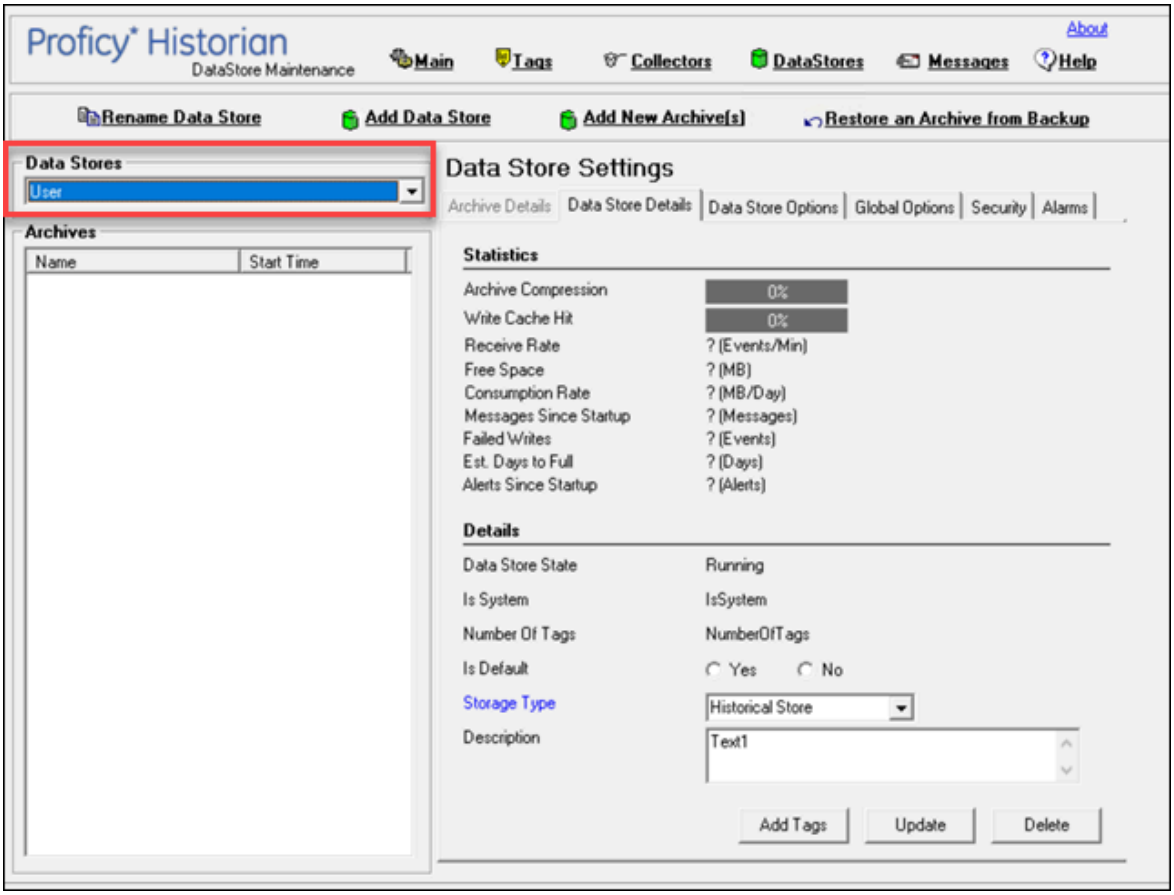
- You can only delete user data stores. You cannot delete the system data store.
- If you have only one user data store, you cannot delete it.

If there are any tags assigned to the data store, reassign them and manually move the data to another data store.

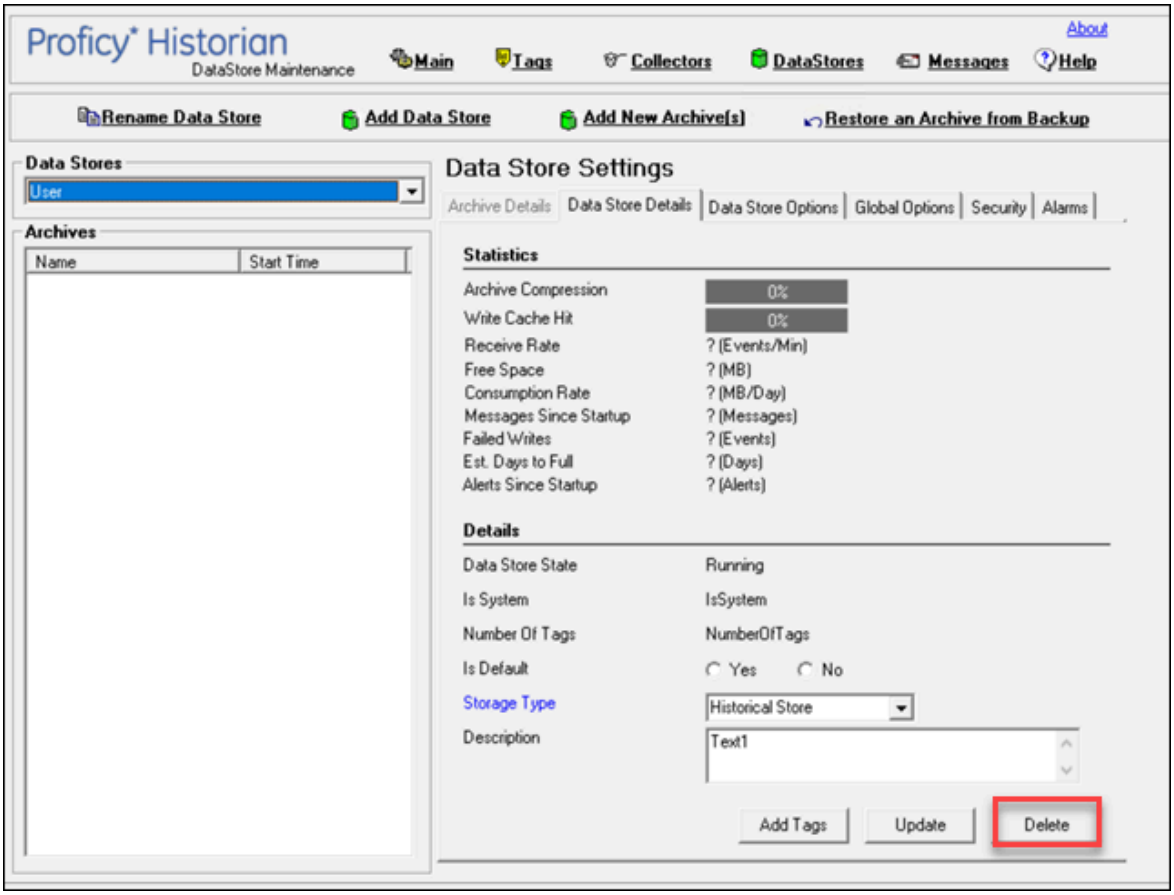
1. Access Historian Administrator *(on page 823)*.
2. Select **DataStores**.




3. In the **Data Stores** field, select the data store that you want to delete.




4. Select **Delete**.




A message appears, asking you to confirm that you want to delete the data store.

5. Select **Yes**.
   The data store is deleted.

# Managing Archives

## About Archives

Historian archives are data files, each of which contains data gathered from all data sources during a specific period of time.

### Types of Archive Files:

- ***machine name*_Config.ihc:** Contains information about the archiver, tag configuration, and collector configuration.
- ***machine name*_ArchiveXXX.iha:** Contains tag data, where x is a number indicating the place of the file in a time-based sequence.

### Creation of Archive Files Automatically

Archive files grow to a user-configured maximum size as data is recorded by the server. When data starts loading into an archive file, Historian will automatically create a new blank archive file. When the current archive file becomes full, Historian will immediately serve data to the newly created archive file. This significantly reduces archive creation and transition time.

If, however, the option to automatically create archive files is not enabled, you must create an archive file manually *(on page 868)*.

> **Note:**
> If the option to automatically create an archive is not enabled and you do not create a new archive manually, or if the available disk space is less than the required amount of free disk space, a new archive file will not be created.

### Overriding Old Archive Files

If you enable the **Overwrite Old Archives** option, the system replaces the oldest archived data with new data when the latest archive default size has been reached. Since this action deletes historical data, exercise caution in using this feature. Be sure that you have a backup of the archive so that you can restore it later. Best practice is to create an additional archive to prevent premature loss of data due to overwriting. For example, if you want to save 12 months of data into 12 archives, create 13 archives.

During archiver startup and every 60 seconds while the server is running, Historian verifies that you have configured enough free disk space to save the archives, buffer files, and log files. If there is insufficient disk space, the Data Archiver shuts down and a message is logged into the log file. By default, you can view the Historian archiver log file in `C:\Historian Data\LogFiles`.

```
[03/03/10 15:28:41.398] Insufficient space available in [d:\Historian\Archives\]
        [03/03/10 15:28:41.399] The server requires a minimum of [5000 MB] to continue
        [03/03/10 15:28:41.679] USER: DataArchiver TOPIC: ServiceControl MSG: DataArchiver(DataArchiver)
        Archiver shutdown at 03/03/10 15:28:41.653
        [03/03/10 15:28:41.807] DataArchiver Service Stopped.
        [03/03/10 15:28:41.809] [d:\Historian\LogFiles\DataArchiver-34.log] Closed.
```

## About Remote Storage of Archives

Historian can store current and backup archive files on remote storage devices. These remote storage devices can consist of network shares, Storage Area Networks (SAN), or Network Attached Storage (NAS)

devices. You can also employ a hybrid approach, by storing your current archive on the local disk and writing a script using the Historian SDK to migrate older archives to a remote storage device.

A SAN is a dedicated network apart from a LAN, specifically configured to allow servers to communicate with large storage arrays, usually over fibre-optic cables. The SAN is directly accessible as a disk device to Windows and does not operate through the slower network layer. NAS devices are similar to a SAN, but are directly attached to the LAN, appearing as a server.

**Note:**
Though NAS devices mapped to drives in Windows appear in Historian's Browse windows, they are not supported by the Data Archiver service. In order to access NAS devices in Historian, you must enter their UNC path (\\iHbackup\archive, for example).

If the Data Archiver service is not working, view the messages. Messages like these indicate that Historian is trying to use mapped drives: `"Path N:\ not found." "Mapped drives unsupported as archive or backup file location" "UNC paths inaccessible when running as LocalSystem, configure other logon account for DataArchiver service".` If these messages appear, find and change mapped drives to UNC paths.

### Example: Migrating Non-Current Archives to a Remote Location

The following sample VBScript code will migrate non-current archive to a remote location by accessing the Historian SDK. This script could be run at specific intervals to migrate data from a local disk to a network share.

The Data Archiver service must have permission to write to the remote storage device. If you have configured alarms and events to use an SQL server, the SQL server must also have permissions to write to the remote storage device.

```
Dim FileSystem

set FileSystem = CreateObject("Scripting.FileSystemObject") Dim remoteLocation

remoteLocation = "\\StorageServer\Historian\Archives"

If InStrRev(remoteLocation, "\") Len(remoteLocation) Then remoteLocation = remoteLocation & "\"

' make sure we can access the remote storage before proceeding

Dim TestFile

set TestFile = FileSystem.OpenTextFile(remoteLocation & "Test.txt", 2, True, 0) If TestFile is Nothing Then

err.Raise 1, , "Unable to access remote storage location" End If

TestFile.Close

FileSystem.DeleteFile remoteLocation & "Test.txt" Dim Server
```

```
set Server = CreateObject("iHistorian_SDK.Server") If Server is Nothing Then

err.Raise 1, , "Unable to create iHistorian_SDK.Server object" End If

Dim archivesToMigrate(), i

If Server.Connect() Then

With Server.Archives

Dim backupPath

backupPath = .ArchivingOptions("ArchiveBackupPath")

If InStrRev(backupPath, "\") Len(backupPath) Then backupPath = backupPath & "\" ReDim archivesToMigrate(.Item.Count, 2)

For Each archive in .Item

If Not archive.IsCurrent Then

If UCase(Left(archive.FileName, InStrRev(archive.Filename, "\"))) UCase(remoteLocation) Then i = i + 1

archivesToMigrate(i, 0) = archive.Name

archivesToMigrate(i, 1) = backupPath & "Offline\" & cstr(archive.Name) & ".zip" archivesToMigrate(i, 2) =

 archive.FileSizeTarget

End If

End If

Next

Dim j

For j = 1 To i

If .Delete(cstr(archivesToMigrate(j, 0))) Then FileSystem.MoveFile archivesToMigrate(j, 1), remoteLocation Dim archive

set archive = .Add(cstr(archivesToMigrate(j, 0)), cstr("%%inplace%%" & remoteLocation & archivesToMigrat

If Not (archive Is Nothing) Then

FileSystem.DeleteFile remoteLocation & archivesToMigrate(j, 0) & ".zip" End If

End If

Next End With Server.Disconnect

Else

err.Raise 1, , "Failed connecting to server"

End If
```

## Guidelines for Setting Archive Size

Since archived data files can become quite large, you must adjust system parameters carefully to limit data collection to meaningful data only and thus minimize the required size of system storage. You can allocate up to 256 GB per archive.

For each archive, you need approximately 1MB of archive space for every 1000 tags to store tag information. Archive size is a function of the rate at which you archive data and the time period you want the archive to cover. A typical user wants the archive to cover a time period of, say, 30 days.

The following factors affect the rate at which you archive data:

- Number of tags
- Polling frequency of each tag
- Compression settings
- Data types

Based on these parameters, the archive size is calculated as follows:

$$\#Tags \times \frac{Values}{Tag} \times \frac{Tags}{Second} \times \%PassComp \times \frac{Bytes}{Value} \times \frac{Seconds}{Hour} \times \frac{Hours}{Day} \times \frac{MB}{Bytes} = \frac{MB}{Day}$$

### Calculating Archive Size

Suppose you want to store data, and you have the following parameters:

- Number of tags: 5000
- Polling rate: 1 value/5 seconds
- Pass compression: 5%.

  Pass compression is the number of data values archived relative to the number of values read.
- Bytes/value: 4
- Duration: 30 days

Based on the preceding formula, for the given parameters, the archive size is calculated as follows:

$$5000 \times \frac{1}{1} \times \frac{1}{5} \times \frac{5}{100} \times \frac{4}{1} \times \frac{3600}{1} \times \frac{24}{1} \times \frac{1}{1024 \times 1024} \times 30 = 494 \frac{MB}{Month}$$

The calculation shows that a file size of 500 MB is adequate for archiving one month of data for this application.

Therefore, we recommend that you set the default archive size to 500 MB for systems with 1000 tags or more. If you believe the computed size is too large for your application, you can modify parameters as follows:

- Decrease the polling frequency.
- Increase compression deadband, reducing the pass percentage.

- Reduce the number of tags.
- Add more disk capacity to your computer.

### Archive Size Calculator

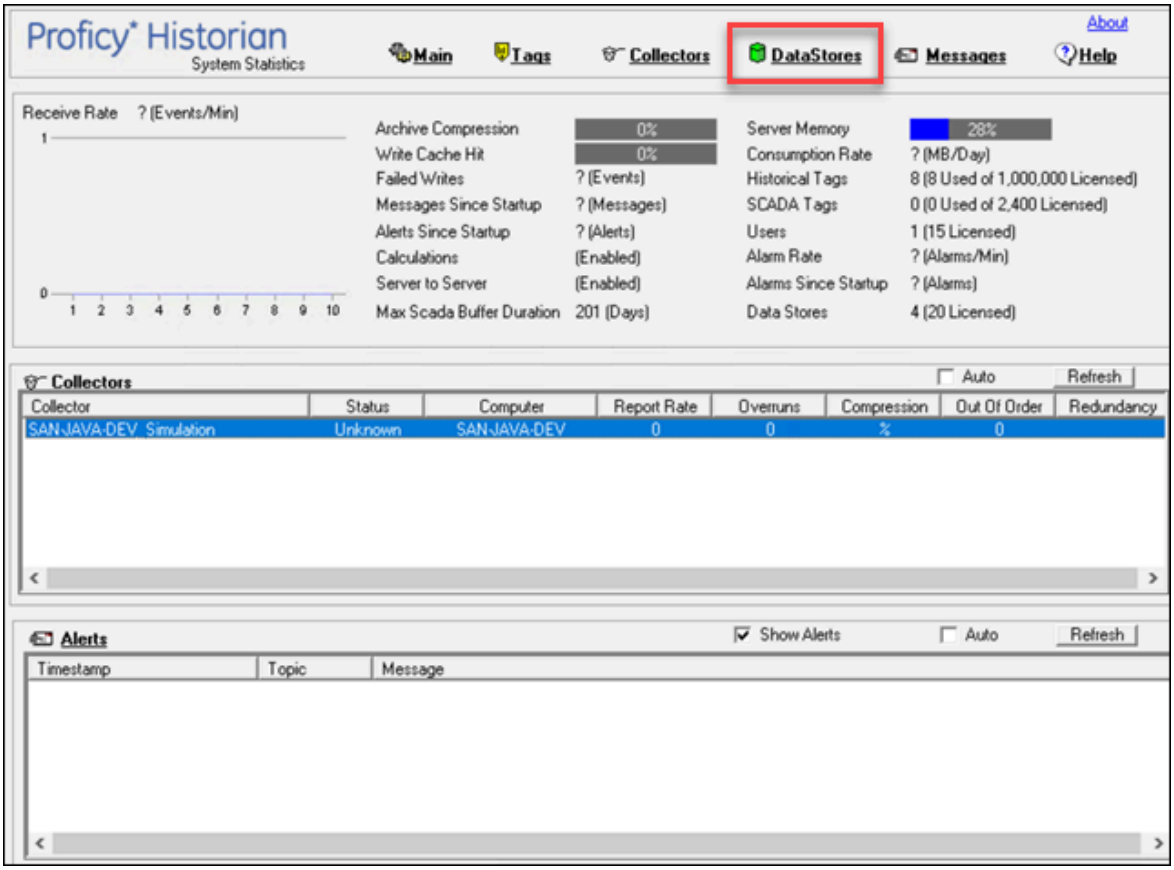
An archive size calculator tool is available to estimate archive size and collector compression based on a tag that has already been configured or based on your inputs. Log on to http://digitalsupport.ge.com to download this tool and other GE Intelligent Platforms freeware product solutions.

## Create an Archive Automatically

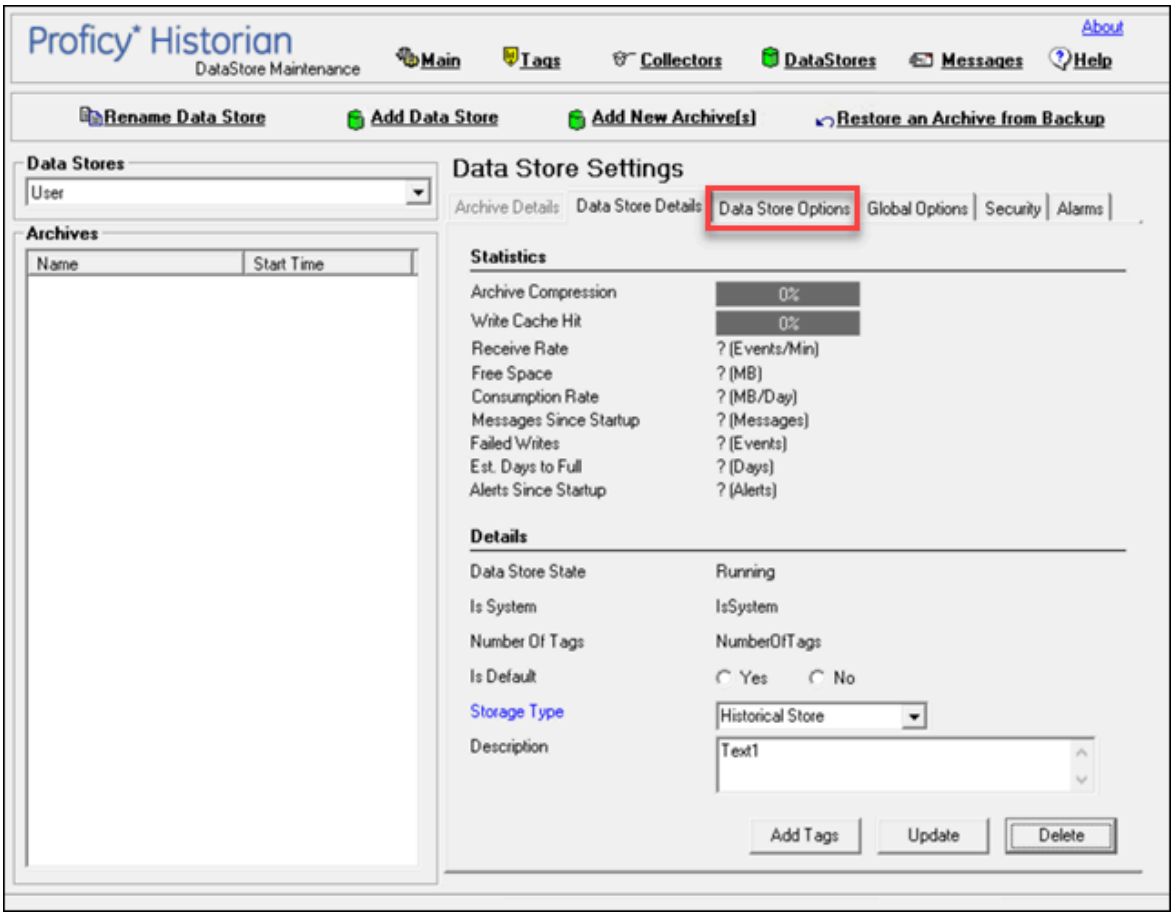
When the current archive reaches a specified size or duration, you can configure Historian to create a new archive automatically. You can also create an archive manually *(on page 870)*. When the current archive is full, the new one is used.

You can allocate maximum 256 GB for an archive.

1. Access Historian Administrator *(on page 823)*.
2. Select **DataStores**.



3. Select **Data Store Options**.




4. Enter values as described in the following table.

| Field | Description |
|---|---|
| **Automatically Create Archives** | Select **Enabled**. |
| **Overwrite Old Archives** | Specify whether you want to overwrite old archives with new ones. Exercise caution in enabling this option. We recommend that you backup archives if you want to enable this option. |
| **Default Size** | Enter the size of the current archive after which you want to create a new archive. This field is available only if you have selected **By Size** in the adjacent drop-down list box.<br><br>If, however, you want to create archives after a duration, select **Days** or **Hours**, and then enter the value in the **Archive Duration** field. |

| Field | Description |
|---|---|
| **Archive Duration** | Enter the duration after which you want to create a new archive. This field is available only if you select **Days** or **Hours** in the adjacent drop-down list box.<br><br>If, however, you want to create an archive when the current one reaches a particular size, select **By Size**, and then enter the value in the **Default Size** field. |
| **Default Archive Path** | Enter the path to the folder in which you want to store the archive files. |
| **Default Backup Path** | Enter the path to the folder in which you want to store the backup files. |
| **Base Archive Name** | Enter the base name for the archive. This name will be appended with a number to identify the sequence of the archive. |
| **Free Space Required** | Enter the free space that is required to create the archives. |
| **Store OPC Quality** | Specify whether you want to store OPC quality in the archive. |
| **Use Caching** | Specify whether you want to use caching in the archive. |
| **Data is Read-Only After (Hours)** | Specify the duration, in hours, after which you want to archive to be read-only. |
| **Generate Message on Data Update** | Specify whether you want to generate a message when data is updated in the archive. |

5. Select **Update**.
   Archives will be created automatically when the current one reaches the size (or after the duration) that you have specified.

## Create Archives Manually

If you want to create multiple archives at the same time, access Historian Administrator, and set values for the following fields:

| Field | Value |
|---|---|
| The **Details** Section | |
| **File Attribute** | **Read/Write** |

| Field | Value |
|---|---|
| The **Global Options** Section | |
| Maximum Query Time (seconds) | 60 |
| Maximum Query Intervals | 100000 |
| Automatically Create Archives | Disabled |
| Overwrite Old Archives | Enabled |
| Maintain Auto Recovery Files | Enabled |
| Store OPC Quality | Disabled |
| The **Security** section | |
| Data is Readonly After (Hours) | 1 month |
| Security Groups | Use local |
| Generate Message on Data Update | Disabled |
| Require Point Verification | Disabled |

This topic describes how to create archives manually. You can also create them automatically *(on page 868)*. When the current archive is full, a new archive is used (in a sequential order).

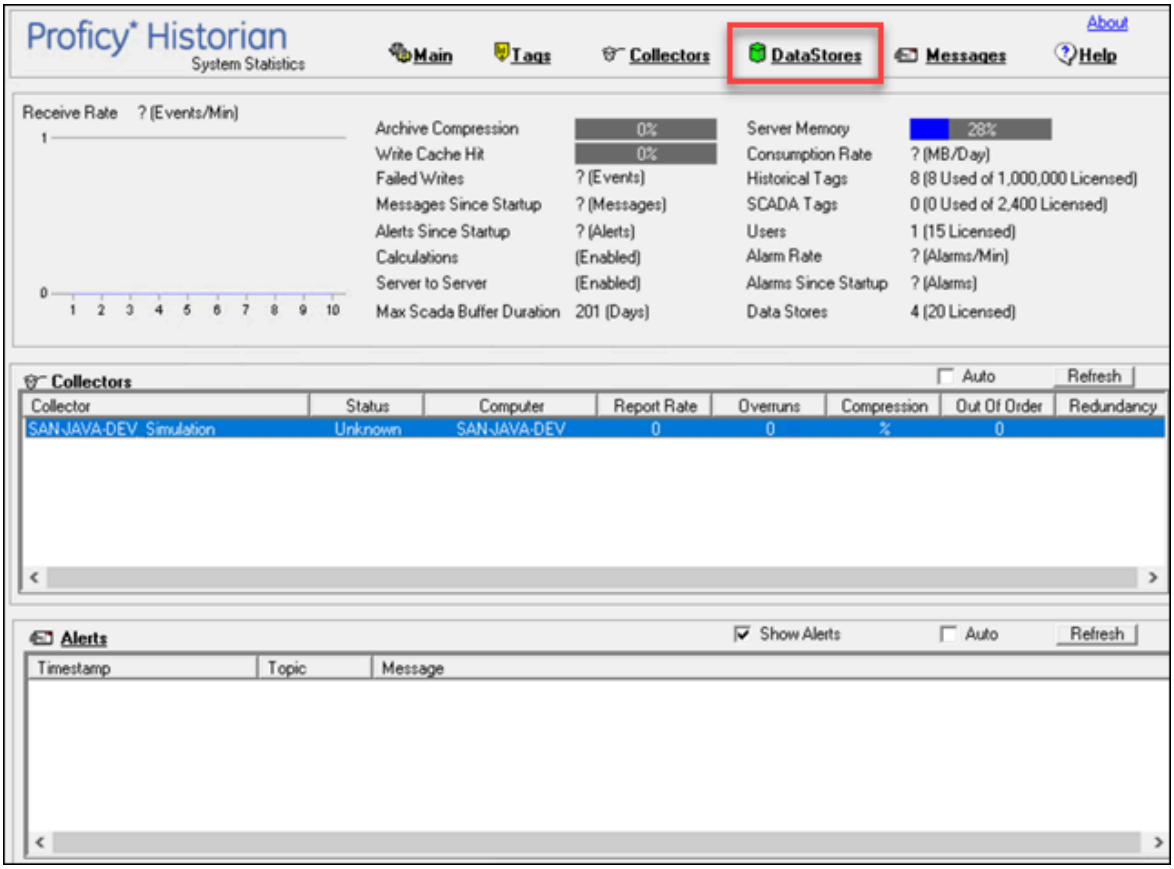
You can create multiple archives at the same time.

**CAUTION:**

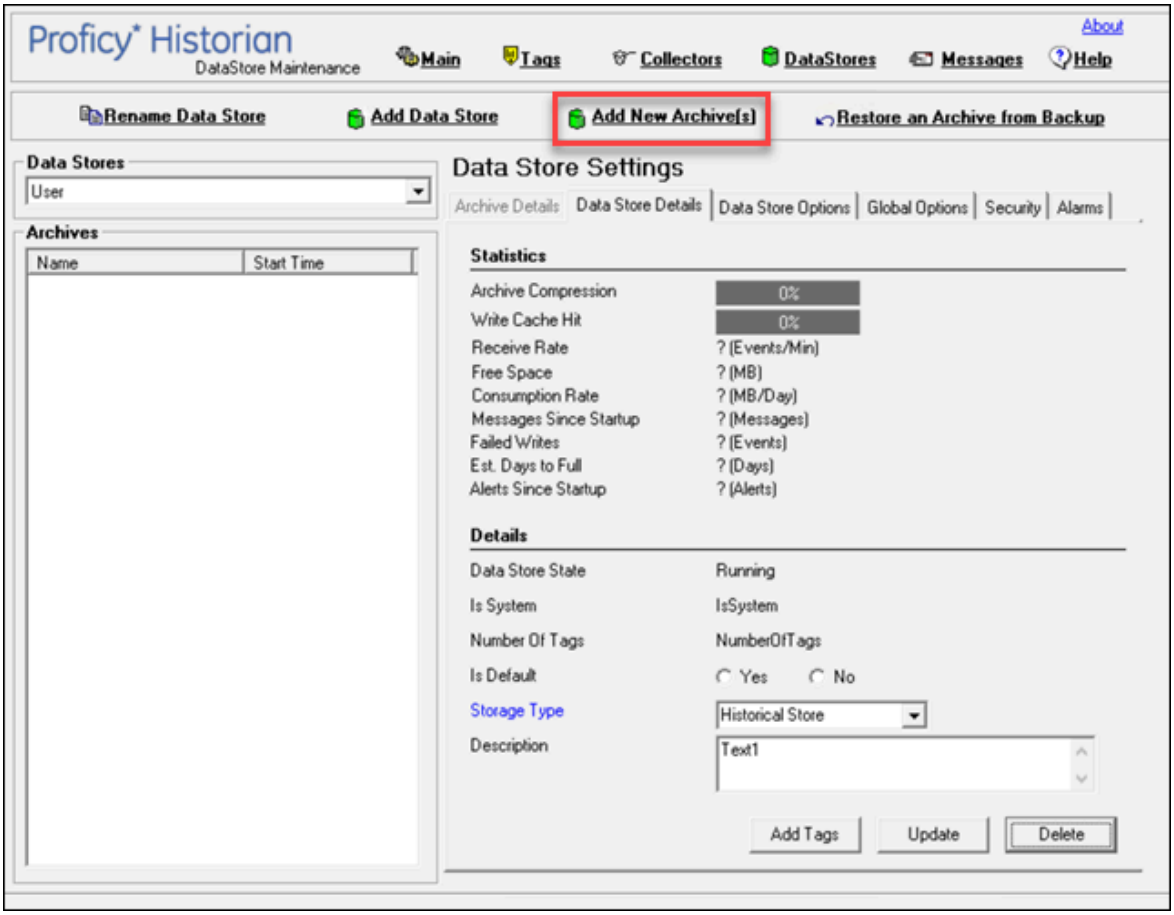
If you want to creating multiple archives on a remote machine, you must ensure that you have enough hard disk space on that machine. The **Allocate Space** field does not display a remote machine's hard disk space; if you are creating multiple archives on a remote machine, you must ignore the ”r;percentage of available disk space will be used” message displayed by the Allocate Space slider.

If you receive the error message ”r;Runtime error 330 Invalid Property Value" while creating multiple archives on a remote machine, it is probably because you do not have enough hard disk space on that machine. When you select OK on the error message, Historian Administrator may disappear. You must now clean up the remote machine's hard disk space and restart Historian Administrator.

1. Access Historian Administrator *(on page 823)*.
2. Select **DataStores**.




3. Select **Add New Archive(s)**.




The **Add New Archive(s)** window appears.

4. Enter values as described in the following table.

| Field | Description |
|---|---|
| **Archive Name** | Enter a unique name for the archives. The value must be the same as the file name. When multiple archives are created, a number is appended to the name to make each name unique (and to maintain a sequence). |
| **Data Store** | Select the data store in which you want to create the archives. |
| **File Location** | Enter the path to the folder in which you want to store the archives, or specify a UNC path. |
| **EachArchive Size (MB)** | Enter the size, in MB, that you want to allocate to the archives. |
| **Number of Archives** | Enter the number of archives you want to create. |

| Field | Description |
|---|---|
| **Allocate Space** | Specify the percentage of the disk space that you want to allocate for archives. As you increase the space, the number of archives increases accordingly.<br><br>**Note:**<br>The **Allocate Space** field does not display a remote machine's hard disk space; if you are creating multiple archives on a remote machine, you must ignore the ”r;percentage of available disk space will be used” message displayed by the Allocate Space slider. |

5. Select **OK**.
   The archives are created.

## Back up Archives Automatically

You must back up archive files periodically to ensure that your data is protected. These backup files contain tag data as well as alarms and events data. You can send these files to a shared network location or to physical media.

Planning for data recovery means always having up-to-date backup files for important information that you can call up and restore quickly when the need arises.

The .IHC file, which contains all configuration information, is an important file to back up. The .IHC file is automatically backed up when, and only when, you back up the current archive .IHA file. With the .IHC file, you can always restore the system configuration to the state it was in before the event occurred.

It is also important to backup the current online archive files *.IHA. If you restore the archive files, along with the configuration, you can quickly pick up where you left off when the event occurred with a minimum loss of data.

By default, the .IHC backup path is the same as the archives path. The .IHC uses the following naming convention: `ComputerName_Config-Backup.ihc`. If the default backup path is different than the archives path, the .IHC file is copied to the backup folder with the standard .IHC naming convention: `ComputerName_Config.ihc`.

**Important points to remember:**

- The .IHC file is automatically backed up when, and only when, you back up the current archive .IHA file. By default, the .IHC backup path is the same as the archives path.
- The .IHC backup file uses the following naming convention: `<system name>_Config-Backup.ihc`. If the default backup path is different from the archives counterpart, the .IHC file is copied to the backup folder with the standard .IHC naming convention: `ComputerName_Config.ihc`.
- In the mirroring system, Client Manager sends a backup message to Data Archiver located on the Client Manager node to which you are connected. The back up then happens in the specified location on that node. If that Data Archiver is not running, a NOT_CONNECTED error message appears, and the backup will not happen.
- If you back up an archive more than once, by default, the backup tool will attempt to use the same name for the backup file and will detect that an archive with the same name already exists. Rename the backup archive file or move the original backup archive file to a different folder.

1. Access the `ihArchiveBackup.exe` file. By default, it is located at `C:\Program Files \Proficy\Historian\Server`.
2. Run the following command if you want to back up multiple archives at the same time:

   ```
   AT <time> /EVERY:<days of the week> "ihArchiveBackup.exe [-s <Historian server name>] [-u <Username>] [-p <Password>][-t <Timeout seconds>][-n <Number of archives>]
   ```

   Run the following command if you want to back up a single archive file:

   ```
   AT <time> /EVERY:<days of the week> "ihArchiveBackup.exe [-s <Historian server name>] [-u <username>] [-p <password>][-t <timeout seconds>] [-a <archive name>]"
   ```

   The following table describes the parameters.

| Switch | Parameter | Description |
|---|---|---|
| AT | Time | The time at which you want to back up the archives. |
| EVERY | Days of the week | The days of the week on which you want to back up the archives (for example, `EVERY:M,T,W,Th,F`). |
| -s | Historian server name | The IP address or the host name of the Historian server on which the archive is available. |
| -u | Username | The user name required to connect to the Historian server. |

| Switch | Parameter | Description |
|---|---|---|
| -p | Password | The password required to connect to the Historian server. |
| -t | Timeout seconds | The time, in seconds, to wait before the process times out. |
| -n | Number of archives | The number of archives to back up, counting backwards from the current archive. |
| -a | Archive name | The name of a specific archive to back up. |
| -c | | Backs up only the Historian configuration (.ihc) file. |
| -d | Data store name | The name of the data store that stores the archive data.<br><br>**Note:**<br>Using one instance of ihArchiveBackup.exe you can point to one data store, and by using multiple instances you can point to multiple data stores. |

If you do not provide any parameters, only the current archive file is backed up.

```
AT 23:59 /EVERY:M,T,W,Th,F "ihArchiveBackup.exe [-s MyHistorianServer] [-u User1] [-p password123][-t 10] [-a MyHistorianServer_DS1_Archive004]"
```

## Back up an Archive Manually

- Ensure you have enough hard drive space on your default backup location before backing up your archives.
- Always back up archives before a planned Historian software product upgrade.
- Use Microsoft® Volume Shadow Copy Service when backing up archive files that are more than 2 GB in size or when backing up more than the last two archives. For more information, refer to Back Up Archives with Volume Shadow Copy Service *(on page 731)*.

You must back up archive files periodically to ensure that your data is protected. These backup files contain tag data as well as alarms and events data. You can send these files to a shared network location or to physical media.

Planning for data recovery means always having up-to-date backup files for important information that you can call up and restore quickly when the need arises.

The .IHC file, which contains all configuration information, is an important file to back up. The .IHC file is automatically backed up when, and only when, you back up the current archive .IHA file. With the .IHC file, you can always restore the system configuration to the state it was in before the event occurred.

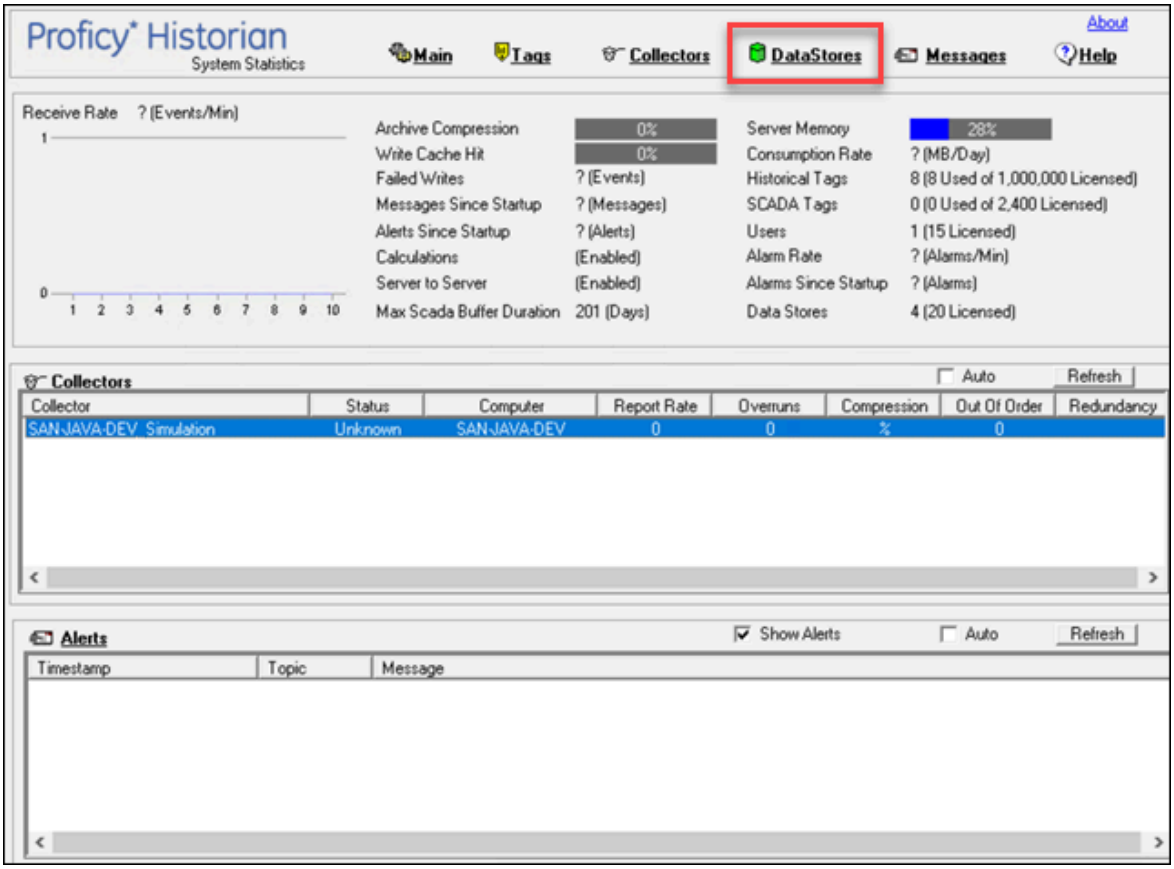
It is also important to backup the current online archive files *.IHA. If you restore the archive files, along with the configuration, you can quickly pick up where you left off when the event occurred with a minimum loss of data.

By default, the .IHC backup path is the same as the archives path. The .IHC uses the following naming convention: `ComputerName_Config-Backup.ihc`. If the default backup path is different than the archives path, the .IHC file is copied to the backup folder with the standard .IHC naming convention: `ComputerName_Config.ihc`.

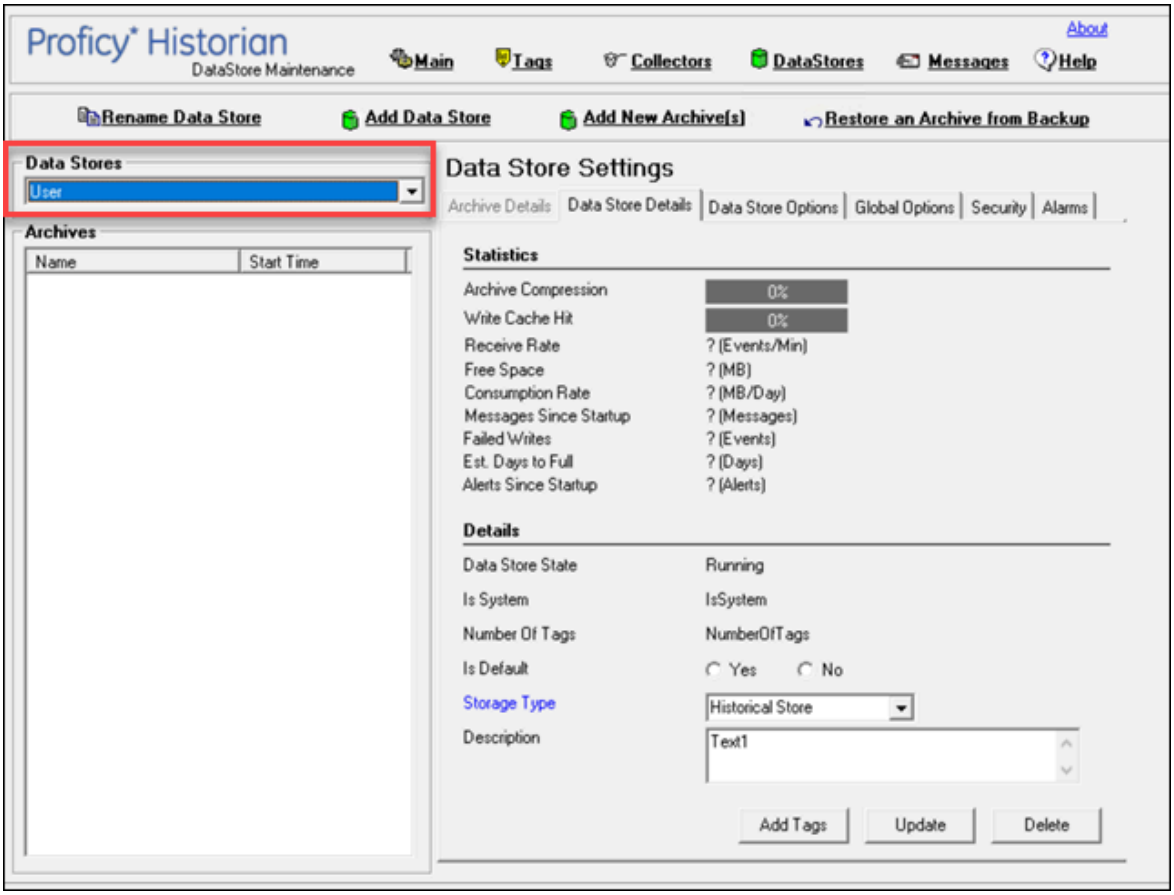
**Important points to remember:**

- The .IHC file is automatically backed up when, and only when, you back up the current archive .IHA file. By default, the .IHC backup path is the same as the archives path.
- The .IHC backup file uses the following naming convention: `<system name>_Config-Backup.ihc`. If the default backup path is different from the archives counterpart, the .IHC file is copied to the backup folder with the standard .IHC naming convention: `ComputerName_Config.ihc`.
- In the mirroring system, Client Manager sends a backup message to Data Archiver located on the Client Manager node to which you are connected. The back up then happens in the specified location on that node. If that Data Archiver is not running, a NOT_CONNECTED error message appears, and the backup will not happen.
- If you back up an archive more than once, by default, the backup tool will attempt to use the same name for the backup file and will detect that an archive with the same name already exists. Rename the backup archive file or move the original backup archive file to a different folder.

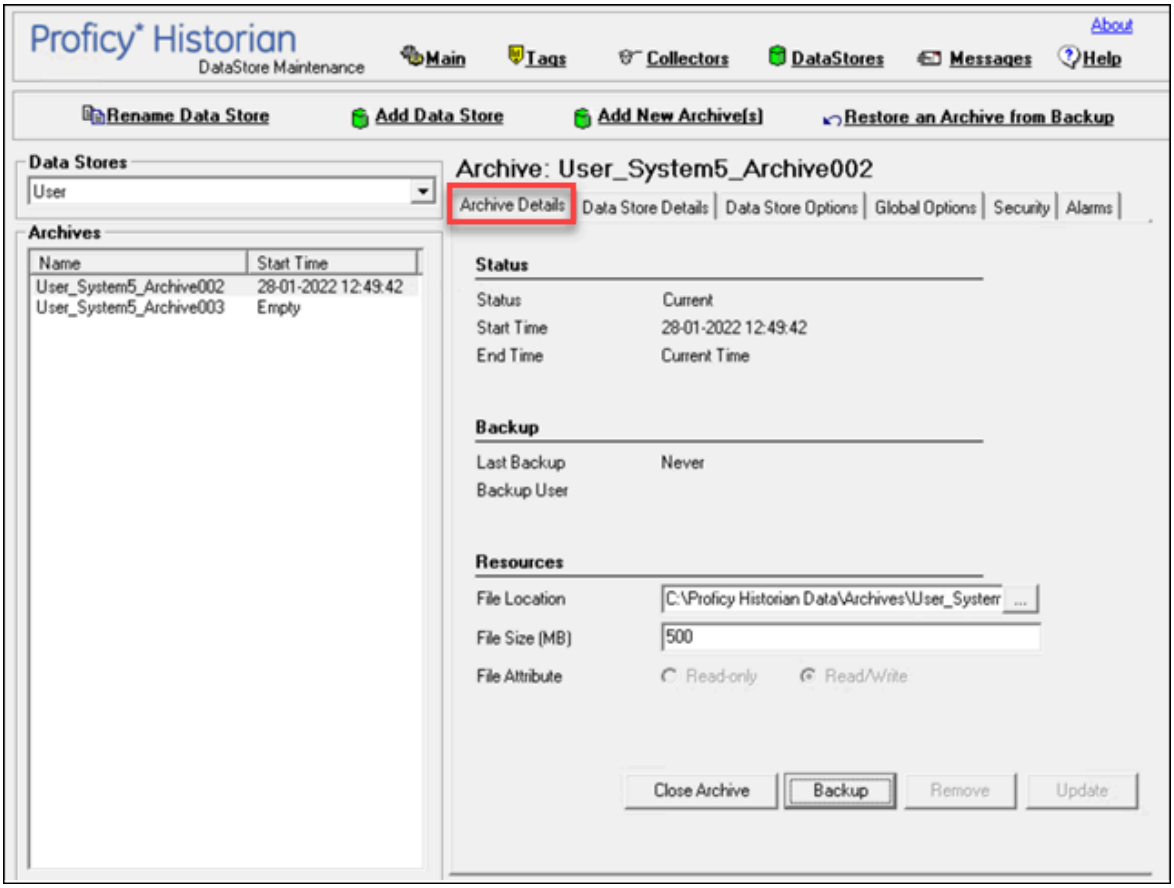
1. Access Historian Administrator *(on page 823)*.
2. Select **DataStores**.

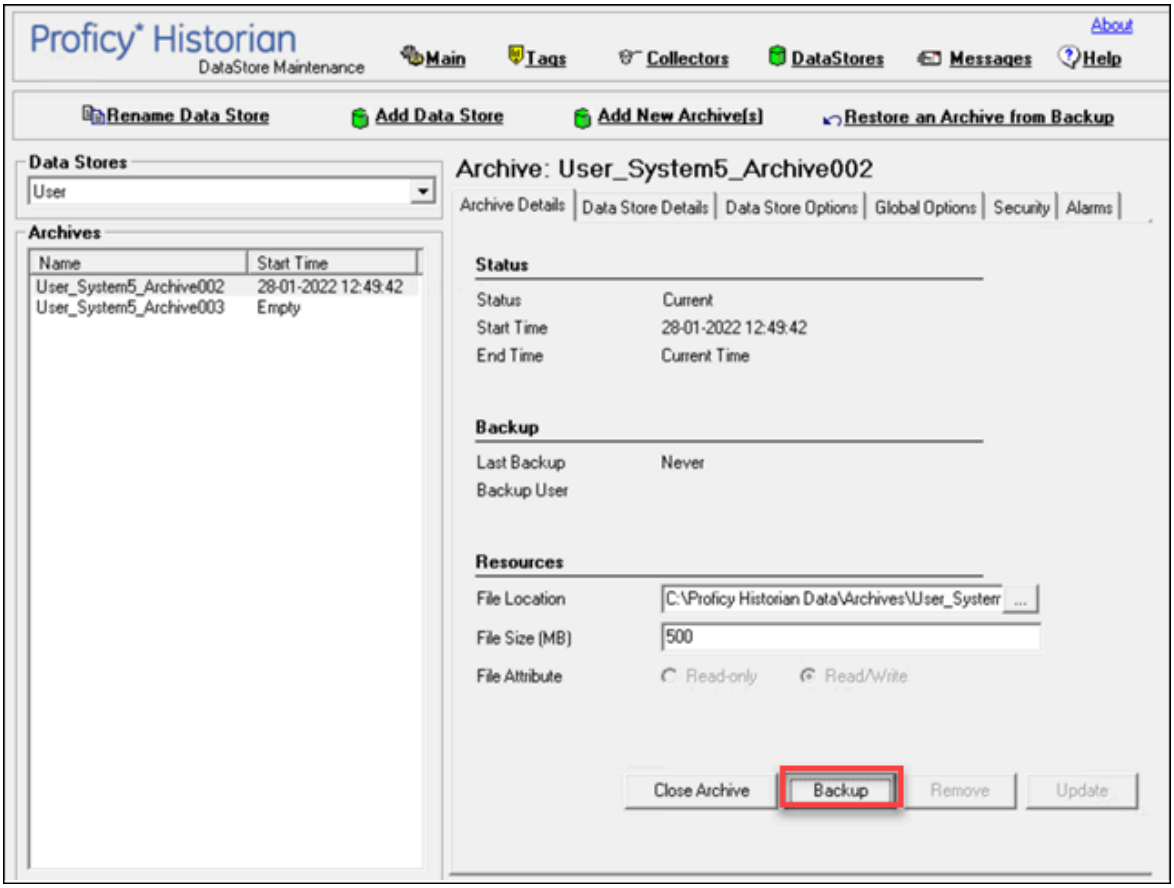



3. Select the archive that you want to back up.




4. Select **Archive Details**, and then select **Backup**.


Proficy Historian
DataStore Maintenance
About
Main
Tags
Collectors
DataStores
Messages
Help
Rename Data Store
Add Data Store
Add New Archive(s)
Restore an Archive from Backup
Data Stores
User
Archives
Name
Start Time
User_System5_Archive002
28-01-2022 12:49:42
User_System5_Archive003
Empty
Archive: User_System5_Archive002
Archive Details
Data Store Details
Data Store Options
Global Options
Security
Alarms
Status
Status
Current
Start Time
28-01-2022 12:49:42
End Time
Current Time
Backup
Last Backup
Never
Backup User
Resources
File Location
C:\Proficy Historian Data\Archives\User_System
File Size (MB)
500
File Attribute
Read-only
Read/Write
Close Archive
Backup
Remove
Update




The **Backing up Archive** window appears. The **Backup File** field is populated with the default folder path that you have specified while creating the data store and a default name.

5. If needed, change the folder path and name of the backup file.
   The archive is backed up. If needed, you can close and remove it from the list of archives (but it will still remain in the disk).

## Restore an Archive

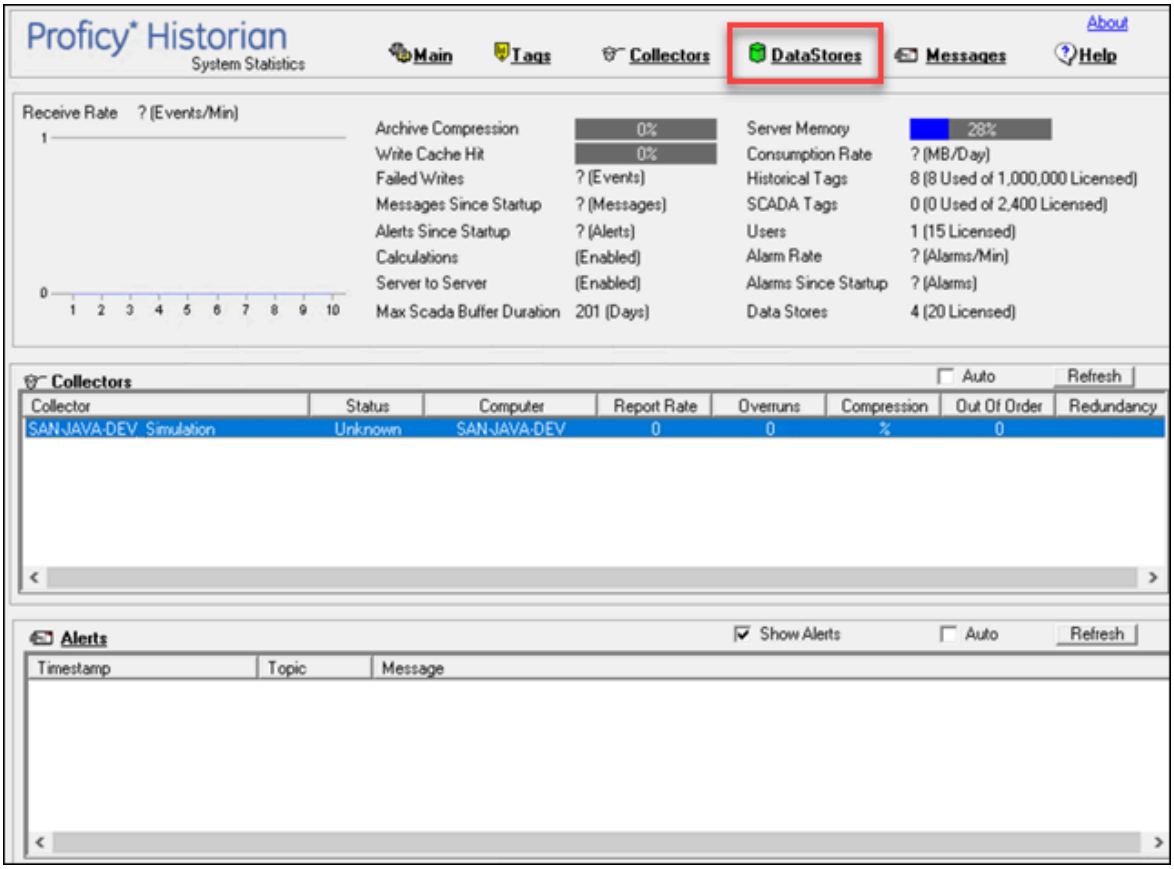
- Before restoring an archive from a removable disk, copy the archive file to the normal archive path and then restore the archive from that location. Leave the original backup file in the backup file folder.
- Copy the backup archive file to the default archive path.

Under certain circumstances, you may want to restore tag data to Historian. This may be after an unplanned shutdown, or you may need to retrieve data from an old, inactive archive.

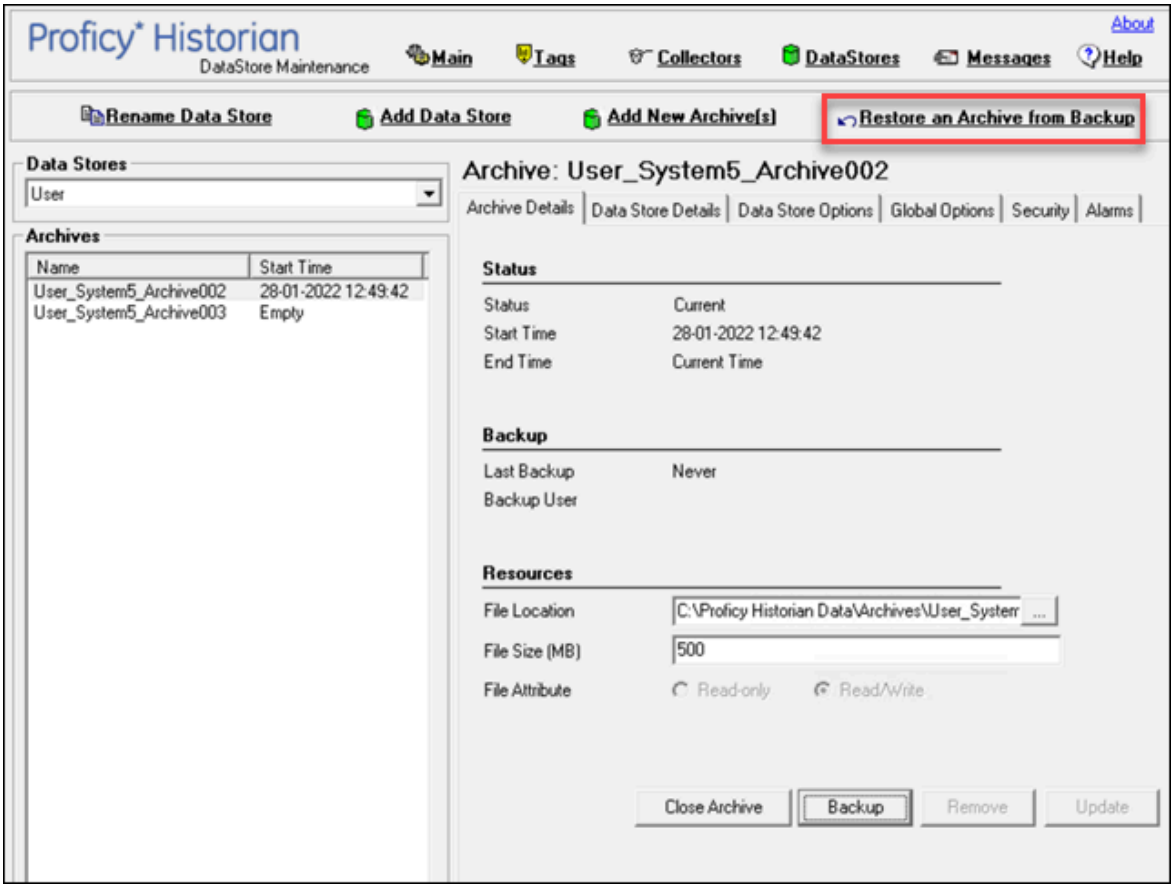
**CAUTION:**

- Restoring an archive is a resource-intensive operation and should be scheduled for non-peak usage times.
- Never restore an archive to a production Historian server without a current archive already online.

1. Access Historian Administrator *(on page 823)*.
2. Select **DataStores**.




3. Select **Restore An Archive From Backup**.




The **Restore Archive** window appears.

4. Enter values as described in the following table.

| Field | Description |
|---|---|
| **Archive Name** | Enter the name of the archive you want to restore. |
| **File Location** | Specify the location of the archive file. |
| **Data Store** | Select the data store to which you want to restore the archive. If the data store is not available, it is restored to the default data store. |

5. Select **OK**.

   The restored archive is moved to the `\Archive` directory and is made available for querying.

## Configure System File Cache Memory

You can specify the maximum disk cache memory that an archiver can use. By default, Historian consumes 25% of system memory. If your computer has extra memory, you can increase the disk cache.

Increasing the disk cache memory optimizes the Historian performance. You should not increase system file caching cache memory if you do not have the necessary system resources.

1. Access Registry Editor.
2. Open the following key folder.`HKEY_LOCAL_MACHINE\SOFTWARE\Intellution, Inc.\iHistorian\Services\DataArchiver\`
3. Create a new DWORD called `FileSystemMaxCacheMB`
4. Set the decimal value to a number. The value that you enter represents the maximum disk cache memory that an archiver can use.
5. Select **OK**, and then close the Registry Editor.
6. Restart the Data Archiver service.

## Configure the Data Archiver Account for Remote Storage

By default, the Data Archiver service is installed under the LocalSystem account, which has no credentials to access network resources. In order to use Remote Storage, you must first configure the Data Archiver service to run under a user account that has read/write access to the network location.

1. Stop the Data Archiver service.
2. Open the Services Control Panel.
3. Right-click the **Historian Data Archiver** service, and select **Properties**
4. Select **Log On**.
5. In the **Log on as** field, select **This Account** .
6. Enter the user account using which you want to run the Data Archiver service. This account must have write privileges to your Storage Area Network.
7. Enter the password for the user account.
8. Restart the Data Archiver service.

## Reuse an Archive Configuration File

You may want to reuse an existing archive configuration file (*.ihc) when a Historian installation uses the same archive configuration as another Historian installation or when renaming a machine and reusing the original .ihc file.

1. Retrieve a copy of the `<oldmachinename>_CentralConfig.ihc` file from the old machine. By default, this is located at `C:\Proficy Historian Data\Archives`
2. On the new machine, select and stop the following services:
    - Historian Client Manager
    - Historian Configuration Manager

    - Data Archiver
    - Historian Diagnostics Manager
3. Copy the `<oldmachinename>_CentralConfig.ihc` file into the local folder on the new machine where the .ihc files are available.
4. Look for a file in that folder on the new machine named `<newmachinename>_CentralConfig.ihc`. If there is a file with that name, delete it.
5. Rename the copied .ihc file on the new machine to `<newmachinename>_CentralConfig.ihc`.
6. Run a command prompt as an administrator.
7. Go to `C:\Program Files\Proficy\Proficy Historian\x64\Server`, and run the following command: `ihConfigManager_x64.exe RenameDHSNode <old machine name> <new machine name>`

    **Note:**
    RenameDHSNode modifies only the local node name. The logical node name (service name) does not change.

RenameDHSNode will not alter service name configmanager_<nodename>

### Reusing a Configuration File

Suppose you have a production Historian with a machine name "WaterSite" and want to load that configuration on a testbed Historian called "LabSite1". The testbed Historian is already running and already has a config file called "LabSite1_CentralConfig.ihc."

In this case:

1. Stop the services and delete `LabSite1_CentralConfig.ihc`.
2. Rename `WaterSite_CentralConfig.ihc` to `LabSite1_CentralConfig.ihc`.
3. At the command line, run the following:

```
ihConfigManager_x64.exe RenameDHSNode WaterSite LabSite1
```

# Managing Tags

## About Tags

A Historian tag is used to store data related to a property.

For example, if you want to store the pressure, temperature, and other operating conditions of a boiler, a tag will be created for each one in Historian.

When you collect data using a collector, tags are created automatically in Historian to store these values. These tags are mapped with the corresponding properties in the source.

For example, suppose you want to store OSI PI data in Historian. You will specify the OSI PI tags for which you want to collect data. The OSI PI collector creates the corresponding tags in Historian, and it stores the values in those tags.

You can also choose to create tags manually (for example, to store the result of a calculation performed by the Calculation collector).

## About Array Tags

You can store a set of values with a single timestamp and single quality and then read the elements individually or as an array.

The following conditions apply when using an array tag:

- You need not specify the size of an array tag. Data Archiver will store the number of elements that were written.
- You can change a tag to an array tag later as well. However, when you do so, only the latest data is retrieved. If you want to get the old data, you must change the tag back to its previous type.
- The maximum number of elements that an array tag can store is 10,000.
- You cannot associate an enumerated set or a user-defined data type (UDT) with an array tag.
- Fixed String and Scaled data types are not supported.
- Scaling, collector compression, and archive compression do not apply to an array tag.
- You cannot use an array element as a calculation trigger.
- You cannot plot a trend chart for an array tag.
- TagStats calculation mode is not supported.

## About Collector and Archive Compression

### Collector Compression

Collector compression applies a smoothing filter to data retrieved from the data source. By ignoring small changes in values that fall within a deadband centered around the last reported value, only significant changes are reported to the archiver. Fewer samples reported yields less work for the archiver and less archive storage space used.

You can specify the deadband value. For convenience, if you enter a deadband percentage, Historian Administrator shows the deadband in engineering units. For example, if you specify a 20% deadband on 0

to 500 EGU span, it is calculated and shown as 100 engineering units. If you later change the limits to 100 and 200, the 20% deadband is now calculated as 20 engineering units.

The deadband is centered around the last reported sample, not simply added to it or subtracted. If your intent is to have a deadband of 1 unit between reported samples, you must enter a compression deadband of 2 so that it is one to each side of the last reported sample. In the previous example of 0 to 500 EGU range, with a deadband of 20%, the deadband is 100 units; This means that only if the value changes by more than 50 units, it is reported.

Changes in data quality from good to bad, or bad to good, automatically exceed collector compression and are reported to the archiver. Any data that comes to the collector out of time order will also automatically exceed collector compression.

It is possible for collected tags with no compression to appear in Historian as if the collector or archive compression options are enabled. If collector compression occurs, you will notice an increase in the percentage of the compression value in the **Collectors** section of the **System Statistics** page in Historian Administrator. When archive compression occurs, you will notice the archive compression value and status bar change on the **System Statistics** page.

For instructions on setting collector compression, refer to Access/Modify a Tag *(on page 894)*.

Even if collector compression is not enabled, you may notice it in the following scenarios:

- When a succession of bad data quality samples appears, Historian collects only the first sample in the series. No new samples are collected until the data quality changes. Historian does not collect the redundant bad data quality samples, and this is reflected in the collector compression percentage.
- For a Calculation or Server-to-Server collector, when calculations fail, producing no results or bad quality data, collector compression is used. The effect of Collector Compression Timeout is to behave, for one poll cycle, as if the collector compression feature is not being used. The sample collected from the data source is sent to the archiver. Then the compression is turned back on, as configured, for the next poll cycle with new samples being compared to the value sent to the archiver.

**Note:**
Array tags do not support archive and collector compression. If the tag is an array tag, then the **Compression** tab is disabled.

### Handling Value Step Changes with Collector Data Compression

If you enable collector compression, the collector does not send values to the archiver any new input values if the value remains within its compression deadband. Occasionally, after several sample intervals inside the deadband, an input makes a rapid step change in value during a single sample interval. Since there have been no new data points recorded for several intervals, an additional sample is stored one interval before the step change with the last reported value to prevent this step change from being viewed as a slow ramp in value. This value marks the end of the steady-state, non-changing value period, and provides a data point from which to begin the step change in value.

**Note:**
You can configure individual tags can be configured to retrieve step value changes.

The collector uses an algorithm that views the size of the step change and the number of intervals since the last reported value to determine if a marker value is needed. The following is an example of the algorithm:

```
BigDiff=abs(HI_EGU-LO_EGU)*(CompressionDeadbandPercent/(100.0*2.0))*4.0

If ( Collector Compression is Enabled )

If ( Elapsed time since LastReportedValue>=( SampleInterval * 5 ) )

If ( abs(CurrentValue-LastReportedValue) > BigDiff )

Write LastReportedValue,Timestamp=(CurrentTime-SampleInterval)
```

In the example above, if a new value was not reported for at least the last 4 sample intervals, and the new input value is at least 4 deltas away from the old value (where a single delta is equal to half of the compression deadband), then a marker value is written.

**Note:**
These settings are also adjustable from the Registry. Please contact technical support for more information.

### Value Spike with Collector Compression

For example, a collector reads a value X once per second, with a compression deadband of 1.0. If the value of X is 10.0 for a number of seconds starting at 0:00:00 and jumps to 20.0 at 0:00:10, the data samples read would be:

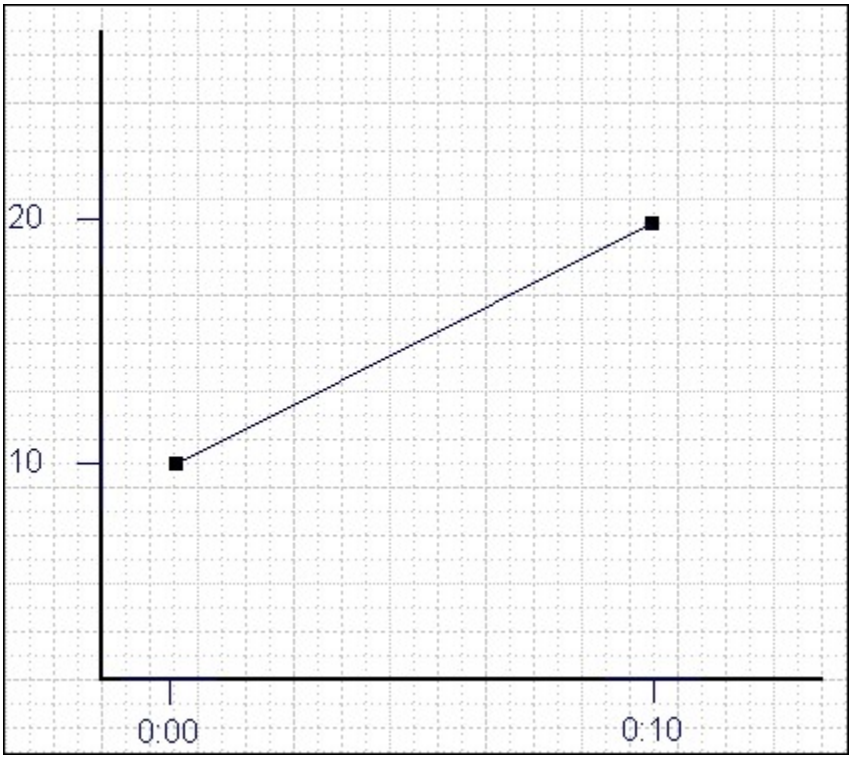
| Time | X Value |
|---|---|
| 0:00:00 | 10.0 (steady state value) |

| Time | X Value |
|---|---|
| 0:00:01 | 10.0 |
| 0:00:02 | 10.0 |
| 0:00:03 | 10.0 |
| 0:00:04 | 10.0 |
| 0:00:05 | 10.0 |
| 0:00:06 | 10.0 |
| 0:00:07 | 10.0 |
| 0:00:08 | 10.0 |
| 0:00:09 | 10.0 |
| 0:00:10 | 20.0 (new value after step change) |

To increase efficiency, the straightforward compression would store only 2 of these 11 samples.

| Time | X Value |
|---|---|
| 0:00:00 | 10.0 (steady state value) |
| 0:00:10 | 20.0 (new value after step change) |

However, without the marker value, if this data were to be put into a chart, it would look like the data value **ramped** over 10 seconds from a value of 10.0 to 20.0, as shown in the following chart.




The addition of a **marker value** to the data being stored results in the following data values:

| Time | X Value |
|---|---|
| 0:00:00 | 10.0 (steady state value) |
| 0:00:09 | 10.0 (inserted Marker value) |
| 0:00:10 | 20.0 (new value after step change) |

If you chart this data, the resulting trend accurately reflects the raw data and likely real world values during the time period as shown in the following chart.

## Evaluating and Controlling Data Compression

You can achieve optimum performance in Historian by carefully controlling the volume of dynamic data it collects and archives. You need enough information to tell you how the process is running, but you do not need to collect and store redundant or non-varying data values that provide no useful information.

**Control Data Flow**

You can control the amount of online or dynamic data the system handles at a given time by adjusting certain system parameters. The general principle is to control the flow of data into the archive either by adjusting the rate at which the collectors gather data or by adjusting the degree of filtering (compression) the system applies to the data collected.

Adjust the following parameters to *reduce* the rate of data flow into the server.

- Reduce the polling rate by increasing the collection interval for unsolicited and polled collection.
- Enable collector compression and optionally use compression timeout.
- Set the compression deadband on the collectors to a wider value.
- Use the collector compression timeout.

Adjust the following parameters to *increase the filtering* applied by the archiver in the server.

- Enable archive (trend) compression.
- Set the archive compression deadband to a wider value.
- Where possible, use the scaled data type and enable input scaling on selected tags.
- Where possible, select milliseconds or microseconds rather than seconds for time resolution. Seconds is optimum for most common devices. This affects disk space.

**Evaluate Data Compression Performance**

You can determine how effectively data compression is functioning at any given time by examining the system statistics displayed on the **System Statistics** page of Historian Administrator.

The compression field at the top of the page shows the current effect of archive compression. Values for this parameter should typically range from 0 to 9%. If the value is zero, it indicates that compression is either ineffective or turned off. If it shows a value other than zero, it indicates that archive compression is operating and effective. The value itself indicates how well it is functioning. To increase the effect of data compression, increase the value of archive compression deadband so that compression becomes more active.

## Archive Compression

Archive compression is used to reduce the number of samples stored when data values for a tag form a straight line in any direction. For a horizontal line (non-changing value), the behavior is similar to collector compression. But, in archive compression, it is not the *values* that are being compared to a deadband, but the *slope of line* those values produce when plotted value against time. Archive compression logic is executed in the data archiver and, therefore, can be applied to tags populated by methods other than collectors.

You can use archive compression on tags where data is being added to a tag by migration, or by the File collector, or by an SDK program for instance. Each time the archiver receives a new value for a tag, the archiver computes a line between this incoming data point and the last archived value.

The deadband is calculated as a tolerance centered about the slope of this line. The slope is tested to see if it falls within the deadband tolerance calculated for the previous point. If the new point does not exceed the tolerance, it is not stored in the archive. This process repeats with subsequent points. When an incoming value exceeds the tolerance, the value held by the archiver is written to disk and the incoming sample is withheld.

The effect of the archive compression timeout is that the incoming sample is automatically considered to have exceeded compression. The withheld sample is archived to disk and the incoming sample becomes the new withheld sample. If the Archive Compression value on the System Statistics page indicates that

archive compression is occurring, and you did not enable archive compression for the tags, the reason could be because of internal statistics tags with archive compression enabled.

For instructions on setting archive compression, refer to Access/Modify a Tag *(on page 894)*.

## About Scaling

Scaling converts a data value from a raw value expressed in an arbitrary range of units, such as a number of counts, to one in engineering units, such as gallons per minute or pounds per square inch. The scaled data type can serve as a third form of data compression, in addition to collector compression and archive compression, if it converts a data value from a data type that uses a large number of bytes to one that uses fewer bytes.

## About Condition-Based Collection

Condition based collection is a method to control the storage of data for data tags by assigning a condition. Data is always collected but it is only written to the Data Archiver if the condition is true; otherwise, the collected data is discarded.

This condition is driven by a trigger tag; a tag collected by the collector evaluating the condition. Ideally, Condition based Collection should be used only with tags that are updating faster than the trigger tag. Condition based collection can be used to archive only the specific data which is required for analysis, rather than archiving data at all times, as the collector is running.

For example, if a collector has tags for multiple pieces of equipment, you can stop collection of tags for one piece of equipment during its maintenance. It is typically used on tags that use fast polled collection but you don't want to use collector compression. While the equipment is running, you want all the data but when the equipment is stopped, you don't want any data stored. The trigger tag would also typically use polled collection. But, either tag could use unsolicited collection.

The condition is evaluated every time data is collected for the data tag. When a data sample is collected, the condition is evaluated and data is either queued for sending to archiver, or discarded. If the condition cannot be evaluated as true or false, like if the trigger tag contains a bad data quality or the collector is not collecting the trigger tag, the condition is considered true and the data is queued for sending.

No specific processing occurs when the condition becomes true or false. If the condition becomes true, no sample is stored to the data tag using that condition, but the data tag will store a sample next time it collects. When the condition becomes false, no end of the collection marker is stored until the data tag is collected.

For example, if the condition becomes false at 1:15 and the data tag gets collected at 1:20, the end of collection marker will be created at 1:20 and have a timestamp of 1:20, not 1:15.

Condition based collection is supported by only archiver and collectors of Historian version 4.5 and above. Condition based collection does not apply to alarm collectors. This condition based collection is applicable to the following collectors only:

- Simulation Collector
- OPC Collector
- iFIX Collector
- PI Collector

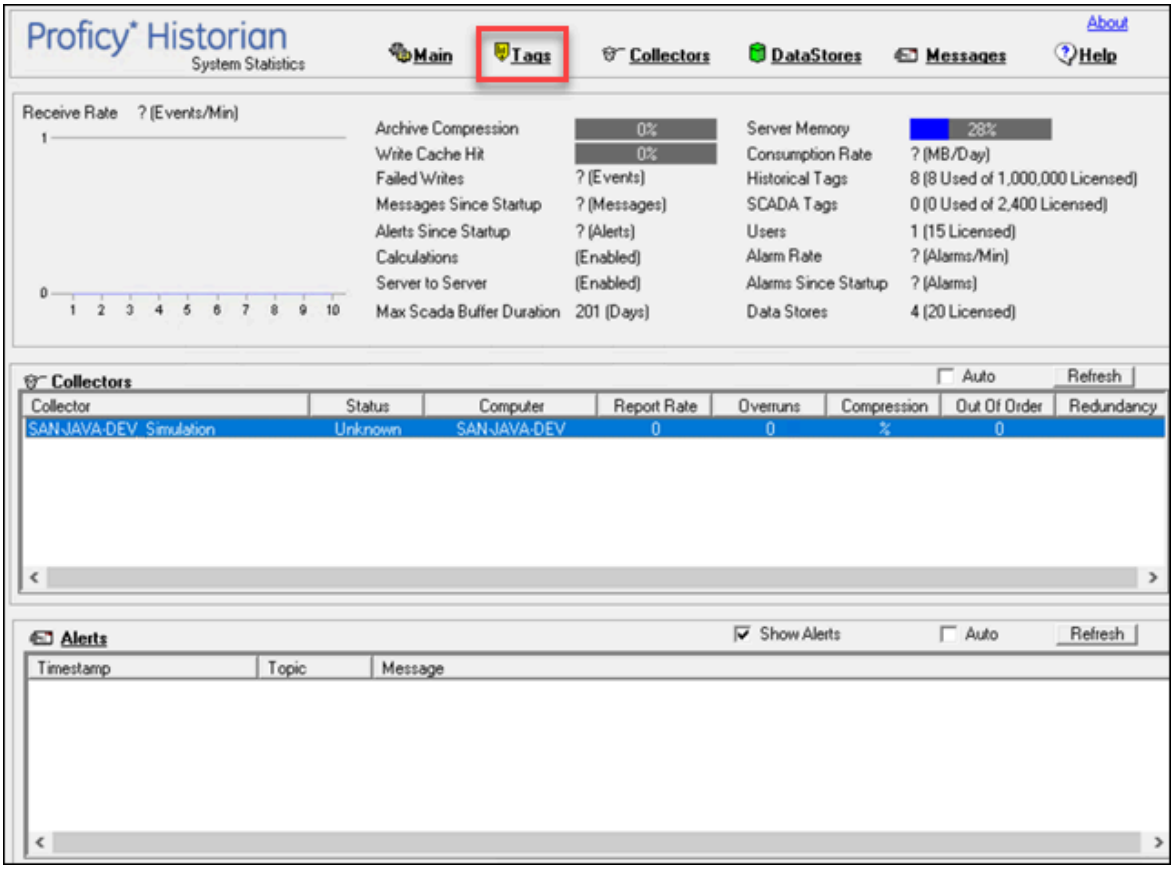
For instructions on setting the condition-based collection, refer to Access/Modify a Tag *(on page 894)*.

## Access/Modify a Tag

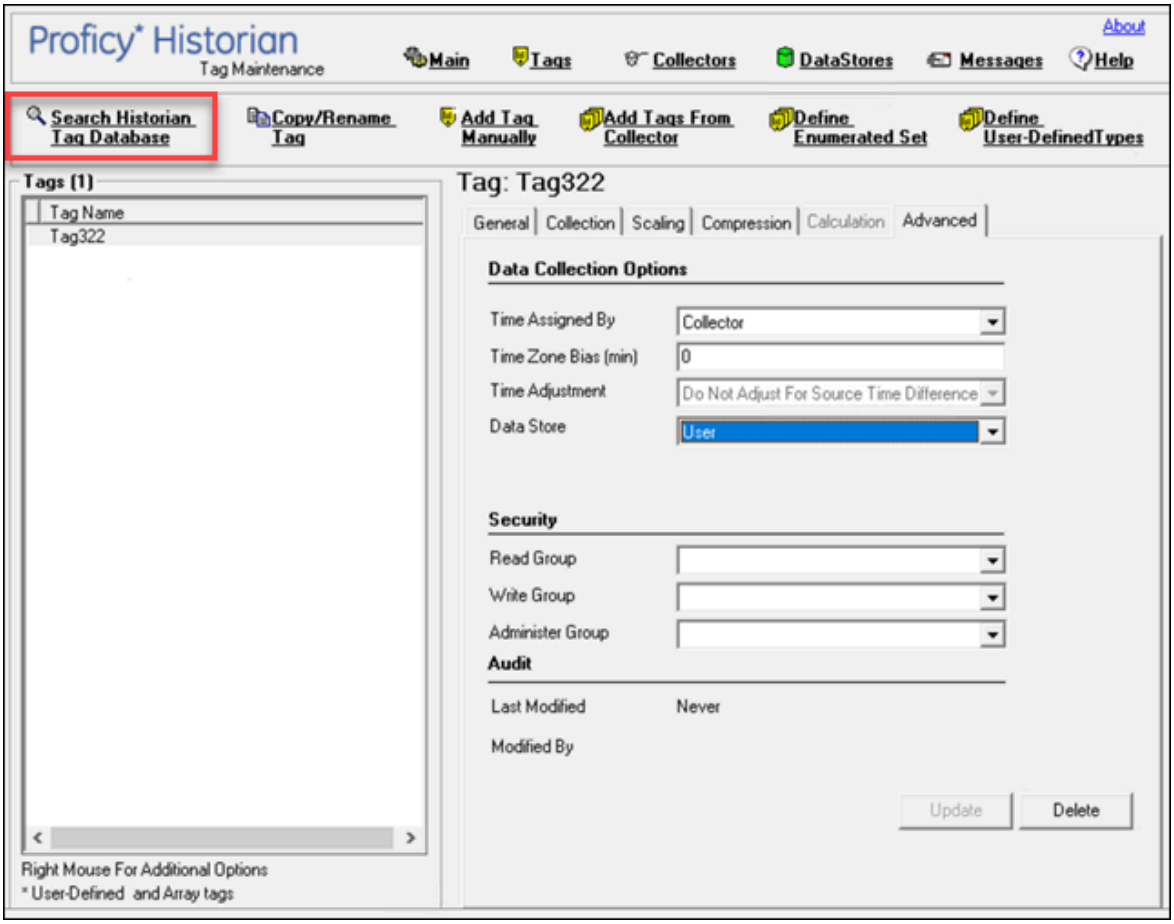
Using Historian Administrator, you can access a list of tags in the Historian database by their name, description, or both.

**Note:**
By default, maximum one million tags are retrieved. If the Historian clients are configured to retrieve more than a million tags, to retrieve all of them, add the `MaxTagsToRetrieve` registry key under `HKEY_LOCAL_MACHINE\SOFTWARE\Intellution, Inc.\iHistorian\Services\DataArchiver\`, and then set the maximum number of tags that you want to retrieve. Restart the Historian Data Archiver service for the change to reflect.

1. Access Historian Administrator *(on page 823)*.
2. Select **Tags**.




3. Select **Search Historian Tag Database**.

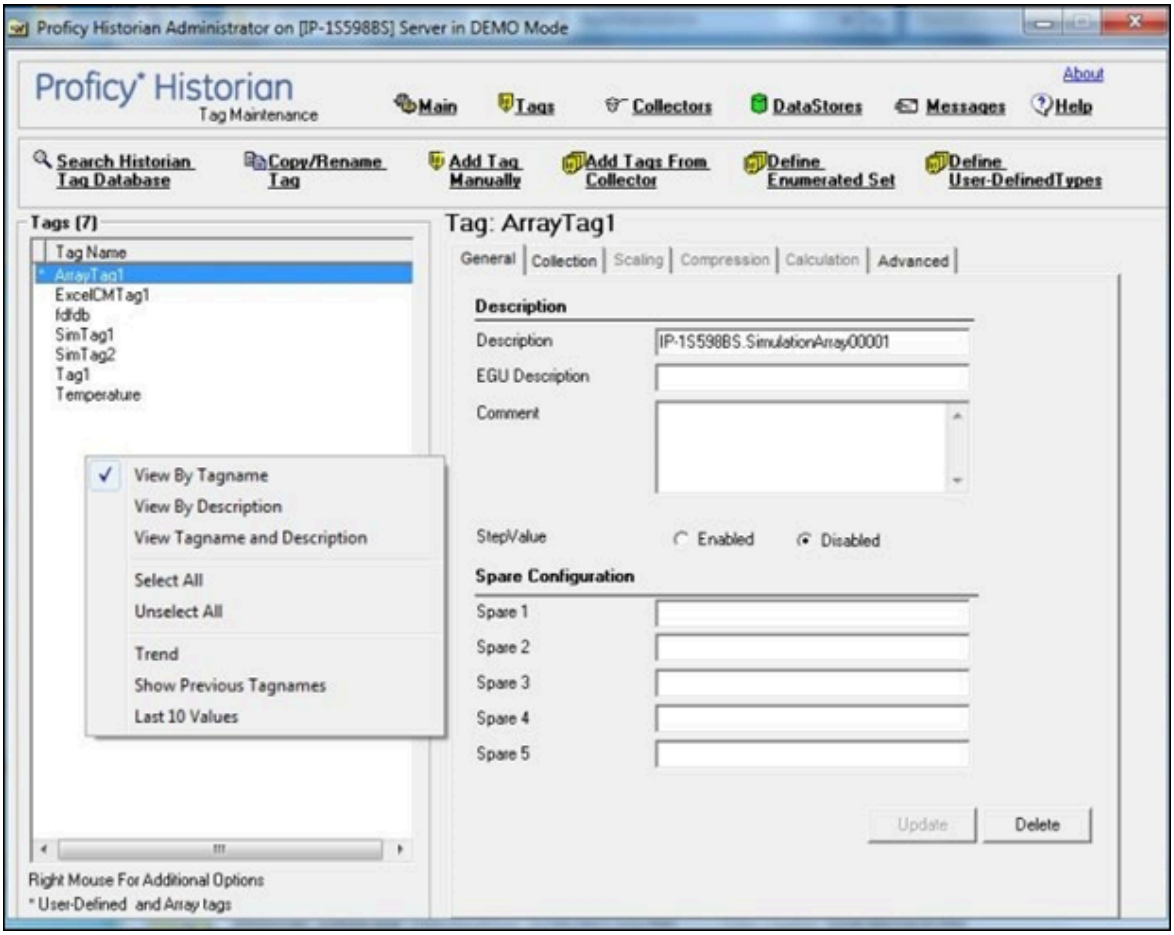



The **Search Historian Tag Database** window appears.

4. Enter values in the available fields to search for the tag, and then select **OK**. You can use the wildcard character asterisk (*).

   A list of tags that meet the search criteria appear in the **Tags** section.

5. Right-click the **Tags** section, and then select one of the following values:
   - **View By TagName**: Select this option to view only the names of the tags.
   - **View By Description**: Select this option to view only the descriptions of the tags.
   - **View Tagname and Description**: Select this option to view both the names and descriptions of the tags.




6. As needed, modify values as described in the following tables, and then select **Update**.

**Table 63. The General Section**

| Field | Description |
|---|---|
| **Description** | The description of the tag. |
| **EGU Description** | The engineering units assigned to the tag. |
| **Comment** | Comments that apply to the tag. |
| **StepValue** | Indicates that the actual measured value changes in a sharp step instead of a smooth linear interpolation. This option is applicable only for numeric data. Enabling this option only affects data retrieval; it has no effect on data collection or storage. |
| **Spare Configuration** | The Spare 1 through Spare 5 fields list any configuration information stored in these fields. |

| Field | Description |
|---|---|
| | **Note:**<br>Do not add or update the following spare configurations as data may get corrupted or overwritten:<br>◦ The **Spare 1** field for OSI PI Distributor reads data from the Historian tag displayed in the **Tag Source Address** field and sends it to the OSI PI tag name displayed in the **Spare 1** field.<br>◦ The **Spare 5** field is used for a Server-to-Server collector and a Server-to-Server distributor for internal purposes. |

**Table 64. The Collection Section**

| Field | Description |
|---|---|
| **Collector** | The name of the collector that collects data for the selected tag. |
| **Source Address** | The address for the tag in the data source. Leave this field blank for tags associated with the Calculation or Server-to-Server collector.<br><br>For Python Expression tags, this field contains the full applicable JSON configuration, which includes an indication of the source address.<br><br>**Note:**<br>When exporting or importing tags using the EXCEL Add-In, the Calculation column, not the SourceAddress column, holds the formulas for tags associated with the Calculation or Server-to-Server collector. |
| **Data Type** | The data type of the tag.<br><br>The main use of the scaled data type is to save space, but this results in a loss of precision. Instead of using 4 bytes of data, it only uses 2 bytes by storing the data as a percentage of the EGU limit. Changing the EGU limits will result in a change in the values that are displayed. For example, if the original EGU values were 0 to 100 and a value of 20 was stored using the scaled data type and if the EGUs are changed to 0 to 200, the original value of 20 will be represented as 40. |

| Field | Description |
|---|---|
| | **Note:**<br>If you change the data type of an existing tag between a numeric and a string or binary data type (and vice versa), the tag's compression and scaling settings will be lost. |
| **Enumerated Set Name** | The name of the enumerated set that you want to assign to the tag. |
| **Data Length** | The number of bytes for a fixed string data type. This field is enabled only for fixed string data types. |
| **Is Array Tag** | Indicates that the tag is an array tag. |
| **Collection** | Indicates whether data collection is enabled or disabled for the tag. If you disable collection for the tag, Historian stops collecting data for the tag, but does not delete the tag or its data. |
| **Collection Type** | The type of data collection used for this tag:<br>◦ **Polled**: The data collector requests data from the data source at the collection interval specified in the polling schedule.<br>◦ **Unsolicited**: The data source sends data to the collector whenever necessary (independent of the data collector polling schedule). |
| **Collection Interval** | The time interval between readings of data from this tag. For unsolicited collection type, this field defines the minimum interval at which unsolicited data should be sent by the data source. |
| **Collection Offset** | Used with the collection interval to schedule collection of data from a tag. For example, to collect a value for a tag every hour at thirty minutes past the hour (12:30, 1:30, 2:30, and so on), enter a collection interval of 1 hour and an offset of 30 minutes. Similarly, to collect a value each day at 8am, enter a collection interval of 1 day and an offset of 8 hours.<br><br>**Note:**<br>If you enter a value in milliseconds, the value must be in intervals of 1000 ms. For example, 1000, 2000, and 3000 ms are valid values, but 500 and 1500 ms are invalid. The minimum value is 1000 ms. |

| Field | Description |
|---|---|
| **Time Resolution** | The precision for timestamps, which can be either seconds, milliseconds or microseconds. |
| **Condition-Based** | Indicates whether condition-based data collection *(on page 666)* is enabled. |
| **Trigger Tag** | The name of the trigger tag used in the condition. |
| **Comparison** | The comparison operator that you want to use in the condition. This field is enabled only if you have enabled condition-based collection. Select one of the following options:<br>◦ **Undefined**: Collection will resume only when the value of the triggered tag changes. This is considered an incomplete configuration, so condition-based collection is turned off and all the collected data is sent to archiver.<br>◦ **< =**: Setting condition as trigger tag value less than or equal to the compare value.<br>◦ **> =** Setting condition as trigger tag value greater than or equal to the compare value.<br>◦ **<**: Setting condition as trigger tag value less than the compare value.<br>◦ **>**: Setting condition as trigger tag value greater than the compare value.<br>◦ **=**: Setting condition as trigger tag value equals compare value.<br>◦ **!=**: Setting condition as trigger tag value not the same as compare value. |
| **Compare Value** | A target value that you want to compare with the value of the trigger tag. If using = and != comparison parameters, ensure that the format of the compared value and triggered tag are the same. For example, for a float type trigger tag, the compare value must be a float value; otherwise, the condition result is an invalid configuration. When the configuration is invalid, condition-based collection is disabled and all data is sent to archiver. |
| **End of Collection Markers** | Indicates whether end-of-collection markers are enabled. This will mark all the tag's values as bad, and sub-quality as ConditionCollectionHalted when the condition becomes false. Trending and reporting applications can use this information to indicate that the real-world value was un- |

| Field | Description |
|---|---|
| | known after this time until the condition becomes true and a new sample is collected. If disabled, a bad data marker is not inserted when the condition becomes false. |

**Table 65. The Scaling Section**

| Field | Description |
|---|---|
| **Hi Engineering Units** | The current value of the upper range limit of the span for this tag.<br><br>Engineering Hi and Lo are retrieved automatically for F_CV fields for iFIX tags; all others are left at default settings. When adding tags from the server using an OPC Collector, the OPC Collector queries the server for the EGU units and EGU Hi/Lo limits. Not all OPC Servers make this information available, however. Therefore, if the server does not provide the limits when requested to do so, the collector automatically assigns an EGU range of 0 to 10,000. |
| **Lo Engineering Units** | The current value of the lower range limit of the span for this tag. |
| **Input Scaling** | Indicates whether input scaling is enabled, which converts an input data point to an engineering units value.<br><br>For example, to rescale and save a 0 - 4096 input value to a scaled range of 0 - 100, enter 0 and 4096 as the low and high input scale values and 0 and 100 as the low and high engineering units values, respectively.<br><br>If a data point exceeds the high or low end of the input scaling range, Historian logs a bad data quality point with a ScaledOutOfRange subquality. In the previous example, if your input data is less than 0, or greater than 4096, Historian records a bad data quality for the data point.<br><br>**OPC Servers and TRUE Values:** Some OPC servers return a TRUE value as -1. If your OPC server is returning TRUE values as |

| Field | Description |
|---|---|
| | -1, modify the following scaling settings in the **Tag Maintenance** page of Historian Administrator:<br>`Hi Engineering Units = 0`<br>`Lo Engineering Units = 1`<br>`Hi Scale Value = 0`<br>`Lo Scale Value = - 1`<br>`Input Scaling = Enabled` |
| **Hi Scale Value** | The upper limit of the span of the input value. |
| **Lo Scale Value** | The lower limit of the span of the input value. |

**Table 66. The Compression Section**

| Field | Description |
|---|---|
| **Collector Compression** | Indicates whether collector compression *(on page 660)* is enabled. |
| **Collector Deadband** | The current value of the compression deadband. This value can be computed as a percent of the span, centered around the data value or given as an absolute range around the data value.<br>**Note:**<br>Some OPC servers add and subtract the whole deadband value from the last data value. This effectively doubles the magnitude of the deadband compared to other OPC servers. To determine how your specific server handles deadband, refer to the documentation of your OPC server.<br>**Example:**<br>Suppose the engineering units are 0 to 200. Suppose the deadband value is 10%, which is 20 units. If the deadband value is 10% and the last reported value is 50, the value will be reported when the current value exceeds 50 + 10 = 60 or is less than 50 - 10 = 40. Note that the deadband (20 units) is split around the last data value (10 on either side.) |

| Field | Description |
|---|---|
| | Alternatively, you could specify an absolute deadband of 5. In this instance, if the last value was 50, a new data sample will be reported when the current value exceeds 55 or drops below 45.<br><br>If compression is enabled and the deadband is set to zero, the collector ignores data values that do not change and records any that do change. If you set the deadband to a non-zero value, the collector records any value that lies outside the deadband. If the value changes drastically, a pre-spike point may be inserted. For information, refer to Enable Spike Logic *(on page 970)*. |
| **Engineering Unit** | Converts the deadband percentage into engineering units and displays the result. This value establishes the deadband range that is centered around the new value.<br><br>This field represents a calculated number created to give an idea of how large a deadband you are creating in engineering units. The deadband is entered in percentage and Historian converts the percentage in to engineering units. |
| **Collector Compression Timeout** | Indicates the maximum amount of time the collector will wait between sending samples for a tag to the archiver. This time is maintained per tag, as different tags report to the archiver at different times.<br><br>For polled tags, this value should be in multiples of your collection interval. After the timeout value is exceeded, the tag stores a value at the next scheduled collection interval, and not when the timeout occurred. For example, if you have a 10-second collection interval, a 1-minute compression timeout, and a collection that started at 2:14:00, if the value has not changed, the value is logged at 2:15:10 and not at 2:15:00.<br><br>For unsolicited tags, a value is guaranteed in, at most, twice the compression timeout interval.<br><br>A non-changing value is logged on each compression timeout. For example, an unsolicited tag with a 1-second collection interval and a 30-second compression timeout is stored every 30 seconds. |

| Field | Description |
|---|---|
| | A changing value for the same tag may have up to 60 seconds between raw samples. In this case, if the value changes after 10 seconds, then that value is stored, but the value at 30 seconds (if unchanged) will not be stored. The value at 60 seconds will be stored. This leaves a gap of 50 seconds between raw samples which is less than 60 seconds.<br><br>Compression timeout is supported in all collectors except the PI collector. |
| **Archive Compression** | Indicates whether archive compression *(on page 660)* is enabled. If enabled, Historian applies the archive deadband settings against all reported data from the collector. |
| **Archive Deadband** | The current value of the archive deadband, expressed as a percent of span or an absolute number. |
| **Engineering Unit** | Converts the deadband percentage into engineering units and displays the result. This value establishes the deadband range that is centered around the new value. |
| **Archive Compression Timeout** | The maximum amount of time from the last stored point before another point is stored, if the value does not exceed the archive compression deadband.<br><br>The data archiver treats the incoming sample after the timeout occurs as if it exceeded compression. It then stores the pending sample. |

**Table 67. The Advanced Section**

| Field | Description |
|---|---|
| **Time Assigned By** | The source of the timestamp for a data value is either the collector or the data source.<br><br>All tags, by default, have their time assigned by the collector. When you configure a tag for a polled collection rate, the tag is updated based on the collection interval. For example, if you set the collection interval to 5 seconds with no compression, then the archive will be updated with a new data point and timestamp every 5 seconds, even if the value is not changing. |

| Field | Description |
|---|---|
| | However, if you set the **Time Assigned By** field to **Source** for the same tag, the archive only updates when the device timestamp changes. For example, if the poll time is still 5 seconds, but if the timestamp on the device does not change for 10 minutes, no new data will be added to the archive for 10 minutes.<br><br>**Note:**<br>This field is disabled for Calculation and Server-to-Server tags. |
| **Time Zone Bias** | The number of minutes from GMT that should be used to translate time-stamps when retrieving data from this tag. For example, the time zone bias for Eastern Standard time is -300 minutes (GMT-5).<br><br>This field is not used during collection. Use this option if a particular tag requires a time zone adjustment during retrieval other than the client or server time zone. For example, you could retrieve data for two tags with different time zones by using the tag time zone selection in the iFIX chart. |
| **Time Adjustment** | If the Server-to-Server collector is not running on the source computer, select the **Adjust for Source Time Difference** option to compensate for the time difference between the source archiver computer and the collector computer.<br><br>**Note:**<br>This field only applies to tags associated with the Server-to-Server collector that use a polled collection type. |
| **Data Store** | Displays the data store to which the tag belongs. |
| **Read Group** | The Windows security group assigned to the selected tag. |
| **Write Group** | The Windows security group assigned to the selected tag. |
| **Administer Group** | The Windows security group assigned to the selected tag. |

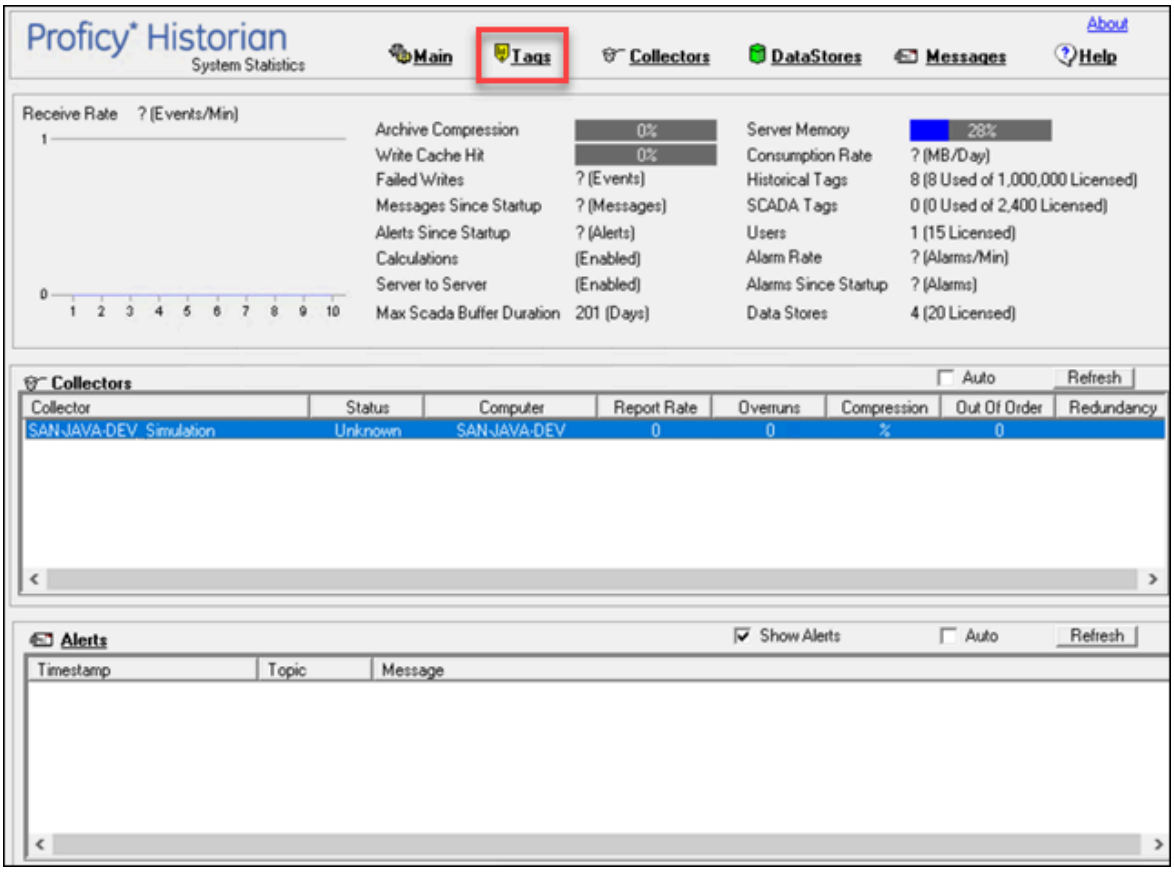
| Field | Description |
|---|---|
| **Last Modified** | The date the last tag parameter modification was made. |
| **Modified By** | The name of the person who last modified the tag configuration parameters. |

## Add Tags from Source

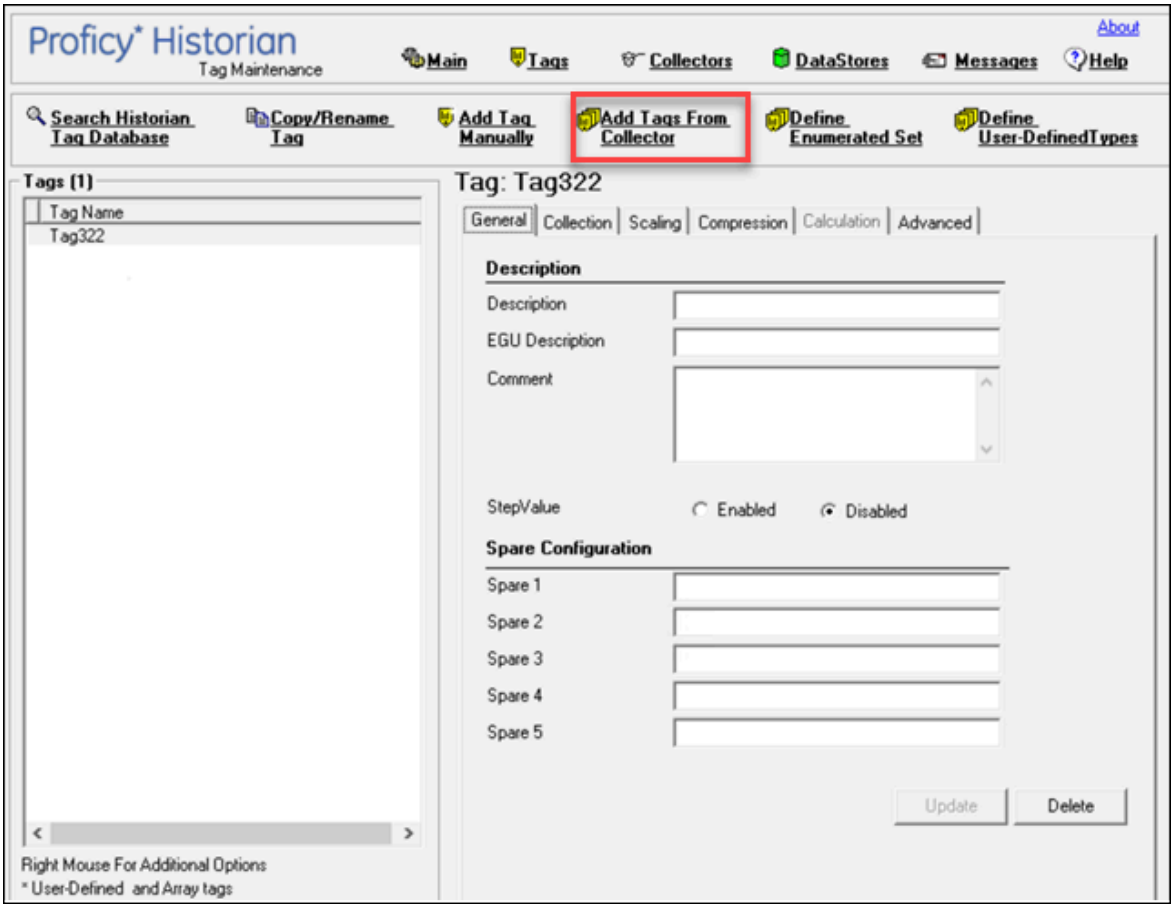
Create a collector instance *(on page 357)* using which you want to browse the source for tags.

This topic describes how to browse for source tags and add them to Historian. These tags are then created automatically in the Historian database. You can also create tags manually *(on page 909)*.

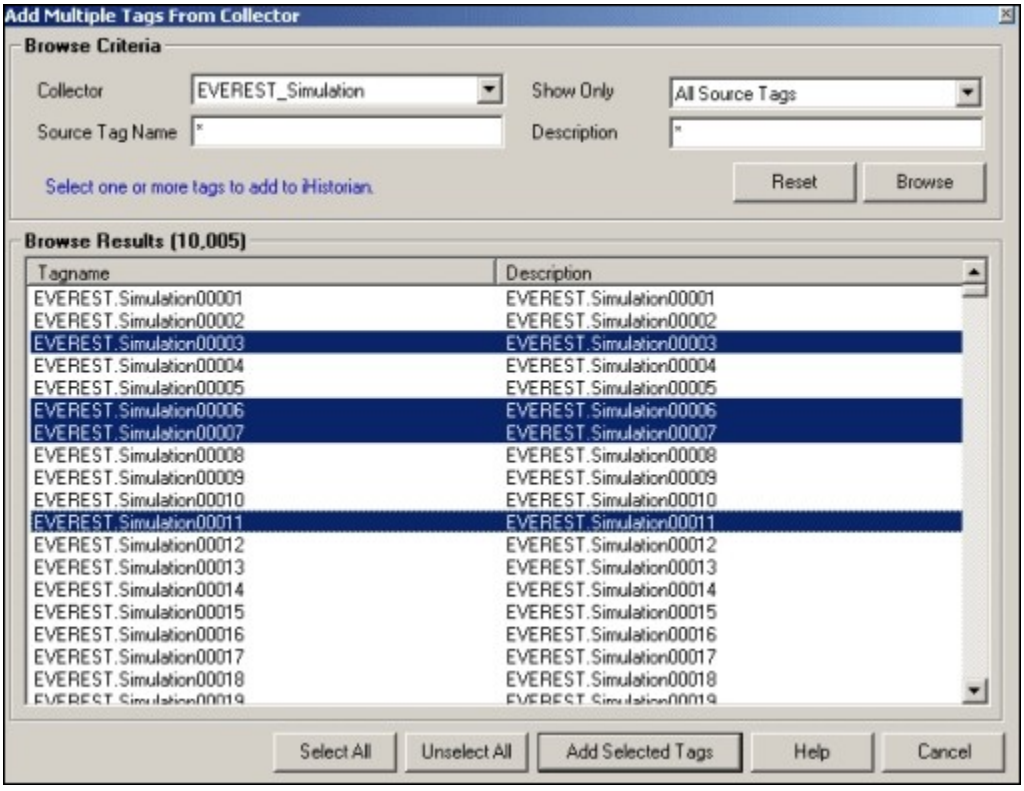
1. Access Historian Administrator *(on page 823)*.
2. Select **Tags**.




3. Select **Add Tags from Collector**.




The **Add Multiple Tags from Collector** window appears.




4. Enter values as described in the following table.

| Field | Description |
|---|---|
| **Collector** | Select the collector instance using which you want to browse the source for tags. |
| **Show Only** | Specify whether you want to see all tags or only the ones that have not been added yet. |
| **Source Tag Name** | Enter the string to narrow down the search results based on the tag name. You can use wildcard characters. |
| **Description** | Enter the string to narrow down the search results based on the tag description. You can use wildcard characters. |

5. Select **Browse**.

   A list of tags based on the search criteria appear.
6. Select the tags that you want to add to Historian.
   - Select a single tag by selecting the name of the tag.
   - Select multiple tags by pressing the **Control** key and selecting the tags.

    - Select a contiguous group by pressing the **Shift** key and selecting the first and last tag of the group.
    - Select all tags by selecting **Select All**.
7. Select **Add Selected Tags**.
   The selected tags are added to the Historian database.

## Create a Tag Manually

This topic describes how to create a tag manually. You can also add tags from source *(on page 906)*; these tags are then automatically created in the Historian database.

Whenever you add tags, delete tags, or modify certain tag properties, the following collectors reload only the modified tags without restarting the collectors.

- OPC Collector
- iFIX Collector
- Calculation Collector
- Simulation Collector
- Server to Server Collector
- PI Collector
- PI Distributor

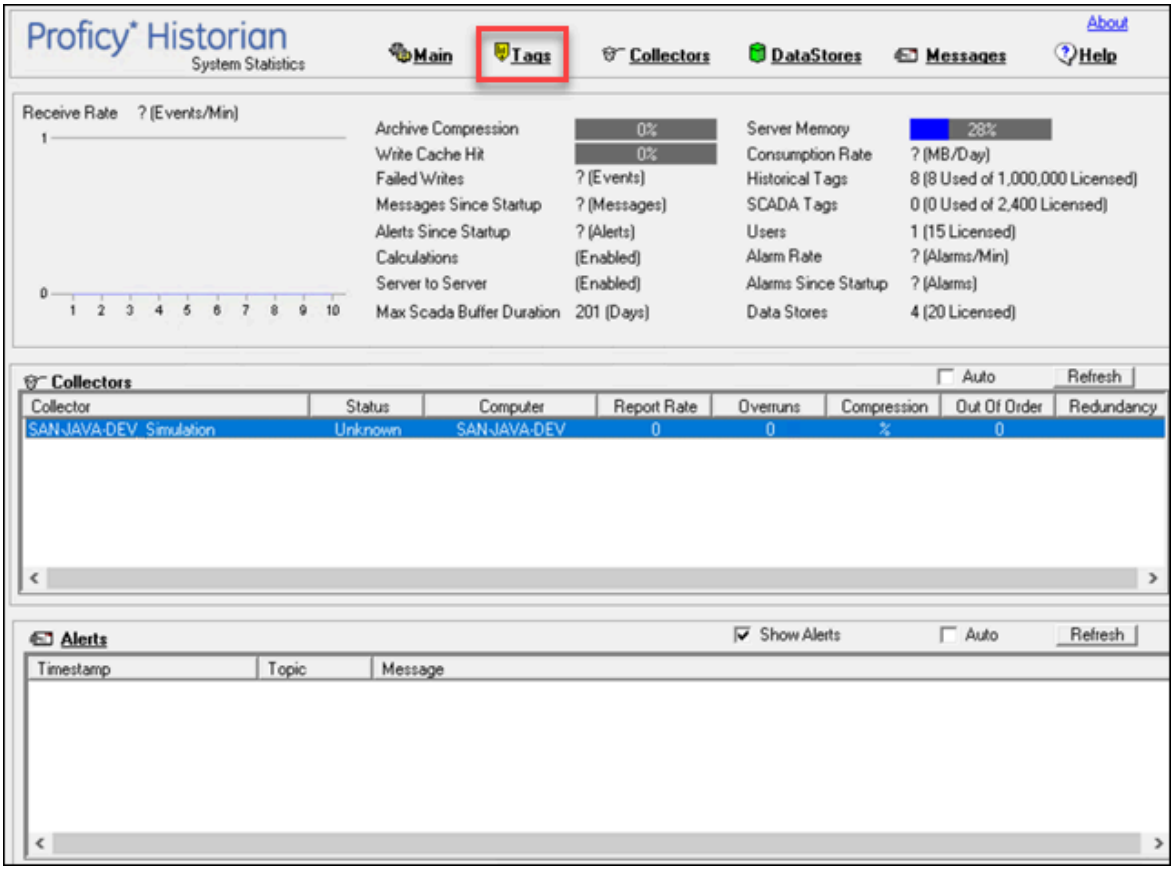
The dynamic collector update feature ensures that any modifications to the tag configuration do not affect all the tags in a collector. Tags that stop data collection may record zero data and bad quality without restarting the collector. Tags that do not stop data collection do not record bad data samples to the collection.

By default, the On-line Tag Configuration Changes option is enabled, which allows a tag to stop and restart data collection without restarting the collector. If you disable the On-line Tag Configuration Changes option, any changes you make to the tags do not affect collection until after you restart the collector. To enable or disable the On-line Tag Configuration Changes option, select **Advanced** on the **Collector Maintenance** page.

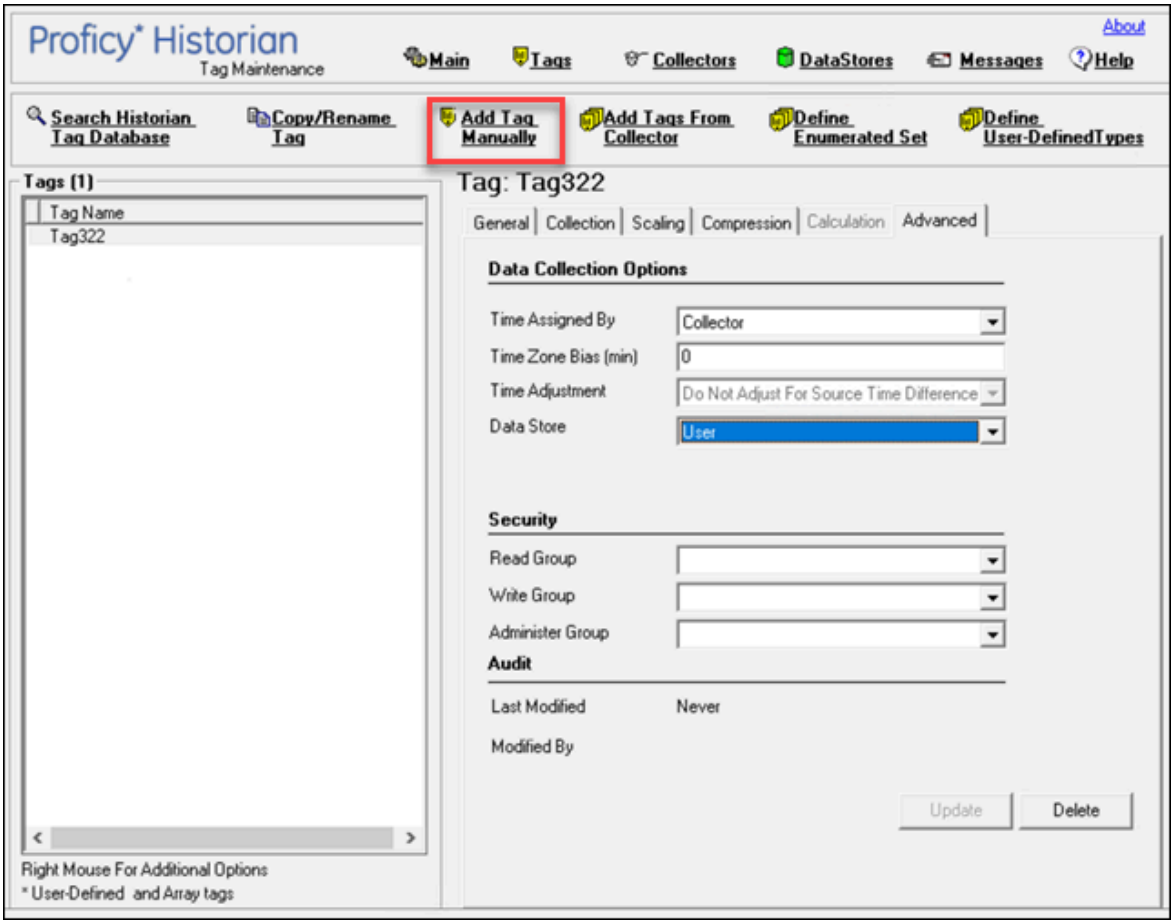
To restart the collector you must stop and start the collector service or executable. Restarting the collector stops and restarts the tag(s) collection and may record bad data samples to the collection. All the collector configuration changes done within a 30 second time frame are batched up together. To collect the modified data faster, update/modify a small set of tags at a time.

**Note:**
When updating large sets of tags at the same time, best practice is to disable the On-line Tag Configuration Changes option and restart the collector after modification.

1. Access Historian Administrator *(on page 823)*.
2. Select **Tags**.




3. Select **Add Tag Manually**.




The **Add Tag Manually** window appears.

4. Enter values as described in the following table.

| Field | Description |
|---|---|
| **Collector Name** | Select the collector using which you want to collect data for the tag. |
| **Source Address** | Enter the source address for the tag. |
| **Tag Name** | Enter a unique name for the tag. |
| **Data Store** | Select the data store in which you want to store the tag data. |
| **Data Type** | Select the data type of the tag data. |
| **Is Array Tag** | Select this check box if the tag is an array tag. For fixed string data types only, enter a value in the field adjacent to the Data Type field. |

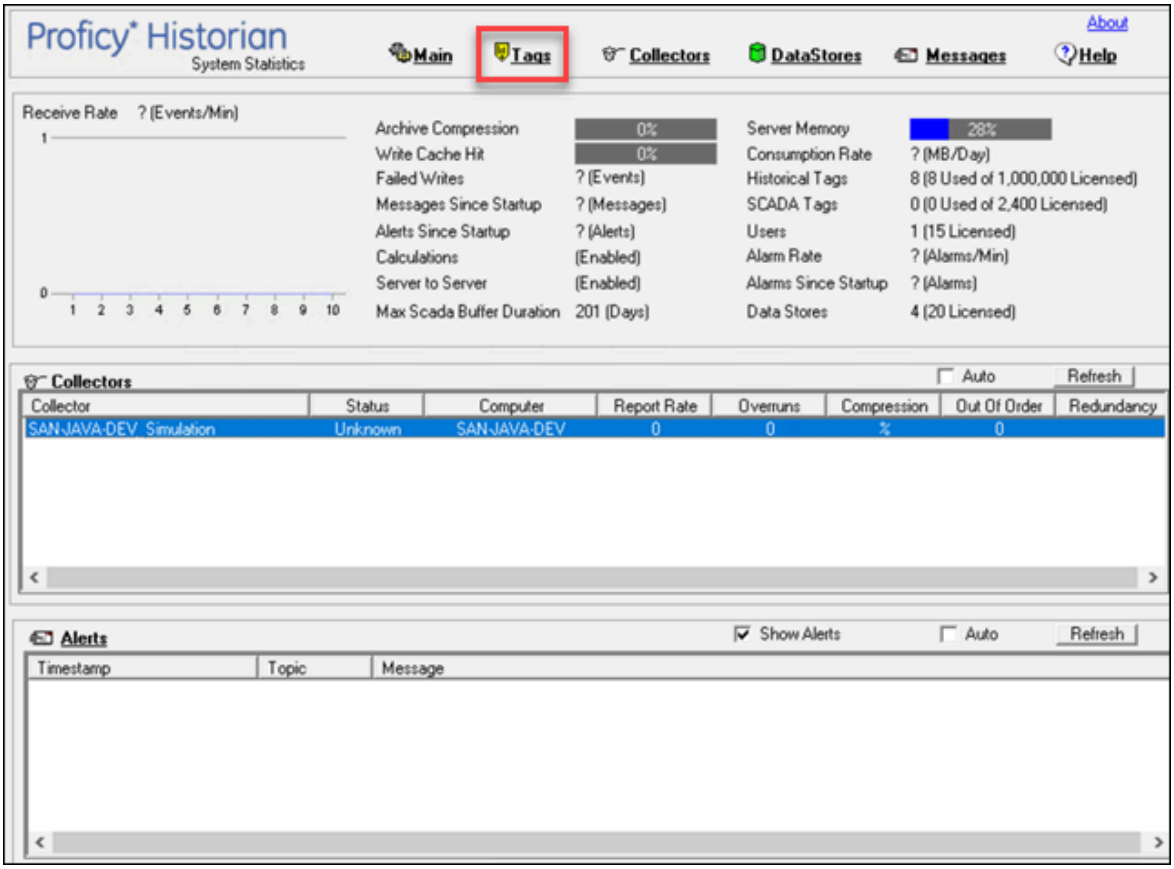
| Field | Description |
|---|---|
| | You can change a tag to an array tag later as well. However, when you do so, only the latest data is retrieved. If you want to get the old data, you must change the tag back to its previous type. |
| **Time Resolution** | Select the duration at which you want to collect data for the tag. For example, if you select **Seconds**, data is collected every second. |

**Note:**
If you add a tag for a Server-to-Server collector, set the **Time Adjustment** field for the tag to **Adjust for Source Time Difference** after you add the tag. This field is available under **Tags > Advanced**. This is applicable only for polled data collection.

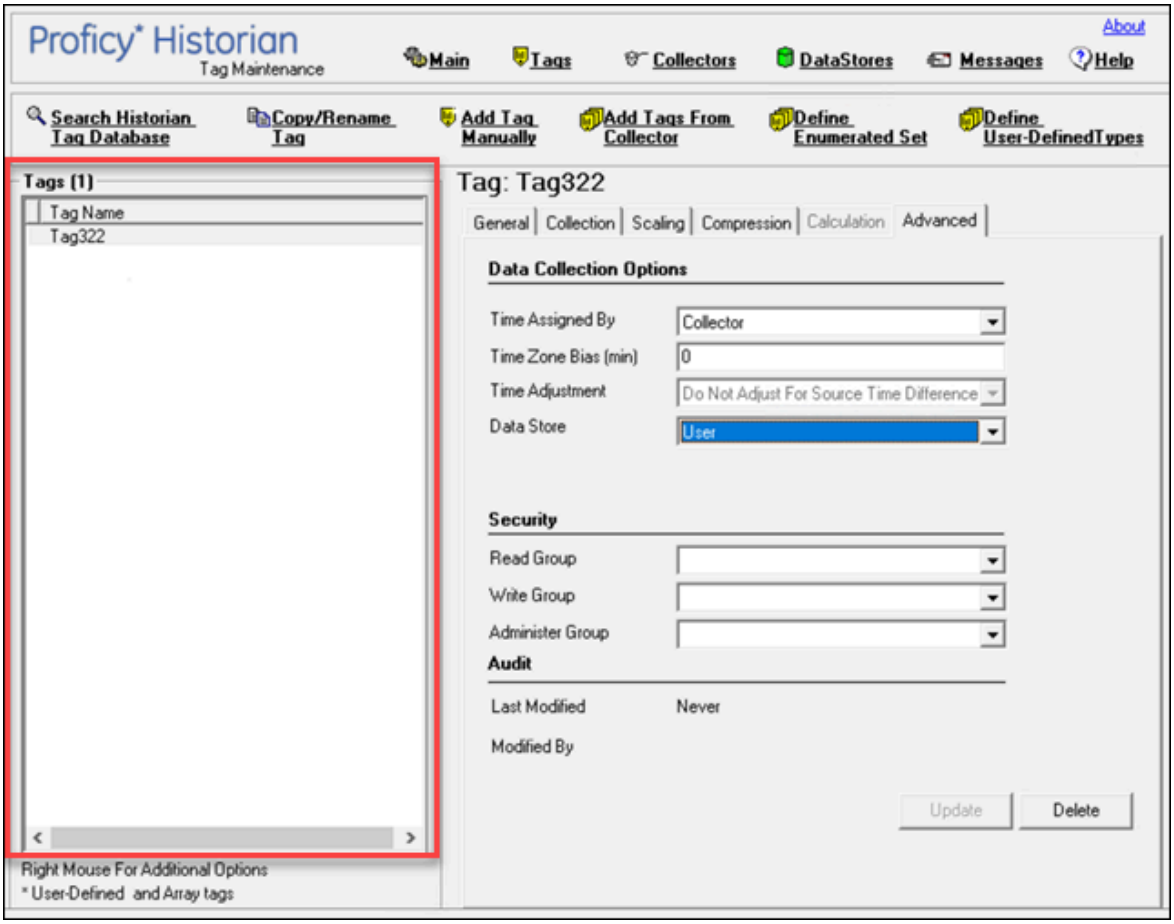
5. Select **OK**.
   The tag is created.

## Copy a Tag

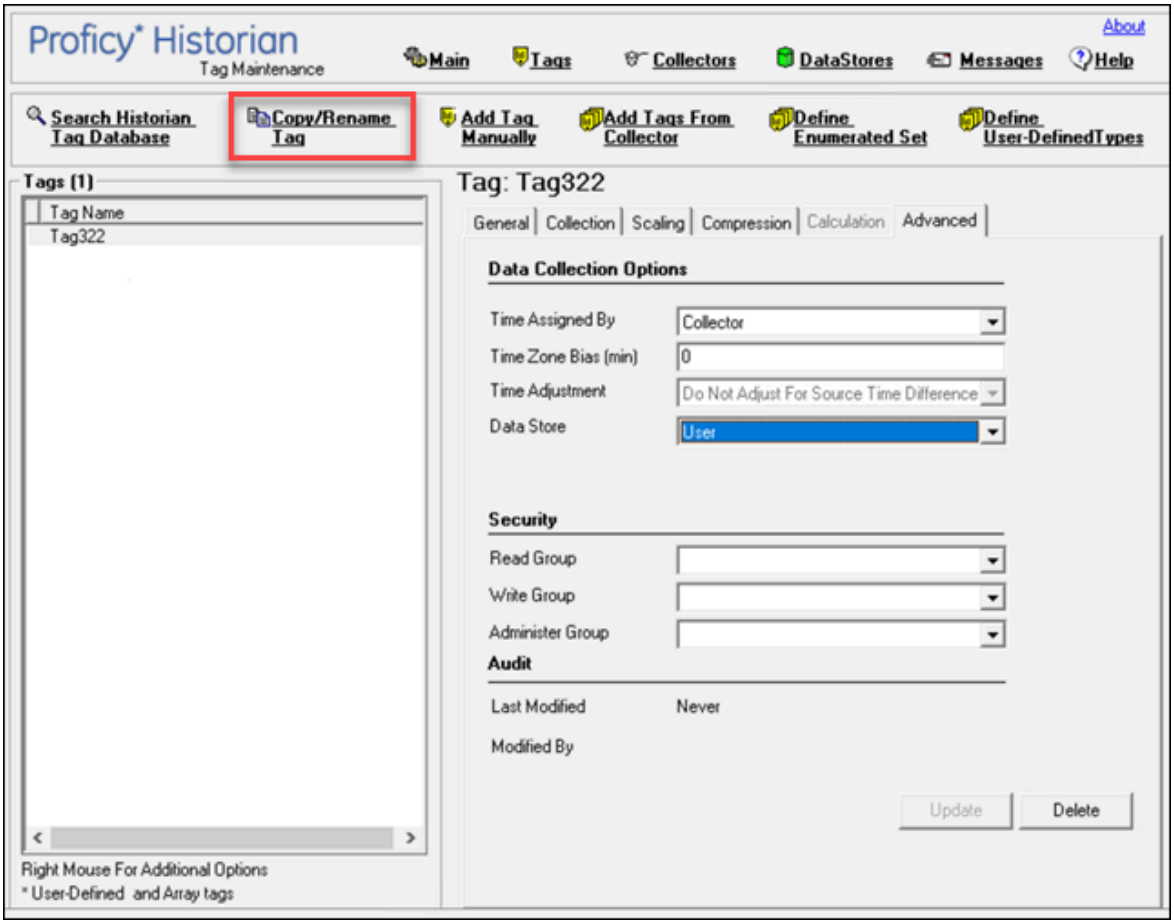
1. Access Historian Administrator *(on page 823)*.
2. Select **Tags**.




3. Select the tag that you want to copy.




4. Select **Copy/Rename Tag**.




The **Copy/RenameTag** window appears.

5. Select **Copy**.
6. Enter a new tag name.
7. Select **OK**.

   The tag is copied.

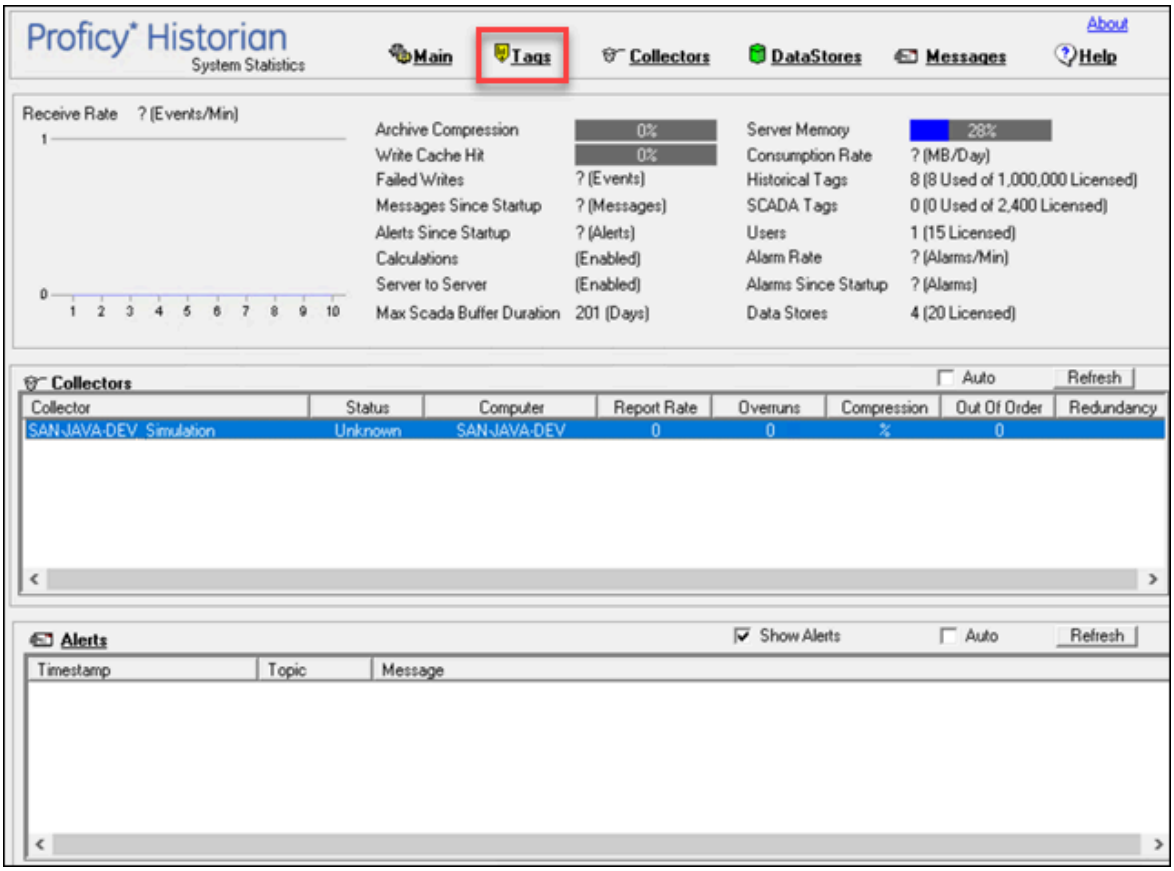
## Rename a Tag

- You must be a member of the administrator's group with tag level security.
- If you want to rename a tag permanently, to avoid loss of data, stop the collector instance.

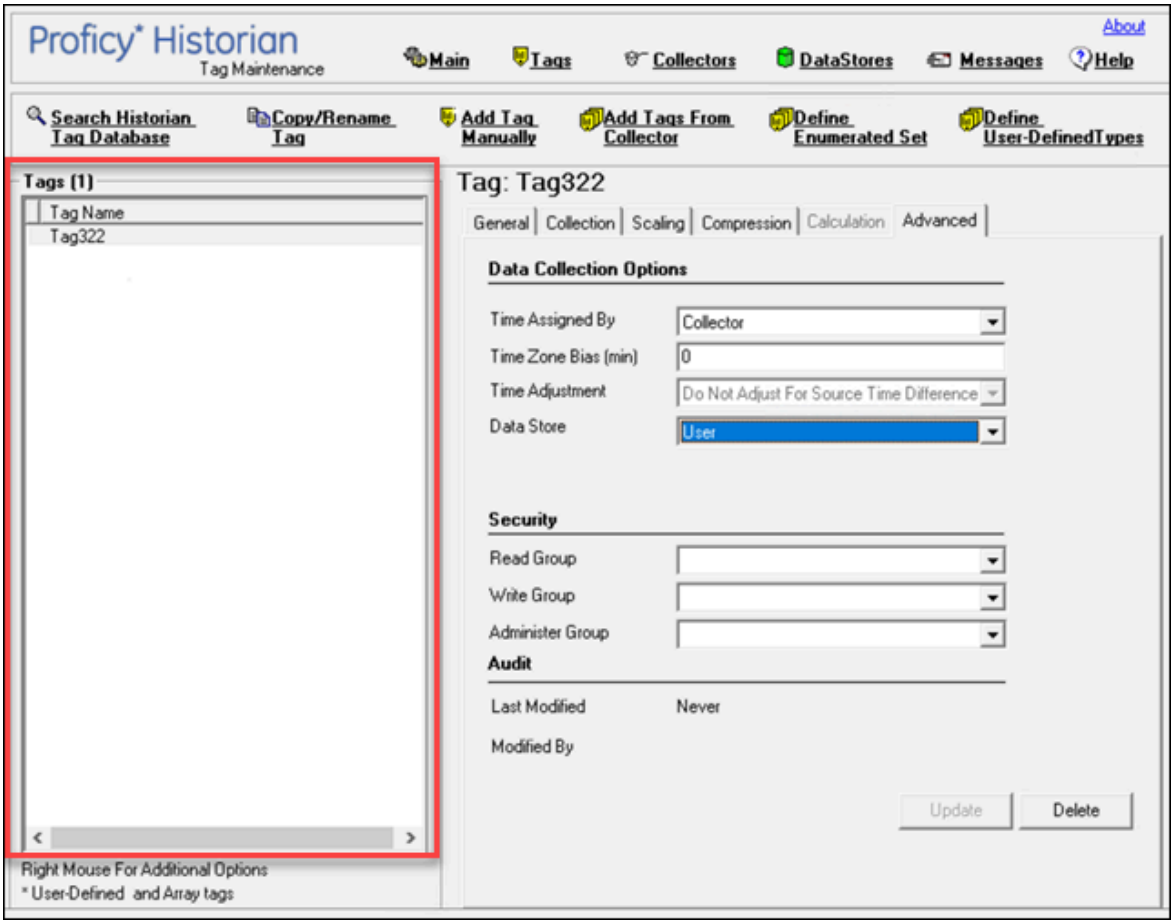
When you rename a tag, you can choose between the following options:

- **Rename using an alias:** In this case, the old name is called the tag alias. You can retrieve tag data using the tag alias as well. When you copy a tag, the tag alias is captured as well to aid in an audit trail.
- **Rename permanently:** In this case, the old name is no longer captured. Therefore, you can create another tag with this old name. You cannot store and forward data using the old name. This implies that data for the tag is collected separately for the new name.

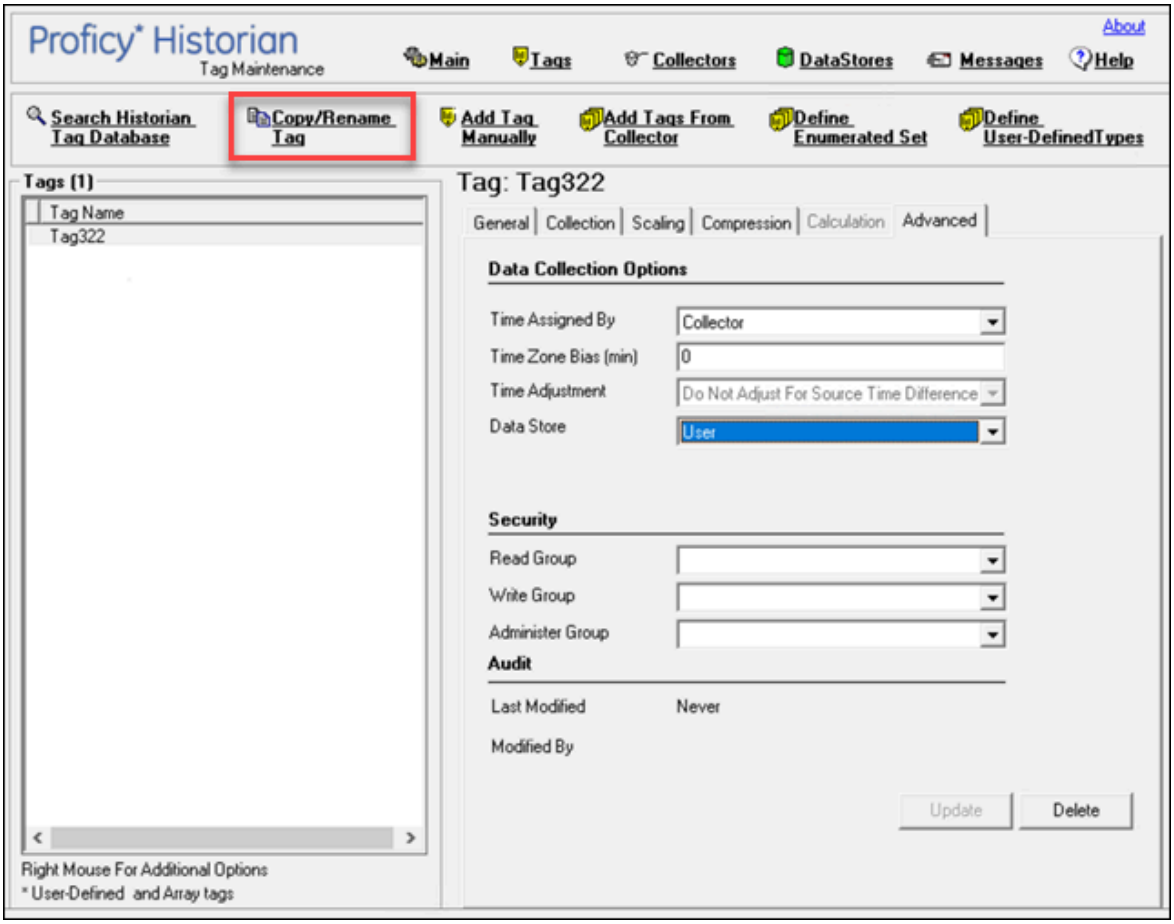
1. Access Historian Administrator *(on page 823)*.
2. Select **Tags**.




3. Select the tag that you want to rename.




4. Select **Copy/Rename Tag** .




The **Copy/RenameTag** window appears.

5. If you want to rename the tag using an alias, select **Rename (Alias)**. If you want to rename the tag permanently, select **Permanent Rename**.
6. Select **OK**.

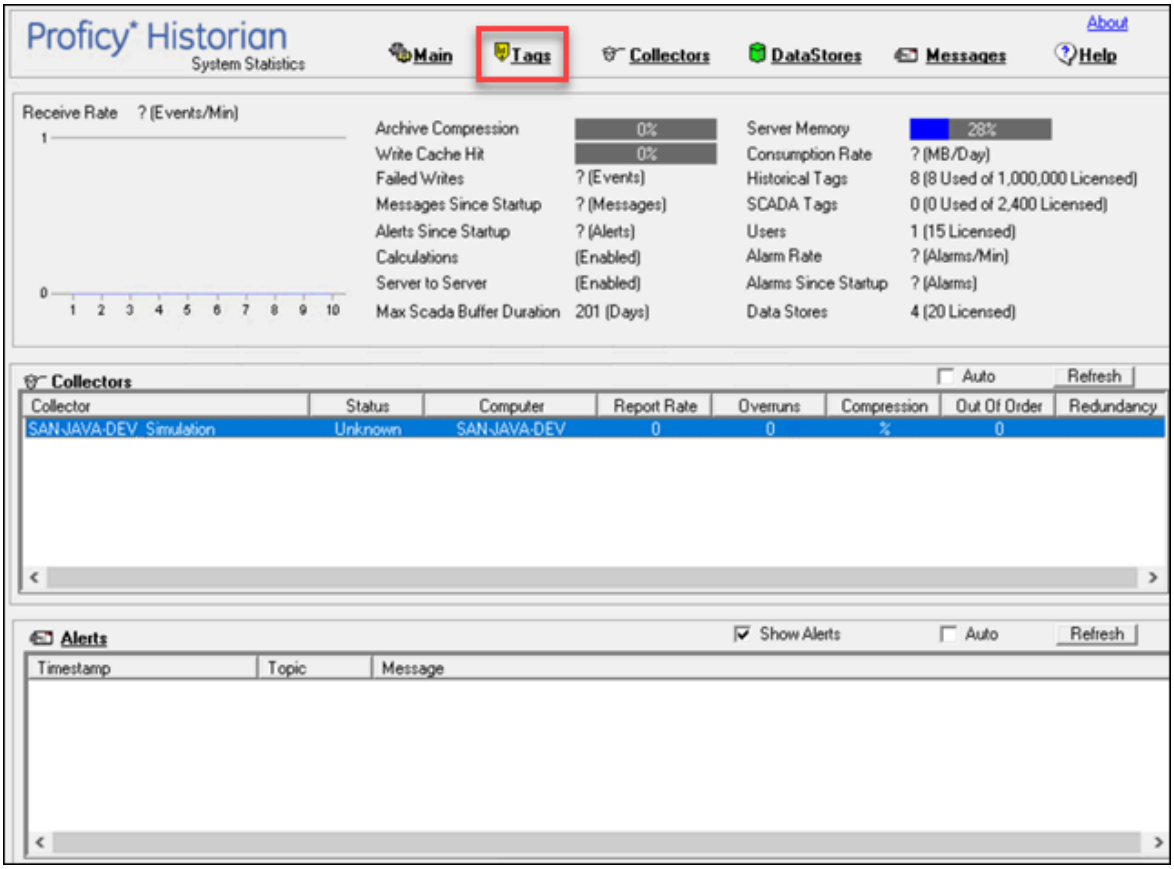
If you have renamed the tag permanently:

- If the tag is used as a trigger, reassign the trigger.
- Restart the collector instance.

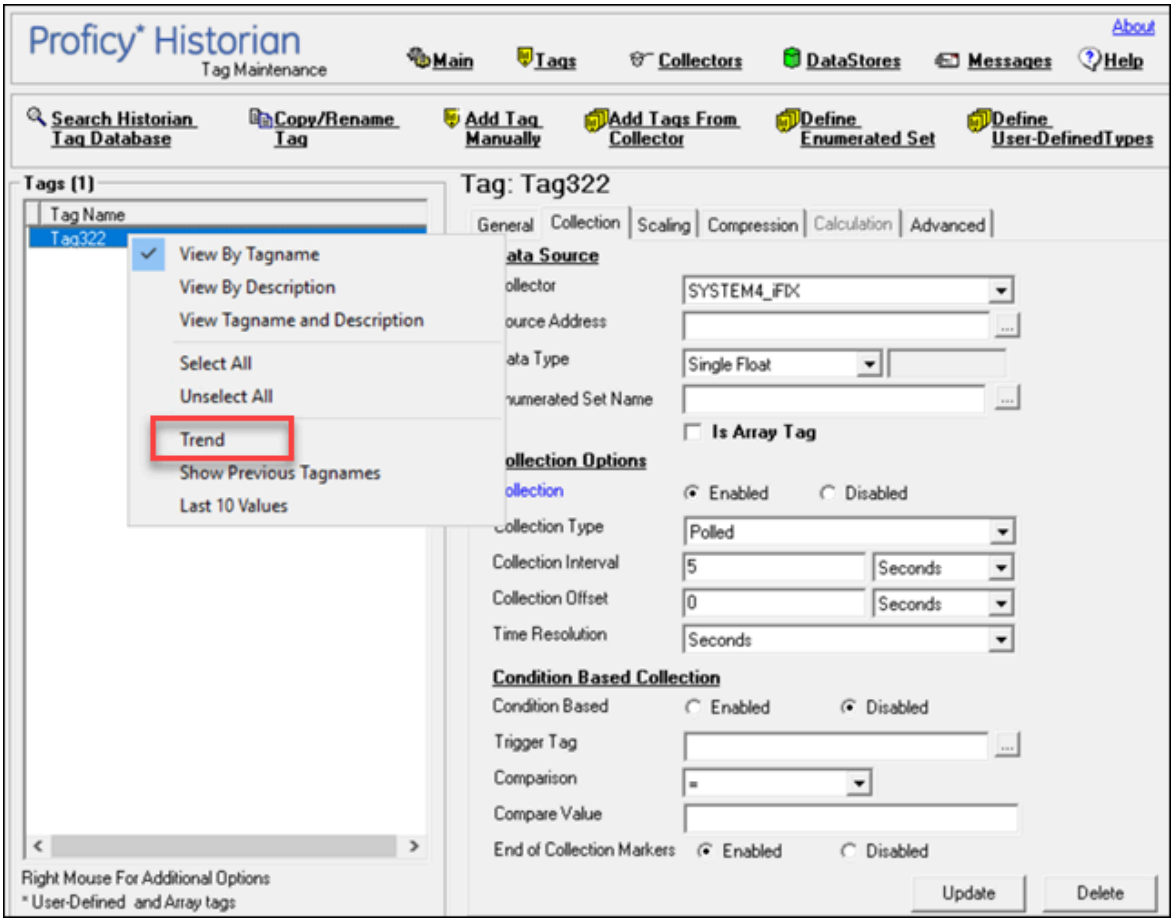
## View Tag Trends and Raw Data

This topic describes how to access the trend chart of tag data. Note that the tag trend should not be used for detailed data. And, trend data is not supported for array tags.

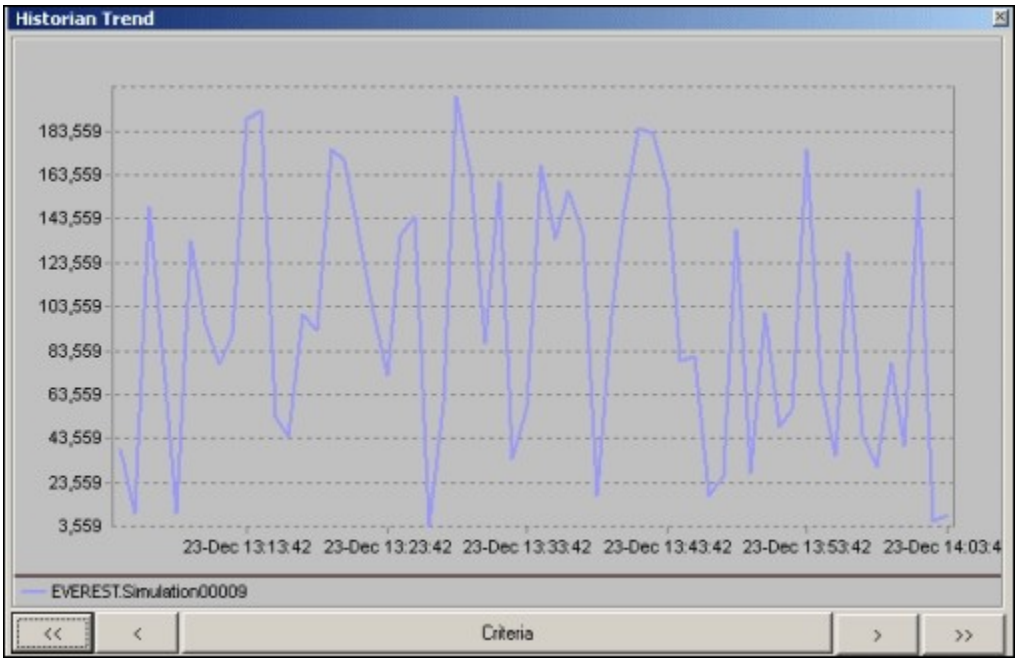
1. Access Historian Administrator *(on page 823)*.
2. Select **Tags**.




3. Right-click the tag whose trend chart you want to access, and then select **Trend**.




The trend chart of the tag values appears.



**Note:**

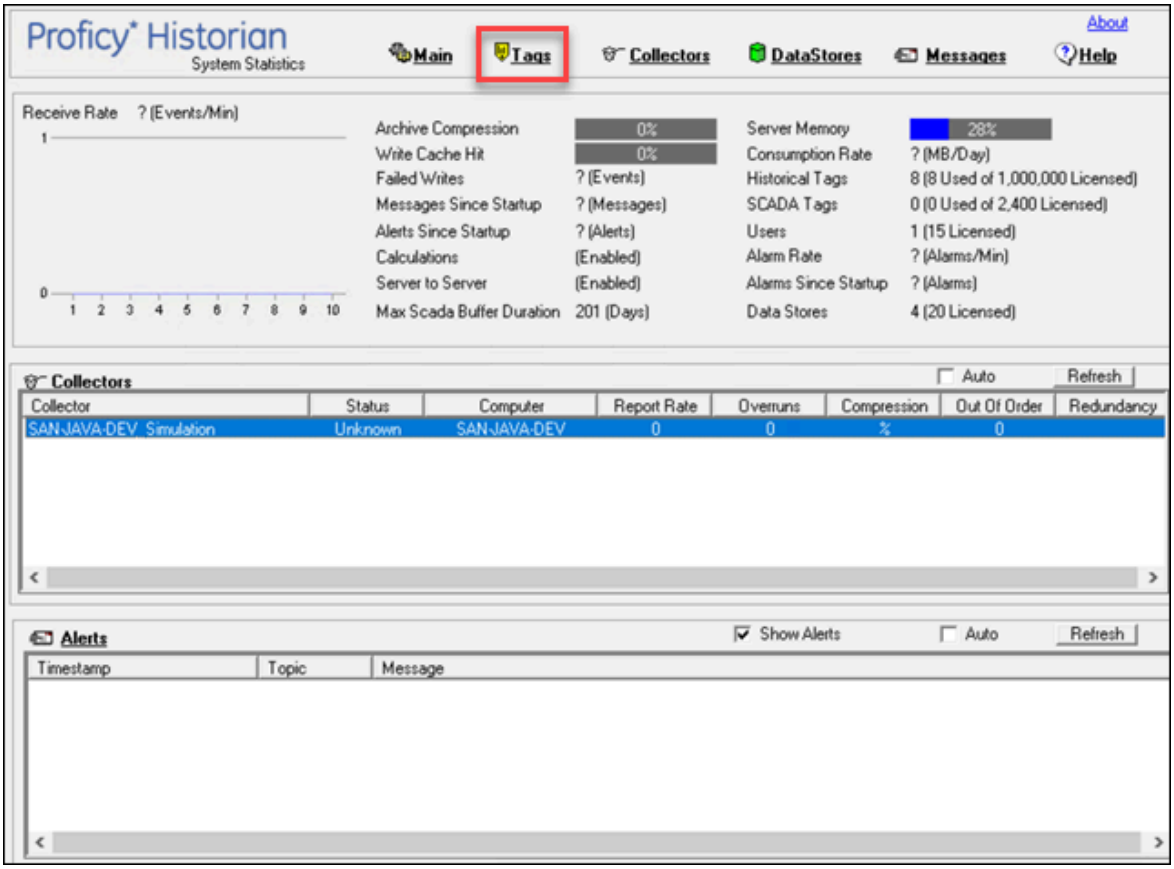
To change the criteria, select **Criteria**, and then enter values as described in the following table.

| Field | Description |
|---|---|
| **Start Time** | Select the start time of the trend chart. |
| **End Time** | Select the end time of the trend chart. |
| **Sampling** | Select the data type that you want to use. |
| **Interval** | Enter the interval at which you want to plot the tag data. |
| **Criteria Strings** | Enter the sampling mode, calculation mode, and/or query modifiers. Query modifiers are used to specify various ways of retrieving data from Historian. For example, you can request raw data with good quality only by specifying the criteria string as: `RAWBYTIME#ONLYGOOD`. The sampling mode specified with criteria strings takes precedence over the mode specified in the **Sampling** field. |

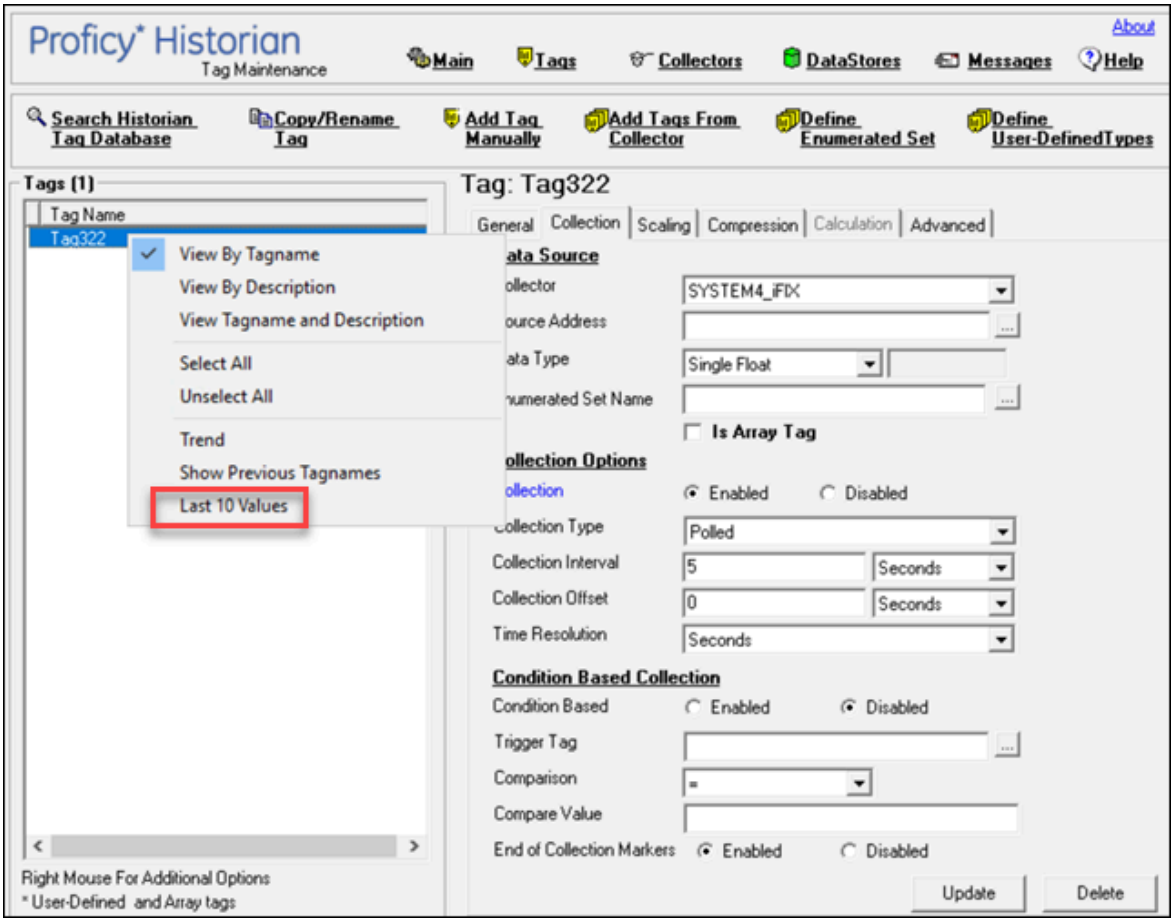
You can also scroll back and forth on the x-axis time scale by selecting on the single and double left and right arrows at the bottom of the page.

## View the Last 10 Raw Values of a Tag

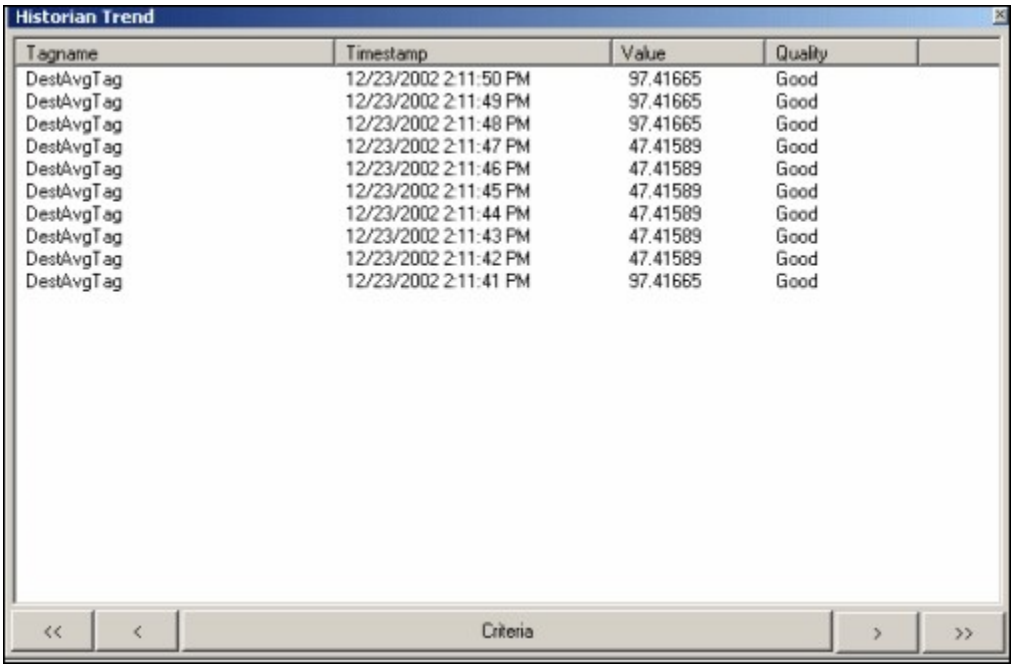
1. Access Historian Administrator *(on page 823)*.
2. Select **Tags**.



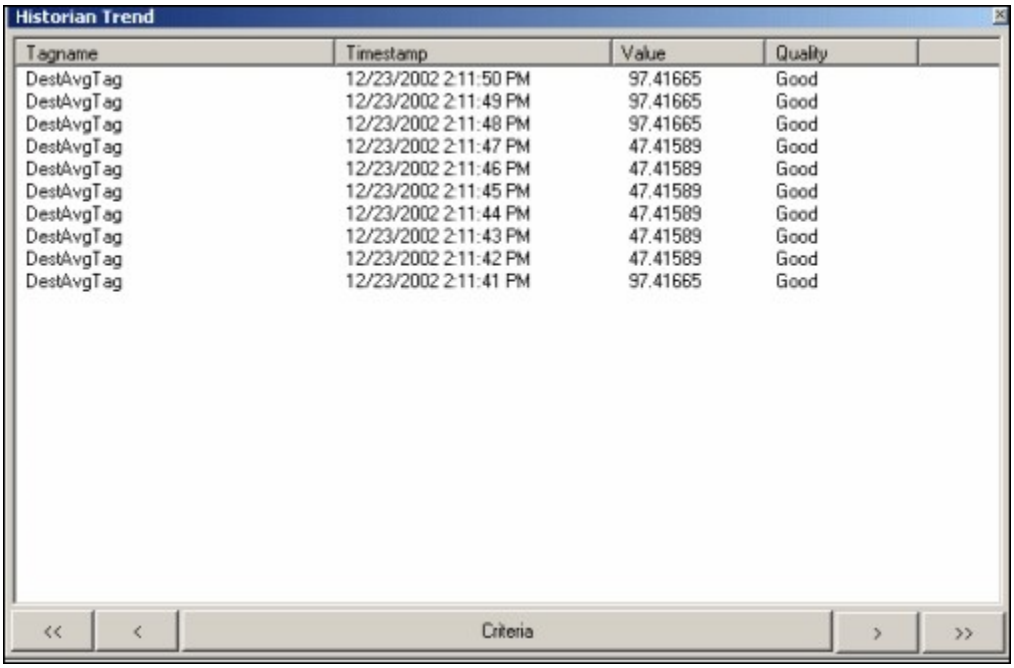

3. Right-click the tag whose last 10 values you want to access, and then select **Last 10 Values**.




The last 10 values of the tag appear.

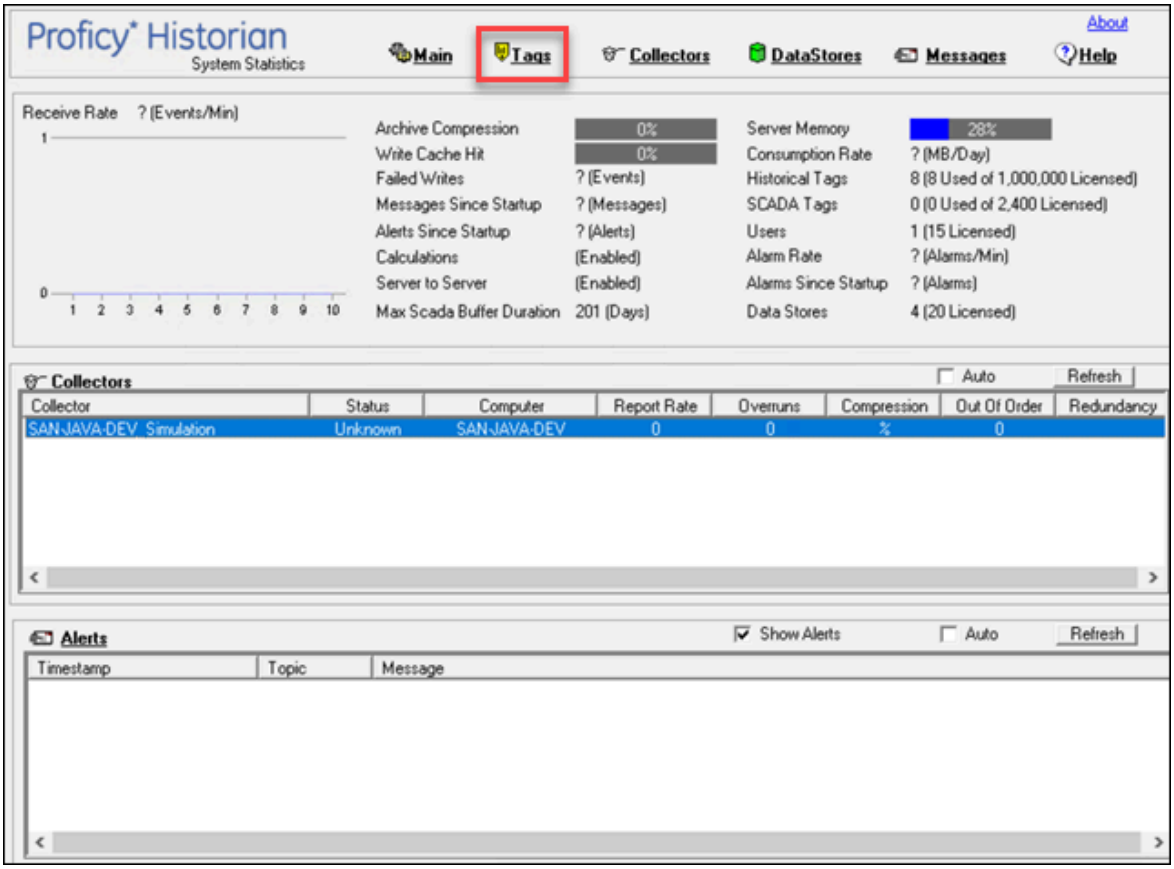


For an array tag, each element is displayed as a separate row with the tag name and the index.

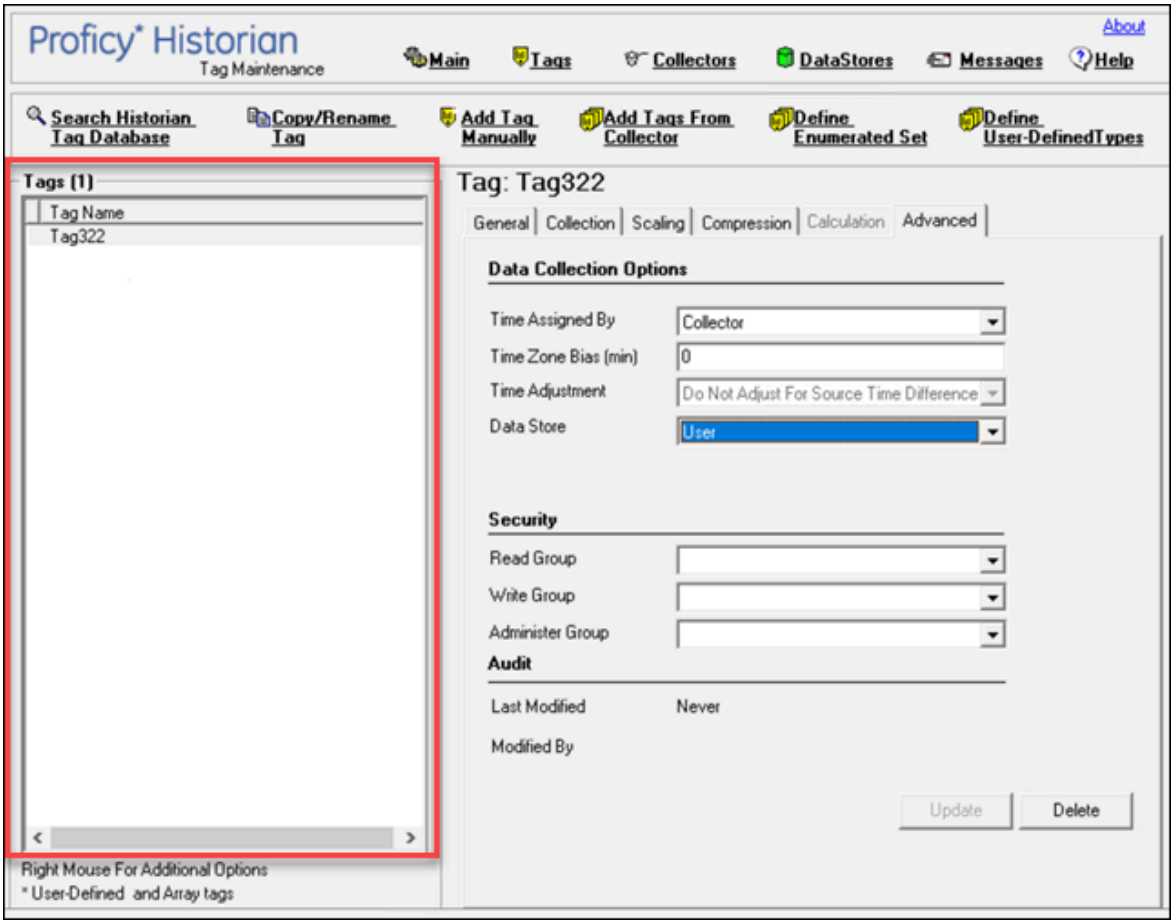



## Stop Data Collection

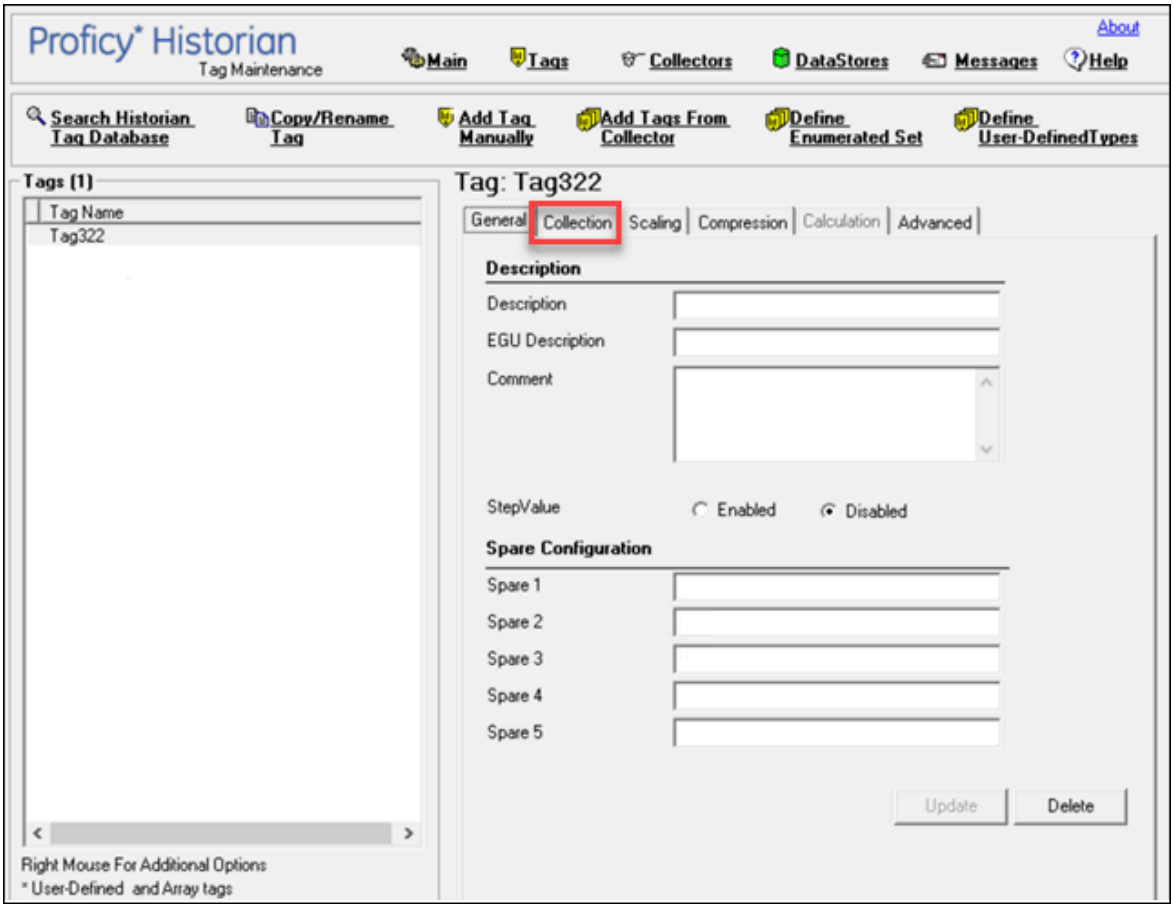
1. Access Historian Administrator *(on page 823)*.
2. Select **Tags**.



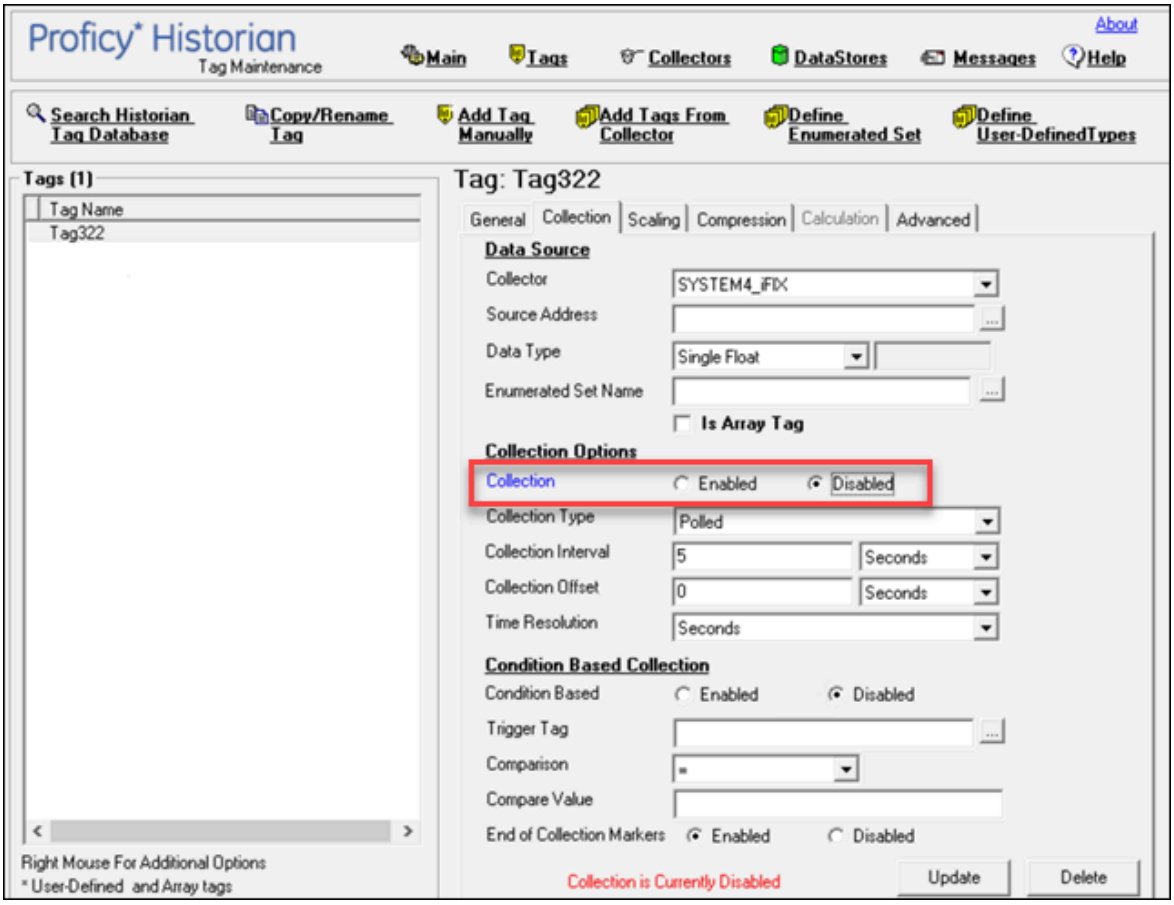

3. Select the tag whose data collection you want to stop.




4. Select **Collection**.

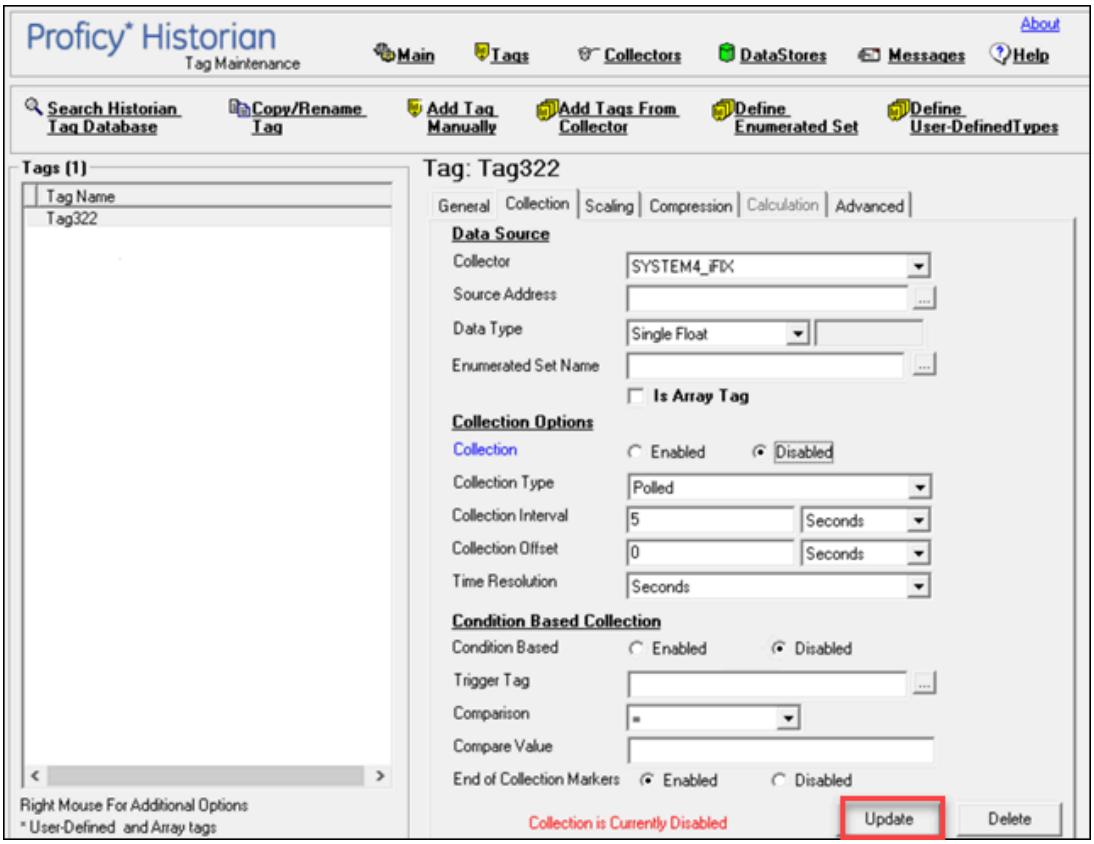



5. For the **Collection** field, select **Disabled**.




6. Select **Update**.
7. To stop data collection on a tag:
    a. From the list in the left-hand window of the page, select a tag.
    b. In the window on the right side of the page, select **Collection**.
    c. For the **Collection** field, select the **Disabled** option.

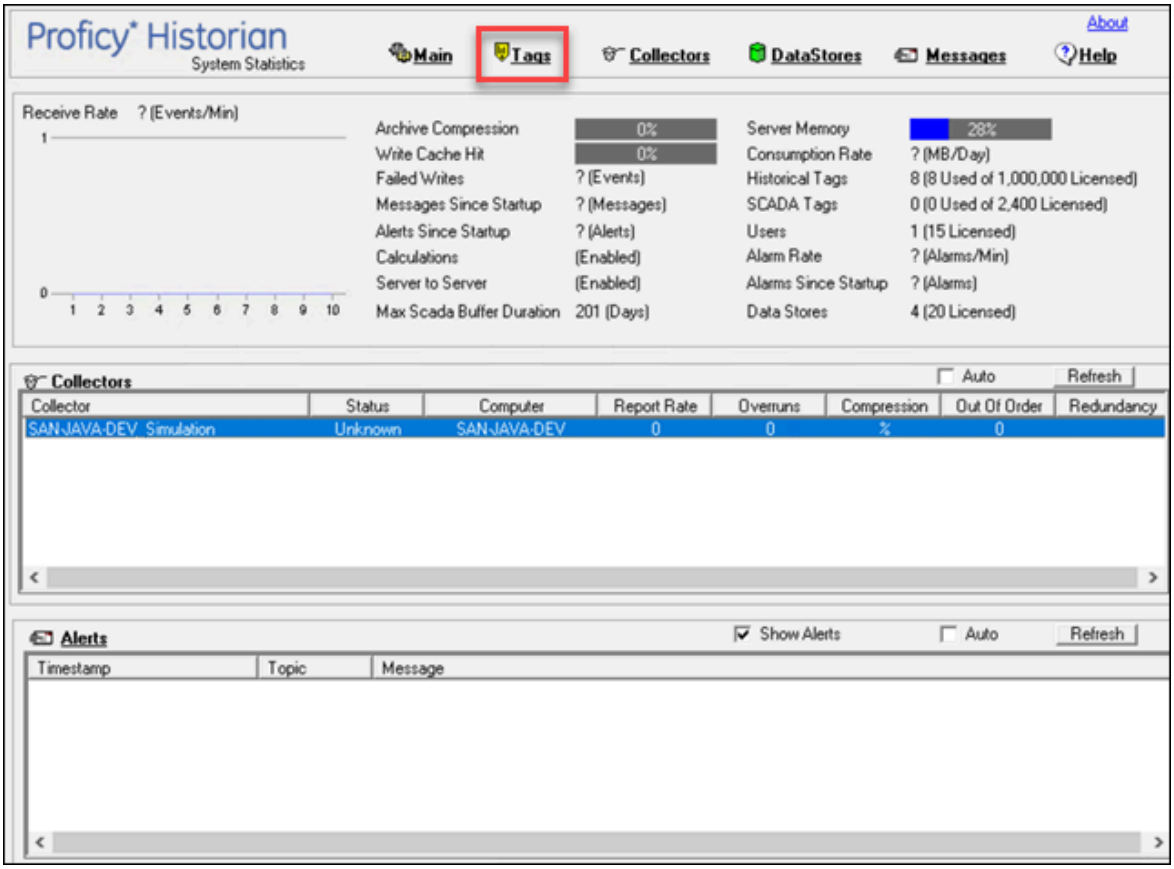
d. Select **Update**.



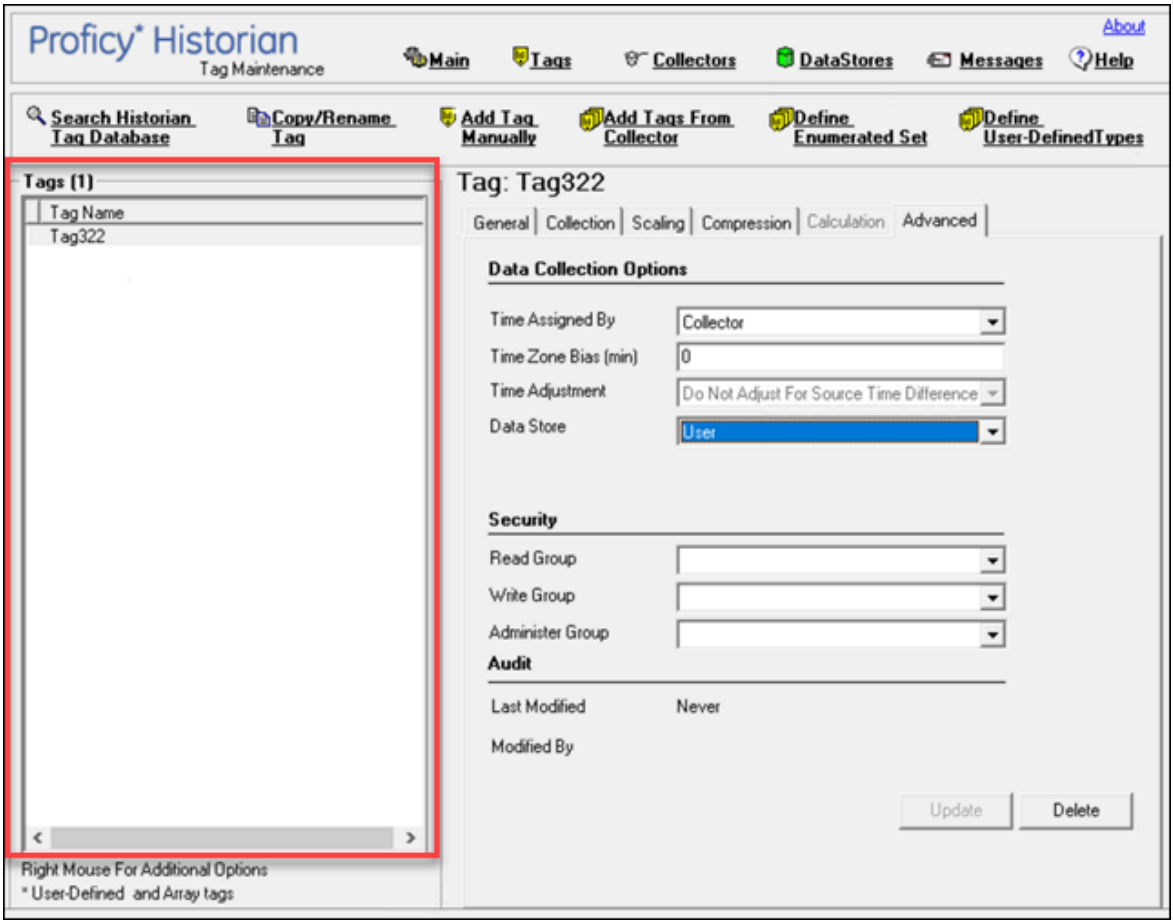

The data collection for the tag is stopped.

## Resume Data Collection

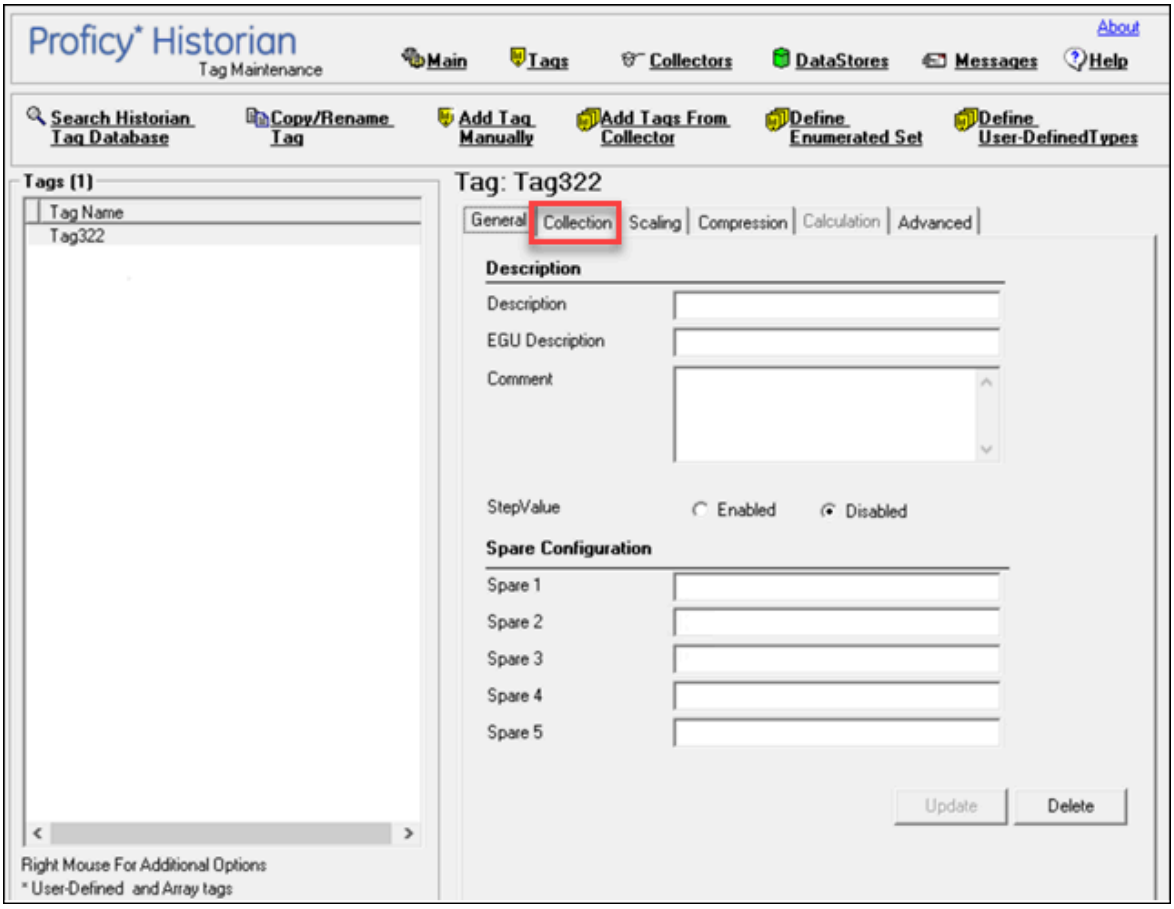
1. Access Historian Administrator *(on page 823)*.
2. Select **Tags**.



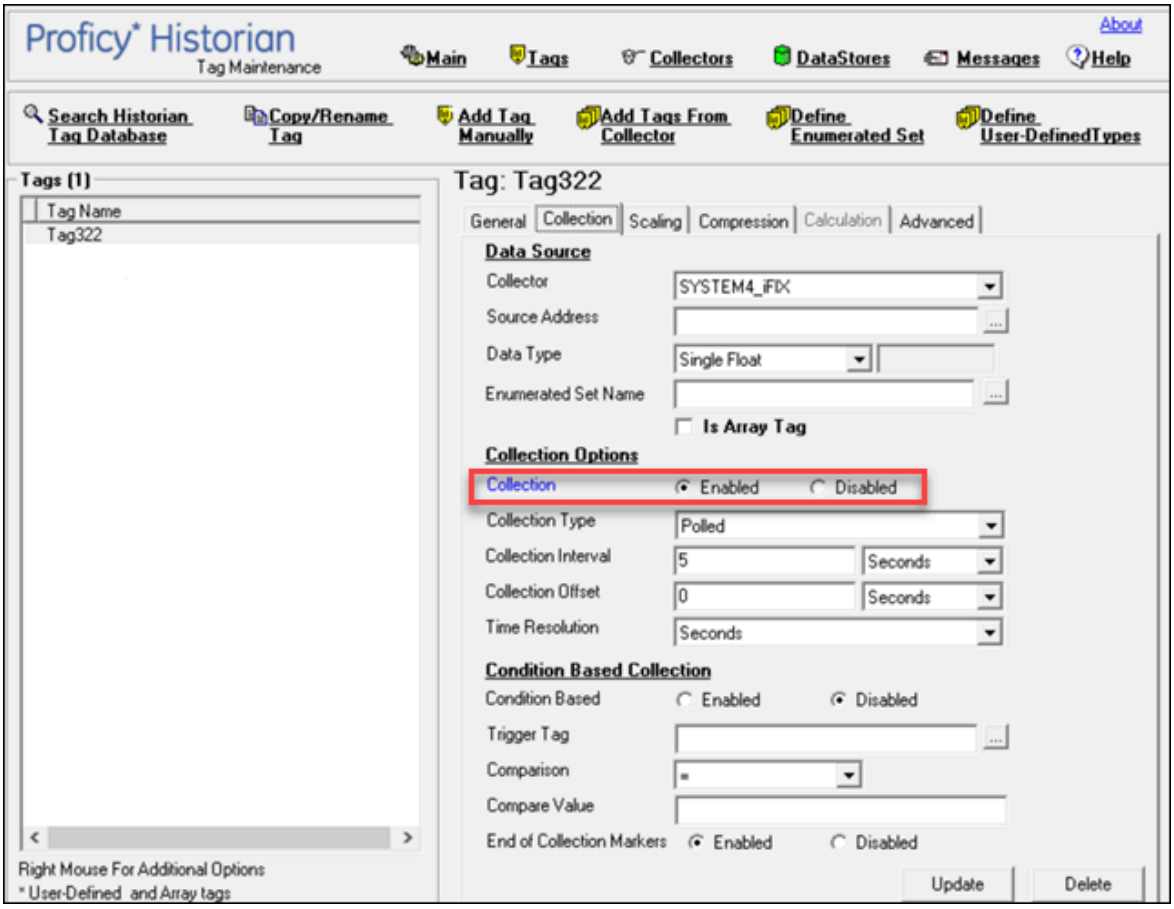

3. Select the tag whose data collection you want to resume.



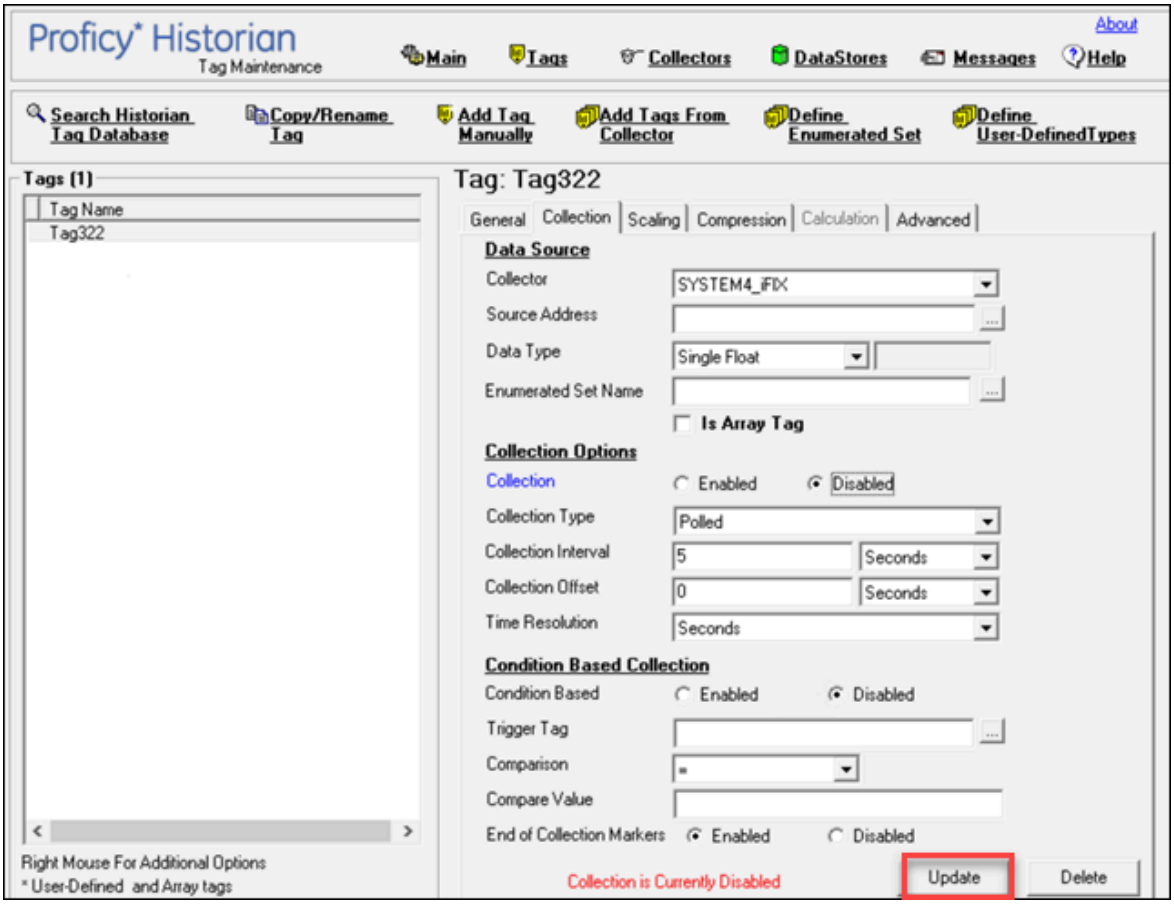

4. Select **Collection**.




5. For the **Collection** field, select **Enabled**.




6. Select **Update**.




The data collection for the tag is resumed.

## Get all the Fields Related to a Tag

1. Access Registry Editor.
2. Create a DWORD (32-bit) registry entry named GetAllTagProps for the collector in the following registry path: `HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Intellution, Inc.\iHistorian\Services\OPCCollector\*_OPC_Intellution_IntellutionGatewayOPCServer`
3. Provide the value 1 for the registry entry.

## Remove A Tag

When you remove a tag, it is removed from the tag database, but all data for that tag is retained in the archive and the tag name cannot be reused. Since the tag data is still available from the archive, you can still reference that tag from within a calculation formula, for example, or by using the Excel Add-In.

If, however, you want to remove the tag data as well from the archive, you can delete it permanently *(on page 938)*.

Whenever you delete/remove tags, the following collectors reload only the modified tag(s) without restarting the collectors.

- OPC Collector
- iFIX Collector
- Calculation Collector
- Simulation Collector
- Server to Server Collector
- PI Collector
- PI Distributor

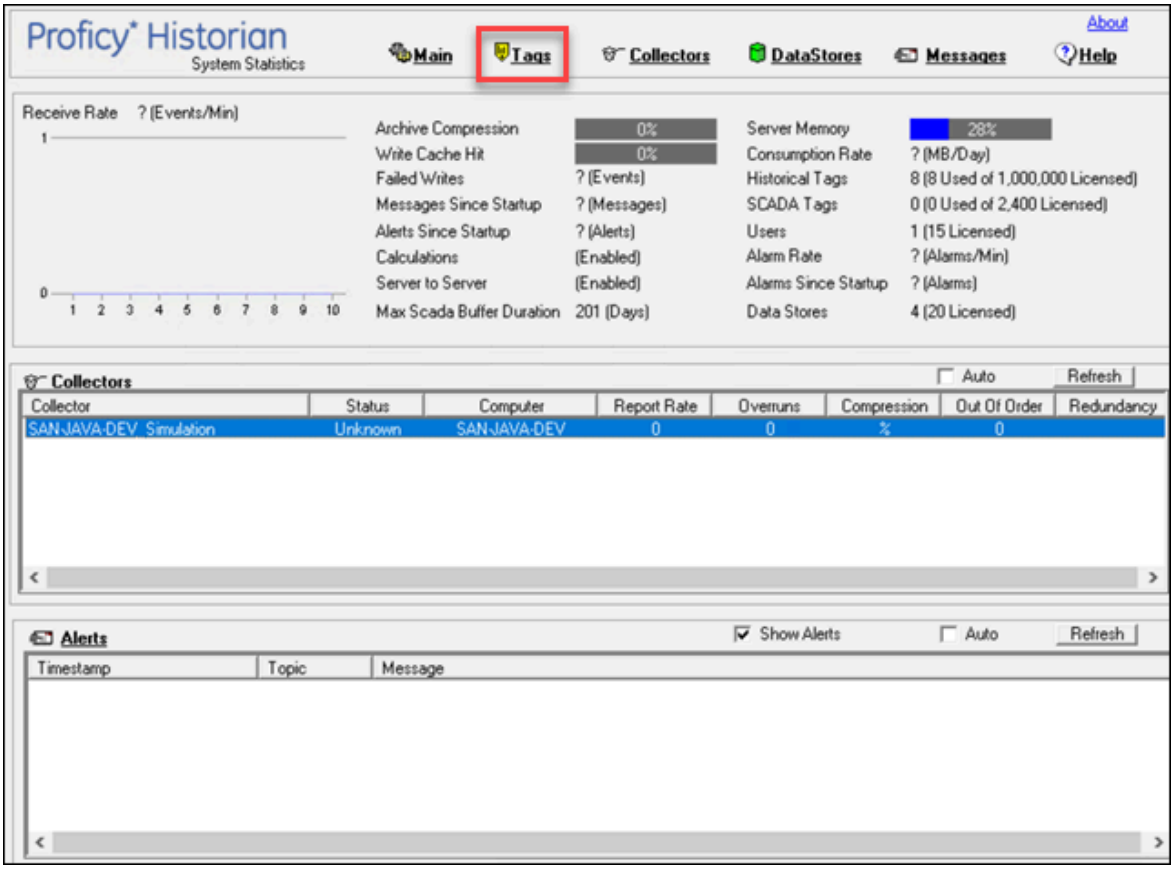
By default, the On-line Tag Configuration Changes option is enabled, which allows a tag to stop and restart data collection without restarting the collector. If you disable the On-line Tag Configuration Changes option, any changes you make to the tags do not affect collection until you restart the collector. To enable or disable the On-line Tag Configuration Changes option, access Historian Administrator, and then select **Collectors > Advanced**.

Restarting the collector stops and restarts the tag(s) collection and may record bad data samples to the collection. All the collector configuration changes done within a 30-second time frame are batched up together. To collect the modified data faster, update/modify a small set of tags at a time. If the modified tags get zero bad markers and available runtime values at the same time, then precedence is given to available runtime values instead of zero bad markers.

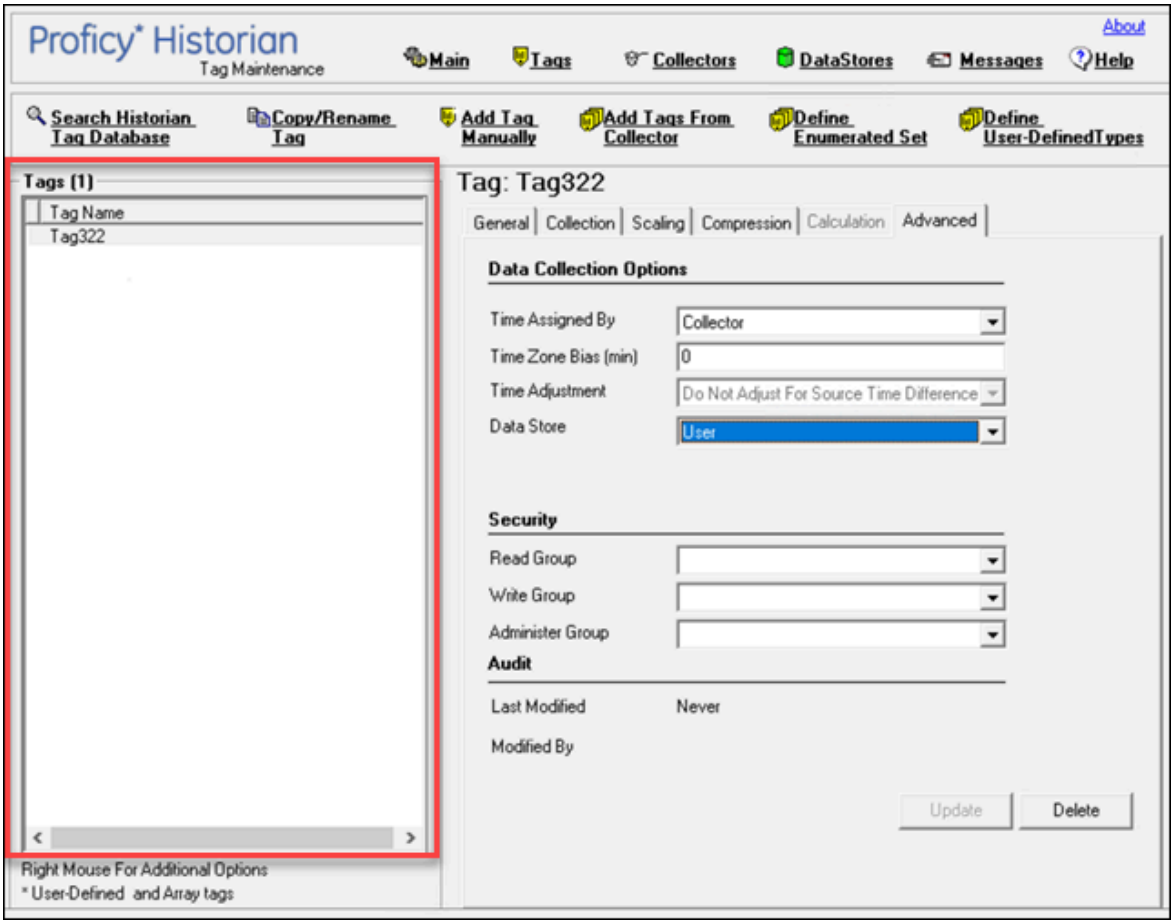
**Tip:**

- To collect the modified data faster, update/modify a small set of tags at a time.
- When updating large sets of tags at the same time, the best practice is to disable the **On-line Tag Configuration Changes** option and restart the collector after modification.

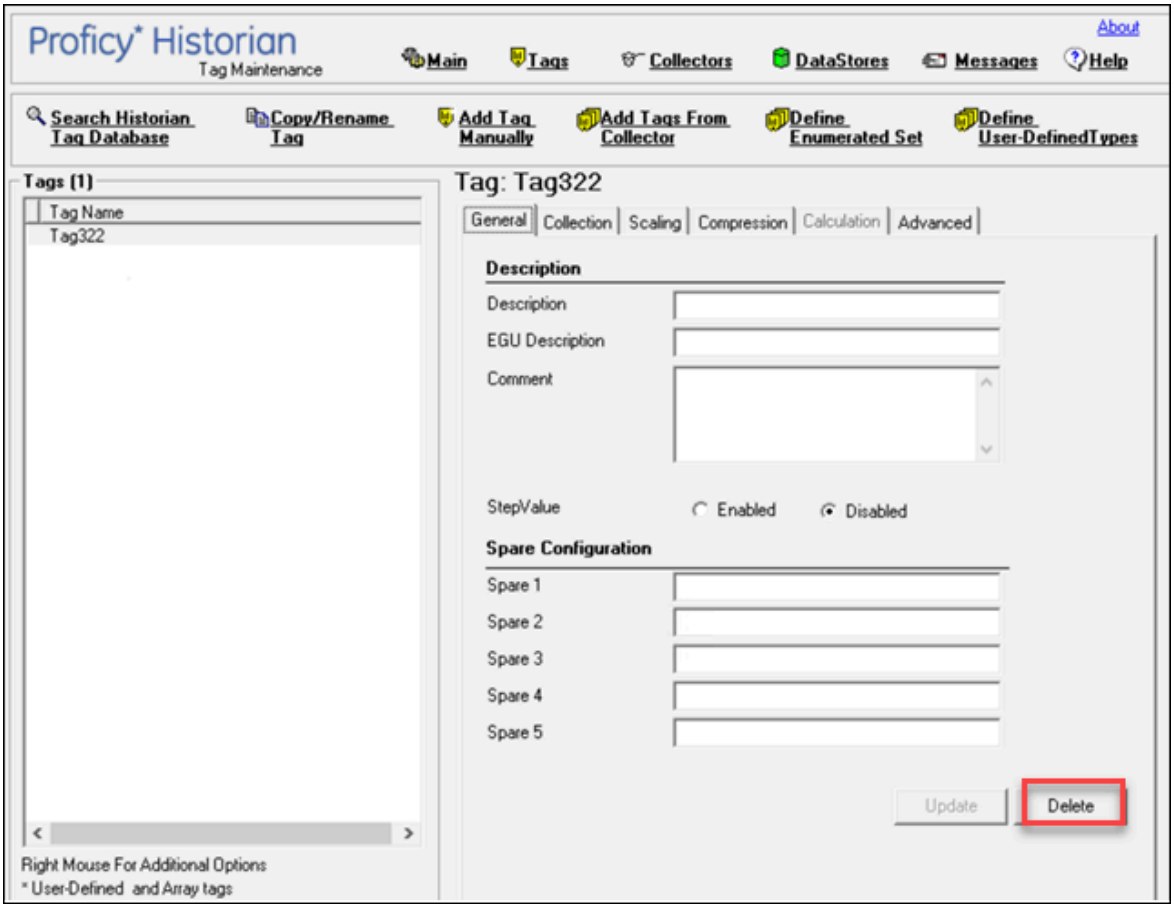
1. Access Historian Administrator *(on page 823)*.
2. Select **Tags**.




3. Select the tag that you want to remove.




4. Select **Delete**.



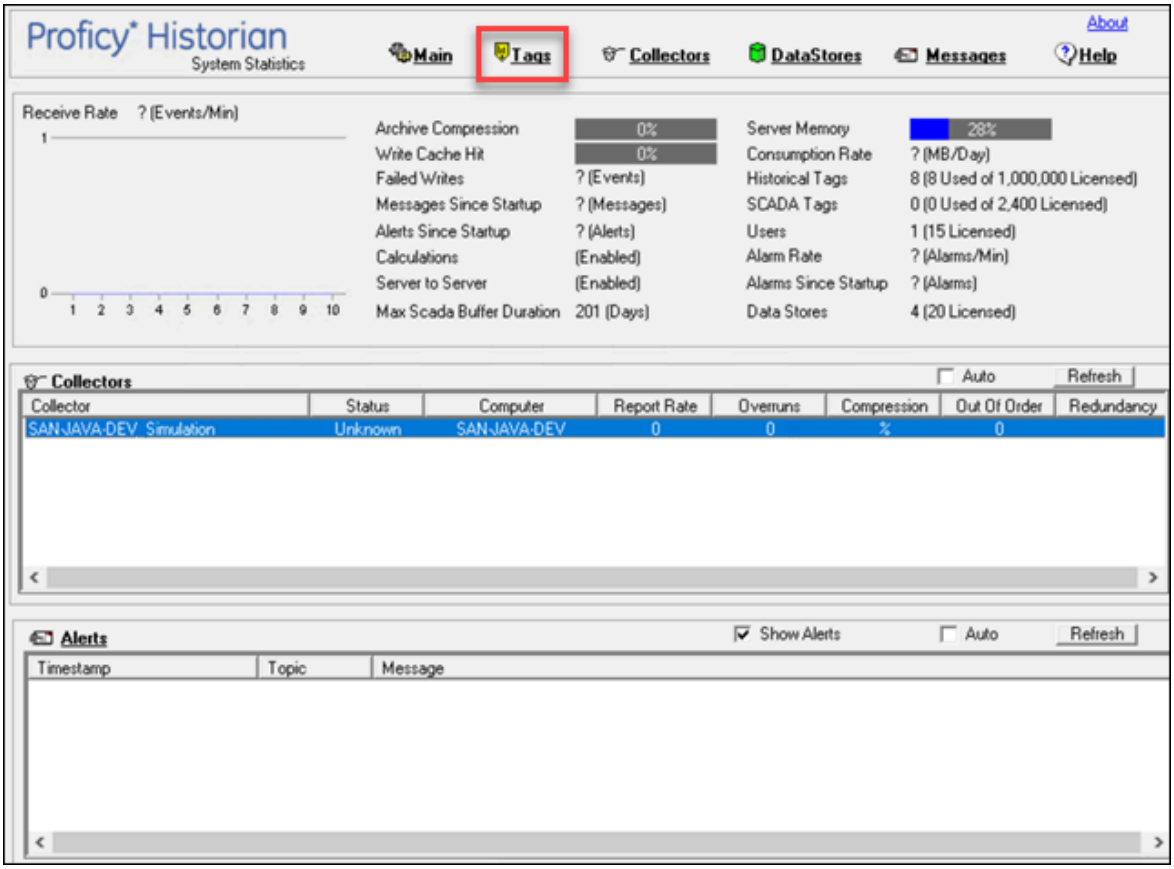

The **Delete Tags** window appears.

5. Select **Remove Tag from System**, and then select **OK**.
   A message box appears, asking you to confirm that you want to remove the tag.
6. Select **Yes**.
   The tag is removed.

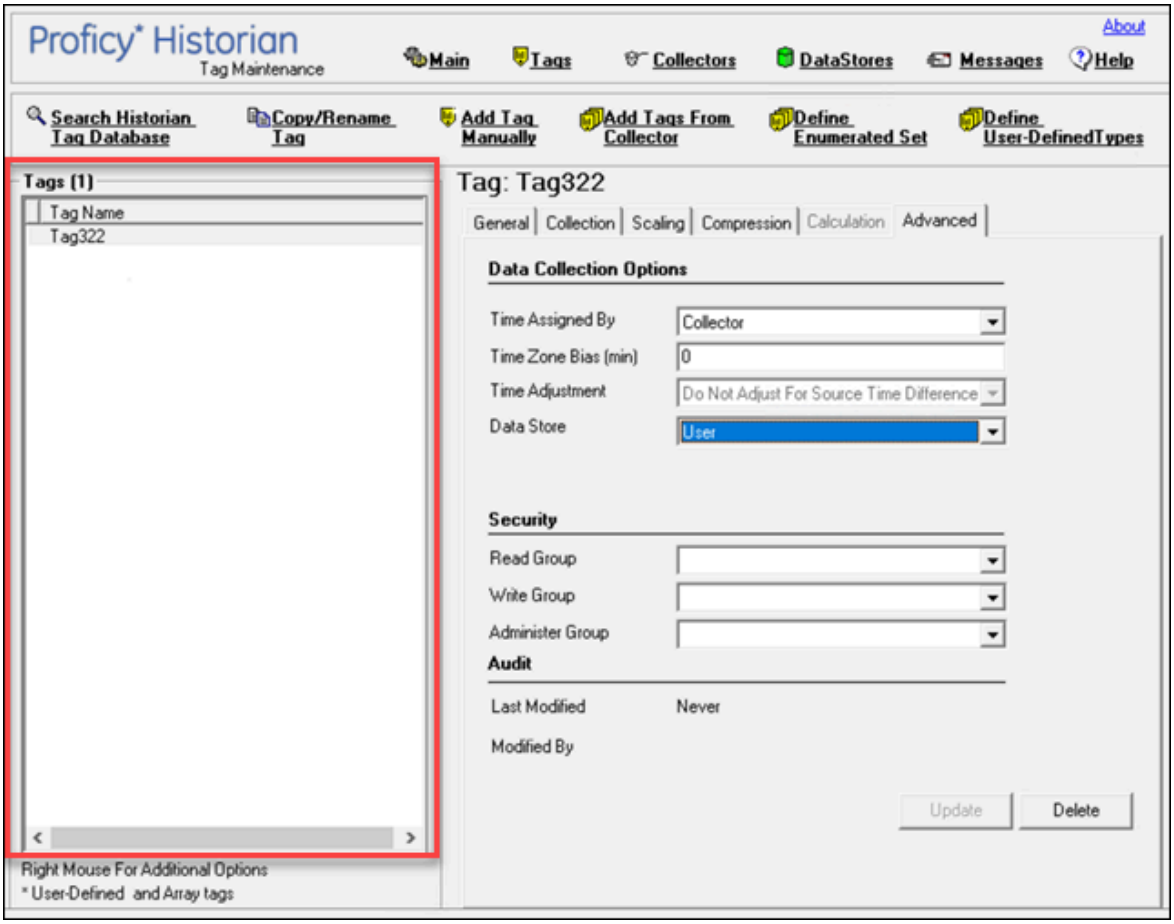
## Deleting Tags Permanently

When you delete a tag, the tag as well as all the data for that tag is removed from the archive and the tag name is available for reuse. You can no longer query the data for that tag. If, however, you want to just remove the tag, but retain the tag data, refer to Remove A Tag *(on page 934)*.

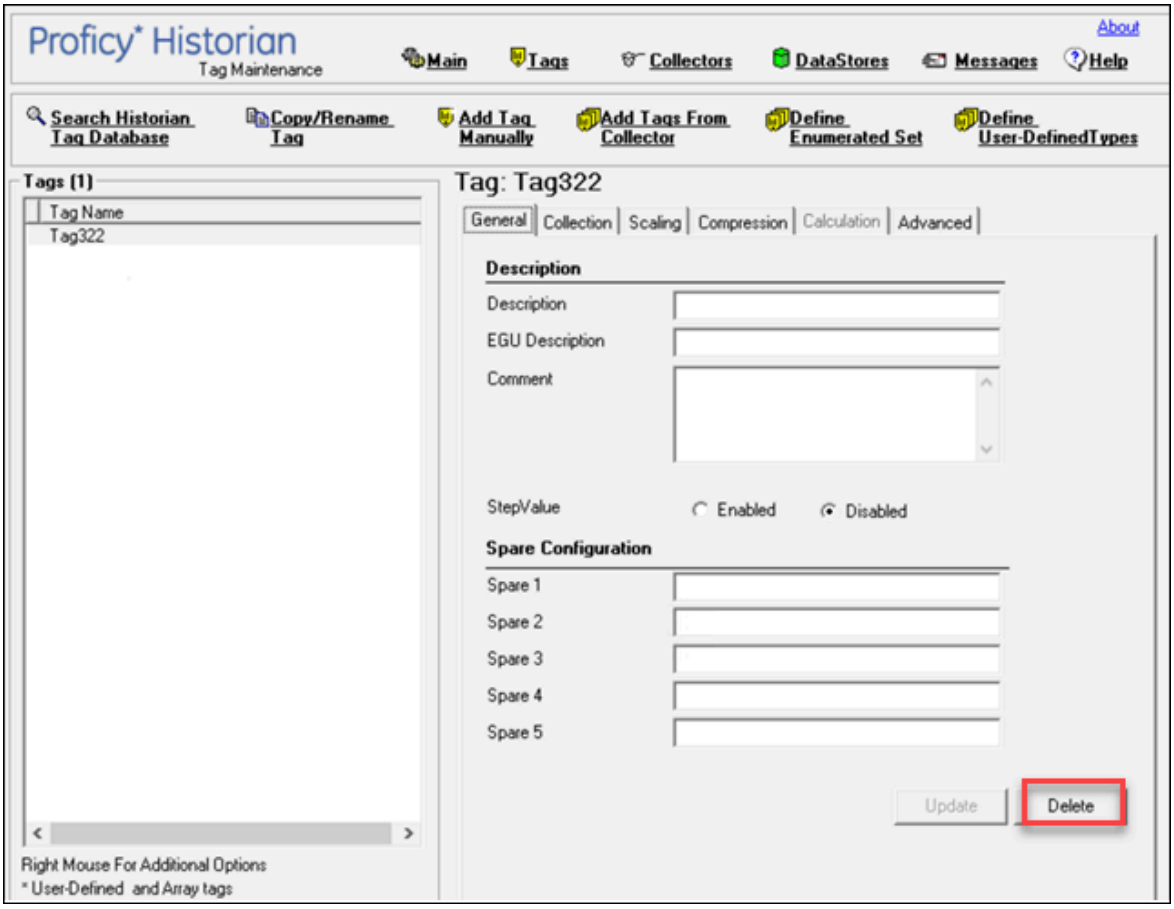
1. Access Historian Administrator *(on page 823)*.
2. Select **Tags**.




3. Select the tag that you want to delete.




4. Select **Delete**.




The **Delete Tags** window appears.

5. Select **Permanently Remove Tags From System**, and then select **OK**.

   A message box appears, asking you to confirm that you want to delete the tag.

6. Select **Yes**.

   The tag is deleted.

## About User-Defined Data Types

Sometimes, a single tag cannot store all the required details of a parameter. For example, if you want to store the name, address, and phone number of the manufacturer of a machine, it may not be feasible to store all these details in a single tag, which uses a single data type. In such cases, you can create a user-defined data type (UDT), which includes one or more fields, and then apply that type to Historian tags. Each of these fields in a UDT can contain a different data type based on your requirement.

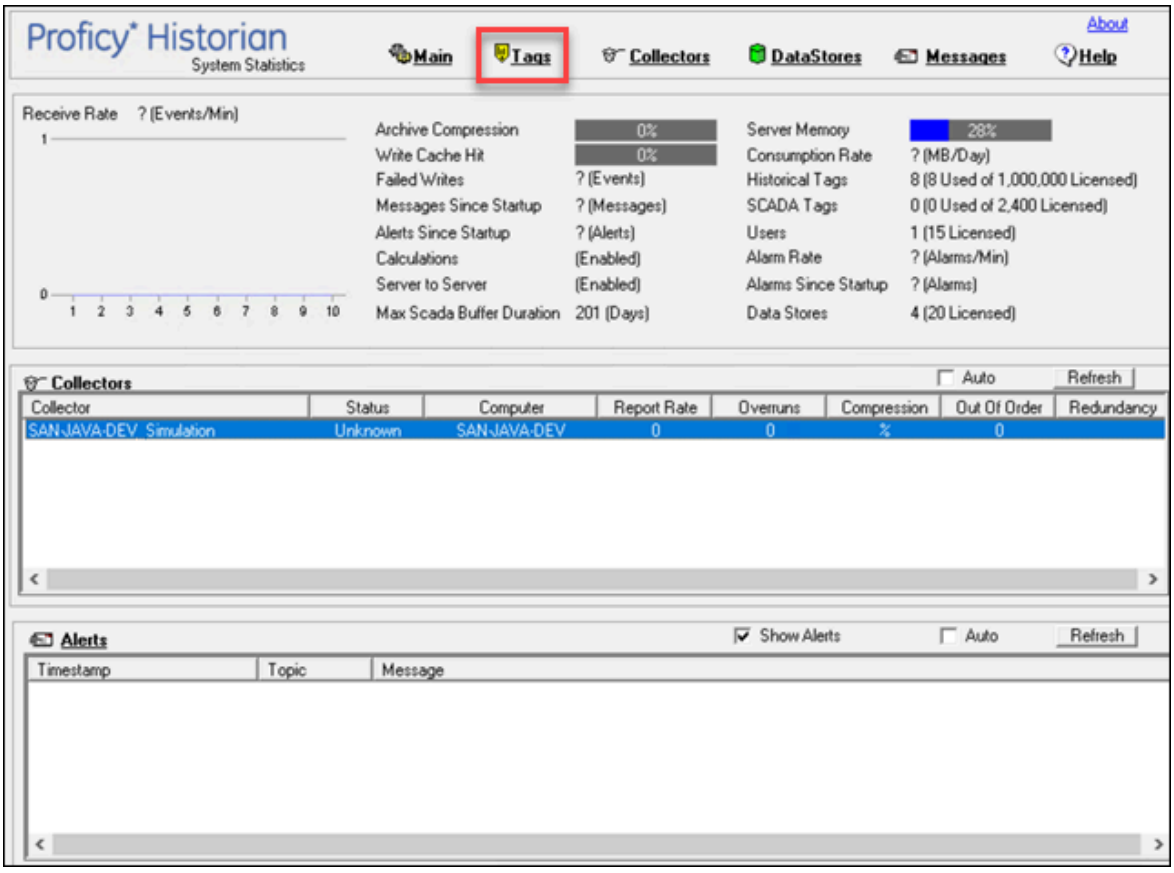
The following conditions apply when working with a UDT:

- You must have appropriate security permissions to create, modify, and delete a UDT. The type can have its own Administrator security group.
- You cannot create an array tag that uses a UDT.
- UDTs cannot have fields of Scaled or FixedString data types.
- Scaling, collector compression, and archive compression do not apply to UDT tags.
- You cannot associate an enumerated set with a UDT tag.
- A UDT supports maximum 100 fields.

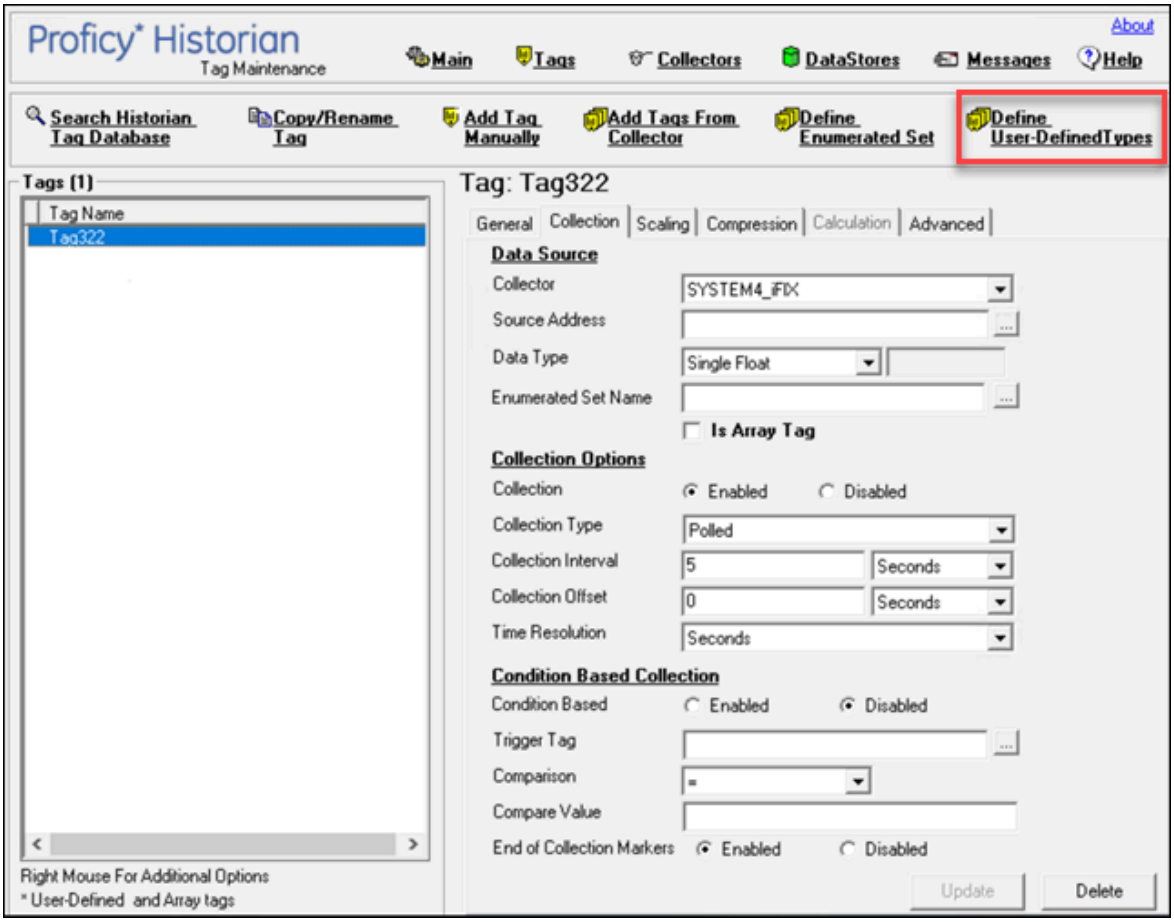
## Manage User-Defined Data Types

You can create a user-defined data type (UDT) *(on page 719)* and assign it to multiple tags. A UDT can have up to 100 fields and must have at least one field.

1. Access Historian Administrator *(on page 823)*.
2. Select **Tags**.




3. Select **Define User-Defined Types**.




The **Define User-Defined Types** window appears.

4. To create a UDT:

    a. Select **Create New Type**.

    b. Enter values as described in the following table.

| Field | Description |
|---|---|
| **Type Name** | Enter a unique name for the UDT. |
| **Description** | Enter a description for the UDT. |
| **Store Individual Quality** | Select the check box to store field-level quality. If this check box is cleared, the data sample will have a single quality similar to how an array tag works. Storing individual qualities consumes more disk space. |

| Field | Description |
|---|---|
| **Administer Group** | Specify the Windows Security Group that you want to assign to the UDT. |

c. Add at least one **Field** to the UDT.

d. Select **Save Type**.
The UDT is created.

5. To add a field to a UDT:

a. Select **New Field**.

b. Enter values as described in the following table.

| Field | Description |
|---|---|
| **Field Name** | Enter a name for the field. |
| **Master Field** | Select this check box if this field is a master field. Only one field can be the Master Field in a UDT. |
| **Field Description** | Enter a description for the field. |
| **Field Data Type** | Specify the data type of the field. |

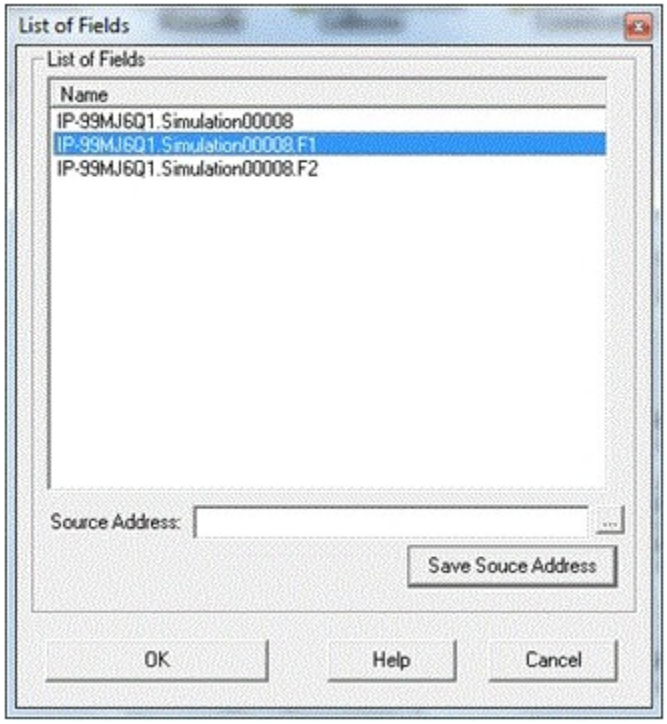
c. Select **Save Field**.
The field is added to the UDT.

6. To delete a field from a UDT:
   a. Select the field that you want to delete.
   b. Select **Delete Field**.
7. To delete a UDT, right-click the UDT that you want to delete, and then select **Delete Selected Type**.

> **Note:**
> You cannot delete a UDT if there are tags still using it.

8. To set a source address for a field:

a. Select the field for which you want to set the source address.



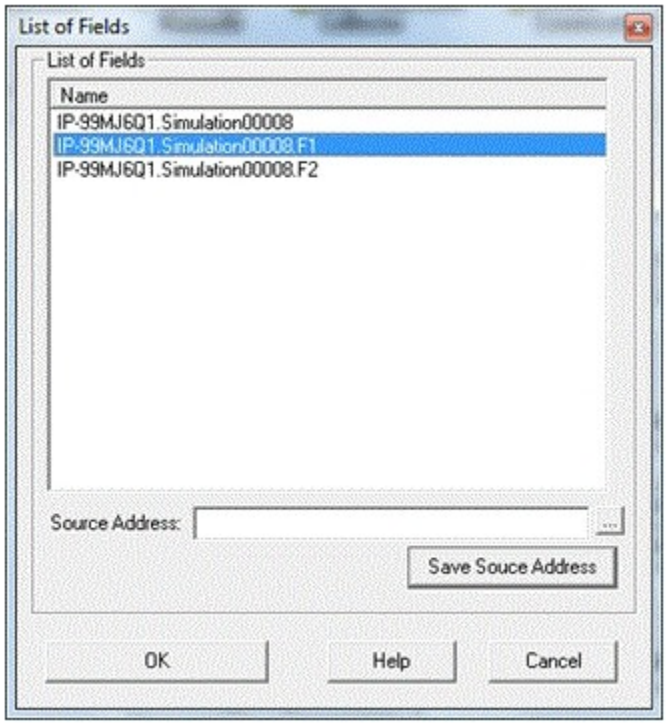

b. Browse and select the tag that you want to assign, and then select **Save Source Address > OK**.

The source address for the multifield tag is set.

9. To view the last 10 values of a tag:

a. Right-click the tag (a tag indicated with *), and select **Last 10 values**.

The **List of Fields** window appears.




b. Select the field from the list, and then select **OK**.
   The last ten values of the field appear.

10. To view the trend chart of a tag:

    a. Right-click the tag, and then select **Trend**.
       The trend chart of the tag values appears.

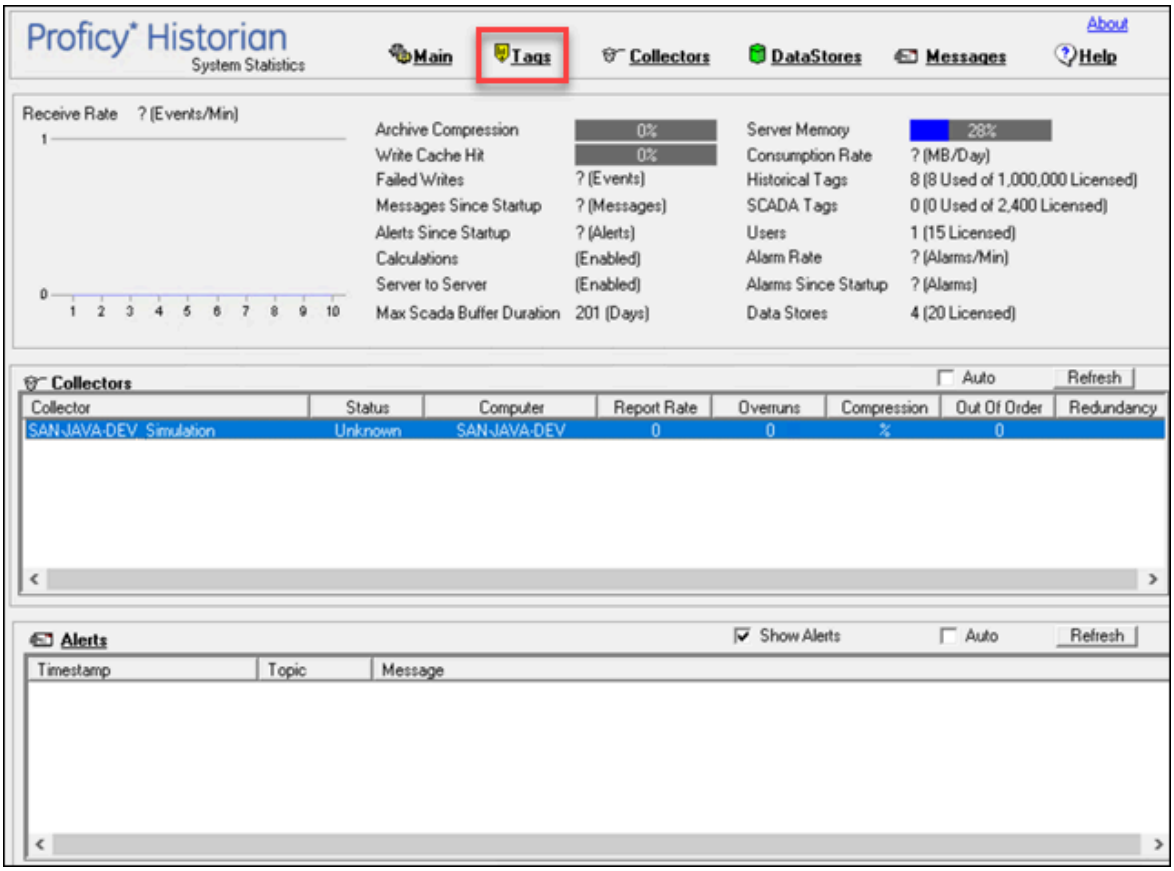
    b. If needed, modify the time period and other parameters by selecting **Criteria**. In the **criteria string** field, you can enter the sampling mode, calculation mode, and/or query modifiers. Query modifiers are used to specify various ways of retrieving data from Historian. Sampling modes specified with criteria strings takes precedence over any mode specified in the Sampling field.
       Request raw data with only good quality by specifying the criteria string: `RAWBYTIME#ONLYGOOD`

    c. Select **OK**.
       The trend chart is plotted again with the new criteria. You can scroll back and forth on the x-axis time scale by selecting on the single and double left and right arrows at the bottom of the page.

## Assign/Remove Tags from a User-Defined Data Type

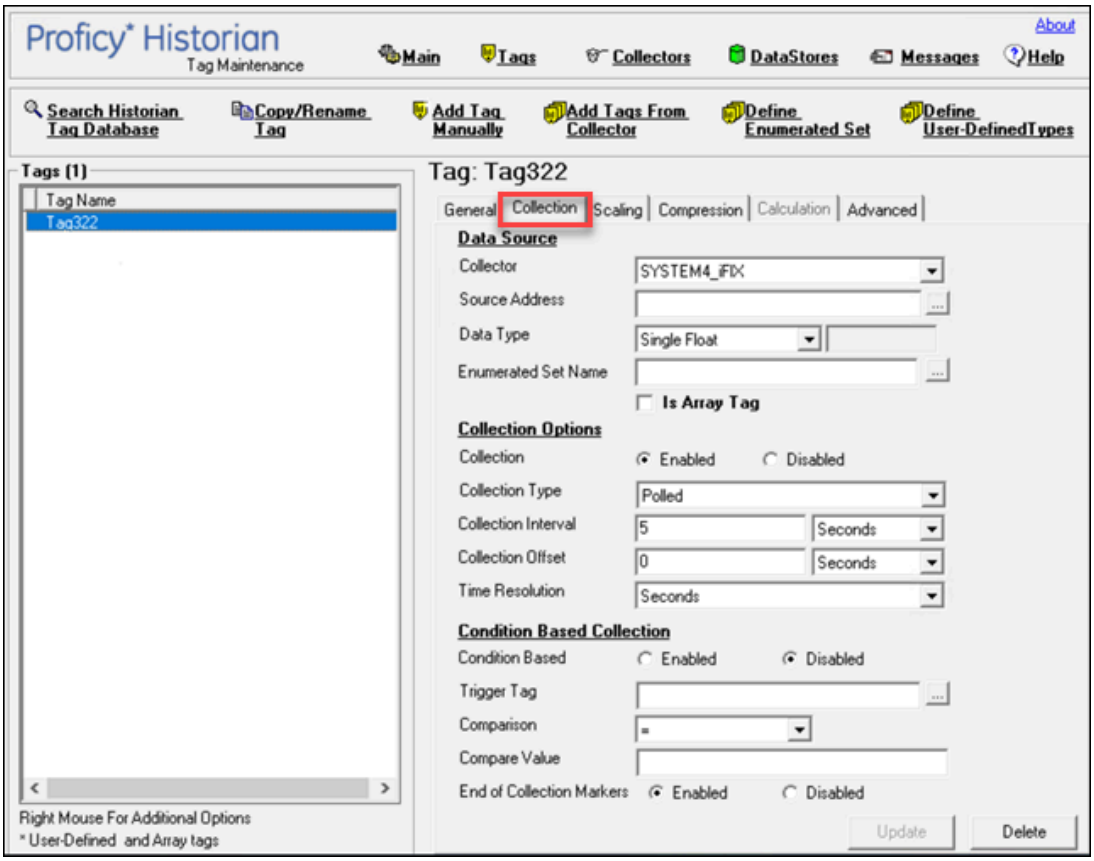
There are two ways to assign tags to a user-defined types. This topic describes both these ways. User Defined tags are indicated with '*' in the Tags section. If the desired type is not already created, you can create it while assigning it. You can select multiple tags from the Tags list and assign the same UDT to all of them at once.

1. Access Historian Administrator *(on page 823)*.
2. Select **Tags**.

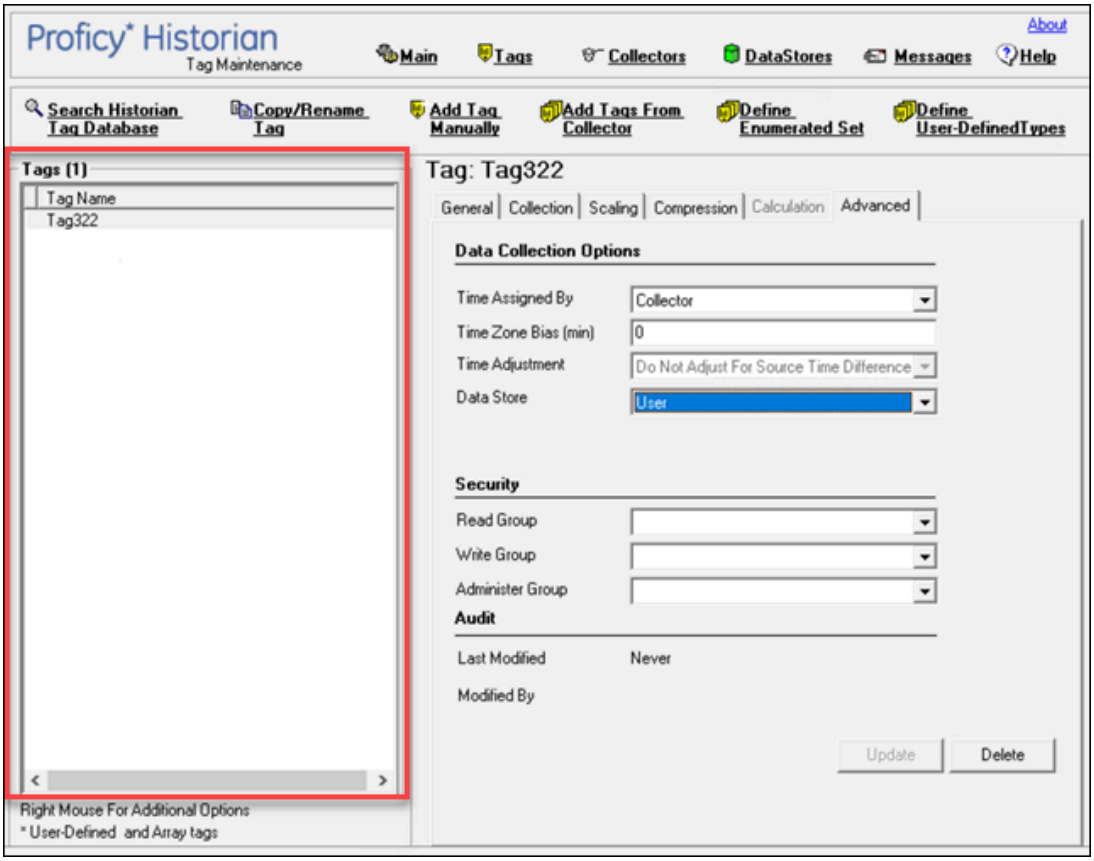



3. To assign a UDT to a tag from the **Collection** section:

a. Select **Collection**.




b. Select the tag for which you want to assign a UDT.




c. In the **Data Type** field, select **MultiField**.
The **User Defined Type** field appears.

d. In the **User Defined Type** field, browse and select the UDT that you want to associate with the tag, and then select **OK**.
If the type has not already been created, create a type now and then continue associating it with the tag. For more information, refer to Create a User Defined Type *(on page 942)*.

e. Select **Update**.
The UDT is assigned to the tag.

4. To assign a UDT to a tag from the **Add Tag Manually** window:

a. Select **Add Tag Manually**.
The **Add Tag Manually** window appears.

b. Enter values as described in the following table.

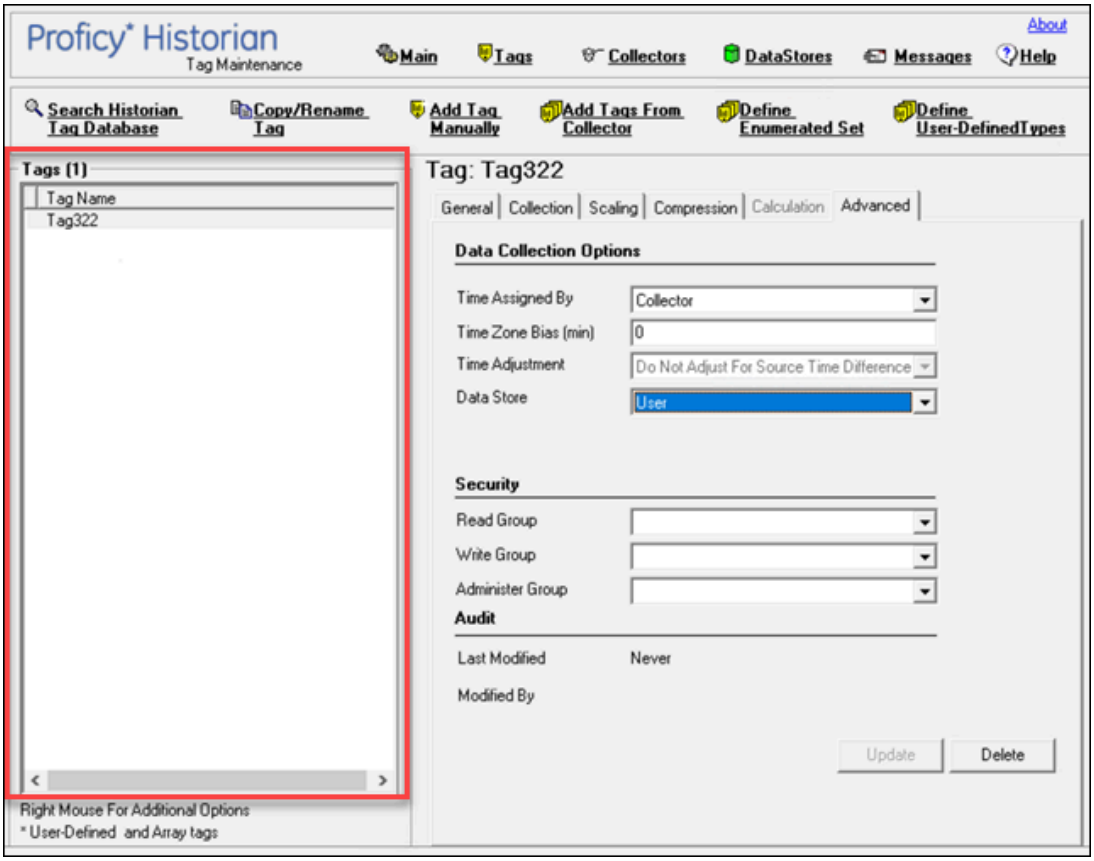
| Field | Description |
|---|---|
| **Collector Name** | Select the collector instance in which the tag has been added. |
| **Source Address** | Browse and select the source address of the tag. |
| **Data Type** | Select **MultiField**. |
| **User Defined Type** | Browse and select the UDT that you want to assign the tag. |
| **Time Resolution** | Select the time resolution. |

c. Select **OK**.

The UDT is assigned to the tag.

5. To remove a UDT from a tag:

a. Select the tag from which you want to remove the UDT.




b. Clear the value in the **User Defined Type** field.

c. In the **Data Type** field, select a data type.

d. Select **Update**.
The UDT is removed from the tag, and the new data type is assigned.

## Managing Enumerated Data Sets

An enumerated data set provides an enhanced way of displaying data. It enables you to retrieve numeric data as string state values. You can use the string values in reports, charts, etc.

An enumerated set contains several states. A state is the number-string value pair in a set. It contains a set of numeric values and their corresponding string values. You can define an enumerated set for a single value or a range of values. These state values are defined for data states stored in Data Archiver. Data is retrieved using the value of the state. You have to define state values within a set to assign enumerated values.

**Table 68. Example of a Single-Value Enumerated Set**

| State Name | State Value |
|---|---|
| Manual | 0 |
| Automatic | 1 |

**Table 69. Example of a Range-of-Values Enumerated Set:**

| State Name | State Value |
|---|---|
| ON | 0 to 100 |
| OFF | 101 to 200 |

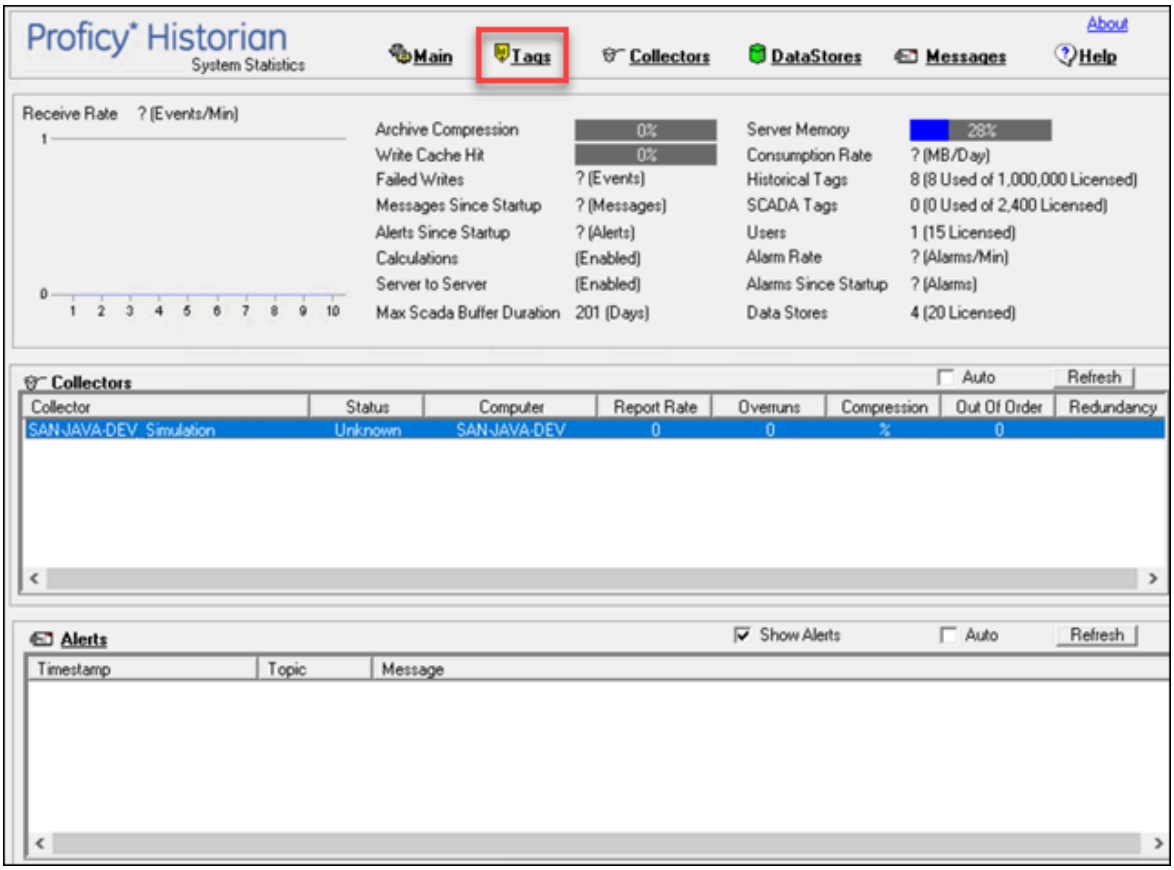
State names can be duplicated. If duplicated states exist, take precautions to avoid unpredictable results. For example, a tag is associated with an enumerated set defined as follows.

| State Name | State Value |
|---|---|
| 0 | Open |
| 1 | Close |
| 2 | Close |
| 2 | Open |

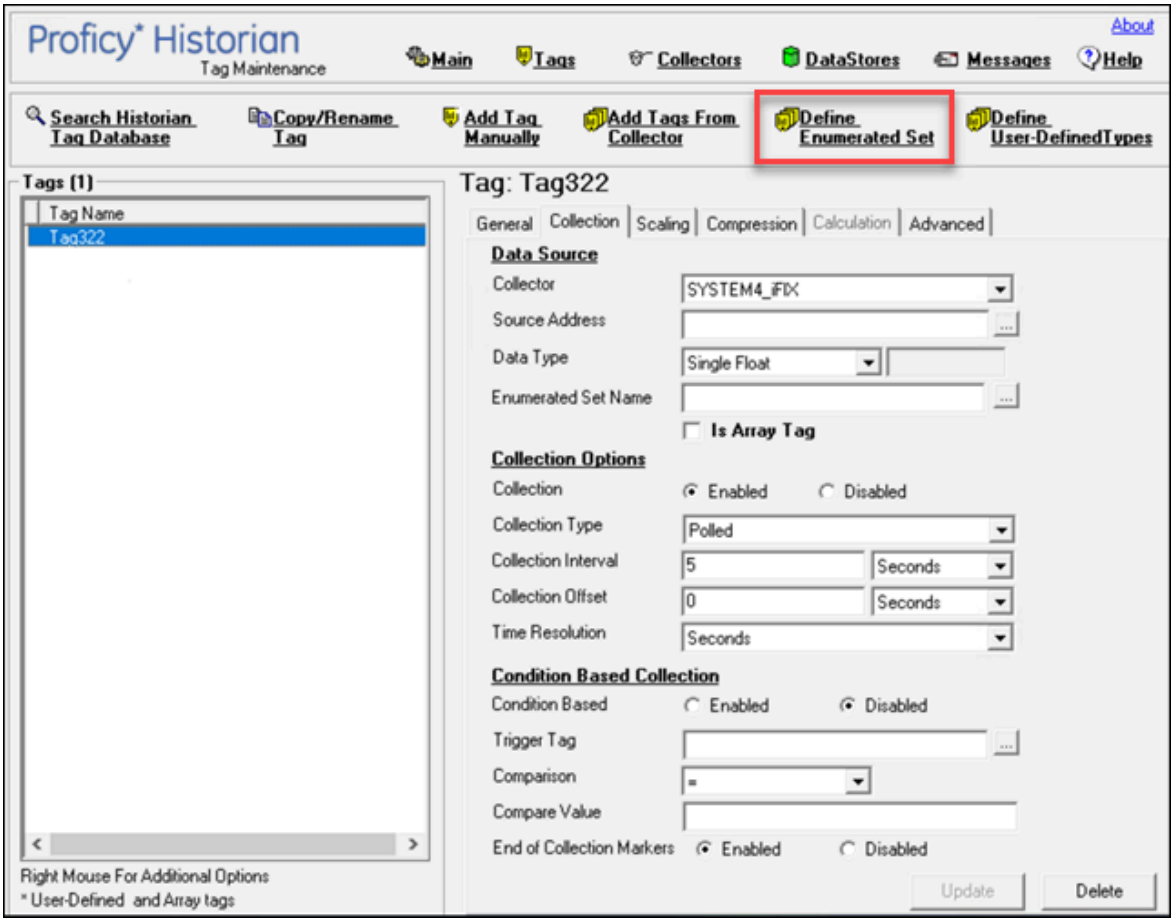
The server will return unpredictable results due to the State Name duplication for an input of 2.

**Note:**
You cannot assign an enumerated set to an array tag.

1. Access Historian Administrator *(on page 823)*.
2. Select **Tags**.




3. Select **Define Enumerated Set**.




The **Define Enumerated Set** window appears.

4. To create a set:

    a. Select **Create New Set**.

    b. Enter values as described in the following table.

| Field | Description |
|---|---|
| **Set Name** | Enter a name for the set. |
| **Description** | Enter a description for the set. |
| **Enumerate By** | Specify whether you want to define a single value or a range of values.<br><br>A single value is best used with integer values because they match exactly. A range of values can be used with floating point val- |

| Field | Description |
| --- | --- |
| | ues because they may not match exactly due to rounding. |

c. Add a state.

d. Select **Save Set**.
The enumerated set is created.

5. To create a state:

a. In the **State** section, select **New State**.

b. In the **Enumerate by** field, specify whether you want to use a single value or a range of values.
If you select **Single Value**, the **State Name**, **State Value**, and **Description** fields appear. If you select **Range**, the **State Name**, **Start Range**, **End Range**, and **Description** fields appear.

c. Enter values in the respective fields.
Enter only numeric values in the **State Values** field; string values such as ON/OFF are not supported.

d. Select **Save to List**.
The state is created and added to the enumerated set.

6. To delete a state:

a. To delete a *single* state, select the state and select **Delete State**. To delete *all* the states, select **Select All** , then select **Delete State**.
A message appears, asking you to confirm that you want to delete the state.

b. Select **Yes**.
The state is deleted.

7. To delete a set:

a. Right-click the set that you want to delete, and then select **Delete Selected Set**.
A message appears, asking you to confirm that you want to delete the set.

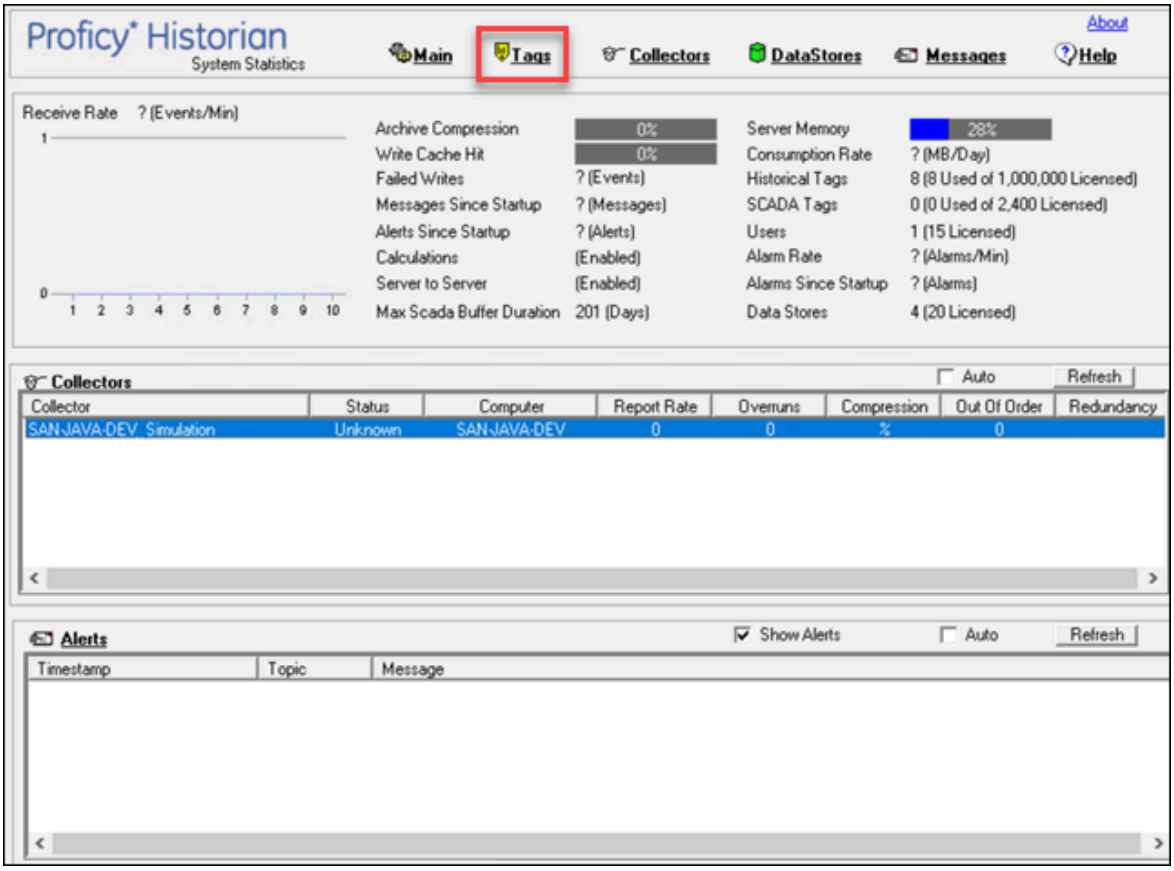
b. Select **Yes**.
The enumerated set is deleted.

## Assign/Remove Tags from Enumerated Sets

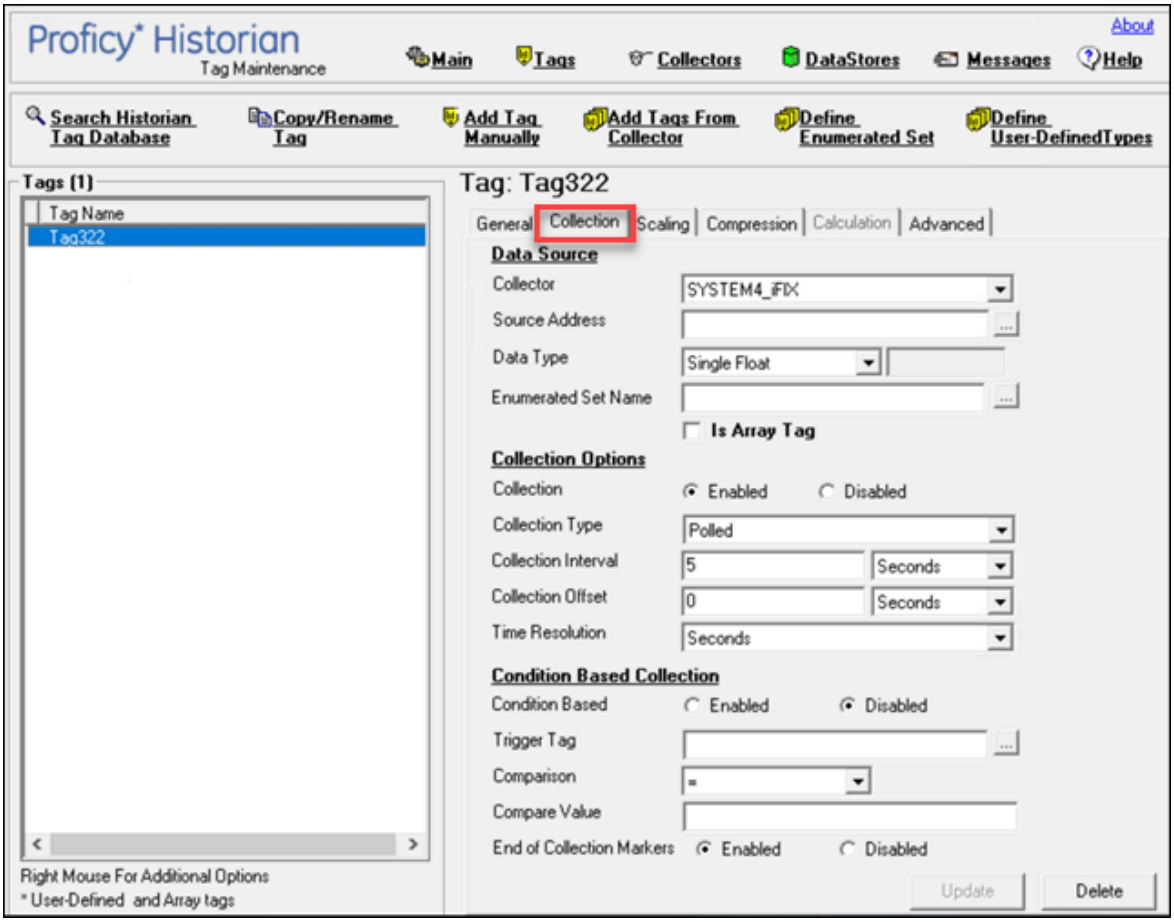
To view data for a tag in an enumerated state value format, assign a set to the tag. You can remove an assigned set from a tag to assign a new tag or not assign any tag at all.

**Note:**
You cannot assign enumerated sets to array tags.

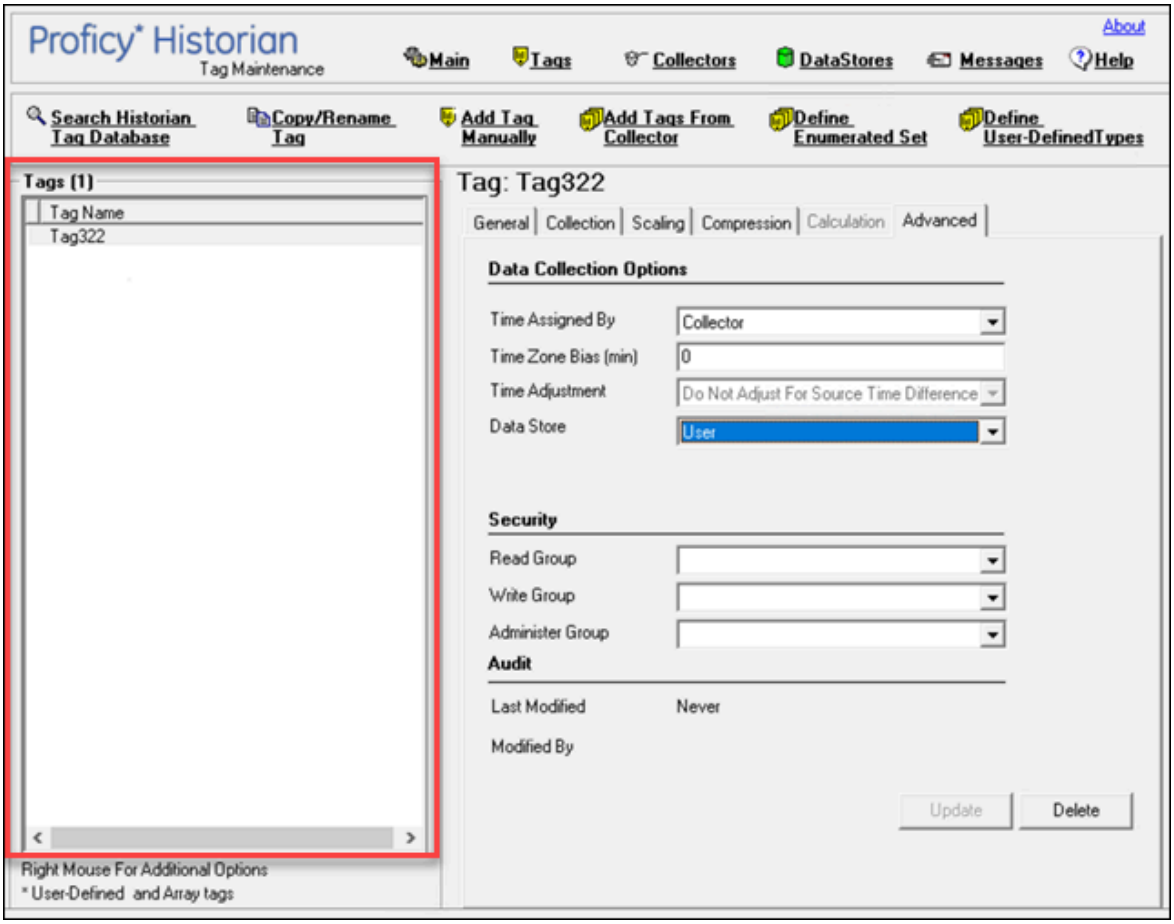
1. Access Historian Administrator *(on page 823)*.
2. Select **Tags**.




3. Select **Collection**.

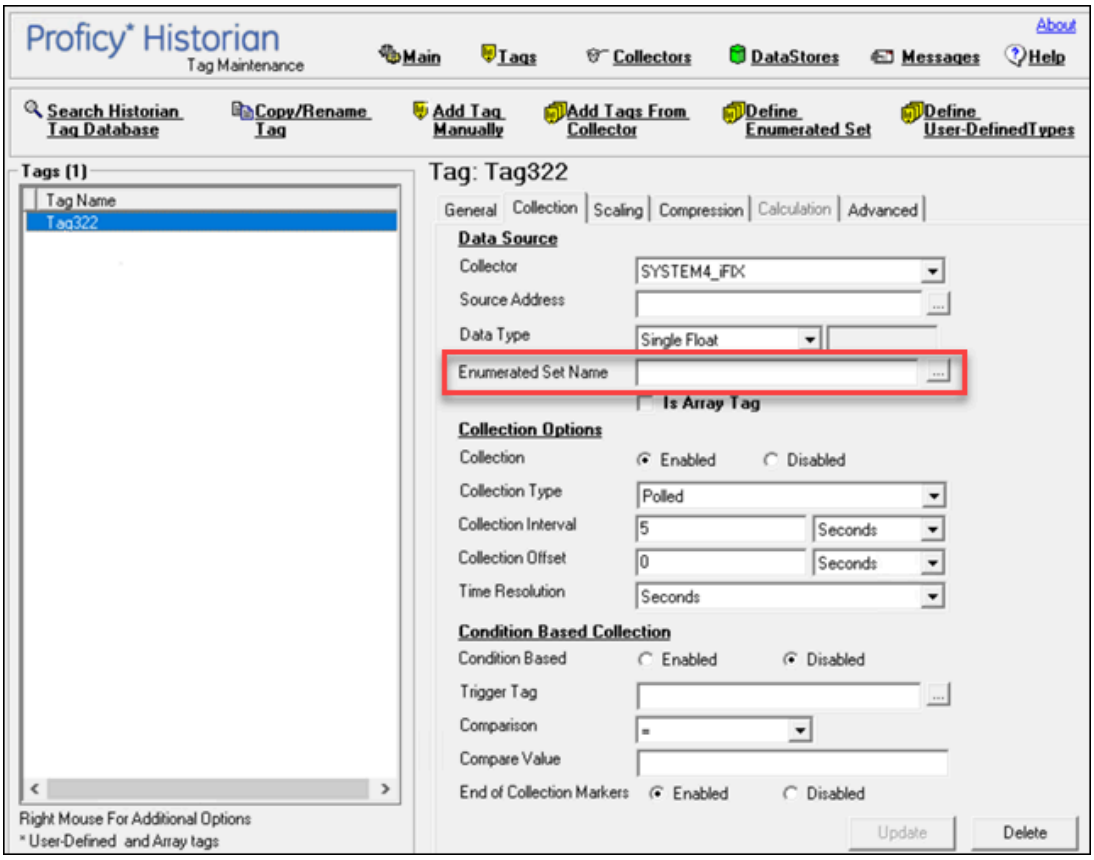



4. Select the tag that you want to assign/remove.




5. To assign a set to a tag:

a. In the **Enumerated Set Name** field, select ...



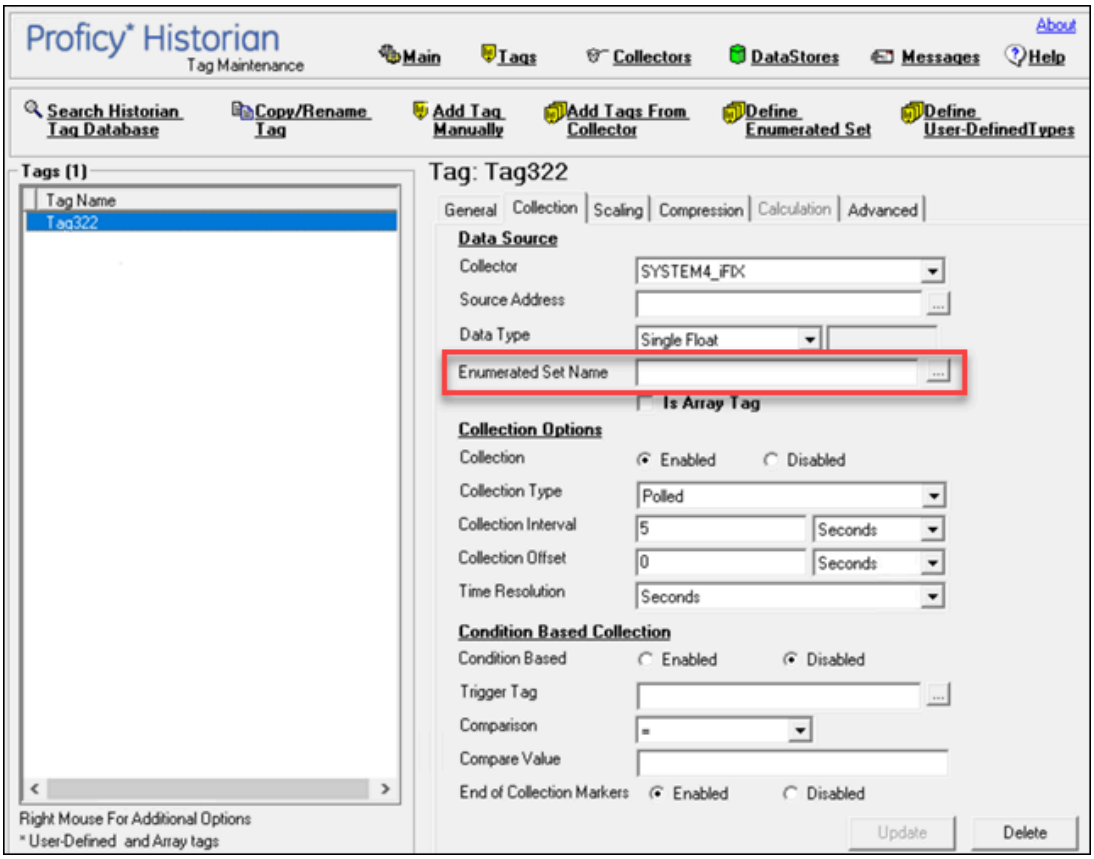

The **Define Enumerated Set** window appears.

b. From the **List of Enumerated Sets** section, select the enumerated set that you want to assign, select **Set**, and then select **OK**.
The **Enumerated Set** field is populated with the enumerated set that you have selected.

c. Select **Update**.
The set is now assigned to the tag and you can view the data in the enumerated state value format.

6. To remove an assigned set from a tag:

a. Clear the value in the **Enumerated Set Name** field.




b. Select **Update**.

The enumerated set is no longer assigned to the tag.

# Managing Collectors

## About Collectors

Collectors are used to collect data from various data sources and send it to Historian. For a list of collectors and their usage, refer to About Historian Data Collectors *(on page 1990)*.

After you install collectors and Remote Management Agent, the following artefacts will be available:

- **Executable files:** These files are required to add a collector instance.
- **Instances of the following collectors:**
  - The iFIX collector
  - The iFIX Alarms & Events collector

    - The OPC Classic Data Access collector for CIMPLICITY
    - The OPC Classic Alarms and Events collector for CIMPLICITY

  These instances will be created only if iFIX and/or CIMPLICITY are installed on the same machine as the collectors.
- **The Remote Collector Management agent:** Provides the ability to manage collectors remotely.

To send data using a collector, you must:

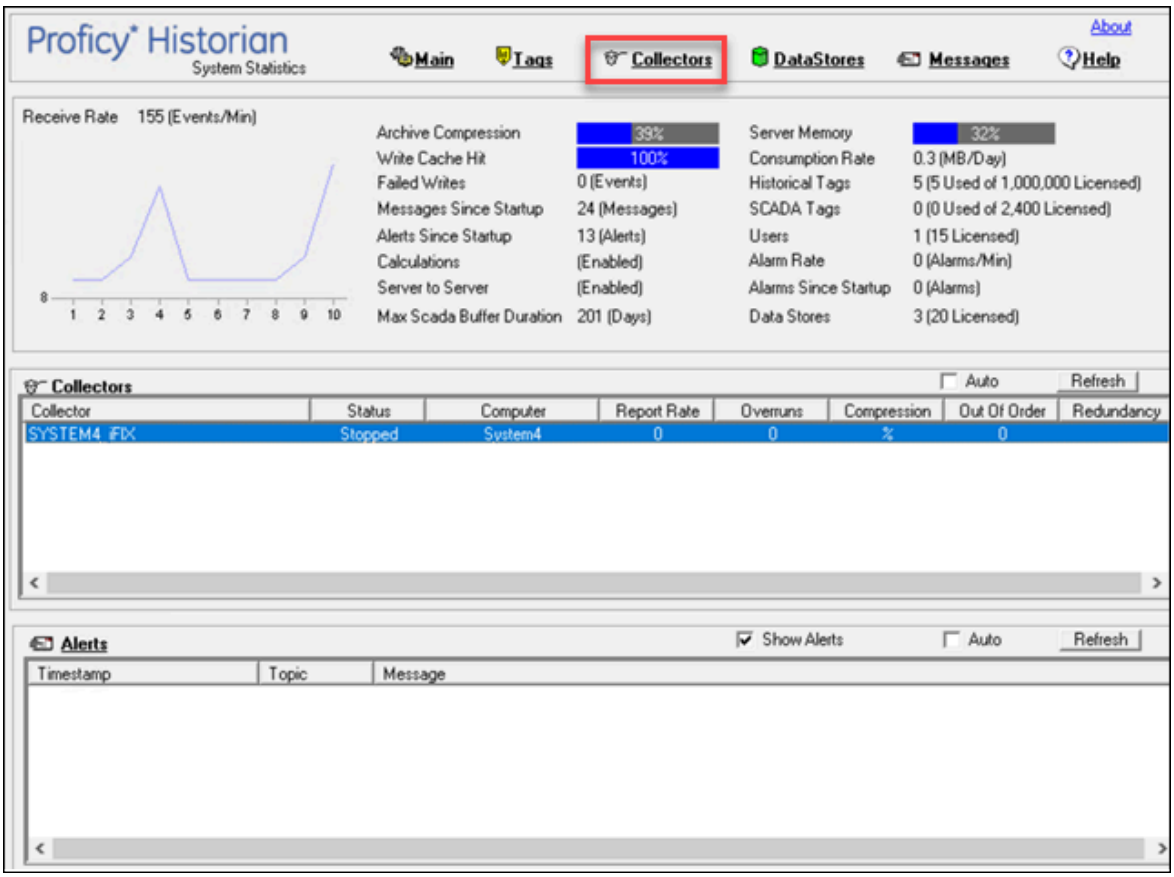
1. Install collectors *(on page 142)*.

   You can install collectors on multiple Windows machines. These machines can be on-premises or on a virtual private cloud (VPC).

2. Create a collector instance.

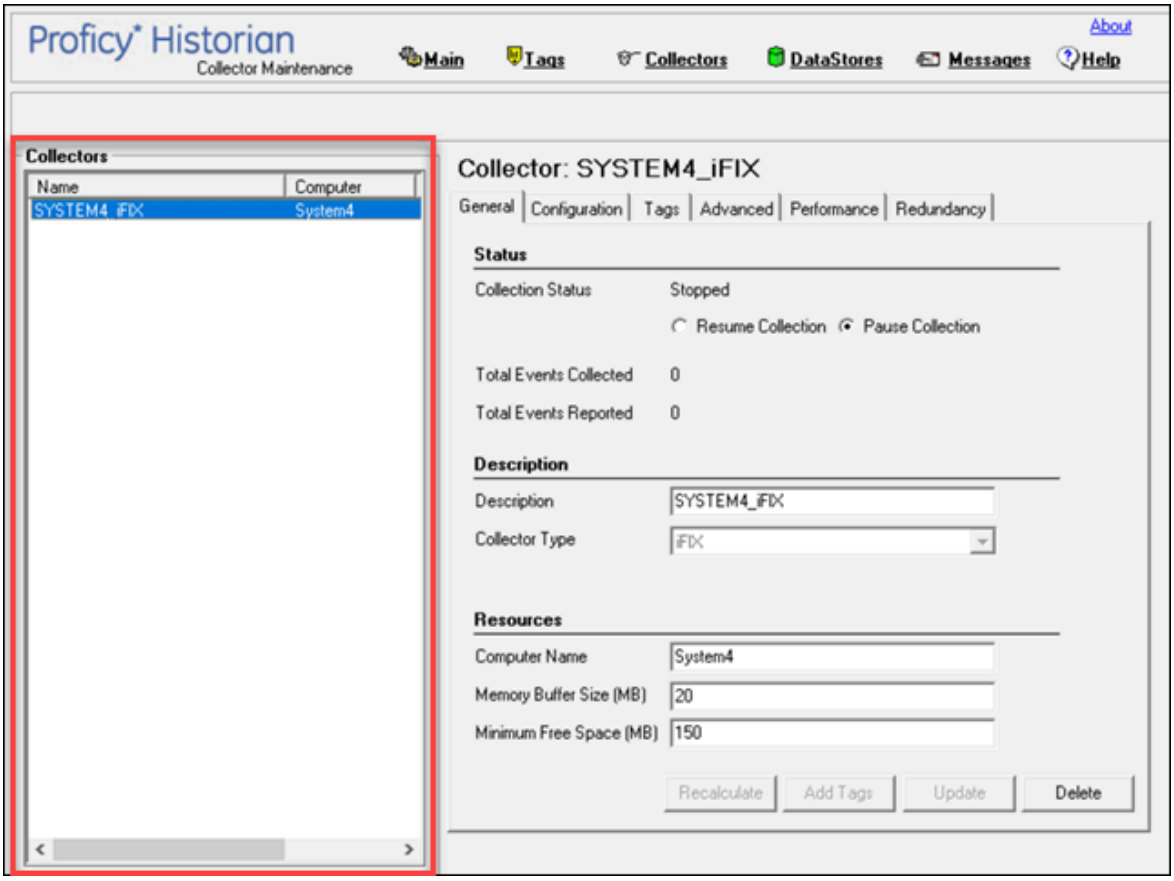
## Access/Modify a Collector

1. Access Historian Administrator *(on page 823)*.
2. Select **Collectors**.



A list of collectors appears.

3. Select the collector whose details you want to access/modify.




4. As needed, modify values as described in the following tables, and then select **Update**.

**Table 70. The General Section**

| Field | Description |
|---|---|
| **Collection Status Field** | The current operating status of the collector. Contains one of the following values:<br>◦ **Running:** The collector is operating and collecting data.<br>◦ **Stopped:** The collector is in pause mode and not collecting data.<br>◦ **Unknown:** The status information about the collector is unavailable at present, perhaps as a result of a lost connection between collector and server or because the collector was shut down improperly. |

| Field | Description |
|---|---|
| | You can specify whether you want to pause or resume data collection. |
| **Total Events Collected** | The total number of events collected from the data source by the collector. |
| **Total Events Reported** | The total number of events reported to the archive from the collector. This number may not match the **Total Events Collected** value due to collector compression. |
| **Description** | The name of the collector. |
| **Collector Type** | The type of the collector. |
| **Computer Name** | The machine name of the computer on which the collector is installed. |
| **Memory Buffer Size (MB)** | The size of the memory buffer currently assigned to the store-and-forward function. The memory buffer stores data during short-term or momentary interruptions of the server connection; the disk buffer handles long duration outages. To estimate the size you need for this buffer, you need to know how fast the collector is trying to send data to the server and how long the server connection is likely to be down. With those values and a safety margin, you can compute the required size of the buffer.<br><br>**Note:**<br>If you enter a new value for this parameter, the change is effective the next time you restart the collector. |
| **Minimum Free Space (MB)** | The minimum free disk space that must be available on the computer. If the minimum space required is not available when the collector starts, the collector will shut down. |

**Table 71. The Tags Section**

| Field | Description |
|---|---|
| **Add Prefix to Tag** | A prefix that is automatically added to all tag names when you add tags. |
| **Naming Convention** | When digital sets are automatically added via the OSI PI Collector: |

| Field | Description |
|---|---|
| | ◦ The enumerated set added to Historian is prefixed using the value configured here. For Example, if "IP- 4LJPD02." has been configured as the tag prefix value, then the digital set with name "SYSTEMSET" in PI Historian will be added as an enumerated set with name "IP-4LJPD02.SYSTEMSET" in Historian.<br>◦ If the value of the tag prefix is modified, then a new set of enumerated sets prefixed with the updated value will be added to Historian server. The enumerated sets prefixed with the old value will not be deleted and will not receive any later updates. |
| **Collection Interval** | The time required to complete a poll of a given tag on the collector. It is also used in unsolicited collection. In effect, it specifies how frequently data can be read from a tag. The collection interval can be individually configured for each tag.<br><br>**Note:**<br>To avoid collecting repeat values with the OPC collector when using device timestamps, specify a collection interval that is greater than the OPC server update rate. |
| **Collection Type** | Indicates whether this collector is configured for polled or unsolicited data collection. |
| **Time Assigned By** | Indicates whether the timestamp for the data value is provided by the collector or the data source. |
| **Collector Compression** | Indicates whether collector compression *(on page 660)* is enabled as a default setting. This option is overridden by tag-level settings. |
| **Deadband** | The default setting of the collector compression deadband in absolute or percentage range values. |
| **Compression Timeout** | The default setting for the collector compression time-out for tags added through the **Add Multiple Tags From Collector** window. You must enable collector compression to use this field. |
| **Spike Logic Control** | Indicates whether incoming data samples for spikes are captured in tag values. If spike logic is enabled, a sample of equal value to the previous- |

| Field | Description |
|---|---|
| | ly archived sample is inserted into the archive in front of the spike value. For more information, refer to Enable Spike Logic *(on page 970)*. |

**Table 72. The Advanced Section**

| Field | Description |
|---|---|
| **On-line Tag Configuration Changes** | Indicates whether you can make changes to tags without having to restart the collector. If you disable this option, any changes you make to tags do not affect collection until you restart the collector. |
| **Browse Source Address Space** | Indicates whether you want the collector to respond to requests to browse the tags in the source. You may sometimes want to disable this feature to reduce processing load on the collector. |
| **Synchronize Timestamps to Server Time** | Adjusts all outgoing data timestamps to match the server clock. This option is not applicable when you configure timestamps to be provided by the data source. Note that this does not change collector times to match the server time; it adds or subtracts an increment of time to compensate for the relative difference between the clocks of the server and collector, independent of time zone or day light savings time (DST) differences. If the collector system clock is greater than 15 minutes ahead of the archiver system clock, and the **Synchronize Timestamps to Server** option is disabled, data will not be written to the archive. |
| **Source/Device Timestamps** | The time source for the timestamps. This field applies only if you are using source timestamps. The collector uses this field to determine whether the timestamps coming from the data source are in local machine time or UTC. |
| **Delay Collection at Startup** | The number of seconds to delay collection on startup (after loading its tag configuration). |
| **Rate Output Address** | Address in the source database into which the collector writes the current value of the events/minute output, letting an operator or the HMI/SCADA application know the performance of the collector. This should be connected to a writable analog field. The value is written once a minute. |

| Field | Description |
|---|---|
| | For an iFIX collector, use an iFIX tag for the output address. Enter the address in the following format: NODE.TAG.FIELD (for example, MyNode.MySIM_AO.F_CV).<br><br>For an OPC collector, use a writable OPC address in the server. Refer to your OPC documentation for more information. |
| **Status Output Address** | Address in the source database into which the collector writes the current value of the collector status, letting an operator or the HMI/SCADA application know the current status of the collector.<br><br>This address should be connected to a writable text field of at least 8 characters. This value is only updated upon a change in status of the collector.<br><br>For an iFIX collector, use TX tag for the output address. Enter the address in the following format: NODE.TAG.FIELD (for example, MyNode.MyCollector_TX.A_CV).<br><br>For an OPC collector, use an OPC address in the server. Refer to your OPC documentation for more information. |
| Heartbeat Output Address | Address in the source database into which the collector writes the heartbeat signal output. This address should be connected to a writable analog field.<br><br>For an iFIX collector, use an iFIX tag for the output address. Enter the address in the following format: NODE.TAG.FIELD (for example, MyNode.MyCollector_AO.F_CV).<br><br>For an OPC collector, use the OPC address in the server. Refer to your OPC documentation for more information.<br><br>The data collector writes the value of 1 to this location every 60 seconds while it is running. You can program the iFIX database to generate an alarm if the Heartbeat Output Address is not written once every 60 seconds, notifying you that the data collector has stopped. |

**Table 73. The Performance Section**

| Section | Description |
|---|---|
| **Report Rate** | Displays the average rate at which data is coming into the server from the collector. This is a general indicator of load on the Historian collector. Since this chart displays a slow trend of compressed data, it may not always match the instantaneous value of report rate. |
| **Compression** | Displays the effectiveness of collector compression. If the chart displays a low current value, you can widen the compression deadbands to pass fewer values and increase the effect of compression. |
| **Overruns** | Displays the value at which data overruns are occurring. This value is calculated by the following equation:<br>`OVERRUN_PCT = OVERRUNS / ( OVERRUNS + TOTAL_EVENTS_COLLECTED )`<br>Overruns are a count of the total number of data events not collected. In normal operation and under normal conditions, the current value should always be zero. If the current value is not zero, which indicates that data is being lost, you should take steps to reduce peak load on the system by increasing the collection interval. |

**Table 74. The Redundancy Section**

| Field | Description |
|---|---|
| **Redundant Collector** | Indicates that this is a redundant collector. |
| **Backup For** | The primary collector.<br>**Note:**<br>This configuration will be preserved if you disable collector redundancy. This allows you to temporarily take a redundant collector offline without losing its configuration. |
| **Backed Up By** | The name of the collector providing redundancy for the primary collector. |

| Field | Description |
|---|---|
| **Collector Status (Status)** | The current status of the primary collector. |
| **Redundancy Status** | The current redundancy status of the collector. If a secondary collector has been activated, this field will appear. |
| **Make Active Collector Now** | Select this button to bring the selected collector online immediately. This is useful for testing, or in situations where the primary collector must be brought offline quickly. |
| **Collector Status (Failover Triggers)** | Indicates whether the collector will failover if the status changes to Unknown. |
| **Watchdog Tag** | The tag to use to determine the status of the collector. If the watchdog tag meets any of the conditions specified below, the secondary collector will be brought online to replace it. |
| **Failover on Bad Quality** | Indicates whether the secondary collector must be promoted when a data sample from the watchdog tag is received with bad quality. Failover happens on every write of a bad data sample to the watchdog, not just on the transition from good to bad quality. |
| **Failover When Value Transitions from Zero** | Indicates whether the secondary collector is promoted when a data sample from the watchdog tag with a non-zero value is received from the primary collector. Failure happens every time when a non-zero value is received, not just when the value promotes from zero to non-zero value. |
| **Failover When No Value Changes for X Seconds** | Indicates that the secondary collector is promoted when no data value changes have been received within the time period specified. This could be tied into a heartbeat status indicator. The value is checked every 5 seconds. To prevent failure, there must be a value change. |

**The Actions Buttons**

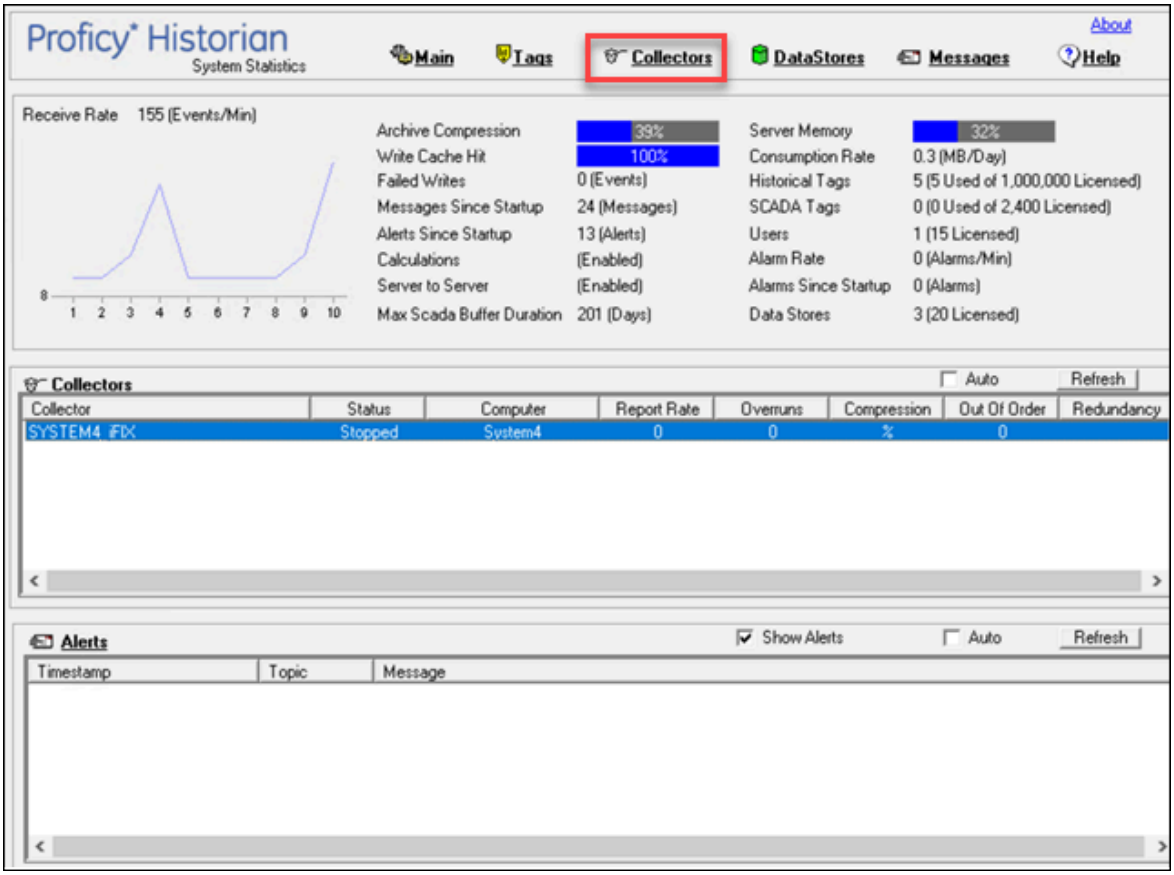
All tabs in the Collector Maintenance screen contain action buttons. Click a button to perform the action indicated by the name.

| Button | Function |
|---|---|
| **Recalculate** | Recalculate collector data for a specified period. |
| **Add Tags** | Browse and add tags from this collector to the archiver. |
| **Update** | Apply all parameter changes you have made on any tabs in this screen. To cancel changes and return to the original values or settings, open a different screen and then return to the Collector Maintenance screen. |
| **Delete** | Delete the selected collector. You can choose whether you want to delete only the collector or the collector and its tags. |

## Delete a Collector

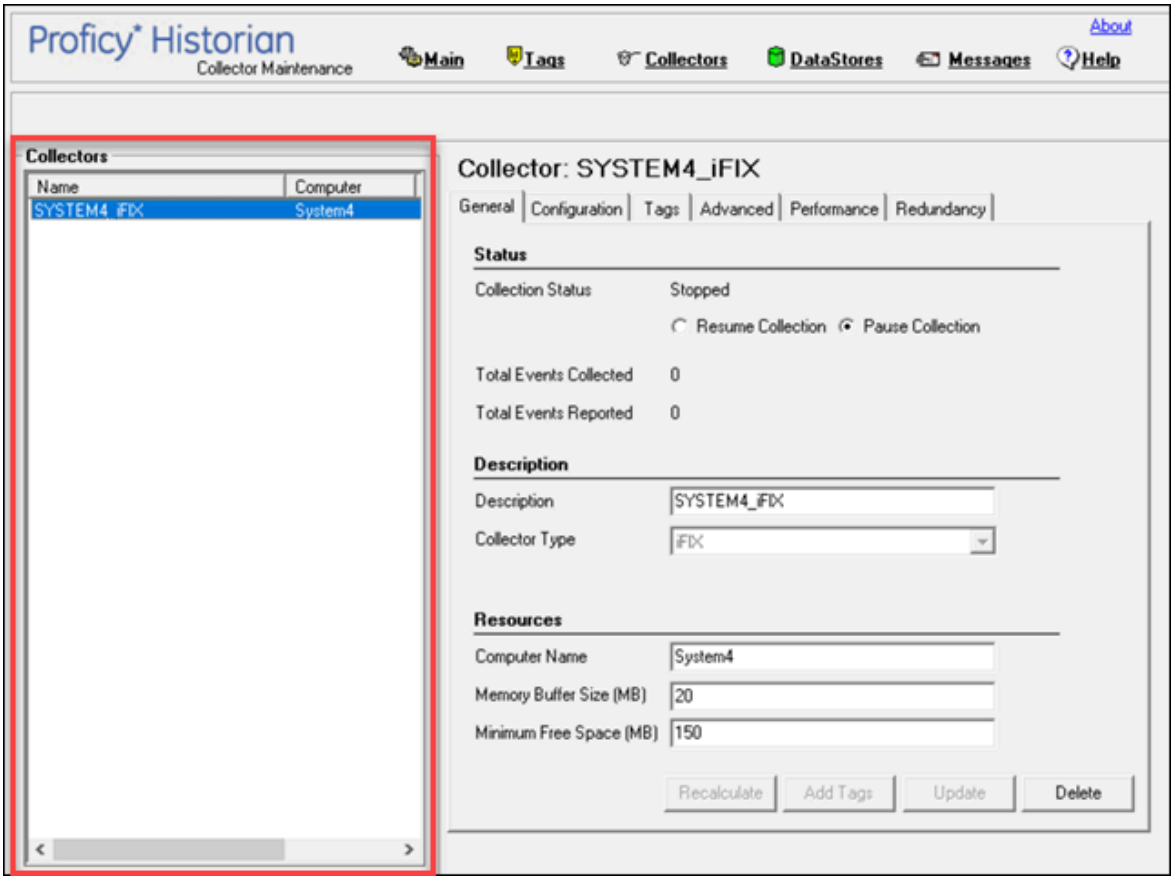
When you delete a collector, all of its tags are deleted from the Historian database.

1. Access Historian Administrator *(on page 823)*.
2. Select **Collectors**.




A list of collectors appears.

3. Select the collector whose details you want to delete.



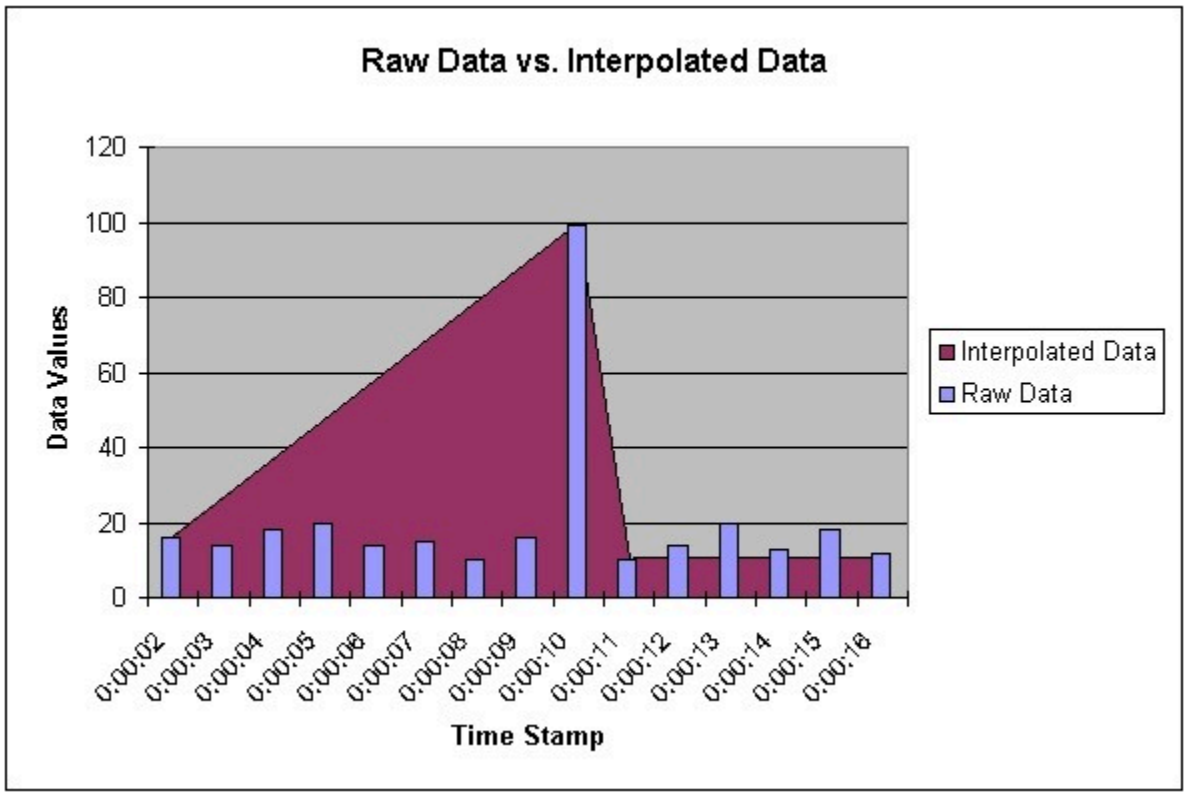

4. Select **Delete**.
   A message appears, asking you to confirm that you want to delete the collector.
5. Select **Yes**.
   The collector is deleted.

## Enable Spike Logic

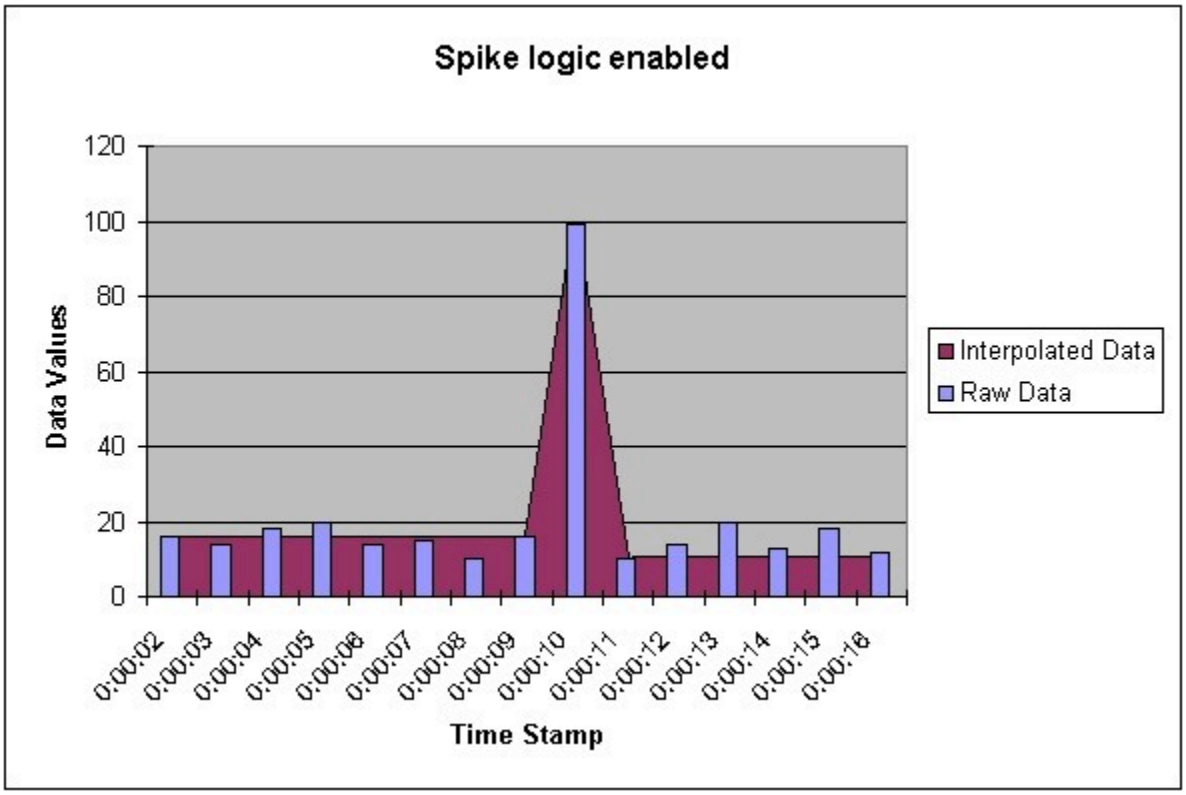
When compression is enabled in the Historian archive, only the first instance in a series of data falling within a deadband range will be collected to the Historian archive. When that data is plotted using interpolation, false values are inserted into the chart to create a smooth trend between intervals in a given time period. In most cases, interpolation gives a reasonable portrayal of the actual data for a given time period.

Unfortunately, in the event of a spike in data values, an unrealistic set of samples is created when the data is plotted. Instead of showing the results of compression (the same values over a series of intervals), a rising or falling slope is created in the chart. This gives the impression that values for a given time stamp are higher or lower than they actually were. The figure below shows the difference between the raw

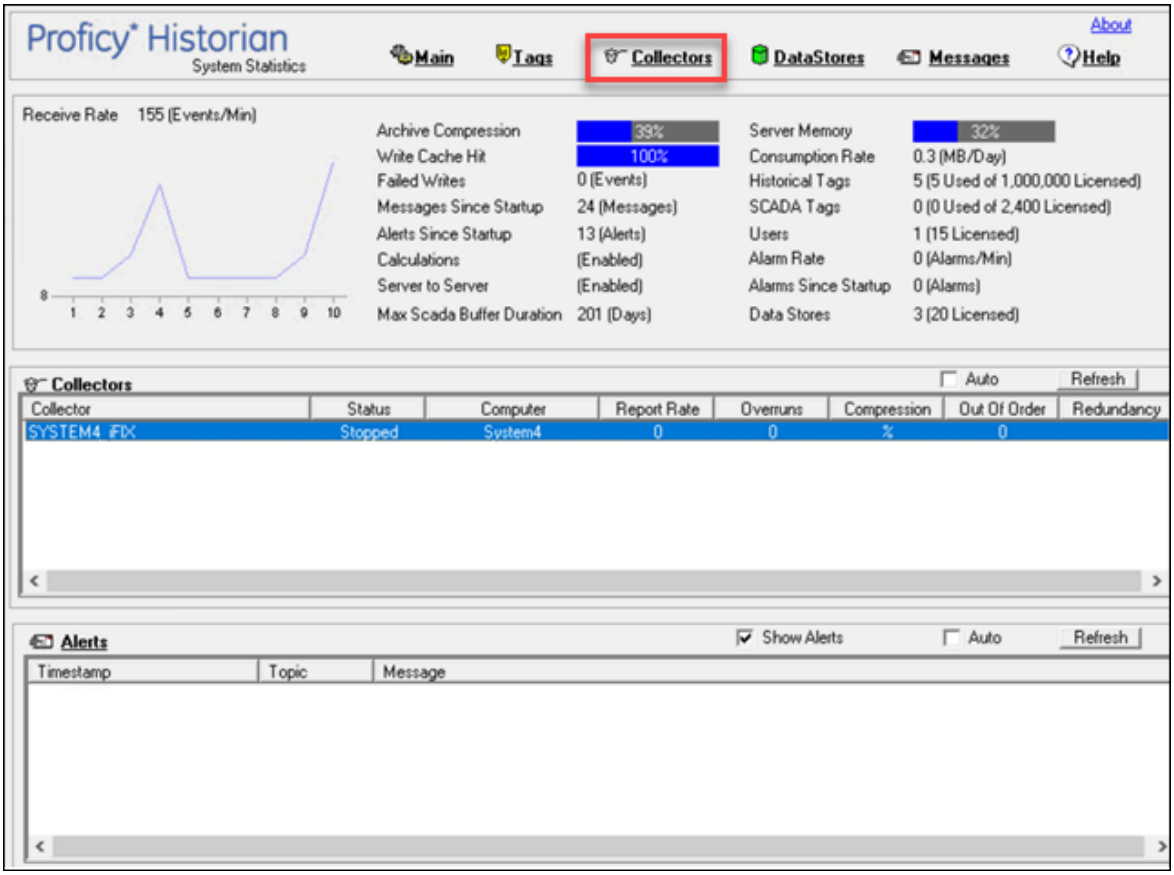
data for a series of samples, and how the samples would be plotted if data compression were enabled, assuming all values between 10 and 20 are in the deadband range.




Spike logic monitors incoming data samples for spikes in a tag's values. If spike logic is enabled, a sample of equal value to the previously archived sample is inserted into the archive in front of the spike value. The time stamp of the inserted value is determined by your polling interval. If samples are collected at one-second intervals, the inserted sample's time stamp will be one second before the spike. This helps identify the spike, and retains a more accurate picture of the data leading up to it, as shown in the following image.

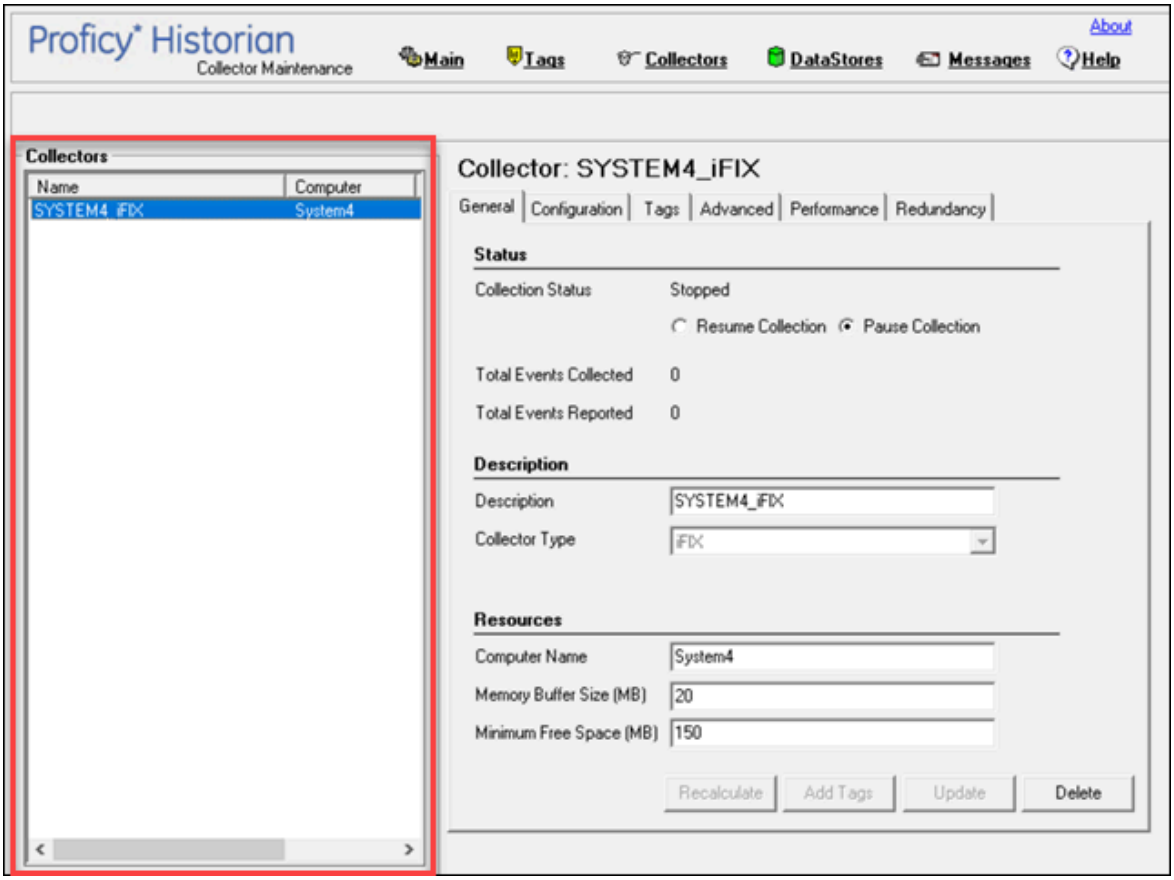



1. Access Historian Administrator *(on page 823)*.
2. Select **Collectors**.

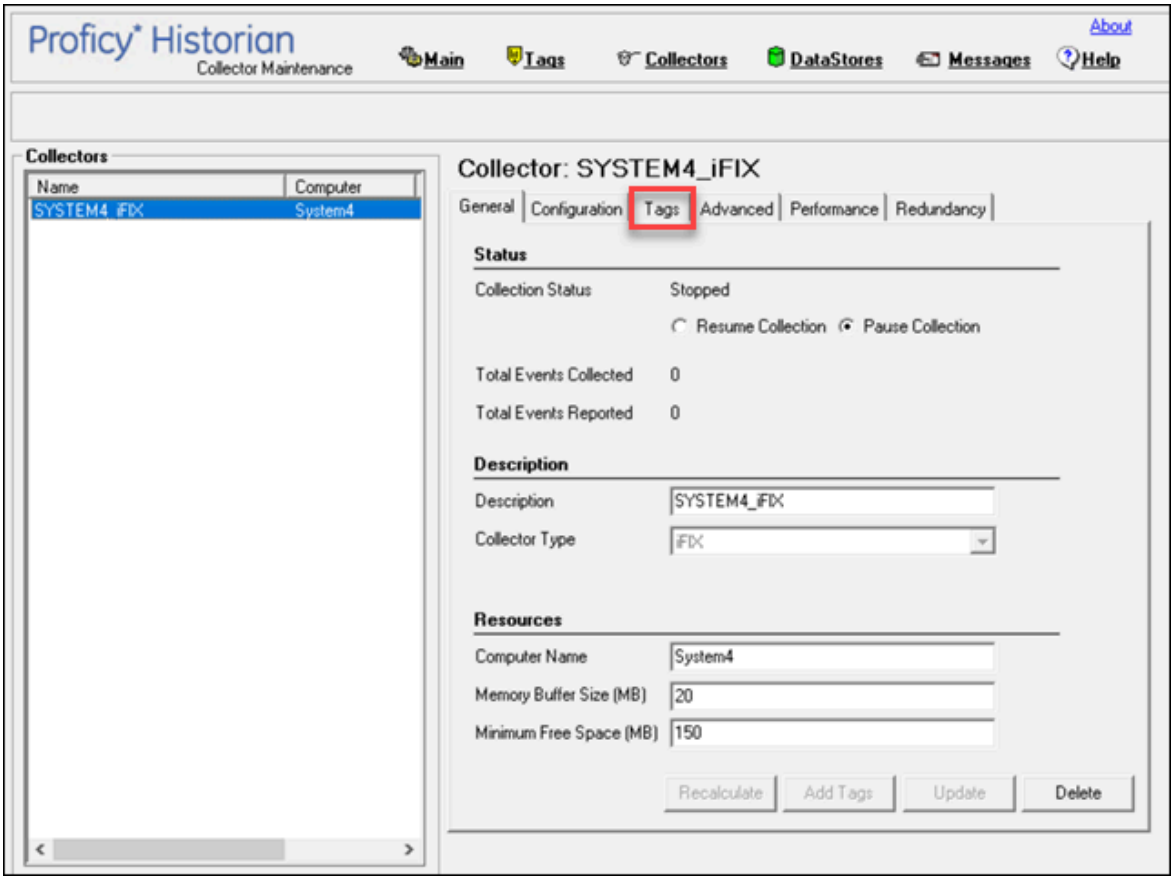



A list of collectors appears.

3. Select the collector to which you want to apply spike logic.




4. Select **Tags**.




5. Under **Default Compression**, in the **Spike Logic Control** field, select **Enabled**, and then select one of the following options:
    - **Multiplier**: Specifies how much larger a spike value must be than the deadband range before spike logic will be invoked. For example, if you enter 3, and the deadband percentage was set to 5%, spike logic will not be invoked until the difference between the spike value and the previously archived data point is 15% of the EGU range.
    - **Interval**: Specifies how many samples must have been compressed before the spike logic will be invoked. For example, if you enter 4, and 6 values have been compressed since the last archived data sample, spike logic will be invoked.
6. Select **Update**.
   The spike logic is enabled.

## About Collector Redundancy

Collector redundancy decreases the likelihood of lost data due to software or hardware failures. It ensures that collection of your data remains uninterrupted. It uses two or more collectors that gather data from a single source.

**Note:**
Collector redundancy is not available and supported for the following collectors:

- File Collector.
- Calculation Collector.
- Python Collector.
- Server-to-Server Collector.
- Server-to-Server Distributor.
- OSI PI Collector.
- OSI PI Distributor.

All collectors in the group actively gather the same tags from a data source but only the *active* collector forwards its samples to the Historian server. The non-active (standby) collectors buffer their data against failover of the active collector. The Historian server actively monitors the health of the redundant collectors and will automatically switch to a backup if certain trigger conditions are met.

**Trigger conditions:** You can configure the following triggers:

- The collector is stopped.
- The collector sends bad quality data for a specified tag (called the watchdog tag).
- The values of the watchdog tag do not change for a specified duration.

If any of these trigger conditions are met, the primary collector goes into the standby mode, and the secondary collector becomes active. That means, data is then collected by the secondary collector. Similarly, if the secondary collector satisfies the trigger conditions, it goes into the standby mode, and the primary collector becomes active.

You can also manually trigger the failover.

For instructions on configuring collector redundancy, along with settings the triggers, refer to Configure Collector Redundancy *(on page 651)*.

**Synchronized Configuration:** The configuration for each redundant collector is managed by the Historian server. If you change the configuration on one redundant collector, all other collectors will be updated automatically with the new configuration.

**Note:**
Redundant collectors must be of the same collector type.

**Offline Collectors:** To reduce the possibility of lost data, a collector will immediately send its buffered data to the Historian archiver when brought online. The Historian server will ignore any data that is already collected in the archive.

**Adding Tags to a Redundant Collector:** To add tags in a redundant collector system, you must add them to the primary collector. If tags are added to a redundant collector, data for these tags will not be collected until the redundant collector is taken out of the redundant system. If the primary collector is not available, you must add the tag manually.

**Note:**

- Use polled tags only as watchdog tags.
- Historian redundant collector configuration does not force the active Historian collector to run on the active iFIX SCADA, since both redundant collectors provide data. Also, when both iFIX SCADAS become active, they lose connection with each other.

### Important information about the failover of redundant collectors:

- In the **Redundancy** section of the **Collectors** page, you can use the **Make Active Collector Now**button to manually force a failover to a backup collector.
- In an Enterprise system, collector redundancy failover happens only after 5 minutes after a tag change. You must select the **Make Active Collector Now** button after the first 5 minutes for the failover to happen.

  Also, when you shut down an active collector, it does failover. However, if there was a tag change then shutting down the active collector does not cause failover immediately, it is just delayed by 5 minutes. There will not be any data loss since the backup collector sends data for the past 15 minutes when it becomes active.
- Failover precedent is cyclical; the last collector in a redundant group will automatically failover to the first collector in the group.
- Configuration Manager must be running for failover to happen in a mirrored environment.

## Configure Redundant Collectors and Groups

1. Create the collector instances *(on page 357)* that you want to use for collector redundancy. All these collectors must be of the same type.

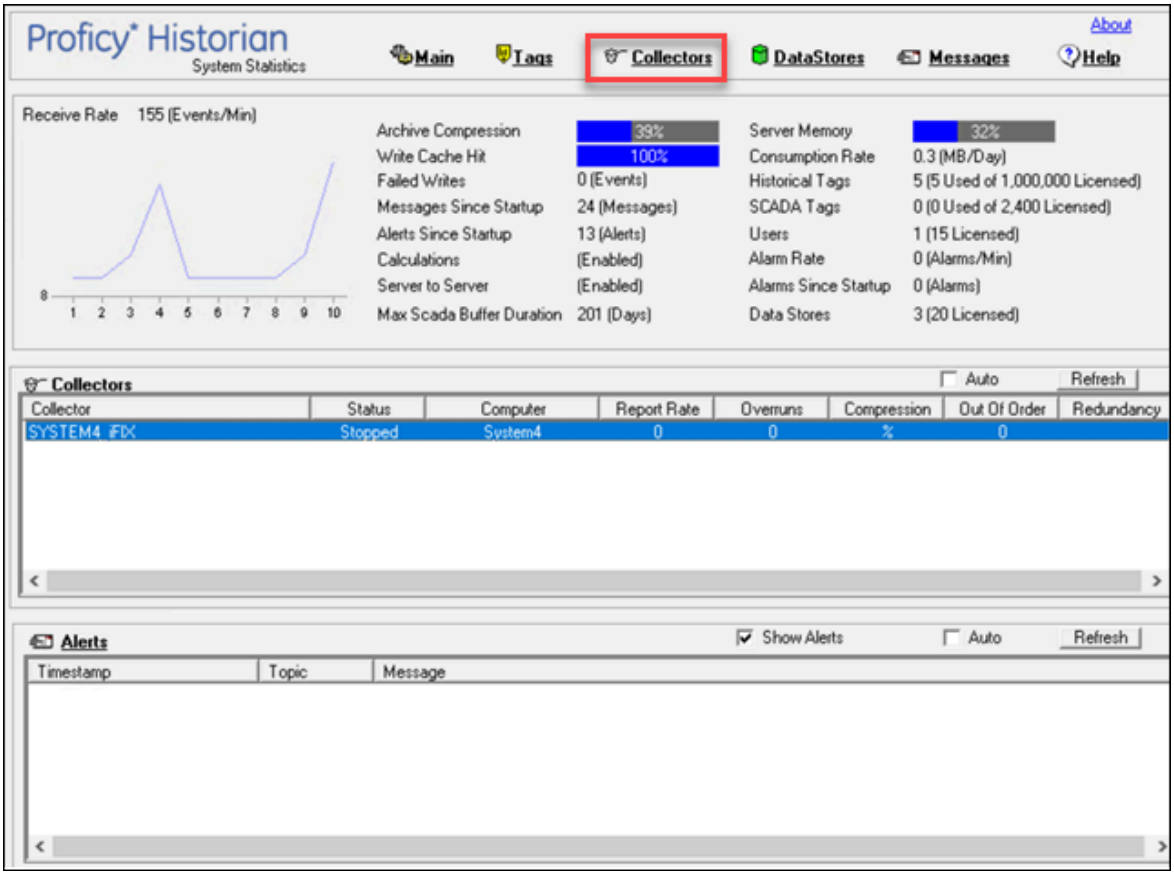
   > **Note:**
   > Collector redundancy is not available and supported for the following collectors:
   > - File Collector.
   > - Calculation Collector.
   > - Python Collector.
   > - Server-to-Server Collector.
   > - Server-to-Server Distributor.
   > - OSI PI Collector.
   > - OSI PI Distributor.

2. Create tags *(on page 909)* in the primary collector.
3. If you want to use the values of a watchdog tag as a failover trigger, create the watchdog tag *(on page 909)* in the primary collector.

Collector redundancy ensures that collection of your data remains uninterrupted. It uses two or more collectors that gather data from a single source. For more information, refer to About Collector Redundancy *(on page 975)*

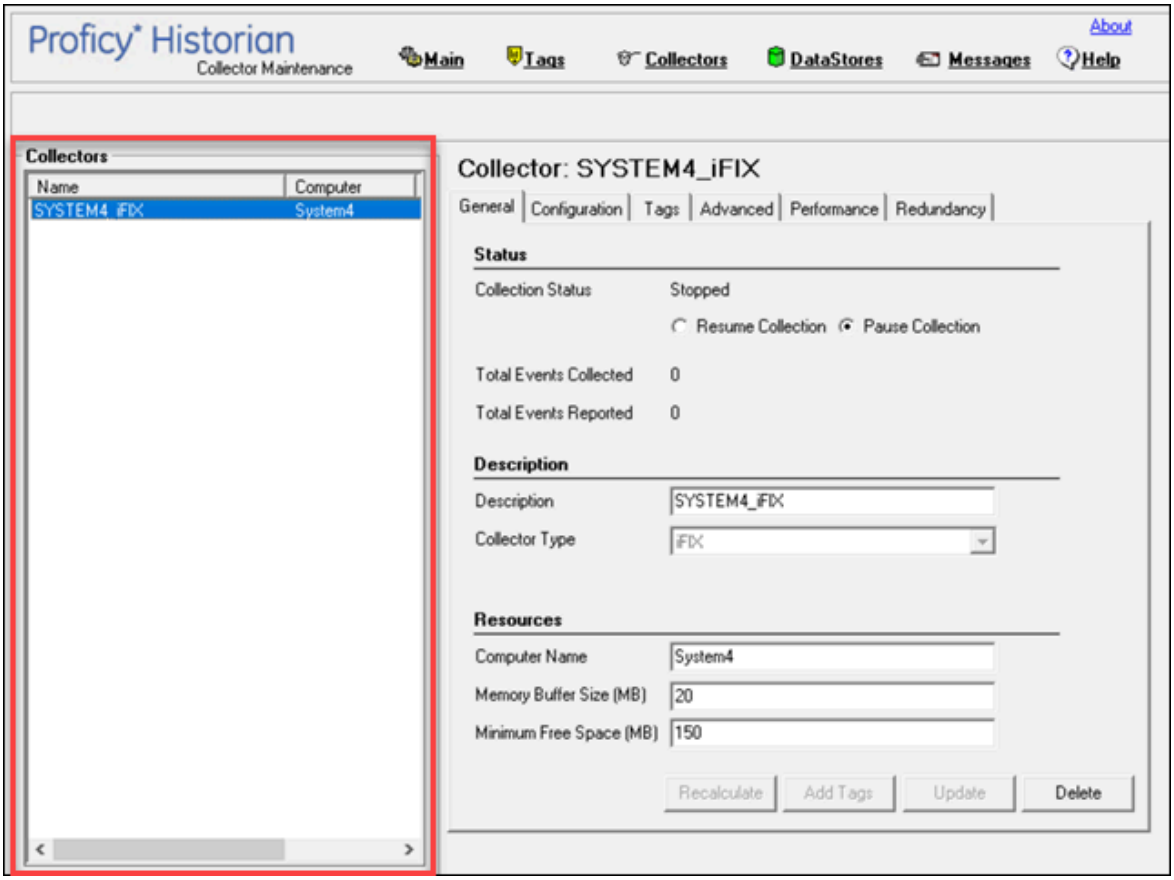
This topic describes how to set up redundancy between two collectors - one primary and the other secondary. You can, however, set up multiple secondary collectors.

1. Access Historian Administrator *(on page 823)*.
2. Select **Collectors**.

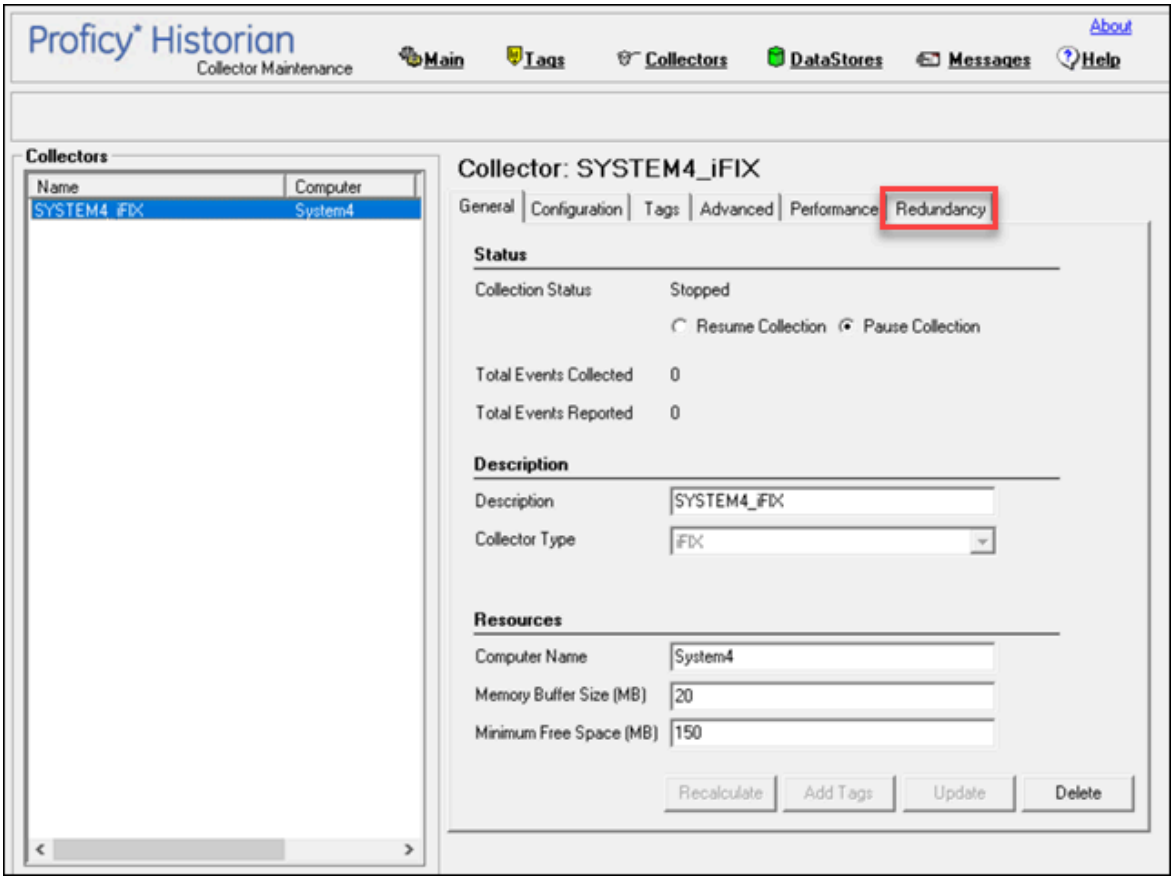



A list of collectors appears.

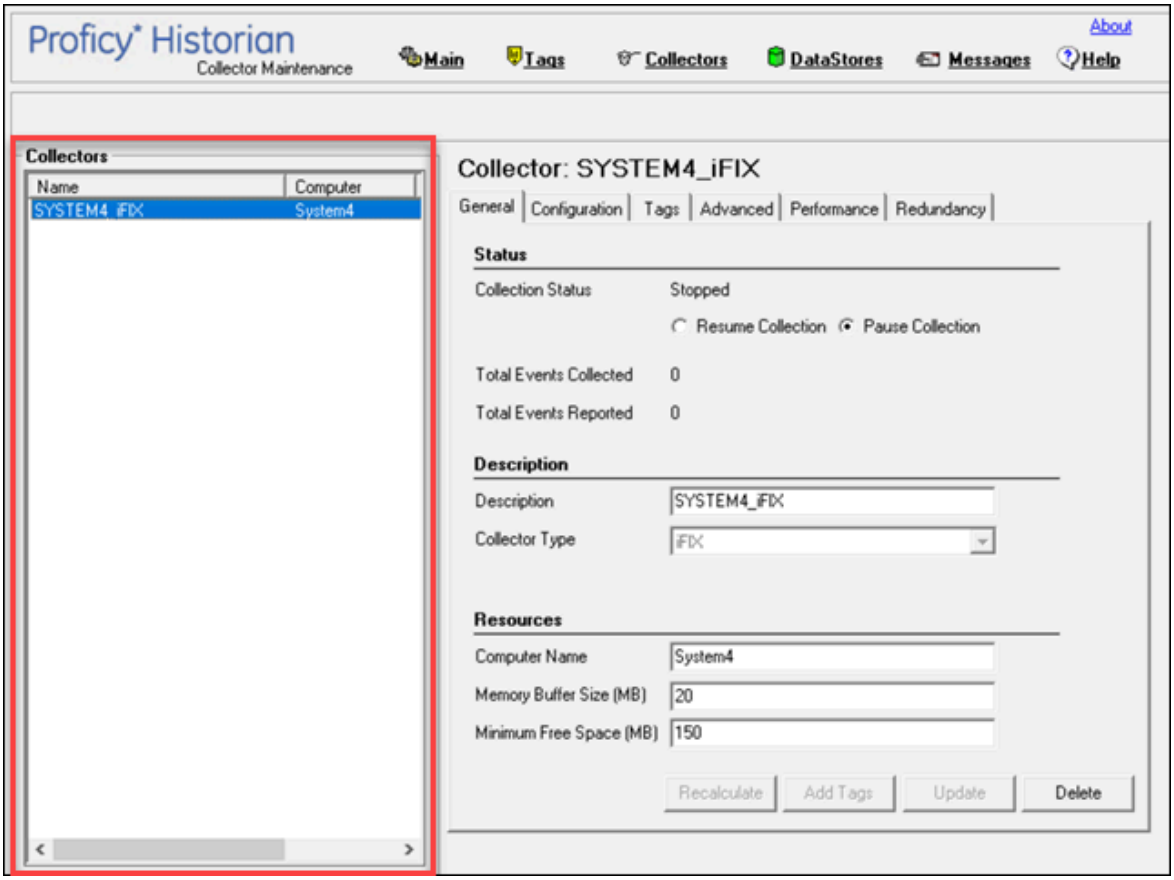
3. Select the collector that you want to set as the first collector in your redundant group.




4. Select **Redundancy**.




5. In the **Redundant Collector** field, select **Enabled**, and then select **Update**.
6. Select the collector that will be your second collector in your redundant group.




7. Select **Redundancy**, and then enter values as described in the following table.

| Field | Description |
|---|---|
| **Redundant Collector** | Select **Enabled**. |
| **Backup For** | Select the collector that this collector will back up (in this case, it is the first collector in your redundant group). |

8. Select **Update**.
   A message appears, asking you to confirm that tags configured for the backup collector will no longer be collected.
9. Select **Yes**.
   Redundancy is now configured for these two collectors. On the main page, the **Redundancy Status** of the first collector will be Active and the backup collector, Standby.

   **To add more collectors to the redundant group**: Repeat steps 6 through 9. In the **Backup For** field, select the collector that backs up the last collector in the group. If you want to have collectors in

the redundant group failover when the active collector's status is unknown, select **Enabled** in the **Collector Status** field. Make sure you are browsing and defining the watchdog tag in the principal (or first) collector in your redundant group.

# Maintaining, Operating, and Monitoring Historian

## Maintain, Operate, and Monitor Historian

To ensure reliable, error-free operation over a long period of time, develop and execute a consistent maintenance program for the Historian system and the data it collects. The subsequent topics provide guidelines for setting up such a plan and for monitoring and interpreting system performance indicators.

## Data Types

Historian uses the following data types.

| Data Type | Size | Description | Valid Values |
|---|---|---|---|
| Single Float | 4 bytes | Stores decimal values up to 6 places. | 1.175494351e-38F to 3.402823466e+38F |
| Double Float | 8 bytes | Stores decimal values up to 15 places. | 2.2250738585072014e-308 to 1.7976931348623158e +308 |
| Single Integer | 2 bytes | Stores whole numbers. | -32767 to +32767 |
| Double Integer | 4 bytes | Stores whole numbers. | - 2147483648 to +2147483648 |
| Quad Integer | 8 bytes | Stores whole numbers. | -9,223,372,036,854,775,808 (negative 9 quintillion) to +9,223,372,036,854,775,807 (positive 9 quintillion) |
| Unsigned Single Integer | 2 bytes | Stores whole numbers. | 0 to 65535 |
| Unsigned Double Integer | 4 bytes | Stores whole numbers. | 0 to 4,294,967, 295 (4.2 billion) |
| Unsigned Quad Integer | 8 bytes | Stores whole numbers. | 0 to 18,446,744,073,709,551,615 (19 quintillion) |

| Data Type | Size | Description | Valid Values |
|---|---|---|---|
| Byte | 1 byte | Stores integer values. | -128 to +127 |
| Boolean | 1 byte | Stores boolean values. | 0=FALSE and 1=TRUE (any value other than zero is treated as one) |
| Fixed String | Configured by user | Stores string data of a fixed size. | 0 and 255 bytes |
| Variable String | No fixed size | Stores string values of undetermined size. This data type is useful if you cannot rely on a constant string length from your data source. | |
| Binary Object | No fixed size | Stores binary data. This is useful for capturing data that can not be classified by any other data type. | |
| Scaled | 2 bytes | Stores a 4 byte float as a 2 byte integer. The scaled data type saves disk space but sacrifices data precision as a result. | |

**Note:**
Tags associated with Quad Integer, Unsigned Double Integer, or Unsigned Quad Integer data types may suffer a loss of precision value due to a Visual Basic limitation.

**Note:**
The Calculation collector supports only calculations performed using the current value calculation. It does not support other calculations due to a Visual Basic script limitation.

### Scaled Data Types

Historian uses the high and low EGU values to store and retrieve archived values for the scaled data type. This allows you to store 4 byte floats as 2 byte integers in the archive. Though this saves disk space, it sacrifices data precision. The smaller the span is between the high and low EGU limits, the more precise the retrieved value will be. When calculating the value of a scaled data type, you can use this formula:

```
ArchivedValue = (((RealWorldValue - EngUnits->Low) /

 (EngUnits->High - EngUnits->Low) * (float) HR_SCALED_MAX_VALUE) + .5);
```

For example: A value of 12.345 was stored in a scaled tag whose high EGU was 200 and low EGU was 0. When later retrieved from the Historian archive, a value of 12.34473 will be returned.

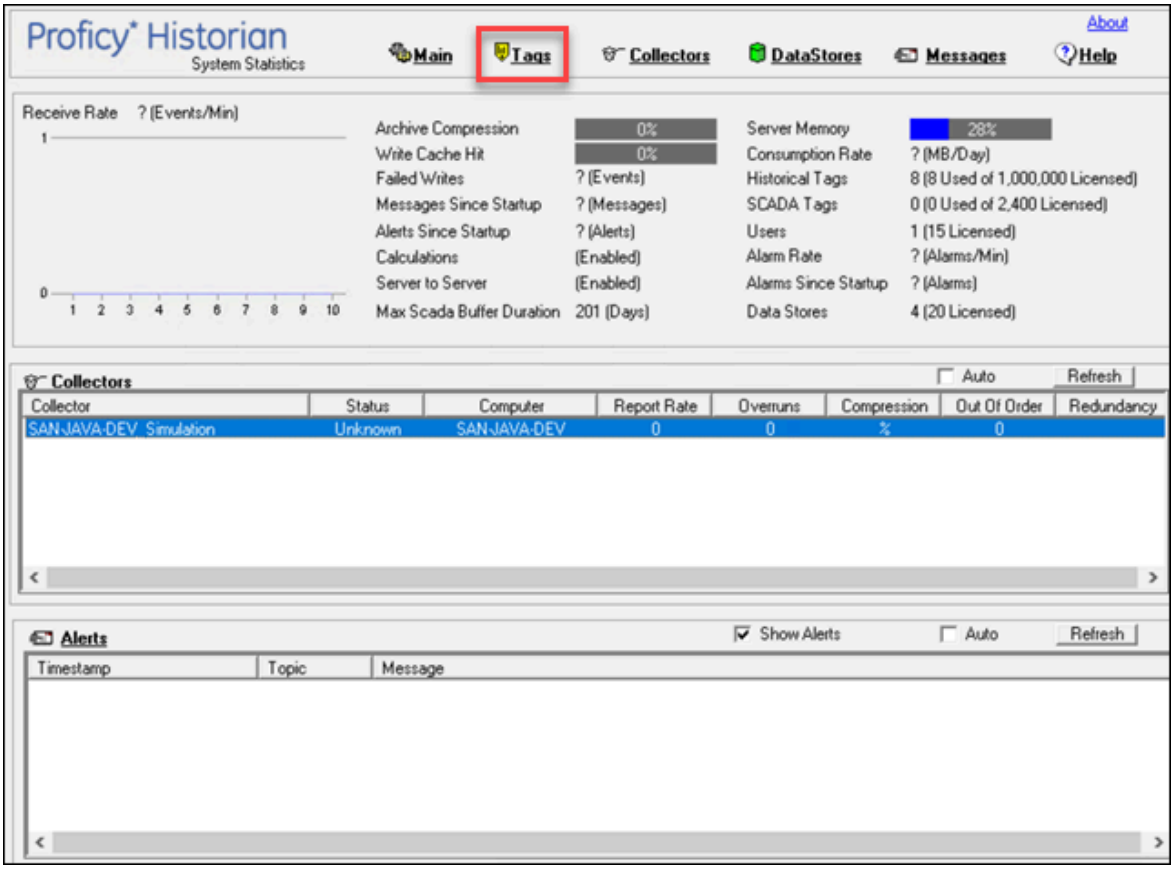
**Important:**
Values that are outside of the EGU range of a scaled data type tag are stored as bad, scaledoutofrange in Historian. Changing either the High or Low EGU tags does not affect existing data, but only affects the new data with new timestamps. You cannot correct values for scaled data types that were inserted while EGUs were incorrect. If necessary, contact technical support for additional information.

**Quad Integer Data Types:** The high and low EGU limits for Quad Integer, Unsigned Single Integer, Unsigned Double Integer, Unsigned Quad Integer are between 2.2250738585072014e-308 to 1.7976931348623158e+308.

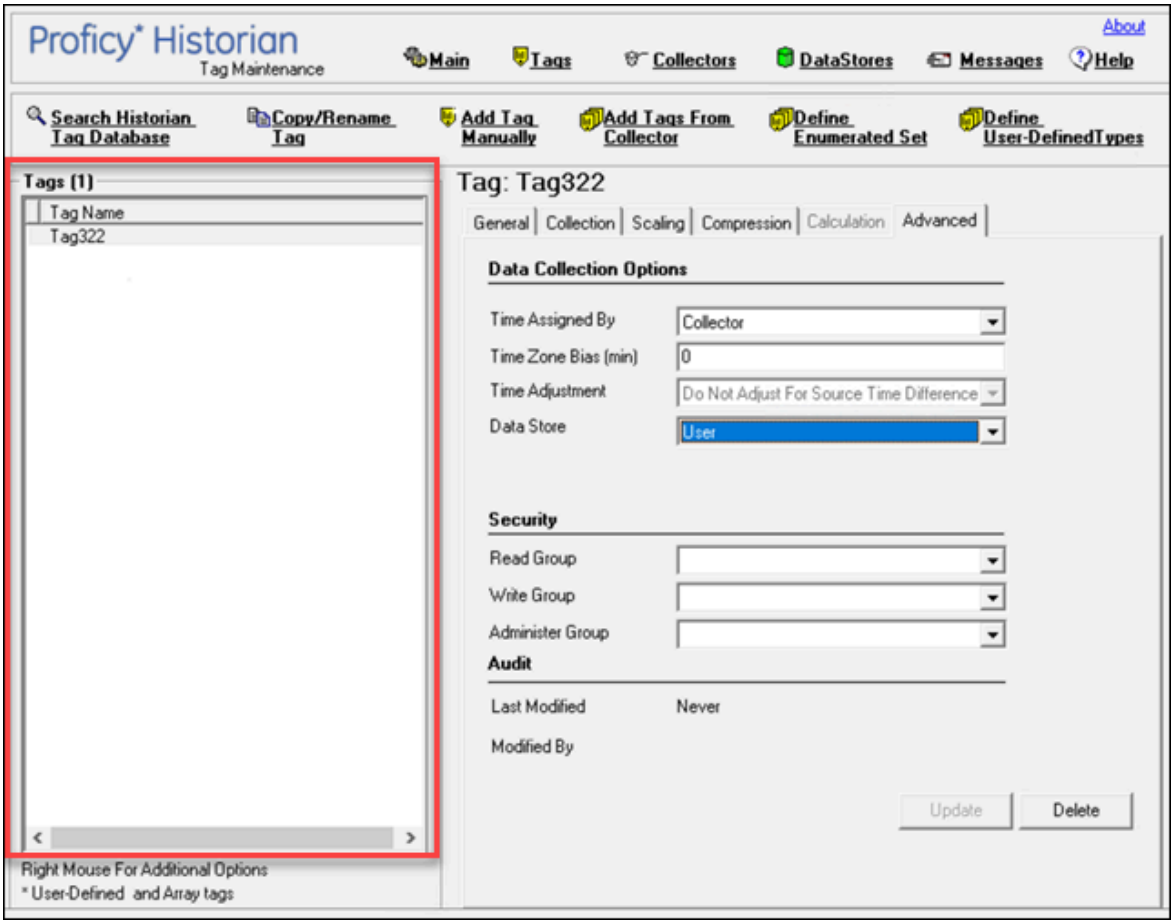
## Set the Size of a Fixed String Data Type

Using the fixed string data type, you can store string data of a fixed size. This is useful when you know exactly what data will be received by Historian. If a value is larger than the size specified in the **Data Length** field, it will be truncated.

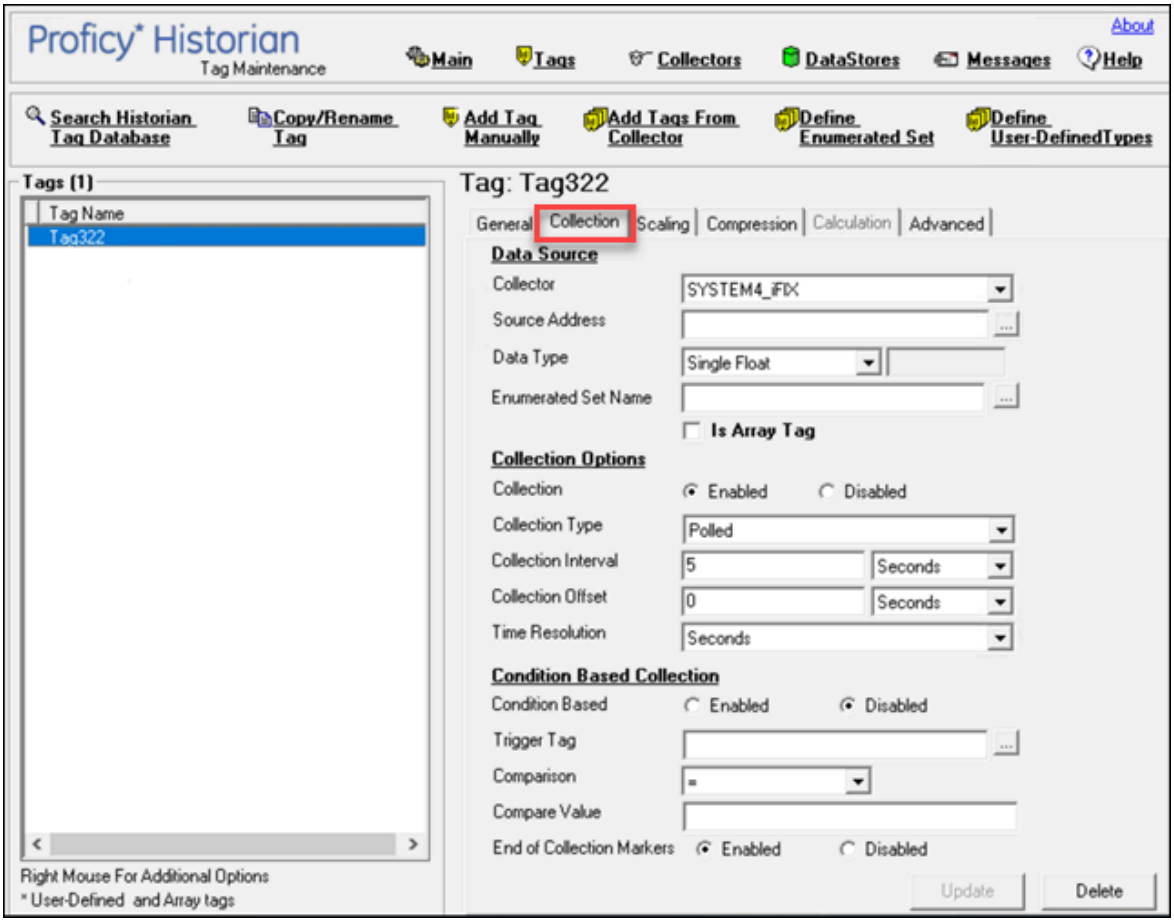
1. Access Historian Administrator *(on page 823)*.
2. Select **Tags**.




3. Select the tag for which you want to set a fixed string data type.




4. Select **Collection**.




5. In the **Data Type** box, select **Fixed String**.
6. Enter a value in bytes in the adjacent field.
   The fixed string data type is set for the tag.

## Plan for Data Recovery

Planning for data recovery means always having up-to-date backup files for important information that you can call up and restore quickly when the need arises.

The .IHC file, which contains all configuration information, is an important file to back up. The .IHC file is automatically backed up when, and only when, you back up the current archive .IHA file. With the .IHC file, you can always restore the system configuration to the state it was in before the event occurred.

It is also important to backup the current online archive files *.IHA. If you restore the archive files, along with the configuration, you can quickly pick up where you left off when the event occurred with a minimum loss of data.

By default, the .IHC backup path is the same as the archives path. The .IHC uses the following naming convention: `ComputerName_Config-Backup.ihc`. If the default backup path is different than the archives path, the .IHC file is copied to the backup folder with the standard .IHC naming convention: `ComputerName_Config.ihc`.

## Develop a Maintenance Plan

The primary goal of a maintenance plan is to maintain integrity of the data collected. If you are successful in this regard, you will always be able to recover from a service interruption and continue operation with minimal or no loss of data. Since you can never ensure 100% system uptime, you must frequently and regularly back up current data and configuration files, and maintain non-current archive files in a read-only state, following the guidelines for backup and routine maintenance.

**Daily Maintenance**

On a daily schedule, perform the following backup operations, unless you use ihArchiveBackup.exe to back up archives automatically.

1. Use Historian Administrator to back up the current archive and most recent .IHA archived data file. This preserves data collected up to this moment in time. You do not need to back up any read-only archive files after they have been backed up once.
2. Use Windows Explorer to back up the .IHC file if it has been modified, unless it is backed up automatically. This file contains all current configuration information (tag configuration, archive configuration, and collector configuration). Using this file, you can restore the system configuration after an unplanned shutdown.

**Routine Maintenance:** On a regular schedule, examine and analyze the system performance indicators displayed on the System Statistics page of Historian Administrator as follows.

**Table 75. System Statistics Performance Indicators**

| Field | Recommended Action |
|---|---|
| Est. Days to Full | If time is growing short, make sure that the server has sufficient unused storage capacity to open a new archive when the active one fills up. Verify that the Create New Archives Automatically function is enabled. If it is disabled, you must manually create a new archive before the active archive fills up. |

**Table 75. System Statistics Performance Indicators (continued)**

| Field | Recommended Action |
|---|---|
| | **Note:**<br>If you do not have enough unused storage capacity, you may have to enable the Overwrite Old Archives feature. Since this feature overwrites existing data, exercise caution in using it. |
| Consumption Rate of Archive Storage | If the rate is excessively high, reduce the rate at which data flows into the system or increase the filtering applied to the data to lower the rate of archiving. To reduce the collection rate, slow the polling rate on some or all tags. To increase filtering, enable compression at the collector and/or archiver and widen the compression deadbands. |
| Failed Writes | If the display shows a significant number of failed writes, investigate the cause and take corrective action to eliminate the malfunctions. Refer to the `DataArchiver-XX.log` file or query the message database to determine the tags for which failed writes occurred.<br><br>For example, trying to write values to a deleted archive causes failed writes. Trying to archive data with a timestamp that precedes the start time of the first archive, trying to write to a read-only archive, or trying to write a value with a timestamp more than 15 minutes ahead of the current time on an archiver will produce a failed write. |
| System Alerts | Examine the messages and alerts and take appropriate action to correct any problems. |

On a regular schedule, examine and analyze the performance indicators displayed in the **Performance** section.

**Table 76. Collector Performance Indicators**

| Field | Recommended Actions |
|---|---|
| Avg. Event Rate Chart | Is the rate at a normal level? Does the chart exhibit an acceptable trend line?<br><br>If not, determine why. Balance polling schedules, adjust scan frequencies (collection intervals), and modify compression deadbands to lighten load. |
| Compression Chart | Is compression effectiveness acceptable? |

**Table 76. Collector Performance Indicators (continued)**

| Field | Recommended Actions |
|---|---|
| | If not, verify that compression is enabled and then widen the deadbands to increase the effect of compression. |
| Overruns Chart | If the value is anything other than zero, determine the severity and cause of the problem and take corrective action. |

## Evaluate and Control Data Compression

You can achieve optimum performance in Historian by carefully controlling the volume of dynamic data it collects and archives. You need enough information to tell you how the process is running, but you do not need to collect and store redundant or non-varying data values that provide no useful information. More information can be found in the Notes on Collector and Archive Compression reference.

**Control Data Flow**

You can control the amount of online or dynamic data the system handles at a given time by adjusting certain system parameters. The general principle is to control the flow of data into the archive either by adjusting the rate at which the collectors gather data or by adjusting the degree of filtering (compression) the system applies to the data collected.

This manual describes the detailed procedures for executing these adjustments in the sections: Configuring Archives, Configuring Tags, and Configuring Collectors.

Adjust the following parameters to **reduce** the rate of data flow into the server.

- Reduce the polling rate by increasing the collection interval for unsolicited and polled collection.
- Enable collector compression and optionally use compression timeout.
- Set the compression deadband on the collectors to a wider value.
- Use the collector compression timeout.

Adjust the following parameters to **increase the filtering** applied by the archiver in the server.

- Enable archive (trend) compression.
- Set the archive compression deadband to a wider value.
- Where possible, use the scaled data type and enable input scaling on selected tags.
- Where possible, select milliseconds or microseconds rather than seconds for time resolution. Seconds is optimum for most common devices. This affects disk space.

**Evaluate Data Compression Performance**

You can determine how effectively data compression is functioning at any given time by examining the system statistics displayed on the System Statistics page of Historian Administrator, as shown in Historian Administrator (System Statistics) Screen.

The compression field at the top of the page shows the current effect of archive compression. Values for this parameter should typically range from 0 to 9%. If the value is zero, it indicates that compression is either ineffective or turned off. If it shows a value other than zero, it indicates that archive compression is operating and effective. The value itself indicates how well it is functioning. To increase the effect of data compression, increase the value of archive compression deadband so that compression becomes more active.

## Handle Value Step Changes with Collector Data Compression

**Note:**
Individual tags can be configured to retrieve step value changes.

If you enable collector compression, the collector does not send values to the archiver any new input values if the value remains within its compression deadband. Occasionally, after several sample intervals inside the deadband, an input makes a rapid step change in value during a single sample interval. Since there have been no new data points recorded for several intervals, an additional sample is stored one interval before the step change with the last reported value to prevent this step change from being viewed as a slow ramp in value. This value marks the end of the steady-state, non-changing value period, and provides a data point from which to begin the step change in value.

The collector uses an algorithm that views the size of the step change and the number of intervals since the last reported value to determine if a marker value is needed. The following is an example of the algorithm:

```
BigDiff=abs(HI_EGU-LO_EGU)*(CompressionDeadbandPercent/(100.0*2.0))*4.0

If ( Collector Compression is Enabled )

If ( Elapsed time since LastReportedValue>=( SampleInterval * 5 ) )

If ( abs(CurrentValue-LastReportedValue) > BigDiff )

Write LastReportedValue,Timestamp=(CurrentTime-SampleInterval)
```

In the example above, if a new value was not reported for at least the last 4 sample intervals, and the new input value is at least 4 deltas away from the old value (where a single delta is equal to half of the compression deadband), then a marker value is written.

**Note:**
These settings are also adjustable from the Registry. Please contact technical support for more information.

### Value Spike with Collector Compression

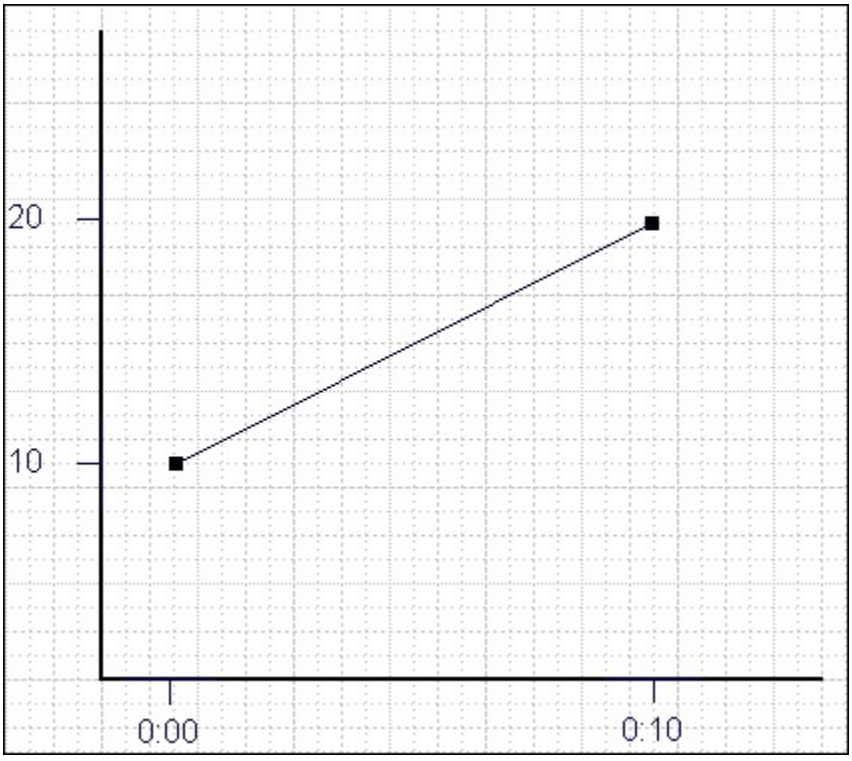
For example, a collector reads a value X once per second, with a compression deadband of 1.0. If the value of X is 10.0 for a number of seconds starting at 0:00:00 and jumps to 20.0 at 0:00:10, the data samples read would be:

| Time | X Value |
|---|---|
| 0:00:00 | 10.0 (steady state value) |
| 0:00:01 | 10.0 |
| 0:00:02 | 10.0 |
| 0:00:03 | 10.0 |
| 0:00:04 | 10.0 |
| 0:00:05 | 10.0 |
| 0:00:06 | 10.0 |
| 0:00:07 | 10.0 |
| 0:00:08 | 10.0 |
| 0:00:09 | 10.0 |
| 0:00:10 | 20.0 (new value after step change) |

To increase efficiency, the straightforward compression would store only 2 of these 11 samples.

| Time | X Value |
|---|---|
| 0:00:00 | 10.0 (steady state value) |
| 0:00:10 | 20.0 (new value after step change) |

However, without the marker value, if this data were to be put into a chart, it would look like the data value **ramped** over 10 seconds from a value of 10.0 to 20.0, as shown in the following chart.




The addition of a **marker value** to the data being stored results in the following data values:

| Time | X Value |
|---|---|
| 0:00:00 | 10.0 (steady state value) |
| 0:00:09 | 10.0 (inserted Marker value) |
| 0:00:10 | 20.0 (new value after step change) |

If you chart this data, the resulting trend accurately reflects the raw data and likely real world values during the time period as shown in the following chart.

## Reviewing System Alerts and Messages

You can examine system alerts by scrolling through the Alerts panel of Historian Administrator System Statistics page or by examining the Message Search page for more detail. Refer to Searching for Messages *(on page 850)* for detailed instructions. You can diagnose most operating problems by examining the archiver LOG file.

You may also find the Windows Event Viewer to be useful in diagnosing a problem. To open the Event Viewer, use the following procedure.

1. Launch search.
2. Type `eventvwr.msc` and select **OK**.
   The Event Viewer window appears.
3. Highlight an item.
   A window appears with an explanation of the event.
4. Select **Previous** or **Next** to step through the messages.

## Monitor Historian Health and Status

Historian provides extensive functionality to determine the health and status of both the data archiver and the collectors. You can access this information through Historian Administrator or set up your Historian

system to automatically alert you to any Historian change. The following sections describe procedures and hints for monitoring your Historian system, including:

- Monitor Operating System Event Files
- Monitor Collector Status Tags
- Monitor Historian Subscriptions
- Use Collector Status Tags
- Subscribe to Historian Alerts and Messages
- Create Subscriptions

**Monitor Operating System Event Files**

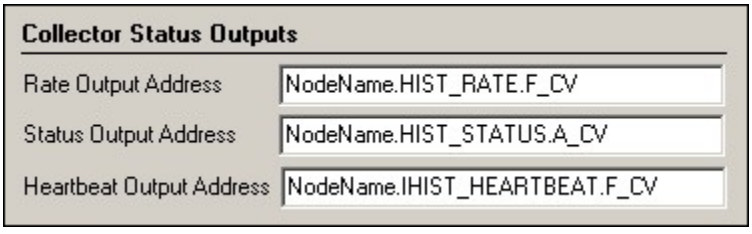
You can view the operating system event files through the Event Viewer. For more information on accessing the Event Viewer, refer to Review System Alerts and Messages *(on page 995)*. The operating system event files contain the Historian Status messages found in the application log. Entries in that file are similar to those found in the Alerts panel, but may contain multiple entries for certain items. Multiple entries occur as a result of distributing the components of Historian. Each component logs its critical status changes to the local event log. For example, a collector and the data archive will both log a collector going off-line.

**Monitor Collector Status Tags**

The iFIX and OPC Collectors contain three status fields that you can write back to the data source for monitoring. Additionally, the iFIX Collectors allow you to display and alarm on the following three status fields.

- **Rate Output** - The number of events per minute the collector processes.
- **Status Output** - The status of the Collector as either Running or Stopped.
- **Heartbeat** - Allows the data source to verify that the collector is working. The collector writes a value of one to this address once a minute.

Other collectors allow you to define tags to record the status of the collector. These tags give an indication of the health of a collector, not the health of the Historian The following figure displays the iFIX collector status tags.



The three tags referenced correspond to tag names within the SCADA node FIX.

- **NodeName.HIST_RATE.F_CV** – An AI block. The F_CV field gives the number of events per minute that the collector is reporting to the server.
- **NodeName.HIST_STATUS.A_CV** – A text block. The A_CV field displays a status such as Running or Stopped depending on the status on the collector. This field does not reflect whether or not the Collector program is running.
- **NodeName.HIST_HEARTBEAT.F_CV** – A digital alarm block that will alarm if the F_CV field remains OPEN for more than 60 seconds. The iFIX collector closes the F_CV field every 60 seconds. You must use a pro- gram block in conjunction with this block to automatically OPEN the block if it has been CLOSED. This provides for a momentary reset of the 60 second alarm timeout. Unlike the HIST_STATUS block that only updates the Collector status if the collection stops, the HIST_ HEARTBEAT detects that the collection has stopped or that the Collector application is no longer running.

## Monitor Historian Subscriptions

Certain objects in the Historian SDK allow you to create subscriptions that fire an event back to the calling application whenever the object's status updates. These subscriptions allow for exception-based monitoring of the Historian status without resorting to intensive polling of the server.

The following table displays the objects and their supported subscription events:

**Table 77. Historian Objects with Subscription Events**

| Object | Subscription Events |
|---|---|
| Messages Object | AlertReceived and MessageReceived Events |
| Tags Object | ChangeReceived Event |
| Data Object | CurrentDataReceived Event |
| Archives Object | StatusReceived Event |
| Collectors Object | StatusReceived Event |

Subscribing to any of these Object's events allows you to receive direct notification of status changes. For example, using iFIX scripting, you could route this notification to any alarm queue in iFIX.

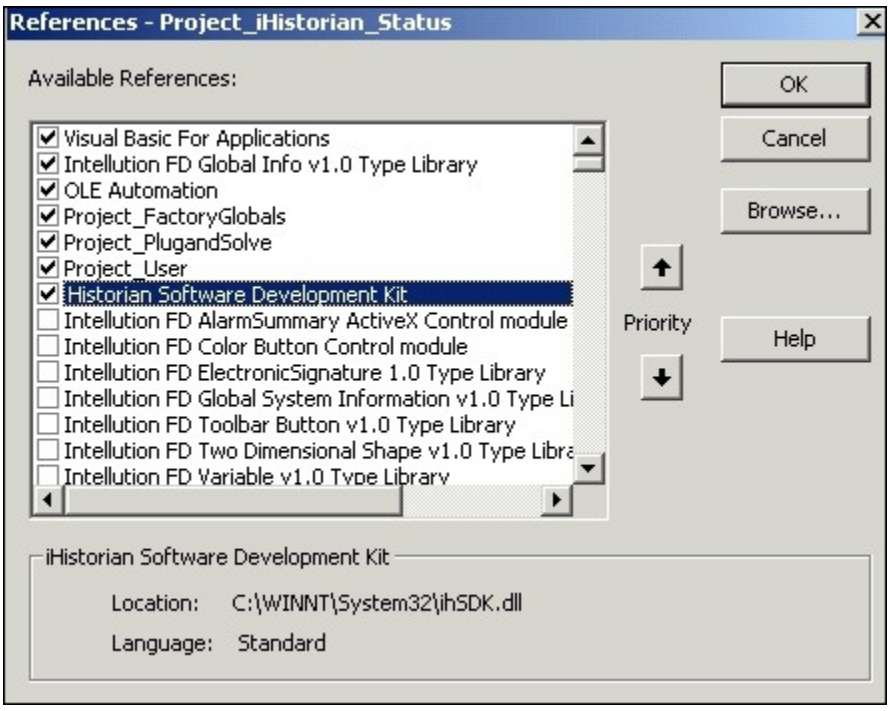
# Subscribe to Historian Alerts and Messages

You can obtain a complete picture of the Historian Server's status by monitoring the server's Alerts and Messages. Additionally, if you are using iFIX with Historian, you can capture these subscriptions and forward them to iFIX alarm queues to ensure operator notification.

You can also use scripting in iFIX to subscribe to the Historian message queues using VBA. If you write a script as part of an iFIX picture, you will need to either keep the picture with the scripts running at all times or run the scripts in the FIXBackground Server application.

## Creating Subscriptions in iFIX

To subscribe to messages or alerts in Historian, you must reference the SDK from within the VBA project. To access the References window, shown in the following figure, use the Tools menu in the iFIX VBA Editor.




Once you reference a VBA project to the Historian SDK, you must define two module level objects: a Server Object and a Messages Object. Declare these Objects to remain in scope for the duration that the subscription is active.

**Note:**
You must declare the Messages Object using the WithEvents keyword. This keyword is only valid at the module or class level.

Typically, the Server and Messages Object are declared as private module level objects. Subscriptions are made during the module initialization or upon loading the method. You can subscribe to Alerts or Messages using either the SubscribeAlerts or SubscribeMessages Method of the Messages Object.

### Specify Topics

When subscribing to Historian Messages you must specify which topics you are subscribing to. There are six topics.

- Message Topics: Configuration Audit, Connections, General
- Alert Topics: Performance, Security, ServiceControl

Once you make the subscription, the AlertReceived or MessagesReceived Event will fire when a new message is created on the Historian Server. The topic of the message determines which event fires. Additionally, the event receives a copy of the Message or Alert Object properties listed as follows:

- MessageNumber – The NLS string number.
- MessageString – Translated string, including substitutions for time stamp, user, or tag name.
- Substitutions – A collection of substitutions used to make the message string.
- TimeStamp – The time the message was created.
- Topic – The topic number of the message.
- TopicName – The string topic name of the message.
- UserName – The user name for the message.

You can also send Alerts and Messages to the iFIX Operator Message alarm queue using the SendOperatorMessage method of the System object. This method will send the alerts and messages to all alarm queues, but NOT the Alarm Summary queue. To send a message that will appear on an alarm summary, send the message to the Alarm Extension field of a digital alarm block and toggle the alarm state.

## Monitor Historian Performance

Historian provides a variety of counters and tags that can be used to monitor how well the Historian components are performing. You can use performance tags or counters to determine the resource usage on the computer that runs the Historian application. Performance counters are also useful when Historian Administrator is not installed or cannot connect.

- Use performance **tags** to view information in an Excel report or SDK program, possibly along with other Historian tags.
- Use performance **counters** to view information using Windows Performance monitor, possibly along with non-Historian counter information.

Like any Windows performance counter, you must add the counters for collection to view history. Performance tags are always being collected and you can view past data any time.

Performance counters are updated in real time. Performance tags are updated once per minute with the activity over the last minute.

Performance counters contain more information than tags.

Any counter can be collected to a tag using the Historian Windows Performance Collector. Those tags will count against your licensed tag count.

For more information, refer to the following sections:

- Historian Server Performance Tags
- Historian Server Performance Counters

## Historian Server Performance Tags

The following table provides information about the various Historian Server Performance tags.

**Table 78. Server Performance Tags**

| Tag Name | Description |
|---|---|
| PerfTag_CompressionRatio | Specifies the current effect of archive data compression. |
| PerfTag_MinimumCompressionRatio | Specifies the minimum compression ratio. |
| PerfTag_MaximumCompressionRatio | Specifies the maximum compression ratio. |
| PerfTag_TotalEvents | Specifies the total number of data samples reported to the Historian archive from all sources. |
| PerfTag_TotalOutOfOrder | Specifies the total out of order data samples. |
| PerfTag_AverageEventRate | Specifies the average number of data samples per minute sent to archiver from all sources |
| PerfTag_MinimumEventRate | Specifies the minimum number of data samples per minute sent to archiver from all sources. |

**Table 78. Server Performance Tags (continued)**

| Tag Name | Description |
|---|---|
| PerfTag_MaximumEventRate | Specifies the maximum number of data samples per minute sent to archiver from all sources. |
| PerfTag_WriteCacheHitRatio | Specifies the hit ratio of the write cache in percent of the total writes. |
| PerfTag_TotalFailedWrites | Specifies the total number of samples since startup that failed to be written. |
| PerfTag_TotalMessages | Specifies the total messages (for example, connection or audit messages) received by the archiver since startup |
| PerfTag_TotalAlerts | Specifies the total number of alerts received by the data archiver since startup. |
| PerfTag_FreeSpace | Indicates the free disk space left in the current archive. |
| PerfTag_SpaceConsumptionRate | Specifies an archive disk space consumption rate in megabytes per day. |
| PerfTag_PredictedDaysToFull | Indicates the approximate number of days required for an archive to fill. |
| PerfTag_MemoryUsage | Specifies the amount of RAM used by the Data Archiver. |
| PerfTag_MemoryVMSize | Specifies the amount of virtual memory used by the Data Archiver. |
| PerfTag_TotalAlarms | Specifies the total number of alarms received by the Data Archiver since starting up. |
| PerfTag_AverageAlarmRate | Specifies the average alarm rate in alarms per minute received by Data Archiver. |
| PerfTag_TotalFailedAlarms | Specifies the total number of alarms since startup that failed to be written. |
| Perftag_ReadQueueSize | Specifies the total number of messages present in the Read queue. |
| Perftag_AverageReadRate | Specifies the total number of data samples per minute returned from the Data Archiver for all read requests. |

**Table 78. Server Performance Tags (continued)**

| Tag Name | Description |
|---|---|
| Perftag_ReadQueuePushRate | Specifies the number of read requests per minute that came into the archiver from all clients. A read request can return multiple data samples. |
| Perftag_WriteQueuePushRate | Specifies the number of write requests per minute that came into the archiver from all clients. A write request can contain multiple data samples. |

## View Data Trends for Tags

1. On the **Tag Maintenance** page, select a tag.
2. Right-select the tag and select **Trend**.
   The trend for the selected tag displays.

## Add a Performance Tag

1. In the **Tag Maintenance** page, select the **Tags** link on the toolbar.
   The Tag Maintenance page appears.
2. Select the **Add Tag Manually** link on the toolbar.
   The Add Tag window appears.
3. Enter the performance tag name.
4. Select **OK**
   The Tag Maintenance page displays with the specified tag properties.

## Historian Collector Performance Counters

The following table provides information about collector-specific performance counters.

**Note:**
Replace the placeholder value `%CollectorName%` with the actual name of the collector.

**Table 79. Collector Performance Counters**

| Counter Name | Description |
|---|---|
| PerfTag_`%CollectorName%`_InterfaceStatus | Specifies the status of an interface. |

**Table 79. Collector Performance Counters (continued)**

| Counter Name | Description |
|---|---|
| PerfTag_%CollectorName%_InterfaceTotalEventsCollected | Specifies the total number of events collected by an interface. |
| PerfTag_%CollectorName%_InterfaceTotalEventsReported | Specifies the total number of events reported by an interface. |
| PerfTag_%CollectorName%_InterfaceOutOfOrderEvents | Specifies the total out of order events. |
| PerfTag_%CollectorName%_InterfaceAverageEventRate | Specifies the average event rate of an interface. |
| PerfTag_%CollectorName%_InterfaceMinimumEventRate | Specifies the minimum event rate of an interface. |
| PerfTag_%CollectorName%_InterfaceMaximumEventRate | Specifies the maximum event rate of an interface. |
| PerfTag_%CollectorName%_InterfaceOverruns | Specifies the interface overruns. |
| PerfTag_%CollectorName%_InterfaceCompression | Specifies the compression of an interface. |
| PerfTag_%CollectorName%_InterfaceOverrunsPercent | Specifies the number of overruns in relation to the total events collected since startup. |

## Historian Server Performance Counters

The Windows performance counters are exposed as objects with counters. In the tables below, you can see each counter and the object to which it belongs.

Each object has one or more instances as shown in the Windows Performance Monitor. Counters belong to a specific instance of an object. For example, there is an instance of the Historian Archive object for each IHA in the system.

There is also a LatestArchive instance for each data store. The LatestArchive instance lets you collect the information from the newest archive even as archives become full and new archives are created. The LatestArchive is the newest archive receiving data, not the prepared archive.

To monitor performance counters, you must add the performance counter tags manually.

**Table 80. Historian Archive Object**

| Archive Counter | Description |
|---|---|
| Cache Priority | Relative priority of items from the archive stored to the Windows Cache. A higher priority means the item is more likely to stay in cache. Disk Read Time (usec) Duration of last disk read in microseconds. |
| Disk Read Time (usec) | Duration of last disk read in microseconds. |
| Disk Read Time Max (usec) | Maximum duration across all disk reads from the archive in microseconds. |
| Disk Reads | Number of disk reads from archive. |
| Disk Write Time (usec) | Time of last disk write in microseconds. |
| Disk Write Time Max (usec) | Maximum duration of all disk writes to the archive in microseconds. |
| Disk Writes | Number of disk writes to archive. |
| File Size (MB) | Size of the archive file in MB. |
| Read Calls | Number of read calls to the archive since startup. |
| Read Rate (Calls/min) | Number of read calls to the archive per minute. |
| Write Count | Number of data samples written to archive since startup. |
| Write Count Rate | Number of data samples written to archive per minute. |
| Writes Compressed | Number of data samples since startup that were compressed writes to the archive. |
| Writes Expensive | Number of data samples since startup that were expensive or slow. |
| Writes Failed | Number of data samples that were failed writes to the archive. |
| Writes OutofOrder | Number of data samples that were out of time order writes to the archive. |

**Table 81. Historian Cache Object**

| Counter | Description |
|---|---|
| Hit Percentage | Hit rate percentage (0-100) for successful data retrieval calls to the cache. Higher numbers represent more efficiency. |

**Table 81. Historian Cache Object (continued)**

| Counter | Description |
|---|---|
| Hits | Number of hits in the cache since startup. To reset the count, restart the Archiver. |
| Misses | Number of misses in the cache. |
| Num Adds | Total number objects added to cache. |
| Num Deletes | Total number of objects deleted from cache. |
| Num High Prio Objs | Number high priority objects available for deletion. |
| Num Low Prio Objs | Number of low priority objects available for deletion. |
| Num Med Prio Objs | Number of medium priority objects available for deletion. |
| Obj Count | Number of objects in the cache. |
| Size (KB) | Size of cache in KB. |

**Table 82. Historian DataStores Object**

| Counter | Description |
|---|---|
| Compression Ratio (Max) | Archive compression ratio for this data store. |
| Compression Ratio (Max) | Maximum archive compression ratio for the data store. |
| Compression Ratio (Min) | Minimum archive compression ratio for the data store. |
| Messages (Total Alerts) | Total alerts since startup |
| Messages (Total) | Total messages since startup. |
| Read Calls | Number of read calls to the data store. |
| Read Rate (Calls/min) | Average read rate across all archives in the data store. (Read Calls/ Minute) |
| Read Samp Rate (Samp/min) | Average read rate across all archives in the data store. (Samples/Minute) |
| Space (Consumption MB/day) | Disk space consumption rate. (MB/day) |
| Space (Days To Full) | Number of days until current archive is full. |
| Space (Free in MB) | Free disk space in the current archive. |

**Table 82. Historian DataStores Object (continued)**

| Counter | Description |
|---|---|
| Write Rate (Average) | Average event rate across all archives. (Samples/Minute) |
| Write Rate (Max) | Maximum event rate across all archives. (Samples/Minute) |
| Write Rate (Min) | Minimum event rate across all archives. (Samples/Minute) |
| Writes (Cache Hit Ratio) | Write Cache hit ratio. |
| Writes (Compressed) | Total number of compressed data samples since startup. |
| Writes (Total Failed) | Total failed data sample writes since startup. |
| Writes (Total OutOfOrder) | Total out of order data samples since startup. |
| Writes (Total) | Total data samples across all archives since startup. |

**Table 83. Historian Overview Object**

| Counter | Description |
|---|---|
| Compression Ratio (Average) | Average archive compression ratio of all data stores. |
| Compression Ratio (Max) | Average maximum compression ratio of all data stores. |
| Compression Ratio (Min) | Average minimum compression ratio of all data stores. |
| Memory Usage (KB) | Private bytes memory usage for Data Archiver. |
| Memory VM Size (KB) | Virtual Bytes memory usage for Data Archiver. |
| Messages (Total Alerts) | Sum of total alerts of all data stores since startup. |
| Messages (Total) | Sum of total messages of all data stores since startup. |
| Read Rate (Calls/min) | Sum of all average read rates of all data stores. (Samples/Minute) |
| Read Samp Rate (Samp/min) | Average read rate across all archives. (Samples/Minute) |
| Space (Consumption MB/day) | Sum of space consumption rate (MB/day) of all data stores. |
| Space (Days To Full) | Minimum number of days until current archive is full for all data stores. |
| Space (Free in MB) | Sum of all free space in the current archive of all data stores. |
| Write Rate (Average) | Sum of all average event rates of all data stores. (Samples/Minute) |
| Write Rate (Max) | Sum of all maximum event rates of all data stores. |

**Table 83. Historian Overview Object (continued)**

| Counter | Description |
|---|---|
| Write Rate (Min) | Sum of all minimum event rates of all data stores. |
| Writes (Cache Hit Ratio) | Average write Cache hit ratio of all data stores. |
| Writes (Compressed) | Sum of total number of compressed data samples of all data stores. |
| Writes (Expensive) | Sum of total number of expensive writes data samples of all data stores. One of the reasons for expensive writes is out-of-order data. |
| Writes (Total Failed) | Sum of total failed data sample writes of all data stores. |
| Writes (Total OutOfOrder) | Sum of total out of order data samples of all data stores. |
| Writes (Total) | Sum of total data samples across all archives of all data stores. |

**Table 84. Historian Queue Object**

| Counter | Description |
|---|---|
| ClientQueues with Msgs | The number of client queues with messages current on them. A lower number means all clients are up to date. A higher number means that the archiver is not up to date with incoming network traffic |
| Count (Max) | Maximum number of messages on the queue. (memory and disk) |
| Count (Total) | Number of messages on the queue. (memory and disk) |
| Disk Buf Msg Reads | Number of messages read from the disk buffer file. |
| Disk Buf Msg Writes | Number of messages written to the disk buffer file. |
| Processed Count | Number of messages processed from the queue since startup. |
| Processed Rate (msg/min) | Recent rate at which messages have been processed for the queue. |
| Processing Time (Ave) | Average time in milliseconds to process a message. |
| Processing Time (Last) | Time in milliseconds to process the last message. |
| Processing Time (Max) | Maximum time in milliseconds to process a message. |
| Recv Count (msgs) | Number of messages received into the queue. |
| Recv Rate (msgs/min) | Recent rate at which messages have been received for the queue. |
| Size Kb (Mem&Disk Max) | Max size of messages in Kb on the queue. (memory and disk). |

**Table 84. Historian Queue Object (continued)**

| Counter | Description |
|---|---|
| Size Kb (Mem&Disk) | Size of messages in Kb on the queue. (memory and disk) |
| Size Kb (Mem&Disk) | Size of messages in Kb on the queue. (memory only) |
| Threads | Number of worker threads allocated to process this queue. This is the number of created threads but they may be idle. |
| Threads Working | Number of queue processing worker threads currently processing messages. |
| Time in Queue (Ave) | Average time in milliseconds that a message was in the queue, waiting to be processed. |
| Time in Queue (Last) | Time in milliseconds that the last message was in the queue, waiting to be processed. |
| Time in Queue (Max) | Maximum time in milliseconds that a message was in the queue, waiting to be processed. |

**Table 85. Historian Config Counters**

| Counter | Description |
|---|---|
| File Size | The size of the Configuration File in MB |
| Hist Tags(Actual) | Number of the Historical tags in the system |
| Hist Tags (Licensed) | Total licensed Historian tags. |
| Hist Tags (Used) | Effective number of Historical Licensed Tags in the system. Can be greater than the number of tags because some tags count as more than one Licensed Tag. |
| Hist Tags (UsedByArrays) | Effective number of Historical Licensed Array Tags in the system (Not the raw tag count, the effective licensed count). |
| Hist Tags (UsedByUserDef) | Effective number of Historical Licensed User Defined Tags in the system (Not the raw tag count, the effective licensed count). |
| Number of Collectors | Number of collectors defined on the system. |
| Number of EnumSets | Number of enumerated sets defined on the system. |
| Number of UserDefTypes | Number of user defined types defined on the system. |

**Table 85. Historian Config Counters (continued)**

| Counter | Description |
|---|---|
| SCADA Tags (Actual) | Number of SCADA Tags in the system. |
| SCADA Tags (Licensed) | Total Licensed SCADA tags. |
| SCADA Tags (Used) | Effective number of SCADA Licensed Tags in the system. Can be greater than the number of tags because some tags count as more than one Licensed Tag. |
| SCADA Tags (UsedByArrays) | Effective number of SCADA Licensed Array Tags in the system (Not the raw tag count, the effective licensed count). |
| SCADA Tags (UsedByUserDef) | Effective number of SCADA Licensed User Defined Tags in the system (Not the raw tag count, the effective licensed count). |

# Troubleshooting

## Solve Minor Operating Problems

The following is a table of troubleshooting tips for solving minor operating problems with Historian.

| Issue | Suggested Action |
|---|---|
| After setting the system clock back, browsing the collector from Historian Administrator produces a Visual Basic script error. | Delete temporary Internet files and restart Internet Explorer. |
| With the Historian Administrator, switching usernames causes the system to reject the login if the User must change password at next login option is selected at time of user creation. | New users with this setting must log in to the appropriate Windows operating system at least once. If the login attempt fails, run Historian Administrator as an administrator, and log in with a new username and password. |
| A long error message from the File collector does not fit within the message field in the Historian File collector console window, | Open the File collector log file to see the complete text of the error message. |

| Issue | Suggested Action |
|---|---|
| resulting a partial display of the message. | |
| Excel Sample Reports do not display data. | When opening a Sample Excel report, you may receive a message prompting you to update all linked information in the workbook (Yes) or keep the existing information (No). It is recommended that you select No and keep the existing information. The links will be automatically updated for your worksheet. Save your worksheet after the links have been updated. |
| Need to connect an Historian Server to an Historian Client through a firewall. | Open port 14000 to enable client to server connection through a firewall. |
| Receiving archive offline failed writes messages. | These occur when the timestamps of data being sent to the archiver are not in the valid time range of any online archives. For example, failed writes occur when the timestamps appear *before* the oldest archive, the archive is offline, or timestamps are more than 15 minutes *past* the current time on another archiver. |

## FAQ: Run a Collector as a Service

The following list is frequently asked questions about running a collector as a service.

**Can all collectors be run as a Windows service? If not, which ones cannot?**

The OPC Collector, File collector, Simulation collector, Calculation collector, and Server-to-Server collector can be run as services. The iFIX collector run as a background task and cannot be run as a service.

**Can all collectors be run as an application? If not, which ones cannot?**

All collectors can run as applications (console programs). This includes the Simulation Collector. To make a collector run as a console program, pass a RUNASDOS command line parameter.

**What does "running as a service" mean?**

It means that the collector appears in the Control Panel list of services. It can run at system boot or be run with a different username and password from the currently logged-in user.

**How can the iFIX collector be set up to run when no one is logged in?**

It can be set up to run without a user login by adding it to the iFIX SCU task list as a background task and by configuring iFIX to continue running after logoff in local startup.

**How do you shut down a collector running as console application?**

Collectors started as console applications should be shut down by typing S at the command prompt in the DOS window and pressing Enter.

**Can a collector be run as a Windows service and then stopped and restarted?**

Yes. Collectors that can run as a service can be stopped and started in Control Panel Services. They can be paused/resumed through Historian Administrator.

**What is the difference between running a collector to start as a service on boot up using the Services applet in Control Panel versus running iFIX as a service, which starts the collector through the startup task configuration in the SCU?**

The collectors that can run as a service would not be started from iFIX. They can be started from the Control Panel start at system boot. Although you cannot run an iFIX collector as a service, you can log off and on while it is running.

## Changing the Base Name of Automatically Created Archives

When the IHC file is created, it stores the name of the server inside the IHC file. Automatically created archives use that server name from the IHC file as a base name, not the Base Archive Name configured in Historian Administrator.

When you *manually* create an archive, however, the archive uses the Base Archive Name from Historian Administrator.

If you move an IHC file from one machine to another, you may want to change the default base archive name to match the new server. To change the default Base Archive Name, create a new .IHC file.

This topic describes how to change the base name of an archive using Excel Add-in. You can also change the base name using Configuration Hub *(on page 441)*.

1. Export your tags using the Excel Add-in.
   The Fields to Export window appears.
2. Select all tag attributes and select **OK**.
   The data is exported to a new Excel worksheet.
3. Examine the **Comments** column in the new worksheet. To ensure a clean import, the Comments column must be completely full or completely empty. If no comments are found, this column must

be deleted to ensure a clean import. If only some comments are missing, the missing fields must be filled out with comments.
4. Save the Excel spreadsheet.
5. Stop the Data Archiver service.
6. Open the default archive path in Windows Explorer and rename the .IHC file.
   Rename `MyMachine.IHC` to `MyMachine.OLD`.
7. Restart the Data Archiver service.
   A new, blank IHC file is created for the machine.
8. Import your tags to this new configuration using the Excel Add-In.

## Configuring the Inactive Timeout Value

Historian Administrator offers a configurable timeout. This configurable timeout determines how long Historian Administrator will wait before severing its connection to an inactive Historian archive. The default timeout value is 90 seconds.

1. Assuming Historian is already open, double-click on the **Main** button to open Historian Administrator Login window.
2. Select the **Browse for Server** button.
3. Select the Historian server you wish to configure from the **Servers** list.
4. In the **Connection Timeout** field, select the **Use Value** option and enter a timeout value in seconds.

## Configuring Deep Data Tree Warnings

Reading and writing to deep trees with large time ranges can be very inefficient. Create a **MaxIndexRecursionDepth** registry key to configure the depth at which the archiver will warn about deep data trees.

1. From the **Start** menu, select **Run**, and then enter `Regedit`.
2. Open the following key folder: `HKEY_LOCAL_MACHINE\SOFTWARE\Intellution, Inc.\iHistorian\Services\DataArchiver`
3. Create a value as DWORD called **MaxIndexRecursionDepth**.
4. Set **MaxIndexRecursionDepth** to a number higher or lower than the default value of 900.
   To get more warning messages, set a smaller number if you have an archive which is 10 to 100 deep.
5. Select **OK**.
6. Close the Registry Editor.
7. Restart the archiver for the changes to take effect.

## Control Data Flow Speeds

**Configure buffer flush speed with the BufferFlushMultiplier key**

Store-and-forward buffering is a key feature of Historian collectors. It prevents data loss during planned or unplanned network outages between a collector and Historian server.

If the collector is disconnected from the archiver for several hours or days, many megabytes of data can be buffered and must be delivered by the collector to the Data Archiver upon reconnect. Since all data is sent from the collector to the archiver in time order, the design goal has been to catch up to real time as quickly as possible by sending data as fast as possible.

If this is not the desired behavior because you want to limit the network load on a slow, shared Wide Area Network (WAN) or you want to limit the CPU load on the Data Archiver caused by the incoming data, you can configure the collector to throttle the data it is sending. Throttling Store and Forward does not affect the Alarms and Events collector.

**Important:**
Because data is sent in time order (oldest first), you will not be able to retrieve current historical data until the throttled flush is complete.

**To configure the buffer flush speed using Configuration Hub:**

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Systems**.

   A list of systems appears in the main section.
3. Right-click the system associated with the collector whose buffer flush speed you want to configure, and then select **Advanced Settings**.

   The advanced settings of the system appear, displaying the **Server** section by default.
4. Expand **Collector**, and then select the collector whose settings you want to configure.

   The fields specific to the selected collector appear.
5. In the **BUFFER FLUSH MULTIPLIER** field, select the multiplier to the buffer flow speed while using the store-and-forward feature:

    - **0**: Select this option if you want to disable throttling.
    - **1**: Select this option if you want normal speed.
    - **2**: Select this option if you want the collector to never send data faster than twice the normal speed.
6. Select **Save**.

   A message appears, asking you whether you want to save and restart the collector as well.
7. If you want to save your changes and restart the collector as well, select **Save and Restart**. If you want to just save your changes, select **Save**. In that case, you must restart the collector later for the changes to reflect.

   Your changes are saved. If you have selected **Save and Restart**, the collector is restarted.

**To configure the buffer flush speed without using Configuration Hub:** Configuring the throttle without using Configuration Hub requires modifying a registry key, so it should be done with caution.

A DWORD registry key called **BufferFlushMultiplier** is present under each collector. For Windows 32-bit, it is located under `HKey_Local_Machine\Software\Intellution, Inc.\iHistorian\Services\YOUR_COLLECTOR_TYPE`. For Windows 64-bit, it is located under `HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Intellution, Inc.\iHistorian\Services\YOUR_COLLECTOR_TYPE`.

- To **slow** the store and forward throttling, set the value of **BufferFlushMultiplier** to 2. The 2 means that the collector should never send data at more than 2 times its normal rate to limit network and CPU load.
- To **disable** throttling, set the value of **BufferFlushMultiplier** to 0 or delete the registry key.

**Control archiver speed using Configuration Hub:**

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Systems**.

   A list of systems appears in the main section.
3. Right-click the system associated with the collector whose archive speed you want to configure, and then select **Advanced Settings**.

   The advanced settings of the system appear, displaying the **Server** section by default.
4. Expand **Collector**, and then select the collector whose settings you want to configure.

   The fields specific to the selected collector appear.

5. In the **NUM INTERVALS FLUSH** field, specify how quickly you want the collector to send data to Data Archiver. The value you enter in this field is multiplied by 100 milliseconds. For example, if you enter 5, the collector sends data to Data Archiver every 500 milliseconds. We recommend that you enter 5.
6. Select **Save**.

   A message appears, asking you whether you want to save and restart the collector as well.
7. If you want to save your changes and restart the collector as well, select **Save and Restart**. If you want to just save your changes, select **Save**. In that case, you must restart the collector later for the changes to reflect.

   Your changes are saved. If you have selected **Save and Restart**, the collector is restarted.

**Configuring archiver speed without using Configuration Hub:** The NumIntervalsFlush registry key controls how quickly the collector sends data to the archiver. The collector collects from the data source at the user configured rate, but for efficiency it bundles data samples in a single write to archive. By default, the collector will send data to archiver every 2 seconds or 10,000 samples, whichever happens first. Most often, it sends every 2 seconds because the collector is not collecting that many samples that fast.

If you need collected data sent to archiver right away, so that it is available for retrieval or for calculations, use the NumIntervalsFlush registry key.

You will have to create the registry key, as it does not exist by default. Create a DWORD value called **NumIntervalsFlush** under the collector, in the same place as HISTORIANNODENAME and INTERFACENAME. On a 64-bit Windows Operating System, all 32-bit component-related registry keys (such as collectors, Client Tools, and APIs) will be located under `HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Intellution, Inc.\iHistorian\`.

The preferred setting for **Num Intervals Before Flush** is 5. The intervals are 100 millisecond increments. The default of 20 means `(20 * 100msec) = 2000 msec = 2 seconds`. Set the value to 5 and the collector will send every 500msec.

**Note:**
Changes to the registry key do not take effect until the collector is restarted. This setting affects the sending of data whether it was collected polled or unsolicited.

## Configure Inactive Server Reset Timeout

You can configure inactive server connections to reset automatically with the SocketRecvTimeOut registry key. SocketRecvTimeOut configures a timeout that forces the connection to drop and re-establish if no data is received during the specified time. Consider this configuration when your collector goes to status Unknown for long periods of time even when the connection between collector and archiver is good.

Create a DWORD registry key **SocketRecvTimeOut** under the collector where the problem is occurring and set to a value greater than 90 seconds. A typical value would be 300 seconds. If no bytes are received by the collector for 300 seconds, then the network connection will be closed and re-established.

## Historian Errors and Message Codes

When you review errors and messages, for example, in an Historian archiver log file, full descriptions are usually included. If a number appears instead of a description, use the following table to determine the meaning of the error or message.

**Table 86. Historian Error Codes and Messages**

| Number | Description |
|---|---|
| -32 | Operation not permitted |
| -31 | The requested data store was not found |
| -30 | The requested Enumerated Set was not found |
| -29 | A supplied argument is outside the valid range |
| -28 | A supplied argument is NULL |
| -27 | A supplied argument is invalid |
| -26 | Alarms and Events subsystem unreachable |
| -25 | Attempted data delete outside allowed modification interval |
| -24 | Data Retrieval Count Exceeded |
| -23 | Invalid Server Version |
| -22 | Backup Exceeded Space |
| -21 | Calculation Circular Reference |
| -20 | Not Licensed |
| -19 | Duplicate Interface |

**Table 86. Historian Error Codes and Messages (continued)**

| Number | Description |
|---|---|
| -18 | No Value |
| -17 | License: Invalid License DLL |
| -16 | License: Too Many Users |
| -15 | License: Too Many Tags |
| -14 | Invalid Tagname |
| -13 | Write No Archive Available |
| -12 | Write Outside Active |
| -11 | Archive Read Only |
| -10 | Write Archive Offline |
| -9 | Write in Future |
| -8 | Access Denied |
| -7 | Not Valid User |
| -6 | Duplicate Data |
| -5 | Not Supported |
| -4 | Interface Not Found |
| -3 | Not Connected |
| -2 | API Timeout |
| -1 | FAILED |
| 0 | OK |
| 0 | Undefined |
| 1 | Connection Successful |
| 2 | Connection Unsuccessful |
| 3 | Audited Write |
| 4 | Audited Write Update |
| 5 | Audited Write Out Of Order |

**Table 86. Historian Error Codes and Messages (continued)**

| Number | Description |
|---|---|
| 6 | Audited Write Update Out Of Order |
| 7 | Message On Update |
| 8 | Message On Update Out Of Order |
| 9 | License Library Function Missing |
| 10 | License Library Missing |
| 11 | Failed Write |
| 12 | Tag Added |
| 13 | Tag Modified |
| 14 | Tag Deleted |
| 15 | Interface Added |
| 16 | Interface Modified |
| 17 | Interface Deleted |
| 18 | Archive Added |
| 19 | Archive Add Failure Time Overlap |
| 20 | Archive Deleted |
| 21 | Archive Overwritten |
| 22 | Archive Backup Began |
| 23 | Archive Backup Failed |
| 24 | Archive Backup Completed |
| 25 | Archive Five Days Till Closing |
| 26 | Archive Three Days Till Closing |
| 27 | Archive One Day Till Closing |
| 28 | License Key Removed |
| 29 | License Max Tags Exceeded |
| 30 | License Max Users Exceeded |

**Table 86. Historian Error Codes and Messages (continued)**

| Number | Description |
|---|---|
| 31 | License Max Tags Exceeded Shutdown |
| 32 | License Max Users Exceeded Shutdown |
| 33 | License Library Invalid |
| 34 | Buffer Normal |
| 35 | Buffer On Disk |
| 36 | Buffer Out Of Space |
| 37 | Incomplete Shutdown |
| 38 | Archive Modified |
| 39 | License Expired |
| 40 | Buffer Could Not Create |
| 41 | Archiver Startup |
| 42 | Archiver Shutdown |
| 43 | Audit Status Changed |
| 44 | Option Modified |
| 45 | Write Processing Stopped |
| 46 | Write Processing Resumed |
| 47 | Interface Status Unknown |
| 48 | Archive Closed |
| 49 | Interface Stopped |
| 50 | Interface Started |

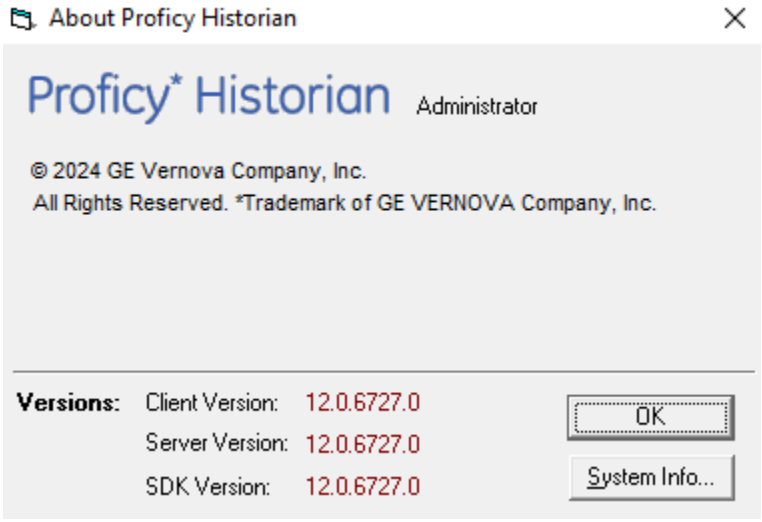
## Determining the Version of the Historian Server

The version of the Historian Server that Historian Administrator is connected to appears in the About window. If the version number is "Unknown", you are connected to an Historian v1.0 Server.

1. Open Historian Administrator.
2. Select the About link.

The About Historian window appears.

3. Note the version number that appear in Versions section of the About Historian window, as shown in the following figure.




## Return a List of Valid Field Options

This topic describes how to determine the fields that are available for an SDK object.

1. Open Microsoft Visual Basic for Applications (VBA).
2. Ensure that you have selected the **Historian Software Kit** reference in the **References** window.
3. Select the **Object Browser** from the **View** menu.
4. Select **Historian_SDK** from the **Library** drop-down list.
5. Select an item in the **Classes** list to generate a list of valid fields for that item.
   when you select **ihCollectionType**, you see a list of fields that includes **Polled** and **Unsolicited**.

## Scheduled Software Performance Impact

Running continuous backup or disk scan software applications such as anti-virus scans, automated backup software, or any other software that accesses disk drives to a high degree may affect the overall performance of your Historian System by competing with Historian for disk resources.

If your Historian System requires that you need an extremely high throughput (20k/sec or greater), consider **disabling** the scheduled software execution.

## Intellution 7.x Drivers as OPC Servers

The ABR and the ABC drivers are OPC v2.0 compliant. All other Intellution 7.x drivers, including the MB1, support OPC v1.0 compliance.

Version 7.x drivers also comply with the OLE for Process Control (OPC) v1.0a standard. Any 1.0-compliant OPC client application can access process hardware data through the I/O Server.

The ABC I/O Server also supports the v2.0 standard and with the OPC Alarms and Events v1.0 specification. Refer to Using OLE for Process Control (OPC) Functionality to learn more about the advantages of OPC.

## Troubleshooting Failed Logins

The following is a table of error messages, possible causes, and recommended corrective actions for failed logins sometimes experienced with Historian Administrator.

| Error Message | Suggested Action |
|---|---|
| User does not have authority to read messages. | The user is NOT a member of iH Readers nor a member of the iH Security Admins security groups. To access the Main Page, the user must have read access. |
| [07/18/2001 03:00:46.071 PM] USER: DOMAIN1\administrator TOPIC: Security MSG: DOMAIN1\administrator(administrator) unsuccessfully connected at 07/18/2001 03:00:46.071 PM. Not able to establish session to the server from a remote Web-based Clients. Page cannot be displayed. | Error in Internet Explorer. Check network connection or IIS on the server. |
| [07/17/2001 07:56:06.950 PM] USER: DataArchiver TOPIC: Security MSG: DataArchiver(DataArchiver) Exceeded number of licensed users at 07/17/2001 07:56:06.950 PM (NumUsers=0 MaxUsers=0) | Outdated or failed HASP key is attached. Obtain new key from technical support. |
| [07/17/2001 07:58:18.980 PM] USER: \baduser TOPIC: Security MSG: \baduser(baduser) | Bad password for user account. Enter correct password. |

| Error Message | Suggested Action |
|---|---|
| unsuccessfully connected at 07/17/2001 07:58:18.980 PM. | |
| [07/17/2001 07:58:48.712 PM] USER: \administrator TOPIC: Security MSG: \administrator(administrator) unsuccessfully connected at 07/17/2001 07:58:48.712 PM. | DataArchiver service is not running. Results in a Failed to connect to server error. Make sure data archiver service is running and that the user did not enter a bad password. |
| [07/17/2001 07:23:44.397 PM] USER: DataArchiver TOPIC: Security MSG: DataArchiver(DataArchiver) Exceeded number of licensed tags at 07/17/2001 07:23:44.397 PM (NumTags=1021 MaxTags=0) Must Shutdown | Number of configured tags exceeds number of tags allowed by the key. Delete enough tags to meet the license limit or obtain a license for more tags from technical support. |
| [07/17/2001 07:35:32.134 PM] USER: DataArchiver TOPIC: Security MSG: DataArchiver(DataArchiver) Licensed expired. Must shutdown 07/17/2001 07:35:32.134 PM. To troubleshoot, refer to the DataArchiver.log file. | License on key has expired (archiver will not start). Obtain a new license from technical support. |

## Troubleshoot Data Collector Configuration

Troubleshooting Data Collector configuration and/or performance requires a thorough understanding of how Historian works and how the various parameters affect system operation. Armed with this knowledge, you can usually localize a problem to one or more functions or parameter settings and take effective corrective action.

Historian offers several tools to help find the cause of an operating problem.

**LOG files**

The system creates a new log file each time an archiver or collector is started. You can open these files in Notepad or another text editor. The `-nn` suffix in the file name indicates the place of each log file in the time sequence.

The data collector log files, located in the `LogFiles` folder within the Historian program folder, are a historical journal of every event affecting operation of the collector.
The `DataArchiver-nn.LOG` files are sequential files for the archiver only.
The `iFIX collector-nn.LOG` files contain performance information on iFIX collector functioning.

**SHW file**

The Data Collector .SHW file shows configuration data for collectors and is also located in the `LogFiles` directory under Historian. Verify that the parameter and configuration settings match what you configured. You can open this file in Notepad or another text editor.

**Collector Maintenance Performance Indicators**

These performance and status indicators can be a major aid in identifying, localizing, and diagnosing a problem with a collector.

**Collector Maintenance pages**

The Collector Maintenance pages can provide useful information about settings and selections of various options and parameter values. Examine each field and verify that it is appropriate for the current application.

**iHistorianSDKerrors.log**

When performing any troubleshooting of the File collector Administrator or SDK program you should examine the `iHistorianSDKerrors.log` file, which is located in the `LogFiles` folder in your Historian program folder.

## Troubleshoot Tags

**Tag Configuration**

To diagnose a problem in tag configuration, examine the .LOG and .SHW files in the Historian/Logfiles directory. Since these files are a journal record of all system events and parameter modifications important to a system administrator, they can be helpful in identifying and localizing the source of a system malfunction or data error.

You may also find the Windows Event Viewer helpful in troubleshooting. For more information on the Windows Event Viewer, refer to Review System Alerts and Messages *(on page 995)*.

When troubleshooting SDK applications, such as Historian Administrator, the Excel Add-In, or File collector, examine the `iHistorianSDKerrors.log` file, located in the `LogFiles` folder in the Historian program folder.

**Stale Tags**

If you are not seeing all stale tags in the Historian Web Admin, the ClientManager service may be down. If so, the thread that processes stale tags is suspended until connection is restored.

If the connection to the ClientManager service is lost while processing a batch of 100 tags, that set of tags is skipped and a Tag Property Update error is logged. When the ClientManager service is back up again, the **next** set of 100 tags is considered for scanning. The skipped tags must wait until the Diagnostics Manager's next cycle. Until then, they are not marked as stale or not stale.

Alternately, the DataArchiver or ConfigManager services may be down. If so, all the remaining tags are skipped, and a DataOpenRecordset/Tag Update failure is logged. Even though the thread that processes stale tags is not suspended, all the remaining tags have to wait until the Diagnostics Manager's next cycle. Until then, they are not marked as stale or not stale.

## Troubleshoot Historian Performance

There are many parameters that affect Historian Server performance and scalability. This topic emphasizes the parameters that help improve performance and scalability.

**Buffer Memory Max**

Buffer Memory Max allows you to specify the maximum memory buffer size that an archiver queue can take. Use this parameter to control memory allocation to the write queues. If you plan to collect large amounts of data with high update rates, increase the Buffer Memory Max size. Increasing the Buffer Memory Max size provides enough memory for write queues, thus increasing the Historian performance. Refer to The Global Options Section *(on page 843)* for more information.

**Performance Counters**

Historian provides a variety of performance counters that can be used to monitor how well the application is performing. Using the following performance counters you can determine the Historian Server performance and scalability:

- Perftag_GenericWriteQueueSize: displays the total number of messages present in the Generic Write queue.
- Perftag_FastReadQueueSize: displays the total number of messages present in the Fast Read queue.
- Perftag_CollectorDataWriteQueueSize: displays the total number of messages present in the Collector Data Write queue.
- Perftag_EventMonitorQueueSize: displays the total number of events present in the Event Monitor queue.
- Perftag_DataWriteQueueSize: displays the total number of messages present in the Data Write queue
- Perftag_ReadQueueSize: displays the total number of messages present in the Read queue

By monitoring the above performance counters, make sure that you manage application performance. Ideally, all write queues should be zero for optimum archiver performance. If you see an increase in back logs in write queues, then you can:

- Control archive transitions by having larger archive file size.
- Control update rate.
- Perform a backup when an archiver is in stable state.
- Control the caching evaluation period. Refer to Historian Performance Counters *(on page 1002)* for more information.

**System File Cache Tuning**

Historian allows you to specify the maximum disk cache memory that an archiver can use. Ideally, Historian consumes 25% of system memory. If your computer has extra memory, then you can increase the disk cache memory to optimize Historian performance.

**Note:**
You should not increase disk cache memory if you do not have the necessary system resources.

## Troubleshoot the Archive Service

**Archiver cannot create a new archive:** If an archive is full and you have disabled Overwrite Old Archives in the **Global Options** section, Historian can only create a new archive if there is enough disk space available. If you do not have enough disk space to create a new archive, Historian does not generate an alert on the Main page, but instead adds the following messages to the Data Archiver log file: `Could`

```
not locate empty archive file, nor create one, nor use old one. Stopped processing data write
requests. Free up disk space or enable Overwrite Old Archives to continue archiving your data.
```

**Archive service not starting automatically:** On some systems the Archiver does not start automatically after install. Start the Archiver manually through the Services window or restart the computer. The Archiver should start on all future restarts. The Archiver service may also fail to start if it is started by an improperly configured domain account. In order to add a user from the domain as the Archiver service start up account, the machine Historian is installed on must have joined the domain. The user changing the service log on account must also have administrative rights to the Historian machine. This user the service is configured with must be a member of the Domain Administrators group and have administrative rights to the Historian machine. The user also needs to be allowed to log on as a service and act as part of the operating system. This is configured in the local policy editor of the Historian machine.

**Investigate failed backups:** Refer to the `ArchiveBackup-XX.log` file to investigate failed backups. The backup can fail if the user does not have valid permissions or if the archive name supplied is not valid. For more information, refer to Knowledge Base article: i014491 at http://iglobalcare.gefanucautomation.com.

# Chapter 6. Historian Advanced Topics

## Historian Advanced Topics Overview

### About Historian Advanced Topics

The intended audience for this content is a user with a high level of computing and technical skills, specifically with Microsoft Windows and associated networking products. It is also assumed that the user will have prior experience working with the Historian product.

This Help describes advanced content on the typical flow of data in a Historian system. It also describes advanced functions which you can run at the command line for data stores.

- **Data Input**: describes the time up until the data is sent over the network to the archiver.
- **Storage**: covers the time the archiver receives it until it is requested.
- **Retrieval**: covers the whole round trip from the client to the archiver and back with the requested data.

Because Historian can store and retrieve data, comments, and messages, the input, storage, and retrieval of each is discussed.

Buffering applies to both input and storage and is mentioned in those sections.

There are additional software tools introduced in this document that do not ship with the product.

Contact your technical support agent about such tools, should you require them.

## Storage

### Archive Compression

**Archive Compression Overview**

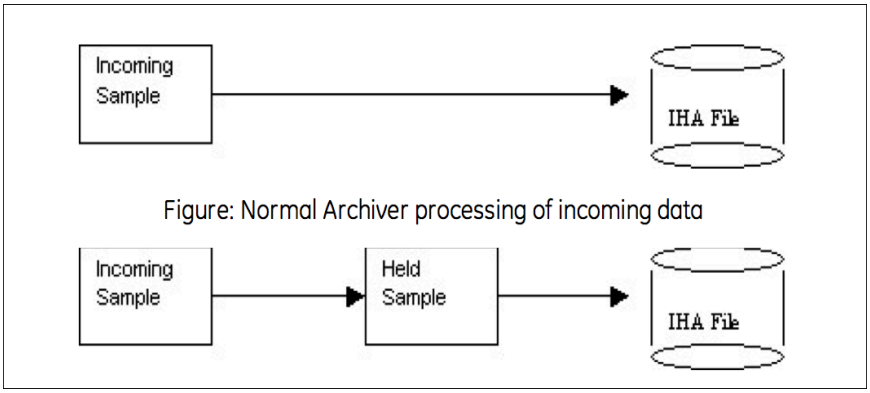
The Data Archiver performs archive compression procedures to conserve disk storage space. Archive compression can be used on tags populated by any method (collector, migration, File collector, SDK programs, Excel, etc.)

Archive compression is a tag property. Archive compression can be enabled or disabled on different tags and can have different deadbands.

Archive compression applies to numeric data types (scaled, float, double float, integer and double integer). It does not apply to string or blob data types. Archive compression is useful only for analog values, not digital values.

Archive compression can result in fewer raw samples stored to disk than were sent by collector.

If all samples are stored, the required storage space cannot be reduced. If we can safely discard any samples, then some storage space can be reduced. Briefly, points along a straight or linearly sloping line can be safely dropped without information loss to the user. The dropped points can be reconstructed by linear interpolation during data retrieval. The user will still retrieve real-world values, even though fewer points were stored.




Archive Compression uses a held sample. This is a sample held in memory but not yet written to disk. The incoming sample always becomes the held sample. When an incoming sample is received, the currently-held sample is either written to disk or discarded. If the currently-held sample is always sent to disk, no compression occurs. If the currently-held sample is discarded, nothing is written to disk and storage space is conserved.

In other words, collector compression occurs when the collected incoming value is discarded. Archive compression occurs when the currently-held value is discarded.

Held samples are written to disk when archive compression is disabled or the archiver is shut down.

Any sample written to disk is a true incoming sample. No timestamp or value or quality is ever changed or interpolated.

**Note:**
internal performance tags use an archive compression deadband of 0% or close to 0%.

**Archive Compression Logic**

The following describes the logic that is executed on every sample written to the archiver while archive compression is enabled for the tag:

```
IF the incoming sample data quality = held sample data quality

IF the new point is a bad

Toss the value to avoid repeated bads. Do we toss new bad or old bad?

ELSE

Decide if the new value exceeds the archive compression deadband/

ELSE//

data quality is changed, flush held to disk

IF we have exceeded deadband or changed quality

// Store the old held sample in the archive

// Set up new deadband threshold using incoming value and value written to disk.


// Make the incoming value the held value
```

### Example: Change of data quality

The effect of archive compression is demonstrated in the following examples.

This example demonstrates that:

- A change in data quality causes held samples to be stored.
- Held samples are returned only in a current value sampling mode query.
- Restarting the archiver causes the held sample to be flushed to disk.

Normally, a flat straight line would never cause the held value to be written to disk. An important exception is that changes in data quality force the held value to be written to disk. Assume a large archive compression deadband, such as 75% on a 0 to 100 EGU span.

| Time | Value | Quality |
|---|---|---|
| t) | 2 | Good |
| t1 | 2 | Bad |
| t2 | 2 | Good |

The following SQL query lets you see which data values were stored:

```
Select * from ihRawData where samplingmode=rawbytime and tagname = t20.ai-1.f_cv and timestamp > today
```

Notice that the value at t2 does not show up in a RawByTime query because it is a held sample. The held sample would appear in a current value query, but not in any other sampling mode:

```
select * from ihRawData where samplingmode=CurrentValue and tagname = t20.ai-1.f_cv
```

The points should accurately reflect the true time period for which the data quality was bad.

Shutting down and restarting the archiver forces it to write the held sample. Running the same SQL query would show that all 3 samples would be stored due to the change in data quality.

### Archive Compression of Straight Line

In this case we have a straight flat line. Assume a small archive compression deadband, say 2% on a 0 to 100 EGU span. Since data occurs on a straight flat line, the deadband will never be exceeded.

| Time | Value | Quality |
|---|---|---|
| t0 | 2 | Good |
| t0+5 | 2 | Bad |
| t0+10 | 2 | Good |
| t0+15 | 2 | Good |
| t0+20 | 2 | Good |

Shut down and restart the archiver, then perform the following SQL query:

```
select * from ihRawData where samplingmode=rawbytime and tagname = t20.ai-1.f_cv and timestamp > today
```

Only t0 and t0+20 were stored. T0 is the first point and T0+20 is the held sample written to disk on archiver shutdown, even though no deadband was exceeded.

### Bad Data

This example demonstrates that repeated bad values are not stored. Assume a large archive compression deadband, such as, 75%.

| Time | Value | Quality |
|---|---|---|
| t0 | 2 | Good |
| t0+5 | 2 | Bad |

| Time | Value | Quality |
|---|---|---|
| t0+10 | 2 | Bad |
| t0+15 | 2 | Bad |
| t0+20 | 2 | Good |
| t0+25 | 3 | Good |

- The t0+5 value is stored because of the change in data quality.
- The t0+10 value is not stored because repeated bad values are not stored.
- The t0+15 value is stored when the t0+20 comes in because of a change of quality.

### Disabling Archive Compression for a Tag

Assume a large archive compression deadband, such as 75%

| Time | Value | Quality |
|---|---|---|
| t0 | 2 | Good |
| t0+5 | 10 | Good |
| t0+10 | 99 | Good |
| t0+15 | Archive compression disabled | |

- The t0 value is stored because it is the first sample.
- The t0+5 is stored when the t0+10 comes in.
- The t0+10 is stored when archive compression is disabled for the tag.

### Archive Compression of Good Data

This example demonstrates that the held value is written to disk when the deadband is exceeded.

In this case, we have an upward ramping line. Assume a large archive compression deadband, such as 75% on a 0 to 100 EGU span.

| Time | Value | Quality |
|---|---|---|
| t0 | 2 | Good |
| t0+5 | 10 | Good |
| t0+10 | 10 | Good |

| Time | Value | Quality |
|---|---|---|
| t0+15 | 10 | Good |
| t0+20 | 99 | Good |

Shut down and restart the archiver, then perform the following SQL query:

```
select * from ihRawData where samplingmode=rawbytime and tagname = t20.ai-1.f_cv and timestamp > today
```

Because of archive compression, the t0+5 and t0+10 values are not stored. The t0+15 value is stored when the t0+20 arrives. The t0+20 value would not be stored until a future sample arrives, no matter how long that takes.

## Determining Whether Held Values are Written During Archive Compression

When archive compression is enabled for a tag, its current value is held in memory and not immediately written to disk. When a new value is received, the actual value of the tag is compared to the expected value to determine whether or not the held value should be written to disk. If the values are sufficiently different, the held value is written. This is sometimes described as "exceeding archive compression".

Archive compression uses a deadband on the slope of the line connecting the data points, not the value or time stamp of the points themselves. The archive compression algorithm calculates out the next expected value based on this slope, applies a deadband value, and checks whether the new value exceeds that deadband.

The "expected" value is what the value would be if the line continued with the same slope. A deadband value is an allowable deviation. If the new value is within the range of the expected value, plus or minus the deadband, it does not exceed archive compression and the current held value is not written to disk. (To be precise, the deadband is centered on the expected value, so that the actual range is plus or minus half of the deadband.)

### Exceeding Archive Compression

EGUs are 0 to 200000 for a simulation tag.

Enter 2% Archive compression. This displays as 4,000 EGU units in the administration UI.

When a sample arrives, the archiver calculates the next expected value based on the slope and time since the last value was written. Let's say that the expected value is 17,000.

The deadband of 4,000 is centered, so the archiver adds and subtracts 2,000 from the expected value. Thus, the actual value must be from 15,000 to 19,000 inclusive for it to be ignored by the compression algorithm.

In other words, the actual value must be less than 15,000 or greater than 19,000 for it to exceed compression and for the held value to be written.

## Determining Expected Value

The Archive Compression algorithm calculates the expected value from the slope, time, and offset (a combination of previous values and its timestamp):

```
ExpectedValue = m_CompSlope * Time + m_CompOffset;
```

Where

```
m_CompSlope = deltaValue / deltaT

m_CompOffset = lastValue - (m_CompSlope * LastTimeSecs)
```

### Determining Expected Value

Values arriving into the archiver for tag1 are

| Time | Value |
|---|---|
| t0 | 2 |
| t0+5 | 10 |
| t0+10 | 20 |

The expected value at time stamp t0+15 is calculated based on the samples at t0+5 and t0+10:

```
m_CompSlope = deltaValue / deltaTime m_CompSlope = (20-10) / 5

m_CompSlope = 2

m_CompOffset = lastValue - (m_CompSlope * LastTimeSecs)

m_CompOffset = 20 - (2 * 10)

m_CompOffset = 0

ExpectedValue = m_CompSlope * Time + m_CompOffset;

ExpectedValue = 2 * 15 + 0;

ExpectedValue = 30
```

The expected value at t0+15 is 30.

### Archive Compression of a Ramping Tag

An iFIX tag is associated with an RA register. This value ramps up to 100 then drops immediately to 0.

Assume a 5-second poll time in Historian. How much archive compression can be performed to still "store" the same information?

With an archive compression of 75%, 25% or 5%, only the change in direction is logged:

```
11-Mar-2003 19:31:40.000 0.17 Good NonSpecific

11-Mar-2003 19:32:35.000 90.17 Good NonSpecific

11-Mar-2003 19:32:40.000 0.17 Good NonSpecific

11-Mar-2003 19:33:35.000 91.83 Good NonSpecific

11-Mar-2003 19:33:40.000 0.17 Good NonSpecific
```

An archive compression of 1% stores the most samples.

An archive compression of 0% logs every incoming sample. Even on a perfectly ramping signal with no deviations, 0% compression conserves no storage space and essentially disables archive compression.

### Archive Compression of a Drifting Tag

A drifting tag is one that ramps up, but for which the value barely falls within the deadband each time. Even though a new held sample is created and the current one discarded, the slope is not updated unless the current slope exceeded. With a properly chosen deadband value, this is irrelevant: by specifying a certain deadband, the user is saying that the process is tolerant of changes within that deadband value and that they do not need to be logged.

### Archive Compression of a Filling Tank

In the case of a filling tank, the value (representing the fill level) ramps up, then stops. In this case, the system also uses collector compression, so when the value stops ramping, no more data is sent to the archiver. At some point in the future, the value will begin increasing again.

As long as nothing is sent to the archiver, no raw samples are stored. During this flat time (or plateau), it will appear flat in an interpolated retrieval because the last point is stretched. This illustrates that you should use interpolated retrieval methods on archived compressed data.

## How Archive Compression Timeout Works

The *Archive Compression Timeout value* describes a period of time at which the held value is written to disk. If a value is held for a period of time that exceeds the timeout period, the next data point is considered to exceed the deadband value, regardless of the actual data received or the calculated slope.

After the value is written due to an archive compression timeout period, the timeout timer is reset and compression continues as normal.

## Archive De-fragmentation - An Overview

What is De-fragmentation? Having an IHA file with contiguous data values for a particular tag is called Archive De-fragmentation.

An Historian IHA (Historian Data Archive) file could contain data values for multiple tags and data written for a particular tag may not be in continuous blocks. This means data values in IHA files are fragmented.

Archive De-fragmentation improves the performance of reading and processing of archive data dramatically.

**De-fragmentation of existing archives**:

- An archive can be de-fragmented, when it is not active.
- A command line based tool can be used to run on any existing archive as needed.
- De-fragmentation can be done on all versions of archives, and the resulting archive will be the latest version.
- The de-fragmentation must be started manually.

## De-fragmenting an existing archive

A command line based tool is available to de-fragment an existing archive. This tool can be run on any existing archive(s). After the de-fragmentation, the new archive will be slightly smaller in size than the original one.

To de-fragment an archive using the tool:

1. Select a read-only archive to backup.
2. Backup the archive.
3. Transfer the archive from the production machine to another machine.
4. Run the ihArchiveDefrag tool on the archive.

```
--------------------------------------------------------------------------------

Usage (Single File Mode): Operates on one archive at a time.


ihArchiveDefrag [-v][-d][-y][-h][-s] SrcArchive [DestArchive] [Config.ihc]


--------------------------------------------------------------------------------
```

```
Usage (Directory Mode): Processes all archives in the source directory.


ihArchiveDefrag [-v][-d][-y][-h][-s] SrcDir [DestDir] [Config.ihc]


-------------------------------------------------------------------------------


      [-v]   Verbose logging to the logfile.

      [-c]   Skip the defrag step.  Only compare/verify the archives.

      [-d]   Skip the verify step.  Only defrag the archive.

      [-h]   Displays this help information.

      [-s]   Source is on a SSD.  Skip the pre-read step as it is not needed.

      [-y]   Yes.  Don't ask questions, answer YES.  Just do the work.


  * The Config.ihc is only needed for 4.0 or older archives

  * If the DestArchive is not provided, a name will be generated.

  * In directory mode, if the DestDir is not provided a new directory called 'DefragFiles' will be created

 under the SrcDir.


Examples:


   ihArchiveDefrag User_Node_Archive001.iha

   ihArchiveDefrag User_Node_Archive001.iha D:\Output\User_Node_Archive001.iha

   ihArchiveDefrag -y c:\Historian\Archives\Backups

   ihArchiveDefrag -y c:\Historian\Archives\Backups d:\DefragOuput
```

**Note:**

- As part of this process the source archive MAY be modified, It is HIGHLY recommended that this tool be run on a disposable copy of the archive just in case it is modified.
- De-fragmenting an archive is a disk intensive operation and can be slow (minutes to hours to run). Running on a machine with memory greater than twice the archive size is helpful.
- If the performance is not satisfactory it is recommended that the source archive be on an SSD when running ihArchiveDefrag.
- It is recommended that de-fragmentation should not be run on a production system as it can affect the performance of the production system.

5. Transfer the resulting new archive back to the production machine.
6. Unload the existing archive.
7. Load the new de-fragmented archive.

## About Storing Future Data

You can store future data in Historian. This future data is the predicted data, which has a future timestamp. You can use this data to perform a predictive or forecast analysis, and revise the forecasting algorithms as needed.

The data is stored in the Historian Archiver. You can use it to analyse both the historical data and future data (for example, using trend charts), and take necessary actions. This allows you to refine the way the data to be stored in Historian is received and processed.

By default, Historian stores up to 1200 seconds of future data. However, if you enable storage of future data, you can store data up to the following timestamp: 03:14:07 on Tuesday, 19 January, 2038.

**Note:**
You can store future data beyond the license expiry date of Historian. For example, even if the license expires by May 31, 2022, you can store future data that is predicted till 19 January, 2038. However, after the license expires, you cannot use this feature.

You can store future data related to all the data types used in Historian. You can store future data for all the tags associated with a data store.

The following collectors support future data (that is, collect the future timestamp of the data):

- The OPC DA collector
- The OPCUA DA collector
- The OPCHDA collector

To use this feature, you must enable the storage of future data. You can do this using sample programs *(on page 1039)* or Command Line *(on page 1039)*.

You can then retrieve and/or extract this data using any of the available options such as Historian Administrator, Rest APIs, Excel Add-In, and the Historian Interactive SQL application (ihsql.exe).

The following conditions apply when you store future data:

- You can enable the future data storage only for a data store.
- You can collect future data only using the OPC Data Access, OPCUA Data Access, and OPC HDA collectors. You cannot collect future data using collectors such as the Server-to-Server collector, the Server-to-Server distributor, and the Calculation collector.
- When you select the **Last 10 Values** option for a tag, the results include the last ten values till the current date and time; the results do not include future data. If you want to view future data, you can filter the data based on the start and end dates.
- Future data is stored till 19 January, 2038.

> **Note:**
> You can store future data beyond the license expiry date of Historian. For example, even if the license expires by May 31, 2022, you can store future data that is predicted till 19 January, 2038. However, after the license expires, you cannot use this feature.

**Best practices:**

- For a tag for which you want to store future data, do not store past data.
- As this feature can lead to out-of-order data, use this feature carefully to avoid load on the server. For example, store future data in the chronological order of time.

**Known Issues:**

- Data recalculation does not work for future data.

Suppose you have enabled the storage of future data for a data store named FDataStore.

Suppose the current time is 9.00 am, April 13, 2020. And, future data is stored from 11.00 am onwards, and a size-based archive is created to store the data.

Data can be stored in the archive file only from 11.00 am onwards.

**Scenario 1: The timestamp of the data is in the past.** For example: 11.00 am, April 12, 2020. A new archive file will be created to store this data. Therefore, to reduce load on the server, we strongly recommend that you store future data in the chronological order of time.

**Scenario 2: The timestamp of the data is beyond the outside-active-hours of the archive file.** For example: 11.00 am, December 12, 2020. Suppose the current archive file is active only for today. Data will not be stored in the archive file because the timestamp of the data falls beyond the outside-active-hours value of the archive file. To avoid this issue, a new archive file must be created in such scenarios. To do so, you must enable offline archive creation *(on page 1043)*.

**Scenario 3: The timestamp of the data is much further in the future.** For example: 11.00 am, April 12, 2025. The current archive file may be used to store the data for this timestamp as well. This results in an optimum usage of the archive file instead of creating multiple archive files.

## Enable Storing Future Data Using Configuration Hub

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Systems**.
   A list of systems appears in the main section.
3. Right-click the system associated with the data store for which you want to enable storing future data, and then select **Advanced Settings**.
   The advanced settings of the system appear, displaying the **Server** section by default.
4. Expand **Datastore**, and then, select the data store in which you want to store future data.
   The fields specific to the selected data store appear,
5. Switch the **ALLOW FUTURE DATA** toggle on.
6. As needed, enter values in the remaining fields, and then select **Save**.
   Storing future data is now enabled for the data store.

## Enable Storing Future Data Using Command Line

Ensure that you have a mechanism to generate future data that you want to store in Historian.

By default, Historian stores up to 1200 seconds of future data. However, if you enable storage of future data, you can store data up to the following timestamp: 03:14:07 on Tuesday, 19 January, 2038. This topic describes how to enable storage of future data using Command Line. You can perform this task for any data store other than the default USER data store.

1. Stop the Historian DataArchiver service.
2. Open Command Prompt with elevated privileges or administrator privileges.
3. Navigate to the folder in which the `ihDataArchiver_x64.exe` file is located. By default, it is `C:\Program Files\Proficy\Proficy Historian\x64\Server`.
4. Run the following command: `ihDataArchiver_x64 OPTION.<data store name> ihArchiverAllowFutureDataWrites 1`
5. Run the following command: `ihDataArchiver_x64 OPTION.<data store name> ihArchiveActiveHours <number of active hours>`
6. Start the Historian DataArchiver service.

## Enable Storage of Future Data

Ensure that you have a mechanism to generate future data that you want to store in Historian.

By default, Historian stores up to 1200 seconds of future data. However, if you enable storage of future data, you can store data up to the following timestamp: 03:14:07 on Tuesday, 19 January, 2038. This topic describes how to enable storage of future data.

1. For each data store for which you want to store future data:

   a. Enable the `ihArchiverAllowFutureDataWrites` option.

   > **Tip:**
   > You can perform this step using the sample programs *(on page 1040)* or Command Line *(on page 1039)*.

   b. Enable the `ihArchiveCreateOfflineArchive` option.

   This is to avoid receiving an outside-active-hours error. It happens if you attempt to store data when the current archive file is set to read-only.

   > **Tip:**
   > You can perform this step using Command Line *(on page 1043)*.

2. For each collector that sends data to the data store, set the `Time Assigned By` property to `Source`.

> **Note:**
> You can perform this step only for the OPC Data Access, OPCUA Data Access, and OPC Classic HDA collectors.

Future data is now stored in Historian. You can retrieve the data using any of the available options such as Historian Administrator, Rest APIs, Excel Add-In, and the Historian Interactive SQL application (ihsql.exe).

## Sample Program to Enable Future Data

### Using C++

The following lines of code provide a sample program to enable the `ihArchiverAllowFutureDataWrites` option using C++:

```
void SetFutureDataWriteOptions(void)

{
```

```
 ihChar enable[50];
 printf("\n\n Please enter 1/0 for enable/disable future data write per data store: ");
 _getws(enable);
 ihChar dataStoreName[50];
 printf("\n\n Please enter data store name to enable future data writes ");
 _getws(dataStoreName);

 ihAPIStatus Status;
 ihOptionEx option;
 option.Option = ihArchiverAllowFutureDataWrites;
 option.DataStoreName = dataStoreName;
 Status = ihArchiveSetOption(SrvHdl, &option,enable);

}

void GetFutureDataWriteOptions()
{
 ihChar dataStoreName[50];
 printf("\n\n Please enter the name of the data store Future Data write Option:  ");
 _getws(dataStoreName);

 ihAPIStatus  Status;
 ihChar   *Value;
 ihOptionEx temp;

 temp.Option = ihArchiverAllowFutureDataWrites;
 temp.DataStoreName = dataStoreName;

 Status = ihArchiveGetOption(SrvHdl,&temp,&Value);

 printf("Archive Future Data Write Option ihArchiverAllowFutureDataWrites value is  - [%ls]  for the data store
 [%ls]\n", (Value ? Value : L"NULL"), dataStoreName);
 Pause();
}
```

### Using C#

The following lines of code provide a sample program to enable the `ihArchiverAllowFutureDataWrites` option using C#:

```
static void Main(string[] args)
        {
            string swtValue = "0";
            string dsName = "";
            ConsoleKeyInfo kInfo;
            connection = ClientConnect.NewConnection;
            do
            {
                Console.WriteLine("1 Set/Enable Data Store Create Offline Archives Option Value");
                Console.WriteLine("2 Set/Enable Data Store Allow Future Data Writes Option Value");
                Console.WriteLine("3 Get Data Store Create Offline Archives Option Value");
                Console.WriteLine("4 Get Data Store Allow Future Data Writes Option Value");
                Console.WriteLine("Enter Option Value");
                swtValue = Console.ReadLine();
                switch (swtValue)
                {
                    case "1":
                        Console.WriteLine("Enter Data Store Name");
                        dsName = Console.ReadLine();
                        connection.IServer.SetOption(HistorianOption.ServerCreateOfflineArchives, "1", dsName);
                        Console.WriteLine("Option value set to " +
 connection.IServer.GetOption(HistorianOption.ServerCreateOfflineArchives, dsName));
                        Console.ReadKey();
                        break;
                    case "2":
                        Console.WriteLine("Enter Data Store Name");
                        dsName = Console.ReadLine();
                        connection.IServer.SetOption(HistorianOption.ArchiverAllowFutureDataWrites, "1", dsName);
                        Console.WriteLine("Option value set to " +
 connection.IServer.GetOption(HistorianOption.ArchiverAllowFutureDataWrites, dsName));
                        Console.ReadKey();
                        break;
                    case "3":
```

```
                        Console.WriteLine("Enter Data Store Name");
                        dsName = Console.ReadLine();
                        Console.WriteLine("Option value is: " +
connection.IServer.GetOption(HistorianOption.ServerCreateOfflineArchives, dsName));
                        Console.ReadKey();
                        break;
                    case "4":
                        Console.WriteLine("Enter Data Store Name");
                        dsName = Console.ReadLine();
                        Console.WriteLine("Option value is: " +
connection.IServer.GetOption(HistorianOption.ArchiverAllowFutureDataWrites, dsName));
                        Console.ReadKey();
                        break;
                    default:
                        Console.WriteLine("Please enter valid number");
                        Console.ReadKey();
                        break;
                }
                Console.WriteLine("Please Enter X to exit");
                kInfo = Console.ReadKey();

            } while (kInfo.Key != ConsoleKey.X);
        }
    }
```

## Enable Offline Archive Creation

This topic describes how to enable offline archive creation. This is to avoid receiving an outside-active-hours error. It happens if you attempt to store data when the current archive file is set to read-only.

1. Stop the Historian DataArchiver service.
2. Open Command Prompt with elevated privileges or administrator privileges.
3. Navigate to the folder in which the `ihDataArchiver_x64.exe` file is located. By default, it is `C:\Program Files\Proficy\Proficy Historian\x64\Server`.
4. To enable the `ihArchiveCreateOfflineArchive` option for a data store, run the following command: `ihDataArchiver_x64 OPTION.<data store name> ihArchiveCreateOfflineArchive 1`
5. Start the Historian DataArchiver service.

# Retrieval

## Retrieval

When retrieving data from Historian, you specify either a raw or non-raw sampling mode. Non-raw retrieval can include a calculation mode so that calculations are performed in the archiver before data is returned. This is detailed in the following sections:

- Sampling Modes
- Hybrid Modes
- Filtered Data Queries

Some sampling and calculation modes are better suited to retrieving compressed data. Understanding the available modes helps you choose the best method for your archiving process.

The retrieval topics include descriptions of all methods of retrieval:

- API
- SDK
- OLE DB
- Charting
- Reporting via OLE DB and Excel Add In

## Sampling Modes

Many different sampling and calculation modes can be used on retrieval of data that has already been collected in the archive. Available sampling modes in Historian include:

**Note:**
A filtered data query can be performed with each sampling mode except CurrentValue. Calculation modes are used when the sampling mode is set to "Calculated".

These topics explain some of these retrieval concepts. Each sampling mode (except calculated) is described with details and examples, including how sample attributes are determined. Each sample returned by Historian during data retrieval has the following properties:

- **Timestamp** — time stamp of the collected sample or an interval time stamp
- **Value** — The collected value or sampled value
- **Quality** — Each sample in Current Value and Raw retrieval has a quality of "good" or "bad". Interpolated and Lab Retrieval express quality as a "per cent good".

Current value sampling is the simplest retrieval mode. Raw data retrieval is the second simplest method of retrieval. Intervals and interpolation concepts are common to Interpolation and Lab sampling. Interpolation and lab sampling are presented together so that they can be contrasted for values and qualities returned from the same set of collected data.

**Example Data:** Each topic contains all necessary data for executing each example in the form of a CSV file that can be imported by the Historian File collector. You will have to copy and paste the appropriate data into a separate file with a CSV file name extension. Delete all archives before importing the data. You will not be able to import the data unless you adjust the active hours setting; this is true any time you import old data with the File collector. For details, see *Historian* documentation.

## Current Value Sampling Mode

*Current Value Sampling Mode* retrieves the data sample value with newest timestamp of any quality that was received by the archiver. This is not the same as retrieving the newest raw sample stored in the archive, since archive compression sometimes discards raw samples sent by the collector during the compression process.

Current Value Sampling retrieves a single sample containing the current value of the tag, not a series of historical samples. The sample has a timestamp, value, and quality.

**Timestamp**

Returns the time stamp on the sample sent to the archiver. The time stamp is not necessarily the current time. If collector compression is enabled and the deadband on the collector has not been exceeded for some time, the time stamp may be much earlier than the current time.

If data is sent to the archiver out of order, the current value is always the newest timestamp, even when the most recent value received is older than previous samples.

### Retrieving the current value of out of order data

1. Import this file that contains out of order data for a tag

```
* Example of Out Of Order data
* [Tags] Tagname,DataType,HiEngineeringUnits,LoEngineeringUnits
OUTOFORDERTAG,SingleFloat,60,0
[Data]
Tagname,TimeStamp,Value,DataQuality
OUTOFORDERTAG,29-Mar-2002 14:50:00.000,50.0,Good
OUTOFORDERTAG,29-Mar-2002 14:20:00.000,20.0,Good
```

```
OUTOFORDERTAG,29-Mar-2002 14:30:00.000,30.0,Good

OUTOFORDERTAG,29-Mar-2002 14:10:00.000,10.0,Good
```

2. Retrieve the data using current value sampling, using the following query:

```
select timestamp, tagname, value, quality from ihrawdata where samplingmode = CURRENTVALUE and

tagname = OUTOFORDERTAG
```

The time stamp of the current value should be the newest timestamp with the value and quality that was sent to the archiver, as shown here:

| Timestamp | Tagname | Value | Quality |
|---|---|---|---|
| 29-Mar-200214:50:00.000 | OUTOFORDERTAG | 50.00 | GoodNonSpecific |

- **Value**: Simply the value sent by the collector. The value is not interpolated to the current time or modified by the archiver during retrieval. The data type of the value will be the same data type as the tag's raw data.
- Data Quality: Returns the quality of the data sent by the collector. The current value can be of a bad data quality and will be flagged if the collector sends a sample with a bad data quality to the archiver. When the collector shuts down cleanly, it sends a bad data quality marker at shutdown time for all its tags. If the collector simply loses its connection to the archiver or crashes, the current value's quality will not automatically change to bad.

### Retrieving the current value of a tag

The following sequence of steps displays the behavior of **Current Value** sampling mode. After each step, retrieve the tag current value using this query:

```
select timestamp, tagname, value, quality from ihrawdata where samplingmode = CURRENTVALUE and

 tagname = IFIX.RAMP.F_CV
```

1. Configure the tag `IFIX.RAMP.F_CV` in an iFIX collector running on different PC than archiver. Configure it to have a one-second collection interval. The **Current Value** should be within one second of the value shown in a data link.
2. Stop the iFIX collector. The end-of-collection marker is sent to the data, so the **Current Value** quality should be marked as **bad** and its value set to zero.
3. Restart the iFIX collector. The **Current Value** quality should be marked as **good** and it should have a valid value.

4. Put the block off scan in the PDB. The **Current Value** quality should be marked as **bad**. (Put the block back on scan when you've verified this.)
5. Pull the network cable from the iFIX collector running on another machine. The current value remains unchanged as the value was good at the time the cable was pulled. To ensure that the **Current Value** is accurate, you would have to use the **Heartbeat Address** of the iFIX collector to verify that the collector is running.
6. Enable collector compression for the point and ensure that the tag's value does not change. The time stamp of the current value will stay the same until the collector reports a change.

### Anticipated Usage

The current value can be used in any operator display. You should also display the data quality of the current value. You may choose to use the **Heartbeat** address of the collector so that you can confirm that the collector is running and that the current value is therefore up to date.

If the collector was shut down gracefully, then the current value would correctly display a bad data quality (and a value of 0). If the collector crashed or was disconnected from the server, then the current value will be the last value sent before the crash or disconnect.

## Lab Sampling Mode

Lab Sampling is designed to duplicate the way iFIX classic Historian (HTA/HTC) returned data. This sampling mode returns only collected values. Each collected value is repeated until the next collected value, resulting in a jagged step plot instead of a smooth curve.

Interpolated values are used in other calculation modes. Lab sampling is never used by calculation modes. Each sample has the following attributes:

- **Timestamp** - Lab sampling determines intervals and timestamps the same as interpolated retrieval.
- **Value** - Any value returned is an actual collected raw value; the data value is never interpolated.
- **Data Quality** - Lab sampling uses the same logic as interpolated sampling to determine percent good quality.

### Retrieving lab sample values of an interval with GOOD data

This sample uses exactly the same parameters as the interpolated sampling example, except that the sampling mode should be specified as `lab`.

```
select timestamp, value, quality from

               ihrawdata where samplingmode=lab and timestamp >= '29-Mar-2002 13:50' and
```

```
timestamp <= '29-Mar-2002 14:30' and tagname  = tag1 and numberofsamples =
8
```

This supplies the following results:

| Timestamp | Value | Quality |
|---|---|---|
| 29-Mar-200213:55:00.000 | 0.00 | 0.00 |
| 29-Mar-200214:00:00.000 | 22.70 | 100.00 |
| 29-Mar-200214:05:00.000 | 22.70 | 100.00 |
| 29-Mar-200214:10:00.000 | 12.50 | 100.00 |
| 29-Mar-200214:15:00.000 | 7.00 | 100.00 |
| 29-Mar-200214:20:00.000 | 7.00 | 100.00 |
| 29-Mar-200214:25:00.000 | 4.80 | 100.00 |
| 29-Mar-200214:30:00.000 | 4.80 | 100.00 |

The value is never anything other than a collected value. This differs from interpolated sampling. A plot of this data would look like a series of steps, rather than a smooth, interpolated curve.

**Anticipated Usage:** Since lab sampling returns real, collected values, it is more accurate when a sufficient number of raw samples are stored. Use interpolated sampling for highly compressed data. It is generally not useful with archive compression. Collector compression can be used to filter out non-changing values, but a high deadband reduces the number of raw samples and therefore reduces the accuracy of lab sampling.

## Interpolated Sampling Mode

This topic describes *interpolated retrieval mode*. It also presents concepts that are common to interpolated, lab, calculated, and trend retrieval modes. Interpolation is a separate sampling mode and is also used in the various calculation modes.

Data compression necessitates interpolation. A minimal number of real data points is stored in the archive. On retrieval, interpolation is performed to produce an evenly spaced list of the most likely real world values. Even if you are not using compression, you can use interpolation if you want samples spaced on intervals other than the "true" collection rate.

The following data is used in the examples below. You can import this data into Historian if you want to try the examples yourself:

```
*Example for Interpolated Data Documentation

*

[Tags]

Tagname,DataType,HiEngineeringUnits,

LoEngineeringUnits TAG1,SingleFloat,60,0

BADDQTAG,SingleFloat,60,0

[Data]

Tagname,TimeStamp,Value,DataQuality

TAG1,29-Mar-2002 13:59:00.000,22.7,Good

TAG1,29-Mar-2002 14:08:00.000,12.5,Good

TAG1,29-Mar-2002 14:14:00.000,7.0,Good

TAG1,29-Mar-2002 14:22:00.000,4.8,Good

BADDQTAG,29-Mar-200213:59:00.000,22.7,Good

BADDQTAG,29-Mar-2002 14:08:00.000,12.5,Bad

BADDQTAG,29-Mar-2002 14:14:00.000,7.0,Bad

BADDQTAG,29-Mar-2002 14:22:00.000,4.8,Good
```

**Timestamp**

All sampling and calculation modes (except raw sampling) use the same method for creating intervals from the start and end time. Raw retrieval has no intervals, only a start and end time. Each mode differs in how it arrives at the value to assign to that interval

The simplest case is when the interval is evenly divisible by the number of samples or by the interval in milliseconds. For example, the start and end times are one hour apart and you want data at ten-minute intervals, or 6 samples. The first time stamp occurs at the start time + one interval and represents the samples from a point greater than the start time to less than or equal to the interval time stamp.

## Determining interval timestamps for evenly divisible duration

1. Import this data into the Historian. There is only a tag, with no data.

```
[Tags]
Tagname,DataType,HiEngineeringUnits,LoEngineeringUnits
c1,SingleFloat,100,0
```

2. Retrieve data for that tag over a 1-hour duration with a 10-minute interval. Use the following query:

```
select timestamp from ihrawdata where timestamp >= 14:00 and timestamp <= 15:00 and tagname = c1 and
numberofsamples = 6
```

   or this query

```
select timestamp from ihrawdata where timestamp >= 14:00 and timestamp <= 15:00 and tagname = c1 and
Intervalmilliseconds = 10M
```

Both SQL queries result in the same intervals and interval timestamps

```
3/29/2002 14:10:00
3/29/2002 14:20:003/29/2002 14:30:00
3/29/2002 14:40:00
3/29/2002 14:50:00
3/29/2002 15:00:00
```

When the 1-hour duration is not evenly divisible, interval timestamps will include milliseconds even if the data samples do not use a resolution of milliseconds.

## Example: Determining interval timestamps for a non-divisible duration

Divide the one hour duration from previous example into 7 intervals:

```
select timestamp from ihrawdata where timestamp >= 14:00 and timestamp <= 15:00 and tagname = c1 and
numberofsamples = 7
```

```
3/29/2002 14:08:34.285
3/29/2002 14:17:08.571
3/29/2002 14:25:42.857
3/29/2002 14:34:17.142
3/29/2002 14:42:51.428
3/29/2002 14:51:25.714
3/29/2002 14:59:59.999
```

**Note:**
Trend sampling determines intervals using a different method, described in the trend sampling topic.

## Value

The logic for determining the value through interpolation is as follows:

**Attribute samples to intervals**

Any raw sample is attributed to exactly one interval based on the raw sample and interval time stamp. The rule is that the sample has to have a time stamp greater than the interval start time, but less than or equal to the end time. This is because the end timestamp of the interval is the start timestamp on the next interval.

**Interpolate a value at each interval end time**

For each interval end time, find the raw point before and after the end time. The interval time stamp is the interval end time; we can then interpolate the value at that time.

## Determining interval interpolated value

This example shows how linear interpolation determines the most likely real world value at the interval timestamp.

Using the same data set as above, there are raw points at:

```
14:08:00.000,12.5,Good

14:14:00.000,7.0,Good
```

and you are trying to get an interpolated value at 14:10. The calculation used for linear interpolation would be:

```
interpolated value = previous raw sample + ((deltaY/deltaX) * offset)
```

Substituting the numbers for this example:

`deltaY` = 7.0 12.5 = -5.5

`deltaX` = 14-8 = 6

`offset` = 2 seconds (from 14:08 to 14:10)

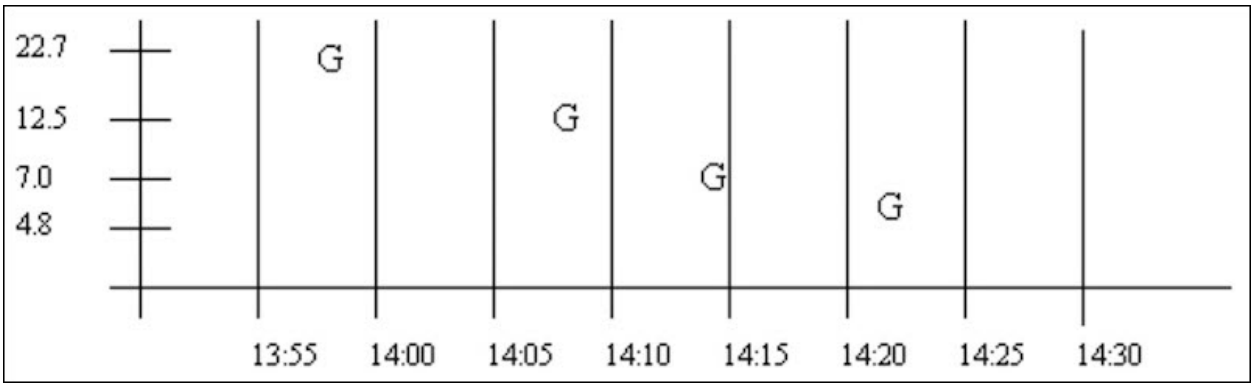
`Interpolated value` = 12.5 + ((-5.5/6)*2) = 10.67

**About Interpolated Data Type**

When interpolating data, the data type of the value will be the same data type as that of the tag's raw data. Only floating point and double floating point values can be interpolated. Integers, strings, and blobs cannot be interpolated. When attempting to interpolate string and integer data, interpolation will simply repeat the collected value for each interval until the next collected value.

### Retrieving interpolated values of an interval with GOOD data

The raw samples for TAG1 can be plotted as follows. The “G” indicates a good data quality raw sample.




Use this SQL query to retrieve the data:

```
select timestamp, value, quality from ihrawdata where samplingmode=interpolated and timestamp >=
 '29-Mar-2002 13:50' and timestamp &lt;= '29-Mar-2002 14:30' and tagname  = tag1 and numberofsamples = 8
```

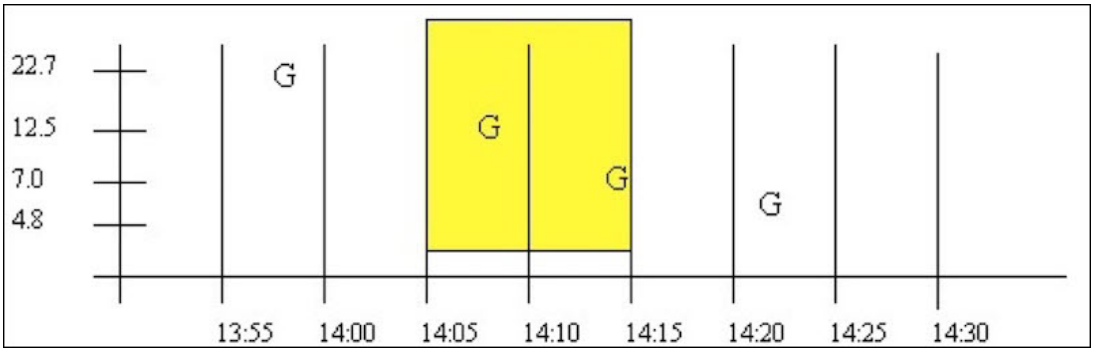
| Timestamp | Value | Quality |
|---|---|---|
| 29-Mar-2002 13:55:00.000 | 0.00 | 0.00 |
| 29-Mar-2002 14:00:00.000 | 21.57 | 100.00 |
| 29-Mar-2002 14:05:00.000 | 15.90 | 100.00 |
| 29-Mar-2002 14:10:00.000 | 10.67 | 100.00 |
| 29-Mar-2002 14:15:00.000 | 6.73 | 100.00 |
| 29-Mar-2002 14:20:00.000 | 5.35 | 100.00 |
| 29-Mar-2002 14:25:00.000 | 4.80 | 100.00 |
| 29-Mar-2002 14:30:00.000 | 4.80 | 100.00 |

**Note:**
The 13:50 to 13:55 interval is represented by the 13:55 timestamp.

There may be many raw points in an interval, but interpolation uses only the last one in the interval and the first one in the next interval. The sections below describe the interpolation behavior in the 3 possible cases.

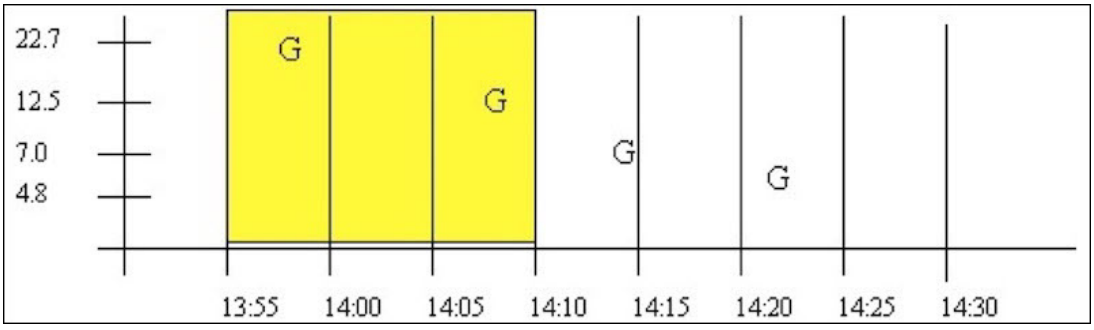
**Case 1: Good Data Samples Before and After the Interval Timestamp**

This is the typical case when compression is not used. There are 2 good data quality raw points. With interpolation, calculate the slope and offset of this line and interpolate the value at the interval timestamp. The 14:10 interval has a sample at 14:08 and at 14:14.



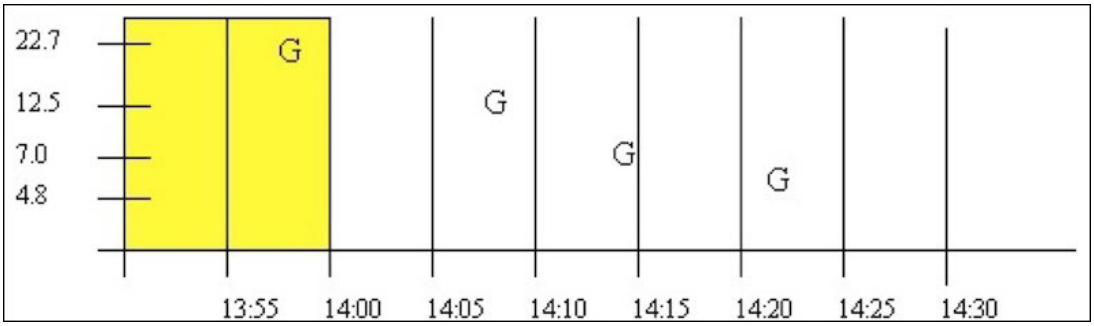

**Case 1a: Good Data Samples between the Interval Timestamp and the Start and End Time**

In a similar case, there may be intervals with no raw samples, such as when data compression is used. Here, there is at least 1 good raw sample between the start time and interval, and at least 1 good raw sample between the interval and end time. The good raw samples are interpolated across intervals to determine values at the 14:00 and 14:05 intervals:



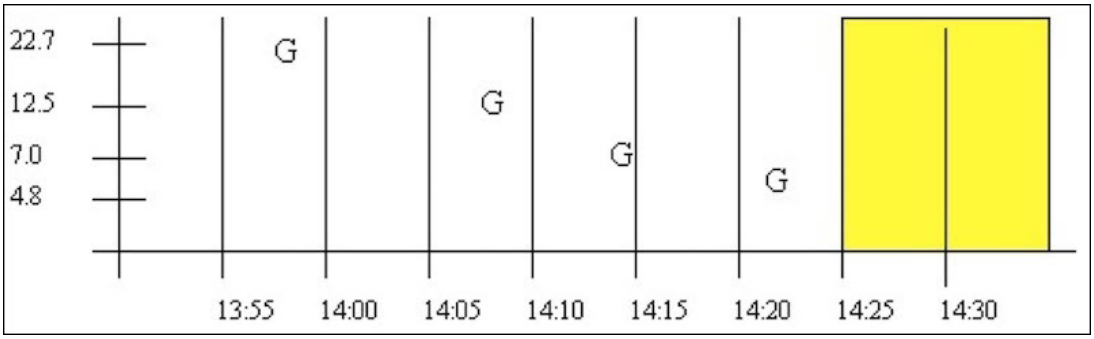

**Case 2: No Good Data between Start Time and Interval Timestamp**

If no or bad data occurs before the interval, then the interval is given a bad data quality. The 13:55 interval is an example of this. Note that bad data is treated identically to no data.




**Case 3: No Good Data between Interval Timestamp and End Time**

If no or bad data occurs after the interval then the interval is given a good data quality, but the value is simply stretched instead of interpolated. The 14:25 interval is an example of this. Note that bad data is treated identically to no data. Good data quality is attributed to the 14:30 interval




## Data Quality

Unlike CurrentValue, RawByTime, and RawByNumber, Interpolated data does not assign an individual data quality to each returned sample. Since Interpolated, Lab, and Calculated retrieval modes can contain multiple samples in an interval, the data qualities of each point are combined and summarized as a percent good value.

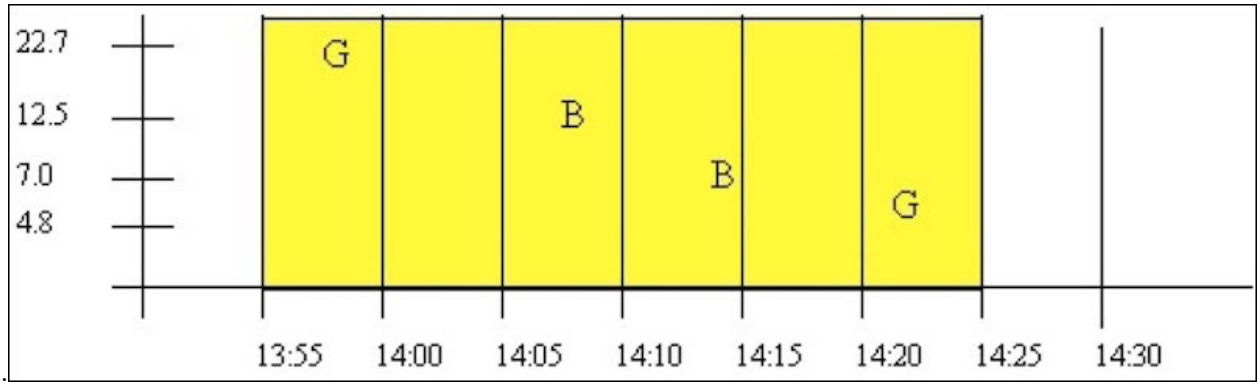
Interpolated and Lab sampling determine the percent good using the same procedure, resulting in a value of either 100 or 0 (though the determined value may be different for each mode even with the same data). Intermediate percent good values are determined only for Calculated retrieval modes.

The following examples illustrate interpolated and lab sampling modes. For each example, you can see that the behavior is the same for lab and interpolated sampling by changing `samplingmode=Interpolated` to `samplingmode=lab`.

### Interpolated and Lab retrieval resulting in percent good of 100

This example illustrates the effect of bad data quality samples on the percent good statistic for an interval. The start and end times vary so that bad samples are included or excluded, which affects the percent good statistic

The data for `BADDQTAG` can be plotted as follows. The `G` is used to indicate a good data quality raw sample and the `B` indicates a sample of bad data quality. A query of the whole data set is shown.




Using this query for a period starting with good data quality:

```
select timestamp, value, quality from ihrawdata where samplingmode=interpolated and timestamp >=

'29-Mar-2002 13:55' and timestamp <= '29-Mar-2002 14:25' and tagname  = baddqtag and numberofsamples = 1
```

This results in the following data quality:

| Timestamp | Value | Quality |
|---|---|---|
| 29-Mar-200214:25:00.000 | 4.80 | 100.00 |

The percent good is 100. Even though the interval contains bad data quality samples, the interval does not end with bad data quality. Percent good is determined this way because the purpose of interpolation and lab sampling is to determine the value and quality at the interval timestamp. On the other hand, Calculation modes operate on the full set of raw samples within an interval and therefore result in percent good values between 0 and 100.

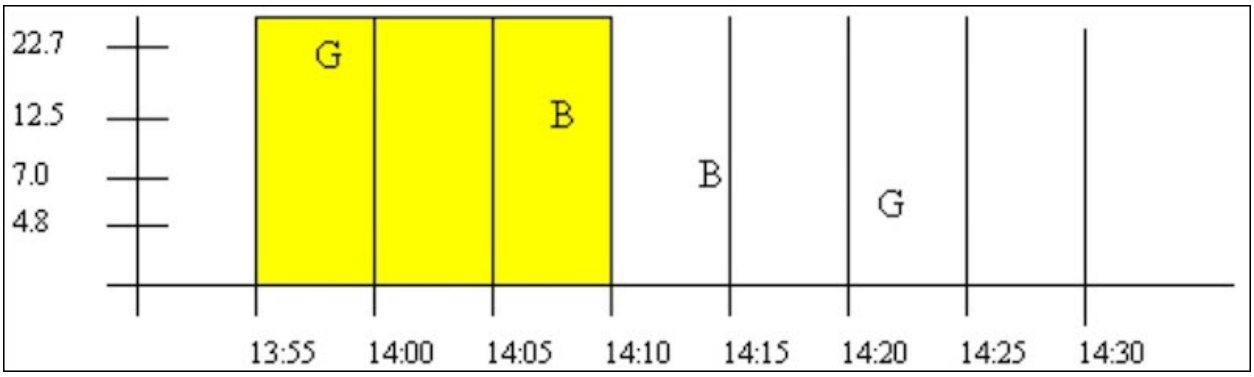
This interval from 14:10 to 14:25 starts with a bad data quality sample but ends with a good sample, so the results are the same. That is, the query:

```
select timestamp, value, quality from ihrawdata where samplingmode=interpolated and timestamp >=

 '29-Mar-2002 14:10' and timestamp <= '29-Mar-2002 14:25' and tagname  = baddqtag and numberofsamples = 1
```

produces the same percent good result of 100.

### Example: Interpolated and Lab retrieval resulting in percent good of 0

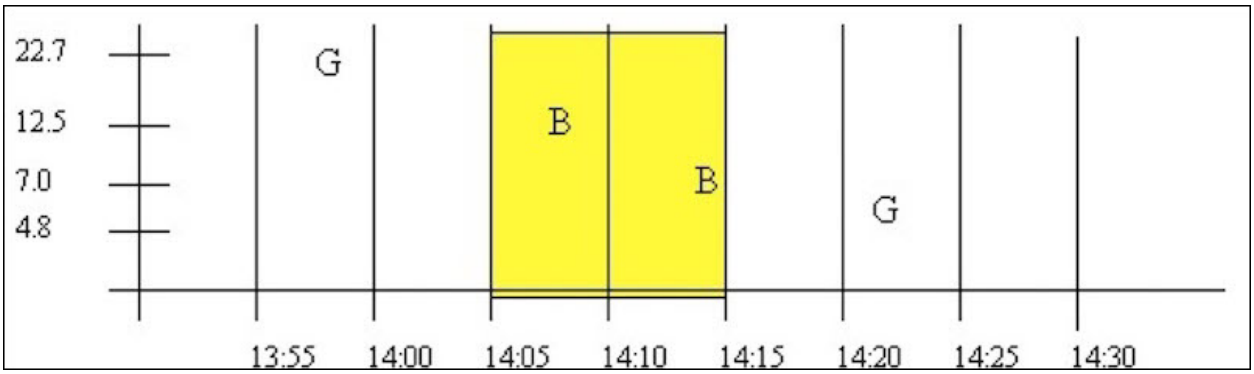
This example shows some data patterns that result in a percent good of 0. An interval ending with a bad data quality sample, always results in a percent good of 0 for the interval.




| Timestamp | Value | Quality |
|---|---|---|
| 29-Mar-2002 14:10:00.000 | 0.00 | 0.00 |

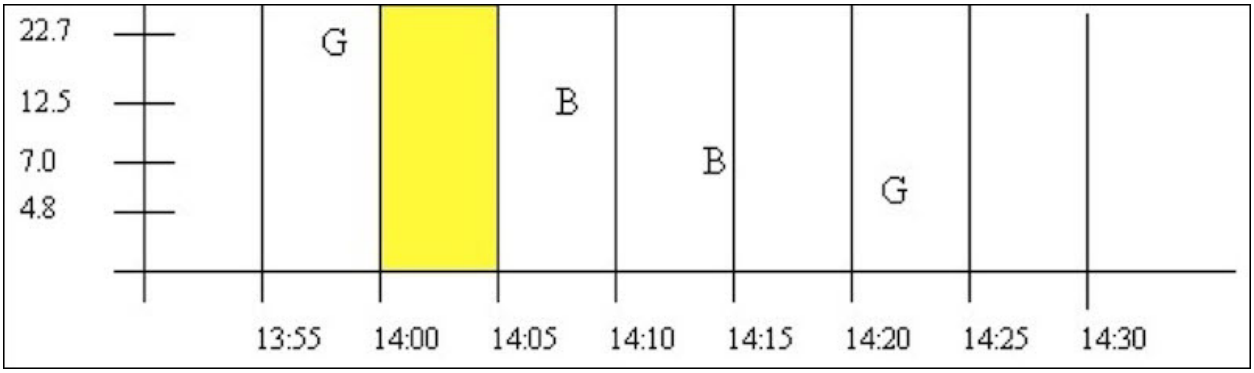
```
select timestamp, value, quality from ihrawdata where samplingmode=interpolated and timestamp >=

'29-Mar-2002 13:55' and timestamp <= '29-Mar-2002 14:10' and tagname  = baddqtag and numberofsamples = 1
```

| Timestamp | Value | Quality |
|---|---|---|
| 29-Mar-2002 14:10:00.000 | 0.00 | 0.00 |



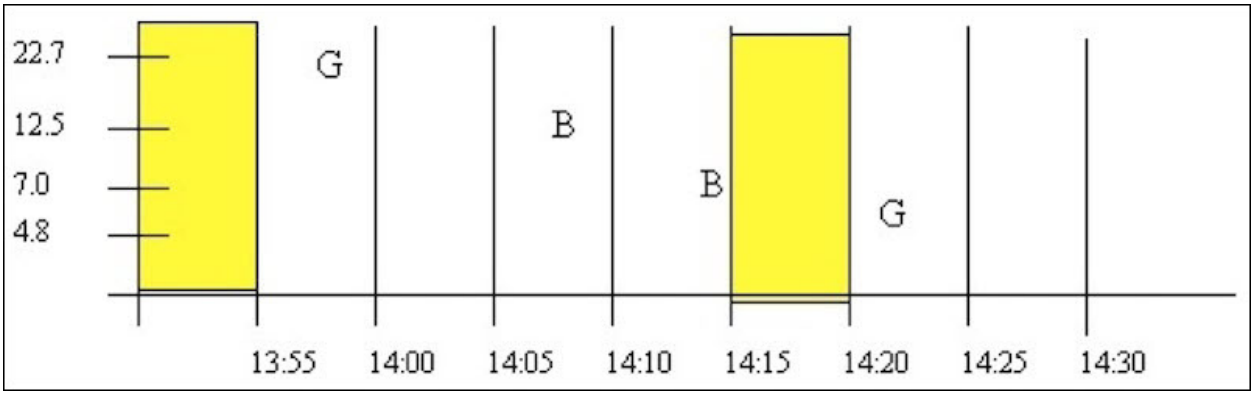

### Example: Interpolated and Lab retrieval of an empty interval

The data quality of an empty interval depends on the previous and following raw samples. Intervals with a prior good data quality sample have a percent good of 100 and intervals preceded by a bad data quality sample (or no sample) have in a percent good of zero.




This query results in a percent good of 100:

```
select timestamp, value, quality from ihrawdata where samplingmode=interpolated and timestamp >=
 '29-Mar-2002 14:00' and timestamp <= '29-Mar-2002 14:05' and tagname  = baddqtag and numberofsamples = 1
```




Both of these queries produce a percent good of 0. The first has no preceding sample and the second is preceded by bad data:

```
select timestamp, value, quality from ihrawdata where samplingmode=interpolated and timestamp >=
'29-Mar-2002 13:50' and timestamp <= '29-Mar-2002 13:55' and tagname  = baddqtag and numberofsamples = 1
select timestamp, value, quality from ihrawdata where samplingmode=interpolated and timestamp >=
'29-Mar-2002 14:15' and timestamp <= '29-Mar-2002 14:20' and tagname  = baddqtag and numberofsamples = 1
```

The lab retrieval at 14:15 has a value of 7 but quality of 0. Note that you should almost always ignore specific values when the percent good is 0.

## Raw Data Sampling Modes

To use raw data retrieval, you need only specify a start and end time, or a start time and number of samples. Any specified interval duration is ignored. Raw data may be retrieved using one of two methods:

- **RawByTime retrieval**: Specify a start and end time for data retrieval. RawByTime returns all raw samples of all qualities with a time stamp greater than the start time and less than or equal to the end time. It will not return a raw sample with same time stamp as the start time. NumberOfSamples is ignored and all raw samples will be returned.
- **RawByNumber Retrieval**: Specify a start time, a number of samples, and a direction (forward or backward). RawByNumber retrieval returns X raw samples of all qualities starting from a time stamp of the indicated start time, moving in the specified direction. It will return a raw sample with the same time stamp as the start time. If there is no sample at the specified start time, the retrieval count begins at the next sample.

Each sample has the following attributes:

- **Timestamp:** The time stamp sent by the collector along with the raw sample.
- **Value:** The value sent by the collector along with the raw sample.
- **Data Quality:** The quality of data sent by the collector, as set by the collector.

Archive compression can reduce the number of raw samples stored in the archive. Archive compression may discard raw samples sent by the collector; these are not stored as raw samples and would not be returned by raw data retrieval.

If the current value has not been stored as a raw sample, will not be returned by a raw data retrieval.

If they exist within the requested time period, collected samples with a bad data quality and collector startup and shutdown markers will be returned in a raw data query.

## RawByTime retrieval of samples over a period of replaced data

1. Import this data into the Historian.

```
[Tags]

Tagname,DataType,HiEngineeringUnits,LoEngineeringUnits

RAWTAG,SingleInteger,100,0

[Data]

Tagname,TimeStamp,Value,DataQuality

RAWTAG,29-Mar-2002 13:59:00.000,7,Good

RAWTAG,29-Mar-2002 14:08:00.000,8,Bad
```

2. Import this data into the Historian so that there is replaced data:

```
[Data]

Tagname,TimeStamp,Value,DataQuality
```

```
RAWTAG,29-Mar-2002 13:59:00.000,22,Good
RAWTAG,29-Mar-2002 14:08:00.000,12,Bad
RAWTAG,29-Mar-2002 14:22:00.000,4,Good
```

3. Retrieve the data using this RawByTime query.

```
select timestamp, value, quality from ihrawdata where samplingmode=rawbytime and timestamp>='29-Mar-2002
13:59' and timestamp<='29-Mar-2002 14:22' and tagname=rawtag
```

The following results are obtained:

| Timestamp | Value | Quality |
|---|---|---|
| 29-Mar-200214:08:00.000 | 12 | Bad NonSpecific |
| 29-Mar-200214:22:00.000 | 4 | Good NonSpecific |

Note that the raw sample exactly at the start time is not returned and that the replaced value of 8 at 14:08 is not returned. If the start time is changed to 13:58:59, then all the samples are returned:

```
select timestamp, value, quality from ihrawdata where samplingmode=rawbytime and timestamp>='29-Mar-2002
13:58:59' and timestamp<='29-Mar-2002 14:22' and tagname=RAWTAG
```

| Timestamp | Value | Quality |
|---|---|---|
| 29-Mar-200213:59:00.000 | 22 | Good NonSpecific |
| 29-Mar-200214:08:00.000 | 12 | Bad NonSpecific |
| 29-Mar-200214:22:00.000 | 4 | Good NonSpecific |

## RawByNumber retrieval over a period of replaced data

The RawByNumber sampling mode returns up to a specified number of raw samples beginning at the start time. The end time is ignored. Unlike the RawByTime, this can return a sample that has the same time stamp as the start time. You must specify a direction forward or backward from the start time to retrieve data.

1. Using the data imported by the previous example, retrieve 10 samples going forward from 13:59:00.

```
select timestamp, value, quality from ihrawdata where samplingmode=rawbynumber and timestamp>='29-Mar-2002
13:59' and numberofsamples=10 and direction=forward and tagname=RAWTAG
```

The following results are obtained.

| Timestamp | Value | Quality |
|---|---|---|
| 29-Mar-200213:59:00.000 | 22 | Good NonSpecific |
| 29-Mar-200214:08:00.000 | 12 | Bad NonSpecific |
| 29-Mar-200214:22:00.000 | 4 | Good NonSpecific |

2. Using the data imported by the previous example, retrieve 10 samples going backward from 14:22:00.

```
select timestamp, value, quality from ihrawdata where samplingmode=rawbynumber and timestamp<='29-Mar-2002
14:22' and numberofsamples=10 and direction=backward and tagname=RAWTAG
```

   The following results are obtained.

| Timestamp | Value | Quality |
|---|---|---|
| 29-Mar-200214:22:00.000 | 4 | Good NonSpecific |
| 29-Mar-200214:08:00.000 | 12 | Bad NonSpecific |
| 29-Mar-200213:59:00.000 | 22 | Good NonSpecific |

**Anticipated usage**

You can use raw sampling to compute a raw minimum or raw maximum over a time period. `Raw Average` is already provided as a native calculation mode

You can also use raw sampling to analyze system efficiency. Count the number of raw samples per period of time, ignoring the values, then compare it to other periods of time.

If you have a high number of raw samples you may decide to implement collector or archive compression. If you have a different count of raw samples than another time period for the same point in your process, you should understand why the data is missing or why the extra data was logged.

You can use the `ihCount` calculation mode to easily count the number of raw samples between the start and end time.

## RawByFilterToggle Sampling Mode

The RawByFilterToggle sampling mode is a form of filtered data query. A filtered data query returns data values for a particular time period whereas RawByFilterToggle sampling mode returns the time periods where the condition becomes TRUE or FALSE. The RawByFilterToggle sampling mode returns the

Timestamp, Value, and Data Quality for the matching entries. The data values returned will have the same tagname which you queried for.

RawByFilterToggle returns only 0 and 1. The value 1 is returned with a timestamp when the filter condition becomes TRUE, and the value 0 is returned with the timestamp when the filter condition becomes FALSE. You can have multiple pairs of 1 and 0 values if the condition becomes TRUE multiple times between the start and end time. If the condition never became TRUE between the start and end time, you will not get any values.

**Timestamp**

The RawByFilterToggle sampling mode returns 0 and 1 as values. The value 1 is returned with a timestamp when the filter condition becomes TRUE, and the value 0 is returned with the timestamp when the filter condition becomes FALSE. You can have multiple pairs of 1 and 0 values if the condition becomes TRUE multiple times between the start and end time. If the condition never became TRUE between the start and end time, you will not get any values. You can use a filterexpression to return the time ranges that match the criteria.

The RawByFilterToggle sampling mode can return any timestamp between the start and end time, depending on if and when the condition becomes TRUE or FALSE. The timestamps returned can be queried further using RawByTime, RawByNumber, Interpolated, or any other sampling or calculation mode.

**Value**

This sampling mode only returns 0 and 1 as values. The value 1 is returned with a timestamp where the filter condition is TRUE and 0 is returned with the timestamp where the filter condition is FALSE.

**Data Quality**

The RawByFilterToggle considers only Good quality data.

## Retrieving Data Using RawByFilterToggle Sampling Mode

The following two examples use this data that is imported into Proficy Historian. This data will be used in the examples for retrieving data with the RawByFilterToggle sampling mode.

```
[Tags]

Tagname,DataType,HiEngineeringUnits,

LoEngineeringUnits RAMP,SingleInteger,10,0

[Data]

Tagname,TimeStamp,Value,Data Quality
```

```
RAMP,25-Feb-2013 07:00:00.000,0,Good,
RAMP,25-Feb-2013 07:00:01.000,1,Good,
RAMP,25-Feb-2013 07:00:02.000,2,Good,
RAMP,25-Feb-2013 07:00:03.000,3,Good,
RAMP,25-Feb-2013 07:00:04.000,4,Good,
RAMP,25-Feb-2013 07:00:05.000,5,Good,
RAMP,25-Feb-2013 07:00:06.000,6,Good,
RAMP,25-Feb-2013 07:00:07.000,7,Good,
RAMP,25-Feb-2013 07:00:08.000,8,Good,
RAMP,25-Feb-2013 07:00:09.000,9,Good,
RAMP,25-Feb-2013 07:00:10.000,10,Good,
RAMP,25-Feb-2013 07:00:11.000,11,Good,
RAMP,25-Feb-2013 07:00:12.000,12,Good,
RAMP,25-Feb-2013 07:00:13.000,13,Good,
RAMP,25-Feb-2013 07:00:14.000,14,Good,
RAMP,25-Feb-2013 07:00:15.000,15,Good,
RAMP,25-Feb-2013 07:00:16.000,16,Good,
RAMP,25-Feb-2013 07:00:17.000,17,Good,
RAMP,25-Feb-2013 07:00:18.000,18,Good,
RAMP,25-Feb-2013 07:00:19.000,19,Good,
RAMP,25-Feb-2013 07:00:20.000,20,Good,
RAMP,25-Feb-2013 07:00:21.000,21,Good,
RAMP,25-Feb-2013 07:00:22.000,22,Good,
RAMP,25-Feb-2013 07:00:23.000,23,Good,
RAMP,25-Feb-2013 07:00:24.000,24,Good,
RAMP,25-Feb-2013 07:00:25.000,25,Good,
RAMP,25-Feb-2013 07:00:26.000,26,Good,
RAMP,25-Feb-2013 07:00:27.000,27,Good,
RAMP,25-Feb-2013 07:00:28.000,28,Good,
RAMP,25-Feb-2013 07:00:29.000,29,Good,
RAMP,25-Feb-2013 07:00:30.000,30,Good,
RAMP,25-Feb-2013 07:00:31.000,31,Good,
RAMP,25-Feb-2013 07:00:32.000,32,Good,
RAMP,25-Feb-2013 07:00:33.000,33,Good,
RAMP,25-Feb-2013 07:00:34.000,34,Good,
RAMP,25-Feb-2013 07:00:35.000,35,Good,
RAMP,25-Feb-2013 07:00:36.000,36,Good,
```

```
RAMP,25-Feb-2013 07:00:37.000,37,Good,
RAMP,25-Feb-2013 07:00:38.000,38,Good,
RAMP,25-Feb-2013 07:00:39.000,39,Good,
RAMP,25-Feb-2013 07:00:40.000,40,Good,
RAMP,25-Feb-2013 07:00:41.000,41,Good,
RAMP,25-Feb-2013 07:00:42.000,42,Good,
RAMP,25-Feb-2013 07:00:43.000,43,Good,
RAMP,25-Feb-2013 07:00:44.000,44,Good,
RAMP,25-Feb-2013 07:00:45.000,45,Good,
RAMP,25-Feb-2013 07:00:46.000,46,Good,
RAMP,25-Feb-2013 07:00:47.000,47,Good,
RAMP,25-Feb-2013 07:00:48.000,48,Good,
RAMP,25-Feb-2013 07:00:49.000,49,Good,
RAMP,25-Feb-2013 07:00:50.000,50,Good,
RAMP,25-Feb-2013 07:00:51.000,51,Good,
RAMP,25-Feb-2013 07:00:52.000,52,Good,
RAMP,25-Feb-2013 07:00:53.000,53,Good,
RAMP,25-Feb-2013 07:00:54.000,54,Good,
RAMP,25-Feb-2013 07:00:55.000,55,Good,
RAMP,25-Feb-2013 07:00:56.000,56,Good,
RAMP,25-Feb-2013 07:00:57.000,57,Good,
RAMP,25-Feb-2013 07:00:58.000,58,Good,
RAMP,25-Feb-2013 07:00:59.000,59,Good,
```

### Determining the Time Range After the Condition Became TRUE

An example of a Query using RawByFilterToggle sampling mode is as follows:

```
starttime='02/25/2013 07:00:00', endtime='02/25/2013 07:10:00'
select timestamp, value, quality from ihrawdata where tagname = RAMP and samplingmode= rawbyfiltertoggle
and filterexpression='(RAMP>50)' and filtermode=AfterTime
```

This query `set` would determine when the ramp value exceeded 50 and returns the time range after that. The following results are obtained:

| Timestamp | Value | Quality |
|---|---|---|
| 02/25/201307:00:00 | 0 | Good NonSpecific |
| 02/25/201307:00:51 | 1 | Good NonSpecific |

| Timestamp | Value | Quality |
|---|---|---|
| 02/25/201307:10:00 | 1 | Good NonSpecific |

You can see in the raw data that the condition became true at 7:00:51 so the sample is returned with a value of 1. The 0 and 1 are bounding values that would make the data easier to plot. You cannot simply count the number of 1s returned to count the number of times the condition became true. You have to exclude the bounding values

### Example 2: Determining the Time Range Before the Condition Became TRUE

An example of a query using RawByFilterToggle sampling mode is as follows

```
set starttime='02/25/2013 07:00:00', endtime='02/25/2013 08:00:00'

select timestamp, value, quality from ihrawdata where tagname = RAMP and samplingmode= rawbyfiltertoggle

and filterexpression='(RAMP>10)' and filtermode=BeforeTime
```

The following results are obtained

| Timestamp | Value | Quality |
|---|---|---|
| 02/25/201307:00:00 | 0 | Good NonSpecific |
| 02/25/201307:00:10 | 1 | Good NonSpecific |
| 02/25/201307:00:59 | 0 | Good NonSpecific |
| 02/25/201308:00:00 | 0 | Good NonSpecific |

You can see in the raw data that the condition became true at 7:00:10 so the sample is returned with a value of 1.

**Anticipated Usage**

This sampling mode can be used for the same reasons as filtered data queries. That is, when you want the Historian Data Archiver to determine the exact time(s) of the event and you have an approximate time range for an event of interest, such as:

- A batch starting or completing.
- A value exceeding a limit.
- A collected value matching a specified value.

Once you have the exact time range(s) as returned from `RawByFilterToggle`, you can use those time ranges in the subsequent data queries or in custom reporting or data analysis applications.

## Trend Sampling Mode

The Trend Sampling mode maximizes performance when retrieving data specifically for plotting.

The Trend Sampling mode identifies significant points and returns them to the caller. These will be raw samples. Significant points are established by finding the raw minimum and raw maximum values within each interval. Note that this is not the same as finding the change in slope direction of a line, as archive compression does.

The Trend Sampling mode approximates a high resolution trend with only as much detail as could be drawn on the page. For example, say you are about to draw a trend on the page and you know that the area with the trend graph is only 100 pixels wide. You could not possibly represent any more than 100 points in those 100 pixels. By using the Trend sampling type, you can ensure you retrieve adjacent highs and lows to draw a visually accurate trend with only 100 points, regardless of whether the time period was one year or one hour.

Since the Trend Sampling mode does not need to acquire all data between the specified start and end times, it is a very efficient method of data retrieval, especially for large data sets. Depending on the requested start and end times (and the amount of data stored for that interval), it could be as much as 100 times faster than other methods.

When displaying data for reports or examination, this principle can be applied to other sampling types too. It is highly inefficient to trend data at a higher resolution than can be drawn on page or printed on hard copy. This is useful even for calculation modes like "Average value". It is not suitable for Interpolated mode, since this results in a loss of detail that ihTrend sampling attempts to recapture.

The `ihTrend` sampling type returns adjacent highs and lows within each interval. If you ask for 100 samples, you will effectively receive 50 high values and 50 low values over 100 intervals. The retrieval process works as follows:

1. Divide the query duration into even-length intervals, like other sampling modes.
2. Determine the raw minimum and raw maximum for each interval. If there is only one point, then that is both the minimum and maximum.
3. Since we want to return 2 samples per interval (a minimum and a maximum), we need twice as many intervals. Divide each interval in half. For example, a one hour interval of 01:00:00 to 02:00:00 becomes 2 intervals (01:00:00 to 01:30:00) and (01:30:00 to 02:00:00) .
4. Put the minimum in one half-interval and the maximum in the other. If minimum comes before maximum, put the minimum in the first half-interval and the maximum in second half-interval, and vice versa.

When doing filtered data queries, your maximum returned intervals must pass the throttle, even if only a few intervals actually match the filtered criteria.

**Timestamp**

There is no difference between full-interval timestamps and half-interval timestamps. Both are valid and all interval timestamps are in ascending order.

The Trend Sampling mode will always have an even number of samples, rounded up when necessary.

For example, if you request num samples = 7 or num samples = 8, you will get 8 samples.

If you request results by interval instead of number of samples, you will get back twice the number of results you expect.

For example, a 5-minute interval for a 40-minute duration is normally 40 / 5 = 8 samples. But with trend sampling, you get 16 evenly-spaced intervals.

**Value**

The raw minimum or raw maximum of the full interval. There is no indication as to which one you are getting.

**Data Quality**

Trend sampling uses the same logic as interpolated sampling to determine the percent good quality.

## Retrieving trend sample value

Using the data from the interpolated example, execute this query

```
select timestamp, value, quality from ihrawdata where samplingmode=trend and timestamp >=
'29-Mar-2002 13:50' and timestamp <= '29-Mar-2002 14:30' and tagname  = tag1 and numberofsamples = 8
```

The following results are returned:

| Timestamp | Value | Quality | Raw Samples |
|---|---|---|---|
| 29-Mar-200213:55:00.000 | 22.70 | 100.00 | None |
| 29-Mar-200214:00:00.000 | 22.70 | 100.00 | 13:59:00.000,22.7, Good |

| Timestamp | Value | Quality | Raw Samples |
|---|---|---|---|
| 29-Mar-200214:05:00.000 | 12.50 | 100.00 | None |
| 29-Mar-200214:10:00.000 | 12.50 | 100.00 | 14:08:00.000,12.5, Good |
| 29-Mar-200214:15:00.000 | 7.00 | 100.00 | 14:14:00.000,7.0, Good |
| 29-Mar-200214:20:00.000 | 7.00 | 100.00 | |

The interval timestamps are the same as for interpolated. The raw minimum and raw maximum are determined for each interval.

For example, a tag has data every second for 1 year (around 31 million data points). We want to perform a query using `ihTrend` with `StartTime` = `LastYear`, `EndTime` = `now`, and `NumSamples` = 364.

The `StartTime` to `EndTime` is broken down into `NumSamples`/2 pseudo-intervals (182). For each pseudointerval, the min and max value is found. These will be the first two data points. With two data points per pseudo-interval multiplied by `NumSamples`/2 gives us the desired `NumSamples`. If the minimum occurs before the maximum, it will be the first of the two samples, and vice versa.

The query:

```
select timestamp, value, quality from ihrawdata where samplingmode=lab and timestamp >=
 '29-Mar-2002 13:50' and timestamp <= '29-Mar-2002 14:30' and tagname  = tag1 and numberofsamples = 8
```

The following results are returned:

| Timestamp | Value | Quality |
|---|---|---|
| 29-Mar-200213:55:00.000 | 0.00 | 0.00 |
| 29-Mar-200214:00:00.000 | 22.70 | 100.00 |
| 29-Mar-200214:05:00.000 | 22.70 | 100.00 |
| 29-Mar-200214:10:00.000 | 12.50 | 100.00 |
| 29-Mar-200214:15:00.000 | 7.00 | 100.00 |
| 29-Mar-200214:20:00.000 | 7.00 | 100.00 |

| Timestamp | Value | Quality |
|---|---|---|
| 29-Mar-200214:25:00.000 | 4.80 | 100.00 |
| 29-Mar-200214:30:00.000 | 4.80 | 100.00 |

### Trend Data returned in the wrong interval

Note that, with trend sampling, data can be returned using an interval timestamp that does not contain the sample. A CSV file includes three values for each of 9 days.

```
[Data]
Tagname,TimeStamp,Value
Dfloattag5,01/05/03 8:00,95.00
Dfloattag5,01/05/03 15:00,88.00
Dfloattag5,01/05/03 16:00,80.00
Dfloattag5,01/06/03 7:00,11.00
Dfloattag5,01/06/03 10:00,13.00
Dfloattag5,01/06/03 13:00,93.00
Dfloattag5,01/07/03 8:00,99.0
Dfloattag5,01/07/03 11:00,86.0
Dfloattag5,01/07/03 12:00,16.0
Dfloattag5,01/08/03 8:00,0.00
Dfloattag5,01/08/03 12:00,99.00
Dfloattag5,01/08/03 14:00,100.00
```

If you use the following query:

```
Select timestamp,tagname,value Quality from ihrawdata where tagname =dfloattag5
And samplingmode= trend and intervalmilliseconds =24h
And timestamp> ‘1/02/2003 07:00:00’ and timestamp<= ‘01/10/2003 12:00:00’
```

then the results include:

| Timestamp | Tag Name | Value | Quality |
|---|---|---|---|
| 6-Jan-200319:00:00 | Dfloattag5 | 13.00 | 100 |
| 7-Jan-200307:00:00 | Dfloattag5 | 93.00 | 100 |
| 7-Jan-200319:00:00 | Dfloattag5 | 99.00 | 100 |
| 8-Jan-200307:00:00 | Dfloattag5 | 16.00 | 100 |

It is expected that the value 93 is listed for 1/6/03 19:00:00, since that is where the timestamp of the raw sample occurs. However, the maximum of 1/6/03 07:00:00 to 1/7/03 07:00:00 is:

```
Dfloattag5,01/06/03 13:00,93.00
```

which comes after the minimum of:

```
Dfloattag5,01/06/03 10:00,13.00
```

Hence, it is placed in the second half-interval, even though its timestamp does not fall into the time range for that half-interval. Raw samples will never be placed in the wrong "real" interval, but may be placed in the wrong "fake" interval.

**Anticipated Usage:** Trend sampling is designed only for graphical plotting applications.

## Trend2 Sampling Mode

The Trend2 sampling mode is a modified version of the Trend sampling mode.

The Trend2 sampling mode splits up a given time period into a number of intervals (using either a specified number of samples or specified interval length), and returns the minimum and maximum data values that occur within the range of each interval, together with the timestamps of the raw values.

The key differences between Trend and Trend2 sampling modes are in:

- How they treat a sampling period that does not evenly divide by the interval length:
  - For the Trend sampling mode, Historian ignores any leftover values at the end, rather than putting them into a smaller interval.
  - For the Trend2 sampling mode, Historian creates as many intervals of the interval length as will fit into the sampling period, and then creates a remainder interval from whatever time is left.
- Spacing of timestamps returned:
  - For the Trend sampling mode, Historian returns evenly-spaced interval timestamps.
  - For the Trend2 sampling mode, Historian returns raw sample timestamps. These timestamps can be unevenly spaced, since raw data can be unevenly spaced.
- Inclusion of start and end times entered:
  - The Trend sampling mode is start time exclusive and end time inclusive.
  - The Trend2 sampling mode is start time inclusive and end time inclusive.

The Trend sampling mode is more suitable for plotting applications that prefer evenly-spaced data.

The Trend2 sampling mode is more suitable for analysis of mins and maxes and for plotting programs that can handle unevenly spaced data.

**Table 87. Parameters**

| Name | Description |
|---|---|
| Tagname(s) | Specify all of the tag(s) on which to perform Trend2 sampling. |
| Starting time | Specify when the time period starts.<br><br>Values in the raw data whose timestamps fall on the starting time will be included in the results, if they are the minimum or the maximum in the interval. |
| Ending time | Specify when the time period ends.<br><br>Values in the raw data whose timestamps fall on the ending time will be included in the results, if they are the minimum or the maximum in the interval. |

The following determine the size of the intervals:

| Name | Description |
|---|---|
| Interval length | If you specify the interval length, then **Historian** splits the time period between start and end into as many intervals of that length as will fit in the period.<br><br>For example, if you have a 30 second time period, and you request intervals of 5 seconds, Historian will break the time period into 6 intervals, each of which covers 5 seconds.<br><br>If the sampling period does not evenly divide by the interval length, then Historian creates as many intervals of that length as will fit, and then create a remainder interval from whatever time is left. So, if we request intervals of 7 seconds for a 30 second time period, **Historian** splits the sampling period into 4 intervals of 7 seconds each, and one remainder interval of 2 seconds.<br><br>This behavior is in contrast to the original Trend sampling, which would simply ignore any leftover values at the end, rather than putting them into a smaller interval. |

| Name | Description |
|---|---|
| Number of samples | If you specify the number of samples to return, **Historian** determines the number of intervals to return. Each interval returns 2 samples, so **Historian** divides the time period between start and end into half as many intervals as there are specified samples.<br><br>For example, if you specify 12 samples, Historian will divide the time period into 6 intervals, because 12/2 = 6.<br><br>If the number of samples specified is odd, then it is rounded up to the nearest even number. So, if you ask for 7 samples, Historian rounds up to 8 samples, from 8/2 = 4 intervals. All intervals are of the same length.<br><br>If the time period from start to finish is 60 seconds and we request 10 intervals, then each interval will be 6 seconds long. |

## About Retrieving Data from Historian

After data collection, the Historian server compresses and stores the information in an .iha file in Data Archive. Any client application can retrieve archived data through the Historian API. The Historian API is a client/server programming interface that maintains connectivity to the Historian Server and provides functions for data storage and retrieval in a distributed network environment.

You can retrieve data from Historian using any number of clients, including but not limited to:

- Configuration Hub
- Historian Analysis
- Knowledge Center
- iFIX
- CIMPLICITY
- Real-Time Information Portal
- Dream Reports
- Excel Add-In
- Custom SDK Applications
- OLE DB

Historian exposes various sampling and calculation modes that are used on retrieval of data that has already been collected to the archive. These modes do not effect data collection. Some sampling modes

are suited to compressed data and should be used when collector compression or archive compression is used.

## Sampling Modes

Sampling modes are used to specify how the data will be retrieved from Historian. Several modes are available, such as CurrentValue, Interpolated, Calculated and RawByTime. Sampling modes are specified in the client you use to retrieve data from Historian.

For more information, refer to the *Advanced Topics* section in the online help.

- For the Trend sampling mode, Historian ignores any leftover values at the end, rather than putting them into a smaller interval.
- For the Trend2 sampling mode, Historian creates as many intervals of the interval length as will fit into the sampling period, and then creates a remainder interval from whatever time is left.

- Spacing of timestamps returned:
    - For the Trend sampling mode, Historian returns evenly-spaced interval timestamps.
    - For the Trend2 sampling mode, Historian returns raw sample timestamps. These timestamps can be unevenly spaced, since raw data can be unevenly spaced.
- Inclusion of start and end times entered:
    - The Trend sampling mode is start time exclusive and end time inclusive.
    - The Trend2 sampling mode is start time inclusive and end time inclusive.

Trend sampling mode is more suitable for plotting applications that prefer evenly-spaced data.

Trend2 sampling mode is more suitable for analysis of mins and maxes and for plotting programs that can handle unevenly spaced data.

**TrendtoRaw2**: The TrendtoRaw2 sampling mode is a modified version of the TrendtoRaw sampling mode.

The TrendtoRaw2 sampling mode almost always produces the same results as the Trend2 sampling mode. The exception is that, when more samples are requested than there are raw data points, the TrendtoRaw2 sampling mode returns all of the available raw data points with no further processing.

**Calculated**: Returns samples based on a selected Calculation mode.

**RawByFilterToggle**: RawByFilterToggle returns filtered time ranges. The values returned are 0 and 1. If the value is 1, then the condition is true and 0 means false.

This sampling mode is used with the time range and filter tag conditions. The result starts with a starting time stamp and ends with an ending timestamp

## Calculation Modes

Calculation modes are used when the sampling mode is set to Calculated. The data type of all calculated values will be DoubleFloat except for MinimumTime, MaximumTime, FirstRawTime and LastRawTime which will be a Date. The data type of the values of FirstRawValue and LastRawValue will be the same as that of the selected tag.

| Calculation Mode | Results |
|---|---|
| Count | Displays the number of raw samples in the specified interval. This only indicates the count and does not display the actual values or qualities of the samples.<br><br>The Count calculation mode is useful for analyzing the distribution of raw data samples. If you have a higher number of raw samples than expected, you may decide to implement collector or archive compression. If samples are missing, then you may want to slow your collection rates. |
| State Count | Displays the number of times a tag has transitioned to another state from a previous state. A state transition is counted when the previous good sample is not equal to the state value and the next good sample is equal to state value. |
| State Time | Displays the duration that a tag was in a given state within an interval. |
| Minimum | Displays the minimum value in a specified interval with good data quality. This value may be raw or interpolated.<br><br>**Note:**<br>The Minimum and MinimumTime calculation retrieve two additional samples per interval; one is interpolated at the interval start time and the other is interpolated at the interval end time. These samples are used to determine the min or max just like any raw value. |
| MinimumTime | Displays the time stamp of the minimum value in a specified interval.<br><br>See the note in Minimum for additional information. |
| Maximum | Displays the maximum value in a specified interval. |

| Calculation Mode | Results |
|---|---|
| | **Note:**<br>The Maximum and MaximumTime calculation internally retrieve two additional samples per interval; one is interpolated at the interval start time and the other is interpolated at the interval end time. These samples are used in the min or max just like any raw or interpolated value. |
| MaximumTime | Displays the time stamp of the maximum value in a specified interval.<br><br>See the note in Maximum for additional information. |
| RawAverage | Displays the arithmetic average of the raw values in a specified interval with good data quality. This is useful only when a sufficient number of raw data values are collected. |
| Average | Similar to RawAverage, but performs a special logic for time weighting and for computing the value at the start of the interval. This is useful for computing an average on compressed data. |
| OPCQOr and OPCQAnd | The OPCQOr is a bit wise OR operation of all the 16 bit OPC qualities of the raw samples stored in the specified interval.<br><br>The OPCQAnd is a bit wise AND operation of all the 16 bit OPC qualities of the raw samples stored in the specified interval. |
| Total | Retrieves the time-weighted total of raw and interpolated values for each calculation interval. The collected value must be a rate per 24 hours. This calculation mode determines a count from the collected rate. |
| Delta Queries | Historian offers the following delta queries to determine the delta over a time interval:<br><br>• DELTAPOS *(on page 1116)*<br>• DELTANEG *(on page 1132)*<br>• DELTA *(on page 1141)* |
| RawTotal | Displays the arithmetic sum of raw values in a specified interval. |
| StandardDeviation | Displays the time-weighted standard deviation of raw values for a specified interval. |

| Calculation Mode | Results |
|---|---|
| RawStandardDeviation | Displays the arithmetic standard deviation of raw values for a specified interval. |
| TimeGood | Displays the amount of time (in milliseconds) during an interval when the data is of good quality and matches filter conditions if the filter tag is used. |
| FirstRawValue | Returns the first good raw value for a specified time interval. |
| FirstRawTime | Returns the timestamp of the first good raw for a specified time interval. |
| LastRawValue | Returns the last good raw value for a specified time interval. |
| LastRawTime | Returns the timestamp of the last good raw for a specified time interval. |
| TagStats | Allows you to return multiple calculation modes for a tag in a single query. |

**Note:**
You can also use INCLUDEBAD or FILTERINCLUDEBAD as query modifiers to include bad quality data. For more information, refer INLUDEBAD and FILTERINCLUDEBAD sections in Advanced Topics.

## Query Modifiers

Query Modifiers are used for retrieving data that has been stored in the archive. They are used along with sampling and calculation modes to get a specific set of data.

- The time interval is great than 1 minute.
- The collection interval is greater than 1 second.
- The data node size is greater than the default 1400 bytes.
- The data type of the tags is String or Blob.

Query performance varies depending on all of the above factors.

Use this query modifier only with FirstRawValue, FirstRawTime, LastRawValue, and LastRawTime calculation modes.

**EXCLUDESTALE**:

- Stale tags are tags that have no new data samples within a specified period of time, and which have the potential to add to system overhead and slow down user queries.
- The EXCLUDESTALE query modifier allows for exclusion of stale tags in data queries.

- Unless permanently deleted, stale tags from the archiver are not removed but are simply marked as stale. Use the query without this query modifier to retrieve the sample values.
- Data is not returned for stale tags. An ihSTATUS_STALED_TAG error is returned instead.

## Filtered Data Queries

Filtered data queries enhance Historian by adding filter tags and additional filtering criteria to standard queries. Unfiltered data queries in Historian allow you to specify a start and end time for the query, then return all data samples within that interval. A filtered data query, however, will allow you to specify a condition to filter the results by, as well as calculation modes to perform on the returned data. Filtered data queries are performed on the Historian server.

For example, a filtered data query is useful when trying to retrieve all data for a specific Batch ID, Lot Number, or Product Code and for filtering data where certain limits were exceeded, such as all data where a temperature exceeded a certain value. Rather than filtering a full day's worth of process data in the client application, you can filter data in the Historian archiver, and only return the matching results to the client application. The result is a smaller, more relevant data set.

You can use filter criteria with raw, interpolated, and calculated sampling modes. You cannot use it with current value sampling. The logic of selecting intervals is always interpolated, even when the data retrieval is raw or calculated. The value that triggers a transition from false to true can be a raw value or interpolated value.

You cannot use a filtered data query in an iFIX chart. For more information, refer to Advanced Topics section in the online help.

## Filter Parameters for Data Queries

Use of filter parameters with a data query is optional.

- AND Condition
- OR Condition
- Combination of both AND and OR

Filter Expression can be used instead of FilterTag, FilterComparisonMode and FilterValue parameters. While using FilterExpression, the expression is passed within single quotes and for complex expressions we write the conditions within a parenthesis. There is no maximum length for a filter expression, but if it is called using OLE DB or Excel, they may have their own limitations.

**Filter Mode**: The type of time filter.

The Filter Mode defines how time periods before and after transitions in the filter condition should be handled.

For example, AfterTime indicates that the filter condition should be True starting at the timestamp of the archive value that triggered the True condition and leading up to the timestamp of the archive value that triggered the False condition.

**ExactTime**

Retrieves data for the exact times that the filter condition is True (only True).

**BeforeTime**

Retrieves data from the time of the last False filter condition up until the time of the True condition (False until True).

**AfterTime**

Retrieves data from the time of the True filter condition up until the time of next False condition (True until False).

**BeforeAndAfterTime**

Retrieves data from the time of the last False filter condition up until the time of next False condition (While True).

**Filter Comparison Mode**: Filter Comparison Mode is only used if Filter Tag is filled in. The Filter Comparison Mode defines how archive values for the Filter Tag should be compared to the Filter Value to establish the state of the filter condition. If a Filter Tag and Filter Comparison Value are supplied, time periods are filtered from the results where the filter condition is False.

The type of comparison to be made on the filter comparison value:

**Equal**

Filter condition is True when the Filter Tag is equal to the comparison value.

**EqualFirst**

Filter condition is True when the Filter Tag is equal to the first comparison value.

**EqualLast**

Filter condition is True when the Filter Tag is equal to the last comparison value.

**NotEqual**

Filter condition is True when the Filter Tag is NOT equal to the comparison value.

**LessThan**

Filter condition is True when the Filter Tag is less than the comparison value.

**GreaterThan**

Filter condition is True when the Filter Tag is greater than the comparison value.

**LessThanEqual**

Filter condition is True when the Filter Tag is less than or equal to the comparison value.

**GreaterThanEqual**

Filter condition is True when the Filter Tag is greater than or equal to the comparison value.

**AllBitsSet**

Filter condition is True when the binary value of the Filter Tag is equal to all the bits in the condition. It is represented as ^ to be used in Filter Expression.

**AnyBitSet**

Filter condition is True when the binary value of the Filter Tag is equal to any of the bits in the condition. It is represented as ~ to be used in Filter Expression.

**AnyBitNotSet**

Filter condition is True when the binary value of the Filter Tag is not equal to any one of the bits in the condition. It is represented as !~ to be used in Filter Expression.

**AllBitsNotSet**

Filter condition is True when the binary value of the Filter Tag is not equal to all the bits in the condition. It is represented as !^ to be used in Filter Expression.

**Alarm Condition**

Specifies an alarm condition to filter data by. For example, Level.

**Alarm SubCondition**

Specifies an alarm sub-condition to filter data by. For example, HIHI.

**Filter Comparison Value**: Filter Comparison Value is only used if Filter Tag is filled in. The value to compare the filter tag with when applying the appropriate filter to the data record set query (to determine the appropriate filter times).

## Filtered Queries in the Excel Add-in Example

This example shows how a filtered data query returns specific data from the Historian archive. The example uses two tags: `batchid` and ramp. The `batchid` tag is updated before a new batch is produced with the new batch's ID. The `ramp` tag contains raw data sent by a device in the process. In this example, it

is requested that Historian return data samples at ten second intervals for the `ramp` tag during the period that the `batchid` tag is set to B1.

A standard query in Historian for the ramp tag's values between 08:00 and 08:01, at ten second intervals, would look like this:

| Time Stamp | Value | Data Quality |
|---|---|---|
| 07/30/2003 08:00:00 | B0 | Good |
| 07/30/2003 08:00:20 | B1 | Good |
| 07/30/2003 08:00:45 | B2 | Good |

## Filtering Data Queries in the Excel Add-in

You can enter your filter conditions using Filter tag, Filter Comparison Mode, and Filter Comparison Value or you can put that all that information in a single FilterExpression. You can enter the filter conditions in the FilterExpression field of the **Historian Data Query** window. The filter conditions are passed within single quotes.

To find the values of the `ramp` tag for the B1 batch, enter the following values into the **Historian Filtered Data Query** window:

1. In the `Tag Name(s)` field, enter the tag you want to receive results from - the `ramp` tag in this example.
2. Select a start and end time for your query.
3. In the `Filter Tag` field, enter the tag you want to enable filtering with - `batchid` in this example.
4. In the `Filter Comparison` field, select your comparison condition.
5. n the `Include Data Where Value Is` field, enter your filter condition value.
6. In the `Include Times` field, select your filter mode.
7. In the `Sampling Type` field, select your sampling mode.
8. In the `Calculation` field, select your calculation mode.
9. Select your `Sampling Interval`.
10. In the `Output Display` field, select the tag values you want to display.

## Hybrid Modes

Hybrid mode is an advanced method of sampling collected data for trending. This mode of sampling has the ability to switch between sampled (like interpolated or trend) and raw data based on the actual and

requested number of samples or a specified time interval. The purpose of these modes is to return the minimum number of points to speed and simplify trending .

Hybrid mode is available for Interpolated, Lab, Trend, and Trend2 modes of sampling.

In these hybrid modes, the behavior is as follows

- If the actual number of stored samples is fewer than requested you will receive the raw data samples.
- If the actual number of stored samples is fewer than requested you will receive the raw data samples.

**Data for Examples**

All queries in this section use this set of data. The data here can be entered into Historian as a CSV file using the File collector. The queries can all be run in Historian Interactive SQL.

```
[Tags]
Tagname,DataType
TagA,DoubleInteger
[Data]
Tagname,Timestamp,Value,Quality
TagA,01/06/2014 12:00:01 PM,40000000,Good
TagA,01/06/2014 12:00:02 PM,30696808,Good
TagA,01/06/2014 12:00:03 PM,1952308224,Good
TagA,01/06/2014 12:00:04 PM,672641664,Good
TagA,01/06/2014 12:00:05 PM,636126336,Good
TagA,01/06/2014 12:00:06 PM,1826624640,Good
TagA,01/06/2014 12:00:07 PM,838753408,Good
TagA,01/06/2014 12:00:08 PM,520660896,Good
TagA,01/06/2014 12:00:09 PM,1293350272,Good
TagA,01/06/2014 12:00:10 PM,1959451264,Good
TagA,01/06/2014 12:00:11 PM,89220576,Good
TagA,01/06/2014 12:00:12 PM,1951745280,Good
TagA,01/06/2014 12:00:13 PM,888276160,Good
TagA,01/06/2014 12:00:14 PM,1031795200,Good
TagA,01/06/2014 12:00:15 PM,1449288960,Good
TagA,01/06/2014 12:00:16 PM,1516603392,Good
TagA,01/06/2014 12:00:17 PM,1843676544,Good
TagA,01/06/2014 12:00:18 PM,1672796672,Good
```

```
TagA,01/06/2014 12:00:19 PM,1533833984,Good
TagA,01/06/2014 12:00:20 PM,1697586560,Good
TagA,01/06/2014 12:00:21 PM,1647121280,Good
TagA,01/06/2014 12:00:22 PM,543921472,Good
TagA,01/06/2014 12:00:23 PM,1141920768,Good
TagA,01/06/2014 12:00:24 PM,540008448,Good
TagA,01/06/2014 12:00:25 PM,731087232,Good
TagA,01/06/2014 12:00:26 PM,631079296,Good
TagA,01/06/2014 12:00:27 PM,1160291968,Good
TagA,01/06/2014 12:00:28 PM,1324413696,Good
TagA,01/06/2014 12:00:29 PM,1875167744,Good
TagA,01/06/2014 12:00:30 PM,390197280,Good
TagA,01/06/2014 12:00:31 PM,192162736,Good
TagA,01/06/2014 12:00:32 PM,646106624,Good
TagA,01/06/2014 12:00:33 PM,210439200,Good
TagA,01/06/2014 12:00:34 PM,675144064,Good
TagA,01/06/2014 12:00:35 PM,1421636224,Good
TagA,01/06/2014 12:00:36 PM,537191872,Good
TagA,01/06/2014 12:00:37 PM,492214752,Good
TagA,01/06/2014 12:00:38 PM,1376227840,Good
TagA,01/06/2014 12:00:39 PM,1085046656,Good
TagA,01/06/2014 12:00:40 PM,924105984,Good
TagA,01/06/2014 12:00:41 PM,1294991488,Good
TagA,01/06/2014 12:00:42 PM,1737416960,Good
TagA,01/06/2014 12:00:43 PM,582910848,Good
TagA,01/06/2014 12:00:44 PM,1745973760,Good
TagA,01/06/2014 12:00:45 PM,1607484928,Good
TagA,01/06/2014 12:00:46 PM,2005492352,Good
TagA,01/06/2014 12:00:47 PM,746677184,Good
TagA,01/06/2014 12:00:48 PM,2143539456,Good
TagA,01/06/2014 12:00:49 PM,2009761664,Good
TagA,01/06/2014 12:00:50 PM,640139968,Good
TagA,01/06/2014 12:00:51 PM,990464704,Good
TagA,01/06/2014 12:00:52 PM,109999792,Good
TagA,01/06/2014 12:00:53 PM,1269805568,Good
TagA,01/06/2014 12:00:54 PM,1111627520,Good
TagA,01/06/2014 12:00:55 PM,60175184,Good
```

```
TagA,01/06/2014 12:00:56 PM,1407366400,Good
TagA,01/06/2014 12:00:57 PM,928761280,Good
TagA,01/06/2014 12:00:58 PM,1666397696,Good
TagA,01/06/2014 12:00:59 PM,438304832,Good
TagA,01/06/2014 12:01:00 PM,1179844864,Good
TagA,01/07/2014 06:00:01 PM,9000,Good
TagA,01/07/2014 06:00:02 PM,5,Good
TagA,01/07/2014 06:00:03 PM,8,Good
TagA,01/07/2014 06:00:04 PM,-1,Good
TagA,01/07/2014 06:00:05 PM,4,Good
TagA,01/07/2014 06:00:06 PM,485,Good
TagA,01/07/2014 06:00:07 PM,-30000,Good
TagA,01/07/2014 06:00:08 PM,2,Good
TagA,01/07/2014 06:00:09 PM,4,Good
TagA,01/07/2014 06:00:10 PM,-60000,Good
TagA,01/07/2014 06:00:11 PM,60000,Good
TagA,01/07/2014 06:00:12 PM,1,Good
TagA,01/07/2014 06:00:13 PM,1,Good
TagA,01/07/2014 06:00:14 PM,30,Good
TagA,01/07/2014 06:00:15 PM,-70000,Good
TagA,01/07/2014 06:00:16 PM,-70000,Good
TagA,01/07/2014 06:00:17 PM,5,Good
TagA,01/07/2014 06:00:18 PM,1,Good
TagA,01/07/2014 06:00:19 PM,8,Good
TagA,01/07/2014 06:00:20 PM,220,Good
TagA,01/07/2014 06:00:21 PM,45,Good
TagA,01/07/2014 06:00:22 PM,44,Good
TagA,01/07/2014 06:00:23 PM,12,Good
TagA,01/07/2014 06:00:24 PM,13,Good
TagA,01/07/2014 06:00:25 PM,-5600,Good
TagA,01/07/2014 06:00:26 PM,15,Good
TagA,01/07/2014 06:00:27 PM,0,Good
TagA,01/07/2014 06:00:28 PM,25000,Good
TagA,01/08/2014 09:00:01 AM,1400,Good
TagA,01/08/2014 09:00:02 AM,0,Good
TagA,01/08/2014 09:00:03 AM,16,Good
TagA,01/08/2014 09:00:04 AM,-1400,Good
```

```
TagA,01/08/2014 09:00:05 AM,-12,Good
TagA,01/08/2014 09:00:06 AM,125,Good
TagA,01/08/2014 09:00:07 AM,150,Good
TagA,01/08/2014 09:00:08 AM,13,Good
TagA,01/08/2014 09:00:09 AM,-56,Good
TagA,01/08/2014 09:00:10 AM,12,Good
TagA,01/08/2014 09:00:11 AM,45,Good
```

This following examples provide various cases of the InterpolatedtoRaw hybrid mode illustrating the switching of data between raw and calculated data.

The following data is used in the example below. You can import this data into Historian if you want to try the example yourself:

```
Tag1 5/16/2011 15:52:24 1,000.0000000 100.0000000
Tag1 5/16/2011 15:52:25 1,001.0000000 100.0000000
Tag1 5/16/2011 15:52:26 1,002.0000000 100.0000000
Tag1 5/16/2011 15:52:27 1,003.0000000 100.0000000
Tag1 5/16/2011 15:52:28 1,004.0000000 100.0000000
Tag1 5/16/2011 15:52:29 1,005.0000000 100.0000000
Tag1 5/16/2011 15:52:30 1,006.0000000 100.0000000
```

### Case 1

Use the following query to retrieve data for Tag 1 where it requests for 5 samples using InterpolatedtoRaw mode.

```
SET starttime= '5/16/2011 15:52:05 PM', endtime= '5/16/2011 15:52:47 PM', numberofsamples = 5, samplingmode=
 Interpolatedtoraw SELECT * FROM ihrawdata where tagname = "TAG1"
```

The query will return interpolated data as shown below because the actual number of raw samples (7) is greater than the requested number of samples (5):

| tagname | timesstamp | value | quality | samplingmode | numberof-samples |
|---|---|---|---|---|---|
| Tag1 | 5/16/2011 15:52:13 | 0.0000000 | 0.0000000 | InterpolatedtoRaw | 5 |
| Tag1 | 5/16/2011 15:52:21 | 0.0000000 | 0.0000000 | InterpolatedtoRaw | 5 |

| tagname | timesstamp | value | quality | samplingmode | numberof-samples |
|---|---|---|---|---|---|
| Tag1 | 5/16/2011 15:52:30 | 1,006.0000000 | 100.0000000 | InterpolatedtoRaw | 5 |
| Tag1 | 5/16/2011 15:52:38 | 1,006.0000000 | 100.0000000 | InterpolatedtoRaw | 5 |
| Tag1 | 5/16/2011 15:52:47 | 1,006.0000000 | 100.0000000 | InterpolatedtoRaw | 5 |

### Case 2

Use the following query to retrieve data for Tag 1 where it requests for 50 samples using InterpolatedtoRaw mode.

```
starttime= '5/16/2011 3:52:05 PM', endtime= '5/16/2011 3:52:47 PM', numberofsamples = 50, sampling- mode=
 Interpolatedtoraw SELECT & FROM ihrawdata where tagname = "TAG1"
```

The query will return raw data as shown below because the actual sample count(7) is less than the requested sample count (50):

| tagname | timesstamp | value | quality | samplingmode | numberof-samples |
|---|---|---|---|---|---|
| Tag1 | 5/16/2011 15:52:24 | 1,000.0000000 | 100.0000000 | InterpolatedtoRaw | 50 |
| Tag1 | 5/16/2011 15:52:25 | 1,001.0000000 | 100.0000000 | InterpolatedtoRaw | 50 |
| Tag1 | 5/16/2011 15:52:26 | 1,002.0000000 | 100.0000000 | InterpolatedtoRaw | 50 |
| Tag1 | 5/16/2011 15:52:27 | 1,003.0000000 | 100.0000000 | InterpolatedtoRaw | 50 |
| Tag1 | 5/16/2011 15:52:28 | 1,004.0000000 | 100.0000000 | InterpolatedtoRaw | 50 |
| Tag1 | 5/16/2011 15:52:29 | 1,005.0000000 | 100.0000000 | InterpolatedtoRaw | 50 |

| tagname | timesstamp | value | quality | samplingmode | numberof-samples |
|---|---|---|---|---|---|
| Tag1 | 5/16/2011 15:52:30 | 1,006.0000000 | 100.0000000 | InterpolatedtoRaw | 50 |

### Case 3

Use the following query to retrieve data for Tag 1 where it requests for samples in a time interval (milliseconds), using InterpolatedtoRaw mode.

```
SET starttime= '5/16/2011 3:52:05 PM', endtime= '5/16/2011 3:52:25 PM', intervalmilliseconds=10s , samplingmode=

 Interpolatedtoraw

Tag1 5/16/2011 15:52:24 1,000.0000000 100.0000000

Tag1 5/16/2011 15:52:25 1,001.0000000 100.0000000
```

The query will return interpolated data as shown below because the actual number of raw samples (7) is greater than the requested number of samples (5):

| Tagname | Timestamp | Value | Quality | Sampling MOde |
|---|---|---|---|---|
| Tag1 | 5/16/2011 15:52:24 | 1,000.0000000 | 1,000.0000000 | InterpolatedtoRaw |
| Tag1 | 5/16/2011 15:52:25 | 1,001.0000000 | 1,000.0000000 | InterpolatedtoRaw |

## Calculation Modes

This information is intended to supplement the information in the Historian product documentation and the SDK Help File.

Sampling and calculation modes are used on retrieval of data that has already been collected to the archive. Calculation modes are used when the sampling mode is set to "Calculated". It is helpful to separate the many modes into 3 main categories from simplest to most complex. A detailed explanation of the calculation modes with examples are discussed in following topics:

**Raw Calculation Modes**: Easiest to understand. Only raw points are used to determine calculated value.

- Count
- RawTotal
- RawAverage

- RawStandardDeviation
- FirstRawValue
- FirstRawTime
- LastRawValue
- LastRawTime

**Interpolated Calculation Modes** – both raw and interpolated points are used to determine a value.

- Minimum
- MinimumTime
- Maximum
- MaximumTime
- TimeGood

**Delta Query Calculation Modes** - Determine the delta over a time interval.

- DELTAPOS *(on page 1116)*
- DELTANEG *(on page 1132)*
- DELTA *(on page 1141)*

**Time Weighted Calculation Modes** – Most complicated. Time weighting on raw and interpolated points is used to determine a value.

- Average
- Total
- StandardDeviation

**Other Calculation Modes**

- STATECOUNT
- STATETIME
- OPCQOR and OPCQAND
- TagStats

Each sample retrieved from Historian has a timestamp, value, and quality.

- **Timestamp** - the same logic as for interpolated values. It is covered in detail in the Understanding Sampling Modes document and not covered at all in this document.
- **Value** – depends entirely on the calculation mode being used
- **Quality** - Depending on the calculation mode, this either means:

- The percent of raw samples vs. total raw samples in the interval that were of good data quality.
- The percent of time in the interval that the data was of good data quality

Filtered data queries are described in the ***Filtered data Queries*** section.

**Note:**
This document intentionally contains all necessary data for executing the examples and reproducing the results. The tags and data are in the form of a CSV to be imported with the Historian File collector. You will not be able to import the data unless you adjust the active hours setting and possibly the Create Offline Archives setting of the archiver. This is true any time you are importing old data with the File collector.

## Raw Calculation Modes

The calculation modes use only collected raw samples to determine the value for each interval.

**Count Mode:**

- **Value:** The count of raw samples with good quality in the interval. The values of the each sample are ignored. The Count does not include any samples with bad quality, including the start and end of collection markers.
- **Quality:** Percent good is always 100, even if the interval does not contain any raw samples or contains only bad quality samples.
- **Anticipated Usage:** Count is useful for analyzing the distribution of the raw data samples to determine the effect of compression deadbands. It is also useful to determine which tags are consuming the most archive space.

**RawTotal Mode:** Retrieves the arithmetic total (sum) of sampled values for each interval.

- **Value:** The sum of the good quality values of all raw samples in the interval. All bad quality samples are ignored.
- **Quality:** Percent good is always 100, even if the interval does not contain any raw samples or it contains only bad quality samples.
- **Anticipated Usage:** RawTotal mode is useful for calculating an accurate total when a sufficient number of raw samples are collected. Note that unlike ihTotal, this is a simple sum with no assumption that the values are rate values.

**RawAverage Mode:** The arithmetic average (mean) of all good quality raw samples in the interval.

- **Value:** The sum of all good quality samples in the interval, divided by the number of good quality samples in the interval. All bad quality samples are ignored. That is, RawAverage is equivalent to the RawTotal divided by Count.
- **Quality:** If there are no raw samples in the interval or they all have bad quality, then the percent good is 0.  Otherwise, percent good is always 100, even if the interval contains bad quality samples.

```
select timestamp, value, quality from ihrawdata where samplingmode=calculated
and calculationmode=rawaverage and timestamp >= '29-Mar-2002 13:30' and
timestamp <= '29-Mar-2002 14:30' and tagname  = counttag and intervalmilliseconds = 10M
```

- **Anticipated Usage:** The RawAverage mode is useful for calculating an accurate average when a sufficient number of raw samples are collected.

**RawAverage Mode:**

- **Value:**
- **Quality:**
- **Anticipated Usage:**

**RawStandardDeviation Mode:** Retrieves the arithmetic standard deviation of raw values for each calculation interval.

- **Value:** Any raw point of bad data quality is ignored.
- **Quality:** If there are no raw samples in the interval or they all have bad quality, then the percent good is 0. Otherwise, percent good is always 100, even if the interval contains bad quality samples.
- **Anticipated Usage:** The RawStandardDeviation mode is useful for calculating an accurate standard deviation when a sufficient number of raw samples are collected.

**FirstRawValue/FirstRawTime Modes:** Retrieve the first good raw sample value and timestamp for a given time interval, respectively.

- **Value:** The value of the raw sample or zero if there are no good raw samples in the interval. The timestamp of the sample or the year 1969 if there are no good raw samples in the interval.
- **Quality:** The quality is the same for FirstRawValue and First RawTime. If there are no good raw samples in the interval, then the percent good is 0. Otherwise, the percent good is always 100, even if the interval contains bad quality samples.

The Raw sample has a quality of Good, Bad or Uncertain, and that is converted to a 0 or 100 percent.

- **Anticipated Usage:**

**LastRawValue/LastRawTime Modes:** Retrieve the last good raw sample value and timestamp for a given time interval, respectively.

- **Value:** The value of the raw sample or zero if there are no good raw samples in the interval. The timestamp of the sample or the year 1969 if there are no good raw samples in the interval.
- **Quality:**

  The quality is the same for LastRawValue and LastRawTime. If there are no good raw samples in the interval, then the percent good is 0. Otherwise, percent good is always 100, even if the interval contains bad quality samples.

  The Raw sample has a quality of Good, Bad or Uncertain, and that is converted to a 0 or 100 percent.

- **Anticipated Usage:**

## Calculating the count of raw samples

The following example demonstrates that only good samples are counted. Importing the following data ensures that at least one interval has 0 samples.

```
[Tags]
Tagname,DataType,HiEngineeringUnits,LoEngineeringUnits
COUNTTAG,SingleInteger,100,0
[Data]
Tagname,TimeStamp,Value,DataQuality
COUNTTAG,29-Mar-2002 13:59:00.000,22,Good
COUNTTAG,29-Mar-2002 14:08:00.000,12,Bad
COUNTTAG,29-Mar-2002 14:22:00.000,4,Good
```

The following query retrieves data with a start time of 14:00 and an end time of 14:30 with a 10-minute interval.

```
select timestamp, value, quality from ihrawdata where samplingmode=calculated and calculationmode=count and timestamp
 >='29-Mar-2002 14:00' and timestamp <= '29-Mar-2002 14:30' and tagname  = counttag and intervalmilliseconds = 10M
```

| Time Stamp | Value | Quality |
|---|---|---|
| 29-Mar-200214:10:00.000 | 0.00 | 100.00 |
| 29-Mar-200214:20:00.000 | 0.00 | 100.00 |
| 29-Mar-200214:30:00.000 | 1.00 | 100.00 |

**Note:**
The bad raw sample at 14:08 is not counted, but the good sample at 14:22 is counted. The 14:11 to 14:20 interval has no raw samples, but still has a percent good of 100 percent.

### Calculating the Raw Total

The following example demonstrates that only good quality samples are included in the sum. Perform the fol- lowing query on the same data set as that in the Count example above:

```
select timestamp, value, quality from ihrawdata where samplingmode=calculated and calculationmode=rawtotal and
 timestamp >= '29-Mar-2002 13:30' and timestamp <= '29-Mar-2002 14:30' and tagname  = counttag and intervalmilliseconds
 = 10M
```

| Time Stamp | Value | Quality |
|---|---|---|
| 29-Mar-200213:40:00.000 | 0.00 | 100.00 |
| 29-Mar-200213:50:00.000 | 0.00 | 100.00 |
| 29-Mar-200214:00:00.00 | 22.00 | 100.00 |
| 29-Mar-200214:10:00.000 | 0.00 | 100.00 |
| 29-Mar-200214:20:00.000 | 0.00 | 100.00 |
| 29-Mar-200214:30:00.000 | 4.00 | 100.00 |

If the same start and end time are used, but the time span is treated as a single interval, then all values are added together:

```
select timestamp, value, quality from ihrawdata where samplingmode=calculated and calculationmode=rawtotal and
 timestamp >= '29-Mar-2002 13:30' and timestamp <= '29-Mar-2002 14:30' and tagname  = counttag
and numberofsamples=1
```

| Time Stamp | Value | Quality |
|---|---|---|
| 29-Mar-200214:30:00.000 | 26.00 | 100.00 |

Even though the time span covers all raw samples, only the two good quality samples are used in the calculation: 26 = 22 + 4

### Calculating RawAverage

The following example demonstrates that only good quality samples are included in RawAverage. Perform the following query on the same data set as that in the Count example above. This query retrieves data using RawAverage, with a start time of 13:30 and an end time of 14:30 at 10-minute intervals.

```
select timestamp, value, quality from ihrawdata where samplingmode=calculated and calculationmode=rawaverage and
timestamp >= '29-Mar-2002 13:30' and timestamp <= '29-Mar-2002 14:30' and tagname  = counttag and intervalmilliseconds
 = 10M
```

| Time Stamp | Value | Quality |
|---|---|---|
| 29-Mar-200213:40:00.000 | 0.00 | 0.00 |
| 29-Mar-200213:50:00.000 | 0.00 | 0.00 |
| 29-Mar-200214:00:00.000 | 22.00 | 100.00 |
| 29-Mar-200214:10:00.000 | 0.00 | 0.00 |
| 29-Mar-200214:30:00.000 | 4.00 | 100.00 |

The interval from 14:11 to 14:20 has no raw samples. The percent good quality of 0.

The interval from 14:01 to 14:10 has 0 good and 1 bad samples. It also has a percent good quality of 0.

The interval from 14:21 to 14:30 has 1 good and 0 bad samples. It has a percent good quality of 100.

If the same start and end time are used, but the time span is treated as a single interval, then all values are averaged together:

```
select timestamp, value, quality from ihrawdata where samplingmode=calculated and calculationmode=rawaverage
and timestamp >= '29-Mar-2002 13:30' and timestamp <= '29-Mar-2002 14:30' and tagname  = counttag and
numberofsamples=1
```

| Time Stamp | Value | Quality |
|---|---|---|
| 29-Mar-200214:30:00.000 | 13.00 | 100.00 |

Even though the time span covers all raw samples, but only the two good samples are used in the calculation:  13 = (22+4)/2 Since the interval includes at least one good quality sample, percent good for the interval is 100, even though 33% of the samples are of bad quality.

## Calculating the Raw Standard Deviation

The following example demonstrates that only good samples are included in the standard deviation. Perform the following query on the same data set as that in the Count example above:

```
select timestamp, value, quality from ihrawdata where samplingmode=calculated and calculationmode=rawstandarddeviation
 and
timestamp >= '29-Mar-2002 13:30' and timestamp <= '29-Mar-2002 14:30' and tagname  = counttag and numberofsamples=1
```

| Time Stamp | Value | Quality |
|---|---|---|
| 29-Mar-200214:30:00.000 | 12.73 | 100.00 |

## Retrieving the FirstRawValue/FirstRawTime Values

Import this data to Historian:

```
[Tags]
Tagname,DataType,HiEngineeringUnits,LoEngineeringUnits
Tag1,SingleFloat,60,0
[Data]
Tagname,TimeStamp,Value,DataQuality
Tag1,07-05-2011 17:24:00,29.72,Bad
Tag1,07-05-2011 17:25:00,29.6,Good
Tag1,07-05-2011 17:26:00,29.55,Good
Tag1,07-05-2011 17:27:00,29.49,Bad
Tag1,07-05-2011 17:28:00,29.53,Bad
Tag1,07-05-2011 17:29:00,29.58,Good
Tag1,07-05-2011 17:30:00,29.61,Bad
Tag1,07-05-2011 17:31:00,29.63,Bad
Tag1,07-05-2011 18:19:00,30,Good
Tag1,07-05-2011 18:20:00,29.96,Good
Tag1,07-05-2011 18:21:00,29.89,Good
```

```
Tag1,07-05-2011 18:22:00,29.84,Good

Tag1,07-05-2011 18:23:00,29.81,Bad
```

#### Using FirstRawValue Calculation Mode

```
set starttime='07-05-2011 16:00:00', endtime='07-05-2011 19:00:00'

select timestamp,value,quality from ihrawdata where tagname like 'Tag1' and samplingMode=Calculated and

CalculationMode=FirstRawValue and intervalmilliseconds=1h
```

The output is as follows:

| Time Stamp | Value | Quality |
|---|---|---|
| 07-05-201117:00:00 | 0.0000000 | 0.0000000 |
| 07-05-201118:00:00 | 29.6000000 | 100.0000000 |
| 07-05-201119:00:00 | 30.0 | 100.0000000 |

For the time interval 16:00 to 17:00 there are no raw values so a value and quality of 0 is returned for both FirstRawValue and FirstRawTime. The first raw sample from17:00 to 18:00 is 29.72 but it is a bad data quality so it is skipped and the 29.6 is returned and its timestamp of 17:25 is returned in FirstRawTime. FirstRawValue calculation mode considers only good quality data. In the last interval the first good raw sample is 30 and is returned and its timestamp is returned as FirstRawTime.

### Retrieving the LastRawValue/LastRawTime Values

Import this data into Historian

```
[Tags]Tagname,DataType

DecimatedOneHour,DoubleInteger

[Data]

Tagname,Timestamp,Value,DataQuality

Tag1,07-05-2011 17:29:00,29,Good

Tag1,07-05-2011 20:00:00,0,Good

Tag1,07-05-2011 20:12:00,12,Good

Tag1,07-05-2011 20:15:00,0,Bad
```

#### Using LastRawValue Calculation Mode

```
set starttime='07-05-2011 17:00:00',endtime=' 07-05-2011 21:00:00'

select timestamp,value,quality from ihrawdata where tagname like Tag1 and samplingmode=Calculated and

CalculationMode=LastRawValue and Intervalmilliseconds=1h
```

The output is as follows:

| Time Stamp | Value | Quality |
|---|---|---|
| 07-05-201118:00:00 | 29 | 100.0000000 |
| 07-05-201119:00:00 | 0 | 0.0000000 |
| 07-05-201120:00:00 | 0 | 100.0000000 |
| 07-05-201121:00:00 | 12 | 100.0000000 |

In the interval from 17:00 to 18:00 the last good value is 29. The 18:00 to 19:00 has no raw samples so the quality is bad. The 20:00 sample is returned as the last good value in the 19:00 to 20:00. In the final interval, the last raw sample is bad quality so it is ignored and the previous sample is returned.

**Using LastRawTime Calculation Mode**

```
set starttime='07-05-2011 17:00:00',endtime=' 07-05-2011 21:00:00'

select timestamp,value,quality from ihrawdata where tagname like Tag1 and samplingmode=Calculated and CalculationMode=

LastRawTime and Intervalmilliseconds=1h
```

The output is as follows:

| Time Stamp | Value | Quality |
|---|---|---|
| 07-05-201117:00:00 | 07-05-201117:29:00 | 100.0000000 |
| 07-05-201118:00:00 | 01-01-197005:30:00 | 0.0000000 |
| 07-05-201119:00:00 | 07-05-201120:00:00 | 100.0000000 |
| 07-05-201120:00:00 | 07-05-201120:12:00 | 100.0000000 |

**Note:**
You can also use the INCLUDEBAD query modifier to include bad quality data.

## Interpolated Calculation Modes

Interpolation is used in many calculation modes. When using interpolated data, it is possible that there are no raw samples in the interval (such as with highly-compressed data) so the archiver requires additional samples to perform calculations.

The Minimum, MinimumTime, Maximum, and MaximumTime all use interpolation to arrive at two additional samples per interval. One is interpolated at the interval start time and one is interpolated at the interval end time. The interpolated samples are used in calculations just like raw, collected samples within the interval. In particular, the minimum or maximum calculated value can be a raw or interpolated value.

All described rules for interpolating a value at an interval's end time also apply to the interval's start time. There is no raw maximum or raw minimum sampling mode. To acquire these values, you must retrieve the raw samples using RawByTime or RawByNumber and compute the minimum or maximum yourself.

Similarly, you must also manually calculate a minimum or maximum when using values acquired through lab sampling.

**Minimum/Maximum and MinimumTime/MaximumTime Modes:** The minimum (or maximum) value in the interval and the time stamp of that value.

- **Value:**

  Maximum returns the raw or interpolated value with the greatest value and good data quality in the interval. Minimum returns the raw or interpolated value with the lowest value and good data quality in the interval

  MaximumTime returns the time stamp of the Maximum value. MinimumTime returns the time stamp of the Minimum value.

  In all cases, all raw samples of bad quality is ignored, both during interpolation and when calculating the maximum.
- **Quality:** If the raw samples in the interval all have bad quality, or if the sample before the interval has bad quality, then percent good is 0. Otherwise, percent good is always 100, even if the interval does not contain any raw samples or contains both good and bad quality samples.

**TimeGood Mode:** The TimeGood mode calculates the amount of time for which the data was of good quality.

The TimeGood mode is most useful when combined with filtered data queries. You can use a filter condition to acquire samples for which a specific condition was true, then calculate for how long that data was of a good quality. For example, you could use a filter condition to determine the amount of time a pump was activated, then calculate for how much of that time the data was of a good quality.

To get the most use out of the TimeGood mode, you should understand how filtered data queries work.

- **Value:** The TimeGood mode retrieves the total number of milliseconds during the interval for which the data is good AND for which the filter condition is true. If there is no filter tag or condition, then TimeGood is the total number of milliseconds in the interval that the data is good.
- **Quality:** The TimeGood mode always has a percent good of 100, even if there are no raw samples or if all samples have bad quality. In the latter case, the Value will be 0, but the percent good is still 100.

### Finding minimum and maximum of Downward Sloping Data

The following example demonstrates how a raw sample is interpolated at the interval's start and end time and how this interpolation is used with raw samples when calculating minimum and maximum values.

Import the following data:

```
[Tags]

Tagname,DataType,HiEngineeringUnits,LoEngineeringUnits

DOWNSLOPE,SingleFloat,100,0

[Data]

Tagname,TimeStamp,Value,DataQuality

DOWNSLOPE,29-Mar-2002 13:59:00.000,22,Good

DOWNSLOPE,29-Mar-2002 14:08:00.000,12,Good

DOWNSLOPE,29-Mar-2002 14:22:00.000,4,Good
```

The following query retrieves the Maximum value:

```
select timestamp, value, quality from ihrawdata where samplingmode=calculated

and calculationmode=Maximum and timestamp >= '29-Mar-2002 13:50' and

timestamp <= '29-Mar-2002 14:30' and tagname  = DOWNSLOPE and numberofsamples = 8
```

The following query retrieves the MaximumTime:

```
select timestamp, value, quality from ihrawdata where samplingmode=calculated

and calculationmode=MaximumTime and timestamp >= '29-Mar-2002 13:50' and

timestamp <= '29-Mar-2002 14:30' and tagname  = DOWNSLOPE and numberofsamples = 8
```

The following query retrieves the Minimum:

```
select timestamp, value, quality from ihrawdata where samplingmode=calculated

and calculationmode=Minimum and timestamp >= '29-Mar-2002 13:50' and

timestamp <= '29-Mar-2002 14:30' and tagname  = DOWNSLOPE and numberofsamples = 8
```

The following query retrieves the MaximumTime value:

```
select timestamp, value, quality from ihrawdata where samplingmode=calculated
and calculationmode=MinimumTime and timestamp >= '29-Mar-2002 13:50' and
timestamp <= '29-Mar-2002 14:30' and tagname  = DOWNSLOPE and numberofsamples = 8
```

The following query retrieves the Minimum value:

```
select timestamp, value, quality from ihrawdata where samplingmode=calculated
and calculationmode=Min
```

The following query retrieves the MinimumTime value:

```
select timestamp, value, quality from ihrawdata where samplingmode=calculated
and calculationmode=MinimumTime and timestamp >= '29-Mar-2002 13:50' and
timestamp <= '29-Mar-2002 14:30' and tagname  = DOWNSLOPE and numberofsamples = 8
```

| Interval Time Stamp | Maximum | Maximum Time | Minimum | Minimum Time | Quality |
|---|---|---|---|---|---|
| 13:55:00.000 | 0.00 | 31-Dec-1969 19:00:00.000 | 0.00 | 31-Dec-1969 19:00:00.000 | 0.00 |
| 14:00:00.000 | 22.00 | 13:59:00.000 | 20.89 | 14:00:00.000 | 100.00 |
| 14:05:00.000 | 20.89 | 14:00:00.000 | 15.33 | 14:05:00.000 | 100.00 |
| 14:10:00.000 | 15.33 | 14:05:00.000 | 10.86 | 14:10:00.000 | 100.00 |
| 14:15:00.000 | 10.86 | 14:10:00.000 | 8.00 | 14:15:00.000 | 100.00 |
| 14:20:00.000 | 8.00 | 14:15:00.000 | 5.14 | 14:20:00.000 | 100.00 |
| 14:25:00.000 | 5.14 | 14:20:00.000 | 4.00 | 14:25:00.000 | 100.00 |
| 14:30:00.000 | 4.00 | 14:30:00.000 | 4.00 | 14:30:00.000 | 100.00 |

The value is 4 for the entire interval of 14:26:00 to 14:30:00. However, the newest value is always returned for MinimumTime and MaximumTime for an interval, so the values instead are calculated as 14:30.

All modes have the same quality. A MaximumTime or MinimumTime of 1969 means there is no value in that interval.

Maximum always begins at the start of the interval because the data forms this is a downwards-sloping line. The Maximum takes the sample interpolated at the interval start time. The timestamp is still the interval end time.

When an interval has no raw samples, such as in the 14:05 interval, samples are interpolated at the beginning and the end of the interval. This means that the 14:05 interval has 2 samples to example at when calculating the Minimum or Maximum.

### Finding Minimum and Maximum of Changing Data

The following example uses a value that continually changes, rather than one that simply slopes upwards or downwards. Any Minimum or Maximum within an interval is necessarily a raw sample. If the minimum or maximum occurred as raw samples in the middle of the interval, these are also detected.

Import the following data:

```
[Tags]
Tagname,DataType,HiEngineeringUnits,LoEngineeringUnits
SAWTOOTH,SingleFloat,60,0


[Data]
Tagname,TimeStamp,Value,DataQuality
SAWTOOTH,29-Mar-2002 13:59:00.000,22.7,Good
SAWTOOTH,29-Mar-2002 14:01:00.000,12.5,Good
SAWTOOTH,29-Mar-2002 14:02:00.000,47.0,Good
SAWTOOTH,29-Mar-2002 14:03:00.000,2.4,Good
SAWTOOTH,29-Mar-2002 14:04:00.000,9.5,Good
SAWTOOTH,29-Mar-2002 14:08:00.000,12.5,Good
SAWTOOTH,29-Mar-2002 14:14:00.000,7.0,Good
SAWTOOTH,29-Mar-2002 14:22:00.000,4.8,Good
```

The following query retrieves the Maximum value:

```
select timestamp, value, quality from ihrawdata where
samplingmode=calculated and calculationmode=Maximum and timestamp >= '29-Mar-2002 13:50'
and timestamp <= '29-Mar-2002 14:30' and tagname  = SAWTOOTH and numberofsamples = 8
```

The following query retrieves the MaximumTime value:

```
select timestamp, value, quality from ihrawdata where
samplingmode=calculated and calculationmode=MaximumTime and timestamp >= '29-Mar-2002 13:50'
and timestamp <= '29-Mar-2002 14:30' and tagname  = SAWTOOTH and numberofsamples = 8
```

The following query retrieves the Minimum value:

```
select timestamp, value, quality from ihrawdata where
samplingmode=calculated and calculationmode=Minimum and timestamp >= '29-Mar-2002 13:50'
and timestamp <= '29-Mar-2002 14:30' and tagname  = SAWTOOTH and numberofsamples = 8
```

The following query retrieves the MinimumTime value:

```
select timestamp, value, quality from ihrawdata where
samplingmode=calculated and calculationmode=MinimumTime and timestamp >= '29-Mar-2002 13:50'
and timestamp <= '29-Mar-2002 14:30' and tagname  = SAWTOOTH and numberofsamples = 8
```

| Interval Time Stamp | Maximum | Maximum Time | Minimum | Minimum Time | Quality |
|---|---|---|---|---|---|
| 13:55:00.000 | 0.00 | 31-Dec-1969 19:00:00.000 | 0.00 | 31-Dec-1969 19:00:00.000 | 0.00 |
| 14:00:00.000 | 22.70 | 13:59:00.000 | 17.60 | 14:00:00.000 | 100.00 |
| 14:05:00.000 | 47.00 | 14:02:00.000 | 2.40 | 14:03:00.000 | 100.00 |
| 14:10:00.000 | 12.50 | 14:08:00.000 | 10.25 | 14:03:00.000 | 100.00 |
| 14:15:00.000 | 10.67 | 14:10:00.000 | 6.73 | 14:15:00.000 | 100.00 |
| 14:20:00.000 | 6.73 | 14:15:00.000 | 5.35 | 14:20:00.000 | 100.00 |
| 14:25:00.000 | 4.80 | 14:20:00.000 | 4.80 | 14:25:00.000 | 100.00 |
| 14:30:00.000 | 4.80 | 14:30:00.000 | 4.80 | 14:30:00.000 | 100.00 |

Querying with a single interval so that all samples are included results in the following:

| Interval Time Stamp | Maximum | Maximum Time | Minimum | Minimum Time | Quality |
|---|---|---|---|---|---|
| 14:30:00.000 | 47.00 | 14:02:00.000 | 2.40 | 14:03:00.000 | 100.00 |

## Finding the Minimum and Maximum with Bad Quality Data and Repeated Values

Import the following data:

```
[Tags]
Tagname,DataType,HiEngineeringUnits,LoEngineeringUnits
MINMAXBAD,SingleFloat,60,0
```

```
[Data]
Tagname,TimeStamp,Value,DataQuality
MINMAXBAD,29-Mar-2002 13:59:00.000,22.7,Good
MINMAXBAD,29-Mar-2002 14:01:00.000,12.5,Good
MINMAXBAD,29-Mar-2002 14:02:00.000,47.0,Bad
MINMAXBAD,29-Mar-2002 14:03:00.000,2.4,Bad
MINMAXBAD,29-Mar-2002 14:04:00.000,9.5,Good
MINMAXBAD,29-Mar-2002 14:08:00.000,12.5,Good
MINMAXBAD,29-Mar-2002 14:14:00.000,7.0,Good
MINMAXBAD,29-Mar-2002 14:22:00.000,4.8,Good
```

Querying with a single interval so that all samples are included results in the following:

| Interval Time Stamp | Maximum | Maximum Time | Minimum | Minimum Time | Quality |
|---|---|---|---|---|---|
| 14:30:00.000 | 22.70 | 13:59:00.00 | 4.80 | 14:30:00.000 | 100.00 |

**Note:**
MinimumTime is not 14:22 but 14:30. When multiple values within an interval have the same Minimum or Maximum value, (such as here, where two samples have a minimum of 4.8), the newest time stamp is always taken.

### Finding the amount of time the collector was running

The following example uses multiple intervals without a filter condition. If the data is good for the entire interval, the returned Value would be the length of the interval in milliseconds.

Import the following data (identical to that used in the examples for Interpolated Data)

```
[Tags]
Tagname,DataType,HiEngineeringUnits,LoEngineeringUnits
BADDQTAG,SingleFloat,60,0

[Data]
Tagname,TimeStamp,Value,DataQuality
BADDQTAG,29-Mar-2002 13:59:00.000,22.7,Good
BADDQTAG,29-Mar-2002 14:08:00.000,12.5,Bad
```

```
BADDQTAG,29-Mar-2002 14:14:00.000,7.0,Bad

BADDQTAG,29-Mar-2002 14:22:00.000,4.8,Good
```

The following SQL query retrieves data with a start time of 14:00 and an end time of 14:30 in 5-minute intervals:

```
select timestamp,value,intervalmilliseconds from ihRawData where

tagname = baddqtag and samplingmode=calculated and calculationmode=timegood

and timestamp > '29-mar-2002 13:55:00' and timestamp <'29-mar-2002 14:30:00'

and intervalmilliseconds=5m
```

| Time Stamp | Value | Intervalmilliseconds |
|---|---|---|
| 29-Mar-2002 14:00:00.000 | 60,000.00 | 300,000 |
| 29-Mar-2002 14:05:00.000 | 300,000.00 | 300,000 |
| 29-Mar-2002 14:10:00.000 | 180,000.00 | 300,000 |
| 29-Mar-2002 14:15:00.000 | 0.00 | 300,000 |
| 29-Mar-2002 14:20:00.000 | 0.00 | 300,000 |
| 29-Mar-2002 14:25:00.000 | 180,000.00 | 300,000 |

The percent quality for each returned interval is 100.00 and is not shown. By including the intervalmilliseconds column, you can compare the returned milliseconds to the total milliseconds for the interval.

- *When data is good for the whole interval*: From 14:01 to 14:05 the data is good, though no raw samples are contained. The value is equal to intervalmilliseconds (300,000).
- *When data starts bad while entering the interval and then turns good*: The data is bad going into the 14:21 to 14:25 interval, resulting in a value of 180,000 (out of 300,000).
- *When data is bad from the middle of the interval to the end the interval*: The data in the 14:06 to 14:10 interval starts with good quality and changes to bad quality. The value is therefore less than the calculated intervalmilliseconds (180,000 out of 300,000).
- *When there are no raw samples in an interval*: The number of raw samples has no effect on the Value; it only affects the percent quality

The interval from 14:01 to 14:05 contains no raw samples. The data quality throughout the entire interval is good. Therefore, for this interval, the Value is 300,000 (the length of the entire interval).

The interval from 14:16 to 14:20 contains no raw samples. The data quality throughout the entire interval is bad. At no time in this interval is there good data, so for this interval, the Value is 0.

The following example demonstrates the case of bad data throughout a section in the middle of an interval. The following query retrieves data from a larger interval that has a section of bad quality in the middle of 2 periods of good quality.

```
select timestamp,value,intervalmilliseconds from ihRawData where
tagname = baddqtag and samplingmode=calculated and calculationmode=timegood and
timestamp >= '29-mar-2002 14:05:00' and timestamp <= '29-mar-2002 14:25:00' and
intervalmilliseconds=20m
```

| Time Stamp | Value | Intervalmilliseconds |
|---|---|---|
| 29-Mar-2002 14:25:00.000 | 360,000.00 | 1,200,000 |

A value of 360,000 milliseconds corresponds to 3 minutes of good quality at the beginning of the interval and 3 minutes of good quality at the end of the interval.

## Time Weighted Calculation Modes

The ihAverage and ihTotal and ihStandardDeviation modes use time-weighted calculations of interpolated and raw samples. The following example illustrates this concept using the ihAverage mode.

Import the following data:

```
[Tags]
Tagname,DataType,HiEngineeringUnits,LoEngineeringUnits
TAG2,SingleFloat,60,0


[Data]
Tagname,TimeStamp,Value,DataQuality
TAG2,29-Mar-2002 14:00:00.000,30.0,Good
TAG2,29-Mar-2002 14:01:00.000,40.0,Good
TAG2,29-Mar-2002 14:01:10.000,50.0,Good
TAG2,29-Mar-2002 14:01:15.000,20.0,Bad
TAG2,29-Mar-2002 14:01:45.000,25.0,Good
```

Attributes for each data sample are Tagname, TimeStamp, Value, and DataQuality. The data can also be organized by duration:

| Value | Duration |
|---|---|
| 30 | 60 seconds |
| 40 | 10 seconds |
| 50 | 5 seconds |
| 20 | 30 seconds |
| 25 | 15 seconds |

We want to analyze the data over the following interval. The time begins at the timestamp for the first sample and ends 15 seconds after the timestamp of the last sample.

```
3/29/2002 14:00:00 - start time

3/29/2002 14:02:00 - end time
```

Calculating a raw average over these two minutes produces the following:

```
(40 + 50 + 25) / 3 = 38.33
```

You can also calculate it with the following query:

```
select timestamp, value, quality from ihrawdata where

samplingmode=calculated and calculationmode=RawAverage and timestamp >= '29-Mar-2002 14:00'

and timestamp <= '29-Mar-2002 14:02' and tagname  = tag2 and numberofsamples = 1
```

| Time Stamp | Value | Quality |
|---|---|---|
| 29-Mar-200214:02:00.000 | 38.33 | 100.00 |

The value of 30 is not used because the RawAverage mode uses only samples whose timestamps are greater than the interval's start time. A value whose timestamp is 14:00 would be associated with the previous interval.

In a time-weighted average, the values are multiplied by the durations. The two-minute time weighted average is:

```
((30 * 59.999) + (40 * 10) + (50 * 5) + (25*15)) / (60 + 10 + 5 + 15) = 31.38
```

You can calculate this with the following query:

```
select timestamp, value, quality from ihrawdata where
samplingmode=calculated and calculationmode=Average and timestamp >= '29-Mar-2002 14:00'
and timestamp <= '29-Mar-2002 14:02' and tagname  = tag2 and numberofsamples = 1
```

This more closely describes the real situation, the value was 30 during most of the queried interval. The value of 30 is assigned to a timestamp of 14:00:00.001, which is the first possible timestamp greater than the interval start time (up to a resolution of milliseconds).

**Note:**
The bad quality sample (whose value was 20) is ignored when calculating the time-weighted average. The quality was bad for 30 seconds out of the 2 minutes, so the percent good quality is 75. When performing time-weighted calculations, percent good represents the percentage of time within the interval that had data with good quality: (90 seconds of good data quality / 120 seconds of the total interval duration ) = 75%

**Computing an Average Without A Raw Sample At Start Time**

There is rarely a raw sample available at the interval start time. However, the archiver needs to know the value at the start of an interval before it can perform time-weighted calculations. The archiver uses interpolation to get values it needs for which no raw samples are available.

For example, if we set the start time for the query to 14:05, then the archiver will interpolate a value at the timestamp of 14:05.

The RawAverage would then be calculated as follows:

```
select timestamp, value, quality from ihrawdata where samplingmode=calculated and
calculationmode=RawAverage and timestamp >= '29-Mar-2002 14:00:05' and
timestamp <= '29-Mar-2002 14:02' and tagname  = tag2 and numberofsamples = 1
```

| Time Stamp | Value | Quality |
|---|---|---|
| 29-Mar-200214:02:00.000 | 38.33 | 100.00 |

Similarly, the time-weighted average would be calculated as follows:

```
select timestamp, value, quality from ihrawdata where samplingmode=calculated
and calculationmode=Average and timestamp >= '29-Mar-2002 14:00:05' and
timestamp <= '29-Mar-2002 14:02' and tagname  = tag2 and numberofsamples = 1
```

| Time Stamp | Value | Quality |
|---|---|---|
| 29-Mar-2002 14:02:00.000 | 32.01 | 73.91 |

**Average and Step Values**

The average of the raw samples is the interval, but there is special logic for time weighting and for computing the value at the start of the interval.

Averages are computed differently depending on the value of the Tag.StepValue property. If `StepValuee=FALSE` then the average works as it always did in 2.0 and 3.0. A value at the start of the interval is determined via interpolation.

If `StepValue=TRUE` then lab sampling, not interpolation, is used to determine the value at interval start time. This would more accurately reflect a value that steps or a value that uses collector compression and did not change for a long period of time.

**ihTotal Mode**

The ihTotal mode retrieves the time weighted rate total for each calculation interval.

A rate total is considered for totalizing a continuous measurement. A factor is applied to the totalized value to convert into the appropriate engineering units. Since this is a rate total, a base rate of Units/Day is assumed. If the actual units of the continuous measurement is Units/Minute, multiply the results by 1440 Minutes / Day to convert the totalized number into the appropriate engineering units.

The formula for total is `total = average &` (interval in milliseconds / 1000) / 86400. The 86400 is number of seconds in a day. This formula takes the average, which is assumed to be already in units per day, and divides it into "units per interval".

## Collecting a Rate from a Data Source

Assume an average of 240 barrels per day.

If your interval is one day, then the "units per interval" is units per DAY. Since the average was already assumed to be in units per day, you just get back the average.

240 = 240 * (86400000/1000) / 86400

240 = 240 * 1

If your interval is 1 hour, you should get back 1/24 of the average.

total= 240 * (3600000/1000) / 86400

total = 240 * 0.0417

total = 10

Ten is 1/24 of 240 and tells you 10 units were produced that hour.

## Filtered Data Queries

You can retrieve data using an optional filter tag or filter expression if the client program or API you are using supports it.

Normally, a data query specifies a start and an end time for the query. Data is returned for `ALL` intervals between the start and end times. A filtered data query allows you to specify a filter tag or expression with additional criteria so that only some of those intervals which match the filter conditions are returned. The method of calculating the value attributed to the interval can be different from a non-filtered query, since the filter criteria can exclude raw samples inside an interval as well as exclude intervals themselves.

The value that triggers a transition from `FALSE` to `TRUE` can be a raw value or interpolated value. If a FilterTag or expression is supplied, the Data Archiver attempts to filter time periods from the results.

The filter data query parameters include:

- FilterTag or FilterExpression
- FilterMode
- FilterComparisonMode
- FilterComparisonValue

Each parameter is described in the following table with examples that demonstrate common usages.

Internally to the Data Archiver, the filter condition is evaluated to get zero or more time ranges. For example, if you query from 1pm to 2pm and the filter condition was never TRUE during that time, nothing is returned.

If the condition was TRUE from 1:40 to 1:45 then only the data for that time range is queried and returned. Together the Filter Tag, Filter value, and Filter Comparison Mode define the criteria to apply to each interval to determine inclusion or exclusion. You can optionally use Filter Expression to include all the above parameters in one condition.

The Include Times defines how the time periods before and after transitions in the filter condition should be handled. An example with actual data and a graphic to clarify the behavior of each of the IncludeTime options is provided in the following topics.

You can retrieve data using a filtered data query or filter expressions.

**Using a Filtered Data Query**

The filter query logic has two problems to solve:

- *Which time ranges should be included?* In the following example, you see that No Filter mode returns all intervals. Each mode has its own logic to determine if an interval passes the filter or not.
- *What value and quality should be attributed to the interval?* If the filter condition is TRUE for the whole interval, then this is just like the non-filtered result. When the filter condition is TRUE only for part of the inter- val, raw samples get filtered out, changing the values returned.

For the following example, we know that we want to use AfterTime since the batch ID is written to the PLC at the start of the batch. The intervals included are the ones for the times the batch is running. You would use the BeforeTime if the batch ID was written at the end. Use the ExactTime if you are comparing 2 or more values at a single point in time.

```
* Example for Filtered Data Documentation
*
[Tags]
Tagname,DataType,HiEngineeringUnits,LoEngineeringUnits
BATCHID,VariableString,10,0
RAMP,SingleInteger,60,0
ONOFF,SingleInteger,60,0
HAS SPACE,SingleInteger,60,0
[Data]
Tagname,TimeStamp,Value,DataQuality
BATCHID,30-Jul-2002 07:00:00.000,B1,Good
BATCHID,30-Jul-2002 07:00:20.000,B2,Good
BATCHID,30-Jul-2002 07:00:34.000,B3,Good
BATCHID,30-Jul-2002 07:00:52.000,B1,Good
RAMP,30-Jul-2002 07:00:00.000,0,Good
RAMP,30-Jul-2002 07:00:01.000,1,Good
RAMP,30-Jul-2002 07:00:02.000,2,Good
RAMP,30-Jul-2002 07:00:03.000,3,Good
RAMP,30-Jul-2002 07:00:04.000,4,Good
RAMP,30-Jul-2002 07:00:05.000,5,Good
RAMP,30-Jul-2002 07:00:06.000,6,Good
RAMP,30-Jul-2002 07:00:07.000,7,Good
```

```
RAMP,30-Jul-2002 07:00:08.000,8,Good
RAMP,30-Jul-2002 07:00:09.000,9,Good
RAMP,30-Jul-2002 07:00:10.000,10,Good
RAMP,30-Jul-2002 07:00:11.000,11,Good
RAMP,30-Jul-2002 07:00:12.000,12,Good
RAMP,30-Jul-2002 07:00:13.000,13,Good
RAMP,30-Jul-2002 07:00:14.000,14,Good
RAMP,30-Jul-2002 07:00:15.000,15,Good
RAMP,30-Jul-2002 07:00:16.000,16,Good
RAMP,30-Jul-2002 07:00:17.000,17,Good
RAMP,30-Jul-2002 07:00:18.000,18,Good
RAMP,30-Jul-2002 07:00:19.000,19,Good
RAMP,30-Jul-2002 07:00:20.000,20,Good
RAMP,30-Jul-2002 07:00:21.000,21,Good
RAMP,30-Jul-2002 07:00:22.000,22,Good
RAMP,30-Jul-2002 07:00:23.000,23,Good
RAMP,30-Jul-2002 07:00:24.000,24,Good
RAMP,30-Jul-2002 07:00:25.000,25,Good
RAMP,30-Jul-2002 07:00:26.000,26,Good
RAMP,30-Jul-2002 07:00:27.000,27,Good
RAMP,30-Jul-2002 07:00:28.000,28,Good
RAMP,30-Jul-2002 07:00:29.000,29,Good
RAMP,30-Jul-2002 07:00:30.000,30,Good
RAMP,30-Jul-2002 07:00:31.000,31,Good
RAMP,30-Jul-2002 07:00:32.000,32,Good
RAMP,30-Jul-2002 07:00:33.000,33,Good
RAMP,30-Jul-2002 07:00:34.000,34,Good
RAMP,30-Jul-2002 07:00:35.000,35,Good
RAMP,30-Jul-2002 07:00:36.000,36,Good
RAMP,30-Jul-2002 07:00:37.000,37,Good
RAMP,30-Jul-2002 07:00:38.000,38,Good
RAMP,30-Jul-2002 07:00:39.000,39,Good
RAMP,30-Jul-2002 07:00:40.000,40,Good
RAMP,30-Jul-2002 07:00:41.000,41,Good
RAMP,30-Jul-2002 07:00:42.000,42,Good
RAMP,30-Jul-2002 07:00:43.000,43,Good
RAMP,30-Jul-2002 07:00:44.000,44,Good
```

```
RAMP,30-Jul-2002 07:00:45.000,45,Good
RAMP,30-Jul-2002 07:00:46.000,46,Good
RAMP,30-Jul-2002 07:00:47.000,47,Good
RAMP,30-Jul-2002 07:00:48.000,48,Good
RAMP,30-Jul-2002 07:00:49.000,49,Good
RAMP,30-Jul-2002 07:00:50.000,50,Good
RAMP,30-Jul-2002 07:00:51.000,51,Good
RAMP,30-Jul-2002 07:00:52.000,52,Good
RAMP,30-Jul-2002 07:00:53.000,53,Good
RAMP,30-Jul-2002 07:00:54.000,54,Good
RAMP,30-Jul-2002 07:00:55.000,55,Good
RAMP,30-Jul-2002 07:00:56.000,56,Good
RAMP,30-Jul-2002 07:00:57.000,57,Good
RAMP,30-Jul-2002 07:00:58.000,58,Good
RAMP,30-Jul-2002 07:00:59.000,59,Good
ONOFF,30-Jul-2002 07:00:00.000,0,Good
ONOFF,30-Jul-2002 07:00:01.000,1,Good
ONOFF,30-Jul-2002 07:01:01.000,0,Good
ONOFF,30-Jul-2002 07:01:16.000,0,Good
ONOFF,30-Jul-2002 07:01:17.000,1,Good
ONOFF,30-Jul-2002 07:01:18.000,1,Good
ONOFF,30-Jul-2002 07:02:01.000,1,Good
ONOFF,30-Jul-2002 07:03:01.000,0,Good
HAS SPACE,30-Jul-2002 07:00:00.000,0,Good
HAS SPACE,30-Jul-2002 07:00:01.000,1,Good
HAS SPACE,30-Jul-2002 07:01:01.000,0,Good
HAS SPACE,30-Jul-2002 07:01:16.000,0,Good
HAS SPACE,30-Jul-2002 07:01:17.000,1,Good
HAS SPACE,30-Jul-2002 07:01:18.000,1,Good
HAS SPACE,30-Jul-2002 07:02:01.000,1,Good
HAS SPACE,30-Jul-2002 07:03:01.000,0,Good
```

The key to understanding this example is that the batch ID is written to the text tag at the start of the batch, and only at the start. It is not repeated during the batch. Other systems may write the batch ID at the end of the interval. You can tell the time period of a batch by looking at the raw samples of the batch ID tag.

```
BATCHID,30-Jul-2002 07:00:00.000,B1,Good

BATCHID,30-Jul-2002 07:00:20.000,B2,Good

BATCHID,30-Jul-2002 07:00:34.000,B3,Good

BATCHID,30-Jul-2002 07:00:52.000,B1,Good
```

In this system, since the batch ID is written at the start of the batch, you can tell that batch B1 ran from 07:00:01 to 07:00:19 and then batch B2 ran. This assumes that all time is attributable to some batch and there is no dead time between batches. If there is equipment downtime after a batch, you need to write some other value to the batch ID tag to indicate the end time of the batch.

The query parameters are given below:

| Query Parameter | Value |
|---|---|
| Start Time | 07/30/2002 07:00:00 |
| End Time | 07/30/2002 07:01:00 |
| Interval | 10 seconds |

**Using Filter Expressions**

You can enter filter expressions in filtered data queries to indicate the desired time range. A Filter Expression has one or more filter conditions.

A filter condition has:

1. A Historian tag
2. A comparison (=, !=, >, <, <=, >=, ^, ~, !~, !^)
3. A value

For example: mytag < 7

You can add more than one filter condition in a filter expression using AND, OR within a parenthesis. For example: (mytag > 3) and (mytag < 7).

You can use bitwise comparison for a tag. By using bitwise comparison you can compare the binary values of the given filter tag with the bits specified in the condition. The Bitwise comparison modes are:

1. AllBitssSet (^)
2. AnyBitSet (~)
3. AnyBitNotSet (!~)
4. AllBitsNotSet (!^)

While using filter expression you should remember the following things:

- You cannot use a NOT operator; you can use != instead.
- You cannot do mathematical operations such as (mytag1+7) > 15.
- You cannot compare two tags such as mytag1 > mytag2.

Your conditions can only include values and not qualities. Values are used only if they are of good quality so you need not check the quality separately. As with any filtered data query, the filter expression determines the time ranges of the data returned. There is no maximum length for an expression but a typical expression will be have 1 or 3 conditions.

Import this data to Historian:

```
*
[Tags]
Tagname,DataType,HiEngineeringUnits,LoEngineeringUnits
BATCHID,VariableString,10,0
RAMP,SingleInteger,60,0
ONOFF,SingleInteger,60,0
HAS SPACE,SingleInteger,60,0
[Data]
Tagname,TimeStamp,Value,DataQuality
BATCHID,30-Jul-2002 07:00:00.000,B1,Good
BATCHID,30-Jul-2002 07:00:20.000,B2,Good
BATCHID,30-Jul-2002 07:00:34.000,B3,Good
BATCHID,30-Jul-2002 07:00:52.000,B1,Good
RAMP,30-Jul-2002 07:00:00.000,0,Good
RAMP,30-Jul-2002 07:00:01.000,1,Good
RAMP,30-Jul-2002 07:00:02.000,2,Good
RAMP,30-Jul-2002 07:00:03.000,3,Good
RAMP,30-Jul-2002 07:00:04.000,4,Good
RAMP,30-Jul-2002 07:00:05.000,5,Good
RAMP,30-Jul-2002 07:00:06.000,6,Good
RAMP,30-Jul-2002 07:00:07.000,7,Good
RAMP,30-Jul-2002 07:00:08.000,8,Good
RAMP,30-Jul-2002 07:00:09.000,9,Good
RAMP,30-Jul-2002 07:00:10.000,10,Good
RAMP,30-Jul-2002 07:00:11.000,11,Good
RAMP,30-Jul-2002 07:00:12.000,12,Good
```

```
RAMP,30-Jul-2002 07:00:13.000,13,Good
RAMP,30-Jul-2002 07:00:14.000,14,Good
RAMP,30-Jul-2002 07:00:15.000,15,Good
RAMP,30-Jul-2002 07:00:16.000,16,Good
RAMP,30-Jul-2002 07:00:17.000,17,Good
RAMP,30-Jul-2002 07:00:18.000,18,Good
RAMP,30-Jul-2002 07:00:19.000,19,Good
RAMP,30-Jul-2002 07:00:20.000,20,Good
RAMP,30-Jul-2002 07:00:21.000,21,Good
RAMP,30-Jul-2002 07:00:22.000,22,Good
RAMP,30-Jul-2002 07:00:23.000,23,Good
RAMP,30-Jul-2002 07:00:24.000,24,Good
RAMP,30-Jul-2002 07:00:25.000,25,Good
RAMP,30-Jul-2002 07:00:26.000,26,Good
RAMP,30-Jul-2002 07:00:27.000,27,Good
RAMP,30-Jul-2002 07:00:28.000,28,Good
RAMP,30-Jul-2002 07:00:29.000,29,Good
RAMP,30-Jul-2002 07:00:30.000,30,Good
RAMP,30-Jul-2002 07:00:31.000,31,Good
RAMP,30-Jul-2002 07:00:32.000,32,Good
RAMP,30-Jul-2002 07:00:33.000,33,Good
RAMP,30-Jul-2002 07:00:34.000,34,Good
RAMP,30-Jul-2002 07:00:35.000,35,Good
RAMP,30-Jul-2002 07:00:36.000,36,Good
RAMP,30-Jul-2002 07:00:37.000,37,Good
RAMP,30-Jul-2002 07:00:38.000,38,Good
RAMP,30-Jul-2002 07:00:39.000,39,Good
RAMP,30-Jul-2002 07:00:40.000,40,Good
RAMP,30-Jul-2002 07:00:41.000,41,Good
RAMP,30-Jul-2002 07:00:42.000,42,Good
RAMP,30-Jul-2002 07:00:43.000,43,Good
RAMP,30-Jul-2002 07:00:44.000,44,Good
RAMP,30-Jul-2002 07:00:45.000,45,Good
RAMP,30-Jul-2002 07:00:46.000,46,Good
RAMP,30-Jul-2002 07:00:47.000,47,Good
RAMP,30-Jul-2002 07:00:48.000,48,Good
RAMP,30-Jul-2002 07:00:49.000,49,Good
```

```
RAMP,30-Jul-2002 07:00:50.000,50,Good
RAMP,30-Jul-2002 07:00:51.000,51,Good
RAMP,30-Jul-2002 07:00:52.000,52,Good
RAMP,30-Jul-2002 07:00:53.000,53,Good
RAMP,30-Jul-2002 07:00:54.000,54,Good
RAMP,30-Jul-2002 07:00:55.000,55,Good
RAMP,30-Jul-2002 07:00:56.000,56,Good
RAMP,30-Jul-2002 07:00:57.000,57,Good
RAMP,30-Jul-2002 07:00:58.000,58,Good
RAMP,30-Jul-2002 07:00:59.000,59,Good
ONOFF,30-Jul-2002 07:00:00.000,0,Good
ONOFF,30-Jul-2002 07:00:01.000,1,Good
ONOFF,30-Jul-2002 07:01:01.000,0,Good
ONOFF,30-Jul-2002 07:01:16.000,0,Good
ONOFF,30-Jul-2002 07:01:17.000,1,Good
ONOFF,30-Jul-2002 07:01:18.000,1,Good
ONOFF,30-Jul-2002 07:02:01.000,1,Good
ONOFF,30-Jul-2002 07:03:01.000,0,Good
HAS SPACE,30-Jul-2002 07:00:00.000,0,Good
HAS SPACE,30-Jul-2002 07:00:01.000,1,Good
HAS SPACE,30-Jul-2002 07:01:01.000,0,Good
HAS SPACE,30-Jul-2002 07:01:16.000,0,Good
HAS SPACE,30-Jul-2002 07:01:17.000,1,Good
HAS SPACE,30-Jul-2002 07:01:18.000,1,Good
HAS SPACE,30-Jul-2002 07:02:01.000,1,Good
HAS SPACE,30-Jul-2002 07:03:01.000,0,Good
```

For the following scenarios, import the data tags provided.

## Counter Delta Queries

A delta counter measures the change in tag values that increase steadily over a time interval and then reset to a minimum value (for example, the electricity meter of a household).

Historian offers the following delta queries to determine the delta over a time interval:

- DELTAPOS *(on page 1116)*
- DELTANEG *(on page 1132)*
- DELTA *(on page 1141)*

**Advantages of using delta queries:**

- Simplified visualization and analysis of counter data.
- Returns the delta over a period rather than the exact value at the end of each counter.
- Handles counter resets and counter wrap around.

Delta counters are useful when you are monitoring data from multiple sources. The following terms are used in a delta counter:

**Counter Reset**

A point where data points are reset to the minimum value. For example, the electricity meter of a household is reset to 0 at the beginning of every month or when a meter is replaced with a new one.

**Counter Wrap Around**

A point where data in an increasing trend suddenly drops to a value less than a specified value. This happens when the reading goes beyond the maximum value that the meter can read. For example, if a meter can read from 0 to 255, any reading greater than 255 is set to 0. This is called a counter wrap around.

**Delta Max Value**

The maximum value that a tag can have. It also called the rollover point of the counter or totalizer. If the tag values exceed MaxValue, the counter is reset to the minimum value. If you do not provide MaxValue, the delta query cannot check for a positive counter wrap. You can set this value while specifying tags for data collection *(on page 357)* in Configuration Hub.

**Delta Min Value**

The minimum value that a tag can have. If the tag values are less than MinValue (and the counter is going in the negative direction), the tag values are reset to MaxValue. If you do not provide MinValue, 0 is considered. You can set this value while specifying tags for data collection *(on page 357)* in Configuration Hub.

**Delta Max Positive RPH**

The maximum rate per hour between two consecutive data points in the positive direction. If two consecutive data points exceed this value, they are not considered in a delta query. You can set this value while specifying tags for data collection *(on page 357)* in Configuration Hub.

**Delta Max Negative RPH**

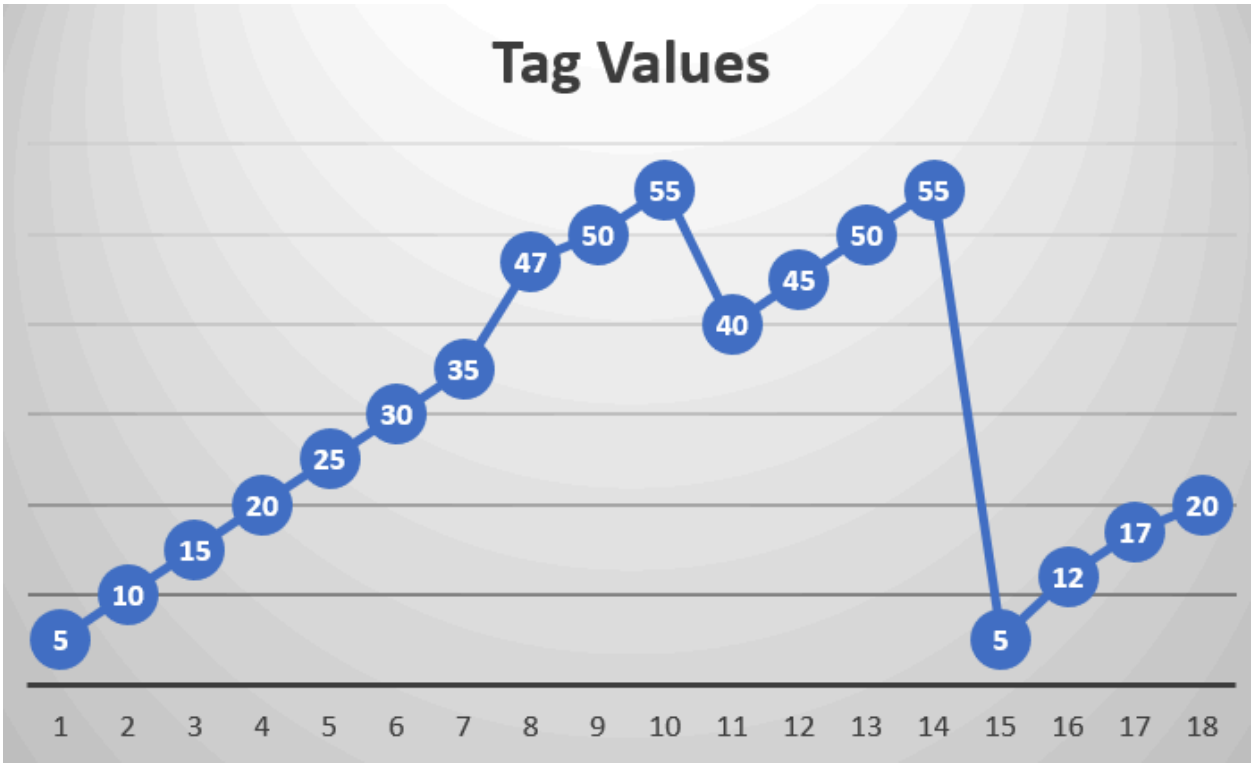
The maximum rate per hour between two consecutive data points in the negative direction. If two consecutive data points exceed this value, they are not considered in a delta query. You can set this value while specifying tags for data collection *(on page 357)* in Configuration Hub.

The Delta Max Positive RPH and Delta Max Negative RPH values are used to determine if a counter wrap has occurred or if the counter has been manually reset. They help ignore data points whose values increase or decrease drastically.

Suppose a tag stores the readings of an electricity meter. And you have provided the following values:

| Field | Value |
|---|---|
| **Delta Max Value** | 255 |
| **Delta Min Value** | 5 |
| **Delta Max Positive RPH** | 10 |
| **Delta Max Negative RPH** | 10 |

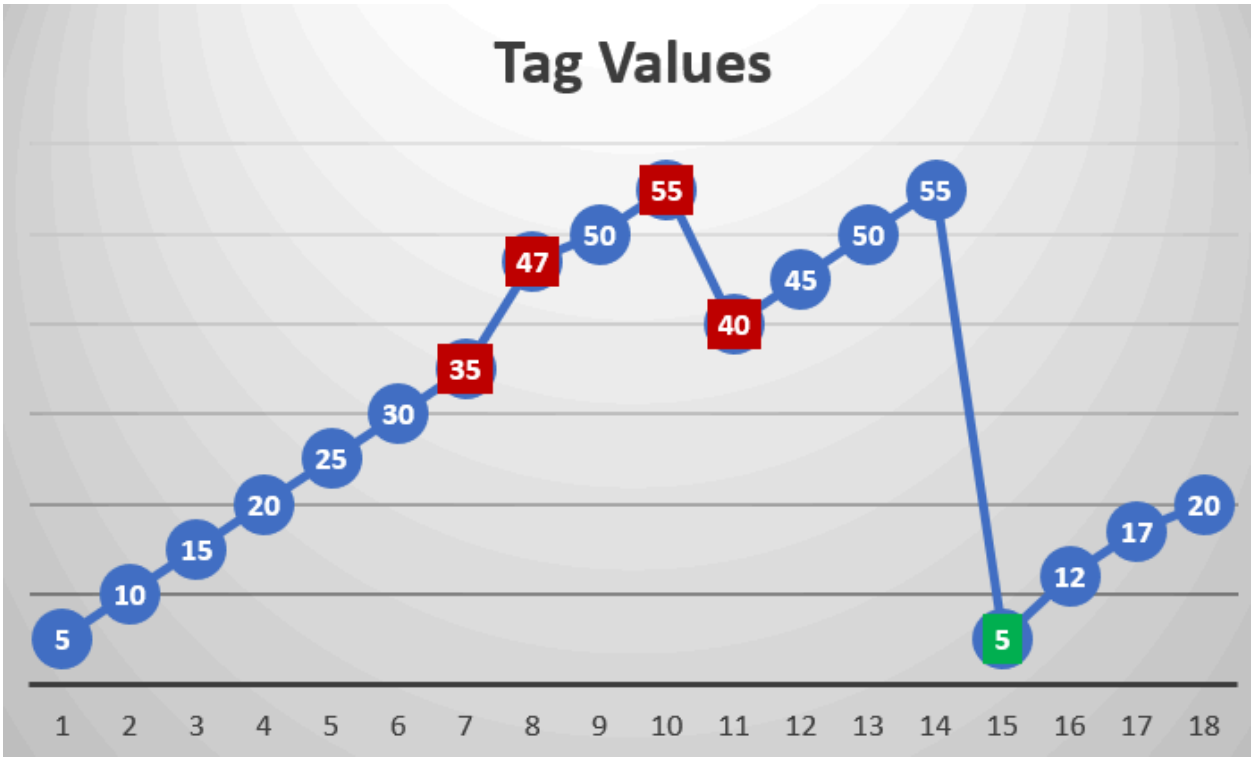
Suppose the following data points are received for the tag:




After 35, the value increases to 47. Since the difference is greater than MaxPositiveRPH, the data points 35 and 47 will not considered in a delta query.

Similarly, the difference between 55 and 40 is greater than MaxNegativeRPH; therefore, these two data points will not considered in a delta query.

After 55, the counter has been reset to its minimum value.




## DELTAPOS Calculation

The following diagrams show how DELTAPOS is calculated. If Delta Min is not considered, 0 is considered.

Figure 2. DELTAPOS Calculation When Delta Max Positive RPH is not Provided

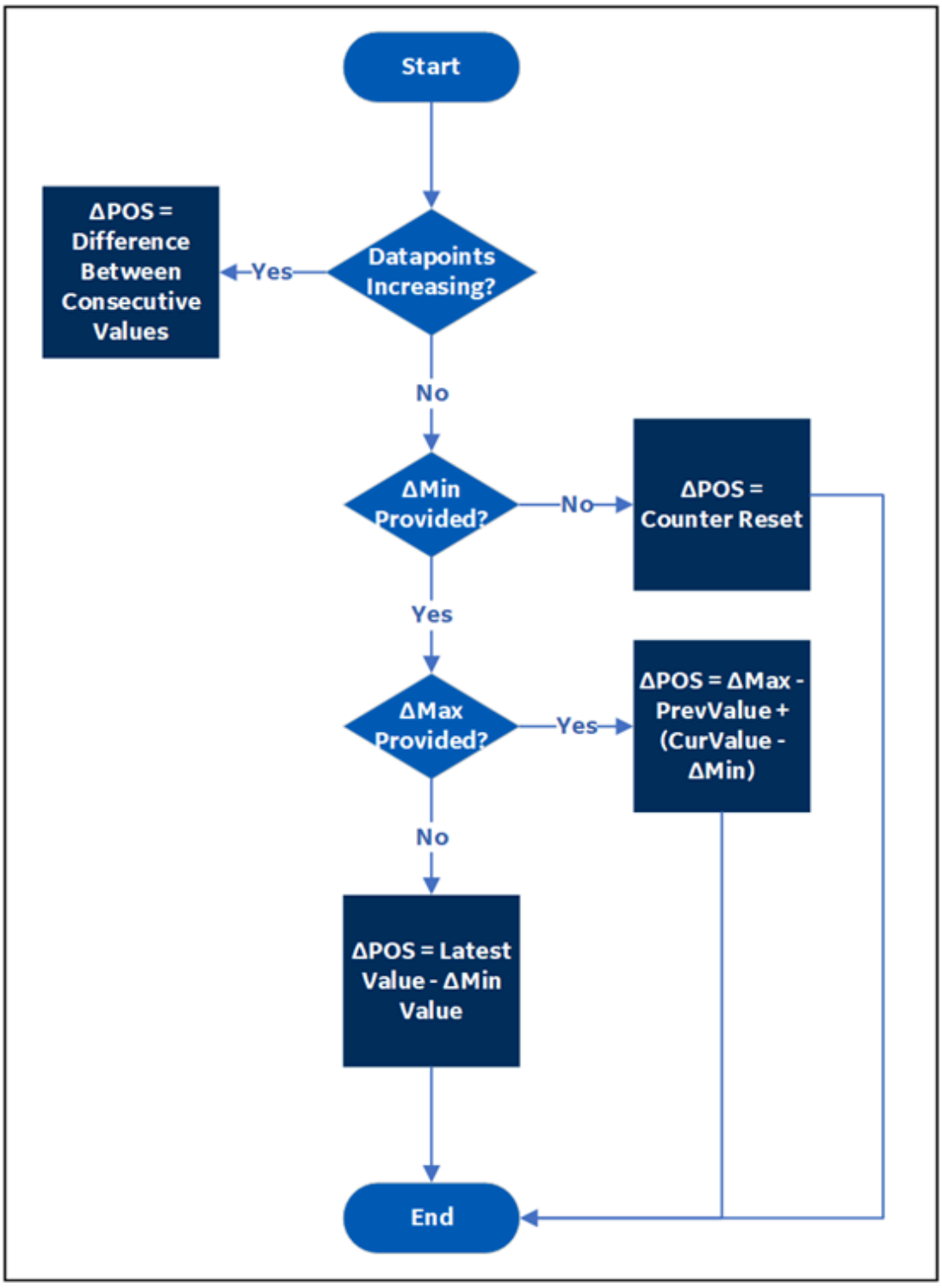
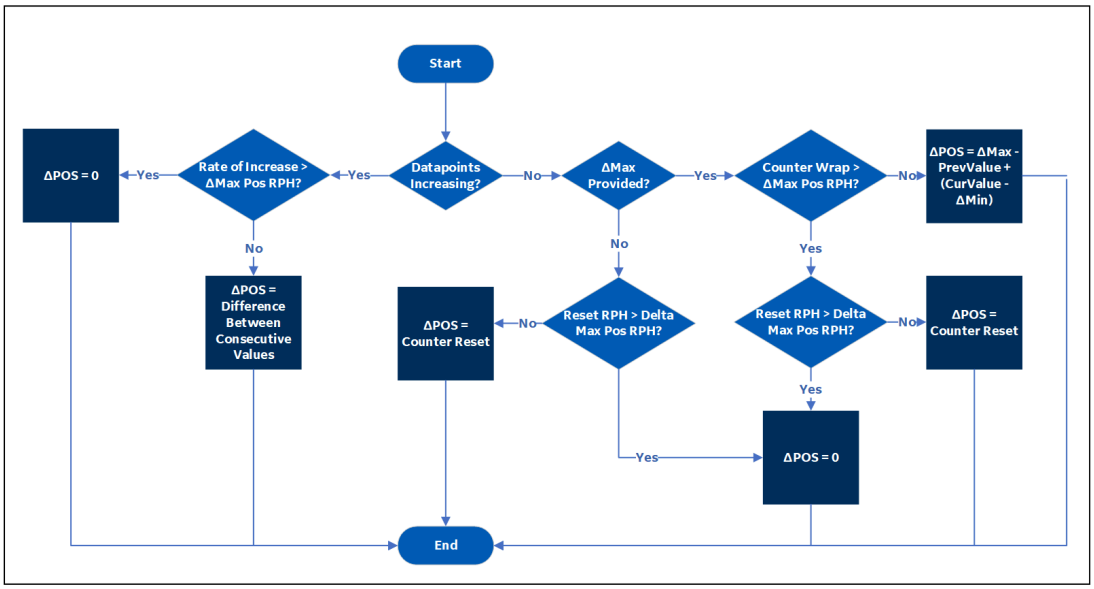


Figure 3. DELTAPOS Calculation When Delta Max Positive RPH is Provided




**Note:**

Delta Max Negative RPH is not used to calculate DELTAPOS.

## Calculating DELTAPOS When Delta Max Positive RPH is Not Provided and Data is in the Increasing Trend

Suppose you have received the following tag data:

| TimeStamp | Tag Value | Data Quality |
|---|---|---|
| 19-Dec-2021 14:01:10 | 0 | Good |
| 19-Dec-2021 14:01:20 | 5 | Good |
| 19-Dec-2021 14:01:30 | 10 | Good |
| 19-Dec-2021 14:01:40 | 15 | Good |
| 19-Dec-2021 14:01:50 | 20 | Good |
| 19-Dec-2021 14:02:00 | 25 | Good |
| 19-Dec-2021 14:02:10 | 30 | Good |
| 19-Dec-2021 14:02:20 | 35 | Good |
| 19-Dec-2021 14:02:30 | 40 | Good |

| TimeStamp | Tag Value | Data Quality |
|---|---|---|
| 19-Dec-2021 14:02:40 | 45 | Good |
| 19-Dec-2021 14:02:50 | 50 | Good |
| 19-Dec-2021 14:03:00 | 55 | Good |
| 19-Dec-2021 14:03:10 | 60 | Good |
| 19-Dec-2021 14:03:20 | 65 | Good |
| 19-Dec-2021 14:03:30 | 70 | Good |
| 19-Dec-2021 14:03:40 | 75 | Good |
| 19-Dec-2021 14:03:50 | 80 | Good |
| 19-Dec-2021 14:04:00 | 85 | Good |
| 19-Dec-2021 14:04:10 | 90 | Good |
| 19-Dec-2021 14:04:20 | 95 | Good |
| 19-Dec-2021 14:04:30 | 100 | Good |
| 19-Dec-2021 14:04:40 | 105 | Good |
| 19-Dec-2021 14:04:50 | 110 | Good |
| 19-Dec-2021 14:05:00 | 115 | Good |
| 19-Dec-2021 14:05:10 | 120 | Good |
| 19-Dec-2021 14:05:20 | 125 | Good |
| 19-Dec-2021 14:05:30 | 130 | Good |
| 19-Dec-2021 14:05:40 | 135 | Good |
| 19-Dec-2021 14:05:50 | 140 | Good |
| 19-Dec-2021 14:06:00 | 145 | Good |

Suppose you have run the following query:

```
Select timestamp,value,quality from IHrawdata

where tagname=BasicTestPos and samplingmode=calculated and

timestamp>='19-Dec-2021 14:00:00.000' and timestamp<='19-Dec-2021 14:06:00.000'

and calculationmode=DeltaPos and intervalmilliseconds=1M
```

Regardless of whether you have provided Delta Min and Delta Max, since you have not provided Delta Max Positive RPH, and since the data points are in the increasing trend, DELTAPOS is calculated as the difference between consecutive data points:

| TimeStamp | DELTAPOS Value | Data Quality |
|---|---|---|
| 19-Dec-2021 14:00:00 | 0 | 100 |
| 19-Dec-2021 14:01:00 | 0 | 100 |
| 19-Dec-2021 14:02:00 | 25.00 | 100 |
| 19-Dec-2021 14:03:00 | 30.00 | 100 |
| 19-Dec-2021 14:04:00 | 30.00 | 100 |
| 19-Dec-2021 14:05:00 | 30.00 | 100 |
| 19-Dec-2021 14:06:00 | 30.00 | 100 |

## Calculating DELTAPOS When Data Quality is Bad

Suppose you have received the following data:

| Time Stamp | Value | Data Quality |
|---|---|---|
| 19-Dec-2021 14:01:10 | 0 | Good |
| 19-Dec-2021 14:01:20 | 5 | Bad |
| 19-Dec-2021 14:01:30 | 10 | Good |
| 19-Dec-2021 14:01:40 | 15 | Bad |
| 19-Dec-2021 14:01:50 | 20 | Good |
| 19-Dec-2021 14:02:00 | 25 | Bad |
| 19-Dec-2021 14:02:10 | 30 | Good |
| 19-Dec-2021 14:02:20 | 35 | Bad |
| 19-Dec-2021 14:02:30 | 40 | Good |
| 19-Dec-2021 14:02:40 | 45 | Bad |
| 19-Dec-2021 14:02:50 | 50 | Good |
| 19-Dec-2021 14:03:00 | 55 | Bad |

| Time Stamp | Val-ue | Data Quality |
|---|---|---|
| 19-Dec-2021 14:03:10 | 60 | Good |
| 19-Dec-2021 14:03:20 | 65 | Bad |
| 19-Dec-2021 14:03:30 | 70 | Good |
| 19-Dec-2021 14:03:40 | 75 | Bad |
| 19-Dec-2021 14:03:50 | 80 | Good |
| 19-Dec-2021 14:04:00 | 85 | Bad |
| 19-Dec-2021 14:04:10 | 90 | Good |
| 19-Dec-2021 14:04:20 | 95 | Bad |
| 19-Dec-2021 14:04:30 | 100 | Good |
| 19-Dec-2021 14:04:40 | 105 | Bad |
| 19-Dec-2021 14:04:50 | 110 | Good |
| 19-Dec-2021 14:05:00 | 115 | Bad |
| 19-Dec-2021 14:05:10 | 120 | Good |
| 19-Dec-2021 14:05:20 | 125 | Bad |
| 19-Dec-2021 14:05:30 | 130 | Good |
| 19-Dec-2021 14:05:40 | 135 | Bad |
| 19-Dec-2021 14:05:50 | 140 | Good |
| 19-Dec-2021 14:06:00 | 145 | Bad |

Suppose you have run the following query:

```
Select timestamp,value,quality from IHrawdata

where tagname=BasicTestWithBad and samplingmode=calculated

and timestamp>='19-Dec-2021 14:00:00.000' and timestamp<='19-Dec-2021 14:06:00.000'

and calculationmode=DeltaPos and intervalmilliseconds=1M
```

Data with bad quality is not considered in the calculation. Therefore, DELTAPOS is the difference between consecutive data points with good quality.

| Time Stamp | Val-ue | Data Quality | DELTAPOS |
|---|---|---|---|
| 19-Dec-2021 14:01:10 | 0 | Good | 0 |
| 19-Dec-2021 14:01:20 | 5 | Bad | 0 |
| 19-Dec-2021 14:01:30 | 10 | Good | 10 |
| 19-Dec-2021 14:01:40 | 15 | Bad | 0 |
| 19-Dec-2021 14:01:50 | 20 | Good | 10 |
| 19-Dec-2021 14:02:00 | 25 | Bad | 0 |
| 19-Dec-2021 14:02:10 | 30 | Good | 10 |
| 19-Dec-2021 14:02:20 | 35 | Bad | 0 |
| 19-Dec-2021 14:02:30 | 40 | Good | 10 |
| 19-Dec-2021 14:02:40 | 45 | Bad | 0 |
| 19-Dec-2021 14:02:50 | 50 | Good | 10 |
| 19-Dec-2021 14:03:00 | 55 | Bad | 0 |
| 19-Dec-2021 14:03:10 | 60 | Good | 10 |
| 19-Dec-2021 14:03:20 | 65 | Bad | 0 |
| 19-Dec-2021 14:03:30 | 70 | Good | 10 |
| 19-Dec-2021 14:03:40 | 75 | Bad | 0 |
| 19-Dec-2021 14:03:50 | 80 | Good | 10 |
| 19-Dec-2021 14:04:00 | 85 | Bad | 0 |
| 19-Dec-2021 14:04:10 | 90 | Good | 10 |
| 19-Dec-2021 14:04:20 | 95 | Bad | 0 |
| 19-Dec-2021 14:04:30 | 100 | Good | 10 |
| 19-Dec-2021 14:04:40 | 105 | Bad | 0 |
| 19-Dec-2021 14:04:50 | 110 | Good | 10 |
| 19-Dec-2021 14:05:00 | 115 | Bad | 0 |
| 19-Dec-2021 14:05:10 | 120 | Good | 10 |
| 19-Dec-2021 14:05:20 | 125 | Bad | 0 |

| Time Stamp | Val-ue | Data Quality | DELTAPOS |
|---|---|---|---|
| 19-Dec-2021 14:05:30 | 130 | Good | 10 |
| 19-Dec-2021 14:05:40 | 135 | Bad | 0 |
| 19-Dec-2021 14:05:50 | 140 | Good | 10 |
| 19-Dec-2021 14:06:00 | 145 | Bad | 0 |

DELTAPOS for a given duration is calculated as the sum of the individual DELTAPOS values in that duration as shown in the following table.

| Time Stamp | Val-ue | Data Quality |
|---|---|---|
| 19-Dec-2021 14:00:00 | 0.00 | 100 |
| 19-Dec-2021 14:01:00 | 0.00 | 100 |
| 19-Dec-2021 14:02:00 | 20.00 | 100 |
| 19-Dec-2021 14:03:00 | 30.00 | 100 |
| 19-Dec-2021 14:04:00 | 30.00 | 100 |
| 19-Dec-2021 14:05:00 | 30.00 | 100 |
| 19-Dec-2021 14:06:00 | 30.00 | 100 |

### Calculating DELTAPOS When Rate of Increase is Greater than Delta Max Positive RPH

Suppose you have provided the following data:

| | |
|---|---|
| Delta Max | 265 |
| Delta Min | 50 |
| Delta Max Positive RPH | 10 |
| Delta Max Negative RPH | 10 |

Suppose you have received the following data:

| TimeStamp | Tag Value | Data Quality |
|---|---|---|
| 19-Dec-2021 14:01:10 | 55 | Good |
| 19-Dec-2021 14:01:20 | 65 | Good |
| 19-Dec-2021 14:01:30 | 80 | Good |
| 19-Dec-2021 14:01:40 | 85 | Good |
| 19-Dec-2021 14:01:50 | 95 | Good |
| 19-Dec-2021 14:02:00 | 105 | Good |
| 19-Dec-2021 14:02:10 | 120 | Good |
| 19-Dec-2021 14:02:20 | 125 | Good |
| 19-Dec-2021 14:02:30 | 140 | Good |
| 19-Dec-2021 14:02:40 | 145 | Good |
| 19-Dec-2021 14:02:50 | 155 | Good |
| 19-Dec-2021 14:03:00 | 165 | Good |
| 19-Dec-2021 14:03:10 | 175 | Good |
| 19-Dec-2021 14:03:20 | 185 | Good |
| 19-Dec-2021 14:03:30 | 195 | Good |
| 19-Dec-2021 14:03:40 | 205 | Good |
| 19-Dec-2021 14:03:50 | 215 | Good |
| 19-Dec-2021 14:04:00 | 225 | Good |
| 19-Dec-2021 14:04:10 | 235 | Good |
| 19-Dec-2021 14:04:20 | 245 | Good |
| 19-Dec-2021 14:04:30 | 255 | Good |
| 19-Dec-2021 14:04:40 | 265 | Good |
| 19-Dec-2021 14:04:50 | 55 | Good |
| 19-Dec-2021 14:05:00 | 65 | Good |
| 19-Dec-2021 14:05:10 | 75 | Good |

| TimeStamp | Tag Value | Data Quality |
|---|---|---|
| 19-Dec-2021 14:05:20 | 85 | Good |
| 19-Dec-2021 14:05:30 | 95 | Good |
| 19-Dec-2021 14:05:40 | 105 | Good |
| 19-Dec-2021 14:05:50 | 115 | Good |
| 19-Dec-2021 14:06:00 | 125 | Good |

Suppose you have run the following query:

```
Select timestamp,value,quality from IHrawdata

where tagname=DeltaPosSample4 and samplingmode=calculated

and timestamp>='19-Dec-2021 14:00:00.000' and timestamp<='19-Dec-2021 14:06:00.000'

and calculationmode=DeltaPos and intervalmilliseconds=1M
```

In the cases where the rate of increase is greater than Delta Max Positive RPH, DELTAPOS = 0. For the other values, DELTAPOS is the difference between the consecutive values (highlighted in bold formatting):

| Time Stamp | Val-ue | Data Quality |
|---|---|---|
| 19-Dec-2021 14:00:00 | 0.00 | 100 |
| 19-Dec-2021 14:01:00 | 0.00 | 100 |
| 19-Dec-2021 14:02:00 | 50.00 | 100 |
| 19-Dec-2021 14:03:00 | 60.00 | 100 |
| 19-Dec-2021 14:04:00 | 60.00 | 100 |
| 19-Dec-2021 14:05:00 | 55.00 | 100 |
| 19-Dec-2021 14:06:00 | 60.00 | 100 |

### Calculating DELTAPOS When Delta Max is not Provided and Data does not Follow a Trend

Suppose you have received the following data:

| Time Stamp | Val-ue | Data Quality |
|---|---|---|
| 19-Dec-2021 14:01:10 | 0 | Good |

| Time Stamp | Val-ue | Data Quality |
|---|---|---|
| 19-Dec-2021 14:01:20 | 5 | Good |
| 19-Dec-2021 14:01:30 | 10 | Good |
| 19-Dec-2021 14:01:40 | 8 | Good |
| 19-Dec-2021 14:01:50 | 20 | Good |
| 19-Dec-2021 14:02:00 | 25 | Good |
| 19-Dec-2021 14:02:10 | 30 | Good |
| 19-Dec-2021 14:02:20 | 19 | Good |
| 19-Dec-2021 14:02:30 | 40 | Good |
| 19-Dec-2021 14:02:40 | 45 | Good |
| 19-Dec-2021 14:02:50 | 50 | Good |
| 19-Dec-2021 14:03:00 | 55 | Good |
| 19-Dec-2021 14:03:10 | 60 | Good |
| 19-Dec-2021 14:03:20 | 38 | Good |
| 19-Dec-2021 14:03:30 | 70 | Good |
| 19-Dec-2021 14:03:40 | 75 | Good |
| 19-Dec-2021 14:03:50 | 80 | Good |
| 19-Dec-2021 14:04:00 | 85 | Good |
| 19-Dec-2021 14:04:10 | 90 | Good |
| 19-Dec-2021 14:04:20 | 95 | Good |
| 19-Dec-2021 14:04:30 | 100 | Good |
| 19-Dec-2021 14:04:40 | 105 | Good |
| 19-Dec-2021 14:04:50 | 110 | Good |
| 19-Dec-2021 14:05:00 | 115 | Good |
| 19-Dec-2021 14:05:10 | 120 | Good |
| 19-Dec-2021 14:05:20 | 125 | Good |

| Time Stamp | Val-ue | Data Quality |
|---|---|---|
| 19-Dec-2021 14:05:30 | 130 | Good |
| 19-Dec-2021 14:05:40 | 135 | Good |
| 19-Dec-2021 14:05:50 | 140 | Good |
| 19-Dec-2021 14:06:00 | 145 | Good |

Suppose you have run the following query:

```
Select timestamp,value,quality from IHrawdata

where tagname=DeltaPosSample1 and samplingmode=calculated

and timestamp>='19-Dec-2021 14:00:00.000' and timestamp<='19-Dec-2021 14:06:00.000'

and calculationmode=DeltaPos and intervalmilliseconds=1M
```

Since you have not provided Delta Max Positive RPH, for data points that are in the increasing trend, the difference is used. For data points that are in the decreasing trend, DELTAPOS is calculated as the difference between the latest value and Delta Min (highlighted in bold formatting). Since Delta Min is not provided, it is considered as 0.

| Time Stamp | Val-ue | DELTAPOS |
|---|---|---|
| 19-Dec-2021 14:01:10 | 0 | 0 |
| 19-Dec-2021 14:01:20 | 5 | 5 |
| 19-Dec-2021 14:01:30 | 10 | 5 |
| 19-Dec-2021 14:01:40 | 8 | **8** |
| 19-Dec-2021 14:01:50 | 20 | 12 |
| 19-Dec-2021 14:02:00 | 25 | 5 |
| 19-Dec-2021 14:02:10 | 30 | 5 |
| 19-Dec-2021 14:02:20 | 19 | **19** |
| 19-Dec-2021 14:02:30 | 40 | 21 |
| 19-Dec-2021 14:02:40 | 45 | 5 |
| 19-Dec-2021 14:02:50 | 50 | 5 |

| Time Stamp | Val-ue | DELTAPOS |
|---|---|---|
| 19-Dec-2021 14:03:00 | 55 | 5 |
| 19-Dec-2021 14:03:10 | 60 | 5 |
| 19-Dec-2021 14:03:20 | 38 | **38** |
| 19-Dec-2021 14:03:30 | 70 | 32 |
| 19-Dec-2021 14:03:40 | 75 | 5 |
| 19-Dec-2021 14:03:50 | 80 | 5 |
| 19-Dec-2021 14:04:00 | 85 | 5 |
| 19-Dec-2021 14:04:10 | 90 | 5 |
| 19-Dec-2021 14:04:20 | 95 | 5 |
| 19-Dec-2021 14:04:30 | 100 | 5 |
| 19-Dec-2021 14:04:40 | 105 | 5 |
| 19-Dec-2021 14:04:50 | 110 | 5 |
| 19-Dec-2021 14:05:00 | 115 | 5 |
| 19-Dec-2021 14:05:10 | 120 | 5 |
| 19-Dec-2021 14:05:20 | 125 | 5 |
| 19-Dec-2021 14:05:30 | 130 | 5 |
| 19-Dec-2021 14:05:40 | 135 | 5 |
| 19-Dec-2021 14:05:50 | 140 | 5 |
| 19-Dec-2021 14:06:00 | 145 | 5 |

DELTAPOS for a given duration is calculated as the sum of the individual DELTAPOS values in that duration as shown in the following table.

| Time Stamp | DELTAPOS Value | Data Quality |
|---|---|---|
| 19-Dec-2021 14:00:00 | 0.00 | 100 |
| 19-Dec-2021 14:01:00 | 0.00 | 100 |

| Time Stamp | DELTAPOS Value | Data Quality |
|---|---|---|
| 19-Dec-2021 14:02:00 | 35.00 | 100 |
| 19-Dec-2021 14:03:00 | 60.00 | 100 |
| 19-Dec-2021 14:04:00 | 90.00 | 100 |
| 19-Dec-2021 14:05:00 | 30.00 | 100 |
| 19-Dec-2021 14:06:00 | 30.00 | 100 |

### Calculating DELTAPOS When Delta Max is Provided and Data does not Follow a Trend

Suppose you have provided the following values:

| | |
|---|---|
| Delta Max | 265 |
| Delta Min | 50 |

Suppose you have received the following data:

| Time Stamp | Val-ue | Data Quality |
|---|---|---|
| 19-Dec-2021 14:01:10 | 50 | Good |
| 19-Dec-2021 14:01:20 | 60 | Good |
| 19-Dec-2021 14:01:30 | 55 | Good |
| 19-Dec-2021 14:01:40 | 80 | Good |
| 19-Dec-2021 14:01:50 | 90 | Good |
| 19-Dec-2021 14:02:00 | 100 | Good |
| 19-Dec-2021 14:02:10 | 110 | Good |
| 19-Dec-2021 14:02:20 | 80 | Good |
| 19-Dec-2021 14:02:30 | 130 | Good |
| 19-Dec-2021 14:02:40 | 110 | Good |
| 19-Dec-2021 14:02:50 | 150 | Good |
| 19-Dec-2021 14:03:00 | 160 | Good |

| Time Stamp | Val-ue | Data Quality |
|---|---|---|
| 19-Dec-2021 14:03:10 | 170 | Good |
| 19-Dec-2021 14:03:20 | 180 | Good |
| 19-Dec-2021 14:03:30 | 190 | Good |
| 19-Dec-2021 14:03:40 | 200 | Good |
| 19-Dec-2021 14:03:50 | 210 | Good |
| 19-Dec-2021 14:04:00 | 220 | Good |
| 19-Dec-2021 14:04:10 | 230 | Good |
| 19-Dec-2021 14:04:20 | 240 | Good |
| 19-Dec-2021 14:04:30 | 250 | Good |
| 19-Dec-2021 14:04:40 | 260 | Good |
| 19-Dec-2021 14:04:50 | 50 | Good |
| 19-Dec-2021 14:05:00 | 60 | Good |
| 19-Dec-2021 14:05:10 | 70 | Good |
| 19-Dec-2021 14:05:20 | 80 | Good |
| 19-Dec-2021 14:05:30 | 90 | Good |
| 19-Dec-2021 14:05:40 | 100 | Good |
| 19-Dec-2021 14:05:50 | 110 | Good |
| 19-Dec-2021 14:06:00 | 120 | Good |

Suppose you have run the following query:

```
Select timestamp,value,quality from IHrawdata

where tagname=DeltaPosSample3 and samplingmode=calculated

and timestamp>='19-Dec-2021 14:00:00.000' and timestamp<='19-Dec-2021 14:06:00.000'

and calculationmode=DeltaPos and intervalmilliseconds=1M
```

Since you have not provided Delta Max Positive RPH, for data points that are in the increasing trend, the difference is used. For data points that are in the decreasing trend, DELTAPOS is calculated as follows: `DELTAPOS = Delta Max - Previous Value + Current Value - Delta Min` These values as highlighted in bold formatting in the following table.

| Time Stamp | Val-ue | DELTAPOS Value |
|---|---|---|
| 19-Dec-2021 14:01:10 | 50 | 0 |
| 19-Dec-2021 14:01:20 | 60 | 10 |
| 19-Dec-2021 14:01:30 | 55 | **210** |
| 19-Dec-2021 14:01:40 | 80 | 25 |
| 19-Dec-2021 14:01:50 | 90 | 10 |
| 19-Dec-2021 14:02:00 | 100 | 10 |
| 19-Dec-2021 14:02:10 | 110 | 10 |
| 19-Dec-2021 14:02:20 | 80 | **185** |
| 19-Dec-2021 14:02:30 | 130 | 50 |
| 19-Dec-2021 14:02:40 | 110 | **195** |
| 19-Dec-2021 14:02:50 | 150 | 40 |
| 19-Dec-2021 14:03:00 | 160 | 10 |
| 19-Dec-2021 14:03:10 | 170 | 10 |
| 19-Dec-2021 14:03:20 | 180 | 10 |
| 19-Dec-2021 14:03:30 | 190 | 10 |
| 19-Dec-2021 14:03:40 | 200 | 10 |
| 19-Dec-2021 14:03:50 | 210 | 10 |
| 19-Dec-2021 14:04:00 | 220 | 10 |
| 19-Dec-2021 14:04:10 | 230 | 10 |
| 19-Dec-2021 14:04:20 | 240 | 10 |
| 19-Dec-2021 14:04:30 | 250 | 10 |
| 19-Dec-2021 14:04:40 | 260 | 10 |
| 19-Dec-2021 14:04:50 | 50 | **5** |
| 19-Dec-2021 14:05:00 | 60 | 10 |
| 19-Dec-2021 14:05:10 | 70 | 10 |

| Time Stamp | Val-ue | DELTAPOS Value |
|---|---|---|
| 19-Dec-2021 14:05:20 | 80 | 10 |
| 19-Dec-2021 14:05:30 | 90 | 10 |
| 19-Dec-2021 14:05:40 | 100 | 10 |
| 19-Dec-2021 14:05:50 | 110 | 10 |
| 19-Dec-2021 14:06:00 | 120 | 10 |

DELTAPOS for a given duration is calculated as the sum of the individual DELTAPOS values in that duration as shown in the following table.

| Time Stamp | Value | Data Quality |
|---|---|---|
| 19-Dec-2021 14:00:00 | 0.00 | 100 |
| 19-Dec-2021 14:01:00 | 0.00 | 100 |
| 19-Dec-2021 14:02:00 | 265.00 | 100 |
| 19-Dec-2021 14:03:00 | 490.00 | 100 |
| 19-Dec-2021 14:04:00 | 60.00 | 100 |
| 19-Dec-2021 14:05:00 | 55.00 | 100 |
| 19-Dec-2021 14:06:00 | 60.00 | 100 |

## DELTANEG Calculation

The following diagrams show how DELTANEG is calculated. If Delta Min is not considered, 0 is considered.

Figure 4. DELTANEG Calculation When Delta Max Negative RPH is not Provided

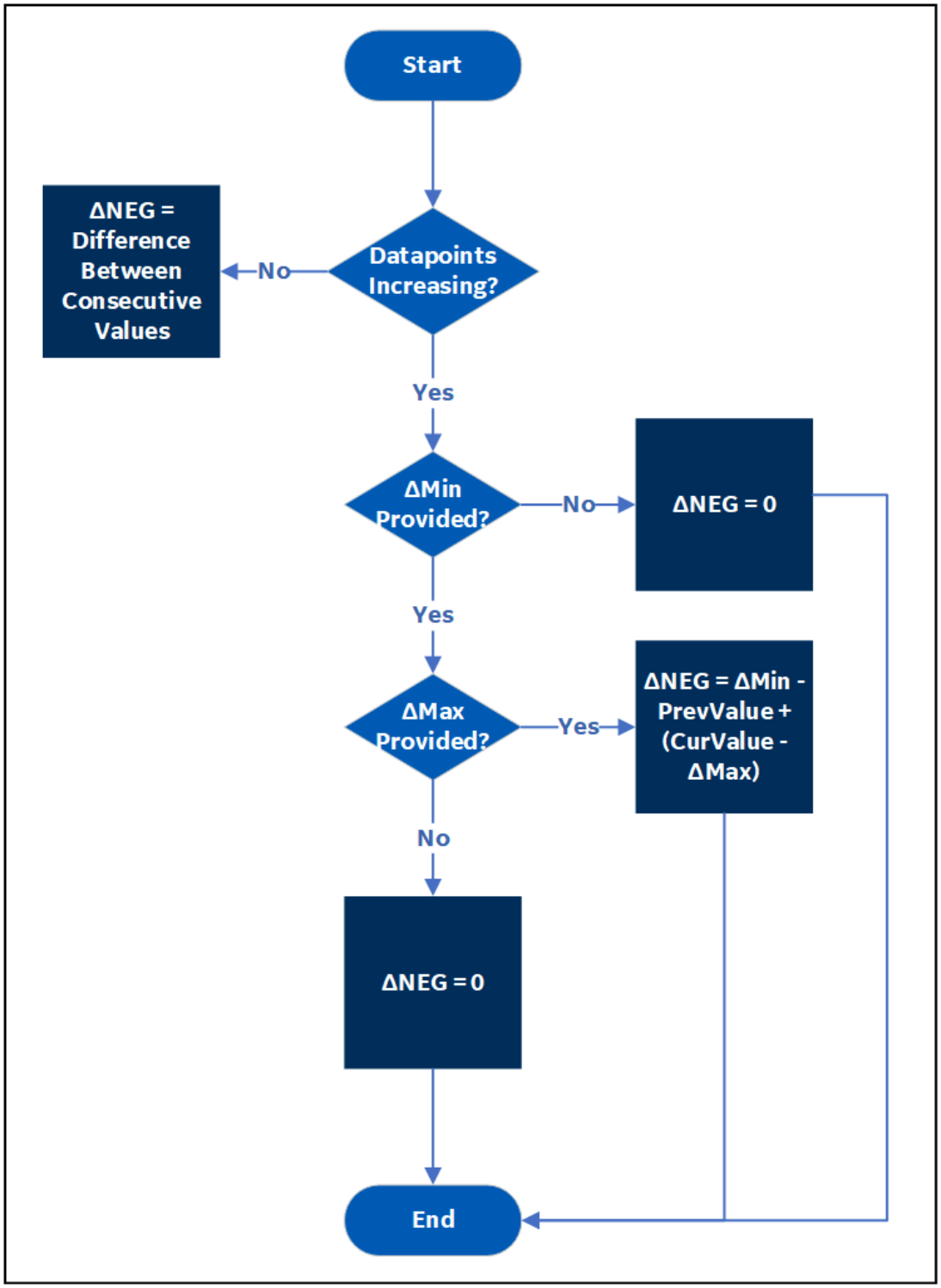
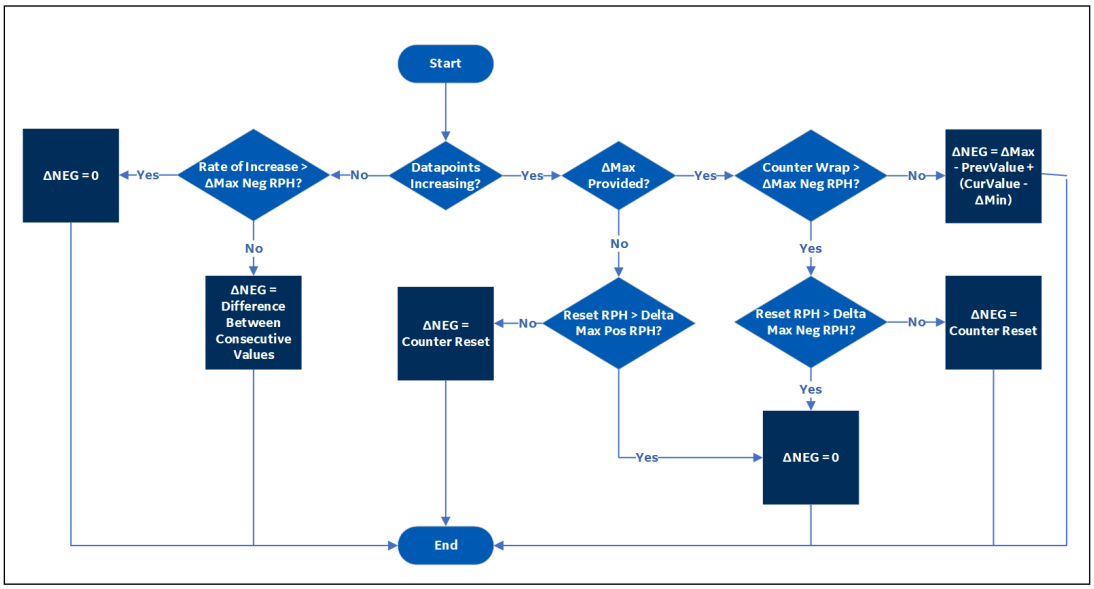


Figure 5. DELTANEG Calculation When Delta Max Negative RPH is Provided




**Note:**

Delta Max Positive RPH is not used to calculate DELTANEG.

## Calculating DELTANEG When Delta Max Negative RPH is Not Provided and Data is in the Decreasing Trend

Suppose you have received the following tag data:

| TimeStamp | Tag Value | Data Quality |
|---|---|---|
| 19-Dec-2021 14:01:10 | 265 | Good |
| 19-Dec-2021 14:01:20 | 260 | Good |
| 19-Dec-2021 14:01:30 | 255 | Good |
| 19-Dec-2021 14:01:40 | 250 | Good |
| 19-Dec-2021 14:01:50 | 245 | Good |
| 19-Dec-2021 14:02:00 | 240 | Good |
| 19-Dec-2021 14:02:10 | 235 | Good |
| 19-Dec-2021 14:02:20 | 230 | Good |
| 19-Dec-2021 14:02:30 | 225 | Good |

| TimeStamp | Tag Value | Data Quality |
|---|---|---|
| 19-Dec-2021 14:02:40 | 220 | Good |
| 19-Dec-2021 14:02:50 | 215 | Good |
| 19-Dec-2021 14:03:00 | 210 | Good |
| 19-Dec-2021 14:03:10 | 205 | Good |
| 19-Dec-2021 14:03:20 | 200 | Good |
| 19-Dec-2021 14:03:30 | 195 | Good |
| 19-Dec-2021 14:03:40 | 190 | Good |
| 19-Dec-2021 14:03:50 | 185 | Good |
| 19-Dec-2021 14:04:00 | 180 | Good |
| 19-Dec-2021 14:04:10 | 175 | Good |
| 19-Dec-2021 14:04:20 | 170 | Good |
| 19-Dec-2021 14:04:30 | 165 | Good |
| 19-Dec-2021 14:04:40 | 160 | Good |
| 19-Dec-2021 14:04:50 | 155 | Good |
| 19-Dec-2021 14:05:00 | 150 | Good |
| 19-Dec-2021 14:05:10 | 145 | Good |
| 19-Dec-2021 14:05:20 | 140 | Good |
| 19-Dec-2021 14:05:30 | 135 | Good |
| 19-Dec-2021 14:05:40 | 130 | Good |
| 19-Dec-2021 14:05:50 | 125 | Good |
| 19-Dec-2021 14:06:00 | 120 | Good |

Suppose you have run the following query:

```
Select timestamp,value,quality from IHrawdata

where tagname=BasicTest and samplingmode=calculated

and timestamp>='19-Dec-2021 14:00:00.000' and timestamp<='19-Dec-2021 14:06:00.000'

and calculationmode=DeltaNeg and intervalmilliseconds=1M
```

Regardless of whether you have provided Delta Min and Delta Max, since you have not provided Delta Max Negative RPH, and since the data points are in the decreasing trend, DELTANEG is calculated as the difference between consecutive data points:

| TimeStamp | DELTANEG Value | Data Quality |
|---|---|---|
| 19-Dec-2021 14:00:00 | 0 | 100 |
| 19-Dec-2021 14:01:00 | 0 | 100 |
| 19-Dec-2021 14:02:00 | -25.00 | 100 |
| 19-Dec-2021 14:03:00 | -30.00 | 100 |
| 19-Dec-2021 14:04:00 | -30.00 | 100 |
| 19-Dec-2021 14:05:00 | -30.00 | 100 |
| 19-Dec-2021 14:06:00 | -30.00 | 100 |

### Calculating DELTANEG When Data Quality is Bad

Suppose you have received the following data:

| Time Stamp | Value | Data Quality |
|---|---|---|
| 19-Dec-2021 14:01:10 | 265 | Good |
| 19-Dec-2021 14:01:20 | 260 | Bad |
| 19-Dec-2021 14:01:30 | 255 | Good |
| 19-Dec-2021 14:01:40 | 250 | Bad |
| 19-Dec-2021 14:01:50 | 245 | Good |
| 19-Dec-2021 14:02:00 | 240 | Bad |
| 19-Dec-2021 14:02:10 | 235 | Good |
| 19-Dec-2021 14:02:20 | 230 | Bad |
| 19-Dec-2021 14:02:30 | 225 | Good |
| 19-Dec-2021 14:02:40 | 220 | Bad |
| 19-Dec-2021 14:02:50 | 215 | Good |
| 19-Dec-2021 14:03:00 | 210 | Bad |

| Time Stamp | Value | Data Quality |
|---|---|---|
| 19-Dec-2021 14:03:10 | 205 | Good |
| 19-Dec-2021 14:03:20 | 200 | Bad |
| 19-Dec-2021 14:03:30 | 195 | Good |
| 19-Dec-2021 14:03:40 | 190 | Bad |
| 19-Dec-2021 14:03:50 | 185 | Good |
| 19-Dec-2021 14:04:00 | 180 | Bad |
| 19-Dec-2021 14:04:10 | 175 | Good |
| 19-Dec-2021 14:04:20 | 170 | Bad |
| 19-Dec-2021 14:04:30 | 165 | Good |
| 19-Dec-2021 14:04:40 | 160 | Bad |
| 19-Dec-2021 14:04:50 | 155 | Good |
| 19-Dec-2021 14:05:00 | 150 | Bad |
| 19-Dec-2021 14:05:10 | 145 | Good |
| 19-Dec-2021 14:05:20 | 140 | Bad |
| 19-Dec-2021 14:05:30 | 135 | Good |
| 19-Dec-2021 14:05:40 | 130 | Bad |
| 19-Dec-2021 14:05:50 | 125 | Good |
| 19-Dec-2021 14:06:00 | 120 | Bad |

Suppose you have run the following query:

```
Select timestamp,value,quality from IHrawdata

where tagname=BasicTestWithBad and samplingmode=calculated

and timestamp>='19-Dec-2021 14:00:00.000' and timestamp<='19-Dec-2021 14:06:00.000'

and calculationmode=DeltaNeg and intervalmilliseconds=1M
```

Data with bad quality is not considered in the calculation. Therefore, DELTANEG is the difference between consecutive data points with good quality.

| Time Stamp | Val-ue | Data Quality | DELTANEG |
|---|---|---|---|
| 19-Dec-2021 14:01:10 | 265 | Good | 0 |
| 19-Dec-2021 14:01:20 | 260 | Bad | 0 |
| 19-Dec-2021 14:01:30 | 255 | Good | -10 |
| 19-Dec-2021 14:01:40 | 250 | Bad | 0 |
| 19-Dec-2021 14:01:50 | 245 | Good | -10 |
| 19-Dec-2021 14:02:00 | 240 | Bad | 0 |
| 19-Dec-2021 14:02:10 | 235 | Good | -10 |
| 19-Dec-2021 14:02:20 | 230 | Bad | 0 |
| 19-Dec-2021 14:02:30 | 225 | Good | -10 |
| 19-Dec-2021 14:02:40 | 220 | Bad | 0 |
| 19-Dec-2021 14:02:50 | 215 | Good | -10 |
| 19-Dec-2021 14:03:00 | 210 | Bad | 0 |
| 19-Dec-2021 14:03:10 | 205 | Good | -10 |
| 19-Dec-2021 14:03:20 | 200 | Bad | 0 |
| 19-Dec-2021 14:03:30 | 195 | Good | -10 |
| 19-Dec-2021 14:03:40 | 190 | Bad | 0 |
| 19-Dec-2021 14:03:50 | 185 | Good | -10 |
| 19-Dec-2021 14:04:00 | 180 | Bad | 0 |
| 19-Dec-2021 14:04:10 | 175 | Good | -10 |
| 19-Dec-2021 14:04:20 | 170 | Bad | 0 |
| 19-Dec-2021 14:04:30 | 165 | Good | -10 |
| 19-Dec-2021 14:04:40 | 160 | Bad | 0 |
| 19-Dec-2021 14:04:50 | 155 | Good | -10 |
| 19-Dec-2021 14:05:00 | 150 | Bad | 0 |
| 19-Dec-2021 14:05:10 | 145 | Good | -10 |
| 19-Dec-2021 14:05:20 | 140 | Bad | 0 |

| Time Stamp | Val-ue | Data Quality | DELTANEG |
|---|---|---|---|
| 19-Dec-2021 14:05:30 | 135 | Good | -10 |
| 19-Dec-2021 14:05:40 | 130 | Bad | 0 |
| 19-Dec-2021 14:05:50 | 125 | Good | -10 |
| 19-Dec-2021 14:06:00 | 120 | Bad | 0 |

DELTANEG for a given duration is calculated as the sum of the individual DELTANEG values in that duration as shown in the following table.

| Time Stamp | DELTANEG | Data Quality |
|---|---|---|
| 19-Dec-2021 14:00:00 | 0.00 | 100 |
| 19-Dec-2021 14:01:00 | 0.00 | 100 |
| 19-Dec-2021 14:02:00 | -20.00 | 100 |
| 19-Dec-2021 14:03:00 | -30.00 | 100 |
| 19-Dec-2021 14:04:00 | -30.00 | 100 |
| 19-Dec-2021 14:05:00 | -30.00 | 100 |
| 19-Dec-2021 14:06:00 | -30.00 | 100 |

### Calculating DELTANEG When Delta Max Negative RPH is not Provided and Data is in the Increasing Trend

Suppose you have received the following data:

| TimeStamp | Tag Value | Data Quality |
|---|---|---|
| 19-Dec-2021 14:01:10 | 265 | Good |
| 19-Dec-2021 14:01:20 | 260 | Good |
| 19-Dec-2021 14:01:30 | 262 | Good |
| 19-Dec-2021 14:01:40 | 250 | Good |
| 19-Dec-2021 14:01:50 | 245 | Good |
| 19-Dec-2021 14:02:00 | 240 | Good |

| TimeStamp | Tag Value | Data Quality |
|---|---|---|
| 19-Dec-2021 14:02:10 | 235 | Good |
| 19-Dec-2021 14:02:20 | 230 | Good |
| 19-Dec-2021 14:02:30 | 231 | Good |
| 19-Dec-2021 14:02:40 | 233 | Good |
| 19-Dec-2021 14:02:50 | 215 | Good |
| 19-Dec-2021 14:03:00 | 210 | Good |
| 19-Dec-2021 14:03:10 | 205 | Good |
| 19-Dec-2021 14:03:20 | 220 | Good |
| 19-Dec-2021 14:03:30 | 195 | Good |
| 19-Dec-2021 14:03:40 | 190 | Good |
| 19-Dec-2021 14:03:50 | 185 | Good |
| 19-Dec-2021 14:04:00 | 180 | Good |
| 19-Dec-2021 14:04:10 | 175 | Good |
| 19-Dec-2021 14:04:20 | 170 | Good |
| 19-Dec-2021 14:04:30 | 165 | Good |
| 19-Dec-2021 14:04:40 | 160 | Good |
| 19-Dec-2021 14:04:50 | 155 | Good |
| 19-Dec-2021 14:05:00 | 150 | Good |
| 19-Dec-2021 14:05:10 | 145 | Good |
| 19-Dec-2021 14:05:20 | 140 | Good |
| 19-Dec-2021 14:05:30 | 135 | Good |
| 19-Dec-2021 14:05:40 | 130 | Good |
| 19-Dec-2021 14:05:50 | 125 | Good |
| 19-Dec-2021 14:06:00 | 120 | Good |

Suppose you have run the following query:

```
Select timestamp,value,quality from IHrawdata

where tagname=DeltaNegSample1 and samplingmode=calculated

and timestamp>='19-Dec-2021 14:00:00.000' and timestamp<='19-Dec-2021 14:06:00.000'

and calculationmode=DeltaNeg and intervalmilliseconds=1M
```

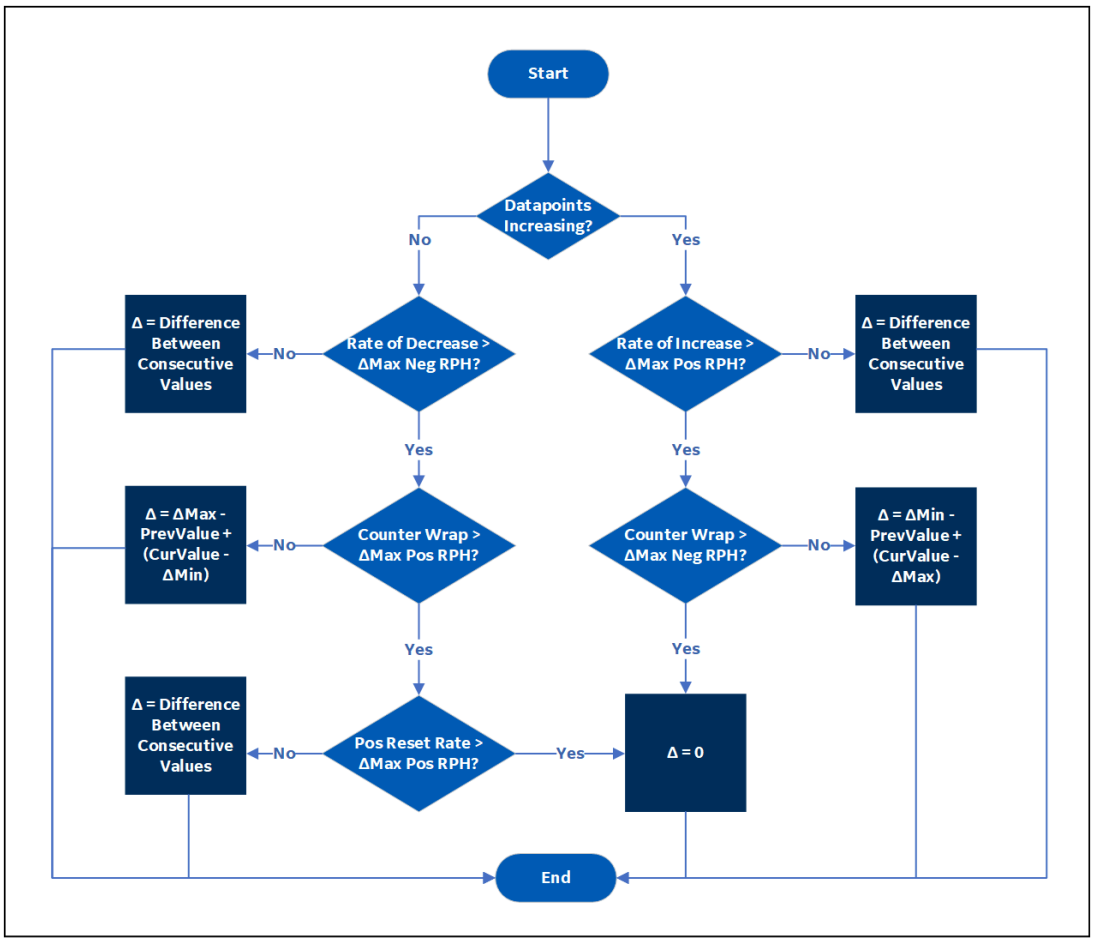
For data that is in the increasing trend, since you have not provided values for Delta Max, Delta Min, and Delta Max Negative RPH, DELTANEG = 0. For the other cases, DELTANEG is the difference between the values.

| Time Stamp | DELTANEG Value | Data Quality |
|---|---|---|
| 19-Dec-2021 14:00:00 | 0.00 | 100 |
| 19-Dec-2021 14:01:00 | 0.00 | 100 |
| 19-Dec-2021 14:02:00 | -27.00 | 100 |
| 19-Dec-2021 14:03:00 | -33.00 | 100 |
| 19-Dec-2021 14:04:00 | -45.00 | 100 |
| 19-Dec-2021 14:05:00 | -30.00 | 100 |
| 19-Dec-2021 14:06:00 | -30.00 | 100 |

## DELTA Calculation

The following diagram shows how DELTA is calculated.

Figure 6. DELTA Calculation




**Note:**
To calculate Delta, all of these values are required: Delta Max, Delta Min, Delta Max Positive RPH, and Delta Max Negative RPH.

## Other Calculation Modes

**STATECOUNT**

The STATECOUNT calculation mode counts the number of times a tag has transitioned to another state from a previous state. A state transition is counted when the previous good sample is not equal to the state value and the next good sample is equal to state value.

The STATECOUNT calculation mode cannot be used on tags of BLOB data type.

- **Value:** The number of transitions into the state in a given time interval.
- **Quality:** The percent good is 100 if there are no bad samples within the time interval. Otherwise, the percent good is the percent of interval time that the value was of good quality.
- **Anticipated usage:** The STATECOUNT calculation mode is useful to determine the number of times a value transitioned to a certain state such as when a digital state was turned on or the enumerated value was of certain value. It should mostly be used with integer values because it may not exactly match a float state value due to rounding.

**STATETIME Calculation Mode:** The STATETIME calculation mode retrieves the duration that a tag was in a given state within an interval.

- **Value:** The STATETIME calculation mode retrieves the total number of milliseconds during the interval for which the data was in the state value.
- **Quality:**

  The percent good is 100 if the data is good quality for the entire the time interval.

  Import this data to use in the examples.

```
[Tags]
Tagname,DataType,HiEngineeringUnits,LoEngineeringUnits
STATECOUNTTAG,SingleInteger,60,0
STATEBADTAG,SingleInteger,60,0
STATEBADTAG2,SingleInteger,60,0
[Data]
Tagname,TimeStamp,Value,DataQuality
STATECOUNTTAG,06-Aug-2012 8:59:00.000,2,Good
STATECOUNTTAG,06-Aug-2012 9:08:00.000,4,Good
STATECOUNTTAG,06-Aug-2012 9:14:00.000,4,Good
STATECOUNTTAG,06-Aug-2012 9:22:00.000,2,Good
STATEBADTAG,06-Aug-2012 8:59:00.000,2,Good
STATEBADTAG,06-Aug-2012 9:08:00.000,0,Bad
STATEBADTAG,06-Aug-2012 9:14:00.000,2,Good
STATEBADTAG,06-Aug-2012 9:22:00.000,4,Good
STATEBADTAG2,06-Aug-2012 8:59:00.000,2,Good
STATEBADTAG2,06-Aug-2012 9:08:00.000,0,Bad
```

```
STATEBADTAG2,06-Aug-2012 9:14:00.000,4,Good

STATEBADTAG2,06-Aug-2012 9:22:00.000,2,Good
```

- **Anticipated usage:** The STATETIME calculation mode is useful to determine the duration the tag was in a particular state. For example, if a tag records the state of a motor you can use state count to determine the duration a motor was in **idle** state.

**OPCQOR and OPCQAND Calculation Modes:**

The OPCQOR calculation mode is a bit-wise OR operation of all the 16 bit OPC qualities of the raw samples stored in the specified interval. This calculation mode can be used only if you have set the ”Store OPC Quality” to ”Enabled” in Historian Administrator and your data is coming from an OPC Collector.

The OPCQAND calculation mode is a bit wise AND operation of all the 16 bit OPC qualities of the raw samples stored in the specified interval. This calculation mode can be used only if you have set the ”Store OPC Quality” to Enabled in Historian Administrator and your data is coming from an OPC Collector.

When collecting data from OPC servers, the Historian OPC collector will convert the 16 bits of OPC quality to a Historian quality and subquality. When ”Store OPC Quality” is enabled, the 16 bits are also stored with the data and can be retrieved here.

Use the returned value from OPCQOR like a data quality. By using OPCQOR and OPCQAND values, you can see if a condition occurred during an interval and therefore know how trustworthy your returned data is.

- **Value:** The 16 bits are in the following format: *VVVVVVVVQQSSSSSS*

  The first 16 bits are for vendor to fill in. The next two are the actual quality, good, bad,uncertain. The rest of the bits are subquality.

  - OPC good is a decimal 192 which is binary 0000000011000000.
  - OPC bad is all zeros 0000000000000000.
- **Quality:** The percent good is 100 if all the samples have good Historian quality. The Historian quality is based on the OPC quality but both the qualities are not the same.
- **Anticipated usage:**

  The OPCQOR and OPCQAND calculation modes are useful if you want to the know the quality of your samples between a time interval. For example,if you want to know how many of your samples from 3pm to 4pm had the following quality:

    - All good - If you do an OPCAND from 3 P.M. to 4 P.M and get the result as 0000000011000000 which is 192 decimal, it means that the value was good for the whole time.
    - All bad - If OPCOR returns 0, then the data was bad the whole time.
    - Some bad - If you do a OPCOR and get the result as 0000000011000000 which is 192 decimal, it means that there were at least some good values. If you do an OPCAND and get the result as 0000000000000000, it means that at least some data was bad.

**TagStats Calculation Mode:** The TagStats calculation mode returns multiple values for a tag in a single query. The TagStats calculation mode returns the values by appending the calculation mode to the tagname. For example, when you query `tag1` the result will be `tag1.Min` which is the result of the minimum calculation mode and `tag1.Max`, the result of the maximum calculation mode. The calculation mode is appended to the tagname.

- **Value:** A query will return multiple values for the same timestamp. They are the results of each individual calculation mode. For more information on different Calculation Modes, refer to the corresponding sections
- **Quality:** There is no single overall quality for the query, only a quality per calculation mode.

### Calculating the state count of good quality data

This example shows a simple case of counting state transitions. In this example, the value 4 means that a machine is running so we want to count the number of times the tag transitioned from some other value to 4.

Import the following data.

```
[Tags]
Tagname,DataType,HiEngineeringUnits,LoEngineeringUnits
STATECOUNTTAG,SingleInteger,60,0
STATEBADTAG,SingleInteger,60,0
STATEBADTAG2,SingleInteger,60,0
[Data]
Tagname,TimeStamp,Value,DataQuality
STATECOUNTTAG,06-Aug-2012 8:59:00.000,2,Good
STATECOUNTTAG,06-Aug-2012 9:08:00.000,4,Good
STATECOUNTTAG,06-Aug-2012 9:14:00.000,4,Good
STATECOUNTTAG,06-Aug-2012 9:22:00.000,2,Good
STATEBADTAG,06-Aug-2012 8:59:00.000,2,Good
```

```
STATEBADTAG,06-Aug-2012 9:08:00.000,0,Bad
STATEBADTAG,06-Aug-2012 9:14:00.000,2,Good
STATEBADTAG,06-Aug-2012 9:22:00.000,4,Good
STATEBADTAG2,06-Aug-2012 8:59:00.000,2,Good
STATEBADTAG2,06-Aug-2012 9:08:00.000,0,Bad
STATEBADTAG2,06-Aug-2012 9:14:00.000,4,Good
STATEBADTAG2,06-Aug-2012 9:22:00.000,2,Good
```

Execute the following query in Historian Interactive SQL:

```
set starttime='08/06/2012 8:00:00',endtime='08/06/2012 10:00:00'
select timestamp,value,quality from ihrawdata where tagname = STATECOUNTTAG and samplingmode=Calculation and
CalculationMode=StateCount and IntervalMilliseconds=20m and statevalue=4
```

The following results are returned:

| Time Stamp | Value | Quality |
|---|---|---|
| 8/6/2012 08:20:00 | 0.000000000000 | 0.0000000 |
| 8/6/2012 08:40:00 | 0.000000000000 | 0.0000000 |
| 8/6/2012 09:00:00 | 0.000000000000 | 5.0000000 |
| 8/6/2012 09:40:00 | 1.000000000000 | 100.0000000 |
| 8/6/2012 09:40:00 | 0.000000000000 | 100.0000000 |
| 8/6/2012 10:00:00 | 0.000000000000 | 100.0000000 |

Note that the transition from 2 to 4 (machine started running) happened at 9:08, so it is included in the 9:00 to 9:20 interval.

The data was of bad quality until 8:59:00 which is for 1 minute of the 20 minute interval. The percent good for that interval is 5.

There are two samples with the value 4. We do not count the number of times the statevalue occurred, but the number of transitions from some other value to the state value.

We only count transitions into the state value not out of the state value. So, the transition from 4 to 2 is not counted.

### Calculating the state count of bad quality data

Note that this tag had a bad data sample when the collector was restarted. This does not, however, affect the state count.

Run the following query:

```
set starttime='08/06/2012 8:00:00',endtime='08/06/2012 10:00:00'

select timestamp,value,quality from ihrawdata where tagname = STATEBADTAG and samplingmode=Calculation and

CalculationMode=StateCount and IntervalMilliseconds=20m and statevalue=4
```

The following results are returned:

| Time Stamp | Value | Quality |
|---|---|---|
| 8/6/2012 08:20:00 | 0.000000000000 | 0.0000000 |
| 8/6/2012 08:40:00 | 0.000000000000 | 0.0000000 |
| 8/6/2012 09:00:00 | 0.000000000000 | 5.0000000 |
| 8/6/2012 09:20:00 | 0.000000000000 | 70.0000000 |
| 8/6/2012 09:40:00 | 1.000000000000 | 100.0000000 |
| 8/6/2012 10:00:00 | 0.000000000000 | 100.0000000 |

Note that the bad value is ignored and the state change that happened at 9:22 is counted. We do not know if the machine had started and stopped while the collector was shutdown.

If the value did change to running while the collector was shut down then that change is counted as in shown in the following example:

```
set starttime='08/06/2012 8:00:00',endtime='08/06/2012 10:00:00'

select timestamp,value,quality from ihrawdata where tagname = STATEBADTAG2 and samplingmode=Calculation and

CalculationMode=StateCount and IntervalMilliseconds=20m and statevalue=4
```

| Time Stamp | Value | Quality |
|---|---|---|
| 8/6/2012 08:20:00 | 0.000000000000 | 0.0000000 |
| 8/6/2012 08:40:00 | 0.000000000000 | 0.0000000 |
| 8/6/2012 09:00:00 | 0.000000000000 | 5.0000000 |
| 8/6/2012 09:20:00 | 1.000000000000 | 70.0000000 |

| Time Stamp | Value | Quality |
|---|---|---|
| 8/6/2012 09:40:00 | 0.000000000000 | 100.0000000 |
| 8/6/2012 10:00:00 | 0.000000000000 | 100.0000000 |

**Note:**
The state change at 9:14 is counted and returned in the 9:20 interval.

### Calculating the state count of enumerated set data

When querying a tag that uses enumerated sets, use the string state name as the state value.

Using the data from previous example, assume that the STATECOUNTTAG had an enumerated set with the values as 2=Stopped and 4=Running.

You should use this query with statevalue of Running instead of the native value 4.

```
set starttime='08/06/2012 8:00:00',endtime='08/06/2012 10:00:00'
select timestamp,value,quality from ihrawdata where tagname = STATECOUNTTAG and samplingmode=Calculation and
CalculationMode=StateCount and IntervalMilliseconds=20m and statevalue='Running'
```

The results match with the results when statevalue=4 is used.

| Time Stamp | Value | Quality |
|---|---|---|
| 8/6/2012 08:20:00 | 0.000000000000 | 0.0000000 |
| 8/6/2012 08:40:00 | 0.000000000000 | 0.0000000 |
| 8/6/2012 09:00:00 | 0.000000000000 | 5.0000000 |
| 8/6/2012 09:20:00 | 1.000000000000 | 100.0000000 |
| 8/6/2012 09:40:00 | 0.000000000000 | 100.0000000 |
| 8/6/2012 10:00:00 | 0.000000000000 | 100.0000000 |

### Calculating the state time of good quality data

Run this query in the Historian Interactive SQL:

```
set starttime='08/06/2012 8:00:00',endtime='08/06/2012 10:00:00'
select timestamp,value,quality from ihrawdata where tagname = STATECOUNTTAG and samplingmode=Calculation and
CalculationMode=StateTime and IntervalMilliseconds=20m and statevalue=4
```

The following results are returned:

| Time Stamp | Value | Quality |
|---|---|---|
| 8/6/2012 08:20:00 | 0.000000000000 | 0.0000000 |
| 8/6/2012 08:40:00 | 0.000000000000 | 0.0000000 |
| 8/6/2012 09:00:00 | 0.000000000000 | 5.0000000 |
| 8/6/2012 09:20:00 | 720,000.000000000000 | 100.0000000 |
| 8/6/2012 09:40:00 | 120,0.000000000000 | 100.0000000 |
| 8/6/2012 10:00:00 | 0.000000000000 | 100.0000000 |

A 20 minute interval is 20*60*1000=1200000 milliseconds. In the 9:00 to 9:20 interval the value was in state 4 from 9:08 to 9:20 which is 12 minutes * 60 *1000 = 720000 milliseconds.

In the 9:20 to 9:40 interval the value was in state 4 from 9:20 to 9:22 which is 2*60*1000 = 120000 milliseconds.

### Calculating the state time of bad quality data

This tag has a bad data sample such as when the collector was restarted. A new value is recorded when the collector is started.

```
set starttime='08/06/2012 8:00:00',endtime='08/06/2012 10:00:00'

select timestamp,value,quality from ihrawdata where tagname = STATEBADTAG2 and samplingmode=Calculation and

CalculationMode=StateTime and IntervalMilliseconds=20m and statevalue=4


Tagname,TimeStamp,Value,DataQuality

STATEBADTAG2,06-Aug-2012 8:59:00.000,2,Good

STATEBADTAG2,06-Aug-2012 9:08:00.000,0,Bad

STATEBADTAG2,06-Aug-2012 9:14:00.000,4,Good

STATEBADTAG2,06-Aug-2012 9:22:00.000,2,Good
```

The following results are returned:

| Time Stamp | Value | Quality |
|---|---|---|
| 8/6/2012 08:20:00 | 0.000000000000 | 0.0000000 |
| 8/6/2012 08:40:00 | 0.000000000000 | 0.0000000 |

| Time Stamp | Value | Quality |
|---|---|---|
| 8/6/2012 09:00:00 | 0.000000000000 | 5.0000000 |
| 8/6/2012 09:20:00 | 360,000.000000000000 | 70.0000000 |
| 8/6/2012 09:40:00 | 120,000.000000000000 | 100.0000000 |
| 8/6/2012 10:00:00 | 0.000000000000 | 100.0000000 |

In the interval between 9:00 to 9:20, the value was in state 4 from 9:14 to 9:20 = 6 minutes * 60 * 1000 = 360000 milliseconds.

In the interval between 9:20 to 9:40, the value was in state 4 from 9:20 to 9:22 = 2 minutes * 60 * 1000 = 120000 milliseconds.

## Calculating the OPCQOR

The following is the data set used to run the query on.

```
TagName,Timestamp,Value
DATA1:Bad-0 OPC-(60)(08/09/12 18:00:01.000,Val=10
DATA2:Bad-0 OPC-(59)(08/09/12 18:00:02.000,Val=10
DATA3:Bad-0 OPC-(58)(08/09/12 18:00:03.000),Val=10
DATA4:Bad-0 OPC-(57)(08/09/12 18:00:04.000),Val=10
DATA5:Bad-0 OPC-(56)(08/09/12 18:00:05.000),Val=10
DATA6:Bad-0 OPC-(55)(08/09/12 18:00:06.000),Val=10
DATA7:Bad-0 OPC-(54)(08/09/12 18:00:07.000),Val=10
DATA8:Bad-0 OPC-(53)(08/09/12 18:00:08.000),Val=10
DATA9:Bad-0 OPC-(52)(08/09/12 18:00:09.000),Val=10
DATA10:Bad-0 OPC-(51)(08/09/12 18:00:10.000),Val=10
DATA11:Bad-0 OPC-(50)(08/09/12 18:00:11.000),Val=10
```

The following query retrieves the OPCQOR data with a start time of 18:00:00 and end time of 18:00:10 with a 2 second time interval.

```
set starttime='08/09/2012 18:00:00',endtime='08/09/2012 18:00:10'
select tagname,timestamp,value,Quality from ihrawdata where tagname like OPCQualityDataTag and samplingmode=Calculated
and calculationmode=OPCQOR and IntervalMilliseconds=2S
```

The following output is retrieved.

| Tag Name | Time Stamp | Value | Quality |
|---|---|---|---|
| OPCQualityDataTag | 8/9/2012 18:00:02 | 63.000000000000 | 50.0000000 |
| OPCQualityDataTag | 8/9/2012 18:00:04 | 59.000000000000 | 100.0000000 |
| OPCQualityDataTag | 8/9/2012 18:00:06 | 63.000000000000 | 100.0000000 |
| OPCQualityDataTag | 8/9/2012 18:00:08 | 55.000000000000 | 100.0000000 |
| OPCQualityDataTag | 8/9/2012 18:00:10 | 55.000000000000 | 100.0000000 |

## Calculating the OPCQAND

The following query retrieves the OPCQAND data with a start time of 18:00:00 and end time of 18:00:10 with a 2 second time interval.

```
set starttime='08/09/2012 18:00:00',endtime='08/09/2012 18:00:10'

select tagname,timestamp,value,Quality,opcquality from ihrawdata where tagname like OPCQualityDataTag and

samplingmode=Calculated and calculationmode=OPCQAND and IntervalMilliseconds=2S
```

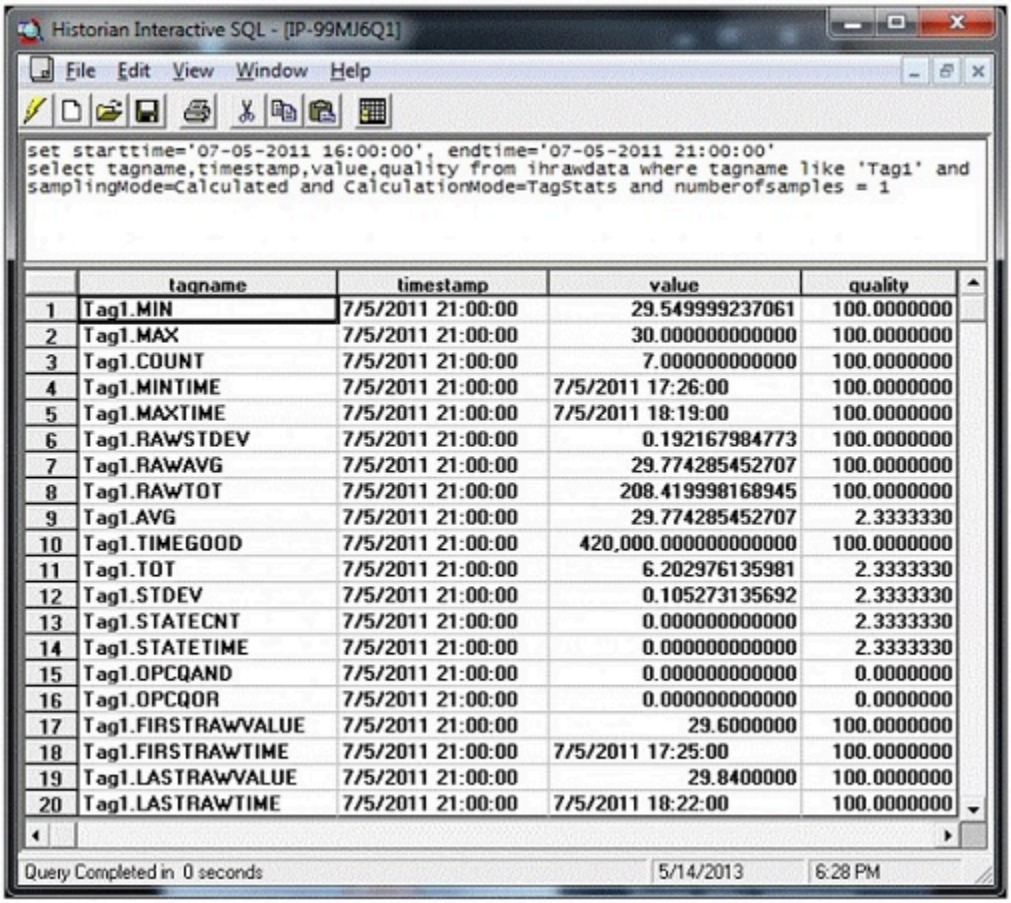
he following output is retrieved.

| Tag Name | Time Stamp | Value | Quality |
|---|---|---|---|
| OPCQualityDataTag | 8/9/2012 18:00:02 | 50.0000000 | 0 |
| OPCQualityDataTag | 8/9/2012 18:00:04 | 100.0000000 | 0 |
| OPCQualityDataTag | 8/9/2012 18:00:06 | 100.0000000 | 0 |
| OPCQualityDataTag | 8/9/2012 18:00: | 100.0000000 | 0 |
| OPCQualityDataTag | 8/9/2012 18:00:10 | 100.0000000 | 0 |

## Using TagStats Calculation Mode

This image displays the TagStats calculation mode example in the Proficy Historian Interactive SQL Application.

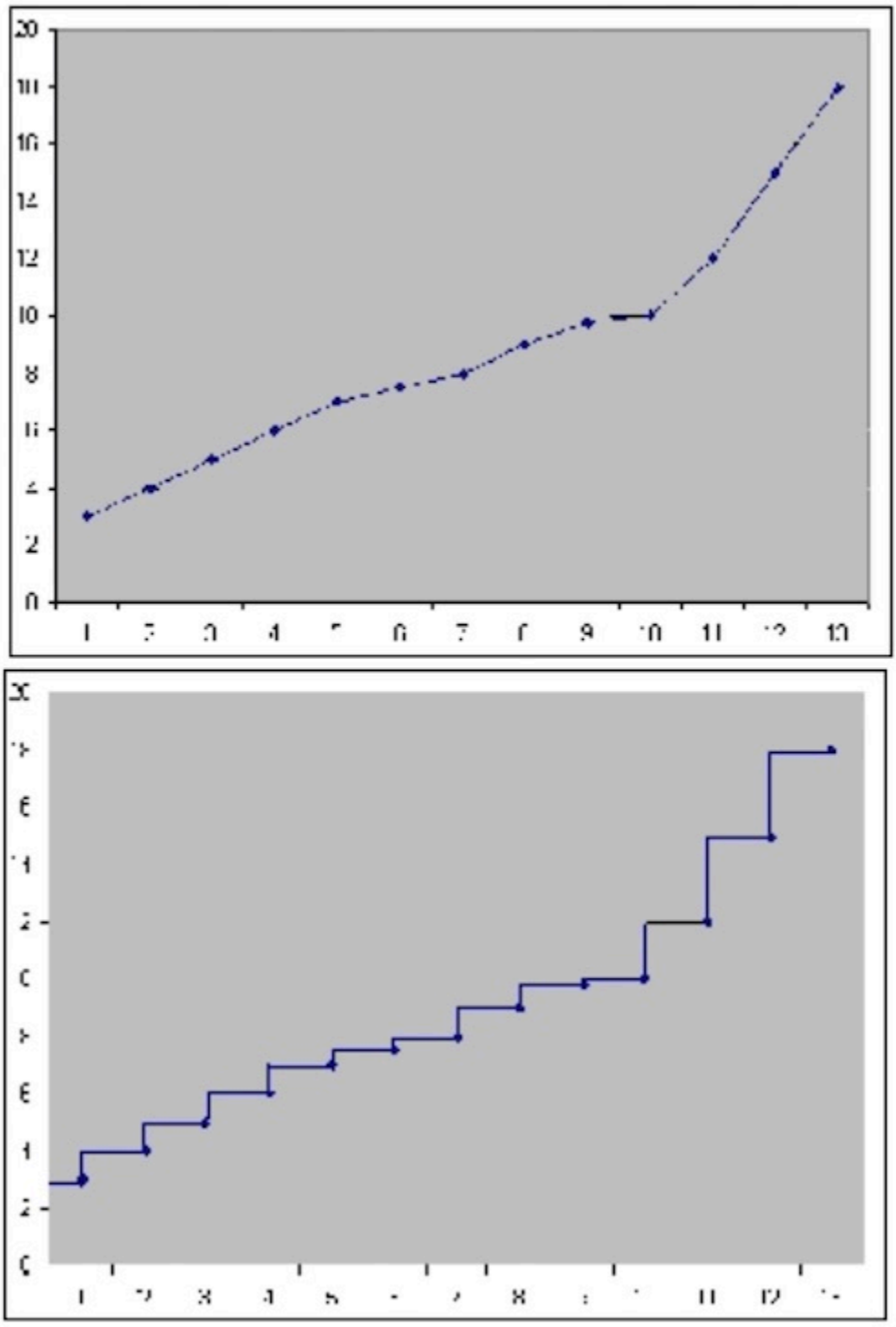
In this example we perform the calculations for a single interval by giving `numberofsamples=1`.




| | tagname | timestamp | value | quality |
|---|---|---|---|---|
| 1 | Tag1.MIN | 7/5/2011 21:00:00 | 29.549999237061 | 100.0000000 |
| 2 | Tag1.MAX | 7/5/2011 21:00:00 | 30.000000000000 | 100.0000000 |
| 3 | Tag1.COUNT | 7/5/2011 21:00:00 | 7.000000000000 | 100.0000000 |
| 4 | Tag1.MINTIME | 7/5/2011 21:00:00 | 7/5/2011 17:26:00 | 100.0000000 |
| 5 | Tag1.MAXTIME | 7/5/2011 21:00:00 | 7/5/2011 18:19:00 | 100.0000000 |
| 6 | Tag1.RAWSTDEV | 7/5/2011 21:00:00 | 0.192167984773 | 100.0000000 |
| 7 | Tag1.RAWAVG | 7/5/2011 21:00:00 | 29.774285452707 | 100.0000000 |
| 8 | Tag1.RAWTOT | 7/5/2011 21:00:00 | 208.419998168945 | 100.0000000 |
| 9 | Tag1.AVG | 7/5/2011 21:00:00 | 29.774285452707 | 2.3333330 |
| 10 | Tag1.TIMEGOOD | 7/5/2011 21:00:00 | 420,000.000000000000 | 100.0000000 |
| 11 | Tag1.TOT | 7/5/2011 21:00:00 | 6.202976135981 | 2.3333330 |
| 12 | Tag1.STDEV | 7/5/2011 21:00:00 | 0.105273135692 | 2.3333330 |
| 13 | Tag1.STATECNT | 7/5/2011 21:00:00 | 0.000000000000 | 2.3333330 |
| 14 | Tag1.STATETIME | 7/5/2011 21:00:00 | 0.000000000000 | 2.3333330 |
| 15 | Tag1.OPCQAND | 7/5/2011 21:00:00 | 0.000000000000 | 0.0000000 |
| 16 | Tag1.OPCQOR | 7/5/2011 21:00:00 | 0.000000000000 | 0.0000000 |
| 17 | Tag1.FIRSTRAWVALUE | 7/5/2011 21:00:00 | 29.6000000 | 100.0000000 |
| 18 | Tag1.FIRSTRAWTIME | 7/5/2011 21:00:00 | 7/5/2011 17:25:00 | 100.0000000 |
| 19 | Tag1.LASTRAWVALUE | 7/5/2011 21:00:00 | 29.8400000 | 100.0000000 |
| 20 | Tag1.LASTRAWTIME | 7/5/2011 21:00:00 | 7/5/2011 18:22:00 | 100.0000000 |

## StepValue Tag Property

Retrieval generally does not take into account how a value changes. When retrieving data from the archive, Historian will attempt to interpolate it, which may result in an inaccurate representation of the data's real world changes, such as that shown in the following figure.



In order for Historian to know that a tag did not ramp down between reported values, the StepValue tag property must be applied. This tag property is used to indicate that the value in the real world changes in a sharp step instead of a smooth linear interpolation. An example would be a digital signal that quickly goes 0 to 1. Or, a flow rate that goes 5 to 25 when an upstream valve is opened.

**Note:**
The StepValue tag property only affects retrieval of Average values in Historian. It does not affect data collection or storage.

**Example: Reporting Step Change**

Copy and paste the following into an empty CSV file and import the file with the File collector.

```
[Tags]
Tagname,DataType,HiEngineeringUnits,LoEngineeringUnits
TAG1,SingleFloat,100,0
[Data]
Tagname,TimeStamp,Value,DataQuality
tag1,9/19/05 05:15:00,26.41,Good
tag1,9/19/05 06:15:00,26.45,Good
tag1,9/19/05 07:15:00,26.59,Good
tag1,9/19/05 08:15:00,26.58,Good
tag1,9/19/05 09:15:00,26.36,Good
tag1,9/19/05 10:15:00,10.74,Good
tag1,9/19/05 11:15:00,11.00,Good
tag1,9/19/05 12:15:00,10.94,Good
tag1,9/19/05 13:15:00,11.03,Good
```

Set the StepValue=TRUE in Historian Administrator. Then, use the following query to retrieve data using Average with a 15 minute interval.

```
select * from ihrawdata where tagname=TAG1 and timestamp > '9/19/05 09:30:00'
and timestamp <= '9/19/05 11:30:00' and calculationmode=average and
intervalmilliseconds=15m
```

You will see the following results, which show two distinct steps:

| Value | Quality |
|---|---|
| 09:45:00 | 26.36 |
| 10:00:00 | 26.36 |
| 10:15:00 | 26.36 |

| Value | Quality |
|---|---|
| 10:30:00 | 10.74 |
| 10:45:00 | 10.74 |
| 11:00:00 | 10.74 |
| 11:15:00 | 10.74 |
| 11:30:00 | 11.00 |

If you set the `StepValue=FALSE` and run the same query, you will see the following results, which reflect interpolated values.

| Value | Quality |
|---|---|
| 09:45:00 | 22.46 |
| 10:00:00 | 18.55 |
| 10:15:00 | 14.64 |
| 10:30:00 | 10.74 |
| 10:45:00 | 10.80 |
| 11:00:00 | 10.87 |
| 11:15:00 | 10.93 |
| 11:30:00 | 11.00 |

**Example: No raw sample at start time**

Copy and paste these lines into an empty CSV file and import the file with the File collector

```
[Tags]
Tagname,DataType,HiEngineeringUnits,LoEngineeringUnits,StepValue
TAG2,SingleFloat,100,0,TRUE
[Data]
Tagname,TimeStamp,Value,DataQuality
TAG2,9/19/05 13:59:00.000,22,Good
TAG2,9/19/05 14:08:00.000,12,Good
TAG2,9/19/05 14:22:00.000,4,Good
```

Use the following query to retrieve the data using Average with a 30 minute interval

```
select * from ihrawdata where tagname=tag2 and timestamp >

'9/19/05 14:00:00' and timestamp <= '9/19/05 14:30:00' and

calculationmode=average and intervalmilliseconds=30m
```

You will see the following results.

| Time Stamp | Value | Quality |
|---|---|---|
| 14:30:00 | 12.53 | 100.00 |

The following table is another way to look at the data as values and durations.

| Point | Value | Duration (Seconds) |
|---|---|---|
| Point 1 | 22.00 | 480 (lab sampled at start) |
| Point 3 | 12 | 1320 |
| Point 4 | 4 | 480 |

The step value average would be:

```
((22.00 * 480) + (12 * 840) + (4 * 480)) / (480 + 840 + 480) = 12.53
```

The percent good is 100 since it was good the whole time.

The interpolated average is 12.24 because the first sample is different.

| Point | Value | Duration (Seconds) |
|---|---|---|
| Point 1 | 20.88 | 480 (lab sampled at start) |
| Point 3 | 12 | 1320 |
| Point 4 | 4 | 480 |

The lab average would be:

```
((20.88 * 480) + (12 * 840) + (4 * 480)) / (480 + 840 + 480) = 12.24
```

**Example: Raw sample at end time**

The point of this example is that if you have a raw sample on the interval end time then it is ignored because of the time weighting.

Copy and paste these lines into an empty CSV file and import the file with the File collector.

```
[Tags]
Tagname,DataType,HiEngineeringUnits,LoEngineeringUnits,StepValue
TAG5,SingleFloat,100,0,TRUE
[Data]
Tagname,TimeStamp,Value,DataQuality
TAG5,9/19/05 13:10:00.000,22,Good
TAG5,9/19/05 14:18:00.000,12,Good
TAG5,9/19/05 14:30:00.000,1,Good
```

Use the following query to retrieve the data using Average with a 30 minute interval.

```
select * from ihrawdata where tagname=tag5 and timestamp >
'9/19/05 14:00:00' and timestamp <= '9/19/05 14:30:00' and
calculationmode=average and intervalmilliseconds=30m
```

You will see the following results.

| Time Stamp | Value |
|---|---|
| | |
| 14:30:0 | 18.00 |

See that the last raw sample is ignored

| Point | Value | Duration (Seconds) |
|---|---|---|
| Point 1 | 22.00 | 1080 (lab sampled at start) |
| Point 3 | 12.00 | 720 |
| Point 4 | 1.00 | 0 |

The lab sampled average is:

```
((22.00 * 1080) + (12 * 720) + (1 * 0)) / (1080 + 720) = 18.0
```

- The interpolated average gives 13.59 because of the different interpolated value at interval start.
- The percent good is 100 since it was good the whole time.

**Example: No raw samples in interval**

This case shows the biggest difference between averages of step value and non step value tags. In this case we lab sample a value at the start time and that is the average.

Copy and paste these lines into an empty CSV file and import the file with the File collector.

```
[Tags]
Tagname,DataType,HiEngineeringUnits,LoEngineeringUnits,StepValue
TAG4,SingleFloat,100,0,TRUE
[Data]
Tagname,TimeStamp,Value,DataQuality
TAG4,9/19/05 13:55:00.000,99,Good
TAG4,9/19/05 14:40:00.000,10,Good
```

Use the following query to retrieve the data using Average.

```
select * from ihrawdata where tagname=tag4 and timestamp >
'9/19/05 14:00:00' and timestamp <= '9/19/05 14:30:00' and
calculationmode=average and intervalmilliseconds=30m
```

You will see the following results.

| Time Stamp | Value | Quality |
|---|---|---|
| 14:30:00 | 99 | 100 |

**Note:**
The single lab sampled value at interval start time is the average.

Retrieving the data when StepVa`lue=FALSE` gives the following:

| Time Stamp | Value | Quality |
|---|---|---|
| 14:30:00 | 89.11 | 100 |

**Note:**
The single interpolated sample at interval start time is the average of the interval.

## Comment Retrieval Mode

The Comment Retrieval Mode returns any comments or annotations that have been stored with the data between the start time and end time of the query.

However, some Sampling and Calculation modes use raw samples beyond the start and end time to interpolate a value. An average will interpolate a value at the start of each interval and this will likely use raw samples outside the interval.

To retrieve the comments from raw values that were used beyond the interval, you can define a registry key on a computer running the Data Archiver.

Create a DWORD value under:

```
HKEY_LOCAL_MACHINE\Software\Intellution, Inc.\iHistorian\Services\DataArchiver
```

- If you have the Data Archiver installed, the registry key should already exist and you are just adding a DWORD value.
- Set `CommentRetrievalMode` to 1.

**Note:**

- You do not have to restart the Archiver for the changes to the registry to take place. The changes to registry setting take effect immediately
- Raw data queries are not affected with this change.
- Any application can be used to query the data
- The Comment Retrieval Mode may result in many comments being returned for a query. Therefore, it is not recommended for users who want to plot the data via the Proficy Real Time Information Portal (RTIP) Chart as it may cause slower performance.

## Query Modifiers

Query Modifiers are used for retrieving data that has been stored in the archive. They are used along with sampling and calculation modes to get a specific set of data. The following sections describe the Query Modifiers in **Historian**.

- **ONLYGOOD**

  The ONLYGOOD modifier excludes bad and uncertain data quality values from retrieval and calculations. Use this modifier with any sampling or calculation mode but it is most useful with Raw and CurrentValue queries.

  All the calculation modes such as minimum or average exclude bad values by default, so this modifier is not required with those.

### Example 1:Demonstrating the Behavior

Import the following data to demonstrate the behavior of ONLYGOOD

```
[Tags]
Tagname,DataType,HiEngineeringUnits,LoEngineeringUnits
BADDQTAG,SingleFloat,60,0
[Data]
Tagname,TimeStamp,Value, DataQuality
BADDQTAG,12-Jul-2012 8:59:00.000,22.7,Good
BADDQTAG,12-Jul-2012 9:08:00.000,12.5,Bad
BADDQTAG,12-Jul-2012 9:14:00.000,7.0,Bad
BADDQTAG,12-Jul-2012 9:22:00.000,4.8,Good
```

### Example 2: Excluding bad data from raw data query

Without any query modifier, all raw samples are returned from a RawByTime query.

```
select timestamp,value,quality
from ihrawdata
where tagname = BADDQTAG and samplingmode=Rawbytime and timestamp < now
```

| Time Stamp | Value | Quality |
|---|---|---|
| 7/12/2012 08:59:00 | 22.7000000 | Good, NonSpecific |
| 7/12/201209:08:00 | 12.5000000 | Bad, NonSpecific |
| 7/12/201209:14:00 | 7.0000000 | Bad, NonSpecific |
| 7/12/201209:22:00 | 4.8000000 | Good, NonSpecific |

**Note:**
The above results have both good and bad samples:

Now by using the ONLYGOOD modifier, you can exclude the bad quality values:

```
select timestamp,value,quality
from ihrawdata
where tagname = BADDQTAG and samplingmode=Rawbytime and timestamp < now and
 criteriastring="#ONLYGOOD"
timestamp             value              quality
```

| Time Stamp | Value | Quality |
|---|---|---|
| 7/12/2012 08:59:00 | 22.7000000 | Good, NonSpecific |
| 7/12/201209:22:00 | 4.8000000 | Good, NonSpecific |

**Note:**
Only the good samples have been retrieved.

**Example 3: Retrieving the last known value**

**Value**

You can use the ONLYGOOD query modifier to show the last known good value for a tag. If the collector loses communication with the data source or has shut down, you can ignore the bad data that is logged.

The following examples demonstrate the ways to retrieve the last known values:

```
[Tags]
Tagname,DataType,HiEngineeringUnits,LoEngineeringUnit
EXAMPLETAG,SingleInteger,60,0
[Data]
Tagname,TimeStamp,Value,DataQuality
EXAMPLETAG,06-Aug-2012 8:59:00.000,2,Good
EXAMPLETAG,06-Aug-2012 9:02:00.000,0,Bad
```

Without any query modifier, the newest raw sample is returned in a current value query as retrieved with the following query.

```
select timestamp,value,quality
from ihrawdata
where tagname = EXAMPLETAG and samplingmode=CurrentValue
```

| Time Stamp | Value | Quality |
|---|---|---|
| 8/6/201209:02:00 | Bad | NonSpecific |

The bad data could be a communication error or collector shutdown marker

When the ONLYGOOD modifier is used, the bad quality value is ignored and last known good value is returned as per the query here.

```
select timestamp,value,quality

from ihrawdata

where tagname = EXAMPLETAG and samplingmode=CurrentValue and

 criteriastring="#ONLYGOOD"
```

**Note:**
Only the Good value has been retrieved as following. timestamp value quality.

| Time Stamp | Value | Quality |
|---|---|---|
| 8/6/201208:59:00 | 2 Good | NonSpecific |

**Anticipated Usage**

You can use the ONLYGOOD modifier to exclude end of collection markers but understand that it excludes all bad data, even communication errors, and out of range errors.

If you want to bring data into Microsoft Excel for further analysis, you can use ONLYGOOD so that good values are brought into a spreadsheet.

- **INCLUDEREPLACED**

Normally, when you query raw data from Historian, any values that have been replaced with a different value for the same timestamp are not returned. The INCLUDEREPLACED modifier helps you to indicate that you want replaced values to be returned, in addition to the currently retrievable data. However, you cannot query only the replaced data and the retrievable values that have replaced. You can query all currently visible data and get the data that has been replaced.

This modifier is only useful with rawbytime or rawbynumber retrieval. Do not use it with any other sampling or calculation mode.

**Example**

Import this data to demonstrate the behavior of the INCLUDEDELETED query modifier.

```
[Tags]
Tagname,DataType,HiEngineeringUnits,LoEngineeringUnits
DELETEDDATA,SingleInteger,60,0
[Data]
Tagname,TimeStamp,Value,DataQuality
DELETEDDATA,06-Aug-2012 9:01:00.000,1,Good
DELETEDDATA,06-Aug-2012 9:02:00.000,2,Good
DELETEDDATA,06-Aug-2012 9:04:00.000,4,Good
```

Delete the raw sample at 9:02 and query the raw data without any modifier and you will get only the non-deleted values.

Run the following query:

```
select timestamp,value,quality
from ihrawdata
where tagname = DELETEDDATA and samplingmode=RawByTime and timestamp < now
```

| Time Stamp | Value | Quality |
|---|---|---|
| 8/6/2012 09:01:00 | 1 | Good NonSpecific |
| 8/6/2012 09:04:00 | 4 | Good NonSpecific |

Query with the INCLUDEDELETED modifier and you will get the deleted sample together with the non-deleted data.

```
select timestamp,value,quality
from ihrawdata
where tagname = DELETEDDATA and samplingmode=RawByTime and timestamp < now and
 criteriastring="#INCLUDEDELETED"
```

| Time Stamp | Value | Quality |
|---|---|---|
| 8/6/2012 09:01:00 | 1 Good | NonSpecific |
| 8/6/2012 09:02:00 | 2 Good | NonSpecific |
| 8/6/2012 09:04:00 | 4 Good | NonSpecific |

**Anticipated Usage**

The INCLUDEDELETED modifier can be used to detect and recover deleted data. Perform a query without the modifier and with the modifier and compare the results. You will detect the deleted samples. You can also determine the deleted samples with a User API program.

**ONLYIFCONNECTED and ONLYIFUPTODATE**

The ONLYIFCONNECTED and ONLYIFUPTODATE modifiers can be used on any sampling or calculation mode to retrieve bad data if the collector is not currently connected and sending data to the archiver. The bad data is not stored in the IHA file but is only returned in the query. If the collector reconnects and flushes data and you run the query again, the actual stored data is returned in the following situations:

- Collector loses connection to the Archiver
- Collector crashes
- Collector compression is used and no value exceeds the deadband

Any data query will return the last known value repeated till the current time with the quality of data as good. But the information could have changed in the real world and has yet to reach the archiver.

Repeating the last known good value data can be misleading. Data should be returned as bad quality if no data is coming in from a collector.

The Data Archiver keeps track of the newest raw sample received for any tag for each collector. If no data is received for any tag, the collector is considered to be idle. If the collector is idle for more than 270 seconds, then either the data is heavily compressed or the collector is crashed or has lost connection. The collector idle time defaults to 270 seconds and this current setting appears in the *dataarchiver.shw* file. You can change the value using an SDK program. The setting applies to all collectors.

Use the ONLYIFUPTODATE modifier to return bad data from the time of the newest raw sample to the current time. However, if there is an unlikely chance that all tags are heavily compressed, then use the ONLYIFCONNECTED modifier. The difference in the behavior of the two modifiers is given in the following examples:

When you add an ONLYIFCONNECTED or ONLYIFUPTODATE modifier to the query, and the collector is disconnected from the archiver, bad values are returned from the time of the disconnect until the current time. Queries of data before the disconnect time are unaffected.

**Note:**

- The ONLYIFCONNECTED and ONLYIFUPTODATE modifiers are applicable to tags that are collected by data collectors.
- For raw by number, if the number of samples collected are greater or equal to the number of samples, bad data quality is not added.
- For raw by time, if the endtime is less than maximum data received time, bad data quality is not added.
- For raw by number backward, bad data quality is added at the beginning.

**Example 1: Using ONLYIFCONNECTED to detect connection loss**

To demonstrate the behavior of ONLYIFCONNECTED, you need to query data currently being collected.

1. Configure the Simulation collector to collect any tag once per second with no compression. For example, collect the simulation RAMP tag.
2. Let the collector run for at least 5 minutes of collection.
3. Disconnect the collector but leave it running. In this test, the collector was disconnected at 20:55:00.
4. After about 5 minutes, query the data without ONLYIFCONNECTED and the last known value repeated with good quality to the current time, even though the collector is not connected.

```
set starttime='22-Aug-2012 20:53:00',endtime='now

select timestamp,value,quality

from ihrawdata

where tagname = RAMP and samplingmode=Interpolated and intervalmilliseconds=5s order by

 timestamp asc
```

| Time Stamp | Value | Quality |
|---|---|---|
| 8/22/2012 20:54:55 | 166.666666666667 | 100.0000000 |
| 8/22/2012 20:55:00 | 0.000000000000 | 100.0000000 |
| 8/22/2012 20:55:05 | 166.666666666667 | 100.0000000 |
| 8/22/2012 20:55:10 | 333.333333333333 | 100.0000000 |

| Time Stamp | Value | Quality |
|---|---|---|
| 8/22/2012 20:55:15 | 500.000000000000 | 100.0000000 |
| 8/22/2012 20:55:20 | 533.333333333333 | 100.0000000 |
| 8/22/2012 20:55:25 | 533.333333333333 | 100.0000000 |
| 8/22/2012 20:55:30 | 533.333333333333 | 100.0000000 |
| 8/22/2012 20:55:35 | 533.333333333333 | 100.0000000 |

5. Run the query again with ONLYIFCONNECTED and the data is marked bad at the time of the collector disconnect:

```
set starttime='22-Aug-2012 20:53:00',endtime='now

select timestamp,value,quality

from ihrawdata

where tagname = RAMP and samplingmode=Interpolated and intervalmilliseconds=5s and

 criteriastring=#onlyifconnect
```

| Time Stamp | Value | Quality |
|---|---|---|
| 8/22/2012 20:54:55 | 166.666666666667 | 100.0000000 |
| 8/22/2012 20:55:00 | 0.000000000000 | 100.0000000 |
| 8/22/2012 20:55:05 | 166.666666666667 | 100.0000000 |
| 8/22/2012 20:55:10 | 333.333333333333 | 100.0000000 |
| 8/22/2012 20:55:15 | 500.000000000000 | 100.0000000 |
| 8/22/2012 20:55:20 | 0.000000000000 | 0.0000000 |
| 8/22/2012 20:55:25 | 0.000000000000 | 0.0000000 |
| 8/22/2012 20:55:30 | 0.000000000000 | 0.0000000 |
| 8/22/2012 20:55:35 | 0.000000000000 | 0.0000000 |

6. Reconnect the collector and once the collector reconnects and flushes its buffered data run the query again with ONLYIFCONNECTED and the period of bad data is filled in with ramping values:

| Time Stamp | Value | Quality |
|---|---|---|
| 8/22/2012 20:54:55 | 166.666666666667 | 100.0000000 |
| 8/22/2012 20:55:00 | 0.000000000000 | 100.0000000 |
| 8/22/2012 20:55:05 | 166.666666666667 | 100.0000000 |
| 8/22/2012 20:55:10 | 333.333333333333 | 100.0000000 |
| 8/22/2012 20:55:15 | 500.000000000000 | 100.0000000 |
| 8/22/2012 20:55:20 | 569.696969985962 | 100.0000000 |
| 8/22/2012 20:55:25 | 615.151515960693 | 100.0000000 |
| 8/22/2012 20:55:30 | 660.606061935425 | 100.0000000 |
| 8/22/2012 20:55:35 | 706.060606002808 | 100.0000000 |

**Example 2: Querying Compressed Data**

If all tags for a collector are compressed, then the newest raw sample across all tags can easily be older than 270 seconds even when the collector is connected to archiver. It is unlikely in a real system that a collector will send 0 raw samples for 270 seconds, but it is possible.

1. Use the simulation collector and collect the constant tag as 1 second polled with a small deadband such as 1. In the example below, the newest raw sample is at 17:27:31 and the current time is 5 minutes or more.
2. Query the data as interpolated with a 5 second interval and no modifier.

```
set starttime='23-Aug-2012 17:00:30',endtime='now,rowcount=0

select timestamp,value,quality

from ihrawdata

where tagname = CONSTANT and samplingmode=Interpolated and intervalmilliseconds=5s order by

 timestamp asc
```

| Time Stamp | Value | Quality |
|---|---|---|
| 8/23/20121 7:27:20 | 500.000000000000 | 100.0000000 |
| 8/23/20121 7:27:25 | 500.000000000000 | 100.0000000 |
| 8/23/20121 7:27:30 | 500.000000000000 | 100.0000000 |

| Time Stamp | Value | Quality |
|---|---|---|
| 8/23/20121 7:27:35 | 0.000000000000 | 100.0000000 |
| 8/23/20121 7:27:40 | 0.000000000000 | 100.0000000 |

**Note:**
The newest sample is repeated to the current time

3. Query with ONLYIFCONNECTED and you get the same results even when the newest raw sample is more than 270 seconds old. The data is old but the collector is currently connected.

```
set starttime='23-Aug-2012 17:00:30',endtime='now,rowcount=0

select timestamp,value,quality

from ihrawdata

where tagname = CONSTANT and samplingmode=Interpolated and intervalmilliseconds=5s and

 criteriastring=#onlyifcon
```

| Time Stamp | Value | Quality |
|---|---|---|
| 8/23/20121 7:27:20 | 500.000000000000 | 100.0000000 |
| 8/23/20121 7:27:25 | 500.000000000000 | 100.0000000 |
| 8/23/20121 7:27:30 | 500.000000000000 | 100.0000000 |
| 8/23/20121 7:27:35 | 500.000000000000 | 100.0000000 |
| 8/23/20121 7:27:40 | 500.000000000000 | 100.0000000 |

4. Query with ONLYIUPTODATE and the data is considered bad quality after the newest raw sample.

```
set starttime='23-Aug-2012 17:00:30',endtime='now,rowcount=0

select timestamp,value,quality

from ihrawdata

where tagname = CONSTANT and samplingmode=Interpolated and intervalmilliseconds=5s and

 criteriastring=#onlyifupt
```

| Time Stamp | Value | Quality |
|---|---|---|
| 8/23/20121 7:27:20 | 500.000000000000 | 100.0000000 |

| Time Stamp | Value | Quality |
|---|---|---|
| 8/23/20121 7:27:25 | 500.000000000000 | 100.0000000 |
| 8/23/20121 7:27:30 | 500.000000000000 | 100.0000000 |
| 8/23/20121 7:27:35 | 0.000000000000 | 0.0000000 |
| 8/23/20121 7:27:40 | 0.000000000000 | 0.0000000 |

**Note:**
If your collector can possibly have no data for any tag due to compression, use ONLYIFCONNECTED. Otherwise, if you want to detect data being old due to collector crash or disconnect, then use ONLYIFUPTODATE and optionally adjust the collector idle time.

**Anticipated Usage**

Use the ONLYIFCONNECTED and ONLYIFUPTODATE modifiers so that your trend lines stop plotting when the collector loses connection.

Use the ONLYIFCONNECTED and ONLYIFUPTODATE modifiers with CurrentValue retrieval so that the current value turns to bad quality if the collector is disconnected. This way you are not misled by looking at an outdated value that does not match the real world.

**ONLYRAW**

The ONLYRAW modifier retrieves only the raw stored samples. It does not add interpolated or lab sampled values at the beginning of each interval during calculated retrieval such as average or minimum or maximum.

Normally, a data query for minimum value will interpolate a value at the start of each interval and use that together with any raw samples to determine the minimum value in the interval. Interpolation is necessary because some intervals may not have any raw samples stored.

**Note:**
Use the ONLYRAW modifier with Calculation modes only, not with raw or sampled retrieval like interpolated modes.

**Example**

Import this data to demonstrate the behavior of the ONLYRAW query modifier.

```
[Tags]
Tagname,DataType,HiEngineeringUnits,LoEngineeringUnits
RAMPUP,SingleFloat,100,0
[Data]
Tagname,TimeStamp,Value,DataQuality
RAMPUP,06-Aug-2012 9:01:00.000,1,Good
RAMPUP,06-Aug-2012 9:02:00.000,2,Good
RAMPUP,06-Aug-2012 9:03:00.000,3,Good
RAMPUP,06-Aug-2012 9:04:00.000,4,Good
RAMPUP,06-Aug-2012 9:05:00.000,5,Good
RAMPUP,06-Aug-2012 9:06:00.000,6,Good
```

When you query the minimum without any modifier, you see that the minimum value may not be one of the stored values.

```
set starttime='06-Aug-2012 09:02:30',endtime='06-Aug-2012 09:05:30'
select timestamp,value,quality
from ihrawdata
where tagname = RAMPUP and samplingmode=Calculated and CalculationMode=minimum and numberofsamples=3
```

| Time Stamp | Value | Quality |
|---|---|---|
| 8/6/2012 09:03:30 | 2.500000000000 | 100.0000000 |
| 8/6/2012 09:04:30 | 3.500000000000 | 100.0000000 |
| 8/6/2012 09:05:30 | 4,500000000000 | 100.0000000 |

```
set starttime='06-Aug-2012 09:02:30',endtime='06-Aug-2012 09:05:30'
select timestamp,value,quality
from ihrawdata
where tagname = RAMPUP and samplingmode=Calculated and CalculationMode=minimum and numberofsamples=3
and criteriastring='#onlyraw'
```

| Time Stamp | Value | Quality |
|---|---|---|
| 8/6/2012 09:03:30 | 3.000000000000 | 100.0000000 |
| 8/6/2012 09:04:30 | 4.000000000000 | 100.0000000 |
| 8/6/2012 09:05:30 | 5.000000000000 | 100.0000000 |

#### Anticipated Usage

Use the ONLYRAW modifier to query the minimum and maximum values of stored data samples, similar to the RawAverage Calculation mode. A minimum or maximum of raw samples is more like doing a MIN() or MAX () in an Excel spreadsheet. Realize that if you use the ONLYRAW modifier, there may be intervals with no raw samples. The ONLYRAW modifier is useful for Calculation modes and not the Sampling modes.

### LABSAMPLING

The LABSAMPLING modifier affects the calculation modes that interpolate a value at the start of each interval. Instead of using interpolation, lab sampling is used. When querying highly compressed data you may have intervals with no raw samples stored. An average from 2 P.M to 6 P.M on a one hour interval will interpolate a value at 2 P.M., 3 P.M., 4 P.M, and 5 P.M and use those in addition to any stored samples to compute averages. When you specify LABSAMPLING, then lab sampling mode is used instead of interpolated sampling mode to determine the 2 P.M., 3 P.M., 4 P.M., and 5 P.M., values.

A lab sampled average would be used when querying a tag that never ramps but changes in a step pattern such as a state value or setpoint.

**Note:**
Use the LABSAMPLING modifier with calculation modes only, not raw or sampled retrieval like interpolated modes.

#### Example

Import this data to demonstrate the behavior of the LABSAMPLING query modifier.

```
[Tags]
Tagname,DataType,HiEngineeringUnits,LoEngineeringUnits
RAMPUP,SingleFloat,100,0
[Data]
Tagname,TimeStamp, Value,DataQuality
RAMPUP,06-Aug-2012 9:01:00.000,1,Good
RAMPUP,06-Aug-2012 9:02:00.000,2,Good
RAMPUP,06-Aug-2012 9:03:00.000,3,Good
RAMPUP,06-Aug-2012 9:04:00.000,4,Good
RAMPUP,06-Aug-2012 9:05:00.000,5,Good
RAMPUP,06-Aug-2012 9:06:00.000,6,Good
```

Run this query without a modifier to see the minimum values using the interpolated values:

```
set starttime='06-Aug-2012 09:02:30',endtime='06-Aug-2012 09:05:30'

select timestamp,value,quality

from ihrawdata

where tagname = RAMPUP and samplingmode=Calculated and CalculationMode=minimum and numberofsamples=3
```

| Time Stamp | Value | Quality |
|---|---|---|
| 8/6/2012 09:03:30 | 2.500000000000 | 100.0000000 |
| 8/6/2012 09:04:30 | 3.500000000000 | 100.0000000 |
| 8/6/2012 09:05:30 | 4.500000000000 | 100.0000000 |

The returned minimum values are stored values but sampled forward to each interval timestamp. This is the behavior of lab sampling and is applied here to calculated values.

**Anticipated Usage**

Use the LABSAMPLING modifier to query the minimum, maximum, and average values of tags that change in a step fashion and never ramp. For example, you may want to retrieve the minimum of a set point. This tag would change from one value directly to another without ramping. And the value may not change in a long period. A minimum should not return a value that ramps over a long period of time from one set point value to the next. The LABSAMPLING modifier is useful for Calculation modes and not the Sampling modes.

**ENUMNATIVEVALUE**

The ENUMNATIVEVALUE modifier retrieves the native, numeric values such as 1 or 2 instead of string values such as on/off for the data that has enumerated states associated with it.

**Note:**
You can use the ENUMNATIVEVALUE modifier with any sampling or calculation mode.

**Example**

Import this data to demonstrate the use of ENUMNATIVEVALUE:

```
[Tags]

Tagname,DataType,HiEngineeringUnits,LoEngineeringUnits

STATETAG,SingleInteger,60,0

[Data]
```

```
Tagname,TimeStamp,Value,DataQuality
STATETAG,06-Aug-2012 9:08:00.000,4,Good
STATETAG,06-Aug-2012 9:14:00.000,4,Good
STATETAG,06-Aug-2012 9:22:00.000,2,Good
```

Assume the tag has an enumerated set associated where 2=Stopped and 4=Running. When you do the interpolated query you get the string value:

```
set starttime='06-Aug-2012 09:10:00',endtime='06-Aug-2012 09:30:00'
select timestamp,value,quality
from ihrawdata
where tagname = STATETAG and samplingmode=Interpolated and numberofsamples=6
```

| Time Stamp | Value | Quality |
|---|---|---|
| 8/6/2012 09:13:20 | running | 100.0000000 |
| 8/6/2012 09:16:40 | running | 100.0000000 |
| 8/6/2012 09:20:00 | running | 100.0000000 |
| 8/6/2012 09:23:20 | stopped | 100.0000000 |
| 8/6/2012 09:26:40 | stopped | 100.0000000 |
| 8/6/2012 09:30:00 | stopped | 100.0000000 |

Using the ENUMNATIVEVALUE query modifier, you can get the numeric value suitable for plotting

```
set starttime='06-Aug-2012 09:10:00',endtime='06-Aug-2012 09:30:00'
select timestamp,value,quality
from ihrawdata
where tagname = STATETAG and samplingmode=Interpolated and numberofsamples=6 and
 criteriastring='#enumnativevalue'
```

| Time Stamp | Value | Quality |
|---|---|---|
| 8/6/2012 09:13:20 | 4 | 100.0000000 |
| 8/6/2012 09:16:40 | 4 | 100.0000000 |
| 8/6/2012 09:20:00 | 4 | 100.0000000 |
| 8/6/2012 09:23:20 | 2 | 100.0000000 |

| Time Stamp | Value | Quality |
|---|---|---|
| 8/6/2012 09:26:40 | 2 | 100.0000000 |
| 8/6/2012 09:30:00 | 2 | 100.0000000 |

**Note:**
For bad data, the values are returned as string values based on the Enumerated State table though the enumerative value is set to FALSE.

**Anticipated Usage**

Use the ENUMNATIVEVALUE query modifier to plot tags that use enumerated values. You can put the string value in a data link and put the native value in a chart.

**INCLUDEBAD**

The INCLUDEBAD modifier directs the Data Archiver to consider raw samples of bad data quality when computing calculation modes. Use INCLUDEBAD modifier to consider both good and bad quality values.

You can use the INCLUDEBAD modifier with any Sampling or Calculation mode only if you want to include bad quality data.

Use the INCLUDEBAD modifier only if you believe the bad quality data has meaningful values and are useful as input to calculations. Most bad data quality values do not have meaningful values; they show 0 or unpredictable numbers. But in some cases, if the data is being written using a user program instead of a collector, you can use this query modifier.

Bad data always has a sub quality such as Comm Error or Configuration Error. When you use the INCLUDEBAD modifier any end of collection raw samples or calculation error raw samples are still ignored because they are not process data, just data markers that are inserted by collectors.

**Example**

Import this data to demonstrate the behavior of the INCLUDEBAD query modifier:

```
[Tags]
Tagname,DataType,HiEngineeringUnits,LoEngineeringUnits
Tag1,SingleFloat,60,0
[Data]
Tagname,TimeStamp,Value,DataQuality
Tag1,07-05-2011 17:24:00,29.72,Bad
```

```
Tag1,07-05-2011 17:25:00,29.6,Good

Tag1,07-05-2011 17:26:00,29.55,Good

Tag1,07-05-2011 17:27:00,29.49,Bad

Tag1,07-05-2011 17:28:00,29.53,Bad
```

**Note:**
The given sample contains three bad quality data.

Run the following query

```
set starttime='07-05-2011 16:00:00', endtime='07-05-2011 21:00:00'

select timestamp,value,quality from ihrawdata where tagname like 'Tag1' and samplingMode=calculated and

 CalculationMode=count and Numberofsamples=1
```

| Time Stamp | Value | Quality |
|---|---|---|
| 7/5/201121:00:00 | 2.000000000000 | 100.0000000 |

The count is 2 because only good quality data is considered. If you want to consider bad quality use the INLCUDEBAD query modifier as given in the following example.

```
set starttime='07-05-2011 16:00:00', endtime='07-05-2011 21:00:00'

select timestamp,value,quality from ihrawdata where tagname like 'Tag1' and samplingMode=calculated and

CalculationMode=count and criteriastring="#INCLUDEBAD" and  Numberofsamples=1
```

| Time Stamp | Value | Quality |
|---|---|---|
| 7/5/201121:00:00 | 5.000000000000 | 100.0000000 |

**Note:**
When we use INCLUDEBAD query modifier all the values are considered and the count is 5.

**Anticipated Usage**

The INCLUDEBAD modifier can be used to force the Data Archiver to consider every raw sample collected from a field device while still excluding Proficy Historian collection markers or calculation errors and timeouts.

The INCLUDEBAD modifier is usually used if you are writing data with a custom program and not a collector and your program stores meaningful values with bad quality.

**FILTERINCLUDEBAD**

The FILTERINCLUDEBAD modifier directs the Data Archiver to consider the values of bad quality data when determining the time ranges that match the filter condition. This modifier is similar to the INCLUDEBAD but that modifier applies to the data tag and this modifier applies to the FilterTag.

You can use the FILTERINCLUDEBAD modifier if you are also using INCLUDEBAD because your application data of bad quality has meaningful values then you can also consider this modifier but, you do not need to use both modifiers at the same time.

**Example: Filtered Data Query containing Bad Quality Filter Values**

Import this data to demonstrate the behavior of the FILTERINCLUDEBAD query modifier:

```
[Tags]
Tagname,DataType,HiEngineeringUnits,LoEngineeringUnits
ExcelTag1,SingleFloat,60,0
[Data]
Tagname,TimeStamp,Value,DataQuality
ExcelTag1,07-05-2011 17:24:00,29.72,Bad
ExcelTag1,07-05-2011 17:25:00,29.6,Good
ExcelTag1,07-05-2011 17:26:00,29.55,Good
ExcelTag1,07-05-2011 17:27:00,29.49,Bad
ExcelTag1,07-05-2011 17:28:00,29.53,Bad
```

The given sample contains three bad quality data samples. Run the following query:

```
set starttime='07-05-2011 16:00:00', endtime='07-05-2011 21:00:00'
select timestamp,value,quality from ihrawdata where tagname=ExcelTag1 and samplingMode=Calculated and
calculationmode=rawtotal and FilterExpression ='ExcelTag1>29.5' and numberofsamples=1
```

In this query, we use a filtered expression where the filter condition is ExcelTag1>29.5, and the result is as follows because it adds the two good values to compute the RawTotal:

| Time Stamp | Value | Quality |
|---|---|---|
| 7/5/201121:00:00 | 59.149999618530 | 100.0000000 |

Bad quality data is not considered while filtering. If you want to consider bad quality data then use the FILTERINLCUDEBAD query modifier together with the INCLUDEBAD query modifier as in the following query:

```
set starttime='07-05-2011 16:00:00', endtime='07-05-2011 21:00:00'

select timestamp,value,quality from ihrawdata where tagname=ExcelTag1 and samplingMode=Calculated and

calculationmode=rawtotal and FilterExpression ='ExcelTag1>29.5' and numberofsamples=1 and

 CriteriaString='#FilterIncludeBad#IncludeBad'
```

The result is as follows

| Time Stamp | Value | Quality |
|---|---|---|
| 7/5/201121:00:00 | 118.399999618530 | 100.0000000 |

**Note:**
The value is 118 because all the values that are greater than 29.5 are added together, not just the good quality values.

**Anticipated Usage**

The FILTERINCLUDEBAD modifier can be used to force the Data Archiver to consider every raw sample collected from a field device when determining the time ranges while still excluding Historian injected end of collection markers or calculation errors and timeouts.

**USEMASTERFIELDTIME**

The USEMASTERFIELDTIME query modifier is used only for the MultiField tags. It returns the value of all the fields at the same timestamp of the master field time, in each interval returned.

The following are the points to remember while using the USEMASTERFIELDTIME query modifier:

1. In your user defined data type, you have to indicate which field is the master field. You can define a master field when you define the type.
2. When you use the USEMASTERFIELDTIME query modifier, the query returns raw values of all the field elements at the timestamp determined by the MasterField.
3. When you use the USEMASTERFIELDTIME query modifier in Excel Add-in, the percentage good value displayed will be incorrect. It is recommended to use this query modifier using APIs.
4. Only a few calculation modes are supported by the USEMASTERFIELDTIME query modifier. The supported calculation modes are:

- Minimum Value
- Maximum Value
- Minimum Time
- Maximum Time
- FirstRawValue
- FirstRawTime
- LastRawValue
- LastRawTime

The supported modes will examine the raw samples for the master field of a Multi Field tag. For each raw sample in the interval, the minimum or maximum or first or last sample is determined depending on the mode. The timestamp of that raw sample is the master field time.

For example you have a multi-field tag called `mytag` with 3 fields and field3 is the master field.

1. You do a LastRawValue query on `mytag` and pass the USEMASTERFIELDTIME query modifier.
2. The Data Archiver determines the last raw sample for mytag.field3 between 3pm and 4pm is at 3:42pm. That is the master field time for this interval. Each interval has a master field time.
3. The Data Archiver gets the values for field1 and field2 at 3:42 so now you have a value for all 3 fields at 3:42.

**Note:**
When there is no raw sample for a field in the given interval then there is no master time for that interval. Most of the calculation modes will then return a 0 Value and Quality Bad for that interval.

**Example**

Import this data to demonstrate the behavior of the USEMASTERFIELDTIME query modifier.

```
[Data]
Tagname,TimeStamp,Value,DataQuality
MUser1.F1,05-22-2013 14:15:00,4,Good
MUser1.F1,05-22-2013 14:15:01,7,Good
MUser1.F1,05-22-2013 14:15:02,9,Good
MUser1.F2,05-22-2013 14:15:00,241,Good
MUser1.F2,05-22-2013 14:15:01,171,Good
MUser1.F2,05-22-2013 14:15:02,191,Good
```

**Note:**
In this sample the MUser1 tag has two fields F1 and F2 and F2 is marked as the MasterField.

Run the following query:

```
set starttime = '5/22/2013 14:15:00', endtime = '5/22/2013 14:15:02'

select tagname, timestamp, value, quality from ihrawdata where tagname = 'MUser1' and

samplingmode = calculated and calculationmode = minimum and

criteriastring = '#USEMASTERFIELDTIME' and numberofsamples = 1
```

The output is as follows:

| Tag Name | Time Stamp | Value | Quality |
|---|---|---|---|
| MUser1.F1 | 05-22-201314:15:02 | 7 | 0.0000000 |
| MUser1.F2 | 05-22-201314:15:02 | 171.000000000000 | 100.0000000 |

Here the minimum value for the Master Field tag F2 is 171 at 14:15:01 timestamp. That is the master time. Then the master time is used to get the value of F1 at the same timestamp which is 7 and this is returned even as the minimum value of F1 is 4.

In a multi field tag it is possible that some fields may be NULL at a given timestamp. In this case if F1 was a NULL value at 14:15:01 you would get a value of null and bad quality.

**HONORENDTIME**

Normally, a query keeps searching through archives until the desired number of samples has been located, or until it gets to the first or last archive. However, there are cases where you would want to specify a time limit as well. For example, you may want to output the returned data for a RawByNumber query in a trend page, in which case there is no need to return data that would be off page.

In cases where you want to specify a time limit, you can do this by specifying an end time in your RawByNumber query and including the HONORENDTIME query modifier. Since RawByNumber has direction (backwards or forwards), the end time must be older than the start time for a backwards direction or newer than the start time for a forwards direction.

**Note:**
Use the HONORENDTIME modifier only with the RawByNumber sampling mode.

### Example using HONORENDTIME with RawByNumber Sampling Mode

Import this data to Historian:

```
[Tags]
Tagname,DataType,HiEngineeringUnits,LoEngineeringUnits
TAG1,SingleInteger,60,0
[Data]
Tagname,TimeStamp,Value,DataQuality
TAG1,09/18/2015 14:00:00.000,00,Good
TAG1,09/18/2015 14:05:00.000,5,Good
TAG1,09/18/2015 14:10:00.000,10,Good
TAG1,09/18/2015 14:15:00.000,15,Good
TAG1,09/18/2015 14:20:00.000,20,Good
TAG1,09/18/2015 14:25:00.000,25,Good
TAG1,09/18/2015 14:30:00.000,30,Good
TAG1,09/18/2015 14:35:00.000,35,Good
TAG1,09/18/2015 14:40:00.000,40,Good
TAG1,09/18/2015 14:45:00.000,45,Good
TAG1,09/18/2015 14:50:00.000,50,Good
TAG1,09/18/2015 14:55:00.000,55,Good
TAG1,09/18/2015 15:00:00.000,60,Good
```

#### Without HONORENDTIME Query Modifier

```
set starttime='9/18/2015 14:00:00',endtime='9/18/2015 14:15:00'
select Timestamp,Value,Quality from ihrawdata where tagname like TAG1 and
 samplingmode=rawbynumber and
direction=forwardand numberofsamples=6
```

The output is as follows:

| Time Stamp | Value | Quality |
|---|---|---|
| 9/18/2015 14:00:00 | 0 | Good NonSpecific |
| 9/18/2015 14:05:00 | 5 | Good NonSpecific |
| 9/18/2015 14:10:00 | 10 | Good NonSpecific |
| 9/18/2015 14:15:0 | 15 | Good NonSpecific |
| 9/18/2015 14:20:00 | 20 | Good NonSpecific |

| Time Stamp | Value | Quality |
|---|---|---|
| 9/18/2015 14:25:00 | 25 | Good NonSpecific |

In the above query, the endtime specified is ignored and 6 values are returned.

**With HONORENDTIME Query Modifier**

```
set starttime='9/18/2015 14:00:00',endtime='9/18/2015 14:15:00'

select TagName,Timestamp,Value,Quality from ihrawdata where tagname like TAG1 and

 samplingmode=rawbynumber and

direction=forward and numberofsamples=6 and criteriastring=#honorendtime
```

The output is as follows:

| Time Stamp | Value | Quality |
|---|---|---|
| 9/18/2015 14:00:00 | 0 | Good NonSpecific |
| 9/18/2015 14:05:00 | 5 | Good NonSpecific |
| 9/18/2015 14:10:00 | 10 | Good NonSpecific |
| 9/18/2015 14:15:0 | 15 | Good NonSpecific |

In the above query, the endtime specified is used and only 4 values are returned.

**Anticipated Usage**

Use the HONORENDTIME modifier when you would want to specify a time limit to a query. For example, you may want to output the returned data for a RawByNumber query in a trend page, in which case there is no need to return data that would be offpage.

**EXAMINEFEW**

Queries using calculation modes normally loop through every raw sample, between the given start time and end time, to compute the calculated values.

When using FirstRawValue, FirstRawTime, LastRawValue, and LastRawTime calculation modes, we can use only the raw sample near each interval boundary and achieve the same result. The EXAMINEFEW query modifier enables this. If you are using one of these calculation modes you may experience better read performance using the EXAMINEFEW query modifier.

**Note:**
Use the EXAMINEFEW query modifier only with FirstRawValue, FirstRawTime, LastRawValue, and LastRawTime calculation modes.

### Examples using EXAMINEFEW with FirstRawValue and FirstRawTime Calculation Modes

Import this data to Historian:

```
[Tags]
Tagname,DataType,HiEngineeringUnits,LoEngineeringUnits
Tag1,SingleFloat,60,0
[Data]
Tagname,TimeStamp,Value,DataQuality
Tag1,07-05-2011 17:24:00,29.72,Bad
Tag1,07-05-2011 17:25:00,29.6,Good
Tag1,07-05-2011 17:26:00,29.55,Good
Tag1,07-05-2011 17:27:00,29.49,Bad
Tag1,07-05-2011 17:28:00,29.53,Bad
Tag1,07-05-2011 17:29:00,29.58,Good
Tag1,07-05-2011 17:30:00,29.61,Bad
Tag1,07-05-2011 17:31:00,29.63,Bad
Tag1,07-05-2011 18:19:00,30,Good
Tag1,07-05-2011 18:20:00,29.96,Good
Tag1,07-05-2011 18:21:00,29.89,Good
Tag1,07-05-2011 18:22:00,29.84,Good
Tag1,07-05-2011 18:23:00,29.81,Bad
```

#### Using FirstRawValue Calculation Mode

```
set starttime='07-05-2011 16:00:00', endtime='07-05-2011 19:00:00'
select timestamp,value,quality from ihrawdata where tagname like 'Tag1' and
 samplingMode=Calculated and
CalculationMode=FirstRawValue and criteriastring="#EXAMINEFEW" and
 intervalmilliseconds=1h
```

The output is as follows:

| Time Stamp | Value | Quality |
|---|---|---|
| 07-05-2011 17:00:00 | 00.0000000 | 0.0000000 |

| Time Stamp | Value | Quality |
|---|---|---|
| 07-05-2011 18:00:00 | 29.6000000 | 100.0000000 |
| 07-05-2011 19:00:00 | 30.0 | 100.0000000 |

**Note:**
The EXAMINEFEW query modifier does not affect query results, but may improve read performance.

**Using FirstRawTime Calculation Mode**

```
set starttime='07-05-2011 16:00:00', endtime='07-05-2011 19:00:00'

select timestamp,value,quality from ihrawdata where tagname like 'Tag1' and

 samplingMode=Calculated and

CalculationMode=FirstRawTime and criteriastring="#EXAMINEFEW" and

 intervalmilliseconds=1h
```

The output is as follows:

| Time Stamp | Value | Quality |
|---|---|---|
| 07-05-2011 17:00:00 | 01-01-1970 05:30:00 | 0.0000000 |
| 07-05-2011 18:00:00 | 07-05-2011 17:25:00 | 100.0000000 |
| 07-05-2011 19:00:00 | 07-05-2011 18:20:00 | 100.0000000 |

**Note:**
The EXAMINEFEW query modifier does not affect query results, but may improve read performance.

**Anticipated Usage**

Using the EXAMINEFEW modifier is recommended when:

- The time interval is greater than 1 minute.
- The collection interval is greater than 1 second.
- The data node size is greater than the default 1400 bytes.
- The data type of the tags is String or Blob.

> **Note:**
> Query performance varies depending on all of the above factors.

**EXCLUDESTALE**

Stale tags are tags that have no new data samples within a specified period of time, and which have the potential to add to system overhead and slow down user queries.

The EXCLUDESTALE query modifier allows for exclusion of stale tags in data queries.

Unless permanently deleted, stale tags from the archiver are not removed but are simply marked as stale. Use the query without the EXCLUDESTALE query modifier to retrieve the sample values.

> **Note:**
> Data is not returned for stale tags. An ihSTATUS_STALED_TAG error is returned instead.

**Example**

**Data**

In this example, the data below is for the last raw samples for Tag1 to Tag7:

```
Tag, Timestamp, Value

Tag1, 9/25/2015 10:00:00, 10

Tag2, 9/18/2015 10:00:00, 20

Tag3, 9/25/2015 10:00:00, 30

Tag4, 9/25/2015 10:00:00, 40

Tag5, 9/18/2015 10:00:00, 50Tag6, 9/25/2015 10:00:00, 60

Tag7, 9/18/2015 10:00:00, 70
```

**Further Assumptions**

- Current System Time: 9/26/2015 11:00:00
- Server configuration
  - Stale Period: 7 Days
  - Stale Period Check: 1 Day

In this case, Tag2, Tag5, and Tag7 were logged more than 7 days ago. They are therefore considered stale.

**Query without EXCLUDESTALE**

The following query is run at 9/26/2015 11:00:00:

```
set StartTime='9/17/2015 10:00:00',EndTime='9/26/2015 11:00:00'
select TagName,Timestamp,Value from ihrawdata where tagname like Tag* and
 Samplingmode=RawByTime and
CriteriaString="#ExcludeStale"
```

Output is returned for the following tags:

```
Tag, Timestamp, Value
Tag1, 9/25/2015 10:00:00, 10
Tag3, 9/25/2015 10:00:00, 30
Tag4, 9/25/2015 10:00:00, 40
Tag6, 9/25/2015 10:00:00, 60
```

In the above query, the stale tags (Tag 2, Tag5, and Tag7) are excluded from the results.

**Query with EXCLUDESTALE:**

The following query is run at 9/26/2015 11:00:00:

```
set StartTime='9/17/2015 10:00:00',EndTime='9/26/2015 11:00:00'
select TagName,Timestamp,Value from ihrawdata where tagname like Tag* and
 Samplingmode=RawByTime and
CriteriaString="#ExcludeStale"
```

Output is returned for the following tags:

```
Tag, Timestamp, Value
Tag1, 9/25/2015 10:00:00, 10
Tag3, 9/25/2015 10:00:00, 30
Tag4, 9/25/2015 10:00:00, 40
Tag6, 9/25/2015 10:00:00, 60
```

In the above query, the stale tags (Tag 2, Tag5, and Tag7) are excluded from the results.

**Anticipated Usage**

Stale tags have the potential to add to system overhead and slow down user queries, without adding new data. The EXCLUDESTALE modifier can be used to exclude such tags, thereby speeding up query time.

# Work with Data Stores from the Command Line

## Using the Command Line to Work with Data Stores

You can create new data stores or delete existing ones from the command line. Before you can work with data stores from the command line, you must follow these steps.

1. Stop all Historian services.
2. Open a command prompt, or shell, and switch to the directory where `ihConfigManager.exe` is located.
   By default, this folder is located in: `C:\Program Files\Proficy\Historian\x64\Server`.
3. Run `ihConfigureManager.exe` with the options you choose.

---

**Related reference**

Examples *(on page 1187)*

**Related information**

Creating a Data Store *(on page 1186)*

Deleting a Data Store *(on page 1186)*

## Creating a Data Store

To create a new data store, run the following:

```
ihConfigManager_x64.exe CreateDataStore DataStoreName [StoreType: Historical = 0, Scada = 1]

[DefaultStore: NotADefault=0, Default = 1] ["OptionalDataStoreDescription"]
```

Where `DataStoreName` is the name of the data store you want to create.

## Deleting a Data Store

To delete an existing data store, run the following:

```
ihConfigManager_x64.exe DeleteDataStore MyStore [NoConfirm]
```

Where:

- MyStore is the name of the data store you want to delete, and
- NoConfirm allows you delete a data store without a validation message appearing.

> **CAUTION:**
> Exercise extreme care in using the NoConfirm option, as it will delete an entire data store without a prompt. This is sometimes helpful in scripting, but it is a dangerous option otherwise.

### Examples

**Example 1: Create a new historical data store, but do not change my default data store**

```
C:\Program Files\Proficy\Historian\x64\Server\ihConfigManager_x64.exe CreateDataStore MyStore 0 0
```

**Example 2: Create a new historical data store and make it a default data store**

```
C:\Program Files\Proficy\Historian\x64\Server\ihConfigManager_x64.exe CreateDataStore MyStore 0 1 "This
 is my default historical store"
```

**Example 3: Delete a data store, and display a validation message after it is removed**

```
C:\Program Files\Proficy\Historian\x64\Server\ihConfigManager_x64.exe DeleteDataStore MyStore
```

**Example 4: Delete a data store, but suppress any validation messages**

```
C:\Program Files\Proficy\Historian\x64\Server\ihConfigManager_x64.exe DeleteDataStore MyStore NoConfirm
```

# Measuring Historian Performance

## About Measuring Performance of Proficy Historian

You can use the Windows Performance Counters to measure activity and performance of the Data Archivers. The Counters are more familiar to the system administrators and monitors the following Historian information.

- Read rates
- Side by side with non-Historian counters such as CPU usage or handle
- Thread counts

The following topics provide the objects and counters most useful for measuring and describing the system activity. The counters that are specified in the following topics are a subset of all the available counters.

**Note:**

- The Historian Advanced Topics documentation is not a replacement of the documentation for the full set of counters. Examples contained in each topic of this documentation use the counters to produce other measurements. Sometimes, the measurement number that you want is not exposed as a single counter. However, the number can be a combination of counters or a comparison of two counters.
- The counters only describe the behavior of the Data Archiver. For more information about troubleshooting and optimizing performance using the Historian and Windows counters, refer to the appropriate Historian documentation.

## About the Proficy Historian Overview Objects

The Overview objects are the counters that measure the samples collected and sent by the Data Archiver. You cannot use the counters to perform the following actions:

- Measure the performance of a specific read.
- Track the reads of a specific client or program.

The Overview objects are preferred to measure and describe a system. It is calculated as the sum total of the numbers in each instance of a data store. After you understand the Overview object, you can identify the most active data store by using the associated counters.

The performance counters are more useful because:

- You cannot access the read rates in the administrator UI.
- The write rate is updated only once in a minute. The counters are updated in real time making it much easier to see exactly when a problem began.
- The administrator UI shows only the data in the last 10 minutes but the counters are displayed over a longer time period to locate active times.
- You can access the counters in relation to the non-historian counters in the same trend.
- The counters are accessible when you cannot access the administrator UI due to performance or security reasons.

The reads vary based on the load on the Data Archiver.

**Note:**
The load on the Data Archiver does not depend only on the number of read calls. The load increases with the increased number of tags, archives, and the raw samples. You can monitor some of these activities using the counters.

You can use the Overview object to measure the following:

- Number of samples examined internally with respect to the number of samples returned to a user at a given time
- Inconsistency of reads and writes in a day, week, or month
- The number of out-of-order writes during a given time range
- The average number of samples examined per read call

| Counter Name | Description |
|---|---|
| Read Rate (Calls/min) | The number of user or program initiated read calls processed over the last minute |
| Read Raw Rate (Samp/min) | The number of raw data samples examined internally over the last minute in response to read calls |
| Read Samp Rate (Samp/min) | The number of raw data samples returned to external programs over the last minute in response to read calls |

**Note:**
The counts and rates provided in the above table are generic across the Data Archiver. The counts and rates do not provide detailed information, such as the reason of the time taken by a read (for example, 8 seconds) and the activities where the time was spent. However, the counts and rates can describe the reason of a scenario, when the same read criteria takes different time frames in two different days. If there are more reads or writes happening in the Data Archiver, the read criteria takes more time.

### Comparing Read Raw Rate and Read Samp Rate

Run a query for a month average of 200 tags that have collected data in every second. Assume that you stored the data in one day archives. The Data Archiver has to examine a large number (200 tags x 60 seconds x 24 hours x 30 days) of raw samples that are spread across 30 one day archives to produce only

200 returned samples. If the query is run in 1 minute without any error, the Read Raw Rate value is a large number and the Read Samp Rate value is 200.

Run the same query with samplingmode Raw By Time. The Read Raw Rate shows the same value because the same number of raw samples were examined. As the query results returned to the caller, the Read Samp Rate = Read Raw Rate.

**Note:**
The number of archives examined is not reflected for the counter. You will get the same Read Raw Rate and Read Samp Rate if you have a 30 days archive instead of a 31 days archive.

You cannot only look at the number of write calls with reads. A write can have samples for multiple tags and the timestamps on the data can affect the number of archives accessed by a write. A collector can typically write data for all its tags, but with the same timestamp and the write call can access the same archive. A migration program can write two years of data for a tag, which can access many archives.

The following table provides the counters that have a rate over the last minute. These counters describe the data write activity in the Data Archiver.

| Counter Name | Description |
|---|---|
| Write Rate (Average) | The number of raw samples received from the external programs in the last minute |
| Write Rate (Max) | The highest number of Write Rates (average) after starting the Data Archiver |

The following table provides the total counters after starting the Data Archiver. If the Data Archiver runs for a long time, the counters are set to zero.

| Counter Name | Description |
|---|---|
| Writes (Expensive) | The total number of raw samples that are expensive writes after the Data Archiver started |
| Writes (Total Failed) | The total number of data samples that failed to be stored after the Data Archiver started |
| Writes (Total) | The total number data samples stored to IHA files after the Data Archiver started |

| Counter Name | Description |
|---|---|
| Writes (Total OutOfOrder) | The total number of data samples written out of time order after the Data Archiver started. The number only includes the successful writes, and performs slower than when the data is in time order. |

**Note:**
Although some counters are rates and some are totals, all the counters are in units of data samples.

### Comparing the number of Raw Samples Read and Written

The Write Rate (Average) is the write equivalent to the Read Samp Rate. You can compare the two counters to see if more reads or writes per minute are created in your Data Archiver.

### Understanding the varying load on the Data Archiver

Trend the Write Rate (Average) and Read Samp Rate over a 24 hour period. You may see certain times of the day where the load varies, such as when reports are run, or a collector has a store and forward flush, or data is recalculated with Calculation collector. Access the data available for a month. A system used for compliance or billing will have a very low read rate until you run the report till the end of month. Compare the value to a system used for real time, auto updating trending. That system will have a more consistent read load throughout the month.

### Calculating the rate of out of order writes during a given time range

Out of order data writes are only exposed as a count, not a rate.

You can compute the number of out of order writes between a specific time (for example, between 3:15pm and 3:25pm) by getting the value of Total Out of Order at each timestamp and subtracting the value. You can convert the value to a rate per minute by dividing the value by 10 minutes.

The measurement is necessary because there are occurrences of out of order data in many systems. There is a base rate of out of order data for the system. If the system has intermittent changes in write performance, you can calculate the out of order rate during those times and compare the data to the base rate.

### Calculating samples examined per read

As both the number of read calls and number of samples examined are exposed, you can divide to get the number of samples examined per read. In some systems, the number is near one, which indicates many small reads while the Calculation collector does many current value reads that examine one sample and return it. The samples per read will also be one, if you query raw data, such as when replicating data. An analytic program can summarize the data into 5 minute averages. For one second uncompressed data, the value is 300 samples examined per read. The number is an overall system wide number so it will not be useful to troubleshoot one read.

## About Proficy Historian Message Queue Object

In any server software, there will be a number of queues. Most of the time, and all the queues should ideally have 0 items. This implies that the server is keeping up with the workload. The read and write counters of the overview object tell you how many read and write operations were performed. However, the queue counters can tell you how many actions are expected to happen, and if the user had to wait for a response.

Measuring the system performance through queues is an excellent way to determine if the server has reached the steady state performance limit. It can also tell you if the usage comes in bursts and needs to be mores spread out over time.

When using the queues for measurement, you should think about what are the “items” on that queue. The “items” or “messages” here are read calls or write calls. One read call can have multiple tag names and one write call can have multiple data samples.

There are 3 queue instances exposed by counters

- Write Queue: Data writes from collectors and non-collectors.
- Read Queue: Anything for data that is not a write. It is not just data reads, it can also be tag browses.
- Msgs Queue: Anything other than read queue and write queue. You can practically ignore this queue as it is only a tiny part of the activity and it is not considered in this document.

You can get basic or very detailed information from the queue counters. At a basic level, if the queues are non-zero at a point in time, you are doing too much work at that point in time. If your queues are always non zero, then you are always expecting too much and have reached your performance limit.

Use the Queue Counters on the Read and Write queues to measure the following parameters as explained in the sections that follow:

- Last Read time vs Average Read Time
- Variability of the current queue counts
- Variability of the processed rate of read or write queue
- Number of samples per write

### Basic Queue Counters

These counters represent concepts that apply to any queue usage in any server software. There is a set of these for the read queue and set for the write queue.

| Counter Name | Description |
|---|---|
| Count (Max) | The highest number reached by the Count (Total). |
| Count (Total) | Number of messages currently on the queue. |
| Processed Count | Number of messages processed from the queue since Data Archiver startup. This number will wrap around and reset to zero if the Data Archiver runs for a long time. |
| Processed Rate (msg/min) | Number of messages processed from the queue in the last minute. |
| Processing Time (Ave) | Average time (in milliseconds) since the Data Archiver startup to process a message. |
| Processing Time (Last) | Time (in milliseconds) to process the most recently processed message. |
| Processing Time (Max) | Highest number the Processing Time (Last) reached since the Data Archiver startup. |
| Recv Count (msgs) | Number of messages received into the queue since the Data Archiver startup. |
| Recv Rate (msgs/min) | Rate at which messages are received in the last minute. |

If your Processed Rate (msgs/min) is more than your Recv Rate (msgs/min), then your Count (Total) will be zero as the Data Archiver will be keeping up with the incoming requests.

The current value of these counters in report view is displayed at all times in the Performance Monitor. You can log these counters to a Performance Monitor group file so that the times can be matched up with periods of slow performance.

### Detailed Queue Counters

These counters require a detailed understanding of how the queues are used.

There is no single read or write queue in memory. They are a virtual queue that is the sum total of all the client queues. Each connection from a client uses a socket. Each socket is monitored by a thread called a client thread. A queue is used between one client thread and the pool of threads that access the IHA files. This can be called as a client queue. No client thread goes directly to the IHA files. There are a fixed number of threads that monitor all client queues and read and write the IHA files.

A default system has one write thread and four read threads. You may have 20 collectors and 35 clients connected to the data archiver, that is, 20+35=55 client threads. That is, 55 x 2 = 110 client queues as each client thread has one read and one write queue.

The four read threads will monitor the 55 client read queues, most of which are empty most of the time. The one write thread monitors the 55 client write queues.

The Count (Total) on the Read Queue instance or the Write Queue instance is the sum total of all the items on the 55 read queues.

| Counter Name | Description |
|---|---|
| Threads | Number of configured threads that go to the IHAs. This number will not change at runtime. It defaults to one write thread and four read threads. |
| Threads Working | Number of configured queue processing worker threads that are currently working on processing a message. If there is not much work to do, there will be idle threads and which will be much less than the Threads counter, possibly zero. |
| Time In Queue (Ave) | The average time since the Data Archiver startup of the “Time In Queue (Last)”. |
| Time In Queue (Last) | Time (in milliseconds) that the last message wait-ed in the queue before a thread started processing |

| Counter Name | Description |
|---|---|
| | it. This should be near zero, meaning the archiver is keeping up with the requests and writes. |
| Time In Queue (Max) | The max time since the Data Archiver startup of the Time In Queue (Last). |
| ClientQueues with Msgs | The number of client queues with messages on them. In the previous example, this is how many of the 55 read client queues have at least one item on it. It doesn't matter how many items are on the client queue, only that it has at least one item. The number would be between 0 and 55.<br><br>This number gives some idea about how balanced the incoming load is and how balanced the servicing of the clients is. You don't want any single client doing too many reads or write causing other clients to have to wait. |

The time to process one read or one write would be the Time In Queue (Last) + the Processing Time (Last). But these are not visible as these are overall system wide counters, and not the way to troubleshoot one read or one client. The Time in Queue (Last) increases when the Threads Working equals Threads meaning all threads are busy.

**Example: Comparing current to average processing time**

Every system is different and has its own "normal" data rate. You can measure it if your current rate is above or below normal. To determine if the Recv Rate or Processed Rate is above or below normal, you must look at the number over a longer period of time, maybe 1 hour or 24 hours.

To determine if the processing time is taking longer than normal, you can trend the Processing Time (Last) to the Processing Time (Average) at the same time range. One line will be above the other to show if the range is above or below normal.

**Example: Measuring the variability of Queue Count Total**

This demonstrates that the Count (Total) can change. The number will change based on the Recv Count and the Processed Count.

The Write Queue Recv Rate is usually consistent. But you may see the Write Queue Recv Rate increase during a Store and Forward flush of a collector. The Write Queue Processed Count will vary more, and that will cause the Write Queue Count (Total) to vary as well. Consider an archive backup done at midnight each day. During a backup, the writes have to stop. The Write Queue Recv Rate will stay the same because collectors are still writing. The Processed Count will be zero during the backup so the Write Count (Total) will grow.

The same happens if there are long reads happening. If there are any reads, then the writes will have to wait and the Write Count (Total) will grow. But the Overview object Read Raw Rate should be busy, indicating the Data Archiver is busy doing some work, but not the writes.

If the writes are out of time order, the exact same number and bundle size of the raw samples can take longer to write. The exact same number of raw samples can take longer to read if there are cache misses and the data archiver does file I/O.

Reads are unlike writes because collectors will keep sending writes, even if they don't get responses. A client that does a read will wait for the response before sending the next read. The reads will not queue up in the Data Archiver. In general, the Read Queue Count (Total) will not grow as high as the Write Queue Count (Total) unless you have many read clients.

You can measure how much your Read and Write Queue Count (Total) vary over a 24 hour period, and understand that Count (Total) variability is caused by the variability of the Recv Rate and Processed Rate. The variability of those is caused by the variability of the sizes of the reads and writes combined with whatever else is happening on the machine.

**Example: Computing the number of samples per write**

The Overview object has a Read Calls counter but does not have a Write calls counter. You don't know the number of write calls nor can you compute a number of samples per write call. But, since one Write Queue Recv Count is one write call, you can use that number.

## About Proficy Historian Cache Object

Caching is used in many kinds of server software. You may have a basic idea of the concept and terminology of caching and just need to know how Historian makes use of a cache to give improved performance. The Historian Data Archiver is used to store and retrieve data from gigabytes of archives on disk. All those raw samples can not be kept in memory. For performance reasons, the Data Archiver will attempt to keep the most recently used information in memory. Cache hits avoid file I/O which is the number one negative performance factor in any server software.

As with multiple queues, there are multiple caches in the Data Archiver, each holding a different type of object. As with “items” in queues you want to understand what “objects” are in a cache. One Read Call in the overview object becomes one Recv Count in the Queue object which becomes one or more cache hits or misses in the Archive Data Cache. This is because one read may span the raw samples stored in multiple data nodes. Some of those data nodes may be in cache and some may not.

There are four caches within the Data Archiver: ArchiveDataCache, ArchiveIndexCache, ArchiveTagCache, and ConfigTagCache. You can ignore three of them and only monitor the Archive Data Cache. These are raw samples. So, this cache is the simplest to understand and has the biggest effect on performance.

The Archive Data Cache starts empty at Data Archiver startup and fills as data is read and written.

Cache counters, like Queue counts are best viewed as current values in the report view of Performance monitor. These are displayed on the Archive Data Cache instance.

| Counter Name | Description |
|---|---|
| Hits | When a program is queried or re-queried a tag and time range, and it was found in the cache. |
| Misses | Data reads where the requested information was either never in cache or had to be removed to make room for more recently accessed data. |
| Hit Percentage | Hits divided by Misses expressed as a percent. A high percentage means most data requests are being satisfied without having to access the disk. |

The objects in the Archive Data Cache are the data nodes. One data node is about 250 consecutive raw samples for one tag.

| Counter Name | Description |
|---|---|
| Obj Count | Number of objects (data nodes containing raw samples) in the cache. |
| Num Adds | Total number objects added to cache since Data Archiver startup. This number will always be increasing as new data is collected and queried. |
| Num Deletes | Total number of objects deleted from cache. Deletes will not happen until the cache has reached its maximum size. |

| Counter Name | Description |
|---|---|
| Size (MB) | The amount of memory used by the cache to contain the raw samples. |

Possible uses of the counters are demonstrated in the sections that follow.

**Example: Computing the cache hits for a specific time range**

All the counters are numbers since Data Archiver startup, which means it is hard to detect a period of time that had many cache misses. If you know that the read was run at 4pm and took 1 minute, you can get the hits and miss counts at 3:59 and 4:02 and subtract them to know what the hit percentage was at the time the read was done. This is more useful than the hit percentage since startup. When subtracting, verify if the counter had rollover and went back to zero.

**Example: Best Case Archive Data Cache hit percentage**

Run the exact same SQL query 10 times with fixed start and end times. Your hits would be nine and misses would be one, that is, the first read. This is a cache hit percent of 90%. If you keep doing the read you will keep hitting the same raw samples in the same data nodes.

You must have an auto updating chart that always shows the data up to current time. You will have a high but not 100% cache hit percentage. This is because, as new data is added, you will have one cache miss accessing that newly created data node.

**Example: Diagnosing Data Archiver memory growth due to cache**

The overall Data Archiver memory usage consists of multiple kinds of objects. But you can monitor the memory usage due to caching in detail.

There are memory usage numbers on the cache and another way to do it is look at the object count in the cache. If Data Archiver Virtual Memory use is increasing, look at the Object Count over the same time period to see if it is also increasing.

**Example: Removing items from cache to limit memory usage**

There is no maximum reserved size for the cache. If adding more objects would put you past the configured Archiver Memory Usage, then adding one object will delete another object. Or, if the archiver memory is used for non-cache reasons like large tag browses, then the Data archiver cache will remove items to meet the target memory usage.

**Example: Monitoring the size in bytes of the cache**

Configure the Archiver Memory Size (MB) in the Admin UI to 100 meg and in the Report View of Performance Monitor look at the Size(MB) counter of the ArchiveDataCache instance. It is zero. Now change the Archiver Memory Size to 1700 and restart the data Archiver. The number is still zero.

This is because the counter measures how much space the cache is currently using, not a configured size nor maximum size. If you start reading and writing data, the Size (MB) will grow.

# Chapter 7. Historian Alarms and Events

## Overview of Alarms and Events

Using Historian, you can archive alarms and events data from any OPC-compliant alarms and events server.

**Alarms:** Alarms are generally defined as tags going into an abnormal condition. For example, you can set an alarm on a boiler when it reaches a specified temperature. Alarms usually have a well-defined life cycle, which is defined by the individual data sources from which the alarms data is collected. They enter an alarm state, are generally acknowledged, then return to normal.

To collect alarms and events data, you can use any of the following collectors:

- iFIX Alarms and Events collector
- OPC Classic Alarms and Events collector

Historian handles alarm data in two ways. You can view the entire Alarm as a single record that contains all information about the alarm, or you can view the Alarm History, which shows the transitions of the alarm as individual records.

**Events:** Events are generally defined as activities in a system that occur once only. For example, a user logging on to a device is an event. When viewing this data in Historian, each event is returned as a record.

**Note:**
If you have configured Historian to use the SQL Server, you must ensure that both Historian and SQL Servers have write access to Default Archive Path and Default Backup Path. (By default, Archive and Backup path is `C:\Proficy Historian Data\Archives\`.) Also, make sure that the SQL service user account has permission to create a new database in the SQL Server.

## Alarms and Events Requirements

To generate alarms and events data, you can use any OPC-compliant alarms and events server such as:

- CIMPLICITY
- iFIX

To collect alarms and events data, install collectors *(on page 142)*, and then add any of the following collector instances:

- iFIX Alarms and Events collector *(on page 505)*
- OPC Classic Alarms and Events collector *(on page 532)*

To store alarms and events data, install Historian Alarms and Events *(on page 140)*. During the installation, provide the credentials of Microsoft SQL server 2012 or later.

To view and analyze alarms and events data, you can use any of the following applications:

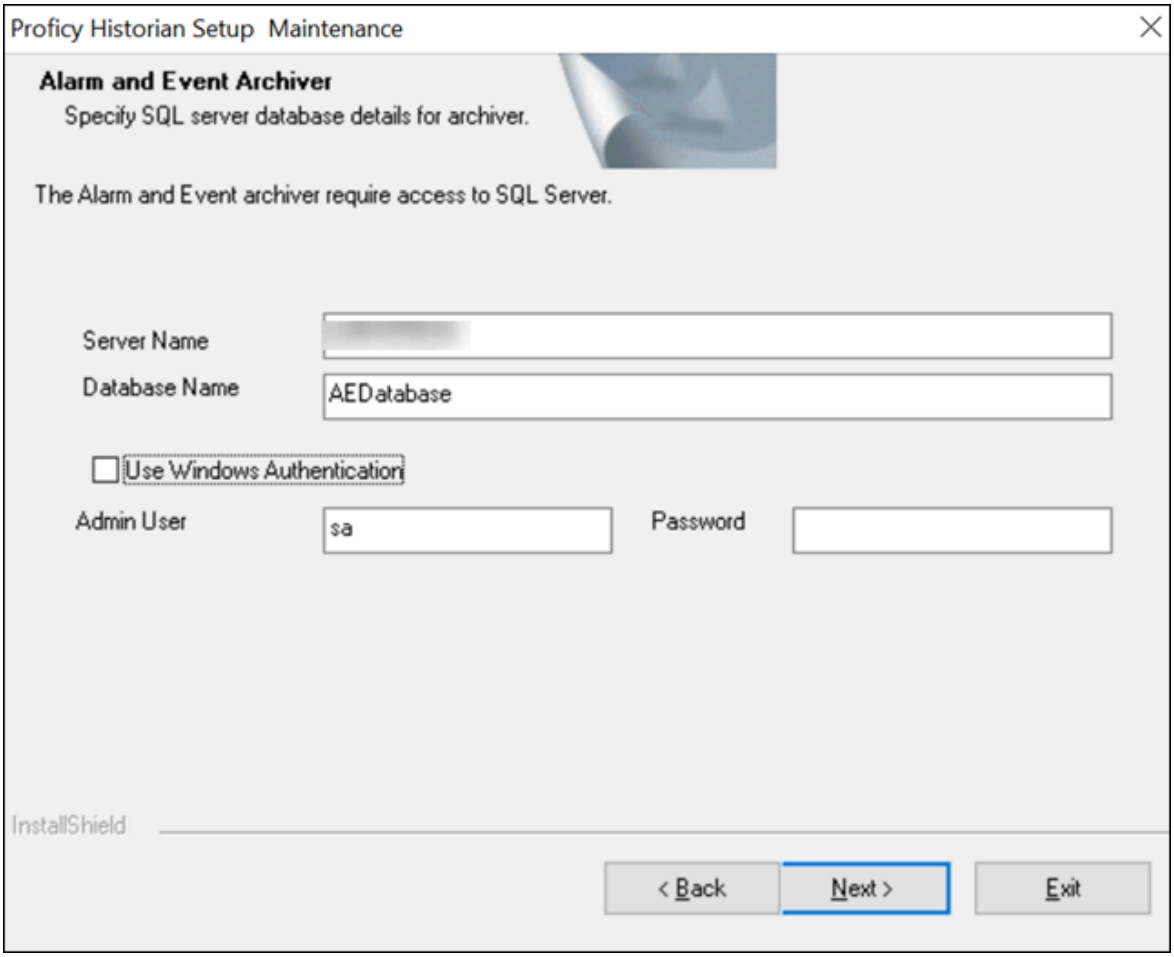
- Configuration Hub
- Historian Administrator
- Crystal Reports 11 or later
- The OLEDB Provider
- REST APIs

**Note:**
Before starting the Alarm Archive service, ensure that "`NT AUTHORITY/SYSTEM`" has "`SysAdmin`" privileges.

# Install Alarms and Events

- You must install Historian Alarms and Events on the same machine as the data archiver.
- If you have chosen to connect Historian to a remote SQL server, the following conditions must be satisfied:
  - The Historian Alarm Archiver service must be run on a user account that has privileges to log in to the SQL server using Windows authentication.
  - The default backup path, which you can set on the Archive page, must be a shared directory that is accessible to both the Historian Data Archiver and the remote SQL server. It is recommended that this shared directory be placed on the same computer as the Historian Data Archiver service.

1. Run the `InstallLauncher.exe` file.
2. Select **Install Alarms and Events**.
   The **Alarms and Events Archiver** page appears.




3. If needed, change the values in the **Server Name** and **Database Name** fields to provide the name of the SQL server and the name of the database where the alarms and events data is archived.
4. If you want to use the SQL server credentials, clear the **Use Windows Authentication** check box, and then enter the SQL server login credentials in the **Admin User** and **Password** fields. If you want to use Windows authentication, select the **Use Windows Authentication** check box. When you do so, the **Admin User** and **Password** fields are disabled.
5. Select **Next**.
6. When prompted to restart your system, select **Yes**.
   Historian Alarms and Events is installed in the following folder: `<installation drive>:\Program Files\Proficy\Proficy Historian\x86\Server`, and the following registry path is created: `HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\ Intellution, Inc.\iHistorian\Services\AlarmArchiver`

7. To verify that the Alarms service has started, access the **Services** window, and check the status of the Historian Alarm Archiver service.
   If the **Startup Type** field is set to **Automatic**, the service is started automatically when the system is started or restarted.

# Upgrade Alarms and Events

- If Alarms and Events were installed prior to Historian 7.0, you must install them separately.
- If you want to upgrade from Historian 4.5, since the database schema are different, if you select the same database name that is pre-populated by default, you will get an error message: `Later or Higher version of Alarms and Events database is already installed. Hence, you cannot proceed further.` You need to enter a different database name and then proceed with the upgrade.

Install Alarms and Events *(on page 140)*.
Alarms and Events are upgraded to the latest version.

# Change the SQL Server to Store Alarms and Events Data

If you have chosen to connect Proficy Historian to a remote SQL server:

- The Historian Alarm Archiver service must be run on a user account that has privileges to log into the SQL Server using Windows Authentication.
- The Default Backup Path, found on the Archive page, must be a shared directory that is accessible to both the Historian Data Archiver and the remote SQL Server. It is recommended that this shared directory be placed on the same computer as the Historian Data Archiver service.

1. Run the `Proficy.Historian.AandE.Migration.exe` file. By default, it is located at `C:\Program Files (x86)\Proficy\Proficy DataBase`.
   The **Proficy Historian Alarm and Event Data Migration** window appears.
2. Select **Specify SQL Server Instance**.
3. In the **Server Name**, **Database Name**, **Admin User**, and **Admin User Password** fields, enter values.
4. Select **Use default SQL path as data and log path** if you want to place the data and log files at the default location as the SQL server.
   If you want to install the data and log files in a different location, cancel the selection and in the **Alarms Data Path** and **Alarms Log Path** fields, and provide the path to place the data and log files.

# About Backing up Alarms

It is a good practise to back up alarm data periodically. It helps restore the data if you accidentally delete it. In addition, if you want to migrate alarms and events data from one version of Historian to another, you must first back up the data.

When you back up alarms, a copy of the alarm data is created on your machine, which you can restore *(on page 1209)* later.

You can back up alarm data using any of the following methods:

- Using Configuration Hub *(on page 753)*
- Using the Proficy Historian Alarm and Event Data Migration utility *(on page 1205)*
- Using Historian Administrator *(on page 1206)*
- At a command prompt *(on page 1208)*

## Back up Alarms Using Configuration Hub

1. Access Configuration Hub *(on page 336)*
2. In the **NAVIGATION** section, under the Configuration Hub for Historian, select **Alarms**.
   The **Alarms** section appears.
3. Select , and then select **Backup Alarms**.
   The **Backup Alarms** window appears.
4. Enter values as described in the following table.

| Field | Description |
|---|---|
| **GENERATED FROM** | Enter the start time for which you want to back up alarms. |
| **GENERATED TO** | Enter the end time for which you want to back up alarms. |
| **PROVIDE FILE NAME** | Enter the file name and location of the back up file that you want to create. By default, the backup location of the default data store is considered. And, the file name is in the following format: `mm_dd_yyyy_hh_mm_ss`. For example: 11_25_2022_11_23_26.zip. The time format followed is the 24-hour time notation.<br>**Note:**<br>This file naming convention for alarm backup is standard to Historian. Even if you use other date settings |

| Field | Description |
|---|---|
| | such as `dd-mm`, the file name will still be saved in the mm-dd format.<br><br>The end time stamp in the file name indicates the time at which the alarms have been backed up but not the time till when the alarms are backed up. For example, suppose, at 8.00 am, you back up alarms for the past two hours, the backup will contain alarms from 6:00 am to 8:00 am but may not contain an alarm generated at 8:00 am. The last alarm may have been at 7:50 am. But the backup file name will have the time stamp of 8:00:00 along with the date. |

5. Select **Backup**.
   The alarms data is backed up.

## Back Up Alarms and Events Using the Utility

1. Run the `Proficy.Historian.AandE.Migration.exe` file. By default, it is located at `C:\Program Files (x86)\Proficy\Proficy DataBase`.
   The **Proficy Historian Alarm and Event Data Migration** window appears.
2. Select **Backup Existing Alarms and Events**.
3. Enter values as described in the following table.

| Field | Description |
|---|---|
| **Time Range** | Enter the duration in which you want to back up alarms. |
| **SQL Server Instance Name** | Enter the SQL server instance name in which the alarms are stored. |
| **Database Name** | Enter the name of the database from which you have to backup the data. Typically, this will be the same as the SQL Server you are currently using. |
| **Use Windows Authentication** or **Use SQL Authentication** | Select either of these options to specify the type of authentication you want to use. |
| **User Id** | Depending on whether you have selected **Use Windows Authentication** or **Use SQL Authentication**, enter the Windows or SQL |

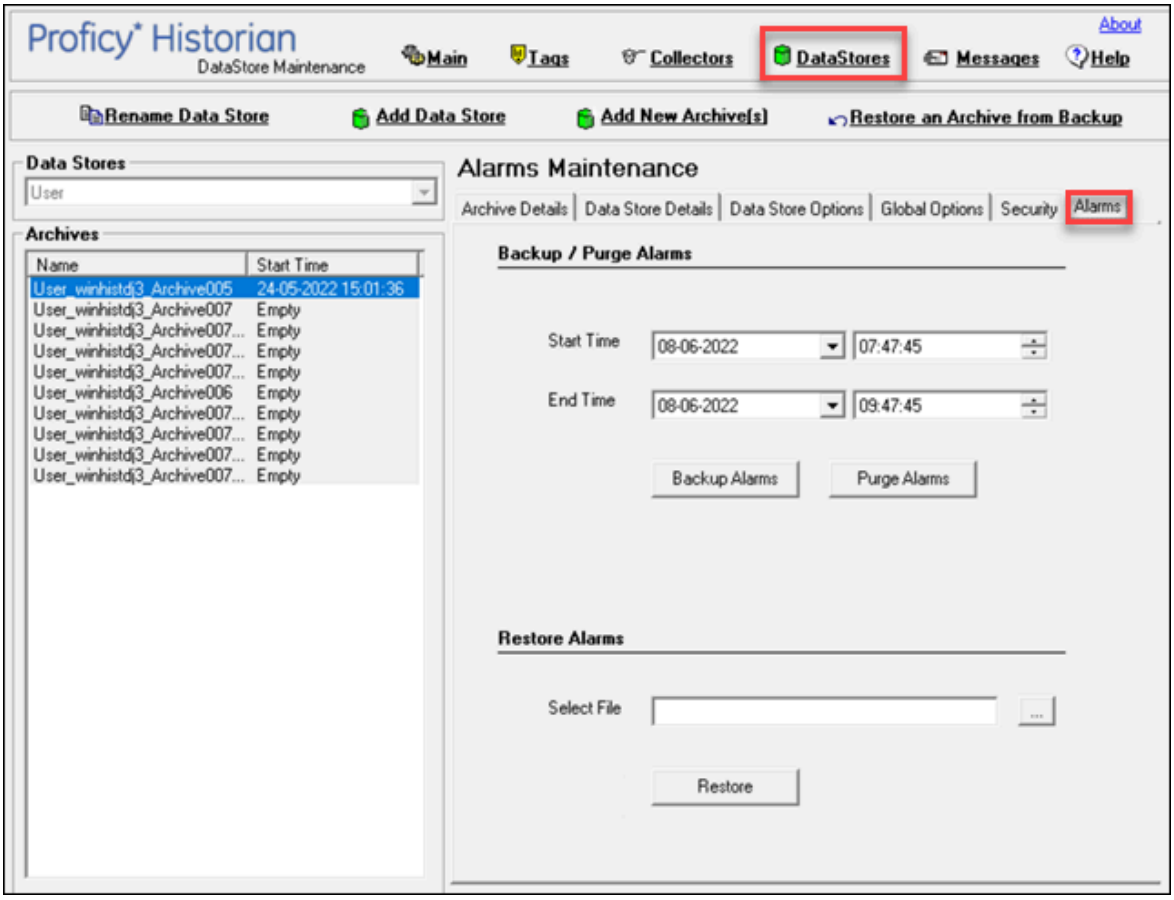
| Field | Description |
|---|---|
| | server username to connect to the SQL server. Ensure that this user has the permission to back up alarms. |
| **Password** | Depending on whether you have selected **Use Windows Authentication** or **Use SQL Authentication**, enter the Windows or SQL server password to connect to the SQL server. |
| **Backup Folder Path** | Enter the absolute path, including file name, to store the backed up alarms (for example, `c:\temp\March2010.bak`).<br><br>If the SQL server is on a remote machine, enter a path that exists on the remote machine. |

4. Select **Test Connection** to verify that the source database is active and the information is accurate. The **Begin Backup** button is enabled after the connection is successful.
5. Select **Begin Backup**.
   The alarm data is backed up.

## Back Up Alarms Using Historian Administrator

When you back up alarms, a copy of the alarm data is created on your machine, which you can restore later.

1. Access Historian Administrator *(on page 823)*.
2. Select **Data Stores > Alarms**.




3. In the **Start Time** and **End Time** fields, enter the duration in which you want to back up alarms.
4. Select **Backup Alarms**, provide a name for the backup file, and then save the file. If the SQL server is on a different machine, ensure that you specify a path available to the SQL server.
   By default, the file name is in the following format: `mm-dd-yyyy hh:mm:ss`. The time format followed is the 24-hour time notation.

> **Note:**
> This file naming convention for alarm backup is standard to Historian. Even if you use other date settings such as `dd-mm`, the file name will still be saved in the mm-dd format.

The end time stamp in the file name indicates the time at which the alarms have been backed up but not the time till when the alarms are backed up. For example, suppose, at 8.00 am, you back up alarms for the past two hours, the backup will contain alarms from 6:00 am to 8:00 am but may not contain an alarm generated at 8:00 am. The last alarm may have been at 7:50 am. But the backup file name will have the time stamp of 8:00:00 along with the date.

The alarms data is backed up.

## Back up Alarms at a Command Prompt

When you back up alarms, a copy of the alarm data is created on your machine, which you can restore later.

1. Run the `ihBackupAlarms.exe` file. By default, it is located at `C:\Program Files\Proficy\ProficyHistorian\Server\ihBackupAlarms.exe`
2. Use the following parameters to back up alarms data:

| Parameter | Description |
|---|---|
| `-s <Historian server>` | The Historian server containing the alarm data that you want to back up. If you do not provide this parameter, the local machine is considered. |
| `-u <username>` | The username to connect to the archiver. |
| `-p <password>` | The password to connect to the archiver. |
| `-b "<file path>"` | The file path where you want to place the backup file. If the path is not specified, the default backup path of the archive is considered.<br><br>**Note:**<br>You must enter the file path in double quotes (for example, `"C:\\backup\\"`). |
| `-d <number of Days prior to the current time>` | Indicates the time in days till the current date for which you want to back up alarms. For example, to back up alarms of the past seven days, enter:<br>`-d 7` |
| `-h <number of hours prior to the current time>` | Indicates the time in hours till the current time for which you want to back up alarms. For example, to back up alarms of the past two and half days, enter:<br>`-d 2 -h 12` |
| `-m <number of minutes prior to the current time>` | Indicates the time in minutes till the current time for which you want to back up alarms. For example, to back up alarms of the past 3 hours 15 minutes, enter: |

| Parameter | Description |
|---|---|
| | `-h 3 -m 15` |
| `-sec <number of seconds prior to the current time>` | Indicates the time in seconds till the current time for which you want to back up alarms. For example, to back up alarms of the past 30 seconds, enter:<br>`-sec 30` |

The alarm data is backed up.

# Restoring Data

## Restore Alarms Using Configuration Hub

Restoring alarms to a running system makes them available for query and analysis. You can restore alarms that have been backed up or deleted previously.

1. Access Configuration Hub *(on page 336)*
2. In the **NAVIGATION** section, under the Configuration Hub for Historian, select **Alarms**.
   The **Alarms** section appears.
3. Select , and then select **Restore Alarms**.
   The **Restore Alarms** window appears.
4. In the **PROVIDE FILE NAME** field, provide the backup file in the .zip format that you want to restore.

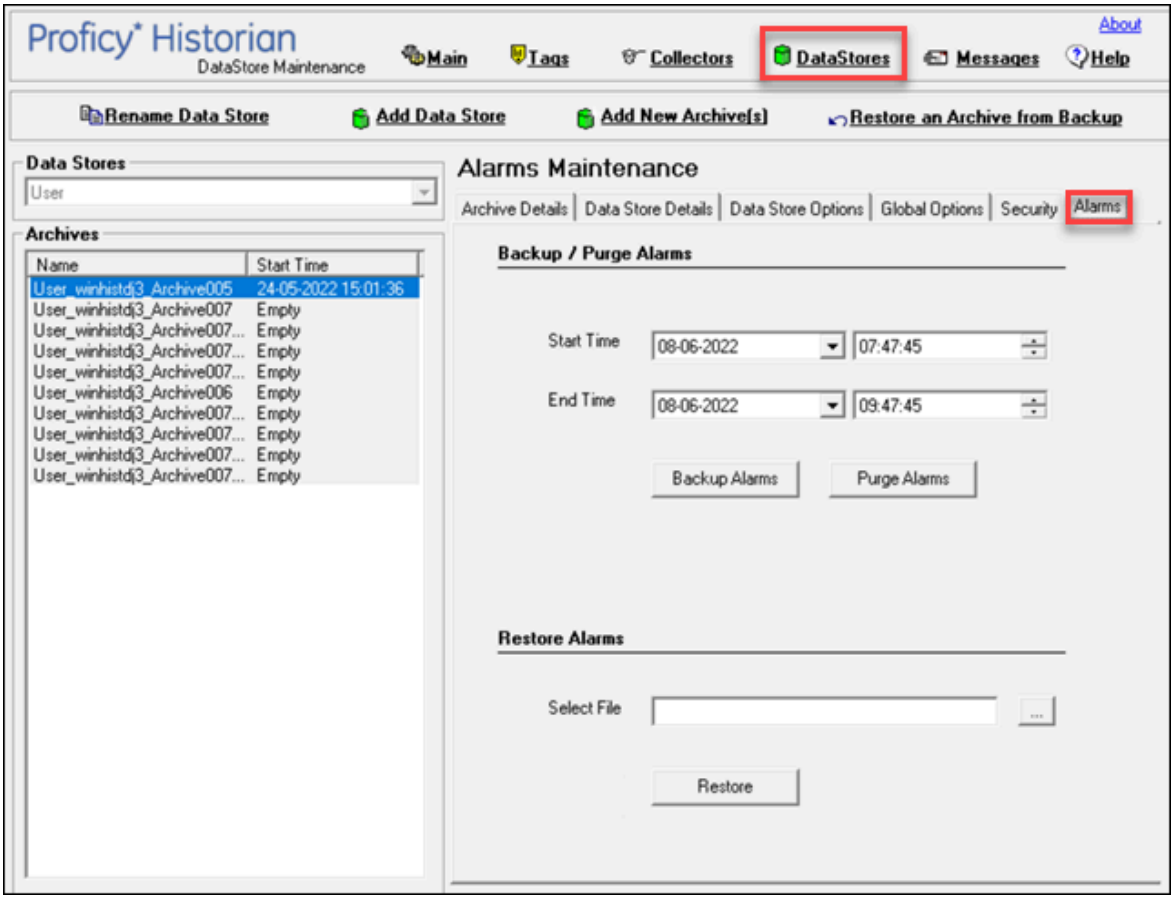
   **Note:**
   Remember to include .zip at the end of the file name.

5. Select **Restore**.
   The alarm data is restored.

## Restore Alarms Using Historian Administrator

Restoring alarms to a running system makes them available for query and analysis. You can restore alarms that have been backed up or deleted previously.

1. Access Historian Administrator *(on page 823)*.
2. Select **Data Stores > Alarms**.




3. In the **Select File** field, provide the backup file in the .zip format that you want to restore.
4. Select **Restore**.

   The alarm data is restored.

## About Migrating Alarms and Events Data

If you are upgrading to the latest version of Historian, and you have already collected alarms, you can migrate the alarms and events data after upgrading to latest version of Historian. Only after you migrate the data, you can retrieve alarms.

> **Note:**
> Before migrating alarms and events data, ensure that you have backed up the data *(on page 1205)*.

To migrate your alarms into the new alarm database, you must back up the old alarms, and restore them into the new database. You can back up the data before the upgrade using Configuration Hub or Historian

Administrator, or you can back up the data after the upgrade using the Proficy Alarms Database Migration Tool.

You may need to migrate small time periods if you have many alarms. If you need to migrate the alarms in blocks of time, do the oldest alarms first.

## Migrate Alarms and Events Using the Utility

Back up the alarms and events data *(on page 1205)*.

1. Run the `Proficy.Historian.AandE.Migration.exe` file. By default, it is located at `C:\Program Files (x86)\Proficy\Proficy DataBase`.
   The **Proficy Historian Alarm and Event Data Migration** window appears.
2. Select **Migrate Alarms and Events Backup**.
3. Enter values as described in the following table.

| Field | Description |
|---|---|
| **Backup File Path** | Enter the location of the backup file. If your server is running on a remote computer, enter a path that exists on the remote computer. |
| **SQL Server Instance Name** | Enter the SQL server instance name in which the backup file is stored. |
| **Database Name** | Enter the name of the database using which you have backed up the data. Typically, this will be the same as the SQL Server you are currently using. |
| **Use Windows Authentication** or **Use SQL Authentication** | Select either of these options to specify the type of authentication you want to use. |
| **User Id** | Depending on whether you have selected **Use Windows Authentication** or **Use SQL Authentication**, enter the Windows or SQL server username to connect to the SQL server. Ensure that this user has the permission to restore alarms. |
| **Password** | Depending on whether you have selected **Use Windows Authentication** or **Use SQL Authentication**, enter the Windows or SQL server password to connect to the SQL server. |

4. Select **Test Connection** to verify that the source database is active and the information is accurate.

> **Note:**
> You can find this information in the registry under: `HKEY_LOCAL_MACHINE\SOFTWARE\Intellution, Inc.\iHistorian\Services\AlarmArchiver`.

The **Begin Migration** is activated and the alarm data is restored.

5. Select **Begin Migration**.
   The data is migrated.

## Migrate Alarms and Events at a Command Prompt

Back up the alarms and events data *(on page 1205)*.

1. Run the `ihBackupAlarms.exe` file. By default, it is located at `C:\Program Files\Proficy\ProficyHistorian\Server\ihBackupAlarms.exe`
2. Use the following parameters to migrate data:

| arameter | Description |
|---|---|
| `/TASK` | Specify whether you want to back up or restore data. Enter one of the following values:<br>◦ BACKUP<br>◦ RESTORE<br>This parameter is required. |
| `/SERVER` | Enter the SQL server instance name in which the backup file is stored. This parameter is required. |
| `/DBNAME` | Enter the name of the database using which you have backed up the data. Typically, this will be the same as the SQL Server you are currently using. This parameter is required. |
| `/SDATETIME` | Enter the start time from which you want to migrate alarms. Enter a value in the following format: mm/dd/yyyy HH:MM:ss (for example 5/21/2023 10:20:30). Alarms and events data collected from this time will be considered. This parameter is mandatory if you are backing up alarms. |
| `/EDATETIME` | Enter the end time up to which you want to migrate alarms. Enter a value in the following format: mm/dd/yyyy HH:MM:ss (for example 5/21/2023 10:20:30). Alarms and events data collect- |

| arameter | Description |
|---|---|
| | ed from this time will be considered. This parameter is mandatory if you are backing up alarms. |
| /BKPPATH | Fully qualified path of the .dat file where the backup of the database will be taken. This parameter is mandatory if you are backing up alarms. |
| /RSTPATH | Fully qualified path of the .dat file from where the database will be restored. This parameter is mandatory if you are restoring alarms. |
| /USERNAME | Enter the Windows or SQL server username to connect to the SQL server. If you do not provide this parameter, Windows authentication is used. |
| /PWD | Enter the Windows or SQL server password to connect to the SQL server. By default, the password is blank, but we recommend that you enter a password. |

To back up using SQL authentication:

```
Proficy.Historian.AandE.Migration.exe /TASK=Backup
/SERVER=CPHIST45\PROFICYHIST /DBNAME=CPHIST45_plantdatabase
/SDateTime="3/9/2013 21:00:00" /EDateTime="3/9/2013 23:59:59"
/BKPPATH=C:\temp\PDB_0309.dat /USERNAME=sa /PWD=Pr0f1cyhist
```

To back up using Windows authentication:

```
Proficy.Historian.AandE.Migration.exe /TASK=Backup
/SERVER=Domain\Node /DBNAME=AEDB /SDateTime="5/10/2012 10:20:30"
/EDateTime="6/10/2012 10:20:30" /BKPPATH=E:\DBBkpPath\AEDB_1.dat
```

To restore using SQL authentication:

```
Proficy.Historian.AandE.Migration.exe /TASK=Restore
/SERVER=CPHIST45\PROFICYHIST /DBNAME=CPHIST45_AEDatabase
/RSTPATH=C:\temp\PDB_0309.dat /USERNAME=sa /PWD=Pr0f1cyhist
```

To restore using Windows Authentication:

```
Proficy.Historian.AandE.Migration.exe /TASK=RESTORE
/SERVER=Domain\Node /DBNAME=AEDB /RSTPATH=E:\DBBkpPath\AEDB_1.dat
```

To perform multiple back ups using a batch file:

```
cd C:\Program Files\Proficy\Proficy DataBase
```

```
start/b/wait Proficy.Historian.AandE.Migration.exe /TASK=Backup
/SERVER=CPHIST45\PROFICYHIST /DBNAME=CPHIST45_plantdatabase
/SDateTime="3/9/2013 21:00:00" /EDateTime="3/9/2013 23:59:59"
/BKPPATH=C:\temp\PDB_0309.dat /USERNAME=sa /PWD=Pr0f1cyhist
```

```
start /b /wait Proficy.Historian.AandE.Migration.exe /TASK=Backup
/SERVER=CPHIST45\PROFICYHIST /DBNAME=CPHIST45_plantdatabase
/SDateTime="3/10/2013 00:00:00" /EDateTime="3/10/2013 02:00:00"
/BKPPATH=C:\temp\PDB_0310_1.dat /USERNAME=sa /PWD=Pr0f1cyhist
```

```
start /b /wait Proficy.Historian.AandE.Migration.exe /TASK=Backup
/SERVER=CPHIST45\PROFICYHIST /DBNAME=CPHIST45_plantdatabase
/SDateTime="3/10/2013 02:00:00" /EDateTime="3/10/2013 04:00:00"
/BKPPATH=C:\temp\PDB_0310_2.dat /USERNAME=sa /PWD=Pr0f1cyhist
```

To perform multiple restores using a batch file:

**Multiple Restores Using a Batch File:**

```
cd C:\Program Files\Proficy\Proficy DataBase
```

```
start/b/wait Proficy.Historian.AandE.Migration.exe /TASK=Restore
/SERVER=CPHIST45\PROFICYHIST /DBNAME=CPHIST45_AEDatabase
/RSTPATH=C:\temp\PDB_0309.dat /USERNAME=sa /PWD=Pr0f1cyhist
```

```
start /b /wait Proficy.Historian.AandE.Migration.exe /TASK=Restore
/SERVER=CPHIST45\PROFICYHIST /DBNAME=CPHIST45_AEDatabase
/RSTPATH=C:\temp\PDB_0310_1.dat /USERNAME=sa /PWD=Pr0f1cyhist
```

```
start /b /wait Proficy.Historian.AandE.Migration.exe
/TASK=Restore /SERVER=CPHIST45\PROFICYHIST /DBNAME=CPHIST45_AEDatabase
/RSTPATH=C:\temp\PDB_0310_2.dat /USERNAME=sa /PWD=Pr0f1cyhist
```

**Note:**

Parameter values with spaces must be enclosed within quotation marks.

# Query Alarms and Events Data Using the Excel Add-In

Querying Alarms and Events data in the Excel Add-In retrieves alarms and events data according to your Query Criteria. Three query types are available: Alarm, Alarm history, and Events.

**Note:**
You cannot use an OPC alarms and events server with FIX32 SCADA systems. To collect alarm data from these systems, you can use a proxy for alarms through an iFIX node with an OPC alarms and events server. The OPC Classic Alarms and Events collector of Proficy Historian can then collect the data. For more information, refer to Using the OPC AE Collector with FIX32 SCADA Systems *(on page 1225)*.

1. Open an Excel worksheet.
2. Select **Historian > Query Alarms & Events**.
   The **Query Alarms & Events** window appears.
3. Enter values as described in the following table.

| Field | Description |
|---|---|
| **Server** | Select the SQL server that contains the alarms and events data. |
| **Query Type** | Select one of the following values:<br>◦ **Alarms**: In Historian, an alarm's entire life cycle is stored as a single record in the alarm archive. Thus, when retrieving from the archive, the entire life cycle of an alarm will be returned in a single record.<br>◦ **Alarm History**: Each change in the alarm's state will be returned in a single record.<br>◦ **Events**: One row per event is returned. |
| **Query Criteria** | Enter the criteria that you want to use in the query. For example, you may want to include alarms where the Alarm ID is equal to a specific Alarm ID occurring after a specific start time. In addition, you can specify which attributes must be displayed and how the results must be sorted in the spreadsheet. |
| **Output Range** | Select a range of cells in a single row or column to determine where the returned data must be placed. |
| **Output Orientation** | Select either **Columns** or **Rows** for the output display. Selecting **Columns** displays a table of values with parameters arranged in |

| Field | Description |
|---|---|
| | columns with header labels at the top. Selecting **Rows** rotates the table 90 degrees. |
| **Maximum Results** | Enter the maximum number of results for the query to return.<br><br>**Note:**<br>An Excel spreadsheet can display up to 255 columns and 32,767 rows. |
| **Output Display** | Specify the attributes that you want to display in the spreadsheet. |
| **Output Sorting** | Select the parameter using which you want to sort the results:<br>◦ **Alarm Time**: Sorts the data by the start time. The results appear in the reverse chronological order.<br>◦ **Custom Sort**: Allows you to select the field using which you want to sort the results.<br>◦ **None**: The results are not sorted at all. They are returned in the order they are received from the alarms and events database. |

4. Select **OK**.
   The query results appear.

## Import Alarms and Events Data into a Spreadsheet

Using the Historian Excel add-in, you can import alarms and events data into a spreadsheet. It helps you include alarms and events data into the Historian archive that is not normally collected by Historian or when you are migrating data from an older system into Historian.

**Note:**

- The Excel worksheet must contain at least the source and timestamp columns.
- If an error occurs while importing data, the whole operation is aborted.

1. Open an Excel worksheet.
2. Select **Historian > Administration > Import Alarms**.
   The alarms and events data is imported.

# Export Alarms and Events Data from a Spreadsheet

Using the Historian Excel add-in, you can export alarms and events data into a new worksheet or into a CSV or an XML file.

1. Open an Excel worksheet.
2. Select **Historian > Administration > Export Alarms**.
   The **Historian Alarm Export** window appears.
3. Enter values as described in the following table.

| Field | Description |
|---|---|
| **Server** | Select the SQL server that contains the alarms and events data. |
| **Query Type** | Select one of the following values:<br>◦ **Alarms**: In Historian, an alarm's entire life cycle is stored as a single record in the alarm archive. Thus, when retrieving from the archive, the entire life cycle of an alarm will be returned in a single record.<br>◦ **Alarm History**: Each change in the alarm's state will be returned in a single record.<br>◦ **Events**: One row per event is returned. |
| **Query Criteria** | Enter the criteria that you want to use in the query. For example, you may want to include alarms where the Alarm ID is equal to a specific Alarm ID occurring after a specific start time. In addition, you can specify which attributes must be displayed and how the results must be sorted in the spreadsheet. |
| **Output Range** | Select a range of cells in a single row or column to determine where the returned data must be placed. |
| **Output Orientation** | Select either **Columns** or **Rows** for the output display. Selecting **Columns** displays a table of values with parameters arranged in columns with header labels at the top. Selecting **Rows** rotates the table 90 degrees. |
| **Maximum Results** | Enter the maximum number of results for the query to return. |

| Field | Description |
|---|---|
| | **Note:** An Excel spreadsheet can display up to 255 columns and 32,767 rows. |
| **Output Display** | Specify the attributes that you want to display in the spreadsheet. |
| **Output Sorting** | Select the parameter using which you want to sort the results:<br>◦ **Alarm Time**: Sorts the data by the start time. The results appear in the reverse chronological order.<br>◦ **Custom Sort**: Allows you to select the field using which you want to sort the results.<br>◦ **None**: The results are not sorted at all. They are returned in the order they are received from the alarms and events database. |
| **Export Options** | Select the format in to which you want to export the data:<br>◦ **To New Worksheet**: Exports the data to a new Excel worksheet.<br>◦ **To CSV File**: Exports the data with comma separated values to a new file.<br>◦ **To XML File**: Exports the data to a new XML file. |

4. Select **OK**.
   The data is exported.

## About Purging Alarms

Purging alarm data involves deleting the data from the database.

**Note:**

- Even after purging, the data is not lost; a backup is created to maintain an audit trail. You can restore the data if needed.
- When using circular archives (that is, archives that roll over), alarms are purged automatically.

You can choose to purge alarm data for any of the following reasons:

- To maintain alarm data efficiently
- The data is outdated or redundant
- The disk space is limited

Data in the following tables is purged:

- Alarm Attribute Values
- Alarm Attribute Value History
- Delete from Alarm History
- Delete from Alarm Table esignatures
- comments

You can purge data using one of the following methods:

- Purge data within a specified duration. You can do this using Configuration Hub *(on page 756)*, using the Proficy Historian Alarm and Event Data Migration utility *(on page 1221)* or at a command prompt *(on page 1223)*.
- Purge data related to a specific alarm ID. You can do this using `Alarms.PurgeAlarmsById` *(on page 1604)* to develop an SDK program.

Purging is performed in batches. You can check the log data in the `Proficy.Historian.AandE.Migration.log` file. By default, this file is located in the `C:\Program Files(x86)\Proficy` folder.

In the case of a failure:

- The batch size is changed to 10. That is, the collector receives an acknowledgement after sending 10 messages, thus reducing the load on the server.
- The waiting time for receiving an acknowledgement is automatically incremented after each failure per batch, starting from 90 seconds to 270 seconds. This gives more time for the server to respond.

**Note:**
After the acknowledgement is received, the batch size and the waiting time are reset for the subsequent batches.

If the time taken to purge exceeds the timeout limit, instead of reverting the entire purging operation, only the current batch, which is still under processing, is purged.

**Best Practices:**

- Restart the Alarms and Events services before purging data.

## Purge Alarms Using Configuration Hub

1. Access Configuration Hub *(on page 336)*
2. In the **NAVIGATION** section, under the Configuration Hub for Historian, select **Alarms**.
   The **Alarms** section appears.
3. Select , and then select **Purge Alarms**.
   The **Purge Alarms** window appears.
4. In the **PROVIDE FILE NAME** field, pro
5. Enter values as described in the following table.

| Field | Description |
|---|---|
| **GENERATED FROM** | Enter the start time for which you want to purge alarms. |
| **GENERATED TO** | Enter the end time for which you want to purge alarms. |
| **PROVIDE FILE NAME** | Enter the file name and location of the back up file that you want to create. This field is enabled only if you select the **BACKUP ALARMS BEFORE PURGE?** check box. By default, the backup location of the default data store is considered. And, the file name is in the following format: `mm_dd_yyyy_hh_mm_ss`. For example: 11_25_2022_11_23_26.zip. The time format followed is the 24-hour time notation.<br><br>**Note:**<br>This file naming convention for alarm backup is standard to Historian. Even if you use other date settings such as `dd-mm`, the file name will still be saved in the mm-dd format.<br><br>The end time stamp in the file name indicates the time at which the alarms have been backed up but not the time till when the alarms are backed up. For example, suppose, at 8.00 am, you |

| Field | Description |
|---|---|
| | back up alarms for the past two hours, the backup will contain alarms from 6:00 am to 8:00 am but may not contain an alarm generated at 8:00 am. The last alarm may have been at 7:50 am. But the backup file name will have the time stamp of 8:00:00 along with the date. |
| **BACKUP ALARMS BEFORE PURGE?** | Select the check box if you want to back up alarms before purging. |

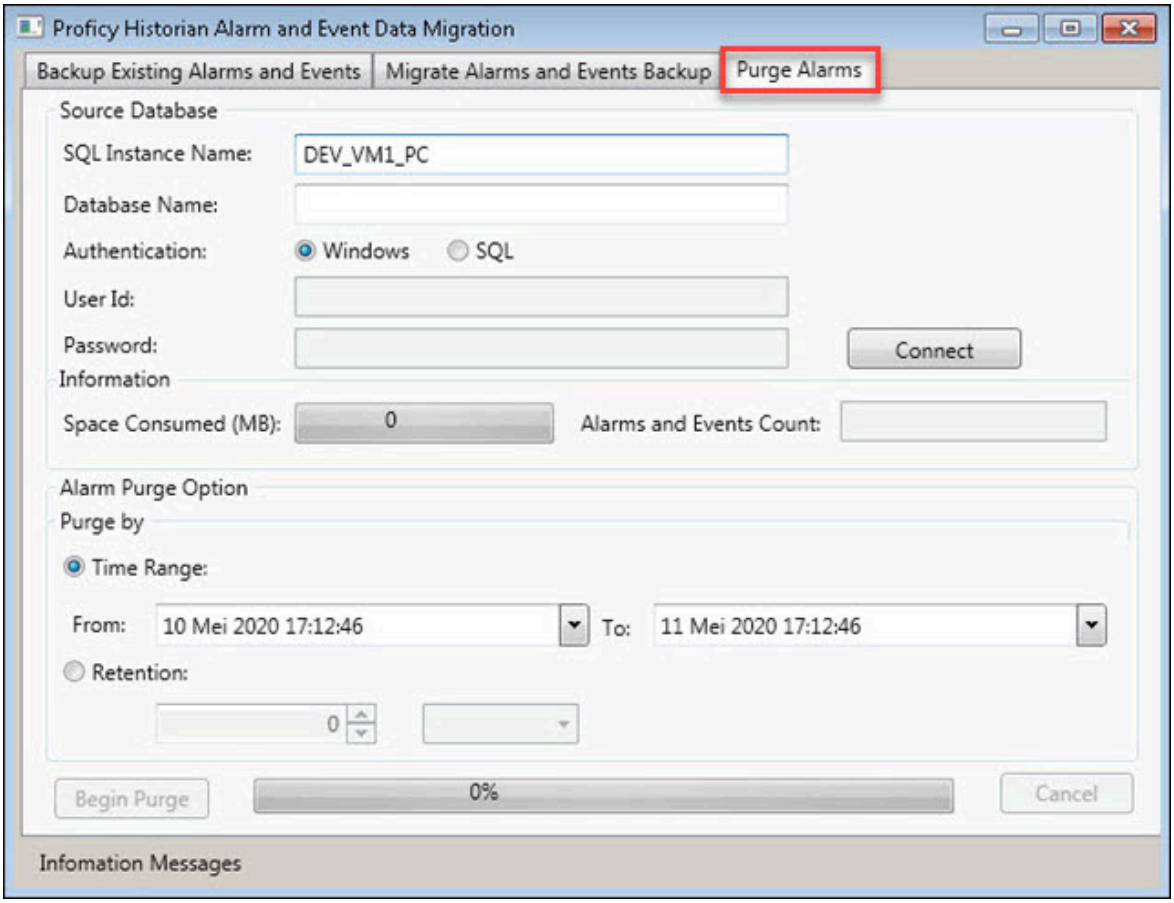
6. Select **Purge**.
   The alarms data is purged. A backup file is created if you have selected the **BACKUP ALARMS BEFORE PURGE?** check box.

## Purge Data Using the Utility

This topic describes how to purge alarms using the Proficy Historian Alarm and Event Data Migration utility. You can also purge them at a command prompt *(on page 1223)*.

Ensure that you are a member of the ihsecurityAdmins security group.

1. Run the `Proficy.Historian.AandE.Migration.exe` file. By default, it is located at `C:\Program Files (x86)\Proficy\Proficy DataBase`.
   The **Proficy Historian Alarm and Event Data Migration** window appears.
2. Select **Purge Alarms**.




The **Purge Alarms** section appears.

3. Enter values as described in the following table.

| Field | Description |
|---|---|
| **SQL Instance Name** | Enter the Microsoft SQL server instance that contains the data that you want to purge. |
| **Database Name** | Enter the name of the database that contains the data that you want to purge. |
| **Authentication** | Select **Windows** or **SQL** depending on whether you want to provide credentials of the Windows machine or the SQL server, respectively. |
| **User Id** | Enter the User ID of the Windows user or the SQL user (depending on whether you have selected **Windows** or **SQL**). |
| **Password** | Enter the password of the Windows or SQL user account (depending on whether you have selected **Windows** or **SQL**). |

4. Select **Connect**.
   A message appears, specifying that a connection to the Microsoft SQL database is established. The **Space Consumed (MB)** and the **Alarms and Events count** fields are populated with the appropriate values.
5. If you want to purge data within a time range, provide the start time and end time in the **From** and **To** fields, respectively.
6. If you want to purge data for a certain duration till the current date and time, enter a value in the **Retention** field, and then select the units of measurement. The default value is 1 minute.
   If you want to purge data until last week, enter 7, and then select **Days**. Data from the beginning until seven days prior to the current date and time is purged.
7. Select **Begin Purge**.
   The data received within the specified duration is purged. You can check the log data in the `Proficy.Historian.AandE.Migration.log` file. By default, this file is located in the `C:\Program Files(x86)\Proficy` folder.

   In addition, an audit log is created at `..\Proficy Historian Data\LogFilesfolder`.

## Purge Alarms at a Command Prompt

This topic describes how to purge alarms at a command prompt. You can also purge them using the Proficy Historian Alarm and Event Data Migration utility *(on page 1221)*.

Ensure that you are a member of the ihsecurityAdmins security group.

1. Run the `ihPurgeAlarms.exe` file. By default, it is located at `C:\Program Files\Proficy\ProficyHistorian\Server\ihPurgeAlarms.exe`.
2. Use the following parameters to purge alarms.

| Parameter | Description |
|---|---|
| `-s <Historian server>` | The Historian server on which you want to access archive data. If you do not provide this parameter, the local machine is considered. |
| `-u <username>` | The username to connect to the archiver. |
| `-p <password>` | The password to connect to the archiver. |
| `-b "<file path>"` | The file path where you want to place the backup file of the purged alarms. If the path is not specified, the default backup path of the archive is considered. |

| Parameter | Description |
|---|---|
| | **Note:**<br>You must enter the file path in double quotes (for example, `"C:\\backup\\"`). |
| `-z 1 <file name>` | The name of the backup file in the .zip file format. For example, if you want to purge alarms older than 36 hours, but want to back up the data in a .zip file format, enter:<br>`ihPurgeAlarms.exe -h 36 -z 1`<br>If you run this command, the backup file will be created in the default archive backup folder. |
| `-d <number of Days prior to the current time>` | Indicates the time in days till the current date for which you want to purge alarms. For example, to purge alarms older than seven days, enter:<br>`-d 7` |
| `-h <number of hours prior to the current time>` | Indicates the time in hours till the current time for which you want to purge alarms. For example, to purge alarms older than two and half days, enter:<br>`-d 2 -h 12` |
| `-m <number of minutes prior to the current time>` | Indicates the time in minutes till the current time for which you want to purge alarms. For example, to purge alarms older than 3 hours 15 minutes, enter:<br>`-h 3 -m 15` |
| `-sec <number of seconds prior to the current time>` | Indicates the time in seconds till the current time for which you want to purge alarms. For example, to purge alarms older than 30 seconds, enter:<br>`-sec 30` |

The alarms are purged, and an audit log is created at `..\Proficy Historian Data \LogFilesfolder`.

# Close an Alarm

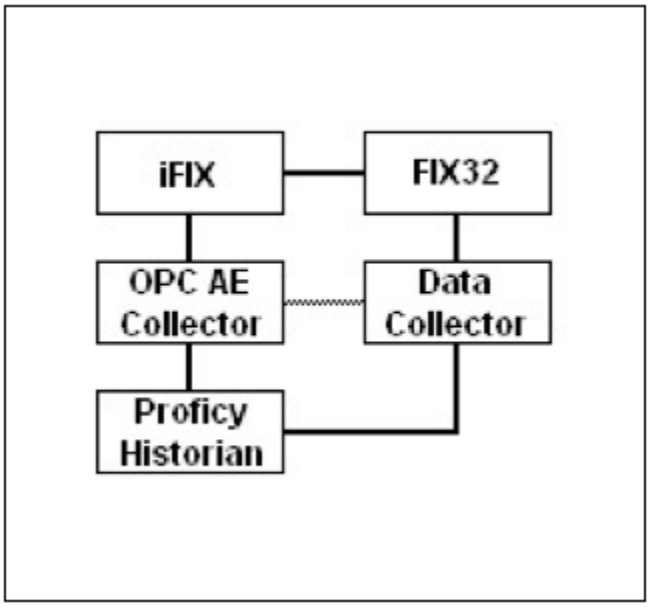
1. Access Historian Administrator *(on page 823)*.
2. Select **Collectors**.
3. From the list of collectors, select the alarms and events collector that you want to close.
4. Select **Close Alarms**.
   The **Close Alarms** window appears.
5. In the **End Date/Time** field, enter the duration for which you want to close alarms.
6. Select **Show Alarms**.
   A list of open alarms in the duration appear.
7. Select the alarms that you want to close, and then select **Close Alarms**.

# Using the OPC AE Collector with FIX32 SCADA Systems

You cannot use an OPC alarms and events server with FIX32 SCADA systems. To collect alarm data from these systems, you can use a proxy for alarms through an iFIX node with an OPC alarms and events server. The OPC Classic Alarms and Events collector of Proficy Historian can then collect the data.

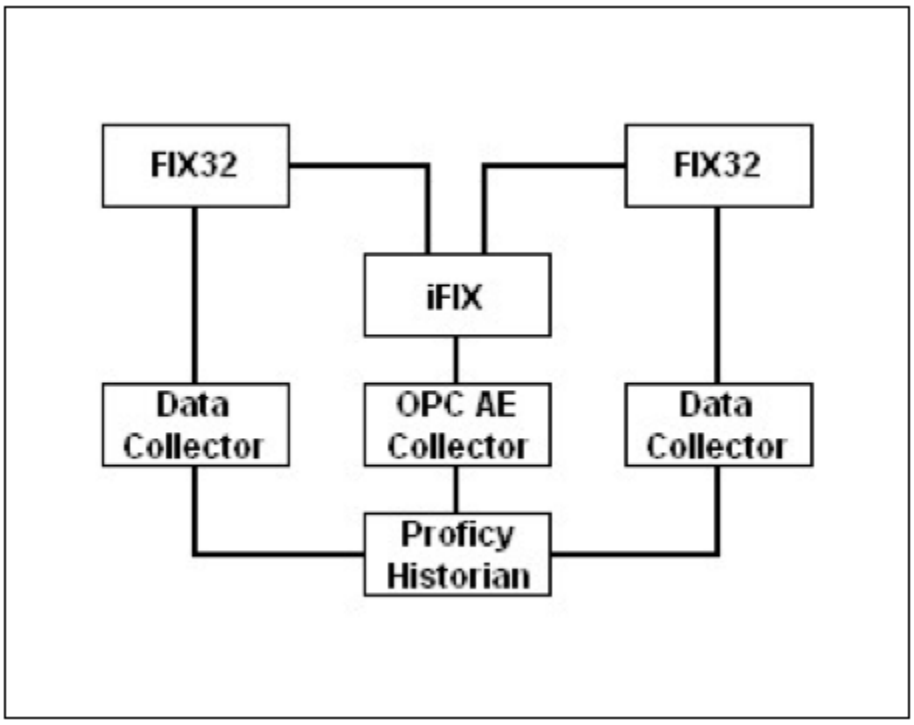
There are two different configurations that can occur in this instance:

1. A single proxy in which only one FIX32 system forwards its alarms through a particular iFIX node.




   In this case, you can link the OPC Classic Alarms and Events collector to the FIX32 data collector. Refer to Configure the OPC Alarms and Events Collector *(on page 2392)*. You can then query the data using clients such as Configuration Hub, Historian Excel add-in, and so on.

2. Multiple FIX32 system proxy alarms through a single iFIX node.




In this case, you cannot set the `Link To Data Collector` parameter to both FIX32 data collectors. As a result, it is not possible to create a relationship between alarm and tag data. Therefore, you cannot link the data using Historian Excel add-in.

To link the alarm and tag data in this scenario, perform an OLE DB query with a join between the `ItemID` column of the `ihAlarms` database and the `TagName` column of the `ihRawData` database. You can do this only if `No Prefix` has been added to the tagname.

The following example shows a parametrized query that uses the `AlarmID` property to find all values read from the corresponding tag for that alarm during the duration of the alarm.

```
SELECT   ihRawData.'Value', ihRawData.Tagname, ihRawData.'TimeStamp'

FROM     ihRawData, ihAlarms

WHERE    ihRawData.Tagname = ihAlarms.ItemID

AND    ( ihRawData.'TimeStamp' >= ihAlarms.Starttime )

AND    ( ihRawData.'TimeStamp' <= ihAlarms.'Timestamp' )

AND    ( ihAlarms.AlarmID = 13240 )

AND    ( ihRawData.SamplingMode = 'RawByTime' )
```

# Chapter 8. Historian REST APIs

## Introduction to Historian REST APIs

### Historian APIs

Historian is a high performance data archiving system designed to collect, store, and retrieve time-based information at extremely high speed efficiently. The Historian environment provides a set of REST APIs to query data from the archives.

This document provides links for setting up your development environment, as well as information for getting started with the Historian services and their associated APIs.

Starting Historian 8.0, the default https port is 443. If you use the default port, you need not include it in the Rest API calls.

Also, the default admin client name is changed from admin to hostname.admin, and it is case-sensitive.

Example:

```
curl -u admin:adminsecret https://<nodename>:443/uaa/oauth/token -d

      'grant_type=client_credentials'
```

should be replaced with

```
curl -u hostname.admin:adminsecret https://<nodename>/uaa/oauth/token -d

      'grant_type=client_credentials'
```

See the following topics for more information:


- About Security and Authentication *(on page 1227)*
- Standards *(on page 1229)*
- API Methods *(on page 1230)*
- API Status Messages *(on page 1230)*

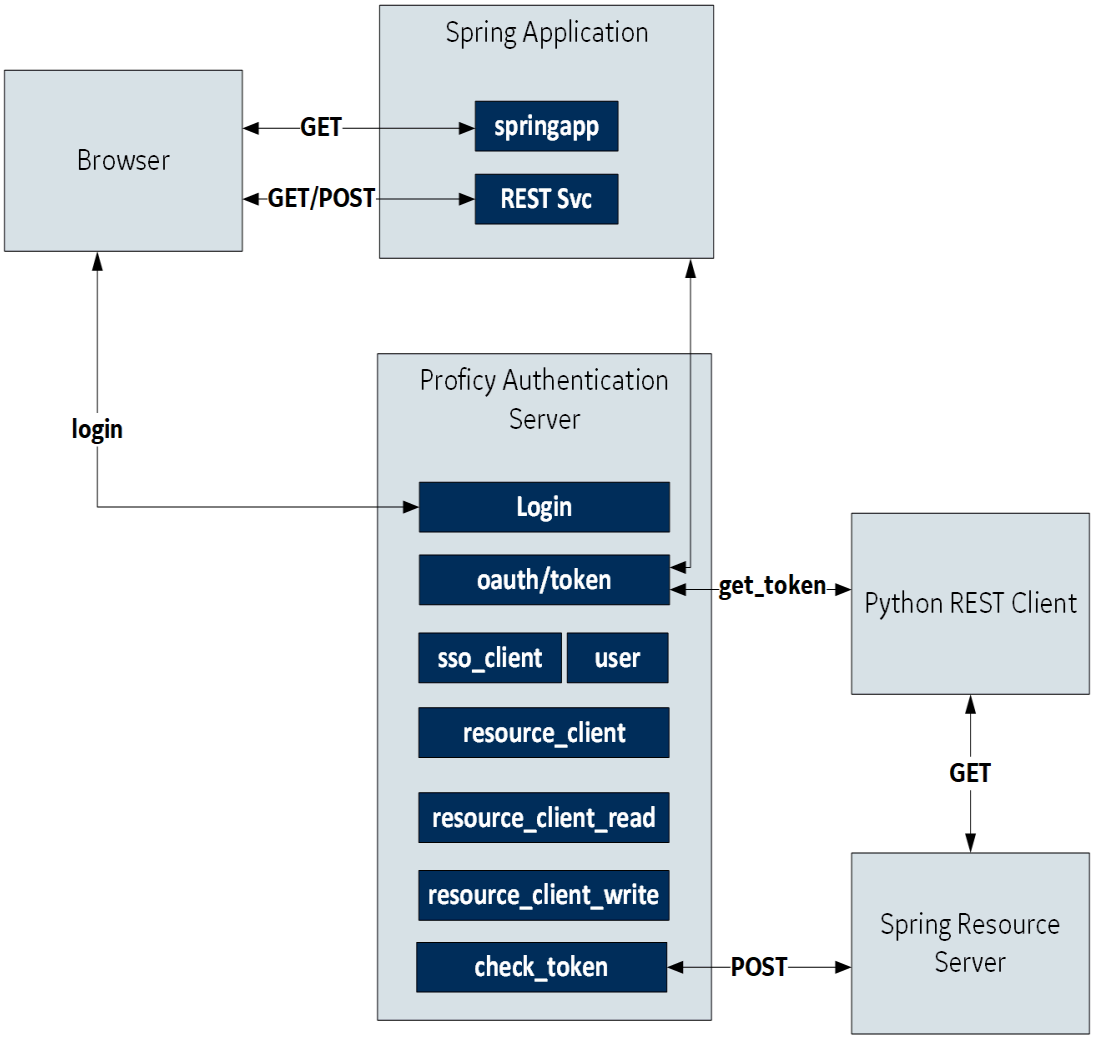

### About Security and Authentication

For security purposes, Historian uses the Proficy Authentication service as a trusted source of tokens issued for authentication. It is a multi-tenant identity management service, used in Cloud Foundry, but also available as a standalone OAuth2 server. Its primary role is as an OAuth2 provider, issuing tokens for client applications to use when they act on behalf of Cloud Foundry users. It can also authenticate users with Cloud Foundry credentials, and can act as an SSO service using those credentials, or others.

It contains endpoints for managing user accounts, registering OAuth2 clients, and other management functions.

The following diagram shows how the Proficy Authentication server functions with a Python REST client:

Figure 7. Proficy Authentication Server and Python REST Client




## Authorization

For exchanging data between the client-server system, user authentication is required. Once you have provided your client credentials, an access or bearer token is generated. This token is used for the REST APIs.

**cURL command format for generating an oauth token for clients**: `curl -u <client ID>:<client secret> https://<node name>:443/uaa/oauth/token -d 'grant_type=client_credentials'`

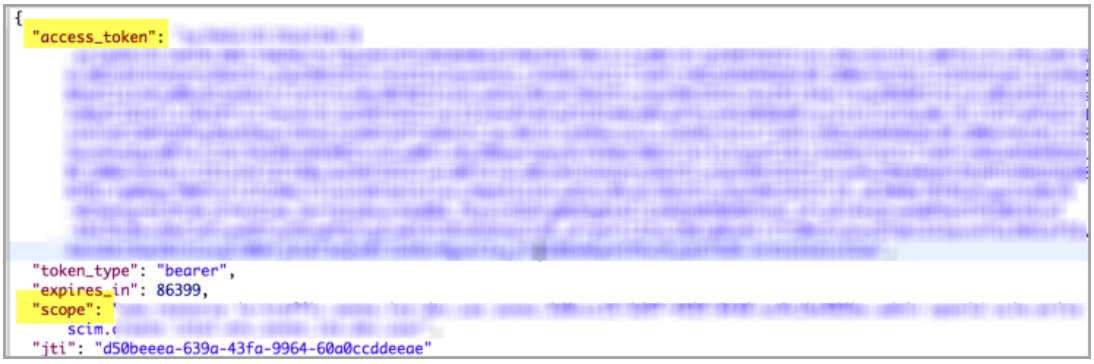
**Example:** `curl -u server1.admin:adminsecret https://server1:443/uaa/oauth/token -d 'grant_type=client_credentials'`

**cURL command format for generating an oauth token for Proficy Authentication users**: `curl -d "client_id=<value>&client_secret=<value>&grant_type=password&username=<value>&password=<value>&token_format=opaqu https://<node name>:8080/uaa/oauth/token`

**Example:** `curl -d "client_id=historian_public_rest_api&client_secret=publicapisecret&grant_type=password&username=<value>&password= https://<nodename>/uaa/oauth/token`

In the following image, the actual token text is blurred for security concerns.

Figure 8. OAuth Access Token Sample




Client applications can access data using service REST API endpoints. Your application makes an HTTP request and parses the response. You can use any web-development language to access the APIs.

## Standards

Historian APIs use a REST application architecture constrained by Hypermedia as the Engine of Application State (HATEOAS) that distinguishes it from most other network application architectures. Therefore, a client interacts with a network application entirely through hypermedia provided dynamically by application servers. The REST client doesn't need prior knowledge about how to interact with a particular application or server beyond a basic understanding of hypermedia.

As defined by the query parameters, the Historian APIs use "search" functions to access raw data using cURL and HTTP, while responses are in JSON format.

cURL is a command-line utility used to transfer data from or to a server, using one of the supported protocols, such as DICT, FILE, FTP, FTPS, GOPHER, HTTP, HTTPS, IMAP, IMAPS, LDAP,LDAPS, POP3, POP3S, RTMP, RTSP, SCP, SFTP, SMTP, SMTPS, TELNET and TFTP. The command is designed to work without user interaction.

cURL offers many useful functions such as proxy support, user authentication, FTP upload, HTTP post, SSL connections, cookies, file transfer resume, user and password authentication, and more.

You can run the sample commands provided in this document from Bash on Windows in the Windows operating system, and also in Linux Shell in the Linux operating system.

As a prerequisite, make sure you install cURL on your system, if it is not already installed. Run the `curl --version` command on Windows Bash or Linux shell to check if cURL is installed on your system.

**Important:**
Do not create your own URIs. Instead, use the links in this document and in the responses to navigate between resources.

## API Methods

The Historian APIs use GET, POST, PUT, and DELETE methods.

| Method | Usage |
|---|---|
| GET | Retrieves a resource. |
| POST | Creates (or adds) a resource. |
| PUT | Updates a resource. |
| DELETE | Removes a resource. |

## API Status Messages

In its use of the following HTTP status codes, the Historian API services adhere as closely as possible to standard HTTP and REST conventions.

| Status Code | Usage |
|---|---|
| 200 OK | Success message. The request has completed. |
| 201 Created | Success message. A new resource has been created. The resource URI is available from the location header in the response. |
| 204 No Content | Success message. An update to an existing resource has been applied. |
| 400 Bad Request | Error message. The request was malformed. The response body provides additional information. |

| Status Code | Usage |
|---|---|
| 401 Unauthorized | Error message. Either you are not authenticated, or the authentication is incorrect. You must re-authenticate and try again. |
| 403 Forbidden | Error message. You do not have permission to access this resource. |
| 404 Not Found | Error message. The requested resource does not exist. |
| 500 Internal Error | Error message. The server encountered an unexpected condition that prevented it from fulfilling the request. |

# Common API Parameters

## Overview of Commonly Used API Parameters

The Historian REST service provides various REST API calls to retrieve the current tags list and query data with different sampling modes. Most of these API calls use the following common parameters:


- tagNames *(on page 1231)*
- Start and End timestamps *(on page 1232)*
- TagSamples *(on page 1232)*
- DataSamples *(on page 1234)*
- SamplingModeType *(on page 1235)*
- Direction *(on page 1237)*
- CalculationModeType *(on page 1237)*
- FilterModeType *(on page 1243)*
- ReturnDataFields *(on page 1244)*
- Payload *(on page 1245)*
- Error Code Definitions *(on page 1251)*


## TagNames Parameter

By default, the Historian REST service provides support to read samples for multiple tags. Multiple tag names are separated by semicolons (**;**). For example, "tagname1;tagname2;tagname3".

```
https://<historianservername>:443/historian-rest-api

/v1/datapoints/currentvalue?tagNames=tagName1;tagname2;tagname3


https://RestTestsNode/historian-rest-api/v1/datapoints/currentvalue?tagNames=TAG01;TAG02
```

Encode the semicolon as `%3B` if using the URI format, as the semicolon is also a valid character for a Historian name, and the web service parses the tag names incorrectly if a tag name contains a semicolon.

## Start and End Timestamps Parameter

For the Start and End Timestamps parameter, the Timestamp format in the URI must be in ISO data format, such as `YYYY-MM-DDTHH:mm:ss.SSSZ`.

EPOCH time (standard base time) is only valid in the JSON-format request body or response body, such as `\/Date(928167600000-0500)\/`. If you use the same timestamp for start and end timestamps, the request returns a single result.

All timestamps passed to the REST service must be formatted as UTC timestamps.

| Object Name | Description |
|---|---|
| StartTime | Start time of the query. This represents the earliest timestamp for any tag contained in the query.<br><br>If no StartTime is specified, the start time is two hours prior to running the query. |
| EndTime | End time of the query. This represents the latest timestamp for any tag contained in the query.<br><br>If no EndTime is specified, the end time is the time that the query runs. |

## TagSamples Parameter

The TagSamples parameter is the output from the REST API calls.

| Property Name | Property Type | Description |
|---|---|---|
| TagName | String | Name of the tag. |
| DataType | String | **Tag Data Type Value:** |

| Property Name | Property Type | Description |
|---|---|---|
| | | • **Blob** – Stores tags as binary large objects. The Blob datatype generally refers to undetermined binary data types, such as an Excel spreadsheet, a PDF file, or a Word file.<br>• **Boolean** (one byte) – Stores boolean values. Valid values for the boolean data type are 0=FALSE and 1=TRUE. If the user sends zero, the value is taken as zero. Anything other than zero, the value is treated as one.<br>• **Byte** (one byte) – Stores integer values. Valid values for the byte data type are -128 to +127.<br>• **SingleFloat** (four bytes) – Stores decimal values up to six places. Valid ranges for the single float data type are 1.175494351e-38F to 3.402823466e+38F<br>• **DoubleFloat** (eight bytes) – Stores decimal values up to 15 places. Valid values for the double float data type are 2.2250738585072014e-308 to 1.7976931348623158e+308.<br>• **SingleInteger** (two bytes) – Stores whole numbers, without decimal places. Valid values for the single integer data type are -32767 to +32767.<br>• **DoubleInteger** (four bytes) – Stores whole numbers, without decimal places. Valid values for the double integer data type are -2147483648 to +2147483648.<br>• **FixedString** (Configured by user) – Stores string data of a fixed size. Valid values are between 0 and 255 bytes.<br>• **Float** – Single float.<br>• **Integer** – Single integer.<br>• **MultiField** – Stores string data that has multiple words.<br>• **QuadInteger** (eight bytes) – Stores whole numbers without decimal places. Valid values for the quad integer data type are -9,223,372,036,854,775,808 (negative nine quintillion) to +9,223,372,036,854,775,807 (positive nine quintillion).<br>• **Scaled** (two bytes) – Lets you store a four-byte float as a twobyte integer in the Historian archive. The scaled data type saves disk space but sacrifices data precision as a result.<br>• **Time** – Returns or sets the type of time stamping applied to data at collection time. |

| Property Name | Property Type | Description |
|---|---|---|
| | | • **UDoubleInteger (Unsigned Double Integer)** (four bytes) – Stores whole numbers without decimal places. Valid values for the unsigned double integer data type are 0 to 4,294,967, 295 (4.2 billion).<br>• **Undefined** – Data type is not defined.<br>• **UQuadInteger (Unsigned Quad Integer)** (eight bytes) – Stores whole numbers without decimal places. Valid values for the unsigned quad integer data type are 0 to 18,446,744,073,709,551,615 (19 quintillion).<br>• **USingleInteger (Unsigned Single Integer)** (two bytes) – Stores whole numbers without decimal places. Valid values for the unsigned single integer data type are 0 to 65535.<br>• **VariableString** (No fixed size) – Stores string values of undetermined size. This data type is useful if you cannot rely on a constant string length from your data source.<br>• **Array** – Returns an array of tags from your data source. You can specify orientation, size, and number of rows returned in the array. |
| ErrorCode | Error Code | Error Code Definition<br><br>See Error Code Definition *(on page 1251)* for more information. |
| Samples | Data Sample | See DataSample Parameter *(on page 1234)* for more information. |

## DataSample Parameter

The DataSample Parameter specifies the number of data samples to retrieve from the archive. Samples are evenly spaced within the time range defined by start time and end time for most sampling modes.

| Property Name | Property Type | Description |
|---|---|---|
| Value | String | Format for a multi-field tag like `{ "field1":"1","field2":"1000.0" }` (user-defined type tag). |

| Property Name | Property Type | Description |
|---|---|---|
| | | JavaScript code can parse the value string as a JSON object. All field values are string. |
| TimeStamp | DateTime | Start and end times of the query. If no start time is specified, the start time is two hours prior to running the query. If no EndTime is specified, the end time is the time the query runs. |
| Quality | Integer<br><br>(Enumerated value of DataQuality.StatusType) | Data type consisting of a set of named values called elements, members or enumerators of the type. Property values reflect quality as "quality is good" or " quality is bad".<br>**Value and Status**<br>• **0** – Bad<br>• **1** – Uncertain<br>• **2** – NA<br>• **3** – Good |

## SamplingModeType Parameter

The SamplingModeType parameter is the mode of sampling data from the archive. The default setting for the Sampling Mode is `Calculated`.

| Properties | Description | Value |
|---|---|---|
| Undefined | Sampling mode is not defined. | 0 |
| CurrentValue | Retrieves the current value. The time- interval criteria are ignored. | 1 |
| Interpolated | Retrieves evenly-spaced, interpolated values based on interval or NumberOfSamples and the time-frame criteria. | 2 |
| Trend | Returns the raw minimum and raw maximum value for each specified interval. | 3 |

| Properties | Description | Value |
|---|---|---|
| | Use the Trend sampling mode to maximize performance when retrieving data points for plotting. For the Trend sampling mode, if the sampling interval does not evenly divide by the interval length, Historian ignores any leftover values at the end, rather than putting them into a smaller interval. | |
| RawByTime | Retrieves raw archive values based on time-frame criteria. | 4 |
| RawByNumber | Retrieves raw archive values based on the StartTime criteria, the NumberOfSamples, and Direction criteria. The EndTime criteria is ignored for this sampling mode. | 5 |
| Calculated | Retrieves evenly spaced calculated values based on NumberOfSamples, interval, the time frame criteria, and the CalculationMode criteria. | 6 |
| Lab | Returns actual collected values without interpolation. | 7 |
| InterpolatedtoRaw | Provides raw data in place of interpolated data when the number of samples are fewer than the available samples. | 8 |
| TrendtoRaw | The TrendtoRaw sampling mode almost always produces the same results as the Trend sampling mode. However, when more samples are requested than there are raw data points, the TrendtoRaw sampling mode returns all available raw data points with no further processing. Use TrendtoRaw in place of Trend when this condition exists. | 9 |

| Properties | Description | Value |
|---|---|---|
| LabtoRaw | Provides raw data for the selected calculated data, when NumberOfSamples is less than the available samples. | 10 |
| RawByFilterToggle | Returns filtered time ranges using the following values:<br>• 1 – true<br>• 0 – false<br>This sampling mode is used with the time range and filter tag conditions. The response string starts with a starting time stamp and ends with an ending timestamp. | 11 |

## Direction Parameter

The Direction Parameter specifies the direction (Forward or Backward from the starting time) of data sampling from the archive. The default value is Forward.

| Direction | Value |
|---|---|
| Forward | 1 |
| Backward | 0 |

## CalculationModeType Parameter

The CalculationModeType parameter is only applied if the Sampling Mode is set to Calculated. It represents the type of calculation to use on the archive data. The default Calculation Mode, if none is specified, is Average.

| Calculation Mode Type | Description | Value |
|---|---|---|
| Undefined | Calculation mode is not defined. | 0 |
| Average | Retrieves the time-weighted average for each calculation interval. | 1 |

| Calculation Mode Type | Description | Value |
|---|---|---|
| StandardDeviation | Retrieves the time-weighted standard deviation for each calculation interval. | 2 |
| Total | Retrieves the time-weighted rate total for each calculation interval.<br><br>Use rate totals when working with a continuous measurement. Time weighting takes into account that compressed data is not evenly spaced in time. A factor must be applied to the total value to convert into appropriate engineering units. As a rate total, the default is Units/Day. If the actual units of the continuous measurement are Units/Minute, you would multiply the results by 1440 (minutes per day) to convert the total into appropriate engineering units. | 3 |
| Minimum | Retrieves the minimum value for each calculation interval. | 4 |
| Maximum | Retrieves the maximum value for each calculation interval. | 5 |
| Count | Counts the number of raw samples found with good quality in the interval.<br><br>Value is the count of raw samples with good quality in the interval. The values of each sample are ignored. The Count does not include any samples of bad quality, including the start and end of collection markers.<br><br>For Quality, the percentage of good samples is always 100, even if the interval does not contain any raw samples, or contains only bad quality samples.<br><br>Count is useful for analyzing the distribution of the raw data samples to determine the ef- | 6 |

| Calculation Mode Type | Description | Value |
|---|---|---|
| | fect of compression deadbands. It is also useful to determine which tags are consuming the most archive space. | |
| RawAverage | Retrieves the arithmetic average of all good quality raw samples for each calculation interval.<br><br>Value is the sum of all good quality samples in the interval, divided by the number of good quality samples in the interval. All bad quality samples are ignored. That is RawAverage is equivalent to RawTotal divided by the Count.<br><br>For Quality, if there are no raw samples in the interval or if they all are bad quality, then the percentage of good is 0. Otherwise, the percentage of good is always 100, even if the interval contains bad quality samples.<br><br>RawAverage is useful for calculating an accurate average when a sufficient number of raw samples are collected. | 7 |
| RawStandardDeviation | Retrieves the arithmetic standard deviation of raw values for each calculation interval.<br><br>For Value, any raw point of bad data quality is ignored.<br><br>For Quality, if there are no raw samples in the interval or they all have bad quality, then the percentage of good is 0. Otherwise, the percentage of good is always 100, even if the interval contains bad quality samples.<br><br>RawStandardDeviation is useful for calculating an accurate standard deviation when | 8 |

| Calculation Mode Type | Description | Value |
|---|---|---|
| | a sufficient number of raw samples are collected. | |
| RawTotal | Retrieves the arithmetic total (sum) of sampled values for each interval.<br><br>Value is the sum of the good quality values of all raw samples in the interval. All bad quality samples are ignored.<br><br>For Quality, the percentage of good samples is always 100, even if the interval does not contain any raw samples or it contains only bad quality samples.<br><br>If the same start and end times are used, and the time span is treated as a single interval, then all values are added together.<br><br>RawTotal is useful for calculating an accurate total when a sufficient number of raw samples are collected. Note that unlike ihTotal, this is a simple sum with no assumption that the values are rate values. | 9 |
| MinimumTime | Retrieves the timestamp of the minimum value found within each calculation interval. It can be a raw or an interpolated value. The minimum must be a good data quality sample. | 10 |
| MaximumTime | Retrieves the timestamp of the maximum value found within each calculation interval. It can be a raw or an interpolated value. The maximum must be a good data quality sample. | 11 |
| TimeGood | Retrieves the amount of time (milliseconds) during the interval when the data is of good quality and the filter condition is met. | 12 |

| Calculation Mode Type | Description | Value |
|---|---|---|
| StateCount | Retrieves the amount of time a tag uses to transition to another state from a previous state during a time interval. | 13 |
| StateTime | Retrieves the duration that a tag stayed in a given state within an interval. | 14 |
| OPCQAnd | Retrieves the OPCQAND, bit-wise AND operation of all the 16-bit OPC qualities of the raw samples stored in the specified interval.<br><br>Note that OPC Quality is a subfield for Quality-Value-Timestamp (QVT), so when this calculation mode is used, OPC Quality is considered for calculation. | 15 |
| OPCQOr | Retrieves the OPCQOR, bit-wise OR operation of all the 16-bit OPC qualities of the raw samples stored in the specified interval.<br><br>Note that OPC Quality is a subfield for Quality-Value-Timestamp (QVT), so when this calculation mode is used, OPC Quality is considered for calculation. | 16 |
| FirstRawValue | Retrieves the first good raw sample value for a given interval.<br><br>Value is the value of the raw sample, or zero if there are no good raw samples in the interval.<br><br>For Quality, if there are not good raw samples in the interval, then the percentage of good is 0. Otherwise, the percentage of good is always 100, even if the interval contains bad quality samples. Note that Quality is the same for FirstRawValue and FirstRawTime. | 17 |

| Calculation Mode Type | Description | Value |
|---|---|---|
| | The Raw sample has a quality of Good, Bad, or Uncertain, and that is converted to a 0 or 100 percent. | |
| FirstRawTime | Retrieves the first good raw timestamp for a given interval.<br><br>Value is the timestamp of the sample or the year 1969 if there are no good raw samples in the interval.<br><br>For Quality, if there are not good raw samples in the interval, then the percentage of good is 0. Otherwise, the percentage of good is always 100, even if the interval contains bad quality samples. Note that Quality is the same for FirstRawValue and FirstRawTime.<br><br>The Raw sample has a quality of Good, Bad, or Uncertain, and that is converted to a 0 or 100 percent. | 18 |
| LastRawValue | Retrieves the last good raw sample value for a given time interval.<br><br>Value is the value of the raw sample or zero if there are no good raw samples in the interval.<br><br>For Quality, if there are no good raw samples in the interval, the percentage of good samples is 0. Otherwise, the percentage of good is always 100, even if the interval contains bad samples. Note that Quality is the same for LastRawValue and LastRawTime.<br><br>The Raw sample has a quality of Good, Bad, or Uncertain, and that is converted to a 0 or 100 percent. | 19 |

| Calculation Mode Type | Description | Value |
|---|---|---|
| LastRawTime | Retrieves the last good timestamp of the last value for a given time interval.<br><br>Value is the timestamp of the sample or the year 1969 if there are no good raw samples in the interval.<br><br>For Quality, if there are no good raw samples in the interval, the percentage of good samples is 0. Otherwise, the percentage of good is always 100, even if the interval contains bad samples. Note that Quality is the same for LastRawValue and LastRawTime.<br><br>The Raw sample has a quality of Good, Bad, or Uncertain, and that is converted to a 0 or 100 percent. | 20 |
| TagStats | Retrieves the statistics for a tag from the archive stored in the specified interval. | 21 |

## FilterModeType Parameter

The FilterModeType parameter defines how time periods before and after transitions in the filter condition should be handled.

When the FilterModeType parameter is applied, then the Start time and End time are specified as:

- ExactTime
- BeforeTime
- AfterTime
- BeforeAndAfterTime

For example, AfterTime indicates that the filter condition should be True starting at the timestamp of the archive value that triggered the True condition, and leading up to the timestamp of the archive value that triggered the False condition.

| Properties | Description | Value |
|---|---|---|
| ExactTime | Retrieves data for the exact times that the filter condition is True. | 1 |
| BeforeTime | Retrieves data from the timestamp of the last False filter condition to the timestamp of the next True condition. | 2 |
| AfterTime | Retrieves data from the timestamp of the True filter condition to the time-stamp of the next False condition. | 3 |
| BeforeAndAfterTime | Retrieves data from the timestamp of the last False filter condition to the timestamp of the next False condition. | 4 |

## ReturnDataFields Parameter

The ReturnDataFields bitwise parameter specifies which data fields are returned in a query. Using it in a query returns data such as TimeStamp, and each field returns a Boolean value.

Each time-series data sample contains QVT (quality, value, and timestamp) values. If ReturnDataFields is not provided, then the default value of 0 is considered, and all QVT values are returned for each data sample. To return one of the data field properties, such as TimeStamp, use the TimeStamp option as shown in the table.

| Properties | Description | Field value (Boolean) |
|---|---|---|
| All Fields | Specifies that all data fields are returned in the query. | 0 (0000) |
| TimeStamp | The time stamp of the col-lected sample or an inter-val time stamp. When speci-fied in the query, returns the TimeStamp property. | 1 (0001) |
| Value | The collected value or sam-pled value; the data type of the value will be the same | 2 (0010) |

| Properties | Description | Field value (Boolean) |
|---|---|---|
| | data type as the tag's raw data. | |
| Quality | When specified in the query, returns the Quality property. Each sample in Current Value and Raw query retrieval has a quality of:<br>• Good (3)<br>• Not Available (2)<br>• Uncertain (1)<br>• Bad (0)<br>Interpolated and Lab Retrieval express quality as "percent good". | 4 (0100) |

## Payload Parameter

This parameter queries for the tag properties requested from the server.

Use the Payload parameter to query for all the tag properties to return from the server. In the Update Tag Configuration API, you must provide the actual tag property value. However, in the Get Tag Properties API, you must provide the property name and value of 1 (true), so the property can be read from the server and returned.

The properties listed in the following table are valid in APIs that use the Payload parameter, unless otherwise specified. For Property Names used in the Get Tag Properties API, the property name is always a Boolean (true/false) value, while it can be a string or integer for other APIs.

| Property Name | Property Type | Description |
|---|---|---|
| AllFields | Boolean | Used for Get Tag Properties API. |
| Name | Boolean, String | Used for the Get Tag Properties API, Add Single Tag API, and Add Bulk Tags API. |
| Description | String | |
| EngineeringUnits | String | |

| Property Name | Property Type | Description |
|---|---|---|
| Comment | String | |
| DataType : ihDataType | SignedIntegral | Type definition is an enumerated type "ihDataType".<br>{<br>ihDataTypeUndefined = 0,<br>ihScaled,<br>ihFloat,<br>ihDoubleFloat,<br>ihInteger,<br>ihDoubleInteger,<br>ihFixedString,<br>ihVariableString,<br>ihBlob,<br>ihTime,<br>ihInt64,<br>ihUInt64,<br>ihUInt32,<br>ihUInt16,<br>ihByte,<br>ihBool,<br>ihMultiField,<br>ihArray,<br>ihMaxDataType<br>} ihDataType; |
| FixedStringLength | UnsignedChar | |
| CollectorName | String | |
| SourceAddress | String | |
| CollectionType : ihCollectionType | SignedIntegral | Type definition is an enumerated type "ihCollectionType".<br>{<br>ihUnsolicited = 1,<br>ihPolled<br>} ihCollectionType; |

| Property Name | Property Type | Description |
|---|---|---|
| CollectionInterval | SignedInte-gral | |
| CollectionOffset | Unsigned-Long | |
| LoadBalancing | Boolean | |
| TimeStampType : ihTimeStamp-Type | SignedInte-gral | Type definition is an enumerated type "ihTimeS-tampType".<br>`{`<br>`ihSource = 1,`<br>`ihInterface,`<br>`} ihTimeStampType;` |
| HiEngineeringUnits | Double | |
| LoEngineeringUnits | Double | |
| InputScaling | Boolean | |
| HiScale | Double | |
| LoScale | Double | |
| CollectorCompression | Boolean | |
| CollectorDeadbandPercentRange | Float | |
| ArchiveCompression | Boolean | |
| ArchiveDeadbandPercentRange | Float | |
| General1 | String | |
| General2 | String | |
| General3 | String | |
| General4 | String | |
| General5 | String | |
| ReadSecurityGroup | String | |
| WriteSecurityGroup | String | |

| Property Name | Property Type | Description |
|---|---|---|
| AdministratorSecurityGroup | String | |
| LastModified | Boolean | Used for Get Tag Properties API. |
| LastModifiedUser | Boolean | Used for Get Tag Properties API. |
| InterfaceType | Boolean | Used for Get Tag Properties API. |
| CollectorType : ihInterfaceType | SignedIntegral | Type definition is an enumerated type "ihInterfaceType".<br>`{`<br>`ihInterfaceUndefined = 0,`<br>`ihIFix,`<br>`ihRandom,`<br>`ihOPC,`<br>`ihFile,`<br>`ihIFixLabData,`<br>`ihManualEntry,`<br>`ihOther,`<br>`ihCalcEngine,`<br>`ihServerToServer,`<br>`ihPI,`<br>`ihOPCAE,`<br>`ihCIMPE,`<br>`ihPIDistributor,`<br>`ihCIMME,`<br>`ihPerfTag,`<br>`ihCustom,`<br>`ihServerToServerDistributor,`<br>`ihWindowsPerfMon,`<br>`} ihInterfaceType;` |
| UTCBias | SignedIntegral | |
| AverageCollectionTime | Boolean | Used for Get Tag Properties API. |
| CalculationDependencies | StringArray | |
| CollectionDisabled | Boolean | |

| Property Name | Property Type | Description |
|---|---|---|
| ArchiveCompressionTimeout | Unsigned-Long | |
| CollectorCompressionTimeout | Unsigned-Long | |
| SpikeLogic | Boolean | |
| SpikeLogicOverride | Boolean | |
| CollectorAbsoluteDeadbanding | Boolean | |
| CollectorAbsoluteDeadband | Double | |
| ArchiveAbsoluteDeadbanding | Boolean | |
| ArchiveAbsoluteDeadband | Double | |
| StepValue | Boolean | |
| TimeResolution : ihTimeResolution | SignedIntegral | Type definition is an enumerated type "ihTimeResolution".<br>`{`<br>`ihSeconds = 0,`<br>`ihMilliseconds,`<br>`ihMicroseconds,`<br>`ihNanoseconds`<br>`} ihTimeResolution;` |
| ConditionCollectionEnabled | Boolean | |
| ConditionCollectionTriggerTag | String | |
| ConditionCollectionComparison : ihConditionCollectionComparison | SignedIntegral | Type definition is an enumerated type "ihConditionCollectionComparison".<br>`{`<br>`ihConditionComparisonUndefined = 0,`<br>`ihConditionComparisonEqual,`<br>`ihConditionComparisonLessThan,`<br>`ihConditionComparisonLessThanEqual,`<br>`ihConditionComparisonGreaterThan,`<br>`ihConditionComparisonGreaterThanEqual,` |

| Property Name | Property Type | Description |
|---|---|---|
| | | ihConditionComparisonNotEqual<br>} ihConditionCollectionComparison; |
| ConditionCollectionCompareValue | String | |
| ConditionCollectionMarkers | Boolean | |
| Calculation | String | When the Calculation field is used, then two more conditions are required. Calculation is not a specific field for a tag property. If the tag's collector or interface type is Server-to-server and the Calculation field is set (not Null), then the field value is set to the source address. |
| TagId | Boolean | Used for Get Tag Properties API. |
| EnumeratedSetName | String | |
| DataStoreName | String | |
| DefaultQueryModifiers | Long Long | |
| UserDefinedTypeName | String | |
| NumberOfElements | SignedIntegral | |
| DataDensity : ihTagDataDensity | SignedIntegral | Type definition is an enumerated type "ihTagDataDensity".<br>{<br>ihDataDensityUndefined = 0,<br>ihDataDensityMinimum = 1,<br>ihDataDensityMedium = 4,<br>ihDataDensityMaximum = 7<br>} ihTagDataDensity; |
| CalcType : ihTagCalcType | SignedIntegral | Type definition is an enumerated type "ihCalcType".<br>{<br>ihRawTag = 0,<br>ihAnalyticTag = 1, |

| Property Name | Property Type | Description |
|---|---|---|
| | | ihPythonExprTag = 2<br>} ihTagCalcType; |
| HasAlias | Boolean | Used for Get Tag Properties API. |
| IsStale | Boolean | Used for Get Tag Properties API. |

## Error Code Definitions

The following table provides the values and definitions for the ErrorCode parameter.

**Table 88. Error Code Definitions**

| Error Code Value: | Error Code Definition |
|---|---|
| Success = 0 | Operation successful. |
| Failed = -1 | Operation failed. |
| Timeout = -2 | Operation failed due to timeout. |
| NotConnected = -3 | Not connected to Historian server. |
| CollectorNotFound = -4 | The given collector does not exist on the server. |
| NotSupported = -5 | Operation not supported. |
| DuplicateData = -6 | Attempt to overwrite an existing data sample. |
| InvalidUsername = -7 | Bad user name or password. |
| AccessDenied = -8 | Insufficient permissions for operation. |
| WriteInFuture = -9 | Attempted data write too far in the future. |
| WriteArchiveOffline = -10 | Attempted data write to an offline archive. |
| WriteArchiveReadonly = -11 | Attempted data write to a read-only archive. |
| WriteOutsideActiveRange = -12 | Attempted data write beyond the configured active range. |
| WriteNoArchiveAvailable = -13 | Attempted data write with no available archives. |
| InvalidTagname = -14 | The requested tag was not found. |
| LicensedTagCountExceeded = -15 | Number of licensed tags exceeded. |

**Table 88. Error Code Definitions (continued)**

| Error Code Value: | Error Code Definition |
|---|---|
| LicensedConnectionCountExceeded = -16 | Number of licensed server connections exceeded. |
| InternalLicenseError = -17 | Internal license error. |
| NoValue = -18 | No available tag data. |
| DuplicateCollector = -19 | The given collector name already exists on the server. |
| NotLicensed = -20 | Server or feature is not licensed. |
| CircularReference = -21 | Circular reference detected in calculation. |
| BackupInsufficientSpace = -22 | Insufficient disk space to perform backup. |
| InvalidServerVersion = -23 | Operation unsupported due to server version. |
| QueryResultSizeExceeded = -24 | Upper limit on query results exceeded. |
| DeleteOutsideActiveRange = -25 | Attempted data delete outside allowed modification interval. |
| AlarmArchiverUnavailable = -26 | Alarms and Events subsystem unreachable. |
| ArgumentException = -27 | A supplied argument is invalid. |
| ArgumentNullException = -28 | A supplied argument is NULL. |
| ArgumentOutOfRangeException = -29 | A supplied argument is outside the valid range. |
| InvalidEnumeratedSet = -30 | The requested Enumerated Set was not found. |
| InvalidDataStore = -31 | The requested data store was not found. |
| NotPermitted = -32 | Operation not permitted. |
| InvalidCustomDataType = -33 | The Custom data type is not supported. |
| ihSTATUS_EXISTING_USERDEF_REFERENCES = -34 | N/A |
| ihSTATUS_INVALID_TAGNAME_DELETEDTAG = -35 | N/A |
| ihSTATUS_INVALID_DHS_NODENAME = -36 | N/A |
| ihSTATUS_DHS_SERVICE_IN_USE = -37 | N/A |

**Table 88. Error Code Definitions (continued)**

| Error Code Value: | Error Code Definition |
|---|---|
| ihSTATUS_DHS_STORAGE_IN_USE = -38 | N/A |
| ihSTATUS_DHS_TOO_MANY_NODES_IN_MIRROR = -39 | N/A |
| ihSTATUS_ARCHIVE_IN_SYNC = -40 | N/A |
| InvalidArchiveName= -41 | N/A |
| InvalidSession = 1 | Session id is invalid. |
| SessionExpired = 2 | Session has expired. |
| UnknownError = 3 | Unknown error, please check server log. |
| NoValidClientBufferManager= 4 | No valid client buffer manager. |
| NoValueInDataSet = 5 | No value in returned data set. |
| TagNotExisting = 6 | Tag doesn't exist. |
| ClientBufferManagerCommunicationError = 7 | Service call to central buffer server fail. |
| TagTypeNotSupported=8 | Tag type is not supported. |
| ValueTypeNotMatchTagDataType = 9 | Value type doesn't match tag data type. |
| InvalidParameter=10 | Invalid query parameter. |
| TagSearchResultIsHuge = 11 | Tag Search Criteria result was more than 5000. |
| InvalidHistorianServer=12 | No valid server or historian server name isn't in the server list. |
| ihSTATUS_INVALID_INTERFACETYPE = -49 | The collector type is not valid. For a list of collector types, refer to Collector Type and Subtype *(on page 1339)*. |
| ihSTATUS_INTERFACE_START_FAIL = -50 | Starting the collector has failed. |
| ihSTATUS_INTERFACE_STOP_FAIL = -51 | Stopping the collector has failed. |

# Historian REST APIs

## Overview of the Historian REST APIs

Historian provides REST APIs to manage Historian systems, collectors, data stores, and tags. In addition, it provides APIs to install and manage collector instances.

**Important:**
Port 443 is used in examples and sample code. If you copy and paste the sample code from Help, you must change this port to your installed port.

## Managing Systems

### The Get DHS Machines API

Using the Get DHS Machines API, you can view the list of DHS machines in a location.

| METHOD | GET |
|---|---|
| URI | `https://<historianservername>/historian-rest-api/v1/dhsmachines?storageName=` |
| SAMPLE QUERY PARAM GET URL | `https://<historianservername>/historian-rest-api/v1/dhsmachines?storageName=xx` |
| SAMPLE RESPONSE | {<br>"ErrorCode": 0,<br>"ErrorMessage": null,<br>"Data": [<br>{<br>"NodeName": "xyz",<br>"IsAlreadyAdded": true<br>} |

| | |
|---|---|
| | ```<br>]<br><br>}<br><br>}<br>``` |
| SAMPLE cURL COMMAND | ```<br>curl -i -H "Accept: application/json" -H<br>"Authorization: Bearer <TOKEN>”<br>https://<historianservername>/historian-rest-a<br>pi/v1/dhsmachines?storageName=xxx<br>``` |

**Table 89. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| `storageName` | The value of the location whose DHS machines you want to view. | Yes | String |

**Table 90. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |

### The Get DHS Services API

Using the Get DHS Services API, you can view the list of DHS services in a data store.

| | |
|---|---|
| METHOD | GET |
| URI | ```<br>https://<historianservername>/historian-rest-a<br>pi/v1/dhsservices?dHSServiceMask=&withReason=fa<br>lse<br>``` |
| SAMPLE QUERY PARAM GET URL | ```<br>https://<historianservername>/historian-rest-a<br>pi/v1/dhsservices?dHSServiceMask<br>=*&withReason=false<br>``` |

SAMPLE RESPONSE


```
{

    "ErrorCode": 0,

    "ErrorMessage": null,

    "Data": [

        {

            "LogicalName":
 "ConfigManager_NPI212611749M1",

            "NodeName": "NPI212611749M1",

            "ServiceType": 4,

            "Status": 1,

            "TCPPort": 14002

        },

        {

            "LogicalName":
 "DataArchiver_NPI212611749M1",

            "NodeName": "NPI212611749M1",

            "ServiceType": 2,

            "Status": 1,

            "TCPPort": 14001

        },
```

```
       {

           "LogicalName":
"ClientManager_NPI212611749M1",

           "NodeName": "NPI212611749M1",

           "ServiceType": 3,

           "Status": 1,

           "TCPPort": 14000

       },

       {

           "LogicalName":
"DiagnosticsManager_NPI212611749M1",

           "NodeName": "NPI212611749M1",

           "ServiceType": 5,

           "Status": 1,

           "TCPPort": 14003

       },

       {

           "LogicalName":
"DataArchiver_distmachine2",

           "NodeName": "distmachine2",
```

```
            "ServiceType": 2,

            "Status": 0,

            "TCPPort": 14001

        },

        {

            "LogicalName":
"DataArchiver_distmachine1",

            "NodeName": "distmachine1",

            "ServiceType": 2,

            "Status": 1,

            "TCPPort": 14001

        },

        {

            "LogicalName":
"ClientManager_distmachine1",

            "NodeName": "distmachine1",

            "ServiceType": 3,

            "Status": 1,

            "TCPPort": 14000
```

| | |
|---|---|
| | <pre>},<br><br>{<br><br>"LogicalName":<br>"DiagnosticsManager_distmachine1",<br><br>"NodeName": "distmachine1",<br><br>"ServiceType": 5,<br><br>"Status": 0,<br><br>"TCPPort": 14003<br><br>}<br><br>]<br><br>}<br><br>}</pre> |
| SAMPLE cURL COMMAND | <pre>curl -i -H "Accept: application/json" -H<br>"Authorization: Bearer <TOKEN>”<br>https://<historianservername>/historian-rest-a<br>pi/v1/dhsservices?dHSServiceMask=*&withReason=fa<br>lse</pre> |

**Table 91. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| `withReason` | Indicates whether the reason must be retrieved in the API response. | Yes | Boolean |
| `dHSService-Mask` | The value of the DHS service mask. | Yes | String |

**Table 92. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |

### The Get Server Properties API

Using the Get Server Properties API, you can view the list of properties of a server.

| | |
|---|---|
| METHOD | GET |
| URI | `https://<historianservername>/historian-rest-api/v1/serverproperties` |
| SAMPLE QUERY PARAM GET URL | `https://<historianservername>/historian-rest-api/v1/serverproperties` |
| SAMPLE RESPONSE | <pre>{<br>"ErrorCode": 0,<br>"ErrorMessage": null,<br>"Data": {<br>"Storages": [<br>{<br>"StorageName": "System Storage",<br>"StorageType": 2,<br>"NumberOfDataStores": 1,<br>"NumberOfArchivers": 0,<br>"DataStores": [</pre> |

```
                    "System"

                ],

                "Id":
"861C2743-72E0-46FC-9B31-90E28CC39B8D",

                "IsDefault": false,

                "LastModifiedUser": null,

                "LastModifiedTime":
"1970-01-01T00:00:00.000Z",

                "ArchiverServices": []

            },

            {

                "StorageName": "xyz",

                "StorageType": 0,

                "NumberOfDataStores": 3,

                "NumberOfArchivers": 1,

                "DataStores": [

                    "ScadaBuffer",

                    "DHSSystem",

                    "User"
```

```
            ],

            "Id":
"5F267DF3-879A-4222-8A0E-D31EDEA83C14",

            "IsDefault": true,

            "LastModifiedUser": null,

            "LastModifiedTime":
"1970-01-01T00:00:00.000Z",

            "ArchiverServices": [

                {

                    "LogicalName":
"DataArchiver_xyz",

                    "NodeName": "xyz",

                    "ServiceType": 2,

                    "IsAlreadyAdded": true,

                    "TCPPort": 14001

                }

            ]

        }

    ],

    "Servers": [
```

```
          {

              "LogicalName":
"DataArchiver_xyz0",

              "NodeName": "xyz",

              "ServiceType": 2,

              "Status": 1,

              "TCPPort": 14001,

              "MemoryVMSize": "4778",

              "TotalFailedWrites": "0",

              "WriteCacheHitRatio": "0.748",

              "TotalOutOfOrder": "3",

              "CompressionRatio": "0.321",

              "ReadQueueSize": "0",

              "WriteQueueSize": "0",

              "MsgQueueSize": "0",

              "ReadQueueProcessingRate": "1",

              "WriteQueueProcessingRate":
"31",

              "MsgQueueProcessingRate": "0"

          }
```

| | |
|---|---|
| | ```<br>]<br><br>}<br><br>}<br>``` |
| SAMPLE cURL COMMAND | ```<br>curl -i -H "Accept: application/json" -H<br>"Authorization: Bearer <TOKEN>”<br>https://<historianservername>/historian-rest-a<br>pi/v1/serverproperties<br>``` |

**Table 93. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |

### The Get System Statistics API

Using the Get System Statistics API, you can view the statistics of a system.

| | |
|---|---|
| METHOD | GET |
| URI | ```<br>https://<historianservername>/historian-rest-a<br>pi/v1/systemstats<br>``` |
| SAMPLE QUERY PARAM GET URL | ```<br>https://<historianservername>/historian-rest-a<br>pi/v1/systemstats<br>``` |
| SAMPLE RESPONSE | ```<br>{<br><br>"ErrorCode": 0,<br><br>"ErrorMessage": null,<br><br>"Data": {<br><br>"Utilization": {<br>``` |

```
            "WriteCacheHitRatio": "0.499",
            "SpaceConsumptionRate": "",
            "CompressionRatio": "0.199",
            "ReadQueueSize": "0",
            "WriteQueueSize": "0",
            "MsgQueueSize": "0",
            "ReadQueueProcessRate": "3",
            "WriteQueueProcessRate": "0",
            "MsgQueueProcessRate": "0",
            "MemoryVMUsage": "62",
            "OutOfOrderRate": "0",
            "ReadThreadUsage": "0",
            "WriteThreadUsage": "0",
            "FailedWriteRate": "0",
            "DiskFreeSpace": "59828"
        },
        "AlarmEvents": {
            "AverageAlarmRate": ""
```

<table>
<tr><td></td><td>

```
        },

        "TotalCollectors": {

            "TotalCollectors": 1,

            "RunningCollectors": 1,

            "StoppedCollectors": 0,

            "UnknownCollectors": 0

        },

        "Licence": {

            "ActualDataStores": 3,

            "MaxDataStores": 200,

            "ActualTags": 0,

            "MaxTags": 2147483647,

            "ActualUsers": 0,

            "MaxUsers": 1000

        }

    }

}
```

</td></tr>
<tr><td>SAMPLE cURL COMMAND</td><td>

```
curl -i -H "Accept: application/json" -H
 "Authorization: Bearer <TOKEN>”
```

</td></tr>
</table>

| | |
|---|---|
| | `https://<historianservername>/historian-rest-api/systemstats` |

**Table 94. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |

### The Get Read Sample and Receive Rate API

Using the Get Read Sample and Receive Rate API, you can view the read rate and receive rate of a system.

| | |
|---|---|
| METHOD | GET |
| URI | Read Sample Rate<br>`https://<historianservername>/historian-rest-api/v1/performancecounter/perftagdata/PerfTag_AverageEventRate/-/-/starttime/endtime/interval`<br>Receive Rate<br>`https://<historianservername>/historian-rest-api/v1/performancecounter/perftagdata/PerfTag_AverageReadRawRate/-/-/starttime/endtime/interval` |
| SAMPLE GET URI | `https://<historianservername>/historian-rest-api/v1/performancecounter/perftagdata/PerfTag_AverageEventRate/-/-/2020-12-15T11:19:01.719Z/2020-12-15T12:19:01.719Z/360000` |
| SAMPLE RESPONSE | `{`<br>`"ErrorCode": 0,` |

```
   "ErrorMessage": null,

   "Data": [

       {

           "TagName":
"PerfTag_AverageEventRate",

           "ErrorCode": 0,

           "DataType": "DoubleFloat",

           "Samples": [

               {

                   "TimeStamp":
"2020-11-18T05:35:22.612Z",

                   "Value": "0",

                   "Quality": 0

               },

               {

                   "TimeStamp":
"2020-11-18T05:47:22.612Z",

                   "Value": "0",

                   "Quality": 0

               },
```

```
            {

                "TimeStamp":
"2020-11-18T05:53:22.612Z",

                "Value": "0",

                "Quality": 0

            },

            {

                "TimeStamp":
"2020-11-18T06:11:22.612Z",

                "Value": "0",

                "Quality": 0

            },

            {

                "TimeStamp":
"2020-11-18T06:29:22.612Z",

                "Value": "0",

                "Quality": 0

            }

        ]

    }
```

| | |
|---|---|
| | ```<br>    ]<br><br>}<br>``` |
| SAMPLE cURL COMMAND | ```<br>curl -i -H "Accept: application/json"<br> -H "Authorization: Bearer <TOKEN>”<br> https://<historianservername>/historian-rest-a<br>pi/v1/performancecounter/perftagdata/<br>PerfTag_AverageEventRate/-/-/2020-12-15T11:19:01<br>.719Z/2020-12-15T12:19:01.719Z/360000<br>``` |

**Table 95. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |

### The Get Storages API

Using the Get Storages API, you can view the list of locations in a system.

| | |
|---|---|
| METHOD | GET |
| URI | ```<br>https://<historianservername>/historian-rest-a<br>pi/v1/storages?storageMask=<br>``` |
| SAMPLE QUERY PARAM GET URL | ```<br>https://<historianservername>/historian-rest-a<br>pi/v1/storages?storageMask=*<br>``` |
| SAMPLE RESPONSE | ```<br>{<br><br>    "ErrorCode": 0,<br><br>    "ErrorMessage": null,<br><br>    "Data": [<br><br>        {<br>``` |

```
            "StorageName": "System Storage",

            "StorageType": 2,

            "NumberOfDataStores": 1,

            "NumberOfArchivers": 0,

            "DataStores": [

                "System"

            ],

            "Id":
"861C2743-72E0-46FC-9B31-90E28CC39B8D",

            "IsDefault": false,

            "LastModifiedUser": null,

            "LastModifiedTime":
"1970-01-01T00:00:00.000Z",

            "ArchiverServices": []

        },

        {

            "StorageName": "srinivaswin10",

            "StorageType": 0,

            "NumberOfDataStores": 3,
```

```
            "NumberOfArchivers": 1,

            "DataStores": [

                "ScadaBuffer",

                "DHSSystem",

                "User"

            ],

            "Id":
"5F267DF3-879A-4222-8A0E-D31EDEA83C14",

            "IsDefault": true,

            "LastModifiedUser": null,

            "LastModifiedTime":
"1970-01-01T00:00:00.000Z",

            "ArchiverServices": [

                {

                    "LogicalName":
"DataArchiver_xyz",

                    "NodeName": "xyz",

                    "ServiceType": 2,

                    "TCPPort": 14001

                }
```

| | |
|---|---|
| | ```<br>]<br><br>}<br><br>]<br><br>}<br>``` |
| SAMPLE cURL COMMAND | ```<br>curl -i -H "Accept: application/json" -H<br>"Authorization: Bearer <TOKEN>"<br>https://<historianservername>/historian-rest-a<br>pi/v1/storages?storageMask=*<br>``` |

**Table 96. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| `storageMask` | The value of the location mask. | No | String |

**Table 97. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |

### The Add Machine API

Using the Add Machine API, you can add a server in a Historian system.

| | |
|---|---|
| METHOD | POST |
| URI | ```<br>https://<historianservername>/historian-rest-a<br>pi/v1/machine<br>``` |
| SAMPLE URI | ```<br>https://<historianservername>/historian-rest-a<br>pi/v1/machine<br><br>Payload<br><br>{<br>``` |

| | |
|---|---|
| | `"nodeName": "node1"`<br>`}` |
| SAMPLE RESPONSE | `{`<br><br>`"ErrorCode": 0,`<br><br>`"ErrorMessage": null`<br><br>`}` |
| SAMPLE cURL COMMAND | `curl -i -H "Accept: application/json" -i`<br>` -H "Content-Type: application/json" -H`<br>` "Authorization: Bearer <TOKEN>"`<br>`-d "{ \"nodeName \":\"name\"}" -X POST`<br>` https://<historianservername>/historian-rest-a`<br>`pi/v1/machine` |

**Table 98. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| `Payload` | Contains the machine name of the server that you want to add. | Yes | Multiple |

**Table 99. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |

### The Delete Machine API

Using the Delete Machine API, you can remove a server from a Historian system.

| | |
|---|---|
| METHOD | DELETE |
| URI | `https://<historianservername>/historian-rest-a`<br>`pi/v1/machine` |

| | |
|---|---|
| SAMPLE URI | `https://<historianservername>/historian-rest-api/v1/machine`<br><br>`Payload`<br><br>`{`<br>`"nodeName": "",`<br><br>`}` |
| SAMPLE RESPONSE | `{`<br><br>`"ErrorCode": 0,`<br><br>`"ErrorMessage": null`<br><br>`}` |
| SAMPLE cURL COMMAND | `curl -i -H "Accept: application/json" -i -H "Content-Type: application/json" -H "Authorization: Bearer <TOKEN>” -d “{ \”nodeName \”:\”name\”}” -X DELETE https://<historianservername>/historian-rest-api/v1/machine` |

**Table 100. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| `Payload` | Contains the name of the machine that you want to remove. | Yes | Multiple |

**Table 101. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |

### The Create Mirror Group API

Using the Create Mirror Group API, you can create a mirror group.

| | |
|---|---|
| METHOD | POST |
| URI | `https://<historianservername>/historian-rest-api/v1/mirrorgroup` |
| SAMPLE URI | `https://<historianservername>/historian-rest-api/v1/mirrorgroup`<br><br>`Payload`<br><br>`{`<br>`"mirrorStorageName": "storagename",`<br><br>`"nodes": "node1;node2"`<br><br>`}` |
| SAMPLE RESPONSE | `{`<br><br>`"ErrorCode": 0,`<br><br>`"ErrorMessage": null`<br><br>`}` |
| SAMPLE cURL COMMAND | `curl -i -H "Accept: application/json" -i -H "Content-Type: application/json" -H "Authorization: Bearer <TOKEN>” -d “{ \” mirrorStorageName \”:\”name\”,\" nodes \": \"xx;yy\"}” -X POST https://<historianservername>/historian-rest-api/v1/mirrorgroup` |

**Table 102. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| `Payload` | Contains the mirror group name and the servers you want to add to the group. | Yes | Multiple |

**Table 103. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |

### The Add Mirror Machine API

Using the Add Mirror Machine API, you can add a server to a mirror group.

| | |
|---|---|
| METHOD | POST |
| URI | `https://<historianservername>/historian-rest-api/v1/mirrormachine` |
| SAMPLE URI | `https://<historianservername>/historian-rest-api/v1/mirrormachine`<br><br>`Payload`<br><br>`{`<br>`"mirrorStorageName": "Mirror2",`<br><br>`"mirrorMachineName": "distmachine1"`<br><br>`}` |
| SAMPLE RESPONSE | `{`<br><br>`"ErrorCode": 0,`<br><br>`"ErrorMessage": null` |

| | |
|---|---|
| | }<br> |
| SAMPLE cURL COMMAND | curl -i -H "Accept: application/json" -i<br>-H "Content-Type: application/json" -H<br>"Authorization: Bearer <TOKEN>”<br>-d “{ \” mirrorStorageName \”:\”name\”,\"<br>nodes \": \"xx;yy\"}” -X POST<br>https://<historianservername>/historian-rest-a<br>pi/v1/mirrormachine |

**Table 104. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| `Payload` | Contains the machine name of the server that you want to add. | Yes | Multiple |

**Table 105. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |

### The Mirror Group Update API

Using the Mirror Group Update API, you can update the name of a mirror group.

| | |
|---|---|
| METHOD | PUT |
| URI | https://<historianservername>/historian-rest-a<br>pi/v1/mirrorgroup |
| SAMPLE URI | https://<historianservername>/historian-rest-a<br>pi/v1/mirrorgroup<br><br>Payload<br><br>{<br>"mirrorStorageName": "Mirror2", |

| | |
|---|---|
| | <pre>"mirrorStorageNewName": "Mirror3"<br><br>}</pre> |
| SAMPLE RESPONSE | <pre>{<br><br>"ErrorCode": 0,<br><br>"ErrorMessage": null<br><br>}</pre> |
| SAMPLE cURL COMMAND | <pre>curl -i -H "Accept: application/json" -i<br>-H "Content-Type: application/json" -H<br>"Authorization: Bearer <TOKEN>”<br>-d “{ \”mirrorStorageName \”:\”name\”,\"<br>mirrorStorageNewName \": \"sname\"}” -X PUT<br>https://<historianservername>/historian-rest-a<br>pi/v1/mirrorgroup</pre> |

**Table 106. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| `Payload` | Contains the existing and new names of the mirror group that you want to rename. | Yes | Multiple |

**Table 107. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |

### The Delete Mirror Machine API

Using the Delete Mirror Machine API, you can remove a server from a mirror group.

| METHOD | DELETE |
|---|---|
| URI | `https://<historianservername>/historian-rest-api/v1/mirrormachine` |
| SAMPLE URI | `https://<historianservername>/historian-rest-api/v1/mirrormachine`<br><br>`Payload`<br><br>`{`<br><br>`"mirrorStorageName": "Mirror 2",`<br><br>`"mirrorMachineName": "distmachine1"`<br>`}` |
| SAMPLE RESPONSE | `{`<br><br>`"ErrorCode": 0,`<br><br>`"ErrorMessage": null`<br><br>`}` |
| SAMPLE cURL COMMAND | `curl -i -H "Accept: application/json" -i -H "Content-Type: application/json" -H "Authorization: Bearer <TOKEN>” -d “{ \”mirrorStorageName \”:\”name\”, \” mirrorMachineName\”:\”name\”}” -X DELETE https://<historianservername>/historian-rest-api/v1/mirrormachine` |

**Table 108. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| `Payload` | Contains the name of the machine that you want to remove | Yes | Multiple |

**Table 108. Query Parameters (continued)**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| | and the name of the mirror group. | | |

**Table 109. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |

### The Delete Mirror Group API

Using the Delete Mirror Group API, you can delete a mirror group.

| | |
|---|---|
| METHOD | DELETE |
| URI | `https://<historianservername>/historian-rest-api/v1/mirrorgroup` |
| SAMPLE URI | `https://<historianservername>/historian-rest-api/v1/mirrorgroup`<br><br>`Payload`<br><br>`{`<br>`"mirrorStorageName": "Mirror3",`<br>`}` |
| SAMPLE RESPONSE | `{`<br><br>`"ErrorCode": 0,`<br><br>`"ErrorMessage": null`<br><br>`}` |

| | |
|---|---|
| SAMPLE cURL COMMAND | `curl -i -H "Accept: application/json" -i -H "Content-Type: application/json" -H "Authorization: Bearer <TOKEN>” -d “{ \”mirrorStorageName \”:\”name\”}” -X DELETE https://<historianservername>/historian-rest-api/v1/mirrorgroup` |

**Table 110. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| `Payload` | Contains the name of the machine that you want to remove. | Yes | Multiple |

**Table 111. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |

### The Local OPC Servers API

Using the Local OPC Servers API, you can view the list of OPC servers installed on a specified machine.

| | |
|---|---|
| METHOD | GET |
| URI | `http://<historianservername>/v1/localopcservers/<machine name>` |
| SAMPLE QUERY PARAM GET URL | `http://<historianservername>/v1/localopcservers/<machine name>` |
| SAMPLE RESPONSE | `{ "ErrorCode": 0, "ErrorMessage": null, "ServerIDs": [ "ID1", "ID2 " ]` |

| | |
|---|---|
| | `}` |
| SAMPLE cURL COMMAND | `curl -i -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>" https://<historianservername>/historian-rest-api/v1/localopcservers/xyz` |

**Table 112. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| `machine name` | The machine name of the OPC server. | Yes | String |

**Table 113. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |

### The Local OPC AE Servers API

Using the Local OPC AE Servers API, you can view the list of OPC Alarms and Events servers installed on a specified machine.

| | |
|---|---|
| METHOD | GET |
| URI | `http://<historianservername>/v1/localopcaeservers/<machine name>` |
| SAMPLE QUERY PARAM GET URL | `http://<historianservername>/v1/localopcaeservers/<machine name>` |
| SAMPLE RESPONSE | `{`<br>`"ErrorCode": 0,`<br>`"ErrorMessage": null,`<br>`"ServerIDs": [`<br>`"ID1",`<br>`"ID2 "`<br>`]` |

| | |
|---|---|
| | `}` |
| SAMPLE cURL COMMAND | `curl -i -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>" https://<historianservername>/historian-rest-api/v1/localopcaeservers/abc` |

**Table 114. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| `machine name` | The machine name of the OPC Alarms and Events server. | Yes | String |

**Table 115. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |

### The Local OPC HDA Servers API

Using the Local OPC HDA Servers API, you can view the list of OPC HDA servers installed on a specified machine.

| | |
|---|---|
| METHOD | GET |
| URI | `http://<historianservername>/v1/localopchdaservers/<machine name>` |
| SAMPLE QUERY PARAM GET URL | `http://<historianservername>/v1/localopchdaservers/<machine name>` |
| SAMPLE RESPONSE | `{ "ErrorCode": 0, "ErrorMessage": null, "ServerIDs": [ "ID1", "ID2 " ]` |

| | |
|---|---|
| | `}` |
| SAMPLE cURL COMMAND | `curl -i -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>" https://<historianservername>/historian-rest-api/v1/localopchdaservers/xyz` |

**Table 116. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| `machine name` | The machine name of the OPC HDA server. | Yes | String |

**Table 117. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |

### The DHS Service Port Update API

Using the DHS Service Port Update API, you can change the port and other details of a DHS service.

| | |
|---|---|
| METHOD | PUT |
| URI | `https://<historianservername>/historian-rest-api/v1/dhsservice/<DHS service name>` |
| SAMPLE URI | `https://<historianservername>/historian-rest-api/v1/dhsservice/ DataArchiver_xxx`<br><br>`{`<br><br>`"LogicalName": "DataArchiver_xxx",`<br><br>`"NodeName": "xxx",` |

<table>
<tr><td></td><td><pre>
   "ServiceType": 2,

   "Status": 1,

   "TCPPort": 14005

}
</pre></td></tr>
<tr><td>SAMPLE RESPONSE</td><td><pre>
{

    "ErrorCode": 0,

    "ErrorMessage": null,

    "Data": {

        "LogicalName": "DataArchiver_xxx",

        "NodeName": "xxx",

        "ServiceType": 2,

        "Status": 1,

        "TCPPort": 14005

    }

}
</pre></td></tr>
<tr><td>SAMPLE cURL COMMAND</td><td><pre>
curl -i -H "Accept: application/json" -i -H
 "Content-Type: application/json"
-H "Authorization: Bearer <TOKEN>” -d “{ \”
 LogicalName \”:\” DataArchiver_xxx \”,
\" NodeName \": \"xxx\"\”,\" ServiceType \":
 2,\" Status\": 1, \" TCPPort \": 14005}” -X PUT
https://<historianservername>/historian-rest-a
pi/v1/dhsservice/DataArchiver_xxx
</pre></td></tr>
</table>

**Table 118. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| `Payload` | Contains the values of the attributes of the data store that you want to change. | Yes | Multiple |

**Table 119. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |

# Managing Historian Model

### The Add Object Type API

Using this API, you can create an object type.

| | |
|---|---|
| METHOD | POST |
| URI | `https://<historianservername>/historian-model/v2/objectTypes` |
| SAMPLE QUERY PARAM POST URL | `https://localhost:443/historian-model/v2/objectTypes` |
| SAMPLE PAYLOAD | <pre>[<br>{<br>"name":"Vehicle",<br>"description":"Vehicle description",<br>"dataVariables":[],<br>"containedObjectTypes":[],<br>"objectTypeTemplates":[],<br>"lastModifiedUser":"",<br>"lastmodifiedTime":"",<br>"versionNumber":0<br>}<br>]</pre> |

**Table 120. Payload Parameters**

| Parameter | Description | Required? | Data Type |
|---|---|---|---|
| `name` | The object type name (must be unique). | Yes | String |
| `description` | The description of the object type. | No | String |
| `containedObjectTypes` | The list of contained types with details | No | Array |
| `objectTypeTemplates` | The list of object type templates, along with their details. | No | Array |
| `lastModifiedUser` | The username of the user who created the object type. | Yes | String |
| `lastModifiedTime` | The current time. | Yes | String |
| `versionNumber` | The version number of the object type. Start with 1 and increment by 1 each time you want to modify the object type. | Yes | Number |

**Table 121. Response Parameters**

| Parameter | Data Type | Description |
|---|---|---|
| `detail` | String | Description of the error. |
| `error_code` | Integer | The error code. A value of 0 indicates that the operation was successful. |
| `status` | String | The http request status. |
| `title` | String | The title of the error. |
| `type` | String | The type of the error. |

### The View Object Type API

Using this API, you can access an object type.

| | |
|---|---|
| METHOD | POST |

| | |
|---|---|
| URI | `https://<historianservername>/historian-model/v2/objectTypes/info` |
| SAMPLE QUERY PARAM POST URL | `https://localhost:443/historian-model/v2/objectTypes/info` |
| SAMPLE RESPONSE | {<br>"containedObjectTypes":[],<br>"dataVariables":[],<br>"description":"Vehicle description",<br>"lastModifiedUser":"XYZ.admin",<br>"lastmodifiedTime":"2022-01-21 08:58:20",<br>"name":"Vehicle",<br>"objectTypeTemplates":[{"containedObjectTypes":[],<br>"dataVariables":[],<br>"defaultTemplate":true,<br>"description":"Default_Template_Vehicle",<br>"lastModifiedUser":"XYZ.admin",<br>"lastmodifiedTime":"2022-01-21 08:58:20",<br>"name":"Default_Template_Vehicle",<br>"objectType":"Vehicle",<br>"substituteParameters":[],<br>"versionNumber":3}],<br>"versionNumber":3<br>} |
| SAMPLE PAYLOAD | `{"databaseName":"XYZ","name":"Vehicle"}` |

**Table 122. Payload Parameters**

| Parameter | Description | Required? | Data Type |
|---|---|---|---|
| `name` | The object type name (must be unique). | Yes | String |
| `databaseName` | The name of the Historian server. | Yes | String |

**Table 123. Response Parameters**

| Parameter | Data Type | Description |
|---|---|---|
| `name` | String | The object type name (must be unique). |
| `description` | String | The description of the object type. |
| `containedObject-Types` | Array | The list of contained types with details |
| `dataVariables` | Array | The list of variables (with details) in the object type. |
| `objectTypeTem-plates` | Array | The list of object type templates, along with their details. |
| `lastModifiedUser` | String | The username of the user who created the object type. |
| `lastModifiedTime` | String | The current time. |
| `versionNumber` | Number | The version number of the object type. Start with 1 and increment by 1 each time you want to modify the object type. |

**Table 124. objectTypeTemplates Array Parameters**

| Parameter | Data Type | Description |
|---|---|---|
| `name` | String | The object type name (must be unique). |
| `description` | String | The description of the object type. |
| `containedObject-Types` | Array | The list of contained types with details |
| `dataVariables` | Array | The list of variables (with details) in the object type. |
| `lastModifiedUser` | String | The username of the user who created the object type. |
| `lastModifiedTime` | String | The current time. |
| `versionNumber` | Number | The version number of the object type. Start with 1 and increment by 1 each |

**Table 124. objectTypeTemplates Array Parameters (continued)**

| Parameter | Data Type | Description |
|---|---|---|
| | | time you want to modify the object type. |
| `ObjectType` | String | The object type name. |
| `substitutePara-meters` | Array | Blank (not applicable) |

### The Update Object Type API

Using this API, you can modify an object type.

| | |
|---|---|
| METHOD | PUT |
| URI | `https://<historianservername>/histori-an-model/v2/objectTypes` |
| SAMPLE QUERY PARAM POST URL | `https://localhost:443/historian-mod-el/v2/objectTypes` |
| SAMPLE RESPONSE | Error code and message, specifying whether the object type has been modified (200 for success and other codes for failure). |
| HEADER | • Content type: multipart/form-data<br>• Boundary: ----WebKitFormBoundary-KeyName |
| SAMPLE PAYLOAD | `------WebKitFormBoundaryViBtYYvOfWhTYyhH`<br>`Content-Disposition: form-data;`<br>` name="objectTypeInfo"; filename="blob"`<br>`Content-Type: application/json`<br><br>`{`<br>`"containedObjectTypes":[],`<br>`"dataVariables":`<br>`[`<br>`{`<br>`"baseType":"NUMBER",`<br>`"description":"",` |

```
"name":"Speed",
"variableType":"Direct"
},
{
"baseType":"STRING",
"description":"",
"name":"Type",
"variableType":"InDirect"
}
],
"description":"Vehicle description",
"lastModifiedUser":"XYZ.admin",
"lastmodifiedTime":"2022-01-21 11:37:56",
"name":"Vehicle",
"objectTypeTemplates":
[
{
"containedObjectTypes":[],
"dataVariables":
[
{
"baseType":"NUMBER",
"description":"",
"name":"Speed",
"properties":
{"blockType":"AI",
"propertyDefinition":[]
},
"required":true,
"templateTagName":"",
"variableType":"DIRECT"
},
{
"baseType":"STRING",
"description":"",
"name":"Type",
"properties":
```

```
{
"blockType":"AI",
"propertyDefinition":[]
},
"required":true,
"templateTagName":"",
"variableType":"INDIRECT"
}
],
"defaultTemplate":true,
"description":"Default_Template_Vehicle",
"lastModifiedUser":"XYZ.admin",
"lastmodifiedTime":"2022-01-21 11:37:56",
"name":"Default_Template_Vehicle",
"objectType":"Vehicle",
"substituteParameters":[],
"versionNumber":5
}
],
"versionNumber":5,
"databaseName":"XYZ",
"containedObjectTypesChanges":
[
{
"newName":"",
"oldName":"Engine"
}
],
"dataVariablesChanges":
[
{
"newName":"",
"oldName":"Color"
}
],
"objectTypeTemplatesChanges":
[
```

| | |
|---|---|
| | <pre>{<br>"newName":"",<br>"oldName":"Car"<br>}<br>]<br>}<br>------WebKitFormBoundaryViBtYYvOfWhTYyhH--</pre> |

**Table 125. Payload Parameters**

| Parameter | Description | Required? | Data Type |
|---|---|---|---|
| `name` | The object type name (must be unique). | Yes | String |
| `description` | The description of the object type. | No | String |
| `dataVariables` | The list of variables in the object type. | No | Array |
| `containedObjectTypes` | The list of contained types with details | No | Array |
| `objectTypeTemplates` | The list of object type templates, along with their details. | No | Array |
| `lastModifiedUser` | The username of the user who created the object type. | Yes | String |
| `lastModifiedTime` | The current time. | Yes | String |
| `versionNumber` | The version number of the object type. Start with 1 and increment by 1 each time you want to modify the object type. | Yes | Number |

**Table 126. Response Parameters**

| Parameter | Data Type | Description |
|---|---|---|
| `name` | String | The object type name (must be unique). |
| `description` | String | The description of the object type. |

**Table 126. Response Parameters (continued)**

| Parameter | Data Type | Description |
|---|---|---|
| `containedObject-Types` | Array | The list of contained types with details. |
| `dataVariables` | Array | The list of variables (with details) in the object type. |
| `objectTypeTem-plates` | Array | The list of object type templates, along with their details. |
| `lastModifiedUser` | String | The username of the user who created the object type. |
| `lastModifiedTime` | String | The current time. |
| `versionNumber` | Number | The version number of the object type. Start with 1 and increment by 1 each time you want to modify the object type. |
| `databaseName` | The name of the Historian server. | Yes |
| `containedObject-TypeChanges` | The list of changes in the contained type. | No |
| `dataVariables-Changes` | The list of changes in the object type variables. | No |
| `objectTypeTem-platesChanges` | The list of changes in the object type templates. | No |

**Table 127. objectTypeTemplates Array Parameters**

| Parameter | Description | Data Type |
|---|---|---|
| `name` | The object type name (must be unique). | String |
| `description` | The description of the object type. | String |
| `containedObject-Types` | The list of contained types with details | Array |

**Table 127. objectTypeTemplates Array Parameters (continued)**

| Parameter | Description | Data Type |
|---|---|---|
| `dataVariables` | The list of variables (with details) in the object type. | Array |
| `lastModifiedUser` | The username of the user who created the object type. | String |
| `lastModifiedTime` | The current time. | String |
| `versionNumber` | The version number of the object type. Start with 1 and increment by 1 each time you want to modify the object type. | Number |
| `ObjectType` | The object type name. | String |
| `substitutePara-meters` | Blank (not applicable) | Array |

**Table 128. dataVariables Array Parameters**

| Parameter | Description | Re-quired? | Data Type |
|---|---|---|---|
| `name` | The name of the object type name (must be unique). | Yes | String |
| `description` | The description of the object type. | No | String |
| `variableType` | The type of the variable (direct, indirect, or static). | No | String |
| `baseType` | Not applicable | No | String |
| `dbTagName` | Not applicable | No | String |

**Table 129. containedObjectTypes Array Parameters**

| Parameter | Description | Re-quired? | Data Type |
|---|---|---|---|
| `name` | The name of the object type (must be unique). | Yes | String |
| `description` | The description of the object type). | No | String |
| `containedOb-jectType` | The name of the object type that you want to use as a contained type. | No | String |

**Table 129. containedObjectTypes Array Parameters (continued)**

| Parameter | Description | Required? | Data Type |
|---|---|---|---|
| `baseType` | The type of the contained type (ThingName indicates a contained type). | No | String |

**Table 130. Response Parameters**

| Parameter | Data Type | Description |
|---|---|---|
| `detail` | String | The description of the error. |
| `error_code` | Integer | The error code. A value of 0 indicates that the operation was successful. |
| `status` | String | The http request status. |
| `title` | String | The title of the error. |
| `type` | String | The type of the error. |

### The Contained Type List API

Using this API, you can get a list of contained types in each object type.

| | |
|---|---|
| METHOD | GET |
| URI | `https://<historianservername> /historian-model/v2/ containedObjectTypesMap` |
| SAMPLE QUERY PARAM GET URL | `https://localhost:443/historian-model/v2/containedObjectTypesMap` |

**Table 131. Response Parameters**

| Parameter | Description | Data Type |
|---|---|---|
| `containedObjectTypes` | The list of contained types. | Array |
| `name` | The object type in which the contained type is used. | String |

### The Create Object Instance API

Using this API, you can create an object instance.

| | |
|---|---|
| METHOD | POST |
| URI | `https://<historianservername> /historian-model/v2/objects` |
| SAMPLE QUERY PARAM POST URL | `https://localhost:443/historian-model/v2/objects` |
| SAMPLE PAYLOAD | [<br>{<br>"containedObjects":[],<br>"containedType":false,<br>"dataVariables":[],<br>"databaseName":"XYZ",<br>"description":"",<br>"fullName":"",<br>"name":"Audi1",<br>"objectDataValueChanges":[],<br>"objectDataValues":[],<br>"objectType":"Vehicle",<br>"objectTypeTemplate":"Car",<br>"override":false,<br>"parentName":"",<br>"substituteParameters":[],<br>"v1Type":false,<br>"versionNumber":0<br>}<br>] |

**Table 132. Payload Parameters**

| Parameter | Description | Required? | Data Type |
|---|---|---|---|
| `contained-Objects` | The list of contained object types. | No | Array |
| `contained-Type` | Indicates whether the object type is used as a contained type (by default, False). | Yes | Boolean |

**Table 132. Payload Parameters (continued)**

| Parameter | Description | Required? | Data Type |
|---|---|---|---|
| `dataVariables` | The list of variables with details. | No | Array |
| `description` | The description of the object instance. | No | String |
| `fullName` | The full name of the object instance, which contains the name of the original object type, followed by the contained type name (as applicable). | No | String |
| `name` | The name of the object instance. | Yes | String |
| `objectDataValueChanges` | The list of changes in the object instance. | No | Array |
| `ObjectType` | The object type from which this instance must be created. | Yes | String |
| `objectTypeTemplate` | The name of the template you want to use in the object instance. | Yes | String |
| `Override` | True or false (required, but not used in Historian). | No | Boolean |
| `parentName` | The name of the object type. | No | String |
| `substituteParameters` | Not applicable | No | String |
| `v1Type` | Indicates whether it is an Operations Hub model or a Historian model (by default, False, indicating a Historian model). | Yes | Boolean |
| `versionNumber` | The version number of the object instance. | Yes | Number |

**Table 132. Payload Parameters (continued)**

| Parameter | Description | Required? | Data Type |
|---|---|---|---|
| `dataBaseName` | The name of the Historian system. | Yes | String |

**Table 133. Response Parameters**

| Parameter | Data Type | Description |
|---|---|---|
| `detail` | String | Description of the error. |
| `error_code` | Integer | The error code. A value of 0 indicates that the operation was successful. |
| `status` | String | The http request status. |
| `title` | String | The title of the error. |
| `type` | String | The type of the error. |

### The Object Instance Info API

Using this API, you can view an object instance.

| | |
|---|---|
| METHOD | POST |
| URI | `https://<historianservername>/historian-model/v2/objects/info` |
| SAMPLE QUERY PARAM POST URL | `https://localhost:443/historian-model/v2/objects/info` |
| SAMPLE RESPONSE | `{ "containedObjects": [ { "alias":"Engine", "baseType":"THINGNAME", "containedObjectType":"Engine", "containedObjectTypeTemplate":"Default_Template_Engine", "description":"Engine", "fullName":"Audi1>Engine",` |

```
"name":"Engine",
"variableType":"ContainedAsset"
}
],
"containedType":false,
"dataVariables":
[
{
"baseType":"NUMBER",
"blockType":"AI",
"dbTagName":"Speed",
"description":"",
"name":"Speed",
"properties":[],
"required":true,
"variableType":"Direct"
},
{
"baseType":"STRING",
"blockType":"AI",
"dbTagName":"Type",
"description":"",
"name":"Type",
"properties":[],
"required":true,
"variableType":"InDirect"
},
{
"baseType":"NUMBER",
"blockType":"AI",
"dbTagName":"Color",
"description":"",
"name":"Color",
"properties":[],
"required":true,
"variableType":"Static"
}
```

| | |
|---|---|
| | ],<br>"databaseName":"XYZ",<br>"description":"",<br>"lastModifiedUser":"XYZ.admin",<br>"lastmodifiedTime":"2022-01-21 11:38:59",<br>"name":"Audi1",<br>"objectType":"Vehicle",<br>"objectTypeTemplate":"Car",<br>"override":false,"parentName":">",<br>"substituteParameters":[],<br>"v1Type":false,<br>"versionNumber":3<br>} |
| SAMPLE PAYLOAD | {"name":"Audi1","databaseName":"XYZ"} |

**Table 134. Payload Parameters**

| Parameter | Description | Required? | Data Type |
|---|---|---|---|
| `name` | The name of the object instance. | Yes | String |
| `databaseName` | The name of the Historian server. | No | String |

**Table 135. Response Parameters**

| Parameter | Description | Data Type |
|---|---|---|
| `containedObjects` | The list of contained object types. | Array |
| `containedType` | Indicates whether the object type is used as a contained type. | Boolean |
| `dataVariables` | The list of variables with details. | Array |
| `description` | The description of the object instance. | String |
| `fullName` | The full name of the object instance. | String |

**Table 135. Response Parameters (continued)**

| Parameter | Description | Data Type |
|---|---|---|
| `name` | The name of the object instance. | String |
| `object-DataValue-Changes` | The list of changes in the object instance. | Array |
| `ObjectType` | The name of the object instance from which this instance is created. | String |
| `objectType-Template` | The name of the object type template used in the object instance. | String |
| `Override` | True or false. | Boolean |
| `parentName` | The name of the parent. | String |
| `substitute-Parameters` | Not applicable | String |
| `v1Type` | Indicates whether it is an Operations Hub model or a Historian model (by default, False, indicating a Historian model). | Boolean |
| `versionNum-ber` | The version number of the object instance. | Number |
| `database-Name` | The name of the Historian server. | String |

**Table 136. containedObjectTypes Array Parameters**

| Parameter | Description | Required? | Data Type |
|---|---|---|---|
| `alias` | The name of the variable. | No | String |
| `variableType` | The type of the variable (direct, indirect, or static). | No | String |
| `name` | The name of the object type (must be unique). | Yes | String |
| `description` | The description of the object type). | No | String |

**Table 136. containedObjectTypes Array Parameters (continued)**

| Parameter | Description | Required? | Data Type |
|---|---|---|---|
| `containedObjectType` | The name of the object type that you want to use as a contained type. | No | String |
| `containedObjectTypeTemplate` | The template in the contained type that you want to use. | No | String |
| `baseType` | The type of the contained type (ThingName indicates a contained type). | No | String |

**Table 137. Data Variables Parameters**

| Parameter | Description | Required? | Data Type |
|---|---|---|---|
| `baseType` | The type of the contained type (ThingName indicates a contained type). | No | String |
| `blockType` | Not applicable | No | String |
| `dbTagName` | The tag mapped with the variable (in the case of a direct or indirect variable). | No | String |
| `description` | The description of the object instance. | No | String |
| `name` | The name of the variable. | No | String |
| `properties` | Not applicable | No | Array |
| `required` | Indicates whether this variable is included in the object instance. | No | String |
| `variableType` | The data type of the variable (direct, indirect, or static). | No | String |

### The Object Instance with Contained Type Variables API

Using this API, you can view the contained type variables in an object instance.

| | |
|---|---|
| METHOD | POST |

| | |
|---|---|
| URI | `https://<historianservername>/historian-model/v2/types-InstancesList/objects` |
| SAMPLE QUERY PARAM POST URL | `https://localhost:443/historian-model/v2/typesIn-stancesList/objects` |
| SAMPLE RESPONSE | [<br>{<br>"hasItems":true,<br>"id":"Instances_Audi1_V_",<br>"loaded":false,<br>"parentId":"Instances_Audi1",<br>"text":"Variables",<br>"type":"objectInstanceVariableRootNode"<br>},<br>{<br>"hasItems":false,<br>"id":"Instances_Audi1_V_0",<br>"loaded":false,<br>"objectFullName":"Audi1",<br>"parentId":"Instances_Audi1_V_",<br>"parentObjectName":"Audi1",<br>"text":"Speed",<br>"type":"objectInstanceVariableNode"<br>},<br>{<br>"hasItems":false,<br>"id":"Instances_Audi1_V_1",<br>"loaded":false,<br>"objectFullName":"Audi1",<br>"parentId":"Instances_Audi1_V_",<br>"parentObjectName":"Audi1",<br>"text":"Type",<br>"type":"objectInstanceVariableNode"<br>},<br>{<br>"hasItems":false,<br>"id":"Instances_Audi1_V_2", |

| | |
|---|---|
| | "loaded":false,<br>"objectFullName":"Audi1",<br>"parentId":"Instances_Audi1_V_",<br>"parentObjectName":"Audi1",<br>"text":"Color",<br>"type":"objectInstanceVariableNode"<br>},<br>{<br>"hasItems":true,<br>"id":"Instances_Audi10",<br>"loaded":false,<br>"objectFullName":"Audi1>Engine",<br>"objectType":"Engine",<br>"objectTypeTemplate":"Default_Template_Engine",<br>"parentId":"Instances_Audi1",<br>"parentObjectName":"Audi1",<br>"status":"state_new",<br>"text":"Engine",<br>"type":"objectInstanceContainedAssetNode"<br>},<br>{<br>"hasItems":false,<br>"id":"Instances_Audi10_V_",<br>"loaded":false,<br>"parentId":"Instances_Audi10",<br>"text":"Variables",<br>"type":"objectInstanceVariableRootNode"<br>}<br>] |
| SAMPLE PAYLOAD | `{"name":"Audi1","databaseName":"XYZ"}` |

**Table 138. Payload Parameters**

| Parameter | Description | Re-quired? | Data Type |
|---|---|---|---|
| `name` | The name of the object instance. | Yes | String |

**Table 138. Payload Parameters (continued)**

| Parameter | Description | Required? | Data Type |
|---|---|---|---|
| `databaseName` | The name of the Historian system. | Yes | String |

**Table 139. Response Parameters**

| Parameter | Description | Data Type |
|---|---|---|
| `hasItems` | Indicates whether the variable is a static text or a variable (be default, false, indicating that it is static text). | Boolean |
| `id` | The sequence ID to render the model tree. | String |
| `loaded` | Not applicable | Boolean |
| `parentId` | The ID of the parent. | String |
| `text` | Contains the static text value or, in the case of variables, contains the text `Variables`. | String |
| `type` | The type of the variable:<br>• objectInstanceVariableRootNode: Indicates static text.<br>• objectInstanceVariableNode: Indicates direct, indirect, or a static variable.<br>• objectInstanceContainedAssetNode: Indicates a variable in a contained type. | String |

### The Object Instance - Variable Information API

Using this API, you can view the variable details of an object instance.

| | |
|---|---|
| METHOD | POST |
| URI | `https://<historianservername> /historian-model/v2/ objects/variables/info` |
| SAMPLE QUERY PARAM POST URL | `https://localhost:443/historian-model/v2/objects/variables/info` |

| | |
|---|---|
| SAMPLE RESPONSE | <pre>{<br>"blockType":"AI",<br>"variableName":"Speed",<br>"variableTagName":"Audi1>Speed",<br>"variableType":"Direct",<br>"variableValue":"Audi1>Speed"<br>}</pre> |
| SAMPLE PAYLOAD | `{"databaseName":"XYZ","name":"Audi1>Speed"}` |

**Table 140. Payload Parameters**

| Parameter | Description | Required? | Data Type |
|---|---|---|---|
| `name` | The name of the object instance. | Yes | String |
| `databaseName` | The name of the Historian system. | Yes | String |

**Table 141. Response Parameters**

| Parameter | Description | Data Type |
|---|---|---|
| `blockType` | Not applicable | String |
| `variableName` | The name of the variable. | String |
| `variableTagName` | The name of the tag associated with the variable (not applicable for a static variable). | String |
| `variableType` | The type of the variable (static, direct, or indirect). | String |
| `variableValue` | The value of the variable (in case of a static variable) or the associated tag (in case of a direct or indirect variable). | String |

### The Historian Model API

Using this API, you can view the Historian model of a Historian system.

| | |
|---|---|
| METHOD | POST |
| URI | `https://<historianservername>/historian-model/v2/typesObjectsList` |
| SAMPLE QUERY PARAM POST URL | `https://localhost:443/historian-model/v2/typesObjectsList` |
| SAMPLE PAYLOAD | `{"databaseName":"XYZ"}` |

**Table 142. Payload Parameters**

| Parameter | Description | Required? | Data Type |
|---|---|---|---|
| `databaseName` | The name of the Historian system. | Yes | String |

**Table 143. Response Parameters**

| Parameter | Description | Data Type |
|---|---|---|
| `assetTemplates` | List of all the templates in the Historian model. | Array |
| `assetTemplatesInfo` | List of the details of all the templates in the Historian model. | Array |
| `assetTypes` | List of all the object types in the Historian model. | Array |
| `assetTypesInfo` | List of the details of all the object types in the Historian model. | Array |
| `assets` | List of all the object instances in the Historian model. | Array |
| `assetsInfo` | List of the details of all the object types in the Historian model. | Array |

**Table 144. assetTemplates Parameters**

| Parameter | Description | Data Type |
|---|---|---|
| `defaultTemplate` | Indicates whether it is the default template. | Boolean |
| `description` | The description of the template. | String |

**Table 144. assetTemplates Parameters (continued)**

| Parameter | Description | Data Type |
|---|---|---|
| `lastModified-User` | The username of the user who last modified the template. | String |
| `lastmodified-Time` | The date and time when the template was last modified. | Date and time |
| `name` | The name of the template. | String |
| `objectType` | The name of the object type associated with the template. | String |
| `versionNumber` | The version number of the template. | Integer |

**Table 145. assets Parameters**

| Parameter | Description | Data Type |
|---|---|---|
| `assetFullName` | The full name of the object instance. | Array |
| `assetName` | The name of the object instance. | String |
| `lastModified-User` | The username of the user who last modified the object instance. | String |
| `lastmodified-Time` | The date and time when the object instance was last modified. | Date and time |
| `assetParent-Name` | The parent name of the asset. | String |
| `containedAs-sets` | List of all the contained types in the object type. | Array |
| `versionNumber` | The version number of the object instance. | Integer |

**Table 146. assetsInfo Parameters**

| Parameter | Description | Data Type |
|---|---|---|
| `containedType` | Indicates whether the object type is a contained type. | Boolean |
| `databaseName` | The name of the Historian system. | String |

**Table 146. assetsInfo Parameters (continued)**

| Parameter | Description | Data Type |
|---|---|---|
| `lastModified-User` | The username of the user who last modified the object instance. | String |
| `lastmodified-Time` | The date and time when the object instance was last modified. | Date and time |
| `description` | The description of the object instance. | String |
| `name` | The name of the object instance. | String |
| `objectType` | The name of the object type associated with the object instance. | String |
| `objectTypeTem-plate` | The name of the template used in the object instance. | String |
| `Override` | Updated or not. | Boolean |
| `ParentName` | The parent name of the asset. | String |
| `status` | Not applicable | String |
| `v1Type` | Not applicable | Boolean |
| `versionNumber` | The version number of the object instance. | number |

### The Export Historian Model API

When you create an object type or an object instance, you can use it only in the Historian system in which you have created it. If, however, you want to use the object type/instance in a different Historian system, you can export it and then import it into the other Historian system.

Using this API, you can export a Historian model, along with the object types, object instances, variables, templates, and contained types, into another Historian system.

| | |
|---|---|
| METHOD | GET |
| URI | `http://<historianservername>/historian-model/v2/export?types=<true/false>&templates=<true/false>&instances=<true/false>&databaseName=<encrypted Historian system name>` |

| | |
|---|---|
| SAMPLE QUERY PARAM GET URL | `http://localhost:443/historian-model/v2/export?types=true&templates=true&instances=true&databaseName=KNQWYA` |
| SAMPLE RESPONSE | <pre>{<br>"admissionTime": 1642600209,<br>"jobId": 0<br>}</pre> |

**Table 147. Query Parameters**

| Parameter | Description | Required? | Data Type |
|---|---|---|---|
| `types` | Indicates whether you want to include object types while exporting the Historian model. | Yes | Boolean |
| `templates` | Indicates whether you want to include templates while exporting the Historian model. | No | Boolean |
| `instances` | Indicates whether you want to include object instances while exporting the Historian model. | No | Boolean |
| `databaseName` | The Historian system from which you want to export the Historian model. | Yes | String (Encrypted) |

**Table 148. Response Parameters**

| Parameter | Description | Data Type |
|---|---|---|
| `admissionTime` | The timestamp in Epoch time format when the model is exported. | Unix Time Stamp |
| `jobId` | The job id with which the operation started. | Integer |

### The Export Historian Model to Historian - Job Status API

Using this API, you can view the status of exporting a Historian model to another Historian system.

| | |
|---|---|
| METHOD | GET |
| URI | `http: //<historianservername>/historian-model/import-export/status/?jobId=<jobId>` |

| | |
|---|---|
| SAMPLE QUERY PARAM GET URL | `http://localhost:443/historian-model/import-export/status/?jobId=1` |
| SAMPLE RESPONSE | {<br>"admissionTime": 1642601025,<br>"completionTime": 1642601025,<br>"isRunning": 0,<br>"processing": "Completed",<br>"processingElement": 14,<br>"startTime": 1642601025,<br>"statusPercentage": 100<br>} |

**Table 149. Query Parameters**

| Parameter | Description | Required? | Data Type |
|---|---|---|---|
| `jobId` | The job created when the export command is run. | Yes | Integer |

**Table 150. Response Parameters**

| Parameter | Description | Data Type |
|---|---|---|
| `admissionTime` | The timestamp in Epoch time format when the model is exported. | Unix Time Stamp |
| `completionTime` | The completing time of the job. | Unix Time Stamp |
| `isRunning` | The status of the job. | String |
| `processing` | Indicates whether the elements are being processed. | String |
| `processingElement` | The element that is being processed. | String |
| `startTime` | The start time of the job. | Unix Time Stamp |
| `statusPercentage` | The percentage of job completion. | Integer |

**The Historian Model to Historian - Job Log API**

Using this API, you can view the job log of exporting a Historian model to another Historian system.

| | |
|---|---|
| METHOD | GET |
| URI | `http: //<historianservername>/historian-model/import-export-export/log/?jobId=<jobId>` |
| SAMPLE QUERY PARAM GET URL | `http://localhost:443/historian-model/import-export/log/?jobId=1` |
| SAMPLE RESPONSE | <pre>[<br>{<br>"Elements": ["2022-01-19T08-08-43\t"],<br>"LogStringCode": 12304<br>},<br>{<br>"Elements": [ "2022-01-19T08-08-43\t","3"],<br>"LogStringCode": 12309<br>},<br>{<br>"Elements": ["2022-01-19T08-08-43\t","2.1"],<br>"LogStringCode": 12310<br>}<br>]</pre> |

**Table 151. Query Parameters**

| Parameter | Description | Required? | Data Type |
|---|---|---|---|
| *jobId* | The job created when you run the export command. | Yes | Integer |

**Table 152. Response Parameters**

| Parameter | Description | Data Type |
|---|---|---|
| Array of Elements | List of elements that are processed. | Array |
| `LogStringCode` | The processing code. | Integer |

**The Export Historian Model to Historian - Job Result API**

Using this API, you can export and retrieve a Historian model in a CSV file. You can then import it into another Historian model.

| | |
|---|---|
| METHOD | GET |
| URI | `http: //<historianservername>/historian-model/import-export/result/?jobId=<jobId>` |
| SAMPLE QUERY PARAM GET URL | `http://localhost:443/historian-model/import-export/result/?jobId=1` |

**Table 153. Query Parameters**

| Parameter | Description | Required? | Data Type |
|---|---|---|---|
| `jobId` | The job created when you run the export command. | Yes | Integer |

**Table 154. Response Parameters**

| Parameter | Description | Data Type |
|---|---|---|
| `CSV file` | Contains the exported model | file |

### The Export Historian Model to Operations Hub API

When you create an object type or an object instance, you can use it only in the Historian system in which you have created it. If, however, you want to use the object type/instance in Operations Hub, you can export it and then import it into Operations Hub.

Using this API, you can export a Historian model to Operations Hub.

| | |
|---|---|
| METHOD | GET |
| URI | `http://<historianservername>/historian-model/v2/export?types=<true/false>&instances=<true/false>&forOpsHub=<true/false>&databaseName=<encrypted Historian system name>` |
| SAMPLE QUERY PARAM GET URL | `http://localhost:443/historian-model/v2/export?types=true&instances=true&forOpsHub=true&databaseName=KNOGEYA` |
| SAMPLE RESPONSE | `{`<br>`"admissionTime": 1642600209,` |

| | |
|---|---|
| | <pre>"jobId": 0<br>}</pre> |

**Table 155. Query Parameters**

| Parameter | Description | Required? | Data Type |
|---|---|---|---|
| `types` | Indicates whether you want to include object types while exporting the Historian model. | Yes | Boolean |
| `instances` | Indicates whether you want to include object instances while exporting the Historian model. | No | Boolean |
| `databaseName` | The Historian system from which you want to export the Historian model. | Yes | String (Encrypted) |

**Table 156. Response Parameters**

| Parameter | Description | Data Type |
|---|---|---|
| `admissionTime` | The timestamp in Epoch time format when the model is exported. | Unix Time Stamp |
| `jobId` | The job id with which the operation started. | Integer |

### The Export Historian Model to Operations Hub Job Status API

Using this API, you can get the job status of exporting a Historian model to Operations Hub.

| | |
|---|---|
| METHOD | GET |
| URI | `http://<historianservername>/historian-model/import-export/status/?jobId=<jobId>` |
| SAMPLE QUERY PARAM GET URL | `http://localhost:443/historian-model/import-export/status/?jobId=1` |
| SAMPLE RESPONSE | <pre>{<br>"admissionTime": 1642601025,<br>"completionTime": 1642601025,<br>"isRunning": 0,<br>"processing": "Completed",<br>"processingElement": 14,</pre> |

| | |
|---|---|
| | <pre>"startTime": 1642601025,<br>"statusPercentage": 100<br>}</pre> |

**Table 157. Query Parameters**

| Parameter | Description | Required? | Data Type |
|---|---|---|---|
| `jobId` | The job created when you run the export command. | Yes | Integer |

**Table 158. Response Parameters**

| Parameter | Description | Data Type |
|---|---|---|
| `admissionTime` | The timestamp in Epoch time format when the model is exported. | Unix Time Stamp |
| `completionTime` | The completion time of the job. | Unix Time Stamp |
| `isRunning` | The status of the job. | String |
| `processing` | Indicates whether the elements are being processed. | String |
| `processingElement` | The element which is being processed. | String |
| `startTime` | The start time of the job. | Unix Time Stamp |
| `statusPercentage` | The percentage of job completion. | Integer |

### The Export Historian Model to Operations Hub Job Log API

Using this API, you can get the log of the job when exporting a Historian model to Operations Hub.

| | |
|---|---|
| METHOD | GET |
| URI | `http: //<historianservername>/historian-model/import-export-log/?jobId=<jobId>` |
| SAMPLE QUERY PARAM GET URL | `http://localhost:443/historian-model/import-export/log/?jobId=1` |

| | |
|---|---|
| SAMPLE RESPONSE | [<br>{<br>"Elements": ["2022-01-19T08-08-43\t"],<br>"LogStringCode": 12304<br>},<br>{<br>"Elements": [ "2022-01-19T08-08-43\t", "3"],<br>"LogStringCode": 12309<br>},<br>{<br>"Elements": ["2022-01-19T08-08-43\t","2.1"],<br>"LogStringCode": 12310<br>}<br>] |

**Table 159. Query Parameters**

| Parameter | Description | Required? | Data Type |
|---|---|---|---|
| `jobId` | The job created when you run the export command. | Yes | Integer |

**Table 160. Response Parameters**

| Parameter | Description | Data Type |
|---|---|---|
| Array of Elements | List of elements that are processed. | Array |
| `LogStringCode` | The processing code. | Integer |

### The Export Historian Model to Operations Hub Job Result API

Using this API, you can export and retrieve a Historian model in a CSV file. You can then import it into Operations Hub.

| | |
|---|---|
| METHOD | GET |
| URI | `http://<historianservername>/historian-model/import-export/result/?jobId=<jobId>` |

| | |
|---|---|
| SAMPLE QUERY PARAM GET URL | `http://localhost:443/historian-model/import-export/result/?jobId=1` |

**Table 161. Query Parameters**

| Parameter | Description | Required? | Data Type |
|---|---|---|---|
| `jobId` | The job created when you run the export command. | Yes | Integer |

**Table 162. Response Parameters**

| Parameter | Description | Data Type |
|---|---|---|
| `CSV file` | Contains the exported model. | file |

### The Import Model to Historian API

Using this API, you can import a Historian model from one system to another.

**Important:**
If the name of a tag associated with a variable in a model contains a period (.), you cannot import the tag while importing the model into a Historian system.

| | |
|---|---|
| METHOD | POST |
| URI | `http://<historianservername>/historian-model/v2/import?types=<true/false>&templates=<true/false>&instances=<true/false>&databaseName=<encrypted Historian system name>` |
| SAMPLE QUERY PARAM GET URL | `http://localhost:443/historian-model/v2/import?types=true&templates=true&instances=true&databaseName=KGEYA` |
| SAMPLE RESPONSE | {<br>"admissionTime": 1642600209,<br>"jobId": 0<br>} |

**Table 163. Payload Parameters**

| Payload | Description | Required? | Data Type |
|---|---|---|---|
| `body` | CSV file that you want to import | Yes | file |

**Table 164. Query Parameters**

| Parameter | Description | Required? | Data Type |
|---|---|---|---|
| `types` | Indicates whether you want to import object types. | Yes | Boolean |
| `templates` | Indicates whether you want to import templates. | No | Boolean |
| `instances` | Indicates whether you want to import object instances. | No | Boolean |
| `databaseName` | The Historian system name in to which you want to import the model. | Yes | String (Encrypted) |

**Table 165. Response Parameters**

| Parameter | Description | Data Type |
|---|---|---|
| `admissionTime` | The timestamp in Epoch time format when the model is imported. | Unix Time Stamp |
| `jobId` | The job id with which the operation started. | Integer |

### The Import Historian Model to Historian - Job Status API

Using this API, you can view the job status of importing a Historian model to another Historian system.

| | |
|---|---|
| METHOD | GET |
| URI | `http://<historianservername>/historian-model/import-export/status/?jobId=<jobId>` |
| SAMPLE QUERY PARAM GET URL | `http://localhost:443/historian-model/import-export/status/?jobId=1` |
| SAMPLE RESPONSE | `{`<br>`"admissionTime": 1642601025,` |

| | |
|---|---|
| | "completionTime": 1642601025,<br>"isRunning": 0,<br>"processing": "Completed",<br>"processingElement": 14,<br>"startTime": 1642601025,<br>"statusPercentage": 100<br>} |

**Table 166. Query Parameters**

| Parameter | Description | Required? | Data Type |
|---|---|---|---|
| `jobId` | The job created when the import command is run. | Yes | Integer |

| Parameter | Description | Data Type |
|---|---|---|
| `admissionTime` | The error in text format. | Unix Time Stamp |
| `completionTime` | The completed time of the job. | Unix Time Stamp |
| `isRunning` | The status of the job. | String |
| `processing` | Indicates whether the elements are processed. | String |
| `processingElement` | The element that is being processed. | String |
| `startTime` | The start time of the job. | Unix Time Stamp |
| `statusPercentage` | The percentage of job completion. | Integer |

### The Import Historian Model to Historian - Job Log API

Using this API, you can view the job log of importing a Historian model into another Historian model.

| | |
|---|---|
| METHOD | GET |
| URI | `http://<historianservername>/historian-model/import-export-log/?jobId=<jobId>` |
| SAMPLE QUERY PARAM GET URL | `http://localhost:443/historian-model/import-export/log/?jobId=1` |
| SAMPLE RESPONSE | [<br>{ |

| | |
|---|---|
| | "Elements": ["2022-01-19T08-08-43\t"],<br>"LogStringCode": 12304<br>},<br>{<br>"Elements": ["2022-01-19T08-08-43\t","3"],<br>"LogStringCode": 12309<br>},<br>{<br>"Elements": ["2022-01-19T08-08-43\t","2.1"],<br>"LogStringCode": 12310<br>}<br>] |

**Table 167. Query Parameters**

| Parameter | Description | Required? | Data Type |
|---|---|---|---|
| `jobId` | The job created when the import command is run. | Yes | Integer |

**Table 168. Response Parameters**

| Parameter | Description | Data Type |
|---|---|---|
| Array of Elements | The list of elements that are processed. | Array |
| `LogStringCode` | The processing code. | Integer |

### The Import Historian Model to Historian - Result API

Using this API, you can view the result of importing a Historian model into another Historian system.

| | |
|---|---|
| METHOD | GET |
| URI | `http://<historianservername>/historian-model/import-export/result/?jobId=<jobId>` |
| SAMPLE QUERY PARAM GET URL | `http://localhost:443/historian-model/import-export/result/?jobId=1` |

| | |
|---|---|
| SAMPLE RESPONSE | <pre>[<br>{<br>"Elements":["2022-01-19T08-06-59\t","72","Contained<br>Asset","Engine1"],<br>"LogStringCode":12214<br>},<br>{<br>"Elements":["2022-01-19T08-06-59\t","73","Contained<br>Asset","cylinder1"],<br>"LogStringCode":12214<br>},<br>{<br>"Elements":["2022-01-19T08-06-59\t","74","Contained<br>Asset","piston1"],<br>"LogStringCode":12214<br>}<br>]</pre> |

**Table 169. Payload Parameters**

| Parameter | Description | Required? | Data Type |
|---|---|---|---|
| `jobId` | The job created when the import command is run. | Yes | Integer |

**Table 170. Response Parameters**

| Parameter | Description | Data Type |
|---|---|---|
| Array of Elements | The list of elements that are processed. | Array |
| `LogStringCode` | The processing code. | Integer |

### The Duplicate Object Instance API

Using this API, you can copy an object instance.

| | |
|---|---|
| METHOD | POST |
| URI | `http: //<historianservername>/historian-model/v2/objects/clone` |

| | |
|---|---|
| SAMPLE QUERY PARAM POST URL | `http://localhost:443/historian-model/v2/objects/clone` |
| SAMPLE PAYLOAD | `{`<br>`"databaseName":"XYZ",`<br>`"source": "Vehicle",`<br>`"destination": "Audi",`<br>`"description": "the new object"`<br>`}` |

**Table 171. Payload Parameters**

| Parameter | Description | Required? | Data Type |
|---|---|---|---|
| `databaseName` | The name of the Historian system. | Yes | String |
| `source` | The name of the object instance that you want to copy. | Yes | String |
| `destination` | The name that you want to provide for the copied object instance. | Yes | String |
| `description` | The description of the new object instance. | No | String |

### The Duplicate Object Type API

Using this API, you can copy an object type. When you copy an object type, all the templates and variables are copied too.

| | |
|---|---|
| METHOD | POST |
| URI | `http: //<historianservername>/historian-model/v2/objectTypes/clone` |
| SAMPLE QUERY PARAM POST URL | `http://localhost:443/historian-model/v2/object-Types/clone` |
| SAMPLE PAYLOAD | `{`<br>`"source": "Vehicle",`<br>`"destination": "Audi",` |

| | |
|---|---|
| | `"description": "the new object"`<br>`}` |

**Table 172. Payload Parameters**

| Parameter | Description | Required? | Data Type |
|---|---|---|---|
| `source` | The name of the object type that you want to copy. | Yes | String |
| `destination` | The name that you want to provide for the copied object type. | Yes | String |
| `description` | The description of the new object type. | No | String |

### The Export Object Instance API

Using this API, you can export an object instance into another Historian system.

| | |
|---|---|
| METHOD | POST |
| URI | `http: //<historianservername>/historian-model/v2/export/objects` |
| SAMPLE QUERY PARAM POST URL | `http://localhost:443/historian-model/v2/export/objects` |
| SAMPLE PAYLOAD | `{`<br>`"objects":`<br>`[`<br>`"object_Instance_1",`<br>`"Object_Instance_2"`<br>`],`<br>`"objectType": "Audi",`<br>`"objectTypeTemplate": "Vehicle",`<br>`"databaseName": "XYZ"`<br>`}` |

**Table 173. Payload Parameters**

| Parameter | Description | Required? | Data Type |
|---|---|---|---|
| `objects` | The list of the object instances that you want to export. | Yes | Array |
| `objectType` | The object type associated with the object instances that you want to export. | Yes | String |
| `objectType-Template` | The template type of the object instances that you want to export. | No | String |
| database-Name | The name of the Historian system. | Yes | String |

### The Export Object Type API

Using this API, you can export an object type into another Historian system.

| | |
|---|---|
| METHOD | POST |
| URI | `http://<historianservername>/historian-model/export/objectTypes` |
| SAMPLE QUERY PARAM POST URL | `http://localhost:443/historian-model/export/object-Types?databaseName= KOGEYA` |
| SAMPLE PAYLOAD | `{`<br>`"objectType":`<br>`["object_type_1","object_type_2"]`<br>`}` |

**Table 174. Payload Parameters**

| Parameter | Description | Required? | Data Type |
|---|---|---|---|
| `objectType` | The list of object types that you want to export. | Yes | Array |

**Table 175. Query Parameters**

| Parameter | Description | Required? | Data Type |
|---|---|---|---|
| `databaseName` | The name of the Historian system from which you want to export the object type. | Yes | String (encrypted) |

### The Delete Object Instance API

Using this API, you can delete an object instance. If there are direct variables in the object type, you can also choose to delete the tags associated with these variables (along with their data).

| | |
|---|---|
| METHOD | DELETE |
| URI | `http://<historianservername>/historian-model/v2/objects` |
| SAMPLE QUERY PARAM DELETE URL | `http://localhost:443/historian-model/v2/objects` |
| SAMPLE PAYLOAD | `{`<br>`"databaseName":"XYZ",`<br>`"name": "Vehicle1",`<br>`"permanentDelete": true`<br>`}` |

**Table 176. Payload Parameters**

| Parameter | Description | Required? | Data Type |
|---|---|---|---|
| `databaseName` | The name of the Historian system. | Yes | String |
| `name` | The name of the object instance that you want to delete. | Yes | String |
| `permanentDelete` | Indicates whether you want to permanently delete the object instance. If you do so, the tags are deleted as well. | Yes | Boolean |

### The Incoming Dependencies API

Using this API, you can view a list of object instances associated with an object type. This is used to check if an object type contains object instances before you delete the object type.

| | |
|---|---|
| METHOD | POST |
| URI | `http://<historianservername>/historian-model/v2/object-Types/IncomingDependencies` |
| SAMPLE QUERY PARAM POST URL | `http://localhost:443/historian-model/v2/objectTypes/ In-comingDependencies` |
| SAMPLE PAYLOAD | `{`<br>`"name": "Vehicle",`<br>`"databaseName":"XYZ"`<br>`}` |

**Table 177. Payload Parameters**

| Parameter | Description | Re-quired? | Data Type |
|---|---|---|---|
| `name` | The name of the object type that you want to delete. | Yes | String |
| `database-Name` | The name of the Historian system associated with the object type. | Yes | String |

**The Delete Object Type API**

Using this API, you can delete an object type. You can delete multiple object types together; however, all the object types must belong to the same Historian system. You cannot delete an object type if it is used in an object instance; you must first delete the object instance.

| | |
|---|---|
| METHOD | DELETE |
| URI | `http://<historianservername>/historian-mod-el/v2/objectTypes` |
| SAMPLE QUERY PARAM DELETE URL | `http://localhost:443/historian-model/v2/object-Types` |
| SAMPLE PAYLOAD | `{`<br>`"databaseName": "XYZ",`<br>`"objectTypes": ["Vehicle","Engine"]`<br>`}` |

**Table 178. Payload Parameters**

| Parameter | Description | Re-quired? | Data Type |
|---|---|---|---|
| `object-Types` | The list of object types that you want to delete. | Yes | Array |
| `database-Name` | The name of the Historian system associated with the object types. | Yes | String |

# Managing Collector Instances

### The Create Collector Instance API

Using the Create Collector Instance API, you can create a collector instance.

| METHOD | POST |
|---|---|
| URI | `https://<historianservername>/historian-rest-api/v1/collector/createnewinstance` |
| SAMPLE URI | `https://<historianservername>/historian-rest-api/v1/collector/createnewinstance`<br>Payload<br>`{`<br>`"mode":1,`<br>`"CollectorSystemName":"xyz",`<br>`"InterfaceDescription":"xyz_Simulation_<IP address>_2",`<br>`"DataPathDirectory":"C:\\Proficy Historian Data",`<br>`"CollectorDestination":"Historian",`<br>`"winUserName":"","winPassword":"",`<br>`"InterfaceSubType":"",`<br>`"DestinationHistorianUserName":"abc",`<br>`"DestinationHistorianPassword":"password",`<br>`"DestinationHistorian":"<IP address>",`<br>`"General1":"",`<br>`"General2":"",`<br>`"General3":"123",` |

<table>
<tr><td></td><td>

```
"General4":"",
"General5":"",
"Type":2,
"InterfaceName":"<source server>_<type of the
 collector>_<destination server>"
}
```

**Note:**

- The DestinationHistorian parameter will not have a value for offline collector configuration.
- To connect to MQTT destinations such as AWS IoT and Google Cloud Platform (GCP), you must provide an encrypted password.

</td></tr>
<tr><td>SAMPLE RESPONSE</td><td>

```
{

    "ErrorCode": 0,

    "ErrorMessage": null

}
```

</td></tr>
<tr><td>SAMPLE cURL COMMAND</td><td>

```
curl -i -H "Accept: application/json" -i -H
 "Content-Type: application/json"
-H "Authorization: Bearer <TOKEN>” -d
 “{ \"mode\":1,\"CollectorSystemName\":\"xyz\",
\"InterfaceDescription\":\"xyz_Simulation_<IP
 address>_2\",\"DataPathDirectory\":\"C:\
\Proficy Historian Data\",
\"CollectorDestination\":\"Historian\",
\"winUserName\":\"\",\"winPassword\":\"\",
```

</td></tr>
</table>

| | |
|---|---|
| | \"InterfaceSubType\": \"\",\"DestinationHistorianUserName\":\"abc\", \"DestinationHistorianPassword\":\"password\", \"DestinationHistorian\":\"<IP address>\",\"General1\":\"\", \"General2\":\"\",\"General3\":\"xyz\", \"General4\":\"\",\"General5\":\"\", \"Type\":2,\"InterfaceName\":\"<*source server*>_<*type of the collector*>_<*destination server*>\"}" -X POST https://<historianservername>/historian-rest-api/v1/collector/createnewinstance |

**Table 179. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |

### The Get Collector Instance Details API

Using the Get Collector Instance Details API, you can view the details of a collector instance.

| | |
|---|---|
| METHOD | GET |
| URI | https://<historianservername>/historian-rest-api/v1/collector/instancedetails/<interface name> |
| SAMPLE QUERY PARAM GET URL | https://<historianservername>/historian-rest-api/v1/collector/instancedetails/<interface name> |
| SAMPLE RESPONSE | { "ErrorCode":0, "ErrorMessage":null, "Data": { "CloudDestination":"", "InterfaceSubType":"", "CollectorSystemName":"xyz", |

| | |
|---|---|
| | <pre>"Type":2,<br>"DefaultCompression":false,<br>"CloudInformationLogLevel":0,<br>"InterfaceDataDir":"C:\\Proficy Historian Data",<br>"SourceServer":"",<br>"Username":"",<br>"Password":"",<br>"DestinationType":"Historian",<br>"DestinationServer":"abc",<br>"DebugLogLevel":0,<br>"InterfaceInstallDrive":"C",<br>"ConnnectionString":"xyz"<br>}<br>}</pre> |
| SAMPLE cURL COMMAND | <pre>curl -i -H "Accept: application/json" -H<br>"Authorization: Bearer <TOKEN>”<br>https://<historianservername>/historian-rest-a<br>pi/v1/collector/instancedetails/xyz_Simulation_<br><IP address>_2</pre> |

**Table 180. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| `interface name` | The interface name of the collector whose details you want to view. | Yes | String |

**Table 181. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |

### The Edit Collector Instance API

Using the Edit Collector Instance API, you can modify the cloud parameters of a collector instance. The collector instance will be restarted after you make changes.

| METHOD | PUT |
| --- | --- |
| URI | `https://<historianservername>/historian-rest-api/v1/collector/editInstance` |
| SAMPLE URI | `https://<historianservername>/historian-rest-api/v1/collector/editInstance`<br>Payload<br>`{"InterfaceName":"<source server>_<type of the collector>_<destination server>",`<br>`"messageCompression":0,`<br>`"azureLogLevel":1,`<br>`"debugMode":0,`<br>`"CollectorDestination":"Predix",`<br>`"DestinationHistorian":"abc",`<br>`"mode":1,`<br>`"CloudDestinationAddress":"wss://def.run.abc.ice.predix.io/v1/stream/messages",`<br>`"IdentityIssuer":"https://1234.predix-uaa.run.abc.ice.predix.io/oauth/token",`<br>`"ClientID":"xyz",`<br>`"ClientSecret":"123",`<br>`"ZoneID":"1234",`<br>`"Proxy":"http://1.2.3.4:80",`<br>`"ProxyUserName":"",`<br>`"ProxyPassword":"",`<br>`"DatapointAttribute1":"",`<br>`"DatapointAttribute2":"",`<br>`"DatapointAttribute3":"",`<br>`"DatapointAttribute4":""}`<br>**Note:**<br>• The DestinationHistorian parameter will not have a value for offline collector configuration. |

| | |
|---|---|
| | • To connect to MQTT destinations such as AWS IoT and Google Cloud Platform (GCP), you must provide an encrypted password. |
| SAMPLE RESPONSE | {<br><br>"ErrorCode": 0,<br><br>"ErrorMessage": null<br><br>} |
| SAMPLE cURL COMMAND | curl -i -H "Accept: application/json" -i -H<br>"Content-Type: application/json"<br>-H "Authorization: Bearer <TOKEN>" -d<br>"{\"InterfaceName\":\"<source server>_<type of<br>the collector>_<destination server>\",<br>\"InterfaceName\":\"<source server>_<type<br>of the collector>_<destination<br>server>\",<br>\"messageCompression\":0,\"azureLogLevel\":1,<br>\"debugMode\":0,\"CollectorDestination\":<br>\"Predix\",\"DestinationHistorian\":\"abc\",<br>\"mode\":1,<br>\"CloudDestinationAddress\":<br>\"wss://wss://def.run.abc.ice.predix.io/v1/stre<br>am/messages\",<br>\"IdentityIssuer\":<br>\"https://1234.predix-uaa.run.abc.ice.predix.<br>io/oauth/token/",<br>\"ClientID\":\"HistQA\",\"ClientSecret\":<br>\"Gei321itc\",\"ZoneID\":\"1234\",<br>\"Proxy\":<br>\"http://1.2.3.4:80\",\"ProxyUserName\":<br>\"\",\"ProxyPassword\":\"\", |

| | |
|---|---|
| | \"DatapointAttribute1\":\"\",\"DatapointAttribute2\":\"\",\"DatapointAttribute3\":\"\", \"DatapointAttribute4\":\"\"}” -X PUT https://<historianservername>/historian-rest-api/v1/collector/editInstance |

**Table 182. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |

### The Azure Log Level API

Using the Azure Log Level API, you can set the debug information log level for destination - Azure IoT Hub. You can set a value ranging from 0 to 4.

| | |
|---|---|
| METHOD | PUT |
| URI | https://<historianservername>/historian-rest-api/v1/collector/azureloglevel |
| SAMPLE URI | https://<historianservername>/historian-rest-api/v1/collector/azureloglevel<br>Payload<br>{"InterfaceName":""InterfaceName":"*<source server>_<type of the collector>_<destination server>*"", "azureLogLevel":1,} |
| SAMPLE RESPONSE | { "ErrorCode": 0, "ErrorMessage": null } |

| | |
|---|---|
| SAMPLE cURL COMMAND | `curl -i -H "Accept: application/json" -i -H "Content-Type: application/json" -H "Authorization: Bearer <TOKEN>" -d "{\"InterfaceName\":\"<source server>_<type of the collector>_<destination server>\",\"azureLogLevel\":1}" -X PUT https://<historianservername>/historian-rest-api/v1/collector/azureloglevel` |

**Table 183. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |

### The Install Component Details API

Using the Install Component Details API, you can view the install component details from the collector machine.

| | |
|---|---|
| METHOD | GET |
| URI | `https://<historianservername>/historian-rest-api/v1/installcomponentdetails/collectorType/collectorSubType/machine` |
| SAMPLE URI | `https://<historianservername>/historian-rest-api/v1/installcomponentdetails/2/-/abc` |
| SAMPLE RESPONSE | `{ "ErrorCode":0, "ErrorMessage":null, "Data": { "InterfaceInstallDrive":"C", "InterfaceDataDir":"C:\\Proficy Historian Data", "CertPathDir":"NONE"` |

| | |
|---|---|
| | }<br>} |
| SAMPLE cURL COMMAND | curl -i -H "Accept: application/json" -H<br>"Authorization: Bearer <TOKEN>"<br>https://<historianservername>/historian-rest-a<br>pi/v1/installcomponentdetails/2/-/abc |

**Table 184. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |

### The Message Compression API

Using the Message Compression API, you can enable or disable message compression.

| | |
|---|---|
| METHOD | PUT |
| URI | https://<historianservername>/historian-rest-a<br>pi/v1/collector/messagecompression |
| SAMPLE REQUEST | {<br>"InterfaceName":"*<source server>_<type of the*<br>*collector>_<destination server>*",<br>"messageCompression":1<br>} |
| SAMPLE RESPONSE | {<br>"ErrorCode":0,<br>"ErrorMessage":null,<br>} |
| SAMPLE cURL COMMAND | curl -i -H "Accept: application/json" -i -H<br>"Content-Type: application/json"<br>-H "Authorization: Bearer <TOKEN>" -d<br>“{\"InterfaceName\":\"*<source server>_<type of*<br>*the collector>_<destination server>*\",<br>\"messageCompression\":1}” -X PUT |

| | |
|---|---|
| | `https://<historianservername>/historian-rest-api/v1/collector/messagecompression` |

**Table 185. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| `interfacename` | The interface name of the collector whose message compression you want to enable or disable. | Yes | String |
| `messagecompression` | Identifies whether you want to enable or disable message compression. The valid values are 0 and 1. | Yes | |

**Table 186. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |

### The Delete Instance API

Using the Delete Instance API, you can delete a collector instance.

| | |
|---|---|
| METHOD | PUT |
| URI | `https://<historianservername>/historian-rest-api/v1/collector/deleteinstance` |
| SAMPLE REQUEST | `https://<historianservername>/historian-rest-api/v1/collector/deleteinstance`<br>Payload<br>{<br>"InterfaceName":"*<source server>_<type of the collector>_<destination server>*",<br>"deleteTags":true<br>} |

| | |
|---|---|
| SAMPLE RESPONSE | <pre>{<br>"ErrorCode":0,<br>"ErrorMessage":null,<br>}</pre> |
| SAMPLE cURL COMMAND | <pre>curl -i -H "Accept: application/json" -i -H<br>"Content-Type: application/json"<br>-H "Authorization: Bearer <TOKEN>"<br>-d "{\"InterfaceName\":\"<source<br>server>_<type of the collector>_<destination<br>server>\",\"deleteTags\":true}" -X PUT<br>https://<historianservername>/historian-rest-a<br>pi/v1/collector/deleteinstance</pre> |

**Table 187. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| `interfacename` | The interface name of the collector whose details you want to delete. | Yes | String |
| `deleteTags` | Identifies whether you want to delete the tags. The valid values are true and false. | Yes | Boolean |

**Table 188. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |

## Collector Type and Subtype

The following table provides a list of collector type and subtype for each collector, which you will provide in APIs.

| Collector | Collector Type | Collector Subtype |
|---|---|---|
| The Calculation collector | 8 | |
| The CygNet collector | 16 | Cygnet |
| The File collector | 4 | |
| The HAB collector | 16 | HAB |
| The iFIX Alarms and Events collector | 11 | iFixAE |
| The iFIX collector | 1 | |
| The MQTT collector | 16 | MQTT |
| The ODBC collector | 16 | ODBC |
| The OPC Classic Alarms and Events collector | 11 | |
| The OPC Classic DA collector | 3 | |
| The OPC Classic HDA collector | 16 | OPCHDA |
| The OPC UA DA collector | 16 | OPCUA |
| The OSI PI collector | 10 | |
| The OSI PI distributor | 13 | |
| The Server-to-Server collector | 9 | |
| The Server-to-Server distributor | 17 | |
| The Simulation collector | 2 | |
| The Windows Performance collector | 18 | |
| The Wonderware collector | 16 | Wonderware |

## Managing Collectors

### The Start Collector API

Using the Start Collector API, you can start a collector.

| METHOD | PUT |
| --- | --- |
| URI | `https://<historianservername>/historian-rest-api/v1/collector/start` |
| SAMPLE URI | Sample URI for the service mode:<br>`https://<historianservername>/historian-rest-api/v1/collector/start`<br><br>Payload<br><br>{<br>"interfaceName":"*<source server>_<type of the collector>_<destination server>*",<br>"mode":1<br>}<br><br>Sample URI for the command line mode:<br>`https://<historianservername>/historian-rest-api/v1/collector/start`<br><br>Payload<br><br>{<br>"interfaceName":"*<source server>_<type of the collector>_<destination server>*",<br>"mode":2,<br>“winUserName”:””,<br>“winPassword”:””<br>} |
| SAMPLE RESPONSE | {<br>"ErrorCode": 0,<br>"ErrorMessage": ""<br>"Data": "Collector Start Initiated"<br>} |
| SAMPLE cURL COMMAND | `curl -i -H "Accept: application/json" -i -H "Content-Type: application/json"` |

| | |
|---|---|
| | `-H "Authorization: Bearer <TOKEN>" -d “{ \”interfaceName\”:\”<source server>_<type of the collector>_<destination server>\”, \"mode\": 1}” -X PUT https://<historianservername>/historian-rest-api/v1/collector/start` |

Query parameters include the Payload parameter, which is a JSON file, which contains the following properties.

**Table 189. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| `interfaceName` | The interface name of the collector. | Yes | String |
| `mode` | The mode to use to manage the collector. | Yes | • 1: service mode<br>• 2: command-line mode |
| `winUserName` | The Windows username. | Yes (only if you want to use the command-line mode) | String |
| `winPassword` | The Windows password | Yes (only if you want to use the command-line mode) | String |

**Table 190. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |
| `Data` | String | No | Indicates if the task has been initiated. |

### The Stop Collector API

Using the Stop Collector API, you can stop a collector.

| METHOD | PUT |
|---|---|
| URI | `https://<historianservername>/historian-rest-api/v1/collector/stop` |
| SAMPLE URI | `https://<historianservername>/historian-rest-api/v1/collector/stop`<br><br>`Payload`<br><br>`{`<br>`“interfaceName":"<source server>_<type of the collector>_<destination server>"`<br>`“winUserName”:”TestAdmin”,`<br>`“winPassword”:”TestAdminPassword”`<br>`}` |
| SAMPLE RESPONSE | `{`<br>`"ErrorCode": 0,`<br>`"ErrorMessage": null,`<br>`"Data": "Collector Stop Initiated"`<br>`}` |
| SAMPLE cURL COMMAND | `curl -i -H "Accept: application/json" -i -H "Content-Type: application/json"`<br>`-H "Authorization: Bearer <TOKEN>” -d`<br>`“{ \”interfaceName\”:\”<source server>_<type of the collector>_<destination server>\”}”`<br>`-X PUT`<br>`https://<historianservername>/historian-rest-api/v1/collector/stop` |

Query parameters include the Payload parameter, which is a JSON file, which contains the following properties.

**Table 191. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| `interfaceName` | The interface name of the collector. | Yes | String |
| `winUserName` | The Windows username. | Yes (only if you want to use the command-line mode) | String |
| `winPassword` | The Windows password | Yes (only if you want to use the command-line mode) | String |

**Table 192. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |
| `Data` | String | No | Indicates if the task has been initiated. |

### The Restart Collector API

Using the Restart Collector API, you can restart a collector.

| | |
|---|---|
| METHOD | PUT |
| URI | `https://<historianservername>/historian-rest-api/v1/collector/restart` |
| SAMPLE URI | `https://<historianservername>/historian-rest-api/v1/collector/restart`<br><br>`Payload`<br>`{` |

| | |
|---|---|
| | “interfaceName":"<*source server*>_<*type of the collector*>_<*destination server*>"<br>"winUserName":"",<br>"winPassword":""<br>} |
| SAMPLE RESPONSE | {<br>"ErrorCode": 0,<br>"ErrorMessage": null,<br>"Data": "Collector Restart Initiated"<br>} |
| SAMPLE cURL COMMAND | curl -i -H "Accept: application/json" -i -H "Content-Type: application/json"<br>-H "Authorization: Bearer <TOKEN>” -d<br>“{ \”interfaceName\”:\”<*source server*>_<*type of the collector*>_<*destination server*>\”}”<br>-X PUT<br>https://<historianservername>/historian-rest-api/v1/collector/restart |

Query parameters include the Payload parameter, which is a JSON file, which contains the following properties.

**Table 193. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| `interfaceName` | The interface name of the collector. | Yes | String |
| `winUserName` | The Windows username. | Yes | String |
| `winPassword` | The Windows password | Yes | String |

**Table 194. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |

**Table 194. Response Parameters (continued)**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `Data` | String | No | Indicates if the task has been initiated. |

### The Pause Data Collection API

Using the Pause Data Collection API, you can pause the data collection of a collector.

| | |
|---|---|
| METHOD | PUT |
| URI | `https://<historianservername>/historian-rest-api/v1/collector/pausecollection/{interfaceName}` |
| SAMPLE URI | `https://<historianservername>/historian-rest-api/v1/collector/pausecollection/RSSERVER2012-02_Simulation` |
| SAMPLE RESPONSE | `{ "ErrorCode": 0, "ErrorMessage": "" }` |
| SAMPLE cURL COMMAND | `curl -i -H "Accept: application/json" -i -H "Content-Type: application/json" -H "Authorization: Bearer <TOKEN>” -X PUT https://<historianservername>/historian-rest-api/v1/collector/pausecollection/RSSERVER2012-02_Simulation` |

**Table 195. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |
| `Data` | String | No | Indicates if the task has been initiated. |

### The Resume Data Collection API

Using the Resume Data Collection API, you can resume the data collection of a collector.

| METHOD | PUT |
|---|---|
| URI | `https://<historianservername>/historian-rest-api/v1/collector/resumecollection/{interfaceName}` |
| SAMPLE URI | `https://<historianservername>/historian-rest-api/v1/collector/resumecollection/RSSERVER2012-02_Simulation` |
| SAMPLE RESPONSE | `{ "ErrorCode": 0, "ErrorMessage": "" }` |
| SAMPLE cURL COMMAND | `curl -i -H "Accept: application/json" -i -H "Content-Type: application/json" -H "Authorization: Bearer <TOKEN>” -X PUT https://<historianservername>/historian-rest-api/v1/collector/resumecollection/RSSERVER2012-02_Simulation` |

**Table 196. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |
| `Data` | String | No | Indicates if the task has been initiated. |

### The Add Tag Comment API

Using the Add Tag Comment API, you can add a comment to a tag.

| METHOD | POST |
|---|---|
| URI | `https://<historianservername>/historian-rest-api/v1/tags/addcomment` |

| | |
|---|---|
| SAMPLE URI | `https://<historianservername>/historian-rest-api/v1/tags/addcomment`<br><br>Payload<br>`{`<br>`"tagName":"rsserver2012-02.Simulation00003",`<br>`"comment":"Retest",`<br>`"timeStamp":"2020-04-22T00:00:00.000Z"`<br>`}` |
| SAMPLE RESPONSE | `{`<br>`"ErrorCode": 0,`<br>`"ErrorMessage": null,`<br>`"Data": ""`<br>`}` |
| SAMPLE cURL COMMAND | `curl -i -H "Accept: application/json" -i -H "Content-Type: application/json" -H "Authorization: Bearer <TOKEN>" -d "{ \"tagName\":\" rsserver2012-02.Simulation00003\",\"comment\":\"Test10\",\"timeStamp\": \"2020-04-22T00:00:00.000Z \"}" -X POST https://<historianservername>:443/historian-rest-api/v1/tags/addcomment` |

Query parameters include the Payload parameter, which is a JSON file, which contains the following properties.

**Table 197. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| `tagName` | The name of the tag. | Yes | String |
| `timestamp` | The timestamp of the comment. | Yes | String |
| `comment` | The comment. | Yes | String |

**Table 198. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |

### The Get Tag Comment API

Using the Get Tag Comment API, you can view the comments added to a tag.

| | |
|---|---|
| METHOD | GET |
| URI | `https://<historianservername>/historian-rest-api/v1/tags/comments/{tagNames}/{start}/{end}` |
| SAMPLE QUERY PARAM GET URI | `https://<historianservername>/historian-rest-api/v1/tags/comments/?tagNames=rsserver2012-02.Simulation00003;rsserver2012-02.Simulation00004&start=2020-04-19T00:00:00.000Z&end=2020-04-24T00:00:00.000Z`<br><br>**Note:**<br>The query parameter supports multiple tags. |
| SAMPLE RESPONSE | `{`<br>`"ErrorCode": 0,`<br>`"ErrorMessage": null,`<br>`"Data": [`<br>`{`<br>`"TagName": "Motor Temperature",`<br>`"ErrorCode": 0,`<br>`"Comments": [`<br>`{` |

| | |
|---|---|
| | "TimeStamp": "2020-04-22T00:00:00.000Z", "Comment": "Heat run test: Starting temperature" }, { "TimeStamp": "2020-04-22T00:00:00.000Z", "Comment": "Heat run test: Temperature of the stator" }, { "TimeStamp": "2020-04-22T00:00:00.000Z", "Comment": "Heat run test: Temperature of the rotor" }, { "TimeStamp": "2020-04-22T00:00:00.000Z", "Comment": "Heat run test: Temperature of the shaft" }, { "TimeStamp": "2020-04-22T00:00:00.000Z", "Comment": "Heat run test: Temperature of the endshield" } ] } ] } |
| SAMPLE cURL COMMAND | `curl -i -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>” http://<historianservername>` |

| | |
|---|---|
| | `/historian-rest-api/v1/tags/comments/<tagNames>/<start>/<end>` |

Query parameters include the Payload parameter, which is a JSON file, which contains the following properties.

**Table 199. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| `tagNames` | The names of the tag as a semi-colon-separated list. For example: HISTWIN20161.ctag1; HISTWIN20161.ctag2 | Yes | String |
| `start` | The start time of the query, in ISO data format (YYYY-MM-DDTHH:mm:ss.SSSZ). | Yes | DateTime |
| `end` | The end time of the query, in ISO format (YYYY-MM-DDTHH:mm:ss.SSSZ). | Yes | DateTime |

**Table 200. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |

### The Set Debug Mode API

Using the Set Debug Mode API, you can set the debug mode of a collector.

| | |
|---|---|
| METHOD | PUT |
| URI | `https://<historianservername>/historian-rest-api/v1/collector/setdebugmode` |
| SAMPLE URI | `https://<historianservername>/historian-rest-api/v1/collector/setdebugmode` |

| | |
|---|---|
| | Payload<br>{<br>“interfaceName":"<*source server*>_<*type of the collector*>_<*destination server*>",<br>“debugMode”:255<br>} |
| SAMPLE RESPONSE | {<br>"ErrorCode": 0,<br>"ErrorMessage": ""<br>} |
| SAMPLE cURL COMMAND | curl -i -H "Accept: application/json" -i -H "Content-Type: application/json"<br>-H "Authorization: Bearer <TOKEN>” -d<br>“{ \”interfaceName\”:\”<*source server*>_<*type of the collector*>_<*destination server*>\",<br>\”debugMode\”}” -X PUT<br>https://<historianservername>/historian-rest-api/v1/collector/setdebugmode |

Query parameters include the Payload parameter, which is a JSON file, which contains the following properties.

**Table 201. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| `interfaceName` | The interface name of the collector. | Yes | String |
| `debugMode` | The debug log level for the collector. | Yes | • 0: Normal log level<br>• 255: Debug log level |

**Table 202. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |

**Table 202. Response Parameters (continued)**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorMessage` | String | Yes | For example, NULL. |

### The Buffer File Control API

Using the Buffer File Control API, you can delete or move the buffer files. It is recommended to move the buffer files to a new folder within the same drive.

> **Note:**
> Moving files to a network shared drive is not supported.

| | |
|---|---|
| METHOD | PUT |
| URI | `https://<historianservername>/historian-rest-api/v1/collector/buffercontrol` |
| SAMPLE URI | Sample URI for moving buffer files:<br>`https://<historianservername>/historian-rest-api/v1/collector/buffercontrol`<br><br>`Payload`<br>`{`<br>`“interfaceName":"<source server>_<type of the collector>_<destination server>",`<br>`“bufferMode”:2,`<br>`"winUserName":"Administrator",`<br>`"winPassword":"xxxxxxx",`<br>`"bufferPath": "C:\\Users\\bufffiles"`<br>`}`<br>Sample URI for deleting buffer files:<br>`https://<historianservername>/historian-rest-api/v1/collector/buffercontrol`<br><br>`Payload`<br>`{` |

| | |
|---|---|
| | “interfaceName":"<*source server*>_<*type of the collector*>_<*destination server*>",<br>“bufferMode”:1<br>"winUserName":"Administrator",<br>"winPassword":"xxxxxxx",<br>} |
| SAMPLE RESPONSE | Sample response for moving buffer files:<br>{<br>"ErrorCode": 0,<br>"ErrorMessage": null,<br>"Data": "BufferFiles Move Initiated. Collector is in the Stopped state."<br>}<br>Sample response for deleting buffer files:<br>{<br>"ErrorCode": 0,<br>"ErrorMessage": null,<br>"Data": "BufferFiles Delete Initiated. Collector is in the Stopped state."<br>} |
| SAMPLE cURL COMMAND | curl -i -H "Accept: application/json" -i -H "Content-Type: application/json"<br>-H "Authorization: Bearer <TOKEN>” -d<br>“{ \”interfaceName\”:\”<*source server*>_<*type of the collector*>_<*destination server*>\",<br>\”bufferMode \”:1,\”bufferPath \”:\” C:\\Users\\bufffiles\”}” -X PUT https://<historianservername>/historian-rest-api/v1/collector/buffercontrol |

Query parameters include the Payload parameter, which is a JSON file, which contains the following properties.

**Table 203. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| `interfaceName` | The interface name of the collector. | Yes | String |
| `bufferMode` | Indicates whether you want to move or delete the files. | Yes | • 1: Indicates that the buffer files will be deleted.<br>• 2: Indicates that the buffer files will be moved to the location specified in the bufferPath parameter. |
| `bufferPath` | The directory to which you want to move the buffer files. For example: `C:\\Data\\NewBufferFilesLocation` or `C:/Data/NewBufferPathLocation`<br><br>**Note:**<br>The double slash (\\) is required in the JSON format. | Yes (only if you want to move the buffer files) | String |
| `winUserName` | The Windows username. | Yes (only if you want to use the command-line mode) | String |
| `winPassword` | The Windows password | Yes (only if you want to use the com- | String |

**Table 203. Query Parameters (continued)**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| | | mand-line mode) | |

**Table 204. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |
| `Data` | String | No | Indicates if the task has been initiated (and if the collector is in the stopped state). |

### The Server Node Change API

Using the Server Node Change API, you can change the server node of a collector to a machine that has Historian 8.1 installed on it.

| | |
|---|---|
| METHOD | PUT |
| URI | `https://<historianservername>/historian-rest-api/v1/collector/historiannodechange` |
| SAMPLE URI | `https://<historianservername>/historian-rest-api/v1/collector/historiannodechange`<br><br>Payload<br>{<br>“interfaceName":"*<source server>_<type of the collector>_<destination server>*",<br>“mode”:2,<br>"winUserName":"TestAdministrator",<br>"winPassword":"TestPassword",<br>"historianNode":"VMHISTWEBAUTO",<br>"historianUserName":" TestAdministrator2 ",<br>"historianpassword":" TestPassword2" |

| | |
|---|---|
| | }<br> |
| SAMPLE RESPONSE | {<br>"ErrorCode": 0,<br>"ErrorMessage": ""<br>} |
| SAMPLE cURL COMMAND | curl -i -H "Accept: application/json" -i -H<br>"Content-Type: application/json"<br>-H "Authorization: Bearer <TOKEN>" -d<br>"{ \"interfaceName\":\"<*source server*>_<*type of the collector*>_<*destination server*>\",<br>\"userName<br>winUserName\":\"tesrt\",\"winpPassword<br>\":\"password\",<br>\"historianNode \":\"nodename\",\"<br>historianUserName \":\"husername\",<br>\" historianpassword \":\"hpassword\"}" -X PUT<br>https://<historianservername>/historian-rest-api/v1/collector/ historiannodechange |

Query parameters include the Payload parameter, which is a JSON file, which contains the following properties.

**Table 205. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| `interfaceName` | The interface name of the collector. | Yes | String |
| `mode` | The mode to use to manage the collector. | Yes | • 1: service mode<br>• 2: command-line mode |
| `winUserName` | The Windows username. | Yes (only if you want to use the command-line mode) | String |

**Table 205. Query Parameters (continued)**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| `winPassword` | The Windows password. | Yes (only if you want to use the command-line mode) | String |
| `historianNode` | The host name of the new Historian destination machine. The destination machine must have Historian 8.1. | Yes | String |
| `historian-UserName` | The Windows username of the new Historian destination machine. | Yes (only if you want to use the command-line mode) | String |
| `historian-Password` | The Windows password of the new Historian destination machine. | Yes (only if you want to use the command-line mode) | String |

**Table 206. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |

### The Get Collector Version API

Using the Get Collector Version API, you can view the version number of a collector.

| | |
|---|---|
| METHOD | GET |

| | |
|---|---|
| URI | `https://<historianservername>/historian-rest-api/v1/collector/version/{interfaceName}` |
| SAMPLE QUERY PARAM GET URL | `https://<historianservername>/historian-rest-api/v1/collector/version/?interfaceName=<source server>_<type of the collector>_<destination server>` |
| SAMPLE RESPONSE | `{ "ErrorCode": 0, "ErrorMessage": null, "Data": { "Version": "8.1.2068.0" } }` |
| SAMPLE cURL COMMAND | `curl -i -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>" http://<historianservername>/historian-rest-api/v1/collector/version/RSSERVER2012-02_Simulation` |

**Table 207. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| `interfaceName` | The interface name of the collector. | Yes | String |

**Table 208. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |
| `Data` | String | Yes | Returns the version of the collector. |

### The Get Collector Status API

Using the Get Collector Status API, you can view the status of a collector.

| | |
|---|---|
| METHOD | GET |
| URI | `https://<historianservername>/historian-rest-api/v1/collector/status/{interfaceName}` |
| SAMPLE GET URI | `https://<historianservername>/historian-rest-api/v1/collector/status/RSSERVER2012-02_Simulation` |
| SAMPLE RESPONSE | `{ "ErrorCode": 0, "ErrorMessage": "null" "Data":{ "Status":"Running" } }` |
| SAMPLE cURL COMMAND | `curl -i -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>" https://<historianservername>/historian-rest-api/v1/collector/status/RSSERVER2012-02_Simulation` |

**Table 209. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |
| `Data` | String | Yes | Returns the status of the collector. |

### The Collector Manager List API

Using the Collector Manager List API, you can view the list of collector agents machines associated with the Historian server.

| | |
|---|---|
| METHOD | GET |

| | |
|---|---|
| URI | `https://<historianservername>/historian-rest-api/v1/collectormanagerlist` |
| SAMPLE QUERY PARAM GET URL | `https://<historianservername>/historian-rest-api/v1/collectormanagerlist` |
| SAMPLE RESPONSE | {<br>"ErrorCode": 0,<br>"ErrorMessage": null,<br>"Data": [<br>{<br>"Name": "CollectorManager::abc",<br>"IPAddress": "[::ffff :<IP address>]",<br>"Status": 1,<br>"ComputerName": "abc "<br>},<br>{<br>"Name": "CollectorManager::xyz",<br>"IPAddress": "[::ffff:<IP address>]",<br>"Status": 1,<br>"ComputerName": "xyz"<br>},<br>{<br>"Name": "CollectorManager::abc",<br>"IPAddress": "[::ffff:<IP address>]",<br>"Status": 1,<br>"ComputerName": "abc"<br>},<br>{<br>"Name": "CollectorManager::123",<br>"IPAddress": "[::ffff:<IP address>]",<br>"Status": 1,<br>"ComputerName": "123"<br>} |

| | |
|---|---|
| | ```<br>    ]<br>}<br>``` |
| SAMPLE cURL COMMAND | ```<br>curl -i -H "Accept: application/json" -H<br> "Authorization: Bearer <TOKEN>”<br>https://<historianservername>/historian-rest-a<br>pi/v1/collectormanagerlist<br>``` |

**Table 210. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |

### The Collector Mode API

Using the Collector Mode API, you can view the running mode of a collector.

| | |
|---|---|
| METHOD | GET |
| URI | ```<br>https://<historianservername>/historian-rest-a<br>pi/v1/collector/mode/<collector interface name><br>``` |
| SAMPLE QUERY PARAM GET URL | ```<br>https://<historianservername>/historian-rest-a<br>pi/v1/collector/mode/<collector interface name><br>``` |
| SAMPLE RESPONSE | ```<br>{<br>    "ErrorCode": 0,<br>    "ErrorMessage": null,<br>    "Data": {<br>        "RunningMode": "Service Mode"<br>    }<br>}<br>``` |
| SAMPLE cURL COMMAND | ```<br>curl -i -H "Accept: application/json" -H<br> "Authorization: Bearer <TOKEN>”<br>https://<historianservername>/historian-rest-a<br>pi/v1/collector/mode/<host name>_Simulation_<IP<br> address>_2<br>``` |

**Table 211. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |

### The Collector Details API

Using the Collector Details API, you can view the details of a collector.

| | |
|---|---|
| METHOD | GET |
| URI | `https://<historianservername>/historian-rest-api/v1/collector/details` |
| SAMPLE QUERY PARAM GET URL | `https://<historianservername>/historian-rest-api/v1/collector/details` |
| SAMPLE RESPONSE | {<br>"ErrorCode":0,<br>"ErrorMessage":null,<br>"Data":<br>[<br>{<br>"Name":"<value>",<br>"ComputerName":"<value>",<br>"Status":"Running",<br>"ReportRate":0,<br>"MaximumEventRate":0,<br>"MinimumEventRate":0,<br>"OutOfOrderEvents":0,<br>"Overruns":0,<br>"OverrunsPercent":0,<br>"TotalEventsCollected":0,<br>"TotalEventsReported":0,<br>"LastDataValue":"\\\"1970-01-01T00:00:00.000Z\\\"",<br>"Redundency":"", |

| | |
|---|---|
| | "Comments":"<username>--test2--\\ \"2020-12-15T07:19:42.000Z\\\";", "Version":"9.0.4326.0", "CollectorCompression":0, "TagsCount":0 } ] } |
| SAMPLE cURL COMMAND | curl -i -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>" https://<historianservername>/historian-rest-api/v1/collector/details |

**Table 212. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |

### The Offline Collectors API

Using the Offline Collectors API, you can view a list of offline collectors.

| | |
|---|---|
| METHOD | GET |
| URI | https://<historianservername>/historian-rest-api/v1/offlinecollectors |
| SAMPLE QUERY PARAM GET URL | https://<historianservername>/historian-rest-api/v1/offlinecollectors |
| SAMPLE RESPONSE | { "ErrorCode": 0, "ErrorMessage": null, "Data": [ { "Name": "DISTMACHINE1_Simulation", "ComputerName": "DISTMACHINE1", |

| | |
|---|---|
| | <pre>"Status": "Stopped"<br>},<br>{<br>"Name": "NPI212611749M1_Simulation",<br>"ComputerName": "NPI212611749M1",<br>"Status": "Stopped"<br>},<br>{<br>"Name": "NPI212611749M1_Mqtt",<br>"ComputerName": "NPI212611749M1",<br>"Status": "Stopped"<br>}<br>]<br>}</pre> |
| SAMPLE cURL COMMAND | <pre>curl -i -H "Accept: application/json" -H<br>"Authorization: Bearer <TOKEN>”<br>https://<historianservername>/historian-rest-a<br>pi/v1/offlinecollectors</pre> |

**Table 213. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |

# Managing Data Stores

### The Get Data Stores API

Using the Get Data Stores API, you can view the list of data stores in a system.

| | |
|---|---|
| METHOD | GET |
| URI | <pre>https://<historianservername>/historian-rest-a<br>pi/v1/datastores?dataStoreMask=</pre> |

| | |
|---|---|
| SAMPLE QUERY PARAM GET URL | `https://<historianservername>/historian-rest-api/v1/datastores?dataStoreMask=*` |
| SAMPLE RESPONSE | {<br><br>"ErrorCode": 0,<br><br>"ErrorMessage": null,<br><br>"Data": [<br><br>{<br><br>"Description": "The System Data Store.",<br><br>"Id": "D3C23639-81CD-40F7-9CB0-37484FC5190D",<br><br>"IsDefault": false,<br><br>"IsSystem": true,<br><br>"Name": "System",<br><br>"NumberOfTags": 0,<br><br>"State": 2,<br><br>"DHSStorageName": "System Storage",<br><br>"StorageType": 0,<br><br>"Links": [<br><br>{ |

```
                    "Rel": "self",

                    "Href": "/datastore/System"

                }

            ]

        },

        {

            "Description": "The Scada Buffer
Data Store.",

            "Id":
"39B39D42-DC7A-4048-9BA8-E4BAB4644B0C",

            "IsDefault": false,

            "IsSystem": false,

            "Name": "ScadaBuffer",

            "NumberOfTags": 0,

            "State": 2,

            "DHSStorageName": "xyz",

            "StorageType": 1,

            "Links": [

                {

                    "Rel": "self",
```

```
                "Href":
"/datastore/ScadaBuffer"

            }

        ]

    },

    {

        "Description": "The DHS System Data
Store.",

        "Id":
"56C1DFE9-D0BF-427F-B5D8-B127E38B5C11",

        "IsDefault": false,

        "IsSystem": false,

        "Name": "DHSSystem",

        "NumberOfTags": 0,

        "State": 2,

        "DHSStorageName": "xyz",

        "StorageType": 0,

        "Links": [

            {

                "Rel": "self",
```

```
                "Href":
"/datastore/DHSSystem"

            }

        ]

    },

    {

        "Description": "The User Data
Store.",

        "Id":
"33BA016D-B005-4702-96DB-42CF7238C8FF",

        "IsDefault": true,

        "IsSystem": false,

        "Name": "User",

        "NumberOfTags": 5,

        "State": 2,

        "DHSStorageName": "xyz",

        "StorageType": 0,

        "Links": [

            {

                "Rel": "self",
```

| | |
|---|---|
| | <pre>"Href": "/datastore/User"<br>}<br>]<br>}<br>]<br>}<br>}<br>}</pre> |
| SAMPLE cURL COMMAND | <pre>curl -i -H "Accept: application/json" -H<br>"Authorization: Bearer <TOKEN>”<br>https://<historianservername>/historian-rest-a<br>pi/v1/datastores?dataStoreMask=*</pre> |

**Table 214. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| `dataStoreMask` | The value of the data store mask. | No | String |

**Table 215. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |

### The Get Data Stores of Storage API

Using the Get Data Stores of Storage API, you can view the list of data stores in a location.

| METHOD | GET |
| --- | --- |
| URI | `https://<historianservername>/historian-rest-api/v1/storage/datastores?storageName=` |
| SAMPLE QUERY PARAM GET URL | `https://<historianservername>/historian-resr-api/v1/storage/datastores?storageName=xx` |
| SAMPLE RESPONSE | {<br>"ErrorCode": 0,<br>"ErrorMessage": null,<br>"Data": [<br>{<br>"Description": "The Scada Buffer Data Store.",<br>"Id": "39B39D42-DC7A-4048-9BA8-E4BAB4644B0C",<br>"IsDefault": false,<br>"IsSystem": false,<br>"Name": "ScadaBuffer",<br>"NumberOfTags": 0,<br>"State": 2,<br>"DHSStorageName": "xyz",<br>"StorageType": 1,<br>"Links": [ |

```
            {

                "Rel": "self",

                "Href":
"/datastore/ScadaBuffer"

            }

        ]

    },

    {

        "Description": "The DHS System Data
Store.",

        "Id":
"56C1DFE9-D0BF-427F-B5D8-B127E38B5C11",

        "IsDefault": false,

        "IsSystem": false,

        "Name": "DHSSystem",

        "NumberOfTags": 0,

        "State": 2,

        "DHSStorageName": "xyz",

        "StorageType": 0,

        "Links": [
```

```
            {

                "Rel": "self",

                "Href":
"/datastore/DHSSystem"

            }

        ]

    },

    {

        "Description": "The User Data
Store.",

        "Id":
"33BA016D-B005-4702-96DB-42CF7238C8FF",

        "IsDefault": true,

        "IsSystem": false,

        "Name": "User",

        "NumberOfTags": 5,

        "State": 2,

        "DHSStorageName": "xyz",

        "StorageType": 0,

        "Links": [
```

| | |
|---|---|
| | <pre>{<br>"Rel": "self",<br>"Href": "/datastore/User"<br>}<br>]<br>}<br>]<br>}</pre> |
| SAMPLE cURL COMMAND | <pre>curl -i -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>" https://<historianservername>/historian-rest-api/v1/storage/datastores?storageName=xx</pre> |

**Table 216. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| `storageName` | The name of the location whose data stores you want to view. | Yes | String |

**Table 217. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |

### The Add Datastore API

Using the Add Datastore API, you can create a data store in a Historian server.

**Note:**
This API is applicable only to an Enterprise Historian.

| METHOD | POST |
|---|---|
| URI | `https://<historianservername>/historian-rest-api/v1/datastoretostorage` |
| SAMPLE PATH PARAM GET URI | `https://<historianservername>/historian-rest-api/v1/datastoretostorage`<br><br>`Payload`<br><br>`{`<br><br>`"dataStoreName": "abc",`<br><br>`"storageName": "storage1",`<br><br>`"description": "test",`<br><br>`"isDefault": true`<br><br>`}` |
| SAMPLE RESPONSE | `{`<br><br>`"ErrorCode": 0,`<br><br>`"ErrorMessage": null`<br><br>`}` |
| SAMPLE cURL COMMAND | `curl -i -H "Accept: application/json" -i -H "Content-Type: application/json" -H "Authorization: Bearer <TOKEN>” -d “{ \”dataStoreName \”:\”name\”,\" storageName \": \"sname\",\"description` |

| | |
|---|---|
| | ```\":\" des\",\" isDefault \":false}”<br>-X POST<br>https://<historianservername>/historian-rest-api/v1/datastoretostorage``` |

**Table 218. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| `Payload` | Contains the details of the data store in the JSON format. | Yes | Multiple |

**Table 219. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |

### The Delete Data Store API

Using the Delete Data Store API, you can delete a data store.

| | |
|---|---|
| METHOD | DELETE |
| URI | ```https://<historianservername>/historian-rest-api/v1/datastore``` |
| SAMPLE URI | ```https://<historianservername>/historian-rest-api/v1/datastore<br><br>Payload<br><br>{<br><br>"dataStoreName": "testdatastore"<br><br>}``` |
| SAMPLE RESPONSE | ```{``` |

| | |
|---|---|
| | "ErrorCode": 0,<br><br>"ErrorMessage": null<br><br>} |
| SAMPLE cURL COMMAND | curl -i -H "Accept: application/json" -i -H<br>"Content-Type: application/json"<br>-H "Authorization: Bearer <TOKEN>" -d<br>"{ \"dataStoreName \":\"name\"}" -X DELETE<br>https://<historianservername>/historian-rest-a<br>pi/v1/datastore |

**Table 220. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |

### The Data Store Update API

Using the Data Store Update API, you can modify a data store.

| | |
|---|---|
| METHOD | PUT |
| URI | https://<historianservername>/historian-rest-a<br>pi/v1/dataStore/<data store name> |
| SAMPLE URI | https://<historianservername>/historian-rest-a<br>pi/v1/dataStore/mirror1DS1<br>Payload<br><br>{<br><br>"Description": "testing",<br><br>"IsDefault": true, |

| | |
|---|---|
| | <pre>"IsSystem": false,<br><br>"Name": "mirror1DS1",<br><br>"NumberOfTags": 0,<br><br>"State": 2,<br><br>"DHSStorageName": "mirror1",<br><br>"StorageType": 0,<br><br>}</pre> |
| SAMPLE RESPONSE | <pre>{<br><br>"ErrorCode": 0,<br><br>"ErrorMessage": null<br><br>}</pre> |
| SAMPLE cURL COMMAND | <pre>curl -i -H "Accept: application/json" -i -H<br>"Content-Type: application/json"<br>-H "Authorization: Bearer <TOKEN>" -d “{ \”<br>Description\”:\”des\”,\"IsDefault \": true,<br>\” IsSystem \”:false, \” Name\”:\”<br>mirror1DS1\”,\”NumberOfTags \”:0,\”State\”:2,<br>\”DHSStorageName\”:\”mirror1\”,\”StorageType<br>\”:0,\}” -X PUT<br>https://<historianservername>/historian-rest-a<br>pi/v1/dataStore/mirror1DS1</pre> |

**Table 221. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| `Payload` | Contains the values of the attributes of the data store that you want to change. | Yes | Multiple |

**Table 222. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |

### The Default Data Store Update API

Using the Default Data Store Update API, you can change the default data store.

| METHOD | PUT |
|---|---|
| URI | `https://<historianservername>/historian-rest-api/v1/storage/<location name>` |
| SAMPLE URI | `https://<historianservername>/historian-rest-api/v1/storage/NPI212611749M1`<br><br>`Payload`<br><br>`{`<br><br>`"StorageName": "NPI212611749M1",`<br><br>`"StorageType": 0,`<br><br>`"NumberOfDataStores": 5,`<br><br>`"NumberOfArchivers": 1,`<br><br>`"DataStores": [` |

<table>
<tr><td></td><td>

<pre>
"User",

"testDS1",

"ScadaBuffer",

"testDS2",

"DHSSystem"

],

"Id":
"9CD06AFB-1566-4CE6-99D4-B2F65857F33A",

"IsDefault": true,

"LastModifiedUser": null,

"LastModifiedTime":
"1970-01-01T00:00:00.000Z",

"ArchiverServices": [

"DataArchiver_NPI212611749M1",
"DataArchiver_distamchine1"

]

}
</pre>

</td></tr>
<tr><td>SAMPLE RESPONSE</td><td>

<pre>
{

"ErrorCode": 0,

"ErrorMessage": null,
</pre>

</td></tr>
</table>

| | |
|---|---|
| | }<br> |
| SAMPLE cURL COMMAND | curl -i -H "Accept: application/json" -i -H<br>"Content-Type: application/json"<br>-H "Authorization: Bearer <TOKEN>" -d<br>"{ \"StorageName\":\"name\",\"StorageType\": 0,<br>\"NumberOfDataStores\":5,\"<br>NumberOfArchivers\":1,<br><br>\"IsDefault\":true,\"ArchiverServices\":<br>[\"DataArchiver_NPI212611749M1\",<br>\"DataArchiver_distamchine1"\"]}" -X PUT<br>https://<historianservername>/historian-rest-a<br>pi/v1/storage/NPI212611749M1 |

**Table 223. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| `Payload` | Contains the values of the attributes of the default data store that you want to change. | Yes | Multiple |

**Table 224. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |

# Managing Tags

### The Tags API

The Tags API retrieves the qualified tag name list by a given `nameMask`.

**Note:**
URI format supports asterisks (*) and question marks (?).

| | |
|---|---|
| **METHOD** | GET |
| **URI** | `https://<historianservername>:443/historian-rest-api/v1/tags/{nameMask}/{maxNumber}` |
| **SAMPLE URI** | `https://<historianservername>:443/historian-rest-api/v1/tags?nameMask=*&maxNumber=100` |
| **SAMPLE cURL COMMAND** | `curl -i -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>" https://<nodename>:443/ historian-rest-api/v1/tags?nameMask=*&maxNumber=<Number_Of_Tags>` |

**Table 225. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| `nameMask` | Tagmask that searches for all tags that match the mask and applies the remaining criteria to retrieve data. The mask can include wildcards, such as asterisks (*). | Optional | String |
| `maxNumber` | Maximum tag number provides the limit while returning the results (0 by default). This means that for a query, if using 0, all tags are returned.<br><br>If a negative number is used, then 0 is used for the maxNumber.<br><br>If a positive number is used, then that number of tags is returned. In addition, an error number of +14 notifies the user that there are more than the requested number of tags in the system. | Optional | Integer<br><br>0 by default |

**Table 226. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| ErrorCode | Number | Yes | For example, ErrorCode = 0, which means the operation was successful. |
| ErrorMessage | String | Yes | For example, NULL. |
| tags | String | Yes | Includes the following:<br>• ALT_SENSOR<br>• tagName1<br>• tagName2 |

### The Tagslist API

The Tags List API `GET` method retrieves the list of tags.

When retrieving large tag lists from Historian, you can paginate the response, allowing you to get the next page, go the end, go back one page, and go to the beginning.

#### Request Parameters

You can use wildcards (*, &,?) with string parameters for pattern matching. Results are sorted in ascending tag names. All parameters use the `AND` operator. The `OR` operator is not supported.

All request parameters are optional.

When there are NO wildcard characters (*, &,?) with string parameters for pattern matching, then search would be a `contains` search

**Example**: “dog” pattern will match “dog1”, “dog2”,”dogs”, “dogx”, “dog12”, “dogs are faithful”, “1dog1” and so on. When wildcards (*,&, ?) are used in the search string parameters for pattern matching, then they work as per the wildcard character definition.

? - Single character matching

* - Multi character matching

**Eg1**: “dog?” pattern will match: “dog1”, “dog2”,”dogs”, “dogx” and so on but not “dog12” or “dogs are faithful”

**Eg2**: “dog*” pattern will match “dog1”, “dog2”,”dogs”, “dogx”, “dog12”, “dogs are faithful” and so on but not “1dog1”

| Parameter Name | Data Type | Default | Description |
|---|---|---|---|
| `calctype` | Integer | `-1` | Returns exact match of calc type (0,1,2). |
| `collectiondisabled` | Boolean | If ignored, all types considered. | Must be only true / false, else error out. |
| `collectioninterval` | Integer | 0 – means all intervals | If `collectorinterval` = 0 consider all intervals, else exact match. |
| `collectorcompression` | Boolean | `*` | Returns exact match of collector compression (true/false). |
| `collectorname` | String | `*` | Default `*` means consider all. |
| `collectortype` | Integer | 0 – means consider all collector types | Returns exact match of collector type. |
| `comment` | String | `*` | Default `*` means consider all. |
| `data storename` | String | `*` | Default `*` means consider all. |
| `datatype` | Integer | `0` – means consider all data types | Returns exact match of data type. |
| `description` | String | `*` | Default `*` means consider all. |
| `egudescription` | String | `*` | Default `*` means consider all. |
| `enumeratedset` | String | `*` | Default `*` means consider all. |
| `hasalias` | Boolean | If ignored, all types considered. | Must be only true / false, else error out. |
| `isstale` | Boolean | If ignored, all types considered. | Must be only true / false, else error out. |

| Parameter Name | Data Type | Default | Description |
|---|---|---|---|
| `lastmodified` | String | `1970-01-01T00:00:00` | `>=` is applied so that last modified tag is returned in the result set. |
| `lastmodifieduser` | String | `*` | Default `*` means consider all. |
| `numberofelements` | Integer | 0 | If 0, ignore this parameter else returns exact match of number of elements. |
| `pageno` | Integer | `1`<br>Must be > `1` | If invalid, no data is returned. |
| `pagesize` | Integer | `128`<br>Max `512`<br>Min `2` | If out of range, returns error. Provide 0 to return all tags at once without any pagination. |
| `sourceaddress` | String | `*` | Default `*` means consider all. |
| `tagname` | String | `*` | Default `*` means consider all. |
| `userdefinedtypename` | String | `*` | Default `*` means consider all. |

**The Tags List Pagination Parameters**

When retrieving large tag lists from Historian, you can paginate the response, allowing you to get the next page, go the end, and go back on page and to the beginning. Results with no errors return these pagination parameters:

| Parameter | Value |
|---|---|
| `pagesize` | Current page size. |
| `pageno` | Current page number |
| `totalcount` | Total result other than current page. |
| Links to URLs | All URLs are part of the HTTP response headers. |

| Parameter | Value |
|---|---|
| | • `first` – First page `tags list` URL (can be null if count is 0).<br>• `last` - Last page `tags list` URL (can be null if count is 0).<br>• `prev` – Previous page `tags list` URL (can be null if current page is 1).<br>• Next - Next page `tags list` URL (can be null if current page is last page). |

**Table 227. Sample cURL commands**

| METHOD | GET |
|---|---|
| `SAMPLE cURL COMMAND: [lastmodified]` | curl -i -H "Accept: application/json" -H "Authorization:Bearer <TOKEN>" http://<nodename>:443/historian-rest-api/v1/tagslist?lastmodified=2017-05-01T00:00:00.00Z |
| `SAMPLE cURL COMMAND: [pageno=0]` | curl -i -H "Accept: application/json" -H "Authorization:Bearer <TOKEN>" http://<nodename>:443/historian-rest-api/v1/tagslist?pageno=0 |
| `SAMPLE cURL COMMAND: [pageno=1]` | curl -i -H "Accept: application/json" -H "Authorization:Bearer <TOKEN>" http://<nodename>:443/historian-rest-api/v1/tagslist?pageno=1 |
| `SAMPLE cURL COMMAND: [complete tagslist]` | curl -i -H "Accept: application/json" -H "Authorization:Bearer <TOKEN>" http://<nodename>:443/historian-rest-api/v1/tagslist |

**Example Queries**

The following request returns first page as `pageno` is ignored and `pagesize` is defaulted to 128, all tags are considered:

```
<baseurl>/v1/tagslist
```

The following request returns first page as `pageno` is ignored and `pagesize` is defaulted to 128, all tags are considered that are modified after `2017-05-01T00:00:00.00Z`.

```
<baseurl>/v1/tagslist?lastmodified=2017-05-01T00:00:00.00Z
```

**Example Results**

The following info is returned for each tag from the criteria provided in the request as an array of tag info.

- `tagid` - String
- `tagname` - String
- `description` - String
- `datatype` - Integer
- `collectorname` - String
- `collectortype` - Integer
- `data storename` - String
- `egudescription` - String
- `comment` - String
- `sourceaddress` - String
- `sourceaddress` - String
- `collectioninterval` - Integer
- `collectorcompression` - Boolean
- `lastmodifieduser` - String
- `enumeratedset` - String
- `userdefinedtypename` - String
- `calctype` - Integer
- `isstale` - Boolean
- `lastmodified` - Long
- `lastmodified` - Long
- `lastmodifiedString` – String – In readable format
- `has alias` - Boolean
- `numberofelements` - Integer
- `collectiondisabled` - Boolean

**Example:**


```
{
 "TotalCount": 1031,
 "Page": 1,
 "PageSize": 4,
 "Tags": [
  {
   "Tagid": "adb70ebf-978f-46dd-ac6f-5e863cdb0739",
   "Tagname": "-anilgwxb.Constant",
```

```
  "Description": "anilgwxb.Constant",
  "DataType": 3,
  "CollectorName": "ANILGWXB_Simulation",
  "CollectorType": 2,
  "DataStoreName": "User",
  "EngineeringUnits": "",
  "Comment": "",
  "SourceAddress": "$Constant",
  "CollectionInterval": 1000,
  "CollectorCompression": false,
  "LastModifiedUser": null,
  "EnumeratedSetName": "",
  "UserDefinedTypeName": "",
  "CalcType": 0,
  "IsStale": false,
  "HasAlias": false,
  "NumberOfElements": 0,
  "CollectionDisabled": false,
  "LastModified": 1496992712,
  "LastModifiedString": "2017-06-09T07:18:32Z"
 },
 {
  "Tagid": "88e1f448-643f-465a-95c2-d2bd08870547",
  "Tagname": "anilgwxb.Constant_1%Noise",
  "Description": "anilgwxb.Constant_1%Noise",
  "DataType": 3,
  "CollectorName": "ANILGWXB_Simulation",
  "CollectorType": 2,
  "DataStoreName": "User",
  "EngineeringUnits": "",
  "Comment": "",
  "SourceAddress": "$Constant_1%Noise",
  "CollectionInterval": 1000,
  "CollectorCompression": false,
  "LastModifiedUser": null,
  "EnumeratedSetName": "",
  "UserDefinedTypeName": "",
```

```
   "CalcType": 0,
   "IsStale": false,
   "HasAlias": false,
   "NumberOfElements": 0,
   "CollectionDisabled": false,
   "LastModified": 1496992712,
   "LastModifiedString": "2017-06-09T07:18:32Z"
   },
<SNIP>
 ],
 "Links": {
  "first": "https://anilgwxb:443/historian-rest-api/v1/tagslist?pageno=1&pagesize=4",
  "last": "https://anilgwxb:443/historian-rest-api/v1/tagslist?pageno=258&pagesize=4",
  "prev": null,
  "next": "https://anilgwxb:443/historian-rest-api/v1/tagslist?pageno=2&pagesize=4"
 }
}
```


### The Raw Data API

The Raw Data API queries raw data, such as a number of samples or the time range for a list of tags. If the count is not zero, then the API service returns the number of raw samples taken beginning from the start time. If the count is zero, then the service returns the raw samples taken between the start time and the end time.

| | |
|---|---|
| **METHOD:** | GET, POST |
| **URI:** | **GET**<br>`https://<historianservername>:443/historian-rest-api/v1/datapoints/raw/{tagNames}/{start}/{end}/{direction}/{count}`<br>**POST**<br>`https://<historianservername>:443/historian-rest-api/v1/datapoints/raw/{start}/{end}/{direction}/{count}` |
| **SAMPLE GET URI:** | **Raw By Number** |

| | |
|---|---|
| | Count value is a non-zero positive number, and end time is greater than start time.<br>`https://<historianservername>:443/historian-rest-api/datapoints/raw/tagName1/2013-10-02T11:30:00.111Z/2013-10-02T11:31:11.111Z/0/100https://<historianservername>:443/historian-rest-api/datapoints/raw/tagNames=tagName1&start=2013-10-02T11:30:00.111Z&end=2013-10-02T11:31:11.111Z&count=100&direction=0`<br>**Raw By Time**<br>The count value equals 0.<br>`https://<historianservername>:443/historian-rest-api/datapoints/raw/tagName1/2013-10-02T11:30:00.111Z/2013-10-02T11:31:11.111Z/0/0https://<historianservername>:443/historian-rest-api/datapoints/raw/tagNames=tagName1&start=2013-10-02T11:30:00.111Z&end=2013-10-02T11:31:11.111Z&count=0&direction=0` |
| **SAMPLE POST URI:** | **Raw By Number**<br>Count value is a non-zero positive number, and end time is greater than start time.<br>`https://<historianservername>:443/historian-rest-api/datapoints/raw/2013-10-02T11:30:00.111Z/2013-10-02T11:31:11.111Z/0/100https://<historianservername>:443/historian-rest-api/datapoints/raw/start=2013-10-02T11:30:00.111Z&end=2013-10-02T11:31:11-.111Z&count=100&direction=0`<br>**Raw By Time**<br>The count value equals 0.<br>`https://<historianservername>:443/historian-rest-api/datapoints/raw/2013-10-02T11:30:00.111Z/2013-10-02T11:31:11.111Z/0/0https://<historianservername>:443/historian-rest-api/datapoints/raw/` |

| | |
|---|---|
| | `start=2013-10-02T11:30:00.111Z&end=2013-10-02T11:31:11-.111Z&count=0&direction=0` |
| **SAMPLE cURL COMMAND (GET): [Raw By Number]** | `curl -i -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>" http://<nodename>:443/historian-rest-api/v1/ datapoints/raw/<tagName>/<start time>/<end time>/<direction>/<count>` |
| **SAMPLE cURL COMMAND (GET): [Raw By Time]** | `curl -i -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>" http://<nodename>:443/historian-rest-api/v1/ datapoints/raw/<tagName>/<start time>/<end time>/<direction>/0` |
| **SAMPLE cURL COMMAND (POST): [Raw By Number]** | `curl –X POST -i -H "Content-Type: application/json" -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>" -d "{\"tagNames\":\"<tagName>;<tagName>\"}" http:// <nodename>/ historian-rest-api/v1/ datapoints/raw/ <start time>/<end time>/<direction>/<count>` |
| **SAMPLE cURL COMMAND (POST): [Raw By Time]** | `curl –X POST -i -H "Content-Type: application/json" -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>" -d "{\"tagNames\":\"<tagName>;<tagName>\"}" http:// <nodename>/ historian-rest-api/v1/ datapoints/raw/ start=<start time>&end=<end time>&direction=<direction>&count=<count>` |

**Table 228. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| TagNames | Queries the specified tag names. | Yes | String |
| Start | Start time of the query, in ISO data format (such as YYYY-MM-DDTHH:mm:ss.SSSZ). | Yes | DateTime |
| End | End time of the query, in ISO data format (such as YYYY-MM-DDTHH:mm:ss.SSSZ). | Yes | DateTime |
| Direction | Specifies the direction (Forward or Backward | Yes | Integer, with a value such as 1. |

**Table 228. Query Parameters (continued)**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| | from the starting time) of data sampling from the archive. The default value is Forward (1). | | |
| Count | Count of archive values within each calculation interval. | Yes | Integer, with a value such as 100. |

**Table 229. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| ErrorCode | Integer | Yes | For example, 0. |
| ErrorMessage | String | Yes | For example, NULL. |
| Data | String | Yes | The object container for the following parameters:<br>**DataType**<br>DoubleFloat, which stores decimal values up to 15 places.<br>**ErrorCode**<br>Value is 0, which means the operation was successful.<br>**TagName**<br>Example: TagName1.<br>**Samples**<br>Provides TimeStamp, Value, Quality, and DtatAttributes for each sample. Where, DataAttributes is |

**Table 229. Response Parameters (continued)**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| | | | the detailed information regarding the Quality.<br>For example,<br>`"TimeStamp": "2013-10-02T11:30:00.111Z", "Value": 34.26155", "Quality": 3, "DataAttributes": []` |

### The Interpolated Data API

The Interpolated Data API queries interpolated values for a list of tags. If the start time equals the end time, the request returns one sample.

| | |
|---|---|
| **METHOD:** | GET, POST |
| **URI:** | **GET**<br>`https://<historianservername>:443/historian-rest-api/v1/datapoints/interpolated/{tagNames}/{start}/{end}/{count}/{intervalMs}`<br>**POST**<br>`https://<historianservername>:443/historian-rest-api/v1/datapoints/interpolated/{start}/{end}/{count}/{intervalMs}` |
| **SAMPLE GET URI:** | `https://<historianservername>:443/historian-rest-api/v1/datapoints/interpolated/tagName1/2013-10-02T11:30:00.111111Z/2013-10-02T11:31:11.111Z/100/10000` |
| **SAMPLE POST URI:** | `https://<historianservername>:443/historian-rest-api/v1/datapoints/interpolated/2013-10-02T11:30:00.111111Z/2013-10-02T11:31:11.111Z/100/10000` |
| **SAMPLE cURL COMMAND (GET):** | `curl -i -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>" http://<nodename>:443/ historian-rest-api/v1/` |

| | |
|---|---|
| | `datapoints/interpolated/<tagName>/<start time>/<end time>/<count>/<intervalMS>` |
| **SAMPLE cURL COMMAND (POST):** | `curl -i –X POST -H "Content-Type: application/json" -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>” -d “{\”tagNames\”:\”<tagName>\”}” http://<nodename>:443/historian-rest-api/v1/ datapoints/interpolated/<start time>/<end time>/<count>/<intervalMS>` |

**Table 230. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| TagName | Queries the tag names specified. | Yes | String |
| Start | Start time of the query, in ISO data format (such as YYYY-MM-DDTHH:mm:ss.SSSZ). | Yes | DateTime |
| End | End time of the query, in ISO data format (such as YYYY-MM-DDTHH:mm:ss.SSSZ). | Yes | DateTime |
| Count | Count of archive values within each calculation interval. | Yes | Integer, with a value such as 100. |
| intervalMS | Interval in milliseconds. | Yes | 64-bit signed integer, with a value such as 10000. |

**Table 231. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| ErrorCode | Integer | Yes | For example, 0. |
| ErrorMessage | String | Yes | For example, NULL. |
| Data | String | Yes | The object container for the following parameters: |

**Table 231. Response Parameters (continued)**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| | | | **DataType**<br>DoubleFloat, which stores decimal values up to 15 places.<br>**ErrorCode**<br>Value is 0, which means the operation was successful.<br>**TagName**<br>Example is TagName1.<br>**Samples**<br>Provides TimeStamp, Value and Quality for each sample. For example, TimeStamp = 2013-10-02T11:30:00.111Z, Value = 34.26155, and Quality = 3. |

### The Current Value API

The Current Value API queries the current value data and reads the current values for a list of tags. If the start time is equal to end time, the request returns one sample.

| | |
|---|---|
| **METHOD:** | GET, POST |
| **URI:** | **GET**<br>`https://<historianservername>:443/historian-rest-api/v1/datapoints/raw/{tagNames}/{start}/{end}/{direction}/{count}`<br>**POST**<br>`https://<historianservername>:443/historian-rest-api/v1/datapoints/currentvalue` |

| | |
|---|---|
| **SAMPLE GET URI:** | `https://<historianservername>:443/historian-rest-api/v1/datapoints/currentvalue?tagNames=tagName1` |
| **SAMPLE POST URI:** | `https://<historianservername>:443/historian-rest-api/v1/datapoints/currentvalue` |
| **SAMPLE cURL COMMAND (GET):** | `curl -i -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>" http://<nodename>:443/historian-rest-api/v1/datapoints/currentvalue/<tagName>` |
| **SAMPLE cURL COMMAND (POST):** | `curl -i –X POST -H "Content-Type: application/json" -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>" -d "{\"tagNames\":\"<tagName>\"}" http://<nodename>:443/historian-rest-api/v1/ datapoints/currentvalue` |

**Table 232. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| TagNames | Queries the specified tag names. | Yes | String |

**Table 233. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| ErrorCode | Integer | Yes | For example, 0. |
| ErrorMessage | String | Yes | For example, NULL. |
| Data | String | Yes | The object container for the following parameters:<br>**DataType**<br>DoubleFloat, which stores decimal values up to 15 places.<br>**ErrorCode**<br>Value is 0, which means the operation was successful. |

**Table 233. Response Parameters (continued)**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| | | | **TagName**<br>Example is TagName1.<br>**Samples**<br>Provides TimeStamp, Value and Quality for each sample. For example, TimeStamp = 2014-01-01T12:00:00Z, Value = 34.26155, and Quality = 3. |

### The Calculated Data API

The Calculated Data API queries the calculated data for a list of tags. Data can be requested using a number of samples or a time range for a list of tags. If the count is not zero, the service returns the number of raw samples beginning from the start time. If the count is zero, the services uses the interval, start time, and end time to calculate the required sample number.

| | |
|---|---|
| **METHOD:** | GET, POST |
| **URI:** | **GET**<br>`https://<historianservername>:443/historian-rest-api/v1/datapoints/calculated/{tagNames}/{start}/{end}/{calculationMode}/{count}/{intervalMs}`<br>**POST**<br>`https://<historianservername>:443/historian-rest-api/v1/datapoints/calculated/{start}/{end}/{calculationMode}/{count}/{intervalMs}` |
| **SAMPLE GET URI:** | **Number of Samples**<br>`https://<historianservername>:443/historian-rest-api/v1/datapoints/calculated/tagName1/2013-10-02T11:30:00.111Z/2013-10-02T11:31:11.111Z/1/100/1000` |

| | |
|---|---|
| | **Time Range for List of Tags**<br>`https://<historianservername>:443/historian-rest-api/v1/datapoints/calculated?tagNames=tagName1&start=2013-10-02T11:30:00.111Z&end=2013-10-02T11:31:11-.111Z&count=100&calculationMode=1&intervalMs=1000` |
| **SAMPLE POST URI:** | **Number of Samples**<br>`https://<historianservername>:443/historian-rest-api/v1/datapoints/calculated/2013-10-02T11:30:00.111Z/2013-10-02T11:31:11.111Z/1/100/1000`<br>**Time Range for List of Tags**<br>`https://<historianservername>:443/historian-rest-api/v1/datapoints/calculated?start=2013-10-02T11:30:00.111Z&end=2013-10-02T11:31:11-.111Z&count=100&calculationMode=1&intervalMs=1000` |
| **SAMPLE cURL COMMAND (GET):** | `curl -i -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>” http://<nodename>:8843/ historian-rest-api/v1/ datapoints/calculated/<tagName>/<start time>/<end time>/<count>/<calculation mode>/<intervalMS>` |
| **SAMPLE cURL COMMAND (POST):** | `curl -i –X POST -H "Content-Type: application/json" -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>” -d “{\”tagNames\”:\”<tagName>\”}” http://<nodename>:8843/ historian-rest-api/v1/ datapoints/calculated/<start time>/<end time>/<count>/<calculationmode>/<intervalMS>` |

**Table 234. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| TagNames | GE identifier for a location. | Yes | 1000000106 |
| Start | Start time of the query, in ISO data format (such as YYYY-MM-DDTHH:mm:ss.SSSZ). | Yes | DateTime |

**Table 234. Query Parameters (continued)**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| End | End time of the query, in ISO data format (such as YYYY-MM-DDTHH:mm:ss.SSSZ). | Yes | DateTime |
| Count | Count of archive values within each calculation interval. | Yes | Integer, with a value such as 100. |
| Calculation Mode | | Yes | Integer, with a value such as 1. For more information on the calculation modes, refer to calculation modes for REST API. |
| IntervalMS | Interval in milliseconds. | | 64-bit signed integer, with a value such as 1000. |

**Table 235. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| ErrorCode | Integer | Yes | For example, 0. |
| ErrorCode | String | Yes | For example, NULL. |
| Data | String | Yes | The object container for the following parameters:<br>**DataType**<br>DoubleFloat, which stores decimal values up to 15 places.<br>**ErrorCode**<br>Value is 0, which means the operation was successful.<br>**TagName** |

**Table 235. Response Parameters (continued)**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| | | | Example is TagName1.<br>**Samples**<br>Provides TimeStamp, Value and Quality for each sample. For example, TimeStamp = 2013-10-02T11:30:00.111Z, Value = 34.26155, and Quality = 3. |

### The Sampled Data API

The Sampled Data API queries the sampled data for a list of tags. Data can be requested using a number of samples or a time range for a list of tags. If the count is not zero, the service returns the number of raw samples beginning from the start time. If the count is zero, the services uses the interval, start time, and end time to calculate the required sample number.

**Note:**
For the query, you can also use optional parameters such as FilterMode and ReturnDataFields. Unused parameters can be omitted.

| | |
|---|---|
| **METHOD:** | GET, POST |
| **URI:** | **GET**<br>`https://<historianservername>:443/historian-rest-api/v1/datapoints/sampled`<br>**POST**<br>`https://<historianservername>:443/historian-rest-api/v1/datapoints/sampled` |
| **SAMPLE GET URI:** | `https://<historianservername>:443/historian-rest-api/v1/datapoints/sampled?tagNames=tagName1&start=2013-10-02T11:30:00.111Z&end=2013-10-02T11:31:11-` |

| | |
|---|---|
| | `.111Z&samplingMode=1&calculationMode=1&direction=0&count=0&intervalMs=1000` |
| **SAMPLE POST URI:** | `https://<historianservername>:443/historian-rest-api/v1/datapoints/sampled` |
| **SAMPLE cURL COMMAND (GET):** | `curl -i -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>" http://<nodename>:443/ historian-rest-api/v1/datapoints/sampled/<tagName>/<start time>/<end time>/<direction>/<count>/<intervalMS>` |
| **SAMPLE cURL COMMAND (POST):** | `curl -i –X POST -H "Content-Type: application/json" -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>" -d "{ \"tagNames\":\"<tagName>\", \"start\": \"<start>\", \"end\": \"<end>\", \"samplingMode\": <samplingMode>, \"calculationMode\": <calculationMode>, \"direction\": <direction>, \"count\": <count>, \"returnDataFields\": <returnDataFields>, \"intervalMs\": <intervalMs>, \"queryModifier\": <queryModifier>, \"filterMode\": <filterMode>, \"filterExpression\": \"<filterExpression>\"}" http://<nodename>:443/historian-rest-api/v1/datapoints/sampled` |

**Table 236. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| TagNames | Queries the tag names specified. | Yes | String |
| Start | Start time of the query, in ISO data format (such as YYYY-MM-DDTHH:mm:ss.SSSZ). | Yes | DateTime |
| End | End time of the query, in ISO data format (such as YYYY-MM-DDTHH:mm:ss.SSSZ). | Yes | DateTime |
| SamplingMode | Also known as SamplingModeType. | Optional | Integer, with a value such as 1. |

**Table 236. Query Parameters (continued)**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| CalculationMode | Also known as CalculationModeType. | Optional | Integer, with a value such as 1. |
| Direction | Specifies the direction (Forward or Backward from the starting time) of data sampling from the archive. The default value is Forward (1). | Optional | Integer, with a value such as 1. |
| Count | The count of archive values within each calculation interval. | Optional | Integer, with a value such as 0. |
| IntervalMS | Interval in milliseconds. | Optional | 64-bit signed integer, with a value such as 1000. |

**Table 237. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| ErrorCode | Integer | Yes | For example, 0. |
| ErrorCode | String | Yes | For example, NULL. |
| Data | String | Yes | The object container for the following parameters:<br>**DataType**<br>DoubleFloat, which stores decimal values up to 15 places.<br>**ErrorCode**<br>Value is 0, which means the operation was successful.<br>**TagName** |

**Table 237. Response Parameters (continued)**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| | | | Example is TagName1.<br>**Samples**<br>Provides TimeStamp, Value and Quality for each sample. For example, TimeStamp = 2013-10-02T11:30:00.111Z, Value = 34.26155, and Quality = 3. |

### The Trend Data API

The Trend Data API queries the trend data for a list of tags.

**Note:**
For the query, you can also use optional parameters such as FilterMode and StatisticsItemFilter. Unused parameters can be omitted.

| | |
|---|---|
| **METHOD:** | GET, POST |
| **URI:** | **GET**<br>`https://<historianservername>:443/historian-rest-api/v1/datapoints/trend`<br>**POST**<br>`https://<historianservername>:443/historian-rest-api/v1/datapoints/trend` |
| **SAMPLE GET URI:** | `https://<historianservername>:443/historian-rest-api /v1/datapoints/trend?tagNames=tagName1&start=2013-10-02T11: 30:00.111Z&end=2013-10-02T11:31:11.111Z&samplingMode=1&calculationMode=1 &direction=0&count=0&intervalMs=1000` |
| **SAMPLE POST URI:** | `https://<historianservername>:443/historian-rest-api/v1/data-points/trend` |

| | |
|---|---|
| **SAMPLE cURL COMMAND (GET):** | `curl -i -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>" http://<nodename>:443/ historian-rest-api/v1/datapoints/trend/<tagName>/<start time>/<end time>/<samplingMode>/<calculationMode>/<direction>/<count>/<intervalMS>` |
| **SAMPLE cURL COMMAND (POST):** | `curl -i –X POST -H "Content-Type: application/json" -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>" -d "{ \"tagNames\":\"<tagName>\", \"start\": \"<start>\", \"end\": \"<end>\", \"samplingMode\": <samplingMode>, \"calculationMode\": <calculationMode>, \"direction\": <direction>, \"count\": <count>, \"returnDataFields\": <returnDataFields>, \"intervalMs\": <intervalMs>, \"queryModifier\": <queryModifier>, \"filterMode\": <filterMode>, \"filterExpression\": \"<filterExpression>\"}" http://<nodename>:443/historian-rest-api/v1/datapoints/trend` |

**Table 238. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| TagNames | Queries the tag names specified. | Yes | String |
| Start | Start time of the query, in ISO data format (such as YYYY-MM-DDTHH:mm:ss.SSSZ). | Yes | DateTime |
| End | End time of the query, in ISO data format (such as YYYY-MM-DDTHH:mm:ss.SSSZ). | Yes | DateTime |
| SamplingMode | Also known as SamplingModeType. | Optional | Integer, with a value such as 1. |
| CalculationMode | Also known as CalculationModeType. | Optional | Integer, with a value such as 1. |
| Direction | Specifies the direction (Forward or Backward | Optional | Integer, with a value such as 1. |

**Table 238. Query Parameters (continued)**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| | from the starting time) of data sampling from the archive. The default value is Forward (1). | | |
| Count | The count of archive values within each calculation interval. | Optional | Integer, with a value such as 0. |
| IntervalMS | Interval in milliseconds. | Optional | 64-bit signed integer, with a value such as 1000. |

**Table 239. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| ErrorCode | Integer | Yes | For example, 0. |
| ErrorMessage | String | Yes | For example, NULL. |
| Data | String | Yes | The object container for the following parameters:<br>**TagName**<br>Name of the tag, such as ahistfile.Simulation00001.<br>**TagSource**<br>Location where tags are being searched for.<br>**DataType**<br>Float, which stores decimal values up to 6 places.<br>**Trend**<br>Provides TimeStamp, Value and Quality for each sample. For ex- |

**Table 239. Response Parameters (continued)**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| | | | ample, TimeStamp = 2016-03-15T04:53:17.000Z, Value = 170903.6563, and Quality = True. |

### The Add Single Tag API

For the Add Single Tag API, you can add a new tag to Historian, and the tag name and data type must be provided in the payload (parameter) of the method. All other tags are optional. If a property is provided, the respective validation is performed at the server end. If the tag exists, then any new properties provided in the payload are applied to the existing tag.

| | |
|---|---|
| **METHOD:** | POST |
| **URI:** | `https://<historianservername>:443/historian-rest-api/v1/tags/addtag` |
| **SAMPLE URI:** | `https://<historianservername>:443/historian-rest-api/v1/tags/addtag`<br><br>`Payload:`<br><br>`{`<br>`"Name" : "SampleTag",`<br>`"DataType" : 3`<br>`}` |
| **SAMPLE cURL COMMAND:** | `curl -i -H "Accept: application/json" -i -H "Content-Type: application/json" -H "Authorization: Bearer <TOKEN>” -d “{ \”Name\”: \”Sampletag\”,\"DataType\":3}” -X POST https://<historianservername>:443/historian-rest-api/v1/tags/addtag` |

**Table 240. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| Payload | JSON array of PropertyName and PropertyValue. | Yes. "Name" and "DataType" properties are required. | Multidata types. See Payload Parameter *(on page 1245)* for a list of tag prop- |

**Table 240. Query Parameters (continued)**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| | | All other properties are optional. | erties used to update a tag configuration. |

**Sample Response**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| ErrorCode | Integer | Yes | For example, 0. |
| ErrorMessage | String | Yes | For example, NULL. |

### The Add Bulk Tags API

For the Add Bulk Tags API, you can add new tags to Historian using an array, and the tag names and data types must be provided in the payload (parameter) of the method. All other tags are optional. If a property is provided, the respective validation is performed at the server end. If the tags exist, then any new properties provided in the payload are applied to the existing tags. The payload is be an array of tags defined.

| | |
|---|---|
| **METHOD:** | POST |
| **URI:** | `https://<historianservername>:443/historian-rest-api/v1/tags/addtags` |
| **SAMPLE URI:** | `https://<historianservername>:443/historian-rest-api/v1/tags/addtags`<br><br>`Payload:`<br><br>`[`<br>`{`<br>`"Name" : "SampleTag1",`<br>`"DataType" : 3`<br>`},`<br>`{`<br>`"Name" : "SampleTag2",`<br>`"DataType" : 3`<br>`}`<br>`]` |

| | |
|---|---|
| **SAMPLE cURL COMMAND:** | `curl -i -H "Accept: application/json" -i -H "Content-Type: application/json" -H "Authorization: Bearer <TOKEN>" -d "[{ \"Name\": \"Sampletag1\"}, { \"Name\":\"Sampletag2\"}]" -X POST https://<historianservername>:443/historian-rest-api/v1/tags/addtags` |

**Table 241. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| Payload | JSON array tags with individual tags of PropertyName and PropertyValue. | Yes. "Name" and "DataType" properties are required. All other properties are optional. | Multidata types. See Payload Parameter *(on page 1245)* for a list of tag properties used to update a tag configuration. |

**Table 242. Response Parameters**

| Parameter | Data Type | Exists? | Description |
|---|---|---|---|
| TagName | String | Yes | Tag name. |
| ErrorCode | Integer | Yes | For example, 0. |
| ErrorMessage | String | Yes | For example, NULL. |

### The Update Tag Configuration API

The Update Tag Configuration API allows you to set or modify any tag property values. You cannot, however, rename a tag using this API.

| | |
|---|---|
| **METHOD:** | PUT |
| **URI:** | `https://<historianservername>:443/historian-rest-api/v1/tags/properties/tagName` |
| **SAMPLE DELETE URI:** | `https://<historianservername>:443/historian-rest-api/v1/tags/properties/tagName`<br><br>`Payload:`<br>`{`<br>`"PropertyName" : "PropertyValue"`<br>`}` |

| | |
|---|---|
| **SAMPLE cURL COMMAND:** | `curl -i -H "Accept: application/json" -i -H "Content-Type: application/json" -H "Authorization: Bearer <TOKEN>" -d "{ \"Description\":\"SampleDesc\"}" -X PUT https://<historianserver-name>:443/historian-rest-api/v1/tags/properties/tagName` |

**Table 243. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| tagName | Tag name for which properties need to be set or modified. | Yes | String |
| Payload | JSON array of PropertyName and PropertyValue. | At least one property must be provided. | Multidata types. See Payload Parameter *(on page 1245)* for a list of tag properties used to update a tag configuration. |

**Table 244. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| ErrorCode | Integer | Yes | For example, 0. |
| ErrorMessage | String | Yes | For example, NULL. |

### The Get Tag Properties API

You can use this API to specify which properties are required for retrieval. If no property names are provided, then all properties are retrieved. When using the Get Tag Properties method, requesting a non-existent tag name returns an error.

| | |
|---|---|
| **METHOD:** | GET / POST |
| **URI: (GET)** | `https://<historianservername>:443/historian-rest-api/v1/tags/properties/tagName`<br><br>This URI returns all tag properties. |
| **URI: (POST)** | `https://<historianservername>:443/historian-rest-api/v1/tags/properties/tagName` |

| | |
|---|---|
| | Payload<br><br>{<br>"PropertyName1" : 1,<br>"PropertyName2" : 1<br>} |
| **SAMPLE GET URI:** | `https://<historianservername>:443/historian-rest-api/v1/tags/properties/tagName` |
| **SAMPLE POST URI:** | `https://<historianservername>:443/historian-rest-api/v1/tags/properties/tagName`<br><br>Payload:<br>{<br>"Description" : 1<br>} |
| **SAMPLE cURL GET COMMAND:** | `curl -i -H "Accept: application/json" -i -H "Content-Type: application/json" -H "Authorization: Bearer <TOKEN>” -X GET https://<historianservername>:443/historian-rest-api/v1/tags/properties/tagName` |
| **SAMPLE cURL POST COMMAND:** | `curl -i -H "Accept: application/json" -i -H "Content-Type: application/json" -H "Authorization: Bearer <TOKEN>” -d “{ \”Description\”: 1}” -X POST https://<historianservername>:443/historian-rest-api/v1/tags/properties/tagName` |

**Table 245. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| tagName | Tag name for which properties need to be retrieved. | Yes | String |
| Payload | JSON array of PropertyName and boolean (true/false). | At least one property must be provided. | Multi data types. See Payload Parameter *(on page 1245)* for a list of tag properties used to update a tag configuration. |

**Note:**
The query payload contains all the tag properties you want returned from the server. In the Update Tag Config method, you need to provide the actual tag property value. However, in the Get Tag Properties method, you need to provide the property and a value of 1 (true), to allow it to be read from the server and returned.

**Table 246. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| ErrorCode | Integer | Yes | For example, 0. |
| ErrorMessage | String | Yes | For example, NULL. |
| Name | String | Optional | If no error, then the tag name of query is returned and all requested parameters. |

### The Delete Tag API

The Delete Tag API provides the ability to delete an existing tag from the Historian server.

Its URI format supports question marks (?).

| | |
|---|---|
| **METHOD:** | DELETE |
| **URI:** | `https://<historianservername>:443/historian-rest-api/v1/tags/tagName?{permanentDelete}` |
| **SAMPLE DELETE URI:** | `https://<historianservername>:443/historian-rest-api/v1/tags/tagName?permanentDelete=true` |
| **SAMPLE CURL COMMAND:** | `curl -i -H "Authorization: Bearer <TOKEN>” -X DELETE https://<historianservername>:443/historian-rest-api/v1/tags/tagName?permanentDelete=<true\|false>` |

**Table 247. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| tagName | Name of the tag to be deleted. | Yes | String |

**Table 247. Query Parameters (continued)**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| permanentDelete | Deletes the tag permanently from the Historian server if the value passed in is true. If the parameter is not provided, then permanentDelete is assumed to be false. | Optional (false is default) | Boolean, true or false |

**Table 248. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| ErrorCode | Number | Yes | For example, ErrorCode=0, which means the operation was successful. |
| ErrorMessage | String | Yes | For example, NULL. |

### The Query Results API

The Query Results API enables you to include the number of samples required, by providing an end point to configure query results.

The minimum number of samples should be 1000.

| | |
|---|---|
| **METHOD:** | PUT |
| **URI:** | `https://<historianservername>:443/historian-rest-api/v1/data-points/ configuration/{maxDataQueryResultSize}` |
| **SAMPLE URI:** | `https://<historianservername>:443/historian-rest-api/v1/data-points/ configuration?maxDataQueryResultSize=6000` |
| **SAMPLE CURL COMMAND:** | `curl -i -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>” https://<nodename>:443/ historian-rest-api/v1/data-points/configuration? maxDataQueryResultSize=<Number_Of_Query_Results>` |

**Table 249. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| maxDataQueryResultSize | Maximum samples that should be configured as part of Query Results. | Yes | Integer |

**Table 250. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| Error Code | Integer | Yes | For example, 0. |
| Error Message | String | Yes | For example, NULL. |
| Maximum DataQueryResultSize | Integer | Yes | Returns the number of samples that were configured as part of query. For example, based on the sample URI, this parameter will be 6000. |

### The Tag Rename API

This API allows the administrator to rename tags.

| METHOD: | PUT |
|---|---|
| **URI** | `https://<historianservername>:443/historian-rest-api/v1/tags/tagrename/oldtagname/newtagname?{truerename}` |
| **SAMPLE URI** | `https://<historianservername>:443/historian-rest-api/v1/tags/tagrename/GDW14NV2E.Simulation0000101/GDW14NV2E.Simulation0000101newname?truerename= <true \| false>` |
| **SAMPLE CURL COMMAND** | `curl -i -H "Accept: application/json" -i -H "Content-Type: application/json"-H "Authorization: Bearer <TOKEN> -X PUT https://<historianservername>:443/historian-rest-api/v1/tags/tagrename/<oldtagname>/<newtagname>?truerename=<true \| false>` |

**Table 251. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| oldtagname | Tag which is to be renamed. | Yes | String |
| newtagname | New name for the selected tag. | Yes | String |
| truerename | Renames the tag permanently if the value entered is true.<br>Creates an alias if the value entered is false. | Optional (false is default) | Boolean (true or false) |

**Table 252. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| Error Code | Integer | Yes | For example, 0. |
| Error Message | String | Yes | For example, NULL. |
| Data | List | Yes | Returns all the properties of the tag. |

### The Write Tag API

Write Tag Data API enables you to create data for tags. You can write data to a tag for different data types such as integer, float, array, multifield and so on. Once created, you can view the data using other end points. Only REST API Administrator and users with write permission can perform this operation.

| Method | POST |
|---|---|
| **URI** | `https://<historianservername>:443/historian-rest-api /v1/datapoints/create` |
| **SAMPLE URI** | `https://<historianservername>:443/historian-rest-api /v1/datapoints/create` |

| Method | POST |
| --- | --- |
| | Payload<br><br>{<br>"TagName": "GDW14NV2E.Simulation00015",<br>"samples": [<br>{<br><br>"TimeStamp":<br>"2019-09-17T15:58:00.000Z",<br><br>"Value": "1",<br><br>"Quality": 3<br>}<br>]<br>} |
| **SAMPLE RESPONSE** | {<br>"ErrorCode": 0,<br>"ErrorMessage": ""<br>} |
| **SAMPLE CURL COMMAND** | curl -i -H "Accept: application/json"<br>-i -H "Content-Type: application/json"<br>-H "Authorization: Bearer <TOKEN>”<br>-d “{ \”TagName\”:\”GDW14NV2E.Simulation00015\”,\"samples\":[{\"TimeStamp\":\ "2019-09-17T15:58:00.000Z\",\"Value\": \"1\",\"Quality\": 3}]}” -X POST<br>https://<historianservername>:443/historian-rest-api/v1/datapoints/create |

**Table 253. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| Payload | JSON format of Property Name and Property Value. | Yes | Multi-data types. It can have integer, float, array, multifield data types. |

**Table 254. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| Error Code | Integer | Yes | For example, ErrorCode = 0, which means the operation was successful. |
| Error Message | String | Yes | For example, NULL. |

## Calculation Modes for REST API

Calculation modes are used on retrieval of data that has already been collected to the archive. Calculation modes are used when the sampling mode is set to "Calculated". The table below lists the different calculation modes and their corresponding value that you can pass as part of the query parameters to manage tags.

| Calculation Mode | Value |
|---|---|
| Average | 1 |
| StandardDeviation | 2 |
| Total | 3 |
| Minimum | 4 |
| Maximum | 5 |
| Count | 6 |
| RawAverage | 7 |
| RawStandardDeviation | 8 |
| RawTotal | 9 |

| Calculation Mode | Value |
|---|---|
| MinimumTime | 10 |
| MaximumTime | 11 |
| TimeGood | 12 |
| StateCount | 13 |
| StateTime | 14 |
| OPCAnd | 15 |
| OPCOr | 16 |
| FirstRawValue | 17 |
| FirstRawTime | 18 |
| LastRawValue | 19 |
| LastRawTime | 20 |
| TagStats | 21 |
| DeltaPos | 22 |
| DeltaNeg | 23 |
| Delta | 24 |

## Managing Alarms and Events

**Overview**

Historian includes Alarms and Events (A&E) archiving to provide the ability to retrieve and store Alarms and Events Data from any OPC-compliant A&E server through the OPC Classic Alarms and Events collector.

Alarms are generally defined as tags going into an abnormal condition. For example, an alarm could be set on a boiler when it reaches a specified temperature. Alarms usually have a well-defined life cycle, which is defined by the individual data sources the alarms are collected from (iFIX, for example). They enter an alarm state, are generally acknowledged, and then return to normal.

Events are generally defined as activities in a system that occur only once. For example, a user logging on to a device is an event. When viewing this data in Historian, each event is

returned as a record. The Historian REST API provide API's to store, retrieve, backup, restore, and delete alarms and events data.

**Add Alarm or Event**

Add alarm API allows you to create alarm in Historian Server.

| METHOD | POST |
|---|---|
| URI | `https://<historianservername>:443/historian-rest-api/v1/alarms/create` |
| SAMPLE URI | `https://<historianservername>:443/historian-rest-api/v1/alarms/create`<br>Payload<br>`{`<br>`"DataSource":"Simulation",`<br>`"AlarmsInfo":[`<br>`{`<br>`"ItemId":"GF7DQL63E_Simulation.Simulation00001",`<br>`"Source":"Simulation",`<br>`"DataSourceName":"Simulation",`<br>`"Tagname":"GF7DQL63E_Simulation.Simulation00001",`<br>`"AlarmType":4,`<br>`"EventCategory":"SimCategory",`<br>`"ConditionName":"Simulation",`<br>`"SubConditionName":"Simulation",`<br>`"StateTransitionTime":"2022-09-21T5:58:00.000Z",`<br>`"StartTime":"2022-09-21T5:58:00.000Z",`<br>`"EndTime":"2022-09-21T6:58:00.000Z",`<br>`"AckTime":"2022-09-21T6:58:00.000Z",`<br>`"TimeStamp":"2022-09-21T5:58:00.000Z",`<br>`"Message":"This is simulation alarm",` |

| | |
|---|---|
| | "Severity":100,<br>"Actor":"Simulator",<br>"Quality":3<br>}<br>]<br>} |
| SAMPLE CURL COMMAND | curl -i -H "Accept: application/json" -i<br>-H "Content-Type: application/json" -H<br>"Authorization: Bearer ” -d “{<br>\"DataSource\":\"Simulation\",<br>\”AlarmsInfo\”:[<br>{<br>\”ItemId\”:<br>\”GF7DQL63E_Simulation.Simulation00001\”,<br>\”Source\”:\”Simulation\”,<br>\”DataSourceName\”:\”Simulation\”,<br>\”Tagname\”:<br>\”GF7DQL63E_Simulation.Simulation00001\”,<br>\”AlarmType\”:4,<br>\”EventCategory\”:\”SimCategory\”,<br>\”ConditionName\”:\”Simulation\”,<br>\”SubConditionName\”:\”Simulation\”,<br>\”StateTransitionTime\”:<br>\”2022-09-21T5:58:00.000Z\”,<br>\”StartTime\”:\”2022-09-21T5:58:00.000Z\”,<br>\”EndTime\”:\”2022-09-21T6:58:00.000Z\”,<br>\”AckTime\”:\”2022-09-21T6:58:00.000Z\”,<br>\”TimeStamp<br>\”:\”2022-09-21T5:58:00.000Z\”,<br>\”Message\”:\”This is simulation<br>alarm\”, |

```
        \"Severity\":100,
        \"Actor\":\"Simulator\",
        \"Quality\":3
      }
   ]
}" -X PUT https://:443/historian-rest-api/v1/
 alarms/create
```

**Table 255. Query Parameters**

| Parameter | Data Type | Description |
|---|---|---|
| `DataSource` | `VT_-BSTR` | The collector interface name associated with the alarm or event. |
| `ItemId` | `VT_-BSTR` | The OPC `ItemID` of the alarm. This contains the source address of the data access tag with which the alarm is associated. This can contain a `NULL` value if an alarm is not associated with a tag. |
| `Source` | `VT_-BSTR` | The unique identifier used by the OPC A&E Collector for the alarm or event. |
| `Data-SourceName` | `VT_-BSTR` | The collector interface name associated with the alarm or event. |
| `Tagname` | `VT_-BSTR` | The Historian tag name associated with the alarm. This value is `NULL` unless the tag is also collected by Historian |
| `AlarmType` | `VT_-BSTR` | The alarm type:<br>• `Alarms`: In Historian, the full life cycle of an alarm is stored as a single record in the alarm archive.<br>• `Alarm_History`: The separate transitions for all alarms. One row per transition is returned.<br>• `Events`: The simple and tracking events. |
| `EventCate-gory` | `VT_-BSTR` | The OPC event category of the alarm or event. |
| `Condition-Name` | `VT_-BSTR` | The OPC condition of the alarm. Does not apply to event data. This value combined with the Source value comprises an alarm. |

**Table 255. Query Parameters (continued)**

| Parameter | Data Type | Description |
|---|---|---|
| `SubConditionName` | `VT_BSTR` | The OPC subcondition of the alarm. Does not apply to event data. This value represents the state of the alarm. |
| `StateTransitionTime` | `VT_DB-Time-Stamp` | The time the state of the tag was generated. |
| `StartTime` | `VT_DB-Time-Stamp` | The start time or timestamp of the alarm or event. |
| `EndTime` | `VT_DB-Time-Stamp` | The end time of the alarm. Does not apply to event data. |
| `AckTime` | `VT_DB-Time-Stamp` | The time the alarm was acknowledged. Does not apply to event data. |
| `TimeStamp` | `VT_DB-Time-Stamp` | The time the alarm was generated. |
| `Message` | `VT_BSTR` | The message attached to the alarm or event. |

**Table 255. Query Parameters (continued)**

| Parameter | Data Type | Description |
|---|---|---|
| `Severity` | `VT_-I4` | The severity of the alarm or event. Stored as an integer value with a range of 1–1000. |
| `Actor` | `VT_-BSTR` | The operator who acknowledged the alarm, or caused the tracking event. |
| `Quality` | `VT_-I4` | The quality of the alarm or event. |

**Table 256. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |

### Query Alarm or Event

The following table contains the query parameters for alarms and events data.

| | |
|---|---|
| METHOD | POST |
| URI | `https://<historianservername>:443/historian-rest-api/v1/alarms/query` |
| SAMPLE URI | `https://<historianservername>:443/historian-rest-api/v1/alarms/query`<br>Payload:<br>`{`<br>`"AlarmType":"4",`<br>`"StartTime":"2022-10-30T3:30:00.000Z",`<br>`"EndTime":"2022-10-31T4:30:00.000Z",`<br>`"DataSourceName":"GF7DQL63E_Simulation",`<br>`"MaxRecords":50`<br>`}`<br>Response: |

| | |
|---|---|
| | {"ErrorCode":0,"ErrorMessage":null,"Data":[{"Description":"This is a simulated alarm.","Source":"Simulation00001_ALM","Start Time":"2022-10-30T03:30:00.000Z","End Time":"2022-10-30T03:30:02.000Z","Condition":"Simulated","Sub Condition":"OK","Event Category":"SimCategory","Severity":0,"Modified Time":"2022-10-30T03:30:02.000Z","Quality":3}]} |
| SAMPLE CURL COMMAND | curl -i -H "Accept: application/json" -i -H "Content-Type: application/json" -H "Authorization: Bearer ” -d “{<br>\”AlarmType\”:\”4\”,<br>\”StartTime\”:\”2022-10-30T3:30:00.000Z\”,<br>\”EndTime\”:\”2022-10-31T4:30:00.000Z\”,<br>\”DataSourceName\”:\”GF7DQL63E_Simulation\”,<br>\”MaxRecords\”:50<br>}<br>” -X PUT https://:443/historian-rest-api/v1/alarms/ query |

**Table 257. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| `DataSourceName` | The collector interface name associated with the alarm or event. | `false` | |
| `AlarmType` | Classifies this AlarmInfo as an alarm or an event. Enter 1 for an event and 4 for an alarm. | `yes` | |
| `StartTime` | The start time or time stamp of the alarm or event. | `yes` | |
| `EndTime` | The end time of the alarm. This does not apply to event data. | `yes` | |
| `MaxRecords` | Maximum records returned | `false` | |

**Table 258. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, ErrorCode = 0 implies the operation was successful. |
| `ErrorMessage` | String | Yes | For example, NULL. |

### Delete Alarms or Events

This REST API allows to purge the alarms between time frame.

| | |
|---|---|
| METHOD | POST |
| URI | `https://<historianservername>:443/historian-rest-api/v1/alarms/delete` |
| SAMPLE URI | `https://<historianservername>:443/historian-rest-api/v1/alarms/delete`<br>Payload<br>`https://<historianservername>:443/historian-rest-api/v1/alarms/delete`<br>`Payload:`<br>`{`<br>`  "StartTime":"2022-09-21T5:58:00.000Z",`<br>`  "EndTime":"2022-09-21T6:58:00.000Z"`<br>`}` |
| SAMPLE CURL COMMAND | `curl -i -H "Accept: application/json" -i -H "Content-Type: application/json" -H "Authorization: Bearer ” -d “{`<br>`\"StartTime\":\"2022-09-21T5:58:00.000Z\",`<br>`  \"EndTime\":\"2022-09-21T6:58:00.000Z\"`<br>`}” -X PUT https://:443/historian-rest-api/v1/alarms/delete` |

**Table 259. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| `StartTime` | The start time or time stamp of the alarm or event. | Yes | |
| `EndTime` | The end time of the alarm or event. | Yes | |

**Table 260. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, -26 A&E Server not connected |
| `ErrorMessage` | String | Yes | For example, NULL. |

### Backup Alarms or Events

This REST API allows to create a copy of the alarm data in an offline file that can be restored later. The alarms are not removed from the online system.

| | |
|---|---|
| METHOD | POST |
| URI | `https://<historianservername>:443/historian-rest-api/v1/alarms/backup` |
| SAMPLE URI | `https://<historianservername>:443/historian-rest-api/v1/alarms/backup`<br>Payload<br>`{ "BackupFileName":"C:\\Proficy Historian Data\\Archives\\abc.zip, "StartTime":"2022-09-21T5:58:00.000Z", "EndTime":"2022-09-21T6:58:00.000Z" }` |

| | |
|---|---|
| SAMPLE CURL COMMAND | `curl -i -H "Accept: application/json" -i -H "Content-Type: application/json" -H "Authorization: Bearer ” -d “{ \"BackupFileName\":\"C:\\Proficy Historian Data\\Archives\\abc.zip\", \"StartTime\":\"2022-09-21T5:58:00.000Z\", \"EndTime\":\"2022-09-21T6:58:00.000Z\" }” -X PUT https://:443/historian-rest-api/v1/alarms/ backup` |

**Table 261. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| `BackupFileName` | Path of the backup file. | Yes | C:\\Proficy Historian Data\\Archives\\abc.zip |
| `StartTime` | The start time or time stamp of the alarm or event. | Yes | |
| `EndTime` | The end time of the alarm or event. | Yes | |

**Table 262. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, -26 A&E Server not connected. |
| `ErrorMessage` | String | Yes | For example, NULL. |

### Restoring Alarms or Events

Restoring alarms to a running system makes them available for query and analysis. You can restore alarms that have been backed up previously..

| | |
|---|---|
| METHOD | POST |

| | |
|---|---|
| URI | `https://<historianservername>:443/historian-rest-api/v1/alarms/restore` |
| SAMPLE URI | `https://<historianservername>:443/historian-rest-api/v1/alarms/restore`<br>Payload:<br>`C:\\Proficy Historian Data\\Archives\\abc.zip` |
| SAMPLE cURL COMMAND | `curl -i -H "Accept: application/json" -i -H "Con tent-Type: application/json" -H "Authorization: Bearer ” -d “C:\\Proficy Historian Data\\Archives\\abc.zip” -X PUT https://:443/historian-rest-api/v1/ alarms/ restore` |

**Table 263. Query Parameters**

| Parameter | Description | Required? | Values |
|---|---|---|---|
| `C:\\Proficy Historian Data\\Archives\\abc.zip` | Path of the restore file. | Yes | C:\\Proficy Historian Data\\Archives\\abc.zip |

**Table 264. Response Parameters**

| Parameter | Data Type | Required? | Description |
|---|---|---|---|
| `ErrorCode` | Integer | Yes | For example, 10 invalid parameter. |
| `ErrorMessage` | String | Yes | For example, NULL. |

# Swagger Documentation

You can now access Historian REST APIs using Swagger UI. This tool enables you to visualize and interact with the API’s resources without having any of the implementation logic in place establishing a fully interactive documentation experience using Swagger.

Install web-based clients or ensure web-based clients are installed.

This topic describes how to access the Historian REST APIs.

1. Access a node on which Historian web-based clients are installed.
2. Access the following URL: `https://<web-based client machine name>/historian-rest-api/swagger-ui.html`

   **Note:**
   To access the link, you must provide the machine name where web-based clients are installed. `https://<machinename>/historian-rest-api/swagger-ui.html`

   The Swagger UI appears.
3. Select **Authorize**. The Available authorizations window appears.
4. In the Available authorizations window, scroll down to the **oauth2schema** (OAuth2, password) section, enter the following values, and then select **Authorize**:

| Field | Description |
|---|---|
| username | Enter the web-based client login username. Default <machinename>.admin |
| password | Enter the web-based client login password. |
| client_id | Enter client id: **historian_public_rest_api** |
| client_secret | Enter **publicapisecret** as client secret. |

5. You can now access the REST APIs for Historian.

# Chapter 9. Historian System API

## Overview of the Historian System API

### Overview of the Historian System API

This document is intended to help in the development of C and C++ programs that interact with Historian at the System API level. With the System API you can develop programs that perform the following functions:

- Create, browse, and modify tags of any data type including User Defined Data Types and array data types.
- Read data as raw samples or using any sampling or calculation mode or filter condition or query modifier.
- Write, delete, and replace data samples.
- Get and Set server properties such as product version, archive size, or data store configuration.

Historian has different APIs based on the programming languages. Both the System API and the User API can be called using C or C++. The System API is used only when the User API does not meet your needs. The User API is more widely used and has greater version compatibility.

System API programs are fully capable but they are subject to Proficy Historian security so you will need to have an understanding of security within Historian.

Message and Alarm functions are not documented. However, for more information you can contact the Technical Support team at GE Intelligent Platforms.

The information that you need to write a collector is not included in the System API, but you can create programs that write data. Refer to the Collector Toolkit documentation if you need to write a collector.

#### Prerequisites

Programs created with the Historian System API must be run on a computer with at least the following software installed and configured:

- Historian 5.5 Client Tools
- Visual Studio .NET 2010

The Historian Client Tools are installed when the product is installed. You will be able to install the updates and bug fixes along with the product installation. Do not distribute Historian DLLs directly.

You will need a Data Archiver running locally or on another machine. That Data Archiver can be of any version of Historian. The Historian System API has been tested with Visual Studio .NET 2010. The Historian System API has no additional hardware requirements.

**Note:**
The Historian version where your program is running should be the same as the version of the program you developed.

# ihapi.h File Overview

## ihapi.h File Overview

All technical information for the System API is included in a single `ihapi.h` header file. This document highlights the areas of `ihapi.h` that might affect your programs.

### System API Functions and Data Structures

```
#pragma pack(push,BeforeihAPI)

#pragma pack(1)
```

The System API uses 1 byte packing for structures:

```
// Setup a link dependency on the ihAPI lib

#define ihAPILIB_NAME "ihAPI55.lib"

#define ihAPIDLL_NAME L"ihAPI55.dll"

#define ihAPIDLL_VERSION 550
```

The version specified in the given example changes for each version of Historian. The version of your program should match the version of Historian installed.

```
#ifdef _WCHAR_T_DEFINED

typedef ihPUBLIC wchar_t ihChar;

#else

typedef ihPUBLIC unsigned short ihChar;

#endif
```

The System API uses Unicode. Your program should use `ihChar` for all strings and you can find the correct data type from `ihapi.h file.`

```
typedef struct ihTimeStruct {

MSO_ULONG Seconds;
```

```
MSO_ULONG Nanoseconds;

} ihTimeStruct; // (Must match MSOTimeStruct)
```

Timestamps are nanosecond resolution in the System API. However, only microseconds are exposed to users.

```
// Error Statuses typedef enum

ihStatus { ihSTATUS_OK = 0,

 ...

} ihStatus;
```

There is a fixed set of error codes and you cannot add new ones. Your program can receive errors on reads or writes or tag adds and those errors will be described with the function documentation.

```
 typedef enum ihQualityStatus {

ihOPCBad = 0,

...

} ihQualityStatus;

typedef enum ihQualitySubStatus {

ihOPCNonspecific = 0,

...

} ihQualitySubStatus;
```

Historian uses a fixed set of quality and subquality data and you cannot add new ones.

```
 typedef enum ihDataType {

ihDataTypeUndefined = 0,

... } ihDataType;
```

There are a fixed set of data types and user-defined data types. You cannot add new native data types.

```
 typedef enum ihSamplingMode {

 ihSamplingModeUndefined=0,

...

} ihSamplingMode;

typedef enum ihCalculationMode {

ihCalculationModeUndefined=0,

...

} ihCalculationMode;
```

To read data you should specify a sampling mode, calculation mode, a filter condition (optional), and a query modifier (optional).

```
typedef ihHIDDEN struct ihBlobData {
ihVoidPtr Blob;
MSO_ULONG BlobSize;
} ihBlobData;
```

The blob data type is a size and a pointer in the System API

```
typedef struct ihTagProperties {
ihString Tagname;
...
} ihTagProperties;
```

A tag has a fixed set of properties as listed in this structure. Some tag properties may not exist in the earlier versions of Historian and new properties may be added in the future. You are limited to the set of properties available in the version of the ihapi.h shipped with this SDK and you cannot add new tag properties.

```
 typedef struct ihDataProperties {
 ihTimeStruct TimeStamp;
 ihDataType ValueDataType;
 ihValue Value;
 ihQuality Quality;
 unsigned char NumberOfComments;
 ihCommentsPtr Comments;
ihGeoLocation Unsupported;
} ihDataProperties;
```

The given prototype is the structure for one raw data sample and it has timestamp, data type, value, and quality. It can optionally have comments. This structure is used on both data reads and data writes.

```
typedef struct ihDataRecordset {
 ihDataFields Fields;
 ihDataCriteria Criteria;
 ihUNSIGNED long NumberOfResults; // (Num items in results)
 ihDataResultPtr Results; // (array. One for each matching tag)
} ihDataRecordset;
```

The given prototype is a collection of data samples that you get back from a data read or subscription.

### Callback Prototypes/typedefs

```
typedef void (__stdcall *ihInterfaceGetCurrentValueCallbackFunction) (ihServerHandle hServer, void *UserParameter,
ihUNSIGNED long NumberOfSourceAddresses,
ihString *SourceAddresses, ihCallbackId CallbackId);
```

A lot of information can be communicated to your program using callback if you subscribe to changes which is optional. The following sections describe this in more detail.

## ihConfiguration Functions

```
extern ihC_DEC ihAPIStatus __stdcall ihConfigurationGetProperties
(ihServerHandle hServer, ihConfigurationProperties *Properties);
extern ihC_DEC void __stdcall ihConfigurationFreeProperties
(ihConfigurationProperties *Properties);
```

You can get information about the Data Archiver in addition to reading and writing data.

### ihServer Function

```
extern ihC_DEC ihAPIStatus  stdcall ihServerConnect(ihString ServerName, ihString Username, ihString Password,
ihString BufferFileName, ihServerHandle *hServer);
extern ihC_DEC ihAPIStatus  stdcall ihServerConnectClient(ihString ServerName, ihString Username, ihString
Password, ihString BufferFileName, ihServerHandle *hServer, ihString ClientName);
extern ihC_DEC ihAPIStatus  stdcall ihServerDisconnect(ihServerHandle hServer);
extern ihC_DEC ihBoolean stdcall ihServerIsConnected(ihServerHandle hServer);
```

You need to connect to the server before you start working.

```
extern ihC_DEC ihAPIStatus  stdcall ihServerAdd(ihString ServerName, ihString Username, ihString Password,
ihBoolean IsDefault, ihString BufferFileName, ihServerHandle *hServer,ihULong ConnectionTimeout);
extern ihC_DEC ihAPIStatus  stdcall ihServerDelete(ihString ServerName);
extern ihC_DEC ihAPIStatus  stdcall ihServerOpenRecordset(ihString ServerNameMask, ihServerRecordset
*ServerRecordset);
extern ihC_DEC void  stdcall ihServerCloseRecordset(ihServerRecordset *ServerRecordset);
```

You can set up a collection of servers with their timeouts and usernames. However, you can simply call connect to establish a connection.

## ihTag Functions

```
extern ihC_DEC ihAPIStatus  stdcall ihTagAdd(ihServerHandle hServer, ihTagFields *hFields, ihTagProperties
*hTag);
extern ihC_DEC ihAPIStatus  stdcall ihTagOpenRecordset(ihServerHandle hServer, ihTagFields *RequestedFields,
ihTagCriteria *Criteria, ihTagFields *CriteriaFields, ihTagRecordset *TagRecordset);
extern ihC_DEC ihAPIStatus  stdcall ihTagGetProperties(ihServerHandle hServer, ihString Tagname, ihTagFields
*hFields, ihTagProperties *hTag);
```

You can add and browse tags using these functions

## ihData Functions

```
extern ihC_DEC ihAPIStatus  stdcall ihDataAdd(ihServerHandle hServer, ihUNSIGNED long NumberOfTags, ihString
*Tagnames, ihDataProperties *DataValues, ihAPIStatus *ErrorStatuses, ihBoolean WaitForReply, ihBoolean ErrorOnRep
```

You can add and delete data and subscribe to data changes

## ihDataStore Functions

```
extern ihC_DEC ihAPIStatus  stdcall ihDataStoreOpenRecordset(ihServerHandle hServer, ihString
DataStoreMask, ihDataStoreRecordset *Recordset);
```

You need not configure data stores, the defaults should be fine.

## ihComment Functions

```
extern ihC_DEC ihAPIStatus  stdcall ihCommentAdd (ihServerHandle hServer, ihString Tagname, ihTimeStruct
*CommentAdd, ihCommentData *ihCommentData, ihString SuppliedUser, ihString SuppliedPassword);
```

You can add comments to data and get them back when you read the data but this is not commonly used.

## ihTime Functions

```
extern ihC_DEC void stdcall ihTimeLCLPartsToUTCStruct(int Year, int Month, int Day, int Hour, int Minute,
int MilliSecond, ihTimeStruct *UTCTime);
extern ihC_DEC void stdcall ihTimeUTCStructToLCLParts(int *Year, int *Month, int *Day, int *Hour, int
*Minute, int *Second, int *MilliSecond, ihTimeStruct *UTCTime);
```

The System API has utility functions for working with time zones and daylight saving time.

### ihUtil Functions

```
extern ihC_DEC void stdcall ihUtilAnsiToUnicode(char *MBStr, ihChar *WCStr);

extern ihC_DEC void stdcall ihUtilUnicodeToAnsi(char *MBStr, ihChar *WCStr);
```

The System API has utility functions for converting between Unicode and ANSI strings.

# System API Programming

## System API Programming

To use System API you should be an experienced programmer. The section that follow list general topics that will help you understand the best use of the System API.

### Unicode

String tag names, string tag properties such as description, and string data values are all Unicode strings.

### Memory

You should never free memory allocated in the API. The API provides many functions to free memory or clear record sets and structures and you should use them as demonstrated in sample programs.

### String Length

In the API you will notice that most string fields such as tag names or descriptions are just pointers and this is because they do not have a maximum length. They must be allocated and freed.

### Networking between the System API and the Data Archiver

The communication between the `ihapi` DLL and the Data Archiver is TCP/IP networking which sends proprietary packet streams via Winsock calls. The details are not documented.

### Multithread Programming

You can have multiple threads reading and writing data, even sharing the same connection.

### Multitag Functions

In most cases you can simply act on one tag at a time when reading data or adding tags. These calls are less frequent than writing data. But there are API calls that can handle multiple tags in one round trip and your design should decide if they are needed for performance reasons.

### Timestamps

You will see that the timestamp structure is in seconds and subseconds. It is implied that the time zone of the seconds is in GMT+0 Universal Time Coordinated time zone. The System API gives utility functions to convert timestamps to and from local time zones.

### Running as a Service

You can develop programs that run as GUI programs, console programs, or services.

### Timeouts and Throttles in the System API and the Data Archiver

The Data Archiver will try to protect the system against unintentional large data queries. There are Data Archiver enforced, configurable throttles for queries that return too many samples or take too long to execute. Use the i`hArchiverMaxIntervalRetrievalCount` and `ihArchiverMaxQueryTime` archiver options to increase the limits if you occasionally get the `ihSTATUS_DATA_RETRIEVAL_COUNT_EXCEEDED` error from intentional large queries.

On the System API side there is a configurable timeout set via `ihServerSetTimeout()` and defaults to 90 seconds. Data writes and setting of options must complete within this time period or the API will return an error to the application. The Data Archiver may have queued the work and will still perform the request even if a timeout is returned to the application.

Reads and tag browses can take longer than 90 seconds and this is because the API needs to receive only a partial response every 90 seconds. Large reads and tag browses are streamed back to the API in pieces as the read or browse is still being performed in the Data Archiver. If we do not receive at least one piece every 90 seconds, a timeout error is returned to application but the Data Archiver will continue working until it hits the limit in `ihArchiverMaxQueryTime` archiver option.

# System API Functions

## System API Connect Functions

This section provides detailed information about each available System API function. Use this information together with the sample programs provided to develop your own applications. The System API functions are grouped as follows:


- Connect Functions *(on page 1437)*
- System API Tag Functions *(on page 1443)*
- Read and Write Functions *(on page 1455)*
- Archiver Configuration Functions *(on page 1469)*

- User Defined Type Functions *(on page 1488)*
- Utility Factors *(on page 1495)*

Not every function and structure in the `ihapi.h` is documented or available for use in user written programs. Please limit your programs to the documented functions or contact the Technical Support team at GE Intelligent Platforms for additional clarifications.

## Connect Functions

To perform a connection you need to just use the connect functions. The API will complete the connection and do further necessary reconnects when the Data Archiver is restarted. Your program does not need to manage and monitor connections as it is done by the API.

Connections are subjected to Historian security, which is based on Microsoft Windows Security.

It is expected that user written programs will connect a minimal number of times for staying connected and using that connection for all read and write calls. Connecting and disconnecting rapidly is not efficient. The connect functions are:

## ihServerConnect

This function establishes the connection with the Historian server.

```
ihServerConnect(


ihString ServerName,// the computer name or IP address of the Machine running the Data Archiver. You can pass
NULL or the empty string &amp;amp;rdquor;” to attempt connection to a Data Archiver on your local machine.
ihString Username,// The windows username to use when connecting or NULL if you want to connect as the process
owner of your program, typically the logged in user.  You would specify a username if the process owner was unable to
connect or did not have sufficient Historian Security permissions.
ihString Password,// if you passed a Username you can pass the password or &rdquor;” if there is no password or
otherwise just pass NULL
ihString BufferFileName, // an optional bufferfilename if your program will be using store and forward to
deliver written data
ihServerHandle *hServer // this is an output parameter that will contain the server handle that you would use in
later read and write calls
);
```

**Remarks**

A server is a computer running a Data Archiver. You can specify a server by including the computer name or IP address.

`ihServerConnect()` will initiate a connection, but you should really use `ihServerIsConnected()` *(on page 1439)* to determine if the connect actually completed.

If you setup a connection callback function with `ihServerRegisterConnectionCallback()` you will be notified of connection changes. Your security permissions for reads and writes will be established at connect time and are based on the user name and password provided.

**Returns**

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | On success. |
| `ihSTATUS_API_TIMEOUT` | If the server name or IP address provided cannot be located. |
| `ihSTATUS_LIC_TOO_MANY_USERS` | If this connection would exceed your licensed number of connections. |
| `ihSTATUS_NOT_VALID_USER` | If the user is not allowed to connect. This can happen whether you provided a username or connected as the process owner. Consult the Historian documentation for security behavior. |
| `ihSTATUS_FAILED` | For any other type of error. |

## ihServerDisconnect

Use this function to disconnect from the Historian server.

**Prototype**

```
ihSeverDisconnect {

ihServerHandle hServer // the handle returned by a previous call to ihServerConnect


);
```

**Remarks**

You should call `ihServerDisconnect()` function even if the connection attempt returned an error.

**Returns**

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | On success. |

## ihServerIsConnected

This function returns whether the Historian server is currently connected or not. It is an indication that if you try to read and write data the call should be successful.

### Prototype

```
ihServerIsConnected {

ihServerHandle hServer // a server handle returned from ihServerConnect


};
```

### Returns

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | On success. |
| `ihSTATUS_FAILED` | On failure. |

## ihServerSetTimeout

This function is used to configure the timeout of the server connection.

### Prototype

```
TihServerSetTimeout {

ihServerHandle hServer, // a server handle returned from ihServerConnect

int Timeout // the timeout, in seconds, to use

};
```

Use this to configure the timeout to a larger or smaller value. If the System API does not receive any response, even a partial response, during the timeout value, a `ihSTATUS_API_TIMEOUT` will be returned from the API call.

Set the timeout any time after connecting and the timeout stays in effect until it is changed or disconnect is issued.

### Returns

| Returns Status | Message |
|---|---|
| `TRUE` | if you are currently successfully connected. |
| `FALSE` | if you were never connected or if the connection is currently down and being re-established. |

## ihServerGetTimeout

This function is used to return the messaging timeout value of the server connection identified by ihServer.

```
ihServerGetTimeout {

ihServerHandle hServer, ihULong *timeout

};
```

### Remarks

Call `ihServerConnect()` to make a connection, then `ihServerSetTimeout()` to set the timeout if you want a non default value. You can use the `ihServerGetTimeout()` function to determine the timeout value that is being used.

The returned timeout value is in seconds.

### Returns

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | On success |
| `ihSTATUS_FAILED` | On failure |

## ihServerGetVersion

This function returns the Historian server version and connection status.

### Prototype

```
ihServerGetVersion {

ihServerHandle hServer, // a serverhandle returned from a previous call to ihServerConnect

int *Major, // output parameter to contain the major version of the Data Archiver.

For example 4 when the Data Archiver version is 4.5.

int *Minor, // output parameter to contain the minor version of the Data Archiver.

For example 5 when the Data Archiver version is 4.5.
```

```
int *Build, // output parameter to contain the build number of the Data Archiver. This is typically not used.

int *Revision // output parameter to contain the revision number of the Data Archiver. This is typically not used


};

You will not receive an immediate callback with the current connection status. You will be notified of the next change
 in status
```

**Remarks**

You will not receive an immediate callback with the current connection status. You will be notified of the next change in status.

**Returns**

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | On success and the desired information can be read from the output parameters. |
| `ihSTATUS_FAILED` | On failure. |

## ihSecurityGetMembership

This function returns your membership list. Your group memberships determine your security permissions and what calls you can make on this Proficy Historian server connection.

**Prototype**

```
ihSecurityGetMembership {

ihServerHandle hServer, // handle from previous call to ihServerConnect

ihSecurityGrpMembership *Memberships // output parameter to contain the group membership list


};
```

**Remarks**

You can optionally use this call after a successful connect to determine whether the Data Archiver considers your membership list. You need to be completely connected, not just initiate a connection, before this call can be used. The best way is to use `ihServerIsConnected()` *(on page 1439)* after `ihServerConnect()` *(on page 1437)* to determine that the connection completed.

**Returns**

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | On success and the desired information can be read from the output parameter. |
| `ihSTATUS_API_TIMEOUT` | If the call could not be completed. |
| `ihSTATUS_NOT_CONNECTED` | When not currently connected to the Data Archiver. |
| `ihSTATUS_FAILED` | When the other three return options otherwise do not apply. |

## ihServerOpenRecordset

This function returns the list of Historian servers that exist in the Windows registry.

### Prototype

```
ihServerOpenRecordset {

ihString ServerNameMask, // typically NULL to get all servers but could be a mask


ihServerRecordset *ServerRecordset // output parameter containing the recordset


};
```

### Remarks

Use this function to get the list of servers that exist in the Registry. This is a local call and does not need the Data Archiver.

The default server is stored in the Registry and can be used to query instead of hardcoding a server name in your program or prompting the user. The default server is established during product install and can be changed anytime later

### Returns

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | On success and the desired information can be read from the output parameter. |
| `ihSTATUS_FAILED` | On failure |

### ihServerCloseRecordset

This function is used to free any memory such as server names that were allocated in the System API.

#### Prototype

```
void ihServerCloseRecordset {

ihServerRecordset *ServerRecordset // recordset returned from ihServerOpenRecordset

};
```

Use this function to free any memory such as server names that were allocated in the System API in the `ihServerOpenRecordset()` call.

#### Returns

void

## System API Tag Functions

You need tags to exist in Historian in order to write and read data samples. A read or write call requires a tagname as a parameter. The tag must exist before the data samples are written, the tag is not created automatically.

You can refer to a tag in the following ways:

- A tag has a `tagname`.
- A tag may have aliases which are previously used names left over from tag renames.
- A tag has a `tagid` which is a long number and does not change. If you convert this number to a string you can pass it in place of a `tagname`. If your tagnames are typically longer than 128 characters then using the `tagid` would be shorter.

A tagname can contain any character except the two wild card characters * and ?.

There are many tag properties as listed in the `ihTagProperties` data structures in the `ihapi.h`. All properties are available to your application and are documented in the Historian product documentation. A subset of import- ant properties are given in the following table. This list that applies to tags being written to by user programs. All of the properties related to collectors are not documented in the following table.

### ihTagProperties

| Property | Type | Description |
|---|---|---|
| Tagname | ihString | The tag name that shows in the tag browse. |
| Description | ihString | The description of the tag that shows in the tag browse. |
| DataType | ihData-Type | The data type of the tag. This can be a custom type also. |
| FixedStringLength | Un-signed-Char | Used only if data type is FixedString. |
| HiEngineeringU-nits | Double | Used only if you are using ArchiveDead- bandPercent. |
| LoEngineeringU-nits | Double | Used only if you are using ArchiveDead- bandPercent. |
| ArchiveCompres-sion | ih-Boolean | TRUE if you are using percent or absolute archive. |
| ArchiveDeadband-PercentRange | Float | It is one way to configure the deadband instead of absolute archive. |
| InterfaceGeneral1 | ihString | Spare string field for application use. |
| InterfaceGeneral2 | ihString | Spare string field for application use. |
| InterfaceGeneral3 | ihString | Spare string field for application use. |
| InterfaceGeneral4 | ihString | Spare string field for application use. |
| InterfaceGeneral5 | ihString | Spare string field for application use. |
| ReadSecurity-Group | ihString | Only needed if you are using tag level security. |
| WriteSecurity-Group | ihString | Only needed if you are using tag level security. |
| AdministratorSe-curityGroup | ihString | Only needed if you are using tag level security. |
| UTCBias | Long | You can store an offset from GMT+0 here, such as -300. But it is up to the client programs to use it. |

| Property | Type | Description |
|---|---|---|
| ArchiveCompressionTimeOut | ihUnsignedLong | -- |
| ArchiveAbsoluteDeadbanding | ihBoolean | -- |
| ArchiveAbsoluteDeadband | Double | -- |
| TimeResolution | ihTimeResolution | If the tag is using seconds, milliseconds or microseconds. |
| TagId | ihTagId | You can refer to tags by their tagname or their TagId |
| EnumeratedSetName | ihString | If you want to retrieve using an Enumerated Set. This is optional. |
| DataStoreName | ihString | Defines what data store the tag belongs to or should be added to during an write function. Blank means default data store. |
| DefaultQueryModifiers | ihQueryModifiers | Query modifier string. This is optional. |
| UserDefinedTypeName | ihString | If you are using custom data types. |
| NumberOfElements | ihUnsignedlong | If you are using an array data type. |
| DataDensity | ihTagDataDensity | -- |

## Tag Functions

## ihTagAdd

Use this function to add a tag to Historian.

**Prototype**

```
ihTagAdd {
ihServerHandle hServer, // handle from previous call to ihServerConnect
ihTagFields *hFields, // the tag fields you are providing in the ihTagProperties
ihTagProperties *hTag // tag property values such as tagname and data type
};
```

**Remarks**

There are many tag properties but the minimum set is that you need tagname and data type. Since you do not need to provide all properties, use the ihTagFields to indicate which properties you are providing in the ihTagProperties

Adding a tag that already exists will update that tag to the properties that you provide.

For examples, refer to the Sample Programs included with the SDK.

**Returns**

| Returns | Description |
|---|---|
| `ihSTATUS_OK` | On success |
| `ihSTATUS_API_TIMEOUT` | If the server name or IP address provided cannot be located. |
| `ihSTATUS_NOT_CONNECTED` | If you are not currently connected. |
| `ihSTATUS_ACCESS_DENIED` | If you are not allowed to add or modify tags. Possibly you are not a member of the “ih Tag Admins” group. |
| `ihSTATUS_LIC_TOO_MANY_TAGS` | If adding this tag would exceed your licensed tag count `ihSTATUS_FAILED` - for any other type of error. |
| `ihSTATUS_FAILED` | For any other type of error. |

## ihTagMultiAdd

Use this function to add more than one tag in a single call to Proficy Historian.

**Prototype**

```
ihTagMultiAdd {
ihServerHandle hServer,
ihTagFields *hFields,
```

```
ihTagProperties *hTag,

int NumTags

};
```

This function takes an array of tag structures and a count. Typically, you call ihTagAdd function to add or modify your tags. For more information on adding a single tag, refer to `ihTagAdd()` function. If you have thousands of tags to add use `ihTagMultiAdd()` function.

**Returns**

| Returns | Description |
|---|---|
| `ihSTATUS_OK` | On success. |
| `ihSTATUS_API_TIMEOUT` | If the server name or IP address provided cannot be located. |
| `ihSTATUS_NOT_CONNECTED` | If you are not currently connected. |
| `ihSTATUS_ACCESS_DENIED` | If you are not allowed to add or modify tags. Possibly you are not a member of the “ih Tag Admins” group. |
| `ihSTATUS_LIC_TOO_MANY_TAGS` | If adding this tag would exceed your licensed tag count. |
| `ihSTATUS_FAILED` | For any other type of error. |

## ihTagDelete

Use this function to delete a tag from Historian.

**Prototype**

```
ihTagDelete {

ihServerHandle hServer, // a server handle returned from ihServerConnect

ihString Tagname, // the tag to delete

ihBoolean DeletePermanent // FALSE if you only want to make the tag not appear in tag browse.

TRUE if you want to delete the tag and its data entirely.

};
```

**Remarks**

You can delete a tag if it was created by mistake or if you no longer need the data. In most cases, passing DeletePermanent=FALSE is sufficient and hides the tag from future tag lists. Only if you need to re-use the tagname should you pass DeletePermanent=TRUE.

**Returns**

| Returns | Description |
|---|---|
| `ihSTATUS_OK` | On success. |
| `ihSTATUS_API_TIMEOUT` | If the server name or IP address provided cannot be located. |
| `ihSTATUS_NOT_CONNECTED` | If you are not currently connected. |
| `ihSTATUS_ACCESS_DENIED` | If you are not allowed to add or modify tags. Possibly you are not a member of the “ih Tag Admins” group. |
| `ihSTATUS_FAILED` | For any other type of error. |

## ihTagRename

Use this function to rename a tag in Historian.

**Prototype**

```
ihTagRename {
ihServerHandle hServer, // a server handle returned from ihServerConnect
ihString Tagname, // the current tagname
ihString NewTagname, // the new tagname
ihBoolean TrueRename // TRUE if you want to reject reads to the old tagname and want to make the
old tagname available for reuse.
};
```

**Remarks**

You can rename tags to a single, new name or keep the old name available to legacy clients. Only the current name appears in a tag browse. Pass TRUE to permanently rename a tag.

**Returns**

| Returns | Description |
|---|---|
| `ihSTATUS_OK` | On success. |
| `ihSTATUS_API_TIMEOUT` | If the server name or IP address provided cannot be located. |
| `ihSTATUS_NOT_CONNECTED` | If you are not currently connected. |
| `ihSTATUS_ACCESS_DENIED` | If you are not allowed to add or modify tags. Possibly you are not a member of the “ih Tag Admins” group. |
| `ihSTATUS_FAILED` | For any other type of error. |

## ihTagFreeproperties (Tag Properties)

Use this function to correctly free one tag properties structure that was returned from an `ihTagGetProperties()` call.

### Prototype

```
void ihTagFreeProperties {
ihTagFields *hFields, // indicates which fields in the tag properties are valid
ihTagProperties *hTagProps // structure containing tag properties to be freed.
};
```

Use this function to correctly free one tag properties structure. You do not need to call this to free the tag recordset returned from a browse. Use `ihTagCloseRecordset()` *(on page 1443)* instead.

### Returns

| Returns | Description |
|---|---|
| Void | Void |

## ihTagOpenRecordset

This function is used to browse tags that exist in the Data Archiver.

```
ihTagOpenRecordset {
ihServerHandle hServer, // a server handle returned from ihServerConnect
ihTagFields *RequestedFields, // indicates which tag properties you want included with the returned tags
ihTagCriteria *Criteria, // used to determine which tags to return
```

```
ihTagFields *CriteriaFields, // used to indicate which criteria fields should be used

ihTagRecordset *TagRecordset  // output parameter that contains the returned tag recordset


};
```

**Remarks**

This is the function used to browse tags that exist in the Data Archiver. You specify a criteria and which fields of each matching tag you want returned and a recordset of matching tags is returned. Since a tag has many properties it is a waste of resources to return all the tag properties if for example you only want the tagnames.

Using filtering criteria you can perform a simple wildcard search on tagname or description, or you can get all tags of integer data type, or otherwise use any condition on any tag property. The CriteriaFields is where you indicate which tag properties to use in the filter.

The tag recordset will contain a number of tags returned. It is possible that no tags match your criteria. After you are finished using the tag recordset you should free it using `ihTagCloseRecordset().` Do not free any individual tag string fields.

An example of tag filtering and browsing is provided in SDK sample program.

**Returns**

| Returns | Description |
|---|---|
| `ihSTATUS_OK` | On success and the tags are returned in the Tag-Recordset. |
| `ihSTATUS_API_TIMEOUT` | The client program did not receive a portion of the tag list before the timeout expired. Large browses are sent back in multiple responses and the SystemAPI must receive at least a partial response before the timeout expires. For example if there is 90 second timeout configured, a browse can take several minutes or as long as at least a part of the response is received every 90 seconds. |
| `ihSTATUS_NOT_CONNECTED` | If you are not currently connected to the Data Archiver. |

| Returns | Description |
|---|---|
| `ihSTATUS_ACCESS_DENIED` | If you are not allowed to add or modify tags. Possibly you are not a member of the “ih Tag Admins” group. |
| `ihSTATUS_FAILED` | For any other type of error. |

## ihTagCloseRecordset

Use this function to return the recordset returned from an ihTagOpenRecordset() call.

### Prototype

```
ihTagCloseRecordset {

ihTagRecordset *TagRecordset

};
```

### Remarks

Use this function to return the recordset returned from an `ihTagOpenRecordset()` call.

### Returns

| Returns | Description |
|---|---|
| `ihSTATUS_OK` | Always |

## ihTagExists

This function checks if a given tag exists in Historian.

```
ihTagExists {

ihServerHandle hServer, // a server handle returned from ihServerConnect


ihString Tagname

};
```

### Remarks

Use this function to check if a tag exists as it is easier than using ihTagOpenRecordset(). Tags that have been deleted are usually hidden and they still exist. ihTagOpenRecordset() only returns non-deleted tags.

### Returns

| Returns | Description |
|---|---|
| `ihSTATUS_OK` | If the tag exists. |
| `ihSTATUS_INVALID_PARAMETER` | On invalid input parameters. |

This function is more convenient than using the ihTagOpenRecordset() function when you know the tagname and want to check for tag existence or get more information about a tag.

If the call succeeds and properties are returned you should call `ihTagFreeProperties()` to free any string properties.

## ihTagGetProperties

This function returns the properties of the tags.

```
ihTagGetProperties {

ihServerHandle hServer, // a server handle returned from ihServerConnect

ihString Tagname, // the tagname to locate. Wildcards are not allowed.

ihTagFields *hFields, // the fields you want returned

ihTagProperties *hTag // an output parameter containing the returned tag properties

};
```

**Remarks**

This function is more convenient than using the ihTagOpenRecordset() function when you know the tagname and want to check for tag existence or get more information about a tag.

If the call succeeds and properties are returned you should call `ihTagFreeProperties()` to free any string properties.

**Returns**

| Returns | Description |
|---|---|
| `ihSTATUS_OK` | If the tag exists. |
| `ihSTATUS_INVALID_PARAMETER` | On invalid input parameters. |

## ihTagFreeProperties

This function used to free the information returned by an `ihTagGetProperties()` call.

**Prototype**

```
void ihTagFreeProperties {

ihTagFields *hFields, // the fields you want freed

ihTagProperties *hTag // contains the tag properties

};
```

### Remarks

Use this function to free the information returned by an `ihTagGetProperties()` call.

### Returns

| Returns | Description |
|---|---|
| `Void` | Void |

## ihTagSetProperties

This function is used to set the tag properties.

### Prototype

```
ihTagSetProperties {

ihServerHandle hServer, // a server handle returned from ihServerConnect

ihTagFields *hFields,

ihTagProperties *hTag


};
```

## ihTagSubscribe

This function is used to setup a subscription to tag changes.

### Prototype

```
ihTagSubscribe {

ihServerHandle hServer, // a server handle returned from ihServerConnect

ihString Tagname, // the tagname to monitor for changes. Wildcards are allowed

ihBoolean Subscribe // TRUE if you want to subscribe or FALSE to unsubscribe


};
```

### Remarks

Use this function to setup a subscription to tag changes. A tag change would be a change to a tag property. Data being written to a tag is not a tag change but can be monitored using ihDataSubscribe.

You must register a callback function using ihTagRegisterCallback() and that function will be called on change.

To detect tag additions pass the wildcard * and you will be notified of all changes to all tags. Discard any incoming subscriptions you are not interested in.

An example of tag subscriptions is provided in the SDK sample program.

## ihTagRegisterCallBack

This function returns which tag properties have changed

```
ihTagRegisterCallBack {

ihServerHandle hServer, // a server handle returned from ihServerConnect

ihTagCallbackFunction CallbackFunction, // the function to be called

void *UserParameter


};
```

When your callback function is called there will be an indication of what tag property changed.

## ihTagClearAllFields

This function clears the tag properties.

### Prototype

```
void ihTagClearAllFields {

ihTagFields *Fields


};
```

### Returns

| Returns | Description |
|---|---|
| `Void` | Void |

# Read and Write Functions

Data reading and writing is a key part of Historian functionality. There are many rules and best practices and many use cases that are not all documented here. With the following documentation you will see how to write raw values and read them back as raw values and then how to instruct the Data Archiver to perform calculations on the raw samples and return the result.

The method to pass filter parameters and query modifiers is shown but not the meaning of those parameters. Store and forward buffering is available for data writes but is an advanced topic not demonstrated here.

The Read and Write functions are:

## ihDataAdd

Use this function to add data to the Data Archiver:

### Prototype

```
ihDataAdd {
ihServerHandle hServer, // server handle returned from previous call to ihServerConnect
ihUNSIGNED long NumberOfTags, // the number of data samples being written
ihString *Tagnames, // the tagname of each sample
ihDataProperties *DataValues, // the timestamp, value, and quality of each sample
ihAPIStatus *ErrorStatuses, // output parameter for per data sample errors
ihBoolean WaitForReply, // TRUE if the call should wait for success or failure from Data Archiver.
FALSE means that you do not want to do any error checking
ihBoolean ErrorOnReplace // TRUE if you want an error returned instead of overwriting data
);
```

### Remarks

Although the parameter is called NumberOfTags it is really NumberOfSamples. Even if you are writing multiple samples to the same tagname you need to pass the tagname with each data sample. Every data sample you pass needs a tagname in the Tagnames parameter and a timestamp, value, and quality in the DataValues parameter.

The application would write data timestamps that are in GMT+0 timezone. The Data Archiver does not do any conversion. If your timestamps are in local time and the data is retrieved in other clients, the timestamps will not be adjusted to your time zone. The System API uses the ihTimeLCLPartsToUTCStructEx () function to help you convert timestamps.

Part of your application design is to bundle multiple data samples into one write. You can send one sample per call to ihDataAdd or send 100,000 samples or more. In most cases you can write data periodically to make it available to clients in bundles of 1,000 to 10,000 samples per call to ihDataAdd

Part of your application design is to determine what to consider bad or uncertain quality and what sub-quality to use. Your tags may be in a Data Archiver with many other tags from many other sources, these include tags written by collectors or Excel Import. You must be consistent with those tags so that clients need not have special logic for specific tags. For example, collectors use a quality of bad and a subquality of ihOffline to indicate that collection has stopped at this time in the real world. Consider including such a data sample and using the same quality and subquality in your program. In this way reports and trends need not know where the data came from.

The data type you write in the ihDataProperties does not need to match the data type of the tag in the Data Archiver. The Data Archiver will do the conversion prior to storing the data.

You should typically write with `WaitForReply=TRUE` so that you can check for errors and implement error handling such as retries. If you have no error handling strategy or if you are relying on the store and forward functionality to deliver data you can write with wait = FALSE. This is a rare use case. Understand that store and forward functionality only make sure that the data reaches the Data Archiver, not that the data is stored in the Historian archive. There could be security errors or the Archiver may be out of disk space. The most reliable way to write data is to check for errors and even possibly to read back the data that was written.

If you do get a per data sample error that does not mean the whole write failed. Samples that return ihSTATUS_OK were written successfully.

If your program only writes data going forward in time, there is no risk of replacing data and you can pass either TRUE or FALSE for the ErrorOnReplace parameter. Otherwise, pass TRUE if you do not want accidental data overwrite or FALSE to intentionally overwrite existing data.

There are rules about writes having timestamps that are too old or too new and those are not all given here. But, you will receive an error if your write is rejected.

**Returns**

If you write with WaitForReply=FALSE then the return code will be ihSTATUS_OK. When using WaitForReply=FALSE the return value of the overall function will be ihSTATUS_OK on success and non ihSTATUS_OK on error which means you should investigate the per data sample error returned in the ErrorStatuses output parameter. Those errors include these values:

If you write with WaitForReply=FALSE then the return code will be ihSTATUS_OK. When using WaitForReply=FALSE the return value of the overall function will be ihSTATUS_OK on success and non ihSTATUS_OK on error which means you should investigate the per data sample error returned in the ErrorStatuses output parameter. Those errors include these values:

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | The data sample was successfully written. |
| `ihSTATUS_API_TIMEOUT` | If the client program did not receive a return code before the timeout expired. This does not necessarily mean the write was not performed at the archiver, only that the client did not receive a response. |
| `ihSTATUS_NOT_CONNECTED` | If you are not currently connected to the Data Archiver. |
| `ihSTATUS_WRITE_IN_FUTURE` | If you specified a timestamp that was more than 15 minutes ahead of the Data Archiver PC clock. The Data Archiver will not accept writes that are too far into the future because it has no data file to hold them. |
| `ihSTATUS_DUPLICATE_DATA` | If you set ErrorOnReplace to TRUE and you would have overwritten data. |
| `ihSTATUS_WRITE_ARCH_OFFLINE` | If you wrote to a time period that has no archive. |
| `ihSTATUS_ARCH_READONLY` | If the archive covering the timestamp written is set to read only in Historian Administrator |
| `ihSTATUS_ACCESS_DENIED` | If you are not allowed to write data to this tag and timestamp. Under most conditions, writes from user programs are allowed. But if you receive this error consider the following scenarios:<br>• Your account is not a member of ihAudited-Writers or ihUnaudited Writers group.<br>• The tag you are writing has tag level security enabled.<br>• The entire archive is set to read only in Historian Administrator. |

| Returns Status | Message |
|---|---|
| | The tag you are writing has tag level security enabled. The entire archive is set to read only in Historian Administrator. |
| `ihSTATUS_WRITE_OUTSIDE_ACTIVE` | The timestamp on the data sample is older than the “data is read only" |
| `ihSTATUS_INVALID_TAGNAME` | The tagname you wrote to does not exist in the Data Archiver configuration. |
| `ihSTATUS_FAILED` | For any other type of error. |

## ihDataDelete

This function is used to delete data. By using this function you are not actually deleting the data from the Data Archiver but instead hiding it and marking it as deleted.

### Prototype

```
ihDataDelete {

ihServerHandle hServer, // server handle returned from previous call to ihServerConnect

ihUNSIGNED long NumberOfTags, // the number of data samples included in the Tagnames and DataValues parameters

ihString *Tagnames, // the tagname of the data sample to be deleted

ihDataProperties *DataValues, // the list of data samples to be deleted

ihAPIStatus *ErrorStatuses // output parameter containing error codes for individual data samples


);
```

### Remarks

Deleting data is not actually done and this function does not clear or reset an entire tag. The reason is that the data is only hidden and not truly deleted. If you write and delete the same time range repeatedly the Historian storage becomes very inefficient.

Use the delete function to delete individual data samples so they are not returned in raw data queries or considered in calculation modes. Delete a small range of data if you have recalculated values and want to discard the previous calculations. You do not need to delete data prior to data overwrite.

The return codes for a delete are much like the ones returned from a write. ihSTATUS_OK - the data sample was successfully deleted.

**Returns**

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | The data sample was successfully deleted. |
| `ihSTATUS_API_TIMEOUT` | If the client program did not receive a return code before the timeout expired. This does not necessarily mean the write was not performed at the archiver, only that the client did not receive a response. |
| `ihSTATUS_NOT_CONNECTED` | If you are not currently connected to the Data Archiver |
| `ihSTATUS_WRITE_ARCH_OFFLINE` | If you wrote to a time period that has no archive. |
| `ihSTATUS_ACCESS_DENIED` | If you are not allowed to write data to this tag and timestamp. Under most conditions, writes from user programs are allowed. But if you receive this error consider the following scenarios:<br>• The tag you are writing has tag level security enabled.<br>• The entire archive is set to read only in Historian Administrator.<br>The tag you are writing has tag level security enabled. The entire archive is set to read only in Historian Administrator. |
| `ihSTATUS_WRITE_OUTSIDE_ACTIVE` | The timestamp on the data sample is older than the “data is read only" |
| `ihSTATUS_FAILED` | For any other type of error. |

## ihDataOpenRecordset

This function is used to read values for one or more tags for a single start and end time, and sampling mode.

**Prototype**

```
ihDataOpenRecordset {
ihServerHandle hServer, // server handle returned from previous call to ihServerConnect
ihDataFields *Fields, // which fields such as value and quality and comments you want returned
ihDataCriteria *Criteria, // all of the parameters for the read including the tagname and time range and
sampling mode
ihDataRecordset *DataRecordset // output parameter containing the retutned data samples or any per tag errors
);
```

**Remarks**

This function is used to read values for a single start and end time and sampling mode. This function is typically used to read a single tagname. However, a wildcard can be passed or multiple individual tagnames can be passed.

The list of sampling and calculation modes, filtering parameters and query modifiers are passed to this function to indicate if you want raw samples returned or you want the Data Archiver to perform some calculation or summarization of the data and return the results. For example you can instruct the Data Archiver to perform hourly averages and return the results instead of returning the samples themselves.

All calculated values are returned as Double Float, regardless of the data type of the tag. Sampling modes such as lab or interpolated will return data in the tag's current data type, even if the data type was changed over a period of time.

The start and end time that you enter are assumed to be in the GMT+0 time zone. If you need to convert local time zone timestamps to UTC use the ihTimeLCLPartsToUTCStructEx() utility function. The returned data will also be in GMT+0 timezone and you can use the ihTimeUTCStructToLCLPartsEx() function to convert to local time in preparation for trending or reporting.

The memory holding the returned data must be freed but do not free each field directly. Simply call the ihDataCloseRecordset() function.

For examples, refer to SDK Sample Programs.

**Returns**

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | The data sample was successfully read. |
| `ihSTATUS_API_TIMEOUT` | The client program did not receive a portion of the returned values before the timeout expired. Large reads are sent back in multiple responses and the |

| Returns Status | Message |
|---|---|
| | SystemAPI must receive at least a partial response before the timeout expires. For example if there is 90 second timeout configured, a read can take several minutes or more, as long as at least a part of the response is received every 90 seconds. |
| `ihSTATUS_NOT_CONNECTED` | If you are not currently connected to the Data Archiver |
| `ihSTATUS_ACCESS_DENIED` | If you are not allowed to read data:<br>• The tag you are reading has tag level security enabled.<br>• You are not a member of the ihReaders security group. .<br><br>The tag you are writing has tag level security enabled. The entire archive is set to read only in Historian Administrator. |
| `ihSTATUS_WRITE_OUTSIDE_ACTIVE` | The timestamp on the data sample is older than the “data is read only" |
| `ihSTATUS_FAILED` | For any other type of error. |

## ihDataCloseRecordset

This function used to free the data recordset that was allocated inside the System API in a previous read call.

### Prototype

```
void ihDataCloseRecordset {
ihDataRecordset *DataRecordset // the recordset to be freed. This comes from a previous call to
ihDataOpenRecordset
);
```

### Remarks

Use this function to free the data recordset that was allocated inside the System API in a previous read calls.

**Returns**

| Returns Status | Message |
|---|---|
| `VOID` | Void |

## ihDataClearAllFields

This function is used to clear the set of requested data fields.

```
void ihDataClearAllFields {
ihDataFields *Fields  // structure to be cleared
);
```

**Remarks**

Use this function to clear the set of requested data fields (timestamp,value,quality and so on) in preparation for doing a read. After

clearing, you should set the fields that you want returned.

**Returns**

| Returns Status | Message |
|---|---|
| `VOID` | Void |

## ihDataSubscribe

This function is used to subscribe or unsubscribe the data changes for a tag or tags in the Data Archiver.

**Prototype**

```
ihDataSubscribe {
ihServerHandle hServer, // server handle returned from previous call to ihServerConnect
ihChar *Tagname, // the tagname or wildcard of tagnames to subscribe
ihUNSIGNED long MinimumElapsedTime, // the fastest rate in milliseconds that you want data returned or 0 to return
 every data change
ihBoolean Subscribe // TRUE to setup a subscription or FALSE to stop receiving changes from an existing subscription
);
```

**Remarks**

Use this function to subscribe or unsubscribe to data changes for a tag or tags in the Data Archiver. Subscriptions will be delivered within a few seconds of data change and are delivered to you as DataRecordset structures through the callback you previously registered with ihDataRegisterCallback() callback function. You must register a callback first if you want to receive changes. This ihDataSubscribe() function indicates the tags you want to monitor.

You can subscribe to one tag by passing a specific tagname or pass a * or other wildcard to get changes for multiple tags. If you need to monitor multiple individual tag names, then call ihDataSubscribe() function once for each tag.

Most applications will pass a 0 for minimum elapsed time indicating they want all data changes. But you can specify a minimum rate. However, you may not be notified of every change. As you are notified, you should receive the data, possibly queuing the data change, and return from the callback.

Subscriptions stay in place even after the connection lost to archiver. However, once you call ihServerDisconnect() all subscriptions are cleared. It is not necessary to unsubscribe before calling disconnect but it is suggested.

For Examples, refer Sample Programs section.

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | if the data was successfully read. |
| `ihSTATUS_API_TIMEOUT` | if the client program did not receive a portion of the returned values before the timeout expired. Large reads are sent back in multiple responses and the SystemAPI must receive at least a partial response before the timeout expires. For example, if there is 90 second timeout configured, a read can take several minutes or more, as long as at least a part of the response is received every 90 seconds. |
| `ihSTATUS_NOT_CONNECTED` | if you are not currently connected to the Data Archiver. |
| `ihSTATUS_FAILED` | For any other type of error. |

## ihDataGetCurrentValue

Use this function to get the current values of the one or more tags.

**Prototype**

```
ihDataGetCurrentValue {
ihServerHandle hServer,
ihDataFields *Fields,
unsigned long NumberOfTags,
ihChar **Tagnames,
ihDataProperties *DataValues,
ihAPIStatus *ErrorStatuses
);
```

**Remarks**

The ihDataGetCurrentValue function returns timestamp, value, quality, and comments. You can choose to return any of these values using the ihDataFields.

Values are returned based on the tag data type. You should free the returned values using `ihDataFreeCurrentValue()` function. Do not free the memory in your program.

You can have per tag errors. For example, if the tag does not exist or you are not allowed to read it.

**Returns**

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | The data sample was successfully read. |
| `ihSTATUS_ACCESS_DENIED` | If you are not allowed to add or modify tags, or you are not a member of the ihReaders security group. |
| `ihSTATUS_FAILED` | If the tag does not exist. |

## ihDataFreeCurrentValue

Use this function to free the memory in your program for the current value of the selected tags.

**Prototype**

```
ihDataFreeCurrentValue {
ihUNSIGNED long NumberOfRecords,
ihDataProperties *DataValues
);
```

**Remarks**

Use this function to free what is returned from ihDataGetCurrentValue() function. Do not free the memory any other way.

**Returns**

| Returns Status | Message |
|---|---|
| `VOID` | Void |

## ihDataSubscribe

Use this function to subscribe or unsubscribe to data changes for a tag or tags in the Data Archiver. Subscriptions will be delivered within a few seconds of data change and are delivered to you as DataRecordset structures through the callback you previously registered with ihDataRegisterCallback() callback function. You must register a callback first if you want to receive changes. This ihDataSubscribe() function indicates the tags you want to monitor.

You can subscribe to one tag by passing a specific tagname or pass a * or other wildcard to get changes for multiple tags. If you need to monitor multiple individual tag names, then call ihDataSubscribe() function once for each tag.

Most applications will pass a 0 for minimum elapsed time indicating they want all data changes. But you can specify a minimum rate. However, you may not be notified of every change. As you are notified, you should receive the data, possibly queuing the data change, and return from the callback.

Subscriptions stay in place even after the connection lost to archiver. However, once you call ihServer-Disconnect() all subscriptions are cleared. It is not necessary to unsubscribe before calling disconnect but it is suggested.

For Examples, refer to the Sample Programs section.

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | The data sample was successfully read. |
| `ihSTATUS_API_TIMEOUT` | The client program did not receive a portion of the returned values before the timeout expired. Large reads are sent back in multiple responses and the SystemAPI must receive at least a partial response before the timeout expires. For example if there is 90 second timeout configured, a read can take sev- |

| Returns Status | Message |
|---|---|
| | eral minutes or more, as long as at least a part of the response is received every 90 seconds. |
| `ihSTATUS_NOT_CONNECTED` | If you are not currently connected to the Data Archiver |
| `ihSTATUS_FAILED` | For any other type of error. |

## ihDataOpenMultiRecordset

This function is used if you need to do a single read but use different sampling or calculation modes or different start and end times.

```
ihDataOpenMultiRecordset {
ihServerHandle hServer,
ihUNSIGNED long NumberOfRequests,
ihDataFields *Fields,
ihDataCriteria *Criteria,
ihDataRecordset *DataRecordset
);
```

### Remarks

This function can be used if you need to do a single read but use different sampling or calculation modes or different start and end times. You do not need to use this function simply to read multiple tags.

After checking the return value of the API call you can check the return value of each included query.

### Returns

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | The data sample was successfully read. |
| `ihSTATUS_API_TIMEOUT` | The client program did not receive a portion of the returned values before the timeout expired. Large reads are sent back in multiple responses and the SystemAPI must receive at least a partial response before the timeout expires. For example if there is 90 second timeout configured, a read can take sev- |

| Returns Status | Message |
|---|---|
| | eral minutes or more, as long as at least a part of the response is received every 90 seconds. |
| `ihSTATUS_NOT_CONNECTED` | If you are not currently connected to the Data Archiver |
| `ihSTATUS_ACCESS_DENIED` | If you are not allowed to read data.<br>• The tag you are reading has tag level security enabled.<br>• You are not a member of the ihReaders security group |
| `ihSTATUS_DATA_RETRIEVAL_COUNT_EXCEEDED` | If the read exceeded the Max Query Time or Max Query Intervals configured in Historian Administrator. |
| `ihSTATUS_FAILED` | For any other type of error. |

## ihDataCloseMultiRecordset

This function is used to free the multiple recordsets returned from a `ihDataOpenMultipleRecordset()` call.

### Prototype

```
void ihDataCloseMultiRecordset {
ihUNSIGNED long NumberOfRequests,
ihDataRecordset *DataRecordsetd
);
```

### Remarks

Use this function to properly free the multiple recordsets that were allocated inside the System API in a previous read call.

### Returns

| Returns Status | Message |
|---|---|
| `VOID` | Void |

## ihArchiver options

### ihArchiverFreeOption Function

### Prototype

```
ihArchiveFreeOption{

ihChar *OptionValue // the string returned from ihArchiveGetOption()

);
```

### Remarks

Use this function to free the memory inside the structure. Do not free the fields in your code.

### Returns

| Returns Status | Message |
|---|---|
| `VOID` | Void |

### ihArchiverGetOption

This function retrieves an Archiver option from the data store.

```
ihArchiverGetOption {

ihServerHandle hServer,// server handle returned from previous call to ihServerConnect

ihOptionEx *Option, // the option and data store name

ihChar **OptionValue // output parameter containing the option value

);
```

This function retrieves an Archiver option from a data store. You can indicate the data store in the ihOptionEx structure. You can pass NULL as the data store name to use the default data store.

You can get an option value to confirm it is set as requested by your application, or you can use this call to verify that a set option call was successful.

## ihArchiverSetOption

This function sets an Archiver option from a data store.

## ihArchiverGetStatistics

This function is used to get performance statistics from the Data Archiver.

```
ihArchiverGetStatistics(
ihServerHandle hServer,
ihArchiveStatistics *Statistics
);
```

This returns the performance statistics about data writes, failed writes, and disk space usage.

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | If the option was successfully read |
| `ihSTATUS_API_TIMEOUT` | If the option could not be read and you can try again later |
| `ihSTATUS_NOT_CONNECTED` | If you are not currently connected to the Data Archiver |
| `ihSTATUS_FAILED` | For any other type of error. |

## Archiver Configuration Functions

You can add and configure tags as part of Data Archiver configuration but there are other options that can be configured.

A subset of all the options and their meanings is given in this section, and the rest are listed in the `ihapi.h` file.

Most options are set as per data store and some are applicable for the overall Data Archiver.

### Menu ihOptions

| Option | Description |
|---|---|
| `ihArchiveDefaultPath` | The default path for new archive creations. |
| `ihArchiveActiveHours` | Set to a larger number if you need to write data with timestamps in the past older than the "data is read only after” setting in Historian Administrator. |
| `ihArchiveDefaultSize` | The default size for new archives, when you are not using `ihArchiveDuration` for time based archives. |
| `ihArchiveAutomatic-Create`, | TRUE if new archives should be created or FALSE if old archives should be reused instead. |

| Option | Description |
|---|---|
| `ihArchiveAutomaticFreeSpace` | The amount of disk space to be left free. |
| `ihArchiveOverwriteOld` | TRUE if you should overwrite old archives instead of refusing incoming data. |
| `ihArchiveDefaultBaseFileName` | The default base file name which typically matches the computer name but you can update this if you move the Data Archiver to a new computer. |
| `ihArchiveDefaultBaseArchiveName` | The default base file name which typically matches the computer name but you can update this if you move the Data Archiver to a new computer. |
| `ihArchiveDefaultBackupPath` | The default location for archive backups; typically this location is the same location as the online archives. |
| `ihArchiveCreateOfflineArchive` | Set to TRUE if you need to write data with a timestamp before the first archive so that an archive gets created. |
| `ihMessageOnDataUpdate` | Set to TRUE if you want an audit trail message for every data sample that is overwritten. |
| `ihArchiverNumReadThreads` | Increase from default if you need more read threads so that you can perform parallel reads. |
| `ihSecurityUseLocalGroups` | Set to FALSE if you want to use domain security groups. |
| `ihArchiverMaxIntervalRetrievalCount` | Increase this number if you typically do large queries that return error `ihSTATUS_DATA_RETRIEVAL_COUNT_EXCEEDED`. |
| `ihArchiverMaxQueryTime` | Increase this number if you typically do large queries that return error `ihSTATUS_DATA_RETRIEVAL_COUNT_EXCEEDED`. |
| `ihArchiverMaintainAutoRecoveryFiles` | Set to FALSE if you do not want to use this feature. |
| `ihArchiverAllowDataOverwrites` | Set to FALSE if you don’t want to allow accidental or intentional data overwrites. |
| `ihArchiverTargetPrivateBytes` | Leave as 0 for system managed unless you have a specific target. |
| `ihArchiverNumWriteThreads` | Leave as default value. |

| Option | Description |
|---|---|
| `ihSecurityStrict-ClientAuthentication` | True to only permit SSPI-based authentication. |
| `ihSecurityStrictCol-lectorAuthentication` | True to only permit collector connections from 5.0 and above. |
| `ihArchiveTotalDura-tion` | Number of Hours that the archives for a DataStore can span (only used for trending DataStores). |
| `ihArchiveDuration-Type` | The type of archive duration. It is the integer value of `ihArchiveDurationType`. |
| `ihArchiveDuration` | Number of Units of time an archive can hold. Defined by `ihArchiveDurationType`. |
| `StructFieldsInBrows-esByDefault` | TRUE if you want each field of a structure to appear as its own tag during tag browse. Useful for legacy clients that are not aware of structures. |
| `ihOptionMax` | The highest option number for this version of historian. Use this to check for invalid option values. |

## Archiver Configuration Functions

## ihArchiveOpenRecordset

This function is used to get the list of archives listed in the server.

### Prototype

```
ihArchiveOpenRecordset
(
ihServerHandle hServer,
ihArchiveCriteria *Criteria,
ihArchiveRecordset *ArchiveRecordset


);
```

### Remarks

Use the criteria if you only want a specific archive or archives from a specific data store. Otherwise you can get all archives and loop through them to see their start and end times and sizes for example.

Call `ihArchiveCloseRecordset()` to free the memory when you are done. Do not free memory in your program.

**Returns**

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | On success. |
| `ihSTATUS_API_TIMEOUT` | If the option could not be read. You can try again later. |
| `ihSTATUS_NOT_CONNECTED` | If you are not currently connected to the Data Archiver. |
| `ihSTATUS_FAILED` | For any other type of error. |

## ihArchiveCloseRecordset

This function is used to free up the memory associated with the `ArchiveRecordset`.

**Prototype**

```
ihArchiveCloseRecordset(


ihArchiveRecordset *ArchiveRecordset

);
```

**Remarks**

Use this function to free the memory associated with the ArchiveRecordset that was returned from ihArchiveOpenRecordset(). Do not free the fields in your code.

**Returns**

| Returns Status | Message |
|---|---|
| `Void` | Void |

## ihArchiveGetOption

This function retrieves an Archiver option from the data store. This option controls the behavior of the Data Archiver for writes, security and timeouts.

**Prototype**

```
ihArchiveGetOption {

ihServerHandle hServer, ihOptionEx *Option, ihChar **OptionValue

);
```

**Remarks**

This function retrieves an Archiver option from a data store. You can indicate the data store in the `ihOptionEx` structure. You can pass NULL as the data store name to use the default data store. You can get an option value to confirm it is set as requested by your application, or you can use this call to verify that a set option call was successful.

**Returns**

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | On success. |
| `ihSTATUS_API_TIMEOUT` | If the option could not be read. You can try again later. |
| `ihSTATUS_NOT_CONNECTED` | If you are not currently connected to the Data Archiver. |
| `ihSTATUS_FAILED` | For any other type of error. |

## ihArchiveSetOption

This function sets an Archiver option from a data store

**Prototype**

```
ihArchiveSetOption {

ihServerHandle hServer, ihOptionEx *Option, ihChar *OptionValue

);
```

**Remarks**

This function sets an Archiver option from a data store. You can indicate the data store in the `ihOptionEx` structure. You can pass NULL as the data store name to use the default data store.

The list of possible options in *ihapi.h* is listed in the Archiver Configuration Functions. Set an option value when the default does not meet the needs of your application or to confirm that no other application has changed it. Archive option changes are audited in the Historian messages and you can see them using `ihMessageOpenRecordset` call.

**Returns**

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | On success. |
| `ihSTATUS_API_TIMEOUT` | If the option could not be read. You can try again later. |
| `ihSTATUS_NOT_CONNECTED` | If you are not currently connected to the Data Archiver. |
| `ihSTATUS_FAILED` | For any other type of error. |

## ihArchiverFreeOption

This function is used to free the memory inside the structure returned from an `ihArchiveGetOption()`.

**Prototype**

```
ihArchiveFreeOption{


ihChar *OptionValue // the string returned from ihArchiveGetOption()


);
```

**Remarks**

Use this function to free the memory inside the structure. Do not free the fields in your code.

**Returns**

| Returns Status | Message |
|---|---|
| `Void` | Void. |

## ihArchiveGetProperties

This function retrieves the properties for an archive in a data store identified by the ArchiveDescriptor.

**Prototype**

```
ihArchiveGetProperties (
ihServerHandle hServer,

ihArchiveDescriptor *ArchiveDescriptor,

ihArchiveProperties *Archive
);
```

**Remarks**

You can pass NULL as the data store name to use the default data store.

The `ihArchiveProperties` contains information such as the StartTime and EndTime and size of the archive.

Make sure to free the archive properties using `ihArchiveFreeProperties()` function.

**Returns**

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | On success. |
| `ihSTATUS_API_TIMEOUT` | If the option could not be read. You can try again later. |
| `ihSTATUS_NOT_CONNECTED` | If you are not currently connected to the Data Archiver. |
| `ihSTATUS_FAILED` | For any other type of error. |

## ihArchiveSetProperties

This function sets the properties for an archive.

**Prototype**

```
ihArchiveSetProperties {
ihServerHandle hServer,

ihArchiveProperties *Archive
);
```

**Remarks**

This is the only archive property set by a user program to make the archive read only.

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | On success. |
| `ihSTATUS_API_TIMEOUT` | If the option could not be read. You can try again later. |
| `ihSTATUS_NOT_CONNECTED` | If you are not currently connected to the Data Archiver. |
| `ihSTATUS_FAILED` | For any other type of error. |

## ihArchiveFreeProperties

This function is used to free up the memory associated with the ArchiveProperties returned from an `ihArchiveGetProperties` call.

**Prototype**

```
ihArchiveFreeProperties {
ihArchiveProperties *ArchiveProperties
);
```

**Remarks**

Use this function to free the memory. Do not free the memory by calling operating system functions.

**Returns**

| Returns Status | Message |
|---|---|
| `Void` | Void. |

## ihArchiveGetStatistics

This function is used to get performance statistics from the Data Archiver of a particular data store.

**Prototype**

```
ihArchiveGetStatistics {
ihServerHandle hServer, ihString DataStoreName, ihArchiveStatistics *Statistics
```

```
);
```

**Remarks**

This returns the performance statistics about data writes, failed writes, and disk space usage. To get the performance of a particular data store you need pass the data store name.

**Returns**

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | On success. |
| `ihSTATUS_API_TIMEOUT` | If the option could not be read. You can try again later. |
| `ihSTATUS_NOT_CONNECTED` | If you are not currently connected to the Data Archiver. |
| `ihSTATUS_FAILED` | For any other type of error. |

## ihArchiveBackupResponse

This function is used by the Alarm Archiver only.

**Prototype**

```
ihArchiveBackupResponse
(
ihServerHandle hServer, ihCallbackId CallbackId, ihAPIStatus Status, ihBoolean FinalMessage
);
```

**Remarks**

None

**Returns**

None

## ihArchiveRemoveResponse

This function is used by the Alarm Archiver only.

**Prototype**

```
ihArchiveRemoveResponse {

ihServerHandle hServer, ihCallbackId CallbackId, ihAPIStatus Status, ihBoolean FinalMessage

);
```

**Remarks**

None

**Returns**

None

## ihConfigurationGetProperties

This function returns the configuration properties such as the Data Archiver version.

**Prototype**

```
ihConfigurationGetProperties {

ihServerHandle hServer, // server handle returned from previous call to ihServerConnect


ihConfigurationProperties *Properties // output parameter to contain the properties

);
```

## ihConfigurationFreeProperties

This function is used to free the memory inside the structure.

**Prototype**

```
void ihConfigurationFreeProperties {

ihConfigurationProperties *Properties // pointer to the structure filled in by

ihConfigurationGetProperties

);
```

**Remarks**

Use this function to free the memory inside the structure. Do not free the fields within your code.

**Returns**

| Returns Status | Message |
|---|---|
| `Void` | Void. |

## Archiver Backup/Restore Functions

You can use Archive Backup/Restore functions to backup archive data.

There are synchronous and asynchronous calls for loading archives, making a backup and removing an archive. The synchronous functions are typically used for smaller archives because the operation completes in a few seconds. The asynchronous functions are typically used for larger archives because the operation can takes longer, one minute or more, to complete.

Archive Backup/Restore functions are as follows:

### ihArchiveBackup

This function is used to back up archive (.iha) files. If you are storing alarms and events data in Historian, an IHA backup also backs up any alarms.

#### Prototype

```
ihArchiveBackup {

ihServerHandle hServer, // server handle returned from previous call to ihServerConnect


ihArchiveDescriptor *ArchiveDescriptor,


ihString BackupFileName


);
```

#### Remarks

This function is used to back up the archive (.iha) files residing on the server specified. The ArchiveDescriptor specifies the archive name that you want to add and the DataStore name to which it should be added. The BackupFileName has the backup of the IHA file with the name and the file location that you specified. For example: `C:\Program Files\Historian Data\Archives\myarchive.iha.`

#### Returns

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | If the option was successfully read. |
| `ihSTATUS_ACCESS_DENIED` | If you are not allowed to perform the operations. For most options you need to be a member of the `ihArchiveAdmins` group |

| Returns Status | Message |
|---|---|
| `ihSTATUS_API_TIMEOUT` | If the option could not be read. You can try again later. |
| `ihSTATUS_NOT_CONNECTED` | If you are not currently connected to the Data Archiver. |
| `ihSTATUS_FAILED` | For any other type of error. |

## ihArchiveBackupEx

This function is used to back up the archive (.iha) files and process Windows messages at the same time.

### Prototype

```
ihArchiveBackupEx {

ihServerHandle hServer,// server handle returned from previous call to ihServerConnect


ihArchiveDescriptor *ArchiveDescriptor,


ihString BackupFileName, long hWnd);
```

### Remarks

This function is used to back up the archive (.iha) files residing on the server specified. The ArchiveDescriptor specifies the archive name that you want to add and the DataStore name to which it should be added. The BackupFileName has the backup of the IHA file with the name and the file location that you specified. For example: `C:\Program Files\Historian Data\Archives\myarchive.iha`.

### Returns

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | If the option was successfully read. |
| `ihSTATUS_ACCESS_DENIED` | If you are not allowed to perform the operations. For most options you need to be a member of the `ihArchiveAdmins` group |
| `ihSTATUS_API_TIMEOUT` | If the option could not be read. You can try again later. |

| Returns Status | Message |
|---|---|
| `ihSTATUS_NOT_CONNECTED` | If you are not currently connected to the Data Archiver. |
| `ihSTATUS_FAILED` | For any other type of error. |

## ihArchiveRegisterBackupCallBack

This function is used to register a callback which should be called after the `ihArchiveBackup()` function completes. Archive backup can take several minutes. By using this callback, your System API program can know that the backup is complete.

### Prototype

```
ihArchiveRegisterBackupCallBack {

ihServerHandle hServer, // server handle returned from previous call to ihServerConnect


ihArchiveBackupCallbackFunction CallbackFunction,

void *UserParameter

);
```

### Remarks

The register callback will be performed when the Archive backup is completed on Historian.

### Returns

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | If the option was successfully read. |
| `ihSTATUS_ACCESS_DENIED` | If you are not allowed to perform the operations. For most options you need to be a member of the `ihArchiveAdmins` group |
| `ihSTATUS_API_TIMEOUT` | If the option could not be read. You can try again later. |
| `ihSTATUS_NOT_CONNECTED` | If you are not currently connected to the Data Archiver. |
| `ihSTATUS_FAILED` | For any other type of error. |

## ihArchiveRemove

This function is used to delete or unload an archive file. You can unload a file if you no longer need it at the current time but may need it later or in another archive. You can delete a file if you do not expect to need it again or if you have made a backup.

**Prototype**

```
ihArchiveRemove {

ihServerHandle hServer, // server handle returned from previous call to ihServerConnect


ihArchiveDescriptor *ArchiveDescriptor,


ihBoolean ShouldDeleteFile


)
```

**Remarks**

This function is used to delete or unload the archive file residing on the server specified. The ArchiveDescriptor specifies the archive name that you want to add and the DataStore name to which it should be added. ShouldDeleteFile determines whether the file should be deleted or not.

The following are the scenarios which you can use with `ihArchiveRemove()` function:

| If Should-DeleteFile is: | ZipReg-istry enabled | Zip suc-cessful | Then... |
|---|---|---|---|
| TRUE | YES | YES | Archive file is deleted. |
| TRUE | YES | NO | The archive file is not deleted but un-loaded. |
| FALSE | YES | YES | The archive file is not deleted. |
| FALSE | YES | NO | The archive file is not deleted. |

**Returns**

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | If the option was successfully read. |

| Returns Status | Message |
|---|---|
| `ihSTATUS_ACCESS_DENIED` | If you are not allowed to perform the operations. For most options you need to be a member of the `ihArchiveAdmins` group |
| `ihSTATUS_API_TIMEOUT` | If the option could not be read. You can try again later. |
| `ihSTATUS_NOT_CONNECTED` | If you are not currently connected to the Data Archiver. |
| `ihSTATUS_FAILED` | For any other type of error. |

## ihArchiveRemoveEx

This function is used to delete or unload the archive files and process Windows messages at the same time.

### Prototype

```
ihArchiveRemoveEx {
ihServerHandle hServer, // server handle returned from previous call to ihServerConnect

ihArchiveDescriptor *ArchiveDescriptor,

ihBoolean ShouldDeleteFile,

long hWnd
);
```

### Remarks

This function is used to delete or unload the archive file residing on the server specified. The `ArchiveDescriptor` specifies the archive name that you want to add and the DataStore name to which it should be added. `ShouldDeleteFile` determines whether the file should be deleted or not.

The following are the scenarios which you can use with ihArchiveRemoveEX() function:

| If Should-DeleteFile is: | ZipReg-istry enabled | Zip suc-cessful | Then... |
|---|---|---|---|
| TRUE | YES | YES | Archive file is deleted. |

| If ShouldDeleteFile is: | ZipRegistry enabled | Zip successful | Then... |
|---|---|---|---|
| TRUE | YES | NO | The archive file is not deleted but unloaded. |
| FALSE | YES | YES | The archive file is not deleted. |
| FALSE | YES | NO | The archive file is not deleted. |

**Returns**

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | If the option was successfully read. |
| `ihSTATUS_ACCESS_DENIED` | If you are not allowed to perform the operations. For most options you need to be a member of the `ihArchiveAdmins` group |
| `ihSTATUS_API_TIMEOUT` | If the option could not be read. You can try again later. |
| `ihSTATUS_NOT_CONNECTED` | If you are not currently connected to the Data Archiver. |
| `ihSTATUS_FAILED` | For any other type of error. |

## ihArchiveRegisterRemoveCallback

This function allows you to register a callback for removing an archive. This callback function is called when you remove the archive from Proficy Historian.

**Prototype**

```
ihArchiveRegisterRemoveCallback {

ihServerHandle hServer, // server handle returned from previous call to ihServerConnect


ihArchiveRemoveCallbackFunction CallbackFunction,


void *UserParamter

);
```

**Remarks**

To register a function as a callback, you must have a callback or a function with the same signature as ihArchiveRemoveCallback function, and then pass it to `ihArchiveRegisterRemoveCallback`.

**Returns**

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | If the option was successfully read. |
| `ihSTATUS_ACCESS_DENIED` | If you are not allowed to perform the operations. For most options you need to be a member of the `ihArchiveAdmins` group |
| `ihSTATUS_API_TIMEOUT` | If the option could not be read. You can try again later. |
| `ihSTATUS_NOT_CONNECTED` | If you are not currently connected to the Data Archiver. |
| `ihSTATUS_FAILED` | For any other type of error. |

## ihArchiveAdd

This function is used to create a new archive or load an existing one.

**Prototype**

```
ihArchiveAdd {

ihServerHandle hServer, // server handle returned from previous call to ihServerConnect


ihArchiveDescriptor *ArchiveDescriptor,


ihString FileLocation, ihBoolean ShouldCreateFile,


ihUNSIGNED long FileSize

);
```

**Remarks**

This function is used to create or load the archive file residing on the server specified. The ArchiveDescriptor specifies the archive name that you want to add and the DataStore name to which it should be added. Then, specify the FileLocation. To determine whether the file should be created or not with the specified FileSize, use ShouldCreateFile.

**Returns**

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | If the option was successfully read. |
| `ihSTATUS_ACCESS_DENIED` | If you are not allowed to perform the operations. For most options you need to be a member of the `ihArchiveAdmins` group |
| `ihSTATUS_API_TIMEOUT` | If the option could not be read. You can try again later. |
| `ihSTATUS_NOT_CONNECTED` | If you are not currently connected to the Data Archiver. |
| `ihSTATUS_FAILED` | For any other type of error. |

## ihArchiveAddEx

This function is used to create a new archive or load an existing one.

**Prototype**

```
ihArchiveAddEx {

ihServerHandle hServer, // server handle returned from previous call to ihServerConnect


ihArchiveDescriptor *ArchiveDescriptor,


ihString FileLocation,


ihBoolean ShouldCreateFile,


ihUNSIGNED long FileSize,


long hWnd


);
```

**Remarks**

This function is used to create or load the archive file residing on the server specified. The ArchiveDescriptor specifies the archive name that you want to add and the DataStore name to which it

should be added. Then, specify the FileLocation. To determine whether the file should be created or not with the specified FileSize, use ShouldCreateFile.

**Returns**

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | If the option was successfully read. |
| `ihSTATUS_ACCESS_DENIED` | If you are not allowed to perform the operations. For most options you need to be a member of the `ihArchiveAdmins` group |
| `ihSTATUS_API_TIMEOUT` | If the option could not be read. You can try again later. |
| `ihSTATUS_NOT_CONNECTED` | If you are not currently connected to the Data Archiver. |
| `ihSTATUS_FAILED` | For any other type of error. |

## ihArchiveRegisterLoadCallback

This function allows you to register a callback for adding an archive file. That registered callback is called when you add the archive file in Historian. It can take several minutes to load large archive files, so by using callbacks your System API program can be notified when the archives finish loading.

**Prototype**

```
ihArchiveRegisterLoadCallback {


ihServerHandle hServer, // server handle returned from previous call to ihServerConnect


ihArchiveLoadCallbackFunction CallbackFunction,


void *UserParameter

);
```

**Remarks**

You need to have a callback or a function with the same signature as ihArchiveLoadCallback function, and then pass it to ihArchiveRegisterLoadCallback to register that function as a callback.

**Returns**

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | If the option was successfully read. |
| `ihSTATUS_ACCESS_DENIED` | If you are not allowed to perform the operations. For most options you need to be a member of the `ihArchiveAdmins` group |
| `ihSTATUS_API_TIMEOUT` | If the option could not be read. You can try again later. |
| `ihSTATUS_NOT_CONNECTED` | If you are not currently connected to the Data Archiver. |
| `ihSTATUS_FAILED` | For any other type of error. |

## User Defined Type Functions

User defined data types should be added to the Data Archiver before they can be used in tags. The User defined type functions are as follows:

ihUserDefinedTypeAdd

This function is used to add a user defined type.

**Prototype**

```
ihUserDefinedTypeAdd(

ihServerHandle hServer, // server handle returned from previous call to ihServerConnect

ihUserDefinedTypeProperties *UserDefinedType, // structure containing the user defined type

ihString *ErrorMsg // output parameter to contain any error string if there is error. The memory, if there is an error,
will be allocated in the System API and should be freed using ihUserDefinedTypeFreeError
);
```

**Remarks**

If the set already exists, it is overwritten by this new set. You need to be an *ihTagAdmin* to add or modify a set. The memory, if there is an error, will be allocated in the System API and should be freed using `ihUserDefinedTypeFreeErrorMessage`.

**Returns**

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | On success. |
| `ihSTATUS_API_TIMEOUT` | If the server name or IP address provided cannot be located. |
| `ihSTATUS_NOT_CONNECTED` | If you are not currently connected. |
| `ihSTATUS_ACCESS_DENIED` | If you are not allowed to add or modify the user defined types, perhaps you are not a member of the `ihTagAdmins` group. |
| `ihSTATUS_FAILED` | For any other type of error. |

## ihUserDefineTypeDelete

This function is used to delete a user defined type.

### Prototype

```
ihUserDefinedTypeDelete(

ihServerHandle hServer, // server handle returned from previous call to ihServerConnect

ihString UserDefinedTypeName, // the name of the user defined type. Any characters can be used other than * or ?

ihString *ErrorMsg // output parameter to contain any error string if there is error. The memory, if there is an error,
will be allocated in the System API and should be freed using ihUserDefinedTypeFreeError
);
```

### Remarks

You need to be an `ihTagAdmin` to be able to delete a User Defined type. The memory, if there is an error, will be allocated in the System API and should be freed using `ihUserDefinedTypeFreeErrorMessage`.

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | On success. |
| `ihSTATUS_API_TIMEOUT` | If the server name or IP address provided cannot be located. |
| `ihSTATUS_NOT_CONNECTED` | If you are not currently connected. |

| Returns Status | Message |
|---|---|
| `ihSTATUS_ACCESS_DENIED` | If you are not allowed to add or modify the user defined types, perhaps you are not a member of the `ihTagAdmins` group. |
| `ihSTATUS_FAILED` | For any other type of error. |

## ihUserDefinedTypeRename

This function is used to rename the existing user defined type name.

### Prototype

```
ihUserDefinedTypeRename(

ihServerHandle hServer, // server handle returned from previous call to ihServerConnect

ihString UserDefinedTypeName, // the current name of the user defined type

ihString NewUserDefinedTypeName // the new name
);
```

### Remarks

After you rename, the old name is available for reuse.

### Returns

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | On success. |
| `ihSTATUS_API_TIMEOUT` | If the server name or IP address provided cannot be located. |
| `ihSTATUS_NOT_CONNECTED` | If you are not currently connected. |
| `ihSTATUS_ACCESS_DENIED` | If you are not allowed to add or modify the user defined types, perhaps you are not a member of the `ihTagAdmins` group. |
| `ihSTATUS_FAILED` | For any other type of error. |

## ihUserDefinedTypeExists

Use this function to check whether the user defined type exists or not.

### Prototype

```
ihUserDefinedTypeExists(

ihServerHandle hServer, // server handle returned from previous call to ihServerConnect

ihString UserDefinedTypeName // the name to check

);
```

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | On success. |
| `ihSTATUS_API_TIMEOUT` | If the server name or IP address provided cannot be located. |
| `ihSTATUS_NOT_CONNECTED` | If you are not currently connected. |
| `ihSTATUS_ACCESS_DENIED` | If you are not allowed to add or modify the user defined types, perhaps you are not a member of the `ihTagAdmins` group. |
| `ihSTATUS_FAILED` | For any other type of error. |

## ihUserDefinedTypeSetProperties

This function is used to set the user defined type properties.

### Prototype

```
ihUserDefinedTypeSetProperties(

ihServerHandle hServer, // server handle returned from previous call to ihServerConnect

ihUserDefinedTypeProperties *UserDefinedType, // the name and properties to set

ihString *ErrorMsg // output parameter to contain any error string if there is error. The memory, if there is a

an error, will be allocated in the System API and should be freed using ihUserDefinedTypeFreeError

);
```

**Remarks**

You need to be an `ihTagAdmin` to set the properties. The memory, if there is an error, will be allocated in the System API and should be freed using `ihUserDefinedTypeFreeErrorMessage`.

**Returns**

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | On success. |
| `ihSTATUS_API_TIMEOUT` | If the server name or IP address provided cannot be located. |
| `ihSTATUS_NOT_CONNECTED` | If you are not currently connected. |
| `ihSTATUS_ACCESS_DENIED` | If you are not allowed to add or modify the user defined types, perhaps you are not a member of the `ihTagAdmins` group. |
| `ihSTATUS_FAILED` | For any other type of error. |

## ihUserDefinedTypeGetProperties

This function returns the user defined type properties.

**Prototype**

```
ihUserDefinedTypeGetProperties(

ihServerHandle hServer, // server handle returned from previous call to ihServerConnect

ihString UserDefinedTypeName , // the name of the custom data type

ihUserDefinedTypeProperties *UserDefinedType // output parameter containing the returned properties. The return
);
```

**Remarks**

You need to be an `ihTagAdmin` to get the properties.

**Returns**

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | On success. |
| `ihSTATUS_API_TIMEOUT` | If the server name or IP address provided cannot be located. |
| `ihSTATUS_NOT_CONNECTED` | If you are not currently connected. |
| `ihSTATUS_ACCESS_DENIED` | If you are not allowed to add or modify the user defined types, perhaps you are not a member of the `ihTagAdmins` group. |
| `ihSTATUS_FAILED` | For any other type of error. |

## ihUserDefinedTypeFreeProperties

This function is used to free the memory allocated by the `ihUserDefinedTypeGetProperties()` call.

### Prototype

```
void ihUserDefinedTypeFreeProperties(


ihUserDefinedTypeProperties *UserDefinedTypeProps // properties as returned from an

ihUserDefinedTypeGetProperties call

);
```

### Remarks

You need to call this once to free all memory in the recordset. Do not free the memory in your program.

### Returns

| Returns Status | Message |
|---|---|
| `Void` | Void. |

## ihUserDefinedTypeOpenRecordset

This function is used to return the list of user defined types.

### Prototype

```
ihUserDefinedTypeOpenRecordset(
```

```
ihServerHandle hServer, // server handle returned from previous call to ihServerConnect


ihString Mask, // * or specific set name or some other mask to specify the list


ihUserDefinedTypeRecordset *RecordSet // output parameter to contain the returned list. The list should be
freed with ihUserDefinedTypeFreed when done using the ihUserDefinedTypeCloseRecordset()
);
```

**Remarks**

You need to specify the user defined type set name to open the recordset. This function call should be freed after the retrieval using `ihUserDefinedTypeCloseRecordset`.

**Returns**

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | On success. |
| `ihSTATUS_API_TIMEOUT` | If the server name or IP address provided cannot be located. |
| `ihSTATUS_NOT_CONNECTED` | If you are not currently connected. |
| `ihSTATUS_ACCESS_DENIED` | If you are not allowed to add or modify the user defined types, perhaps you are not a member of the `ihTagAdmins` group. |
| `ihSTATUS_FAILED` | For any other type of error. |

## ihUserDefinedTypeCloseRecordset

This function used to free the user defined type recordset that was allocated inside the System API in a previous read call.

**Prototype**

```
void ihUserDefinedTypeCloseRecordset(


ihUserDefinedTypeRecordset *Recordset // recordset to be freed
);
```

**Returns**

| Returns Status | Message |
|---|---|
| `Void` | Void. |

### ihUserDefinedTypeFreeErrorMessage

This function is used to free the memory from the previous error message call.

#### Prototype

```
ihUserDefinedTypeFreeErrorMessage(


ihString ErrorMsg // string to be freed as returned from a previous call

);
```

#### Returns

| Returns Status | Message |
|---|---|
| `Void` | Void. |

## Utility Functions

Utility functions help you to make use of the other System API calls such as data read and write and connect. Utility Functions are as follows:

### ihTimeLCLPartsToUTCStructEx

This function returns timestamps that can be used in read and write functions.

```
ihTimeLCLPartsToUTCStructEx {


ihServerHandle hServer, // server handle returned from previous call to ihServerConnect or just pass

ihINVALID_HANDLE if not using server time zone

int Year, // the 4 digit year such as 1998 or 2004 in the local time zone

int Month, // the month 1 (January) to 12 (December) in the local time zone

int Day, // the day 1 to 31 in the local time zone

int Hour, // the hour 0 to 23 in the local time zone

int Minute, // the minute 1 to 59 in the local time zone

int Second, // the second 1 to 59 in the local time zone

int MilliSecond, // the millisecond 0 to 999 in the local time zone

ihTimeZones TimeZoneFlag, // client or server or explicit timezone
```

```
int TimeZoneBiasExplicit, // only used if TimeZoneFlag is explicit

int DaylightSavingsTime, // TRUE if you want to use the Daylight Saving Time setting in Control Panel,

FALSE if you want to never use Daylight Saving Time.

ihTimeStruct *Time // the output parameter containing the converted timestamp

);
```

**Remarks**

Use this function to return timestamps to be used in data read and write functions for start and end time of queries. The function takes a server handle however, you need not be connected to the server unless you specify the TimeZoneFlag as server time zone. If you are just using timestamps in local time zone you can use `ihTimeLCLPartsToUTCStruct()`.

**Returns**

| Returns Status | Message |
|---|---|
| `ihuSTATUS_OK` | On success. |

## ihTimeUTCStructToLCLPartsEx

This function is used to convert the timestamps returned by System API into your local time zone.

```
void ihTimeUTCStructToLCLPartsEx {

ihServerHandle hServer, // server handle returned from previous call to ihServerConnect or just pass

ihINVALID_HANDLE if not using server time zone

int *Year, // output parameter to contain the 4 digit year

int *Month, // output parameter to contain the month 1 to 12

int *Day, // output parameter to contain the day 1 to 31

int *Hour, // output parameter to contain the hour 0 to 23

int *Minute, // output parameter to contain the minute 0 to 59

int *Second, // output parameter to contain the second 0 to 59

int *MilliSecond, // output parameter to contain the milliseconds 0 to 999

ihTimeZones TimeZoneFlag, // client or server or explicit timezone

int TimeZoneBiasExplicit, // only used if TimeZoneFlag is explicit

int DaylightSavingsTime, // TRUE if you want to use the Daylight Saving Time setting in Control Panel,

FALSE if you want to never use Daylight Saving Time.

ihTimeStruct *UTCTime // the timestamp to be converted


);
```

**Remarks**

Use this function to convert timestamps returned by the System API into your local time zone for display. The function takes a server handle however, you need not be connected to the server unless you specify the TimeZoneFlag as server time zone. If you are just using client time zone you can use `ihTimeUTCStructToLCLParts()`.

**Returns**

| Returns Status | Message |
|---|---|
| `Void` | Void. |

## ihTimeCurrentUTCStruct

This function returns the current time stamp as UTC. You can add and subtract seconds from this time to read and write data with timestamps relative to now.

**Prototype**

```
void ihTimeCurrentUTCStruct {

ihTimeStruct *UTCTime // output parameter containing the current time

);
```

**Remarks**

Use this function to get the current time so that you can use it in other System API functions such as the end time of a query or the timestamp of a data write.

**Returns**

| Returns Status | Message |
|---|---|
| `Void` | Void. |

## ihUtilAnsiToUnicode

This function is used to convert Unicode string to ANSI format which can be used by API applications and other programs.

**Prototype**

```
ihUtilAnsiToUnicode {

char *MBStr, // the non Unicode string
```

```
ihChar *WCStr // the output buffer to contain the converted Unicode string
);
```

**Remarks**

Use this function if you want to convert any Unicode string returned by the System API to ANSI format that can be used by other API application and programs.

You must allocate the buffer for the Unicode string. Allocate a large enough buffer as System API does not validate the length.

**Returns**

| Returns Status | Message |
|---|---|
| `Void` | Void. |

## ihUtilUnicodeToAnsi

This function is used to convert ANSI strings to Unicode in order to pass them into System API functions.

**Prototype**

```
ihUtilUnicodeToAnsi {
char *MBStr,ihChar *WCStr
);
```

**Remarks**

Use this function to convert ANSI strings to Unicode for passing them into the System API connect or read or write functions and so on.

You must allocate the buffer for the ANSI string. Allocate a large enough buffer as System API does not validate the length.

**Returns**

| Returns Status | Message |
|---|---|
| `Void` | Void. |

## ihUtilErrorDesc

This function returns a string for a numeric error code. The string is not translated into other languages.

```
ihUtilErrorDesc {
ihStatus ErrorNum, // the error number
ihString ErrorDesc, // the output parameter to contain the string
int Len // the length of the output buffer
);
```

**Remarks**

This utility function returns an English language string for a numeric error code if you don't want to provide your own.

**Returns**

| Returns Status | Message |
|---|---|
| `Void` | Void. |

## Time Functions

## ihTimeLCLPartsToUTCStruct

Use this function to convert Local Date/Time Parts to Universal Time Coordinated Structure.

**Prototype**

```
ihTimeLCLPartsToUTCStruct {
int Year,
int Month,
int Day,
int Hour,
int Minute,
int Second,
int MilliSecond,
ihTimeStruct *UTCTime};
```

**Remarks**

The System API functions that read and write data require timestamps to be in GMT+0 timezone. This function can be used to convert your local timestamps into a format that can be passed into other System API functions such as `ihDataAdd`.

Enter your timestamp in the first 6 parameters and the converted timestamp gets returned in the UTC Time parameter. The time that you enter should be in the timezone of the system that is running your program. If you need additional control of the time zone and Daylight Saving Time parameters then use the `ihTimeLCLPartsToUTCStructEx` function.

**Returns**

| Returns Status | Message |
|---|---|
| `Void` | Void. |

## ihTimeUTCStructToLCLParts

Use this function to convert Universal Time Coordinated (UTC) time zone structure to Local Date/Time parts.

```
ihTimeUTCStructToLCLParts {
int *Year,
int *Month,
int *Day,
int *Hour,
int *Minute,
int *Second,
int *MilliSecond,
ihTimeStruct *UTCTime
};
```

**Remarks**

Enter your timestamp in as the UTCTime parameter and the converted timestamp is returned in the first 6 parameters. For example the samples returned from an `ihDataOpenRecordset` have timestamps that can be converted with this function.

The hours and minutes and seconds will be returned in the timezone of the system running your program. If you need additional control over the time zone or Daylight Saving Time parameters then use the `ihTimeUTCStructToLCLPartsEx` function.

**Returns**

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | On Success. |

| Returns Status | Message |
|---|---|
| `ihSTATUS_FAILED` | |

## ihTimeUTCStructToFileTime

Use this function to convert UTC structure to FILETIME.

### Prototype

```
ihTimeUTCStructToFileTime {
ihTimeStruct *UTCTime,
void *FileTime
};
```

### Remarks

Use this function to convert Historian UTC timestamps into a FILETIME that could be passed into other Microsoft Windows functions.

### Returns

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | On success. |
| `ihSTATUS_FAILED` | For any other type of error. |

## ihTimeLCLFileTimeToUTCStruct

Use this function to convert FILETIME (LOCAL) to UTC structure.

### Prototype

```
ihTimeLCLFileTimeToUTCStruct {
void *FileTime, ihTimeStruct *UTCTime
};
```

### Remarks

If you got a FILETIME from some Microsoft Windows call, use this function to convert it to a format that can be used by other System API functions. The FILETIME is assumed to be in UTC timezone.  If the FILETIME is in local time then use the ihTimeLCLFileTimeToUTCStruct function.

### Returns

| Returns Status | Message |
|---|---|
| `1` | On success. |
| `0` | Any type of error. |

## Query Modifiers Functions

ihQueryModifiersAssign

Use this function to clear the previously assigned modifiers and then set new modifiers.

### Prototype

```
ihQueryModifiersAssign {
ihQueryModifiers *Modifiers, int NumModifiers, ...
};
```

### Returns

This function would not be called by a user program.

| Returns Status | Message |
|---|---|
| `Void` | Void. |

## ihQueryModifiersClear

Use this function to clear the modifiers.

### Prototype

```
ihQueryModifiersClear {
 ihQueryModifiers *Modifiers
};
```

### Remarks

This function would not be called by a user program.

### Returns

| Returns Status | Message |
|---|---|
| `Void` | Void. |

## ihQueryModifiersSet

Use this function to add a modifier to a mask of modifiers.

### Prototype

```
ihQueryModifiersSet {
ihQueryModifiers *Modifiers, ihQueryModifier Modifier
};
```

### Remarks

Use this function to add a querymodifer to a mask of modifiers that would then be passed into a data read call. The set of available modifiers is available in the `ihQueryModifier` enumeration.

### Returns

| Returns Status | Message |
|---|---|
| `Void` | Void. |

## ihQueryModifiersIsSet

Use this function to check whether the specified modifier is already present in the modifier mask.

### Prototype

```
ihQueryModifiersIsSet {
ihQueryModifiers *Modifiers, ihQueryModifier Modifier
};
```

### Remarks

You can use this function to prepare a query modifier mask which can then be passed to a data read call.

### Returns

| Returns Status | Message |
|---|---|
| `True` | The modifier has already been added to the task. |
| `False` | Otherwise. |

## ihQueryModifierOpenRecordset

Use this function to get a list of possible query modifiers from the destination archiver. Older versions of Historian might support fewer modifiers than the version your program is running on.

### Prototype

```
ihQueryModifierOpenRecordset {

ihServerHandle hServer, ihQueryModiferRecordset *QueryModiferRecordset

};
```

### Remarks

You should use this function if your program can be communicating with older Data Archivers to ensure your query modifier is available.

Free the returned recordset using the `ihQueryModifierCloseRecordset` function.

### Returns

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | On success. |
| `ihSTATUS_FAILED` | For any other type of error. |

## ihQueryModifierCloseRecordset

Use this function to free memory returned from `ihQueryModifierOpenRecordset`.

### Prototype

ihQueryModifierCloseRecordset { ihQueryModiferRecordset *RecSet };

### Remarks

Use this function and do not call any Microsoft Windows functions to free the memory.

### Returns

| Returns Status | Message |
|---|---|
| `Void` | Void |

## ihDataCriteriaFromString

Use this function to build a data criteria from the criteria string.

### Prototype

```
ihDataCriteriaFromString {

ihServerHandle hServer, ihDataCriteria *Criteria, ihString CriteriaString

};
```

### Remarks

This function would not be called by a user program.

### Returns

| Returns Status | Message |
|---|---|
| None | None |

## DataStore Functions

## ihDataStoreAdd

Use this function to create additional data stores if your license allows it.

### Prototype

```
ihDataStoreAdd { ihServerHandle hServer,

ihString DataStoreName,

ihBoolean   IsDefault,

ihString Description,

ihDataStorageType StorageType

};
```

### Remarks

Some data stores are created automatically when the Data Archiver is started. Use this function if you need to create additional ones. The number of data stores is licensed on your key.

### Returns

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | On success. |
| `ihSTATUS_NOT_PERMITTED` | If adding the data store would exceed your licensed count. |
| `ihSTATUS_FAILED` | For any other type of error. |

## ihDataStoreDelete

Use this function to delete a data store on the specified Data Archiver.

### Prototype

```
ihDataStoreDelete {

ihServerHandle hServer, ihString DataStoreName

};
```

### Remarks

You can only delete a data store if all tags have been removed from it. You cannot delete the System data store.

### Returns

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | On success. |
| `ihSTATUS_FAILED` | For any other type of error. |

## ihDataStoreRename

Use this function to rename an existing data store.

### Prototype

```
ihDataStoreRename {

ihServerHandle hServer, ihString DataStoreName, ihString NewDataStoreName

};
```

### Remarks

Once the data store is renamed it can only be referred to by its new name.

You must be a member of the ihArchive Admin security group to perform renames.

**Returns**

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | On success. |
| `ihSTATUS_FAILED` | For any other type of error. |

## ihDataStoreOpenRecordset

Use this function to get the details of the configured data stores on the specified Data Archiver.

**Prototype**

```
ihDataStoreOpenRecordset {

ihServerHandle hServer, ihString DataStoreMask, ihDataStoreRecordset *Recordset

};
```

**Remarks**

You can specify a name mask or just pass NULL or ”r;*” to get all data stores.

Free the returned list using the `ihDataStoreCloseRecordset` function.

**Returns**

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | On success. |
| `ihSTATUS_FAILED` | For any other type of error. |

## ihDataStoreCloseRecordset

Use this function to free the memory in the list returned from `ihDataStoreOpenRecordset`.

**Prototype**

```
ihDataStoreCloseRecordset {

ihDataStoreRecordset *Recordset


};
```

**Remarks**

Use this function to free the memory, do not use Microsoft Windows calls to free the memory.

**Returns**

| Returns Status | Message |
|---|---|
| `Void` | Void |

## ihDataStoreSetProperties

Use this function to set or change properties of an existing data store. You use `ihArchiverSetOption` to set options and use this function to indicate a data store is the default or to set the description.

**Prototype**

```
ihDataStoreSetProperties {
ihServerHandle hServer,
ihString DataStoreName,
ihBoolean IsDefault,
ihString Description,
ihDataStorageType StorageType
};
```

**Remarks**

Most data store configuration will be done by setting options but this function is available if you need to change the default data store.

**Returns**

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | On success. |
| `ihSTATUS_API_TIMEOUT` | If the servername or IP address provided cannot be located. |
| `ihSTATUS_NOT_CONNECTED` | If you are not currently connected. |
| `ihSTATUS_ACCESS_DENIED` | If you are not allowed to add or modify tags. Possibly you are not a member of the ”r" ih Tag Admins” group. |

| Returns Status | Message |
|---|---|
| `ihSTATUS_LIC_TOO_MANY_TAGS` | If adding this tag would exceed your licensed tag count. |
| `ihSTATUS_FAILED` | For any other type of error. |

## Security Functions

## ihSecurityGroupOpenRecordset

Use this function to get the list of security groups that exist in the operating system where the Data Archiver is running. This can be a different list than where your client program is running.

All security groups would be returned, not just the specific ihSecurity Admins, ihTag Admins, and so on. You must free the returned recordset using the `ihSecurityGroupCloseRecordset` call.

### Prototype

```
ihSecurityGroupOpenRecordset {

ihServerHandle hServer, ihSecurityGroups *Grps

};
```

### Remarks

To know the list of security groups while setting up a tag to use tag level security, use this function call to give you the name of any groups that exist at the archiver. And when you assign that name to a tag read security group, then the Data Archiver will check that group when the tag is read. This is why you need the list of groups at the Data Archiver and not at the client PC, because the Data Archiver should be able to access the group and its members.

All security groups would be returned, not just the specific ih Security Admins, ih Tag Admins, and so on.

You must free the returned recordset using the `ihSecurityGroupCloseRecordset` call.

### Returns

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | On success. |
| `ihSTATUS_FAILED` | For any other type of error. Other errors on their respective Network or Security failures. |

## ihSecurityGroupCloseRecordset

Use this function to free the list returned from a `ihSecurityGroupOpenRecordset` call.

### Prototype

```
ihSecurityGroupCloseRecordset {

ihSecurityGroups *Grps

};
```

### Remarks

Use this function instead of freeing the memory via operating system calls.

### Returns

| Returns Status | Message |
|---|---|
| `Void` | Void |

## ihSecurityGetOption

Use this function to get the value of security options from the specified Data Archiver.

### Prototype

```
ihSecurityGetOption {

ihServerHandle hServer, ihOption Option, ihChar **OptionValue

};
```

### Remarks

You could get the value of security related options such as `ihSecurityStrictClientAuthentication` or `ihSecurityStrictCollectorAuthentication`.

You need to be a member of ih Readers security group to be able to get

options.

### Returns

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | On success. |

| Returns Status | Message |
|---|---|
| `ihSTATUS_FAILED` | For any other type of error. Other errors on their respective Network or Security failures. |

## ihSecurityFreeOption

Use this function to free the memory returned by a previous `ihSecurityGetOption` call.

### Prototype

```
ihSecurityFreeOption{

ihChar *OptionValue

};
```

### Remarks

Use this function to free the option value. Do not free the memory using operating system calls.

### Returns

| Returns Status | Message |
|---|---|
| Void | Void |

## ihSecurityGetmemberships

Use this function on an established connection to determine what Proficy Historian security groups that the Data Archiver considers you a member of. This will determine what permissions you have within Historian.

### Prototype

```
ihSecurityGetmemberships {

ihServerHandle hServer, ihSecurityGrpMembership *Memberships

};
```

### Remarks

Group memberships are established at connect time. If groups are created or your account is added to groups then you need to disconnect and connect again to get the correct level of permissions.

### Returns

| Returns Status | Message |
|---|---|
| `ihSTATUS_OK` | On success. |
| `ihSTATUS_FAILED` | For any other type of error. Other errors on their respective Network or Security failures. |

# Sample Programs

## Sample Programs

Sample programs are provided with the API, demonstrating how to perform common tasks. The following sample programs are supplied with the System API:

- `ihCopyData.cpp` – demonstrates copying all the tags and data from Server1 to Server2 including UserDefinedType tags then subscribes to tag and data changes.
- `ihPlotLike.cpp` - shows how to determine the default server and reading multiple tags in one call.

**Note:**
No guidance is given regarding application conversion from other products or APIs. It is assumed that the reader is familiar with the Historian features and functionality.

# Chapter 10. Historian User API

## Historian User API Overview

### About the Historian User API

The Historian User API provides high-speed read/write access to Historian data and read access to Historian tags. There is no access to archives, alarms, events, or messages.

Use this API to develop applications that read and write data to the Historian server when the Historian SDK and Historian OLE DB do not meet your project requirements for performance or programming language.

You can connect the Historian API to a local Historian server in the same manner as to a remote Historian server by simply providing the name of the server. This name must be the computer name or the IP address of the target Historian server, and the server must have a TCP/IP connectivity. If you use the computer name of the server rather than the IP address, the IP address must be available to the client through DNS, a WINS server, or through the local host table.

**Note:**
At this time, the Historian User API supports single-byte strings only.

The Historian Client Access API is a .NET Core assembly that interacts with Historian from any .NET Core applications. Since the API uses .NET Core, you can use it on any operating system. The required DLLs are available in the `DotNetCoreCAAPI` folder in the ISO. This folder also contains a sample program, which you can use a starting point to build an application.

**Note:**
You can still use the old Client Access API, which is a .NET assembly. You can use this API only on Windows. It is installed when you install Historian Client Tools. By default, the Install Wizard places both the API and Client Access `.dlls` in GAC (Global Assembly Cache). It is recommended that you add Historian Client Access API references from the `INSTALLPATH` directory since global assembly cache is part of the run-time environment.

The applications that call into the User API are limited by the security access granted at the server level.

## Prerequisites

You must run the programs you create with the Historian User API on a machine with the following software installed and configured:

- Historian 5.5 or greater
- Historian 5.5 or greater Client Tools

The Historian User API has been tested with Visual Studio .NET 2003, 2005, and 2008.

The Historian User API has no additional hardware requirements.

Historian allows you to develop both 32-bit and 64-bit User API programs.

**Note:**
To build a 32-bit User API program on a 64-bit operating system, you must rename `ihuapi32.lib` to `ihuapi.lib` and include it in your program.

# Connect Functions

## Connect Functions Overview

This group of functions provides a means to connect to and disconnect from an Historian server. A minimal number of properties are exposed.

**Connect Functions**


- ihuConnect *(on page 1514)*
- ihuConnectEx *(on page 1516)*
- ihuDisconnect *(on page 1517)*
- ihuSetConnectionParameters *(on page 1517)*
- ihuRestoreDefaultConnectionParameters *(on page 1519)*
- ihuServerRegisterCallbacks *(on page 1519)*
- ihuBrowseCollectors *(on page 1520)*


## ihuConnect

Use the `ihuConnect` function to connect to a Historian server. The function provides a server handle to be used in subsequent calls.

## Prototype

```
ihuConnect {
 in MSO MSO Char * server,
 in MSO MSO Char * username, in MSO MSO Char * password, out long * serverhandle
};
```

## Remarks

The inputs to the function are `server`, `username`, and `password`. Each has a default value if `NULL` is passed.

- `server`: If `NULL` is passed, then the connection attempt is to the local machine.
- `username` and `password`: If `NULL` is passed, then the username that owns the process is used. Most of the time this is the same as the user logged into the operating system. However, in the case of a program running as a service, you can specify a username and password that the process should use.

The output of this function is a server handle.

Server handles are valid only during the lifetime of the process. They should not be saved to a file and reused.

You do not need to call `ihuConnectEx` more than one time for a username and password. If the connection to the server was lost and restored, the handle can be used after reconnection. If the server was not available at connect time, a handle is still returned, which you can use as soon as the connection becomes available. Reconnects are performed inside the API. The application should wait and retry reads and writes with the returned server handle. Reads and writes succeed after the underlying connection is re-established.

You should still call `ihuDisconnect` with the returned server handle, even if an error is returned.

A connection to the server consumes a Client Access License (CAL) only if you have not already accessed the server from your current IP address. There is no way to connect without consuming a CAL.

## Returns

The `ihuConnect` function returns the following values:

- `ihuSTATUS_OK`
- `ihuSTATUS_FAILED`
- `ihuSTATUS_API_TIMEOUT`

- `ihuSTATUS_NOT_VALID_USER`
- `ihuSTATUS_LIC_TOO_MANY_USERS`

## ihuConnectEx

Use the `ihuConnectEx` function to connect to a server with store and forward support.

### Prototype

```
ihuErrorCode IHUAPI ihuConnectEx (
MSO Char * server,
MSO Char * username,
MSO Char * password,
MSO Char * buffername,
unsigned long MaxMegMemory,
unsigned long MinMegDiskFree,
long * serverhandle
);
```

### Remarks

The inputs to this function are:

- `server`: If `NULL` is passed, then the connection attempt is to the local machine.
- `username` and `password`: If `NULL` is passed, then the username that owns the process is used. Most of the time this is the same as the user logged into the operating system. However, in the case of a program running as a service, you can specify a username and password that the process should use.
- `buffername`: The target filename and location to store buffered data. The buffer file name must be unique.
- `MaxMegMemory`: Maximum memory in MB. Buffered data is stored in this memory until it is full and is later stored to disk.
- `MinMegDiskFree`: Minimum free disk space in MB.

The output of this function is a server handle.

Server handles are valid only during the lifetime of the process. They should not be saved to a file and reused.

There is no need to call `ihuConnectEx` more than one time for a username and password. If the connection to the server was lost and restored, the handle can be used after reconnection. If the server was not

available at connect time, a handle is still returned, which you can use as soon as the connection becomes available. Reconnects are performed inside the API. The application should wait and retry reads and writes with the returned server handle. Reads and writes succeed after the underlying connection is re-established.

You should still call `ihuDisconnect` with the returned server handle, even if an error is returned.

A connection to the server consumes a Client Access License (CAL) only if you have not already accessed the server from your current IP address. There is no way to connect without consuming a CAL.

### Returns

The `ihuConnectEx` function returns the following values:

- `ihuSTATUS_OK`
- `ihuSTATUS_FAILED`
- `ihuSTATUS_API_TIMEOUT`
- `ihuSTATUS_NOT_VALID_USER`
- `ihuSTATUS_LIC_TOO_MANY_USERS`

## ihuDisconnect

Use the `ihuDisconnect` function to release connection resources.

### Prototype

```
ihuDisconnect {
in long serverhandle
};
```

### Returns

The `ihuDisconnect` function returns the following values:

- `ihuSTATUS_OK`

## ihuSetConnectionParameters

Use the `ihuSetConnectionParameters` function to set the socket connection timeout.

### Prototype

```
ihuSetConnectionParameters {

in IHU_CONNECTION_PARAMETERS*Params

};
```

### Remarks

The input to the function is `IHU_CONNECTION_PARAMETERS` structure to pass in a connection timeout:

```
/* Client-side, global connection parameters */

typedef struct {

int Size; // Structure size in bytes

int TCPConnectionWindow; // Max time to establish a TCP connection with server in seconds (default 5s)

} IHU_CONNECTION_PARAMETERS;
```

The default connection timeout is 5 seconds. There is no maximum value, but it is not recommended to set the value to longer than 60 seconds. If you increase the timeout, server connection attempts take more time to return if the server is unavailable.

### Sample Code

If an archiver is too busy to process connections, you can set a longer timeout by using the following code:

```
IHU_CONNECTION_PARAMETERS params;

params.Size = sizeof(IHU_CONNECTION_PARAMETERS);

params.TCPConnectionWindow = 30; // extend window to 30s

ihuSetConnectionParameters(&params);
```

**Note:**
This code applies only to the connections made from your program. You must make this call each time you run your program.

### Returns

The `ihuSetConnectionParameters` function returns the following values:

- `ihuSTATUS_OK`
- `ihuSTATUS_FAILED`

## ihuRestoreDefaultConnectionParameters

Use the `ihuRestoreDefaultConnectionParameters` function to reset all connection parameters (for example, the socket connection timeout) to default values.

### Prototype

```
ihuRestoreDefaultConnectionParameters {
 void,
};
```

### Remarks

The example code resets the connection timeout to 5 seconds.

### Returns

`IhuSTATUS_OK`

## ihuServerRegisterCallbacks

Use the `ihuServerRegisterCallbacks` function if you want your program to be notified of changes in buffering or connection state. For example, your program can be notified of connection loss or that buffering is full.

### Prototype

```
ihuErrorCode IHUAPI ihuServerRegisterCallbacks (
long hServer,
void *UserParameter,
long *RegisterCallbacksStatus,
void *BufferCallbackFunction,
void *ConnectionCallbackFunction
);
```

### Remarks

The inputs to this function are:

- `hServer`: Server handle from the connection. Callbacks are as per server handle, so you can have callbacks for some connections and not for others, or specific callbacks for different handles.
- `UserParameter`: `NULL` or an integer value that you want passed to you in the callback.
- `RegisterCallbacksStatus`: Callback setup success or failure.

- `BufferCallbackFunction`: Callback function for buffer state changes.
- `ConnectionCallbackFunction`: Callback function for connection state changes.

### Returns

`ihuServerRegisterCallbacks` returns the following values:

- `ihuSTATUS_OK`
- `ihuSTATUS_FAILED`

## ihuBrowseCollectors

Use the `ihuBrowseCollectors` function to browse collectors that are connected to the archiver.

### Prototype

```
ihuErrorCode IHUAPI ihuBrowseCollectors (
long hServer,
MSO Char *InterfaceNameMask
IHU_COLLECTOR **Collectors,
int *NumOfCollectors
);
```

### Remarks

The inputs to the function are:

- `hServer`: Server handle for the server to be browsed.
- `*InterfaceNameMask`: Pass `*` for all interfaces, or the interface name or detailed mask for efficiency.

The outputs of the function are:

- `**Collectors`: Returns a list of interfaces/collectors.
- `*NumOfCollectors`: Returns the number of interfaces found.

### Returns

The `ihuBrowseCollectors` function returns the following values:

- `ihuSTATUS_OK`
- `ihuSTATUS_FAILED`

# Archiver Functions

## Archiver Functions Overview

This group of functions provides a means to read and write a set of archiver properties. A minimum of properties (only those related to reads, writes, and tag browses) are available. See the code comments in the Historian User API header (`ihuapi.h`) for more details.

### Archiver Functions


- ihuGetArchiverProperty *(on page 1523)*
- ihuSetArchiverProperty *(on page 1521)*


## ihuSetArchiverProperty

Use the `ihuSetArchiveProperty` function to change the value of specific Historian server options.

### Prototype

```
ihuSetArchiverProperty
(long serverhandle,
MSO Char *ArchiveProperty,
MSO Char *PropertyValue)
```

The following properties are supported:

**Table 265. Archiver Properties**

| Property | Read/ Write Access | Values | Description |
|---|---|---|---|
| `ARCHIVER_PROP_CREATE_OFFLIN_EARCHIVES` | Read/ Write | `1`, `0` | When set to `0`. (false) writes before the start time of the earliest archive are not enabled. Set to `1` only when you are migrating legacy data, and set the value back to `0` after migration is complete. Leaving this value set to `1` can cause excessive archive creation, especially when data is sent to the archiver out of order. |
| `ARCHIVER_PROP_CON` | Read-Only | | A large value that changes when the tag configuration changes. Typically used to cache configuration properties such as a set of tags. You can |

**Table 265. Archiver Properties (continued)**

| Property | Read/ Write Access | Values | Description |
|---|---|---|---|
| FIGSERIAL-NUMBER | | | determine when your cache is out of date by comparing your saved serial number with the current serial number. |
| ARCHIVER_-PROP_AC-TIVEHOURS | Read/ Write | 1-232800 hours (30 years) | The number of hours before the present time that the archive data became read-only. When migrating legacy data, you might need to set a large value. |
| SECURI-TY_PROP_-STRICT-CLIENTAU-THENTI-CATION | Read/ Write | 1. 0 | Enables or disables strict client authentication. Set to 1 (true) by default for new Historian installations. When set to 1, only version 4.7 or later clients can access the Historian server. |
| SECURI-TY_PROP_-STRICT-COLLEC-TORAUTHEN-TICATION | Read/ Write | 1. 0 | Enables or disables strict collector authentication. Set to 1 (true) by default for new Historian installations. When set to 1, only version 4.7 or later collectors can access the Historian server. |

## Returns

The ihuSetArchiverProperty returns the following values:

- ihuSTATUS_INVALID_PARAMETER
- ihuSTATUS_OK
- ihuSTATUS_ACCESS_DENIED

| To change this property... | You must be a member of this group... |
|---|---|
| ARCHIVER_PROP_CREATEOFFLINEARCHIVES | ihArchive Admins or ihSecurity Admins |
| ARCHIVER_PROP_ACTIVEHOURS | ihArchive Admins |

| To change this property... | You must be a member of this group... |
|---|---|
| `SECURITY_PROP_STRICTCLIENTAUTHENTICATION` | ihArchive Admins |
| `SECURITY_PROP_STRICTCOLLECTORAUTHENTICATION` | ihArchive Admins |

## ihuGetArchiverProperty

Use the `ihuGetArchiverProperty` function to read the value of certain archiver properties.

### Prototype

```
ihuGetArchiverProperty {
  in long serverhandle,
  in MSO Char *ArchiveProperty,
  out MSO Char *PropertyValue
};
```

### Remarks

To read properties, you must be a member of the ih Readers security group. There is no need for you to be a member of the ih Security Admins or ih Archive Admins groups, unless you want to change property values.

### Returns

- `ihuSTATUS_OK`
- `ihuSTATUS_INVALID_PARAMETER`
- `ihuSTATUS_ACCESS_DENIED`

# Tag Functions

## Tag Functions Overview

Tag functions provide browse, add, modify, and delete access to the configured tags on a server. You can retrieve the properties for each tag with User API calls. For example, you can retrieve user-defined collection parameters for a tag before collection, or retrieve user-configured display parameters to prepare for plotting.

You can retrieve string and numeric tag properties. Every tag property is classified as string or numeric (double float). For example, Boolean properties are converted to double float and time data type properties are converted to string.

You can add, delete, or modify tags using the User API.

You can browse tags in three different ways. The simplest mode uses `ihuFetchTagCache` to filter by tagname. To filter by additional properties, such as data type or collector name, use `ihuFetchTagCacheEx`. Finally, to browse in multiple browser windows at the same time, for example, to compare the tags on one server with those on another server, use `ihuFetchTagCacheEx2` or `ihuFetchTagCacheEx3`.

### Tag Functions


- ihuCreateTagCacheContext *(on page 1527)*
- ihuFetchTagCache *(on page 1527)*
- ihuFetchTagCacheEx *(on page 1528)*
- ihuFetchTagCacheEx2 *(on page 1528)*
- ihuFetchTagCacheEx3 *(on page 1529)*
- ihuGetTagNameCacheIndex *(on page 1530)*
- ihuGetTagNameCacheIndexEx2 *(on page 1530)*
- ihuGetNumericTagPropertyByTagname *(on page 1531)*
- ihuGetNumericTagPropertyByIndex *(on page 1531)*
- ihuGetNumericTagPropertyByIndexEx2 *(on page 1532)*
- ihuGetStringTagPropertyByTagname *(on page 1533)*
- ihuGetStringTagPropertyByTagnameEx2 *(on page 1533)*
- ihuGetStringTagPropertyByIndex *(on page 1534)*
- ihuGetStringTagPropertyByIndexEx2 *(on page 1535)*
- ihuTagAdd *(on page 1535)*
- ihuTagDelete *(on page 1536)*
- ihuTagRename *(on page 1537)*
- ihuTagCacheCriteriaClear *(on page 1539)*
- ihuTagCacheCriteriaClearEx2 *(on page 1539)*
- ihuTagCacheCriteriaSetStringProperty *(on page 1540)*
- ihuTagCacheCriteriaSetStringPropertyEx2 *(on page 1540)*
- ihuTagCacheCriteriaSetNumericProperty *(on page 1541)*
- ihuTagCacheCriteriaSetNumericPropertyEx2 *(on page 1541)*
- ihuTagClearProperties *(on page 1542)*
- ihuTagSetStringProperty *(on page 1542)*
- ihuTagSetNumericProperty *(on page 1542)*
- ihuCloseTagCache *(on page 1542)*
- ihuCloseTagCacheEx2 *(on page 1543)*

## Tag Property Value Types

The following table lists the current set of tag properties that are exposed, and indicates whether they are numeric or string. Use ihuGetNumericTagPropertyByIndex *(on page 1531)* and ihuGetNumericTagPropertyByTagname *(on page 1531)* to retrieve numeric properties, and ihuGetStringTagPropertyByIndex *(on page 1534)* and ihuGetStringTagPropertyByTagname *(on page 1533)* to retrieve string properties.

**Table 266. Tag Property Value Types**

| Property | Value Type |
|---|---|
| `ihuTagPropTagname` | String |
| `ihuTagPropDescription` | String |
| `ihuTagPropEngineeringUnits` | String |
| `ihuTagPropComment` | String |
| `ihuTagPropDataType` | Numeric |
| `ihuTagPropFixedStringLength` | Numeric |
| `ihuTagPropInterfaceName` | String |
| `ihuTagPropSourceAddress` | String |
| `ihuTagPropCollectionType` | Numeric |
| `ihuTagPropCollectionInterval` | Numeric |
| `ihuTagPropCollectionOffset` | Numeric |
| `ihuTagPropLoadBalancing` | Numeric |
| `ihuTagProptime stampType` | Numeric |
| `ihuTagPropHighEngineeringUnits` | Numeric |
| `ihuTagPropLowEngineeringUnits` | Numeric |
| `ihuTagPropInputScaling` | Numeric |
| `ihuTagPropHighScale` | Numeric |
| `ihuTagPropLowScale` | Numeric |
| `ihuTagPropCollectorCompression` | Numeric |
| `ihuTagPropCollectorDeadbandPercentRange` | Numeric |

**Table 266. Tag Property Value Types (continued)**

| Property | Value Type |
|---|---|
| `ihuTagPropArchiveCompression` | Numeric |
| `ihuTagPropArchiveDeadbandPercentRange` | Numeric |
| `ihuTagPropSpare1` | String |
| `ihuTagPropSpare2` | String |
| `ihuTagPropSpare3` | String |
| `ihuTagPropSpare4` | String |
| `ihuTagPropSpare5` | String |
| `ihuTagPropReadSecurityGroup` | String |
| `ihuTagPropWriteSecurityGroup` | String |
| `ihuTagPropAdministratorSecurityGroup` | String |
| `ihuTagPropLastModified` (not currently implemented) | String |
| `ihuTagPropLastModifiedUser` | String |
| `ihuTagPropInterfaceType` | Numeric |
| `ihuTagPropStoreMilliseconds` | Numeric |
| `ihuTagPropUTCBias` | Numeric |
| `ihuTagPropNumberOfCalculationDependencies` | Numeric |
| `ihuTagPropCalculationDependencies` (only the first dependency is returned) | String |
| `ihuTagPropAverageCollectionTime` | Numeric |
| `ihuTagPropCollectionDisabled` | Numeric |
| `ihuTagPropArchiveCompressionTimeout` | Numeric |
| `ihuTagPropCollectorCompressionTimeout` | Numeric |
| `ihuTagPropDataDensity` | Numeric |

**Table 266. Tag Property Value Types (continued)**

| Property | Value Type |
|---|---|
| **Note:** Tag data density categories are minimum (1), medium (4), and maximum (7). | |
| `ihuTagPropNumberOfElements`<br>**Note:** If the `NumberOfElements` value is `-1`, the tag is an array tag. | Numeric |

## ihuCreateTagCacheContext

Use the `ihuCreateTagCacheContext` function to create tag cache context to use in the `ihuFetchTagCacheEx2` or `ihuFetchTagCacheEx3` functions.

### Prototype

```
void * IHUAPI ihuCreateTagCacheContext {
};
```

## ihuFetchTagCache

Use this function to fetch a current set of tags and properties from the Historian server and place them into an API cache for subsequent queries, such as ihuGetNumericTagPropertyByTagname *(on page 1531)*.

### Prototype

```
ihuFetchTagCache {
  in long serverhandle,
  in MSO Char * tagmask,
  out int * NumTagsFound
};
```

### Remarks

The function retrieves properties for each tag received.

There is a single API tag cache. You must free the previous tag cache before you do another fetch from the same or a different server. Keep this in mind when you access the tag cache from multiple threads in the same application. Alternatively, use the `ihuCreateTagCacheContext` and `ihuFetchTagCacheEx2` or `ihuFetchTagCacheEx3` functions.

### Returns

- `ihuSTATUS_OK`
- `ihuSTATUS_INVALID_PARAMETER`

## ihuFetchTagCacheEx

Use the `ihuFetcTagCacheEx` function to fetch a current set of tags and properties from the Historian server and place them into an API cache for subsequent queries, such as `ihuGetNumericTagPropertyByTagname`.

### Prototype

```
ihuFetchTagCacheEx {
  in long serverhandle,
  out int * NumTagsFound,
};
```

### Remarks

This function uses the criteria set up by `ihuTagCacheCriteriaSetStringProperty` and `ihuTagCacheCriteriaSetNumericProperty` for tag browsing.

### Returns

- `ihuSTATUS_OK`
- `ihuSTATUS_INVALID_PARAMETER`

## ihuFetchTagCacheEx2

Use the `ihuFetchTagCacheEx2` function to fetch a current set of tags and properties from the Historian server and place them into an API cache for subsequent queries.

### Prototype

```
ihuFetchTagCacheEx2 {
void * tagCacheContext,
long serverhandle,
MSO Char * tagmask,
int * NumTagsFound
};
```

### Remarks

This function uses the criteria specified in `ihuTagCacheCriteriaSetStringProperty` and `ihuTagCacheCriteriaSetNumericProperty` for tag browsing.

### Returns

- `ihuSTATUS_OK`
- `ihuSTATUS_INVALID_PARAMETER`

## ihuFetchTagCacheEx3

Use the `ihuFetchTagCacheEx3` function to fetch a current set of tags and properties from the Historian server and place them into an API cache for subsequent queries.

### Prototype

```
ihuFetchTagCacheEx3 {
void * tagCacheContext,
long serverhandle,
int * NumTagsFound
};
```

### Remarks

This function uses the criteria specified in `ihuTagCacheCriteriaSetStringProperty` and `ihuTagCacheCriteriaSetNumericProperty` for tag browsing.

### Returns

- `ihuSTATUS_OK`
- `ihuSTATUS_INVALID_PARAMETER`

## ihuGetTagNameCacheIndex

Use the `ihuGetTagNameCacheIndex` function to find the index of the specified tagname in the cache. You can use the index for fast access in subsequent get property by index calls. This call only queries the API tag cache. It does not access the archiver.

### Prototype

```
ihuGetTagnameCacheIndex {
 in MSO Char *Tagname
 out unsigned int *CacheIndex
};
```

### Remarks

The value returned in `CacheIndex` is a zero-based index suitable to be passed into the get property by index functions. The index is valid only if the function returns a success error code.

### Returns

- `ihuSTATUS_OK`
- `ihuSTATUS_INVALID_TAGNAME`

## ihuGetTagNameCacheIndexEx2

Use the `ihuGetTagNameCacheIndexEx2` function to find the index of the specified tagname in the cache. You can use the index for fast access in subsequent get property by index calls. This call only queries the API tag cache. It does not access the archiver.

### Prototype

```
ihuGetTagnameCacheIndexEx2 {
Void *TagCacheContext,
MSO Char *Tagname
unsigned int *CacheIndex
};
```

### Remarks

The value returned in `CacheIndex` is a zero-based index suitable to be passed into the get property by index functions. The index is valid only if the function returns a success error code.

### Returns

- `ihuSTATUS_OK`
- `ihuSTATUS_INVALID_TAGNAME`

## ihuGetNumericTagPropertyByTagname

Use the `ihuGetNumericTagPropertyByTagname` function to retrieve the value of a specified numeric tag property from the API tag cache for a specified tag. This call queries the cache, but not the archiver. The tag property may have changed since the cache was fetched.

### Prototype

```
ihuGetNumericTagPropertyByTagname {
  in MSO Char *Tagname,
  in ihuTagProperties TagProperty,
  out double *Value
};
```

### Remarks

The get by tagname function is slower than the get by index function because it loops through all tags in the cache to find the requested tag. To retrieve multiple properties for a tag, consider calling `ihuGetTagnameCacheIndex` to get the cache index of a tag so that you can use the get by index functions.

### Returns

- `ihuSTATUS_OK`
- `ihuSTATUS_INVALID_PARAMETER`
- `ihuSTATUS_INVALID_TAGNAME`

## ihuGetNumericTagPropertyByIndex

Use the `ihuGetNumericTagPropertyByIndex` function to retrieve the value of a specified numeric tag property from the API tag cache at a specified cache index. This call queries the cache, but not the archiver. The tag property may have changed since the cache was fetched.

### Prototype

```
ihuGetNumericTagPropertyByIndex {
  in int Index,
  in ihuTagProperties TagProperty,
```

```
  out double *Value
};
```

### Remarks

Use the get property by index functions after you locate tags with `ihuGetTagnameCacheIndex`, or when you want to iterate through all tags in the cache.

The index is zero-based.

Tags are returned from the cache in no particular order.

### Returns

- `ihuSTATUS_OK`
- `ihuSTATUS_INVALID_PARAMETER`
- `ihuSTATUS_INVALID_TAGNAME`

## ihuGetNumericTagPropertyByIndexEx2

Use the `ihuGetNumericTagPropertyByIndexEX2` function to retrieve the value of a specified numeric tag property from the API tag cache at a specified cache index. This call queries the cache, but not the archiver. The tag property may have changed since the cache was fetched.

### Prototype

```
ihuGetNumericTagPropertyByIndexEx2 {
void * TagCacheContext,
int Index,
ihuTagProperties TagProperty,
double *Value
};
```

### Remarks

Use the get property by index functions when you do not know the tagname.

The index is zero-based.

Tags are returned from the cache in no particular order.

### Returns

- `ihuSTATUS_OK`
- `ihuSTATUS_INVALID_PARAMETER`
- `ihuSTATUS_INVALID_TAGNAME`

## ihuGetStringTagPropertyByTagName

Use the `ihuGetStringTagPropertyByTagname` function to retrieve the value of a specified string tag property from the API tag cache for a specified tag. This call queries the cache, but not the archiver. The tag property may have changed since the cache was fetched.

### Prototype

```
ihuGetStringTagPropertyByTagname {
  in MSO Char *Tagname,
  in ihuTagProperties TagProperty,
  in int valuelength,
  out MSO Char *Value
};
```

### Remarks

The get by tagname function is slower than the get by index function because it loops through all tags in the cache to find the requested tag. To retrieve multiple properties for a tag, consider calling `ihuGetTagnameCacheIndex` to get the cache index of a tag so that you can use the get by index functions.

### Returns

- `ihuSTATUS_OK`
- `ihuSTATUS_INVALID_PARAMETER`
- `ihuSTATUS_INVALID_TAGNAME`

## ihuGetStringTagPropertyByTagNameEx2

Use the `ihuGetStringTagPropertyByTagnameEx2` function to retrieve the value of a specified string tag property from the API tag cache for a specified tag. This call queries the cache, but not the archiver. The tag property may have changed since the cache was fetched.

### Prototype

```
ihuGetStringTagPropertyByTagnameEx2 {
void * TagCacheContext, MSO
Char * Tagname,
ihuTagProperties TagProperty,
MSO Char * Value,
int valuelength,
};
```

### Remarks

The get by tagname function is slower than the get by index function because it loops through all tags in the cache to find the requested tag. To retrieve multiple properties for a tag, consider calling `ihuGetTagnameCacheIndex` to get the cache index of a tag so that you can use the get by index functions.

### Returns

- `ihuSTATUS_OK`
- `ihuSTATUS_INVALID_PARAMETER`
- `ihuSTATUS_INVALID_TAGNAME`

## ihuGetStringTagPropertyByIndex

Use the `ihuGetStringTagPropertyByIndex` function to retrieve the value of a specified string tag property from the API tag cache at a specified cache index. This call queries the cache, but not the archiver. The tag property may have changed since the cache was fetched.

### Prototype

```
ihuGetStringTagPropertyByIndex {
  in int Index,
  in ihuTagProperties TagProperty,
  out MSO Char *Value,
  in int valuelength,
};
```

### Remarks

Use the get property by index functions after you locate tags with `ihuGetTagnameCacheIndex`, or when you want to iterate through all tags in the cache.

The index is zero-based.

### Returns

- `ihuSTATUS_OK`
- `ihuSTATUS_INVALID_PARAMETER`
- `ihuSTATUS_INVALID_TAGNAME`

## ihuGetStringTagPropertyByIndexEx2

Use the `ihuGetStringTagPropertyByIndexEx2` function to retrieve the value of a specified string tag property from the API tag cache at a specified cache index. This call queries the cache, but not the archiver. The tag property may have changed since the cache was fetched.

### Prototype

```
ihuGetStringTagPropertyByIndexEx2 {

void * TagCacheContext,

int Index,

ihuTagProperties TagProperty,

MSO Char * Value,

int valueLength,

};
```

### Remarks

Use the get property by index functions when you do not know the tag name, or when you want to iterate through all tags in the cache.

The index is zero-based.

### Returns

- `ihuSTATUS_OK`
- `ihuSTATUS_INVALID_PARAMETER`
- `ihuSTATUS_INVALID_TAGNAME`

## ihuTagAdd

Use the `ihuTagAdd` function to add or modify tags.

### Prototype

```
ihuTagAdd {
  in long serverhandle,
};
```

### Remarks

This function uses `serverhandle` as a parameter.

Tag names that contain question marks (`?`) or asterisks (`*`) are not added to the server.

To modify a tag, you must add it again with new properties. Whenever you change or copy a tag name, the old and new tag names and time stamps are recorded in the audit trail.

### Returns

- `ihuSTATUS_OK`: Success
- `ihuSTATUS_INVALID_PARAMETER`: Invalid input parameters
- `ihuSTATUS_LIC_TOO_MANY_TAGS`: Added tags exceed the licensed tag count
- `ihuSTATUS_ACCESS_DENIED`: You do not have rights to add tags

## ihuTagDelete

Use the `ihuTagDelete` function to delete tags from the server.

### Prototype

```
ihuTagDelete {
  in long serverhandle,
  in MSO Char * tagname,
};
```

### Remarks

This function uses `serverhandle` and `tagname` as parameters.

### Returns

- `ihuSTATUS_OK`: Success
- `ihuSTATUS_INVALID_PARAMETER`: Invalid input parameters
- `ihuSTATUS_ACCESS_DENIED`: You do not have rights to delete tags

## ihuTagDeleteEx

You can use the `ihuTagDeleteEx` function to permanently delete a tag by passing `TRUE` as a parameter.

### Prototype

```
ihuTagDeleteEx {
  in long serverhandle,
  in MSO Char * tagname,
  BOOL DeletePermanent
};
```

### Remarks

Use `DeletePermanent` to create a new tag with a previously used name. The `ihuTagDeleteEx` function uses the following parameters:

- `serverhandle`: The Historian server from which to fetch tags
- `tagname`: Name of the tag to be deleted, which must be returned in a tag browse
- `DeletePermanent`: Permanently deletes a tag when set to `TRUE`

**Note:**
After you permanently delete a tag, you can no longer query its data, and the tag name is available for reuse.

### Returns

- `ihuSTATUS_OK`: Success
- `ihuSTATUS_INVALID_PARAMETER`: Invalid input parameters
- `ihuSTATUS_ACCESS_DENIED`: You do not have rights to delete tags

**Note:**
You must be a member of the ihTag Admin security group to delete tags.

## ihuTagRename

Use the `ihuTagRename` function to rename tags.

### Prototype

```
ihuTagRename {
  in long serverhandle,
  in MSO Char * OldTagName,
  in MSO Char * NewTagName,
};
```

### Remarks

This function uses the following parameters:

- `serverhandle`: The Historian 4.0 server from which to fetch tags
- `OldTagname`: Name of the tag to be renamed
- `NewTagname`: New tag name

**Note:**
When you use this function to update or modify tag names, the tag properties are not modified. If you modify a renamed tag property, be aware that all the tag properties for the alias are also updated.

### Returns

- `ihuSTATUS_OK`: Success
- `ihuSTATUS_INVALID_PARAMETER`: Invalid input parameters
- `ihuSTATUS_ACCESS_DENIED`: You do not have rights to rename tags

## ihuTagRenameEx

You can use the `ihuTagRenameEx` function to permanently rename a tag by passing `TRUE` as a parameter.

### Prototype

```
ihuTagRenameEx {
  in long serverhandle,
  in MSO Char * OldTagName,
  in MSO Char * NewTagName,
BOOL TrueRename,
};
```

### Remarks

You can permanently rename tags that you do not want to read and write with their previous names. This function uses the following parameters:

- `serverhandle`: The Historian 4.0 server from which to fetch tags
- `OldTagname`: Name of the tag to be renamed
- `NewTagname`: New tag name
- `TrueRename`: Permanently renames a tag when set to `TRUE`.

### Returns

- `ihuSTATUS_OK`: Success
- `ihuSTATUS_INVALID_PARAMETER`: Invalid input parameters
- `ihuSTATUS_ACCESS_DENIED`: You do not have rights to rename tags

**Note:**
You must be a member of the ihTag Admin security group to rename tags.

## ihuTagCacheCriteriaClear

Use the `ihuTagCacheCriteriaClear` function to clear any cached criteria before you set up tag browse criteria.

### Prototype

```
void IHUAPI ihuTagCacheCriteriaClear();

};
```

## ihuTagCacheCriteriaClearEx2

Use the `ihuTagCacheCriteriaClear` function to clear any cached criteria before you set up tag browse criteria.

### Prototype

```
void IHUAPI ihuTagCacheCriteriaClearEx2

(

void * TagCacheContext,

};
```

## ihuTagCacheCriteriaSetStringProperty

Use the `ihuTagCacheCriteriaSetStringProperty` function to set up tag browse criteria before you call `ihuFetchTagCacheEx`.

### Prototype

```
ihuTagCacheCriteriaSetStringProperty {
  in ihuTagProperties TagProperty,
  in MSO Char *Value,
};
```

### Remarks

This function fetches a current set of tags and properties from the Historian server and places them into an API cache for subsequent queries, such as `ihuGetNumericTagPropertyByTagname`.

### Returns

- `ihuSTATUS_OK`
- `ihuSTATUS_INVALID_PARAMETER`
- `ihuSTATUS_INVALID_TAGNAME`

## ihuTagCacheCriteriaSetStringPropertyEx2

Use the `ihuTagCacheCriteriaSetStringPropertyEx2` function to set up tag browse criteria before you call `ihuFetchTagCacheEx`.

### Prototype

```
ihuTagCacheCriteriaSetStringPropertyEx2 {
void * TagCacheContext,
ihuTagProperties TagProperty,
MSO Char *Value,
};
```

### Remarks

This function fetches a current set of tags and properties from the Historian server and places them into an API cache for subsequent queries, such as `ihuGetNumericTagPropertyByTagname`.

### Returns

- `ihuSTATUS_OK`
- `ihuSTATUS_INVALID_PARAMETER`
- `ihuSTATUS_INVALID_TAGNAME`

## ihuTagCacheCriteriaSetNumericProperty

Use the `ihuTagCacheCriteriaSetNumericProperty` function to set up tag browse criteria.

### Prototype

```
ihuTagCacheCriteriaSetNumericProperty {
  in ihuTagProperties TagProperty,
  in double Value,
};
```

### Returns

- `ihuSTATUS_OK`
- `ihuSTATUS_INVALID_PARAMETER`
- `ihuSTATUS_INVALID_TAGNAME`

## ihuTagCacheCriteriaSetNumericPropertyEx2

Use the `ihuTagCacheCriteriaSetNumericPropertyEx2` function to set up tag browse criteria.

### Prototype

```
ihuTagCacheCriteriaSetNumericPropertyEx2 {
void * TagCacheContext,
ihuTagProperties TagProperty,
double Value,
};
```

### Returns

- `ihuSTATUS_OK`
- `ihuSTATUS_INVALID_PARAMETER`
- `ihuSTATUS_INVALID_TAGNAME`

## ihuTagClearProperties

Use the `ihuTagClearProperties` function to clear tag properties before you add tags to a server.

### Prototype

```
void IHUAPI ihuTagClearProperties()
};
```

## ihuTagSetStringProperty

Use the `ihuTagSetStringProperty` function to set string tag properties before you call `ihuTagAdd`.

### Prototype

```
ihuTagSetStringProperty {
  in ihuTagProperties TagProperty,
  in MSO Char *Value,
};
```

### Returns

- `ihuSTATUS_OK`

## ihuTagSetNumericProperty

Use the `ihuTagSetNumericProperty` function to set numeric tag properties before you call `ihuTagAdd`.

### Prototype

```
ihuTagSetNumericProperty {
  in ihuTagProperties TagProperty,
  in double.Value,
};
```

### Returns

- `ihuSTATUS_OK`

## ihuCloseTagCache

Use the `ihuCloseTagCache` function to free the API cache from memory. Be sure to close the tag cache when not in use to free memory and prevent accidental usage.

### Prototype

```
ihuCloseTagCache {

};
```

### Returns

- `ihuSTATUS_OK` in all cases

## ihuCloseTagCacheEx2

Use the `ihuCloseTagCacheEx2` function to free the API cache from memory. Be sure to close the tag cache when not in use to free memory and prevent accidental usage.

### Prototype

```
ihuCloseTagCache {

void * TagCacheContext

};
```

### Returns

- `ihuSTATUS_OK` in all cases

# Write Functions

## Write Functions Overview

The write functions are designed for high-performance data write access to the archiver. Groups of data samples are built up in the application and then sent as a whole into the API for transmission to the server. Write errors can be returned to the application, which can then implement its own error handling. Or, the program can write without a wait if there is no error handling.

The application is responsible for all error handling, including retrying failed writes.

### Write Functions

- ihuWriteData *(on page 1543)*
- ihuWriteComment *(on page 1546)*

## ihuWriteData

Use the high-performance `ihuWriteData` function to write multiple samples for multiple tags.

## Prototype

```
ihuWriteData {

  in long serverhandle,

  in int number_of_samples,

  in IHU_DATA_SAMPLE *data_values,

  in ihuErrorCode *error_returns ,

  in bool wait_for_reply,

  in bool error_on_replace

};
```

**Important:**
You cannot write more than 100,000 samples in each call to this function. It is recommended that you write 1,000 to 10,000 samples per call.

## Remarks

The Historian archiver has the following strict data-write rules that apply to all applications and collectors:

1. No application is allowed to write data with a time stamp older than now minus the "Data is read only after" active hours setting. Such writes fail.
2. No application is allowed to write data with a time stamp before the start time of the first archive, even if the write satisfies the previous rule.
3. No application can write data to a read-only archive.
4. No application can write data with a time stamp more than 15 minutes in the future.

The write function has two Boolean parameters to control optional behavior:

- `wait_for_reply`: When set to `TRUE`, write operations are blocked until the values are acknowledged by the archiver. This allows you to check the error returns to reach a degree of confidence that the data was written successfully. You cannot check error codes unless you wait for the reply.

  If it is important to return control to your program so that collection can be performed, consider setting `wait_for_reply=FALSE`. You might also want to set `wait_for_reply=FALSE` if you have no specific error-handling strategy, or if you are using data readbacks to verify write success.

  If you are using store and forward in the User API, set `wait_for_reply=FALSE`.

> **Note:**
> If you set `wait_for_reply=FALSE`, you must pass a `NULL` error array for the `ihuErrorCode *error_returns` value.

- `error_on_replace`: This parameter has no effect on performance. Use it to specify how the archiver behaves when an existing archived sample has the same time stamp as a new sample being written. Set `error_on_replace=TRUE` to discard the new sample and return an error that you can see when `wait_ for_reply=TRUE`. Set `error_on_replace=FALSE` to overwrite the existing sample with the new sample.

### Time Stamps

You can use this function to specify a time stamp or pass a time stamp of 0 seconds to use the current system time on the written sample. Pass a time stamp structure where `seconds=0`, and do not pass `NULL`. All samples in the group with a 0 time stamp specified use the system time at the time of the write as a time stamp. Values are in microseconds.

### Values

Values to be written are not required to have matching data types as the Historian tags. The archiver converts written data types to tag data types if needed.

### Qualities

You are required to specify the data quality and subquality for the sample when you use this function. There are no default values.

> **Note:**
> Although the `IHU_DATA_SAMPLE` function has a field for comments, you cannot call it to write a comment. Use the ihuWriteComment *(on page 1546)* function instead.

### Error Handling

The User API does not provide store and forward or retry functionality for failed writes. That is the responsibility of the application. You can write User API programs that perform retries or use store and forward.

If you get a timeout back from a write, the write may be waiting in a write queue on the data archiver.

### Security

Historian has audited and unaudited write security groups and a security administration group. To perform writes, you must be a member of one of these groups. Otherwise, you get an `ihuSTATUS_ACCESS_DENIED` error.

### Returns

This function returns `ihuSTATUS_OK` when values are successfully written.

Errors are returned on timeouts, when you are not a member of the necessary security group, or when tags are not found.

> **Important:**
> The User API passes through any timestamps it is given without adjusting for time offsets. The application is responsible to account for time differences between clients and archivers running on different machines.

## ihuWriteComment

Use the `ihuWriteComment` function to write a single comment to a single data sample.

### Prototype

```
ihuWriteComment {
  in long serverhandle,
  in MSO Char *Tagname,
  in IHU_timestamp *time stamp,
  in MSO Char *Comment,
  in MSO Char *SuppliedUser,
  in MSO Char *SuppliedPassword
};
```

### Remarks

Comments can be written with any time stamp that is valid for a data write. See ihuWriteDatafunction *(on page 1543)* for details on valid time stamps. If there is an existing raw sample for the specified tag and time stamp, the comment is attached with no loss of data values stored with that sample. Otherwise, a new raw sample with no value is created to hold the comment.

If a time stamp with 0 seconds is passed in, the current time is used.

Currently, comments written with the User API must be string text.

The `SuppliedUser` and `SuppliedPassword` parameters are optional and can be `NULL`. If set, values must be a valid username and password with write permissions, or the comment write fails.

### Returns

- `ihuSTATUS_OK` on success, error otherwise.

# Query Modifiers Functions

## Query Modifiers Functions Overview

You can use query modifier functions to specify various ways to retrieve data from Historian. For example, you can use `ONLYGOOD` to request only good-quality data, or `INCLUDEREPLACED` to retrieve replaced values.

### Query Modifier Functions


- ihuBrowseQueryModifiers *(on page 1547)*
- ihuClearQueryModifiers *(on page 1548)*
- ihuRetrieveCalculatedDataEx2 *(on page 1548)*
- ihuSetQueryModifiers *(on page 1549)*


## ihuBrowseQueryModifiers

Use the `ihuBrowseQueryModifiers` function to return a list of all supported query modifiers from the Historian Server. Various versions of the Historian Data Archiver may support different sets of modifiers.

### Prototype

```
ihuBrowseQueryModifiers (
  long serverhandle,
  IHU_QUERY_MODIFIER **QueryModifiers,
  int *NumberOfModifiers
);
```

### Remarks

The `NumberOfModifiers` parameter returns a total count of modifiers. The `QueryModifiers` parameter returns a list of supported query modifiers.

### Returns

- `ihuSTATUS_OK`
- `ihuSTATUS_INVALID_PARAMETER`

## ihuClearQueryModifiers

Use the `ihuClearQueryModifiers` function to clear any previously set query modifiers so that they are not used in subsequent reads.

### Prototype

```
ihuErrorCode ihuClearQueryModifiers(void)
```

### Remarks

Use `ihuSetQueryModifiers()` to set a modifier string to be used in all reads. Call `ihuClearQueryModifiers()` to stop using that modifier.

### Returns

- `ihuSTATUS_OK`
- `ihuSTATUS_OUT_OF_MEMORY`

## ihuRetrieveCalculatedDataEx2

Use the `ihuRetrieveCalculatedDataEx2` function with the `ihuStateCount` and `ihuStateTime` calculation modes to return calculated data based on the raw samples stored in the archive.

You can request data by specifying a number of samples or an interval. Set one to a nonzero value and the other to `0`. To split the duration, divide the time from start to finish into evenly spaced time intervals.

### Prototype

```
ihuRetrieveCalculatedDataEx2

(long serverhandle,

IHU_TIMESTAMP StartTime,

IHU_TIMESTAMP EndTime,

ihuCalculationMode CalculationMode,

ihuDataType StateDataType,

ihuValue StateValue,

unsigned long NumberOfSamples,
```

```
IHU_DATA_INTERVAL Interval,

IHU_RETRIEVED_DATA_RECORDS_EX *DataRecords)
```

### Returns

This function returns `ihuSTATUS_OK` when values are retrieved successfully, and returns the following errors:

- Read timeouts
- User is not a member of the iH Readers security group
- Tag not found

## ihuSetQueryModifiers

Use the `ihuSetQueryModifiers` function to define query modifier criteria for all subsequent data reads.

### Prototype

```
ihuSetQueryModifiers (

long serverhandle,

MSO Char *CriteriaString

)
```

### Returns

- `ihuSTATUS_OK`
- `ihuSTATUS_INVALID_PARAMETER`
- `ihuSTATUS_OUT_OF_MEMORY`

# Read Functions

## Read Functions Overview

You can use read functions to retrieve raw, sampled, and calculated values stored in the Historian server for usage in reporting, plotting, or data analysis applications.

In most cases, multiple tags can be read in a single function call.

Most of the read functions serve the general purpose of retrieving raw, sampled or calculated data. Additionally, targeted functions are exposed to provide easy access to current tag values or values interpolated to a specific date and time stamp.

Comments, if they exist, are returned with retrieved data. See the sample programs for more information.

You can combine read functions with write functions to read back newly written data, which is the most secure way to verify successful data writes.

**Read Functions**


- ihuReadCurrentValue *(on page 1550)*
- ihuReadInterpolatedValue *(on page 1551)*
- ihuReadInterpolatedValueEx *(on page 1552)*
- ihuReadRawDataByTime *(on page 1553)*
- ihuReadRawDataByTimeEx *(on page 1554)*
- ihuReadMultiTagRawDataByCountEx *(on page 1557)*
- ihuReadRawDataByCount *(on page 1555)*
- ihuReadRawDataByCountEx *(on page 1556)*
- ihuReadMultiTagRawDataByCount *(on page 1557)*
- ihuReadMultiTagRawDataByCountEx *(on page 1557)*
- ihuRetrieveSampledData *(on page 1558)*
- ihuRetrieveSampledDataEx *(on page 1559)*
- ihuRetrieveSampledDataEx2 *(on page 1560)*
- ihuRetrieveCalculatedData *(on page 1561)*
- ihuRetrieveCalculatedDataEx *(on page 1562)*
- ihuRetrieveCalculatedDataEx3 *(on page 1563)*


## ihuReadCurrentValue

Use the `ihuReadCurrentValue` function to return the current tag value. The current tag value has a specific definition in Historian, but is generally the last raw sample sent to the archiver for that tag.

### Prototype

```
ihuReadCurrentValue {

  in long serverhandle,

  in int number_of_tags,

  in/out IHU_DATA_SAMPLE *pSamples,

  out ihuErrorCode *error_returns

};
```

### Remarks

You can retrieve current values for multiple tags with one call. A single sample for each requested tag is returned.

For each tag, set only the tagname field in `IHU_DATA_SAMPLE`. All other fields are set by the function.

Data is returned based on the data type of the tag.

### Returns

The `ihuReadCurrentValue` function returns errors in two ways. The function has a return code, and each tag name has an error code. Check function-level errors before you examine tag-level errors.

Both are `ihuSTATUS_OK` when the current value is successfully retrieved.

The `ihuReadCurrentValue` function can return errors on read timeouts or when the user is not a member of the ih Readers security group.

For example, each tag can return an error in cases where tags are not found or have no raw samples.

## ihuReadInterpolatedValue

Use the `ihuReadInterpolatedValue` function to return the value of a tag interpolated to a specified date and time.

### Prototype

```
ihuReadInterpolatedValue {
  in long serverhandle,
  in int number_of_tags,
  in/out IHU_DATA_SAMPLE * pSamples,
  out ihuErrorCode * error_returns
};
```

### Remarks

You can retrieve values for multiple tags with one call. A single sample for each requested tag is returned.

For each tag, set only the tagname and time stamp fields in `IHU_DATA_SAMPLE`. All other fields are set by the function. The time stamp in the first requested tag is used for all tags.

Data is returned based on the data type of the tag.

The data quality returned is either `ihuOPCGood` or `ihuOPCBad`, and the substatus is always `ihuOPCNonspecific`.

You must specify a time stamp in `IHU_DATA_SAMPLE` for the first requested tag, for example:

```
//you only need to populate the time stamp in the first sample
lRet = IHU_timestamp_FromParts(2003,
7,
22,
11,
0,
0,
0,
&(pSamples[0].timestamp));
lRet= ihuReadInterpolatedValue (serverhandle, // handle from connect
number_of_tags, // number of tags, one sample per tag
pSamples, // allocated by caller, value set by API
error_returns); // error code per tag
```

For an example, refer to the ReportLike sample program.

### Returns

The `ihuReadInterpolatedValue` function returns errors in two ways. The function has a return code, and each tag name has an error code. Check function-level errors before you examine tag-level errors.

Both are `ihuSTATUS_OK` when the current value is successfully retrieved.

The `ihuReadInterpolatedValue` function returns errors on read timeouts or when the user is not a member of the ih Readers security group.

For example, each tag can return an error in cases where tags are not found or have no raw samples.

## ihuReadInterpolatedValueEx

Use the `ihuReadInterpolatedValueEx` function to return the number of samples of a tag interpolated to a given date and time with filters and query modifiers by using `pszFilterExpression` and `CriteriaString`.

### Prototype

```
ihuErrorCode IHUAPI ihuReadInterpolatedValueEx
(
```

```
long serverhandle, // [in] used for communication with server

int number_of_tags, // [in] the number of tags to retrieve

IHU_DATA_SAMPLE *pSamples, // [in/out] user fills tagname and timestamp and the API fills other fields

MSO_Char *pszFilterExpression, // [in] Filter Expression e.g. Tag > 20

IHU_FILTER_MODE FilterMode,   // [in] Filter Modes

MSO_Char *CriteriaString, // [in] QueryModifiers to use with data reads

ihuErrorCode *error_returns // [out] populated with per tag error

);
```

### Remarks

To skip filtering, you can pass `NULL` to `pszFilterExpression` and `ihuExactTime` to `FilterMode`, or you can use ihuReadInterpolatedValue() *(on page 1551)*.

Pass `NULL` to `CriteriaString` if you do not use a `QueryModifier`.

You can retrieve values for multiple tags with one call. A single sample for each requested tag is returned.

For each tag, set only the tagname and time stamp fields in `IHU_DATA_SAMPLE`. All other fields are set by the function. The time stamp in the first requested tag is used for all tags.

Data is returned based on the data type of the tag.

The data quality returned is either `ihuOPCGood` or `ihuOPCBad`, and the substatus is always `ihuOPCNonspecific`.

If you want to specify a time stamp, you must do so in `IHU_DATA_SAMPLE` for the first requested tag.

### Returns

The `ihuReadInterpolatedValueEx` function returns errors on:

- Read timeouts
- User is not a member of the ih Readers security group
- Tag not found
- Filter criteria or query modifiers cannot be set

## ihuReadRawDataByTime

Use the `ihuReadRawDataByTime` function to return multiple raw samples for a single tag in a specified time range.

### Prototype

```
ihuReadRawDataByTime {
  in long serverhandle,
  in MSO Char * tagname,
  in IHU_timestamp * StartTime,
  in IHU_timestamp * EndTime,
  out int * number_of_samples,
  out IHU_DATA_SAMPLE **data_values
};
```

### Remarks

The time stamp, value, and quality of each raw sample are returned.

### Returns

The `ihuReadRawDataByTime` function returns `ihuSTATUS_OK` when values are successfully retrieved.

Errors are returned on read timeouts, when the user is not a member of the ih Readers security group, or if the tag is not found.

## ihuReadRawDataByTimeEx

Use the `ihuReadRawDataByTimeEx` function to return multiple raw samples for a single tag in a specified time range with filters and query modifiers by using `pszFilterExpression` and `CriteriaString`.

### Prototype

```
ihuReadRawDataByTimeEx {
long serverhandle, // [in] the serverhandle
MSO_Char *tagname, // [in] the single tagname to fetch data for
MSO_Char *pszFilterExpression, // [in] Filter Expression e.g. Tag > 20
IHU_FILTER_MODE FilterMode,   // [in] Filter Modes
MSO_Char *CriteriaString, // [in] QueryModifiers to use with data reads
IHU_TIMESTAMP * StartTime, // [in] Start time of query
IHU_TIMESTAMP * EndTime, // [in] End Time of query
int * number_of_samples, //[out] the number of samples returned
IHU_DATA_SAMPLE **data_values // [out] the returned data samples, unlimited number
};
```

### Remarks

To skip filtering, you can pass `NULL` to `pszFilterExpression` and `ihuExactTime` to `FilterMode`, or you can use `ihuReadRawDataByTime()`.

Pass `NULL` to `CriteriaString` if you do not use a `QueryModifier`.

The time stamp, value, and quality of each raw sample are returned.

### Returns

The `ihuReadRawDataByTimeEx` function returns `ihuSTATUS_OK` on success, or returns errors on:

- Read timeouts
- User is not a member of the ih Readers security group
- Tag not found
- Filter criteria or query modifiers cannot be set

## ihuReadRawDataByCount

Use the `ihuReadRawDataByCount` function to return the requested number of raw samples for a single tag going forward or backward in time from a specified time stamp. In cases where the number of samples available is less than the value specified in the `number_of_samples` parameter, the function stops returning data.

### Prototype

```
ihuReadRawDataByCount {
  in long serverhandle,
  in MSO Char *tagname,
  in IHU_timestamp * StartTime,
  in/out int * number_of_samples,
  in int TimeForward,
  out IHU_DATA_SAMPLE **data_values
};
```

### Remarks

The time stamp, value, and quality of each raw sample are returned.

### Returns

The `ihuReadRawDataByCount` function returns `ihuSTATUS_OK` when values are successfully retrieved.

Errors are returned on read timeouts, when the user is not a member of the ih Readers security group, or if the tag is not found.

## ihuReadRawDataByCountEx

Use the `ihuReadRawDataByCountEx` function to return the requested number of raw samples for a single tag going forward or backward in time from a specified time stamp by using `pszFilterExpression` and `CriteriaString`. In cases where the number of samples available is less than the value specified in the `number_of_samples` parameter, the function stops returning data.

### Prototype

```
ihuReadRawDataByCountEx {
long serverhandle, // [in] the serverhandle
MSO_Char *tagname, // [in] the single tagname to fetch data for
MSO_Char *pszFilterExpression, // [in] Filter Expression e.g. Tag > 20
IHU_FILTER_MODE FilterMode,   // [in] Filter Modes
MSO_Char *CriteriaString, // [in] QueryModifiers to use with data reads
IHU_TIMESTAMP * StartTime, // [in] Start time of query
IHU_TIMESTAMP * EndTime, // [in] End time of query
int * number_of_samples, // [in/out] the number of samples to return and actually returned
int TimeForward, // [in] TRUE if search should be in forward time order
IHU_DATA_SAMPLE **data_values // [out] the returned data samples, unlimited number
};
```

### Remarks

To skip filtering, you can pass `NULL` to `pszFilterExpression` and `ihuExactTime` to `FilterMode`, or you can use ihuReadRawDataByCount() *(on page 1555)*.

Pass `NULL` to `CriteriaString` if you do not use a `QueryModifier`.

The time stamp, value, and quality of each raw sample are returned.

### Returns

The `ihuReadRawDataByCountEx` function returns `ihuSTATUS_OK` when values are successfully retrieved, and returns errors on:

- Read timeouts
- User is not a member of the ih Readers security group
- Tag not found
- Filter criteria or query modifiers cannot be set

## ihuReadMultiTagRawDataByCount

Use the `ihuReadMultiTagRawDataByCount` function to return up to the requested number of raw samples for multiple tags.

### Prototype

```
ihuErrorCode IHUAPI ihuReadMultiTagRawDataByCount (

long serverhandle,

int number_of_tags,

IHU_ TIMESTAMP *StartTime,

int *number_of_samples, BOOL TimeForward,

ihuErrorCode **error_returns, IHU_RETRIEVED_RAW_VALUES *pSamples

);
```

### Remarks

The `ihuReadMultiTagRawDataByCount` function returns per-tag and overall errors.

### Returns

The `ihuReadMultitagRawDataByCount` function returns `ihuSTATUS_OK` when values are successfully retrieved.

Errors are returned on read timeouts, when the user is not a member of the ih Readers security group, or if the tag is not found.

## ihuReadMultiTagRawDataByCountEx

Use the `ihuReadMultiTagRawDataByCountEx` function to return up to a requested number of multiple raw samples for multiple tags with filters and query modifiers by using `pszFilterExpression` and `CriteriaString`.

### Prototype

```
ihuErrorCode IHUAPI ihuReadMultiTagRawDataByCountEx (

long serverhandle, // [in] the serverhandle

int number_of_tags, // [in] the number of tags to retrieve
```

```
IHU_TIMESTAMP * StartTime, // [in] Start time of query

IHU_TIMESTAMP * EndTime, // [in] End time of query

MSO_Char *pszFilterExpression, // [in] Filter Expression e.g. Tag > 20

MSO_Char *CriteriaString, // [in] QueryModifiers to use with data reads

IHU_FILTER_MODE FilterMode,   // [in] Filter Modes

int * number_of_samples, // [in/out] the number of samples to return and actually returned

BOOL TimeForward, // [in] TRUE if search should be in forward time order ihuErrorCode ** error_returns, // [out]

 populated with per tag error

IHU_RETRIEVED_RAW_VALUES *pSamples // [in/out] the returned data samples, unlimited number memory is allocated

);
```

### Remarks

To skip filtering, you can pass `NULL` to `pszFilterExpression` and `ihuExactTime` to `FilterMode`, or you can use ihuReadMultiTagRawDataByCount() *(on page 1557)*.

Pass `NULL` to `CriteriaString` if you do not use a `QueryModifier`.

The `ihuReadMultiTagRawDataByCount` function returns per-tag and overall errors.

### Returns

The `ihuReadRawDataByCountEx` function returns `ihuSTATUS_OK` when values are successfully retrieved, and returns errors on:

- Read timeouts
- User is not a member of the ih Readers security group
- Tag not found
- Filter criteria or query modifiers cannot be set

## ihuRetrieveSampledData

Use the `ihuRetrieveSampledData` function to return sampled data based on the raw samples stored in the archive. `Interpolated`, `Trend`, and `Lab` are example `SamplingMode` values.

### Prototype

```
ihuRetrieveSampledData {

  in long serverhandle,

  in IHU_timestamp StartTime,

  in IHU_timestamp EndTime,
```

```
  in ihuSamplingMode SamplingMode,
  in unsigned long NumberOfSamples,
  in unsigned long IntervalMilliseconds,
  in/out IHU_RETRIEVED_DATA_RECORDS *DataRecords
};
```

### Remarks

To request data, you can specify a number of samples or a time interval. Set one parameter to a non-zero value and the other to `0`. To split the duration, divide the time from start to finish into evenly spaced time intervals.

### Returns

The `ihuRetrieveSampledData` function returns `ihuSTATUS_OK` when values are successfully retrieved.

Errors are returned on read timeouts, when the user is not a member of the ih Readers security group, or if the tag is not found.

## ihuRetrieveSampledDataEx

Use the `ihuRetrieveSampledDataEx` function to return sampled data based on the raw samples stored in the archive. `Interpolated`, `Trend`, and `Lab` are example `SamplingMode` values. Use this function with hybrid sampling modes such as `TrendToRaw` or `InterpolatedToRaw` to return the sampling mode that was actually used.

### Prototype

```
ihuRetrieveSampledDataEx {
long serverhandle, // [in] which server to fetch from
IHU_TIMESTAMP StartTime, // [in] Start time for the query
IHU_TIMESTAMP EndTime, // [in] End time for the query
ihuSamplingMode SamplingMode, // [in] The requested sampling mode
unsigned long NumberOfSamples, // [in] 0 or num samples to return
IHU_DATA_INTERVAL Interval, // [in] 0 or sampling interval in units provided by IntervalType
IHU_RETRIEVED_DATA_RECORDS_EX *DataRecords // [in/out] - caller fills in tagnames of the structures and API will
};
```

### Remarks

To request data, you can specify a number of samples or a time interval. Set one parameter to a non-zero value and the other to `0`. To split the duration, divide the time from start to finish into evenly spaced time intervals.

If you are not using hybrid sampling modes, use ihuRetrieveSampledData() *(on page 1558)*.

### Returns

The `ihuReadRawDataByTimeEx` function returns `ihuSTATUS_OK` when values are successfully retrieved, and returns errors on:

- Read timeouts
- User is not a member of the ih Readers security group
- Tag not found

## ihuRetrieveSampledDataEx2

Use the `ihuRetrieveSampledDataEx2` function to return sampled data based on the raw samples stored in the archive with filters and query modifiers by using `pszFilterExpression` and `CriteriaString`. `Interpolated`, `Trend`, and `Lab` are example `SamplingMode` values. Use this function with hybrid sampling modes such as `TrendToRaw` or `InterpolatedToRaw` to return the sampling mode that was actually used.

### Prototype

```
ihuRetrieveSampledDataEx2 {

long serverhandle, // [in] which server to fetch from

IHU_TIMESTAMP StartTime, //[in] Start time for the query

IHU_TIMESTAMP EndTime, //[in] End time for the query

ihuSamplingMode SamplingMode, //Sampling Modes

MSO_Char *pszFilterExpression, // [in] Filter Expression e.g. Tag > 20

IHU_FILTER_MODE FilterMode,   // [in] Filter Modes

MSO_Char *CriteriaString, // [in] QueryModifiers to use with data reads

unsigned long NumberOfSamples, //[in] 0 or number of samples to return

IHU_DATA_INTERVAL Interval, // [in] 0 or sampling interval in units provided by IntervalType

IHU_RETRIEVED_DATA_RECORDS_EX *DataRecords //[in/out] - you fill in tagnames of the structures and API will fil

};
```

### Remarks

To request data, you can specify a number of samples or a time interval. Set one parameter to a non-zero value and the other to `0`. To split the duration, divide the time from start to finish into evenly spaced time intervals.

To skip filtering, you can pass `NULL` to `pszFilterExpression` and `ihuExactTime` to `FilterMode`, or you can use ihuRetrieveSampledData() *(on page 1558)*.

Pass `NULL` to `CriteriaString` if you do not use a `QueryModifier`.

### Returns

The `ihuRetrieveSampledDataEx2` function returns `ihuSTATUS_OK` when values are successfully retrieved, and returns errors on:

- Read timeouts
- User is not a member of the ih Readers security group
- Tag not found
- Filter criteria or query modifiers cannot be set

## ihuRetrieveCalculatedData

Use the `ihuRetrieveCalculatedData` function to return calculated data based on the raw samples stored in the archive. `Average`, `Minimum`, and `Count` are example `CalculationMode` values.

### Prototype

```
ihuRetrieveCalculatedData {
  in long serverhandle,
  in IHU_timestamp StartTime,
  in IHU_timestamp EndTime,
  in ihuCalculationMode CalculationMode,
  in unsigned long NumberOfSamples,
  in unsigned long IntervalMilliseconds,
  in/out IHU_RETRIEVED_DATA_RECORDS *DataRecords
};
```

### Remarks

To request data, you can specify a number of samples or a time interval. Set one parameter to a non-zero value and the other to `0`. To split the duration, divide the time from start to finish into evenly spaced time intervals.

### Returns

The `ihuRetrieveCalculatedData` function returns `ihuSTATUS_OK` when values are successfully retrieved.

Errors are returned on read timeouts, when the user is not a member of the ih Readers security group, or if the tag is not found.

## ihuRetrieveCalculatedDataEx

Use the `ihuRetrieveCalculatedDataEx` function to return calculated data based on the raw samples stored in the archive. `Average`, `Minimum`, and `Count` are example `CalculationMode` values.

### Prototype

```
ihuRetrieveCalculatedDataEx {
long serverhandle, // [in] which server to fetch from
IHU_TIMESTAMP StartTime, //[in] Start time of query
IHU_TIMESTAMP EndTime, //[in] End time of query
ihuCalculationMode CalculationMode, //[in] Calculation Mode
unsigned long NumberOfSamples, //[in] Number of samples to be returned
IHU_DATA_INTERVAL Interval, // [in] Interval in Milliseconds
IHU_RETRIEVED_DATA_RECORDS_EX *DataRecords //[in/out] - you fill in tagnames of the structures and API will fil
};
```

### Remarks

To request data, you can specify a number of samples or a time interval. Set one parameter to a non-zero value and the other to `0`. To split the duration, divide the time from start to finish into evenly spaced time intervals.

### Returns

The `ihuReadRawDataByTimeEx2` function returns `ihuSTATUS_OK` when values are successfully retrieved, and returns errors on:

- Read timeouts
- User is not a member of the ih Readers security group
- Tag not found

## ihuRetrieveCalculatedDataEx3

Use the `ihuRetrieveCalculatedDataEx3` function to return sampled data based on the raw samples stored in the archive with filters and query modifiers by using `pszFilterExpression` and `CriteriaString`. `Average`, `Minimum`, and `Count` are example `CalculationMode` values.

### Prototype

```
ihuRetrieveCalculatedDataEx3 {

long serverhandle, // [in] which server to fetch from

IHU_TIMESTAMP StartTime, // [in] Start time of query

IHU_TIMESTAMP EndTime, // [in] End time of query

ihuCalculationMode CalculationMode, //[in] Calculation Mode

MSO_Char *pszFilterExpression, // [in] Filter Expression e.g. Tag > 20

IHU_FILTER_MODE FilterMode,   // [in] Filter Modes

MSO_Char *CriteriaString, // [in] QueryModifiers to use with data reads

ihuDataType StateDataType, // DataType of the StateValue to compare for ihuStateCount calculation mode

ihuValue StateValue, // Value to compare for ihuStateCount calculation mode unsigned long NumberOfSamples,

 IHU_DATA_INTERVAL Interval, // [in] Interval in Milliseconds

IHU_RETRIEVED_DATA_RECORDS_EX *DataRecords // [in/out] - you fill in tagnames of the structures and API will fil

};
```

### Remarks

To request data, you can specify a number of samples or a time interval. Set one parameter to a non-zero value and the other to `0`. To split the duration, divide the time from start to finish into evenly spaced time intervals.

To skip filtering, you can pass `NULL` to `pszFilterExpression` and `ihuExactTime` to `FilterMode`, or you can use ihuRetrieveCalculatedData() *(on page 1561)*.

Pass `NULL` to `CriteriaString` if you do not use a `QueryModifier`.

### Returns

The `ihuRetrieveCalculatedDataEx3` function returns `ihuSTATUS_OK` when values are successfully retrieved, and returns errors on:

- Read timeouts
- User is not a member of the ih Readers security group
- Tag not found
- Filter criteria or query modifiers cannot be set

# Utility Functions

## Utility Functions Overview

Utility functions ease programming. Time stamp conversion functions are necessary to produce the time stamps needed in most API functions.

### Utility Functions


- IHU_timestamp_FromParts *(on page 1564)*
- IHU_timestamp_ToParts *(on page 1565)*
- ihuServerGetTime *(on page 1566)*


## IHU_timestamp_FromParts

Use the `IHU_timestamp_FromParts` function to use supplied time parts (date, hour, minutes, seconds, and so on) to produce a UTC time stamp in the format needed by the Historian User API read and write calls.

### Prototype

```
IHU_timestamp_FromParts {
  in int Year,
  in int Month,
  in int Day,
  in int Hour,
  in int Minute,
  in int Second,
  in long Subsecond,
  out IHU_timestamp *time stamp
};
```

### Remarks

The time parts passed in are assumed to be in the local time zone of the machine where you make the call.

During UTC conversion, the **Use Daylight Saving Time** setting of your local machine is used.

### Returns

The `IHU_timestamp_FromParts` function returns `ihuSTATUS_OK` on success, or an error code on failure.

## IHU_timestamp_ToParts

Use the `IHU_timestamp_ToParts` function to convert UTC time stamps to the following human-readable parts:

- `Year`
- `Month`
- `Date`
- `Hour`
- `Minute`
- `Second`
- `Subsecond`

You can use this function to convert the time stamps of retrieved data samples.

### Prototype

```
IHU_timestamp_ToParts {
  in IHU_timestamp time stamp
  out int *Year,
  out int *Month,
  out int *Day,
  out int *Hour,
  out int *Minute,
  out int *Second,
  out long *Subsecond,
};
```

### Remarks

The time parts produced are in the local time zone of the machine where you make the call.

During conversion from UTC, the **Use Daylight Saving Time** setting of your local machine is used.

### Returns

The `IHU_timestamp_ToParts` function returns `ihuSTATUS_OK` on success, or an error code on failure.

## ihuServerGetTime

Use the `ihuServerGetTime` function to return the current time on the Historian server.

### Prototype

```
ihuErrorCode IHUAPI ihuServerGetTime
(
    long serverhandle,
    IHU_TIMESTAMP *CurrentTime
)
```

### Remarks

The `ihuServerGetTime` function has the following inputs and outputs:

- `hServer`: Server handle from the connection
- `CurrentTime`: Current time on the server in the local time zone

### Returns

The `ihuServerGetTime` function returns `ihuSTATUS_OK` on success, or an error code on failure.

# Enumerated Sets Functions

## Enumerated Sets Functions Overview

You can use the enumerated sets functions to create, modify, delete, and browse enumerated sets, and assign tags to those sets.

When you query tag data by using these functions, the string state name value is returned by default. To retrieve the numeric value, use `ihuEnumeratedSetRawValue`.

When you write to a tag by using these functions, write the numeric value.

Use these functions to retrieve all the enumerated sets that match the `setname` mask criteria. If you want to retrieve all sets, pass `*` as the mask.

### Enumerated Sets Functions


- ihuGetEnumeratedSets *(on page 1567)*
- ihuEnumeratedSetAdd *(on page 1567)*
- ihuEnumeratedSetRawValue *(on page 1568)*
- ihuEnumeratedSetsFree *(on page 1569)*
- ihuEnumeratedSetRename *(on page 1570)*
- ihuEnumeratedSetDelete *(on page 1570)*
- ihuEnumeratedStateAdd *(on page 1571)*
- ihuEnumeratedStateModify *(on page 1572)*
- ihuEnumeratedStateDelete *(on page 1572)*


## ihuGetEnumeratedSets

Use the `ihuGetEnumeratedSets` function to retrieve all the enumerated sets that match the `setname` mask criteria. To retrieve all sets, pass `*` as the mask.

### Prototype

```
ihuErrorCode IHUAPI ihuGetEnumeratedSets (

in long serverhandle,

in MSO Char *EnumeratedSetMask,

out long *numberofrecords,

out ihuEnumeratedSetRecordSet

*EnumeratedSetRecordSet  )
```

### Remarks

The `ihuGetEnumeratedSets` function returns the number of enumerated set records, plus the enumerated sets.

### Returns

The `ihuGetEnumeratedSets` function returns `ihuSTATUS_OK` on success, or errors if invalid input parameters are found or if out-of-memory issues occur during dynamic memory allocation.

## ihuEnumeratedSetAdd

Use the `ihuEnumeratedSetAdd` function to add enumerated sets. You can specify sets that have states, or add states later on.

### Prototype

```
ihuEnumeratedSetAdd
( in long serverhandle,
in ihuEnumeratedSetProperties
*EnumeratedSet  )
```

### Remarks

Any existing set with a name that matches the specified set name is overwritten by the new set.

### Returns

The `ihuEnumeratedSetAdd` function returns `ihuSTATUS_OK` on success, or returns errors on:

- Invalid input parameters
- Out-of-memory issues during dynamic memory allocation
- User not a member of ihTag Admin group

**Note:**
Set the `Administrator` parameter to the ihTag Admin security group to assign the highest precedence to this parameter.

## ihuEnumeratedSetRawValue

Use the `ihuEnumeratedSetRawValue` function to specify whether to return string or numeric values returned when reading tags by using enumerated sets. By default, string values are retrieved for tags associated with enumerated sets. To retrieve numeric values, set `SetRawValue` to `TRUE`. If string values are unavailable, raw values are retrieved.

### Prototype

```
ihuEnumeratedSetRawValue (
in long serverhandle,
in BOOL SetRawValue
)
```

### Remarks

Override the default value of `FALSE` only to return numeric values. The value persists until the function is called again or your program exits.

### Returns

The `ihuEnumeratedSetRawValue` function returns `ihuSTATUS_OK` on success, or returns errors on:

- Invalid input parameters
- User not a member of ihTag Admin group

**Note:**
Set the `Administrator` parameter to the ihTag Admin security group to assign the highest precedence to this parameter.

## ihuEnumeratedSetsFree

Use the `ihuEnumeratedSetsFree` function to clear the enumerated set buffers after you call `ihuGetEnumeratedSets()`.

### Prototype

```
ihuEnumeratedSetsFree (
in long serverhandle,
ihuEnumeratedSetRecordSet
in *EnumeratedSetRecordSet )
```

### Remarks

When you use the `ihuEnumeratedSetsFree` function, memory is cleared from the enumerated set buffers.

### Returns

The `ihuEnumeratedSetsFree` function returns `ihuSTATUS_OK` on success, or returns errors on:

- Invalid input parameters
- User not a member of ihTag Admin group

**Note:**
Set the `Administrator` parameter to the ihTag Admin security group to assign the highest precedence to this parameter.

## ihuEnumeratedSetRename

Use the `ihuEnumeratedSetRename` function to rename an enumerated set.

### Prototype

```
ihuEnumeratedSetRename (
in long serverhandle,
in MSO Char *oldEnumeratedSetName,
in MSO Char *newEnumeratedSetName )
```

### Remarks

You must be an administrator of a set to rename it.

### Returns

The `ihuEnumeratedSetRename` function returns `ihuSTATUS_OK` on success, or returns errors on:

- Invalid input parameters
- Out-of-memory issues during dynamic memory allocation
- No set name that matches `OldEnumeratedSetName` value
- User not a member of ihTag Admin group

**Note:**
Set the `Administrator` parameter to the ihTag Admin security group to assign the highest precedence to this parameter.

## ihuEnumeratedSetDelete

Use the `ihuEnumeratedSetDelete` function to delete an enumerated set.

### Prototype

```
ihuEnumeratedSetDelete (
in long serverhandle,
in MSO Char *EnumeratedSetName )
```

### Remarks

You must be an administrator of a set to delete it.

### Returns

The `ihuEnumeratedSetDelete` function returns `ihuSTATUS_OK` on success, or returns errors on:

- Invalid input parameters
- Out-of-memory issues during dynamic memory allocation
- No set name that matches `EnumeratedSetName` value
- User not a member of ihTag Admin group

**Note:**
Set the `Administrator` parameter to the ihTag Admin security group to assign the highest precedence to this parameter.

## ihuEnumeratedStateAdd

Use the `ihuEnumeratedStateAdd` function to add a state to an existing enumerated set.

### Prototype

```
ihuEnumeratedStateAdd (

in long serverhandle,

in MSO Char *EnumSetName,

in ihuEnumeratedSetState *EnumState )
```

### Remarks

When you add a state to an existing enumerated set, the data type of the new state matches the data type of other states in the set.

Any existing state with a name that matches the specified state name is overwritten by the new state.

### Returns

The `ihuEnumeratedStateAdd` function returns `ihuSTATUS_OK` on success, or returns errors on:

- Invalid input parameters
- Out-of-memory issues during dynamic memory allocation
- No set name that matches `EnumSetName` value
- User not a member of ihTag Admin group

**Note:**
Set the `Administrator` parameter to the ihTag Admin security group to assign the highest precedence to this parameter.

## ihuEnumeratedStateModify

Use the `ihuEnumeratedStateModify` function to modify a state in an enumerated set.

### Prototype

```
ihuEnumeratedStateModify (
  in long serverhandle,
  in MSO Char *EnumeratedSetName,
  in MSO Char *EnumeratedStateName,
  ihuEnumeratedSetState
  in *EnumStateToModify )
```

### Remarks

You can modify state values and names, but not data types. To change a state name, pass the old and new names as string values inside the `EnumStateToModify` parameter.

### Returns

The `ihuEnumeratedStateModify` function returns `ihuSTATUS_OK` on success, or returns errors on:

- Invalid input parameters
- No set name that matches `EnumeratedSetName` value
- No state name that matches `EnumeratedStateName` value
- User not a member of ihTag Admin group

**Note:**
Set the `Administrator` parameter to the ihTag Admin security group to assign the highest precedence to this parameter.

## ihuEnumeratedStateDelete

Use the `ihuEnumeratedStateDelete` function to delete a state from an enumerated set.

### Prototype

```
ihuEnumeratedStateDelete

( in long serverhandle,

in MSO Char *EnumeratedSetName,

in MSO Char *EnumeratedStateName

)
```

### Remarks

When only one state is available in a set, you cannot delete the state. The only way to delete the state is to delete the set. There is no need to delete each state before you delete a set.

You must be an administrator of a set to delete its associated states.

### Returns

The `ihuEnumeratedStateDelete` function returns `ihuSTATUS_OK` on success, or returns errors on:

- Invalid input parameters
- No set name that matches `EnumeratedSetName` value
- No state name that matches `EnumeratedStateName` value
- Out-of-memory issues during dynamic memory allocation
- User not a member of ihTag Admin group

**Note:**
Set the `Administrator` parameter to the ihTag Admin security group to assign the highest precedence to this parameter.

# User-Defined Type Functions

## User-Defined Type Functions Overview

You can use the user-defined type functions to create, modify, delete, and browse user-defined data types that contain multiple fields. You must add user-defined data types to the Data Archiver before you can use them in tags.

### User-Defined Type Functions


- ihuUserDefinedTypeAdd *(on page 1574)*
- ihuUserDefinedTypeDelete *(on page 1574)*
- ihuUserDefinedTypeRename *(on page 1575)*
- ihuUserDefinedTypeExists *(on page 1576)*
- ihuGetUserDefinedTypes *(on page 1576)*
- ihuUserDefinedTypeSetProperties *(on page 1577)*
- ihuUserDefinedTypeFreeProperties *(on page 1577)*


## ihuUserDefinedTypeAdd

Use the `ihuUserDefineTypeAdd` function to create or add `MultiField` type tags. The user-defined type you pass to this function can have multiple fields defined, or you can add fields later by using ihUserDefinedTypeSetProperties() *(on page 1577)*.

You can also use this function to modify user-defined data types. To modify an existing type, add a user-defined data type with the same name.

### Prototype

```
ihuUserDefinedTypeAdd

(

long serverhandle, // [in] connected server handle

ihuUserDefinedTypeProperties *InUserDefinedType //[in] UserDefined type that needs to be added / modified

)
```

### Remarks

Existing types are overwritten by new types with the same names. You must be a member of the ihTag Admin group to add or modify a type.

### Returns

- `ihuSTATUS_OK`: on success
- `ihuSTATUS_INVALID_PARAMETER`: `NULL` value or invalid server handle
- `ihuSTATUS_FAILED`: on any error

## ihuUserDefinedTypeDelete

Use the `ihuUserDefinedTypeDelete` function to delete a user-defined type that you no longer use.

### Prototype

```
ihuUserDefinedTypeDelete

(

long serverhandle, // [in] connected server handle

MSO_Char * UserDefinedTypeName //[in] UserDefined type name to be deleted

)
```

### Remarks

You must be a member of the ihTag Admin group to delete a type.

### Returns

- `ihuSTATUS_OK`: on success
- `ihuSTATUS_INVALID_PARAMETER`: `NULL` value or invalid server handle
- `ihuSTATUS_FAILED`: on any error

## ihuUserDefinedTypeRename

Use the `ihuUserDefinedTypeRename` function to rename a user-defined type.

### Prototype

```
ihuUserDefinedTypeRename

(

long serverhandle, // [in] connected server handle

MSO_Char * UserDefinedTypeName, //[in] UserDefined type name which needs to be renamed

MSO_Char *NewUserDefinedTypeName //[in] new UserDefined type name

)
```

### Remarks

You must be a member of the ihTag Admin group to rename a type.

### Returns

- `ihuSTATUS_OK`: on success
- `ihuSTATUS_INVALID_PARAMETER`: `NULL` value or invalid server handle
- `ihuSTATUS_FAILED`: on any error

## ihuUserDefinedTypeExists

Use the `ihuUserDefinedTypeExists` function to check whether a specific user-defined type exists.

### Prototype

```
ihuUserDefinedTypeExists
(
long serverhandle, // [in] connected server handle
MSO_Char * UserDefinedTypeName // [in] UserDefined type name
)
```

### Returns

- `ihuSTATUS_OK`: if the type exists
- `ihuSTATUS_INVALID_PARAMETER`: `NULL` value or invalid server handle
- `ihuSTATUS_FAILED`: if the type does not exist or the server cannot be reached

## ihuGetUserDefinedTypes

Use the `ihuGetUserDefinedTypes` function to return a list of user-defined types and their values. After you call this function, you must call `ihuUserDefinedTypeFreeProperties()` to release memory allocations.

### Prototype

```
ihuGetUserDefinedTypes
(
long serverhandle, // [in] connected server handle
MSO_Char *StructSetMask, //[in] Pass * to get the all the Structure or pass the name of the structure you want
long *numberofrecords, //[out] Number of records
ihuUserDefinedTypeRecordSet *RecordSet //[out] returns the records for the UserDefined defined types
)
```

### Remarks

You must be a member of the ihReader group to get a list of user-defined types. The list is returned in the `RecordSet` parameter.

### Returns

- `ihuSTATUS_OK`: on success
- `ihuSTATUS_INVALID_PARAMETER`: `NULL` value passed for `StructSetMask` or `RecordSet` or `numberofrecords`
- `ihuSTATUS_FAILED`: on any error

## ihuUserDefinedTypeSetProperties

Use the `ihuUserDefinedTypeSetProperties` function to set user-defined type properties after you add types by using ihuUserDefinedTypeAdd() *(on page 1574)*.

### Prototype

```
ihuUserDefinedTypeSetProperties

(

long serverhandle, // [in] connected server handle

ihuUserDefinedTypeProperties *UserDefinedType //[in] UserDefined type that needs to be added / created

)
```

### Remarks

You must be a member of the ihTag Admin group to set type properties.

### Returns

- `ihuSTATUS_OK`: on success
- `ihuSTATUS_INVALID_PARAMETER`: invalid server handle or `NULL` property pointer value passed
- `ihuSTATUS_FAILED`: on any error

## ihuUserDefinedTypeFreeProperties

Use the `ihuUserDefinedTypeFreeProperties` function to free the memory allocated by the ihuGetUserDefinedTypes() *(on page 1576)* function.

### Prototype

```
ihuUserDefinedTypeFreeProperties

(

ihuUserDefinedTypeRecordSet *UserDefinedTypeRecordSet //[in] UserDefinedTypeRecordSet that needs to be fr

)
```

### Remarks

Call the `ihuUserDefinedTypeFreeProperties` function once to release all memory in the recordset returned by `ihuGetUserDefinedTypes()`. Do not free the memory in your program.

### Returns

Void

# Publish Functions

## Publish Functions Overview

You can use the publish functions to manage the data forwarding of local Historian server tags to one or more remote destination servers by using the Server-to-Server Distributors. When you publish a tag, you initiate continuous transmission of its historical data values to a destination server.

### Publish Functions


- `ihuPublishAddTag` *(on page 1578)*
- `ihuPublishRemoveTag` *(on page 1579)*
- `ihuPublishTagCloseCache` *(on page 1579)*
- `ihuPublishGetTagPropertiesToCache` *(on page 1580)*
- `ihuPublishTagGetNumericPropertyByTagname` *(on page 1581)*
- `ihuPublishTagGetNumericPropertyByIndex` *(on page 1581)*
- `ihuPublishTagGetStringPropertyByTagname` *(on page 1582)*
- `ihuPublishTagGetStringPropertyByIndex` *(on page 1582)*
- `ihuPublishSetTagProperties` *(on page 1583)*
- `ihuPublishTagSetNumericProperty` *(on page 1584)*
- `ihuPublishTagSetStringProperty` *(on page 1584)*
- `ihuPublishTagClearProperties` *(on page 1585)*
- `ihuPublishGetDestinationServer` *(on page 1585)*
- `ihuPublishSetDestinationServer` *(on page 1586)*


## ihuPublishAddTag

Use the `ihuPublishAddTag` function to publish a tag to a destination server. When you publish, a tag is created in the destination server.

### Prototype

```
ihuPublishAddTag (
long hSourceServer,
MSO Char *InterfaceName,
MSO Char *SourceTagname,);
```

### Remarks

After you publish a tag, any new data for the `SourceTagname` is written to the `DestinationTagname`.

The tag is added to the destination server with default parameters that you can modify by using ihuPublishGetTagPropertiesToCache *(on page 1580)* and ihuPublishSetTagProperties *(on page 1583)*.

### Returns

The `ihuPublishAddTag` function returns `ihuSTATUS_OK` on success, or an error if the tag was not added to the destination server.

## ihuPublishRemoveTag

Use the `ihuPublishRemoveTag` function to prevent tag publishing by a specified collector.

### Prototype

```
ihuPublishRemoveTag
( long hServer,
  MSO Char *DestinationTagname,
  MSO Char *SourceTagname );
```

### Remarks

The `ihuPublishRemoveTag` function does not remove the tag from the source server but marks it as deleted in the destination server. To completely stop data collection, you must disable collection.

### Returns

The `ihuPublishRemoveTag` function returns `ihuSTATUS_OK` on success.

## ihuPublishTagCloseCache

Use the `ihuPublishTagCloseCache` function to close a cache returned by the `ihuPublishGetTagPropertiesToCache` function.

### Prototype

```
ihuPublishTagCloseCache

(void);
```

### Remarks

To avoid memory leaks, close the cache.

### Returns

The `ihuPublishTagCloseCache` function returns `ihuSTATUS_OK` on success.

## ihuPublishGetTagPropertiesToCache

Use the `ihuPublishGetTagPropertiesToCache` function to retrieve a list of tags published by the specified Server-to-Server Distributor. Be sure to match tags with the exact tag name or a name with a specified wildcard mask.

**Note:**
A separate published tag properties cache exists for each process thread. To avoid confusion and unexpected results, ensure that all cache-related function calls are invoked on the same thread. When threads are mixed, two or more distinct caches are referenced, which enables concurrent access to multiple different cached results at the cost of additional memory.

### Prototype

```
ihuPublishGetTagPropertiesToCache

( long hServer,

  MSO Char *DistributorName,

  MSO Char *DestinationTagnameOrMask,

  out int *NumTagsFound

);
```

### Remarks

The `ihuPublishGetTagPropertiesToCache` function returns the number of tags found in the `NumTagsFound` parameter.

Use this function to retrieve tags before you get and set string or numeric tag properties by using the `ihuPublishTagGetNumericPropertyByTagname` or `ihuPublishTagSetStringProperty` and similar functions.

### Returns

The `ihuPublishGetTagPropertiesToCache` function returns `ihuSTATUS_OK` on success, even if no tags are found. The function returns errors if any out-of-memory issues occur during dynamic memory allocation.

## ihuPublishTagGetNumericPropertyByTagname

Use the `ihuPublishTagGetNumericPropertyByTagname` function to retrieve a numeric tag property, such as the deadband for a tag. The tag is identified by name within the cache previously populated by ihuPublishGetTagPropertiesToCache *(on page 1580)*.

### Prototype

```
ihuPublishTagGetNumericPropertyByTagname
( MSO Char *Tagname,
ihuTagProperties TagProperty,
*Value );
```

### Remarks

The `ihuPublishTagGetNumericPropertyByTagname` function returns the numeric tag property value.

### Returns

The `ihuPublishTagGetNumericPropertyByTagname` function returns `ihuSTATUS_OK` on success, or returns an error if any out-of-memory issues occur during dynamic memory allocation.

## ihuPublishTagGetNumericPropertyByIndex

Use the `ihuPublishTagGetNumericPropertyByIndex` function to retrieve a single numeric tag property from the cache.

### Prototype

```
ihuPublishTagGetNumericPropertyByIndex
(
  int Index,
  ihuTagProperties TagProperty,
  double *Value );
```

### Remarks

The `ihuPublishTagGetNumericPropertyByIndex` function returns the numeric tag property retrieved. The index is zero-based.

### Returns

The `ihuPublishTagGetNumericPropertyByIndex` function returns `ihuSTATUS_OK` on success, or returns an error if any out-of-memory issues occur during dynamic memory allocation.

## ihuPublishTagGetStringPropertyByTagname

Use the `ihuPublishTagGetStringPropertyByTagname` function to retrieve a single string tag property by tagname from the cache.

### Prototype

```
ihuPublishTagGetStringPropertyByTagname
(
  MSO Char *Tagname,
  ihuTagProperties TagProperty,
  MSO Char *Value,
  int ValueLength );
```

### Remarks

The `ihuPublishTagGetStringPropertyByTagname` function returns a single string tag property by tagname from the cache.

### Returns

The `ihuPublishTagGetStringPropertyByTagname` function returns `ihuSTATUS_OK` on success, or returns an error if any out-of-memory issues occur during dynamic memory allocation.

## ihuPublishTagGetStringPropertyByIndex

Use the `ihuPublishTagGetStringPropertyByIndex` function to retrieve a single string tag property from the cache. The index is zero-based.

### Prototype

```
ihuPublishTagGetStringPropertyByIndex (
  int Index,
```

```
  ihuTagProperties TagProperty, MSO Char *Value,
  int ValueLength );
```

### Remarks

The `ihuPublishTagGetStringPropertyByIndex` function returns the string tag property from the cache. The index is zero-based.

### Returns

The `ihuPublishTagGetStringPropertyByIndex` function returns `ihuSTATUS_OK` on success, or returns an error if any out-of-memory issues occur during dynamic memory allocation.

## ihuPublishSetTagProperties

Use the `ihuPublishSetTagProperties` function to define properties such as the deadband for a published tag. Before you call this function, set properties by using `ihuPublishTagSetStringProperty` or `ihuPublishTagSetNumericProperty`.

A separate collection of tag properties is available for each process thread, similar to the published tag properties cache. To avoid confusion, be sure to invoke this function on the same thread as `ihuPublishTagSetNumericProperty`, `ihuPublishTagSetStringProperty`, and `ihuPublishTagClearProperties`.

### Prototype

```
ihuPublishSetTagProperties (
  long hServer,
  MSO Char *InterfaceName,
  MSO Char *DestinationTagname,
);
```

### Remarks

After you add a tag by using the `ihuPublishAddTag` function, use the `ihuPublishSetTagProperties` function to set tag properties.

### Returns

The `ihuPublishSetTagProperties` function returns `ihuSTATUS_OK` on success, or returns an error if any out-of-memory issues occur during dynamic memory allocation.

## ihuPublishTagSetNumericProperty

Use the `ihuPublishTagSetNumericProperty` function to define a numeric tag property before you invoke ihuPublishSetTagProperties *(on page 1583)*.

### Prototype

```
ihuPublishTagSetNumericProperty
(
  ihuTagProperties TagProperty,
  double Value
);
```

### Remarks

The `ihuPublishTagSetNumericProperty` function returns the numeric tag property.

### Returns

The `ihuPublishTagSetNumericProperty` function returns `ihuSTATUS_OK` on success, or errors on invalid input parameters or if any out-of-memory issues occur during dynamic memory allocation.

## ihuPublishTagSetStringProperty

Use the `ihuPublishTagSetStringProperty` function to define a string tag property before you invoke ihuPublishSetTagProperties *(on page 1583)*.

### Prototype

```
ihuPublishTagSetStringProperty (
  ihuTagProperties TagProperty,
  MSO Char *Value
);
```

### Remarks

The `ihuPublishTagSetStringProperty` function returns the string tag property.

### Returns

The `ihuPublishTagSetStringProperty` function returns `ihuSTATUS_OK` on success, or errors on invalid input parameters or if any out-of-memory issues occur during dynamic memory allocation.

## ihuPublishTagClearProperties

Use the `ihuPublishTagClearProperties` function to clear any previously defined tag properties.

### Prototype

```
ihuPublishTagClearProperties
(
  void
);
```

### Remarks

The `ihuPublishTagClearProperties` function clears previously defined tag properties.

### Returns

The `ihuPublishTagClearProperties` function returns `ihuSTATUS_OK` on success, or returns an error if any out-of-memory issues occur during dynamic memory allocation.

## ihuPublishGetDestinationServer

Use the `ihuPublishGetDestinationServer` function to retrieve the name of the destination Historian server configured for a specific distributor.

### Prototype

```
ihuPublishGetDestinationServer (
long hServer,
MSO Char *InterfaceName,
MSO Char **DestinationServer );
```

### Remarks

The `ihuPublishGetDestinationServer` function returns the destination Historian server name.

### Returns

The `ihuPublishGetDestinationServer` function returns `ihuSTATUS_OK` on success, or returns an error if any out-of-memory issues occur during dynamic memory allocation.

## ihuPublishSetDestinationServer

Use the `ihuPublishSetDestinationServer` function to define the destination Historian server for a specific collector.

### Prototype

```
ihuPublishSetDestinationServer (
  long hServer,
  MSO Char *InterfaceName,
  MSO Char *DestinationServer
);
```

### Remarks

The `ihuPublishSetDestinationServer` function returns the Historian server name that is defined for the specified collector. The new name is assigned without a collector restart, and can be used at any time the distributor is restarted.

### Returns

The `ihuPublishSetDestinationServer` function returns `ihuSTATUS_OK` on success, or returns an error if any out-of-memory issues occur during dynamic memory allocation.

# Historian User API Error Codes

## Error Codes

You might encounter the following error codes while you use the Historian User API:

**Table 267. Historian API Error Codes**

| Code | Message | Description |
|---|---|---|
| `100` | `ihuSTATUS_FAILED` | Generic failure. |
| `101` | `ihuSTATUS_API_TIMEOUT` | Server machine name not found, or server found but archiver service not running. |
| `102` | `ihuSTATUS_NOT_CONNECTED` | Not currently connected to a Historian server. |
| `103` | `ihuSTATUS_INTERFACE_-NOT_FOUND` | Interface not found. |

**Table 267. Historian API Error Codes (continued)**

| Code | Message | Description |
|---|---|---|
| 104 | ihuSTATUS_NOT_SUPPORTED | Reserved. |
| 105 | ihuSTATUS_DUPLICATE_DATA | ihuWriteData was called with error_on_replace = TRUE and the supplied data would have overwritten existing data. |
| 106 | ihuSTATUS_DUPLICATE_DATA | Server found, but invalid username or password. |
| 107 | ihuSTATUS_ACCESS_DENIED | Access denied by the Historian server. Check username and password or security group membership. |
| 108 | ihuSTATUS_WRITE_IN_FUTURE | Write time stamp is too far in the future. |
| 109 | ihuSTATUS_WRITE_ARCH_OFFLINE | There is no archiver to hold the write time stamp. |
| 110 | ihuSTATUS_ARCH_READONLY | The destination archive to hold the write time stamp is marked as read-only. |
| 111 | ihuSTATUS_WRITE_OUTSIDE_ACTIVE | The write time stamp is before the active hours (now − "data is read only after") setting. |
| 112 | ihuSTATUS_WRITE_NO_ARCH_AVAILABLE | No archive is available to hold the write time stamp. |
| 113 | ihuSTATUS_INVALID_TAGNAME | Tagname used is not valid. Tagname does not exist in the Historian server. |
| 114 | ihuSTATUS_LIC_TOO_MANY_TAGS | Exceeded tag license count on the server. |
| 115 | ihuSTATUS_LIC_TOO_MANY_USERS | Exceeded user license count on the server. |
| 116 | ihuSTATUS_LIC_INVALID_LIC_DLL | An invalid license DLL is installed. |
| 117 | ihuSTATUS_NO_VALUE | No value has been passed to the function. |
| 118 | ihuSTATUS_NOT_LICENSED | Your Historian installation is not licensed. |

**Table 267. Historian API Error Codes (continued)**

| Code | Message | Description |
|---|---|---|
| 119 | ihuSTATUS_CALC_CIRC_-REFERENCE | Reserved. |
| 120 | ihuSTATUS_DUPLICATE_IN-TERFACE | Reserved. |
| 121 | ihuSTATUS_BACKUP_EX-CEEDED_SPACE | Reserved. |
| 122 | ihuSTATUS_INVALID_SERV-ER_VERSION | You are attempting to use this API on an invalid version of Historian. |
| 123 | ihuSTATUS_DATA_RETRIE-VAL_COUNT_EXCEEDED | You requested too many samples in one read request. |
| 124 | ihuSTATUS_INVALID_PARA-METER | Generic failure when an invalid value is passed into the User API. |

# Historian User API Sample Programs

## Sample Programs Overview

Sample programs are provided with the Historian User API to demonstrate how to perform common tasks. The following sample programs are supplied with the User API:

- `CollectorLike.c`: Shows an example of a program that writes thousands of samples per second to Historian tags.
- `CollectorLikeSAF.c`: Shows an example of callbacks and store and forward in the User API.
- `PlotLike.cpp`: Shows an example of a program that retrieves calculated and interpolated data suitable for plotting.
- `ReportLike.cpp`: Shows how to retrieve raw data samples from Historian.
- `MigrationLike.c`: Shows how to transfer legacy historical data into Historian.
- `QueryModifiers.c`: Shows how to retrieve data using query modifiers.
- `Blobdatatype.c`: Shows how to create tags of blob (binary object) data type and read and write data.
- `ByTag`: Shows how to read and write data for a tag by using its tag ID instead of its tag name.

No guidance is provided on how to convert applications from other products or APIs, including the Historian SDK.

To work with these samples, you must be familiar with Historian features and functionality. Refer to the Historian product documentation. This is especially important if you are using security groups with Historian, since the applications that call into the User API are limited by the security access granted at the server level.

### Compiling the Samples

Only a release build configuration is provided. A debug configuration is not provided because the release mode configuration produces a program database and has compiler optimizations turned off, so that you can easily step through the sample code in the debugger.

Review the sample program source code for additional details.

**Note:**
The sample programs provided in this help document are for review only. Compilable source code is included in the Historian User API installation directory.

### CollectorLike

This sample demonstrates how to read string and numeric tag properties and write groups of data samples to multiple tags. Data is sent to the archiver until a key is pressed. Sample code is provided for writing data of various data types and data qualities. Tips are included for achieving optimal performance. This sample program also demonstrates how to iterate through all archiver tags to find tags that belong to the collector.

### PlotLike

This sample program demonstrates how to read numeric tag properties and read evenly spaced sampled or calculated data suitable for plotting. Percent good data quality indicators are described. Any comments retrieved are printed to the program window. Finally, an example is provided for how to write a comment in cases where your trending application allows users to enter comments.

### ReportLike

This sample program demonstrates how raw samples are retrieved, one tag per call, by timespan and by number of samples. Comments are retrieved if they exist. The retrieved samples can be used in custom quality calculations or to implement calculation modes not available in Historian. An example of how to

calculate a raw maximum is provided. The program has a loop that fetches the current value of a tag until a key is pressed.

### MigrationLike

This sample program demonstrates how to migrate data from a legacy system into Historian. Tips are provided for the most efficient sequence to use to send data to minimize archive disk space and migration time. This sample program also demonstrates how to add tags to the Historian server.

# Chapter 11. Historian SDK

## Object Model Overview

### Historian SDK Overview

The Historian Software Development Kit (SDK) is a COM object designed to simplify access to Historian services and data for the purposes of application development.

This object provides the following functionality to developers:

- Browsing Available Historian Servers
- Browsing and Configuring Tags
- Browsing, Adding, and Modifying Data
- Browsing, Adding, and Modifying Alarms and Events
- Browsing and Adding Messages and Alerts
- Controlling the Archiving Functions of the System
- Controlling the Collection and Interface Functions of the System
- Adherence to Historian Security Constraints

If you need to create customized programming for the Historian server, use the Historian Software Development Kit (SDK) with Visual Basic or any application that provides a VBA programming interface, such as iFIX, Microsoft Excel or Microsoft Word. After you install Client Tools, the Historian SDK is available in the `System32` folder is automatically registered. To use the SDK, set up a project reference with the Historian SDK.

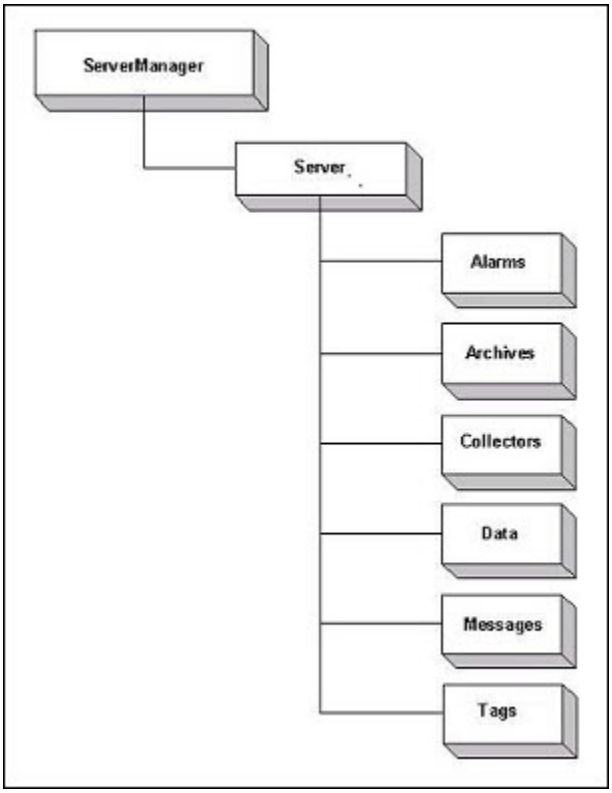
You can use the Historian SDK under any Win32 platform using a development language that supports Microsoft COM/DCOM. The SDK relies on the run-time version of the Historian API. You must install the Historian API prior to the installation of the SDK on any client that accesses the Historian system through the SDK. For instructions, refer to installing Client Tools *(on page 150)*.

The Historian SDK is a COM DL(ihSDK.dll) that must be instantiated prior to use. A single instance of the SDK may connect and converse with many Historian servers simultaneously. You can create multiple instances of the SDK, however, the developer must maintain this collection and respond to events from the appropriate instance.

The following table lists general information regarding SDK naming and dependencies.

| SDK DLL | ihSDK.dll |
|---|---|
| Class Name | iHistorian_SDK |
| Version | 5.0.0.x |
| Dependencies | ihAPI50.dll |

The following diagram describes the object model employed by the Historian SDK. For more information on the individual objects, refer to the SDK Object Reference.




## Working with Comments

Comments are retrieved with a data query. To retrieve comments, request all fields from the DataValue object and then perform a DataRecordset.QueryRecordset. The comments will be contained in the DataValue.Comments collection. Comments are stored to the archive using the DataRecordset.WriteRecordset method. Store comments to the DataValue object first by calling the DataValue.AddComment method.

### Adding Data

Use the DataValue object to insert data by setting the value, quality, and timestamp before calling the WriteRecordset method.

### Sample SDK Program

A sample program for the SDK is included in the Samples\SDK folder when you install Historian. Refer to this sample for more detailed examples than the ones that appear in this Help system.

**Note:**
The SDK sample is designed to work on Visual Basic 6 (VB6).

## Connect the SDK to the Server

After instantiation of the SDK, connect to one or more servers by providing a valid user name and password for the target Historian server domain. The SDK does not support any other functions until you make a successful connection and authentication. The authenticated user also controls which functions may be performed through the SDK and potentially what data may be accessed.

The following example displays the Visual Basic code for instantiation of the SDK and connection to the target server.

```
Dim MyServer as Object

'Instantiate The SDK

Set MyServer = CreateObject("iHistorian_SDK.Server")


'Attempt Connection

If Not MyServer.Connect("USGB014", "Fred", "000") Then

Err.Raise 1, "Failed To Connect To USGB014" End If


In the example above, "MyServer" does not receive events from the Historian server.

In order to receive events, Dim WithEvents MyServer As iHistorian_SDK.Server

'Instantiate The SDK

Set MyServer = New iHistorian_SDK.Server


'Attempt Connection

If Not MyServer.Connect("USGB014", "Fred", "000") Then

Err.Raise 1, "Failed To Connect To USGB014" End If
```

**Note:**
After each session, disconnect each Server connection prior to exiting the application.

## Working with Blob Data

Historian is capable of storing many different data types, such as Floating Point, Integer, String, Binary, and BLOB (undetermined binary data type, such as an Excel spreadsheet, a PDF file, or a Word file). The source of the data defines the ability of Historian to collect specific data types.

The following example demonstrates how to read and write a file into Historian. It contains a sample script for adding, writing, and reading a tag of BLOB data type. You will need to change the server name, folder, and file names as appropriate.

```
Dim MyServer As Server


Private Function Connect1() As Boolean
If Not MyServer Is Nothing Then
MyServer.Disconnect
End If
Set MyServer = Nothing

Set MyServer = New Server

Connect1 = MyServer.Connect("FRIEDENTHAL")
End Function

Private Function AddTag(Tagname As String) As Boolean
Dim Tags As TagRecordset
Dim NewTag As Tag

Set Tags = MyServer.Tags.NewRecordset
Set NewTag = Tags.Add(Tagname)
NewTag.DataType = Blob
AddTag = Tags.WriteRecordset


End Function


Private Function WriteBlob(Tagname As String, TimeStamp As Date, ByteArray() As Byte, FileName As String) As Boolean
```

```
Dim Data As DataRecordset
Dim NewValue As DataValue

Set Data = MyServer.Data.NewRecordset
Set NewValue = Data.Add(Tagname, TimeStamp)
NewValue.Value = ByteArray
NewValue.DataQuality = Good
'Store the file name as a comment
NewValue.AddComment FileName
WriteBlob = Data.WriteRecordset

End Function

Private Function ReadBlob(Tagname As String, TimeStamp As Date, ByteArray() As Byte, FileName As String) As Boolean
Dim Data As DataRecordset
Dim NewValue As DataValue

Set Data = MyServer.Data.NewRecordset
With Data.Criteria
.Tagmask = Tagname
.StartTime = DateAdd("s", -1, TimeStamp)
.EndTime = DateAdd("s", 1, TimeStamp)
.SamplingMode = RawByTime
End With
With Data.Fields
.AllFields
End With

ReadBlob = Data.QueryRecordset
ByteArray = Data.Item(1).Item(1).Value
FileName = Data.Item(1).Item(1).Comments.Item(1).Comment

End Function

Function ReadFile(FileName$, fileDirectory$) As Variant
Dim ByteArray() As Byte
Dim FileLen As Long
```

```
Dim MyByte As Byte

Dim i As Long

Dim fName As String

fName = fileDirectory + FileName

Open fName For Binary Access Read As #1 ' Open file for reading.

FileLen = LOF(1) - 1

ReDim ByteArray(FileLen)

For i = 0 To FileLen

Get #1, , MyByte

ByteArray(i) = MyByte

Next

Close #1

ReadFile = ByteArray


End Function

Function WriteFile(fName, ByteArray() As Byte)


Dim FileLen As Long

Dim i As Long

FileLen = UBound(ByteArray)

Open fName For Binary Access Write As #1

For i = 0 To FileLen - 1

Put #1, , ByteArray(i)

Next

Close #1


End Function


Private Sub RunTest()

Dim ByteArray() As Byte

Dim i As Integer

Dim TheTime As Date

Dim FileName As String

FileName = "iHvbs.log"

Const ReadFileDirectory = "C:\"

Const WriteFileDirectory = "C:\Temp\"

ReDim ByteArray(0 To 9)
```

```
If Not Connect1() Then

MsgBox "Did Not Connect"
Exit Sub
End If

If Not AddTag("TestBlob2") Then
MsgBox "Did Not Add Tag"
Exit Sub
End If
'Read the input file
ByteArray = ReadFile(FileName, ReadFileDirectory)

TheTime = Now
'Write the file to Historian
If Not WriteBlob("TestBlob2", TheTime, ByteArray, FileName) Then
MsgBox "Did Not Write Blob"
Exit Sub
End If

Erase ByteArray
FileName = ""
'Read back the file from Historian
If Not ReadBlob("TestBlob2", TheTime, ByteArray, FileName) Then
MsgBox "Did Not Read Blob"
Exit Sub
End If
FileName = WriteFileDirectory + FileName 'copy file to the write directory
WriteFile FileName, ByteArray
End Sub

Private Sub CommandButton1_Click()
End Sub
```

## Working with Archives

You can backup, restore, and create archives using the SDK. To restore an archive, you add an existing archive file to the archives collection using the Add method.

**Example**

The following code shows an example of how to restore an archive.

```
Dim myarchives As iHistorian_SDK.Archives

Dim myarchive As iHistorian_SDK.Archive

Dim i As Long

Set myarcives = DefaultServer.Archives

Set myarchive = myarchives.Add("MY_SERVER_archive001", "d:\program

 files\Historian\Archives\MY_SERVER_archive001_Backup.IHA", 100)

If myarchive Is Nothing Then

MsgBox "An Error Occured Trying To Restore The Archive." + Chr(10) + Chr(10) + _

"The Details Of The Error Follow:" + Chr(10) + DefaultServer.Archives.LastError, vbCritical, "Historian"

Err.Raise 1,, "Error Restoring Archive: " + DefaultServer.Archives.LastError

Else

MsgBox "success"

End If
```

# SDK Reference

## Object Summary

This section contains the Historian objects that are available in the Historian Software Development Kit.

## Alarms Object

You can use Historian to store data for alarms and events. From the SDK, you can add and query these alarms and events. This class is slightly different from other SDK classes in that you will mostly be accessing lower level functionality to add and query alarms and events. By following the directions below, you should be able to perform these tasks fairly easily.

**Add Historian Type Library to the Project**

As previously mentioned, access to methods for alarms and events is lower level. You must add the Historian COM 1.1 Type library to your project. In Visual Basic, in the Project menu, select References, and then add the library. You can now access the lower level alarms and events methods.

### Query by Alarms and Events

For instructions on querying alarms and events, see the documentation for the AlarmRecordSet function.

### Add Alarms and Events

You can add alarms and events to Historian by using the AlarmInfo object. In general, adding alarms or events is easy; declare a new AlarmInfo object, fill it up with the required details, and then run the Add function on the AlarmInfo object. In practice, you must be aware of the lifecycle of your alarm.

### Create a new AlarmInfo Object

To create a new AlarmInfo object, use the CreateObject method as follows:

```
Dim myAlarmInfo As AlarmInfo

               Set myAlarmInfo = CreateObject("Historian_API.AlarmInfo")
```

### Alarm or Event?

Historian distinguishes between alarms and events. Alarms follow a lifecycle as described below, while events are generally one-shot deals. Example events include Set Point Events, Login Events, or other audit trail events. Alarms are generally characterized by a tag going into and out of an abnormal condition. You must identify your AlarmInfo object as an alarm or event by setting the AlarmType field.

```
Alarm:myAlarmInfo.AlarmType = ihALARM_CONDITION

                   Event: myAlarmInfo.AlarmType = ihALARM_TRACKING
```

### Alarm Life Cycle

As previously mentioned, Alarms generally follow a lifecycle. To avoid bad quality alarms in the archive, when adding alarms, ensure that you follow the lifecycle rules below.

**Note:**
For each lifecycle phase, you must create and add a new AlarmInfo object. Or, if you prefer to use the same AlarmInfo object for each lifecycle phase, you can use the ClearAlarmInfo object after you have added it.

```
myAlarmInfo.AddMyServer myAlarmInfo.Clear
```

### New Alarm

To instantiate a new alarm, specify the start time in the AlarmInfo object.

```
myAlarmInfo.StartTime= Now
```

**State Change**

If the alarm changes states (from HI to HIHI for example), you must specify only the new subcondition (along with the other required fields mentioned below).You can optionally specify the starttime field as the original start time (when the alarm first went to HI), but it is not mandatory.

> **Important:**
> Do not specify a new start time. If you do so, a new alarm will be created instead.

```
myAlarmInfo.SubConditionName= "HIHI"
```

**Wrong:** `'myAlarmInfo.StartTime= Now`

**OK:** `'myAlarmInfo.StartTime=AlarmStartTime`

**Acknowledge an Alarm**

To acknowledge an alarm, set the Acked field in the AlarmInfo to True, and also specify the time of the acknowledgement by populating the AckTime alarm field. In addition, you must populate the starttime field with the start time of the alarm (when it first went into an alarm condition, not the last state change).

```
myAlarmInfo.Acked= TRUE
                myAlarmInfo.AckTime = Now
                myAlarmInfo.StartTime = AlarmStartTime
```

**Return to Normal**

When your alarm condition has ended, specify the endtime in the AlarmInfo Object. Similar to the state change and acknowledgements, you can optionally specify the start time of the alarm in the starttime alarm field.

```
myAlarmInfo.EndTime= Now myAlarmInfo.StartTime = AlarmStartTime
```

**Associating Alarms and Events with Tag Data**

The easiest way to associate alarms and events with tag data is to specify the TagName in the AlarmInfo. Historian will sort out everything behind the scenes. However, if you are adding alarms before the associated tag actually exist in Historian, this avenue will not work. You must populate the ItemId field with the source address of the future tag, and populate the DataSourceName field with the collector name of the tag. When the tag is added to the system, the alarms will be correctly linked.

```
myAlarmInfo.Tagname = TheTagName
```

Or

```
myAlarmInfo.DataSourceName = MyCollector.Name myAlarmInfo.ItemId = TheSourceAddress
```

**AlarmInfoFields**

The following table lists the AlarmInfo fields. The fields that are marked as required must be filled in for every call to add. In addition, populate the correct fields based on the current state in the lifecycle of your alarm (see above).

| FieldName | Data Type | Required for Alarms or Events? | Description |
|---|---|---|---|
| AlarmType | Long | Both | Classifies this AlarmInfo as an alarm or an event. Enter 1 for an event and 4 for an alarm. |
| ItemID | String | None | The ItemID of the alarm or event. This contains the source address of the data access tag that the alarm is associated with. This can be NULL if the alarm is not associated with a tag. |
| Source | String | Both | This is the unique identifier used for the alarm or event. |
| DataSourceName | String | Both | The collector interface name associated with the alarm or event. |
| Tagname | String | None | The Historian Tag Name associated with the alarm. |
| EventCategory | String | None | The event category of the alarm or event. |
| ConditionName | String | Alarms | The condition of the alarm. This does not apply to event data. This, combined with the source, comprises an alarm. |

| FieldName | Data Type | Required for Alarms or Events? | Description |
|---|---|---|---|
| SubConditionName | String | Alarms | The sub-condition of the alarm. This does not apply to event data. This is the state of the alarm. |
| StartTime | Date | None | The start time or time stamp of the alarm or event. |
| EndTime | Date | None | The end time of the alarm. This does not apply to event data. |
| AckTime | Date | None | The time the alarm was acknowledged. This does not apply to event data. |
| Timestamp | Date | Both | The time stamp of the alarm or event. |
| Message | String | None | The message attached to the alarm or event. |
| Acked | Boolean | None | Stores the acknowledgement status of the alarm. If the alarm is acknowledged, this will be set to TRUE. |
| Severity | Long | None | The severity of the alarm or event. This is stored as an integer value with a range of 1-1000. |
| Actor | String | | The operator who acknowledged the alarm, or caused the tracking event. |
| Quality | Long | Alarms | The quality of the alarm or event. 0 for bad, 3 for good. |

**Example**

```
Dim MyServer As iHistorian_SDK.Server

Set MyServer = GetServer

Dim myAlarmInfo As AlarmInfo

Set myAlarmInfo = CreateObject("iHistorian_API.AlarmInfo")

Dim AlarmStartTime As Date

AlarmStartTime = Now
```

```
' New Alarm myAlarmInfo.AlarmType = ihALARM_CONDITION

myAlarmInfo.ConditionName = "SampleCondition"

myAlarmInfo.DataSourceName = "SampleDataSource"

myAlarmInfo.Message = "Sample Alarm"

myAlarmInfo.Source = "SampleSource"

myAlarmInfo.StartTime = AlarmStartTime

myAlarmInfo.SubConditionName = "HI"

myAlarmInfo.Timestamp = Now

myAlarmInfo.Quality = 3

myAlarmInfo.Add MyServer

myAlarmInfo.Clear

' State Change myAlarmInfo.AlarmType = ihALARM_CONDITION

myAlarmInfo.ConditionName = "SampleCondition"

myAlarmInfo.DataSourceName = "SampleDataSource"

myAlarmInfo.Message = "Sample Alarm State II"

myAlarmInfo.Source = "SampleSource"

' Not required, but pointing out the start time of the alarm doesn't hurt

myAlarmInfo.StartTime = AlarmStartTime

myAlarmInfo.SubConditionName = "HIHI"

myAlarmInfo.Timestamp = Now

myAlarmInfo.Quality = 3

myAlarmInfo.Add MyServer

myAlarmInfo.Clear

' Ack the HIHI state

myAlarmInfo.Acked = True

myAlarmInfo.AckTime = Now

myAlarmInfo.AlarmType = ihALARM_CONDITION

myAlarmInfo.ConditionName = "SampleCondition"

myAlarmInfo.DataSourceName = "SampleDataSource"

myAlarmInfo.Source = "SampleSource"

myAlarmInfo.StartTime = AlarmStartTime

myAlarmInfo.SubConditionName = "HIHI"

myAlarmInfo.Timestamp = Now

myAlarmInfo.Quality = 3

myAlarmInfo.Add MyServer myAlarmInfo.Clear

' Return to Normal

myAlarmInfo.AlarmType = ihALARM_CONDITION
```

```
myAlarmInfo.ConditionName = "SampleCondition"

myAlarmInfo.DataSourceName = "SampleDataSource"

myAlarmInfo.EndTime = Now

myAlarmInfo.Source = "SampleSource"

' Not required, but pointing out the start time of the alarm doesn't hurt

myAlarmInfo.StartTime = AlarmStartTime

myAlarmInfo.Timestamp = Now

myAlarmInfo.Quality = 3

myAlarmInfo.Add MyServer

myAlarmInfo.Clear
```

## Archive Object

The Archive object contains the configuration and status information for a single archive file on the Historian server.

## Archives Object

The Archives object provides access to Historian archive configuration information and performance statistics. It also provides functionality to add, delete, and modify Historian archives.

## Alarms.PurgeAlarmsById

The following sample is used to develop an SDK sample for purging alarms by their alarm IDs.

```
Dim Alarms() As String

Dim AlarmIds() As Long

Dim i As Long

Dim numberOfAlarms As Long

Dim AlarmsObj As iHistorian_SDK.Alarms

Dim Status As Boolean

i = 0

numberOfAlarms = 0

Status = False

If CheckConnection = True Then

    Set AlarmsObj = ConnectedServer.Alarms

    If TextAlarmIds.Text = "" Then

    MsgBox "Alarms cannot be empty", vbCritical, "Historian"

    Exit Sub

End If
```

```
Trim (TextAlarmIds.Text)
'Multiple alarms are separated by semicolon
Alarms() = Split(TextAlarmIds.Text, ";")
numberOfAlarms = UBound(Alarms)
If numberOfAlarms <> 0 Then
ReDim AlarmIds(0 To numberOfAlarms) As Long
    For i = 0 To numberOfAlarms
    If Alarms(i) <> "" Then
        AlarmIds(i) = CLng(Alarms(i))
      End If
    Next
    Status = AlarmsObj.PurgeAlarmsById(AlarmIds())
    If Status <> True Then
    MsgBox "An error occurred while deleting the alarms. See the Historian Alerts for more details.", vbCritical,
 "Historian"
  Else
    MsgBox "Successfully deleted the alarms.", vbInformation, "Historian"
  End If
    TextAlarmIds.Text = ""
  Else
   MsgBox "Please enter Alarm Ids followed by ';'", vbCritical, "Historian"
   End If
 Else
   MsgBox "Not connected", vbCritical, "Historian"
 End If
```

## Collector Object

The Collector object contains the configuration and status information for a single collector connected to the Historian Server.

## Collectors Object

The Collectors Object provides access to the Historian Collector configuration information and performance statistics. It also provides functionality to add, delete, and modify Historian Collectors.

## Data Objects

**Data Object**

The **Data** object provides access to Historian data and provides functionality to add, delete, and modify data samples in the Historian archives.

**DataComments Object**

The **DataComments** object contains a collection of comments attached to a specific DataValue record.

**DataCriteria Object**

The **DataCriteria** object sets the criteria for the retrieval of DataValue records into a DataRecordset Object. For example, you may want values between a certain start and end time.

**DataFields Object**

The **DataFields** object identifies which DataValue fields a DataRecordset query retrieves. For example, you may not want to retrieve comments with your data.

**DataRecordset Object**

The **RecordSet** contains a collection of DataValue records. Use the DataRecordset object to add, delete, or modify data samples.

To add, modify, or delete a DataValue record:

1. Create a DataRecordset Object.
2. Add, delete, or modify the DataValue Records.
3. Perform a WriteRecordset operation to save the changes to the Historian archives.

**DataValue Object**

The**DataValue** object contains data for a single tag and timestamp. It also contains a data quality and one or more comments attached to the specific data value. You can modify the Value, DataQuality, and Comments properties and commit them to the Historian server by using the WriteRecordset Method of the DataRecordsetObject. You cannot modify the timestamp.

## Message Objects

**Message Object**

The **Message** object provides access to a single message retrieved from the Historian server.

**MessageCriteria Object**

The **MessageCriteria** object sets the criteria for retrieval of message records into a MessageRecordset object.

**MessageFields Object**

The **MessageFields** object identifies which message fields the MessageRecordset query retrieves.

**MessageRecordset Object**

The **MessageRecordset** contains a collection of messages from the Historian Server. Use the Mes- sageRecordset Object to add and retrieve messages. To add new messages, create a MessageRecordset, add the appropriate messages, and then perform the WriteRecordset operation to store the changes in the Historian archives.

**Messages Object**

The **Messages** object contains Historian messages and provides functionality for adding new messages to the Historian Server. For example, you may want only alerts or you may want messages associated with a certain user.

## OPC Objects

**OPCBrowse Object**

The OPCBrowse object allows you to retrieve hierarchical areas from OPC AE servers.

**OPCFilters Object**

Returns whether Auto-Reconnect logic is Enabled/Disabled.

**Option Object**

This object acts a simple container for providing a name and values for a named parameter. The possible option values are found in *Options.bas*.

## Server Objects

**Server Object**

Use the **Server** object to establish and maintain a connection to a specific Historian server. All other configuration and data access objects are subordinate to the Server object.

**ServerManager Object**

The **ServerManager** object permits you to browse the available servers registered on the client. The ServerManager object also provides functionality for adding new server connections and for setting the default login information.

# Tag Objects

### Tag Object

The **Tag** object contains configuration information for a single tag. You can modify properties of the Tag object and save them to the Historian server by using the WriteRecordset method of the TagRecordset object.

### TagCriteria Object

The TagCriteria object sets the criteria for retrieval of Tag records into a TagRecordset object.

### TagDependencies Object

The**TagDependencies** object represents a list of Historian Tags which trigger a given calculation when their values change.

### TagFields Object

The**TagFields** object identifies which tag fields are retrieved in a TagRecordset query.

### TagRecordset Object

The **TagRecordset** object is a collection of tag records that contain tag configuration information.

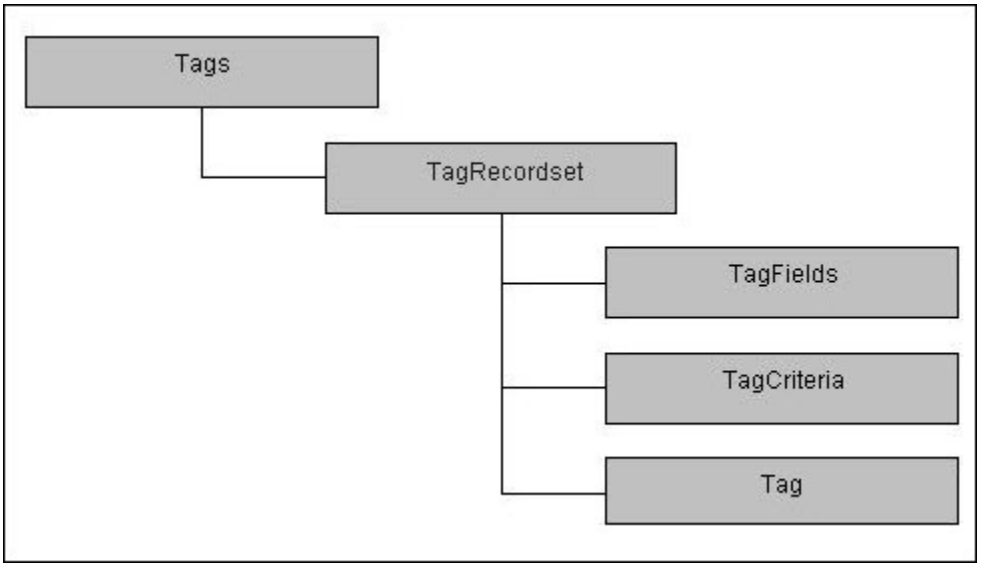
To add, modify, or delete tags:

1. Create a TagRecordset object.
2. Add, delete, and modify tag records.
3. Perform a WriteRecordset operation to save the changes to the Historian tag database.

### Tags Object

The**Tags** object contains Historian tag configuration information and provides functionality for adding, deleting, and modifying this configuration.

The following figure describes the Tags object.




## UserCalcFunction Object

The UserCalcFunction object contains the definition of a user function that can be stored in the Calculation library.

## Property Reference A-B

The following list contains the Historian properties that are available in the Historian Software Development Kit in the alphabetical order.

**A**

**ActualDataStores Property (Server Object)**

Returns the actual data stores currently configured on the Historian Server.

**Syntax**

object.**ActualDataStores**

**Parameters**

None

**Remarks**

ActualDataStores is a read-only property of type Long.

**ActualTags Property (Server Object)**

Returns the number of tags currently configured on the Server. This number is less than or equal to the number of licensed tags.

**Syntax**

object.**ActualTags**

**Parameters**

None

**Remarks**

ActualTags is a read-only property of type Long.

### ActualUsers Property (Server Object)

Returns the number of users currently connected to the Server. This number is less than or equal to the number of licensed users.

**Syntax**

object.**ActualUsers**

**Parameters**

None

**Remarks**

ActualUsers is a read-only property of type Long.

### AdministratorSecurityGroup Property (Tag Object)

Returns or sets the name of the security group controlling configuration changes for the Tag. Changes to tag properties are not committed until you call the WriteRecordset method of the TagRecordset object.

**Syntax**

object.**AdministratorSecurityGroup***[ = String]*

**Parameters**

None

### AdministratorSecurityGroup Property (TagCriteria Object)

Sets the administrator security group for which the TagRecordset query searches.

**Syntax**

object.**AdministratorSecurityGroup***[ = String]*

**Parameters**

None

**AdministratorSecurityGroup Property (TagFields Object)**

Determines whether the TagRecordset query returns the AdministratorSecurityGroup field.

**Parameters**

object.**AdministratorSecurityGroup***[ = Boolean]*

**Parameters**

None

**AdministratorSecurityGroup Property (UserDefinedType Object)**

Returns or sets the name of the Administrator Security Group that controls the definition of a User Defined Type. You can create, modify or delete types if you have Administration Security Group permission to do so. If there is no assigned group then the name will be empty.

**Syntax**

object.**MyAdministratorSecurityGroup**

**Parameters**

None

**Returns**

String

**AdministratorSecurityGroupSet Property (TagCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**AdministratorSecurityGroupSet***[ = Boolean]*

**Parameters**

None

**AdministratorSecurityGroupUpdated Property (Tag Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**AdministratorSecurityGroupUpdated***[ = Boolean]*

**Parameters**

None

**Alarms Property (Server Object)**

Returns the Alarms object for the current server. Use the Alarms object to query Alarms and Events data of the server.

**Syntax**

object.**Alarms**

**Parameters**

None

**Remarks**

Alarms is a read-only property of type Alarms.

**AllAlerts Property (MessageCriteria Object)**

Causes the MessageRecordset query to include all alert topic messages.

**Syntax**

object.**AllAlerts***[ = Boolean]*

**Parameters**

None.

Example

```
MyMessages.Criteria.AllAlerts = True
```

**AllMessages Property (MessageCriteria Object)**

Specifies that all message topic messages (non-alerts) should be returned by the MessageRecordset query.

**Syntax**

object.**AllMessages***[ = Boolean]*

**Parameters**

None

**Example**

```
MyMessages.Criteria.AllMessages = True
```

**ArchiveAbsoluteDeadband Property (Tag Object)**

Returns or sets the Absolute Deadband value for this tag to be used in the Archiver.

**Syntax**

object.**ArchiveAbsoluteDeadband**[ = Double]

**Parameters**

None

**ArchiveAbsoluteDeadband Property (TagCriteria Object)**

Sets the ArchiveAbsoluteDeadband to search for in the TagRecordset query.

**Syntax**

object.**ArchiveAbsoluteDeadband***[ = Double]*

**Parameters**

None

**ArchiveAbsoluteDeadband Property (TagFields Object)**

Determines whether the ArchiveAbsoluteDeadband field should be returned in the TagRecordset query.

**Syntax**

object.**ArchiveAbsoluteDeadband***[ = Boolean]*

**Parameters**

None

### ArchiveAbsoluteDeadbandSet Property (TagCriteria Object)

A flag to indicate whether this property has been set or not.

**Syntax**

object.**ArchiveAbsoluteDeadband*Set****[ = Boolean]***

**Parameters**

None

### ArchiveAbsoluteDeadbandUpdated Property (Tag Object)

A flag to indicate whether this property has been set or not.

**Syntax**

object.**ArchiveAbsoluteDeadbandUpdated*Set****[ = Boolean]***

**Parameters**

None

### ArchiveAbsoluteDeadbanding Property (Tag Object)

Returns or sets whether this tag is using Absolute Deadbanding in the Archiver or not.

**Syntax**

object.**ArchiveAbsoluteDeadbanding*Set****[ = Boolean]***

**Parameters**

None

### ArchiveAbsoluteDeadbanding Property (TagCriteria Object)

Sets the ArchiveAbsoluteDeadbanding to search for in the TagRecordset query.

**Syntax**

object.**ArchiveAbsoluteDeadbanding*Set****[ = Boolean]***

**Parameters**

None

**ArchiveAbsoluteDeadbandSet Property (TagCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**ArchiveAbsoluteDeadbandset*Set****[ = Boolean]***

**Parameters**

None

**ArchiveAbsoluteDeadbandUpdated Property (Tag Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**ArchiveAbsoluteDeadbandupdated*[ = Boolean]***

**Parameters**

None

**ArchiveAbsoluteDeadbanding Property (Tag Object)**

Returns or sets whether this tag is using Absolute Deadbanding in the Archiver or not.

**Syntax**

object.**ArchiveAbsoluteDeadbanding*[ = Boolean]***

**Parameters**

None

**ArchiveAbsoluteDeadbanding Property (TagCriteria Object)**

Sets the ArchiveAbsoluteDeadbanding to search for in the TagRecordset query.

**Syntax**

object.**ArchiveAbsoluteDeadbanding***[ = Boolean]*

**Parameters**

None

**ArchiveAbsoluteDeadbanding Property (TagFields Object)**

Determines whether the ArchiveAbsoluteDeadbanding field should be returned in the TagRecordset query.

**Syntax**

object.**ArchiveAbsoluteDeadbanding***[ = Boolean]*

**Parameters**

None

**ArchiveAbsoluteDeadbandingSet Property (TagCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**ArchiveAbsoluteDeadbandingSet***[ = Boolean]*

**Parameters**

None

**ArchiveAbsoluteDeadbandingUpdated Property (Tag Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**ArchiveAbsoluteDeadbandingUpdated***[ = Boolean]*

**Parameters**

None

**ArchiveAllowDataOverwrites Property (Archives Object)**

Returns whether or not a Data Archiver should allow overwriting of an existing data.

**ArchiveBaseFileName Property (Archives Object)**

Returns a default string that an archive file names must be based on.

**ArchiveCompression Property (Tag Object)**

Returns or sets the archive compression for the Tag. Changes to tag properties are not committed until you call the WriteRecordset method of the TagRecordset object.

**Syntax**

object.ArchiveCompression*[ = Boolean]*

**Parameters**

None

**ArchiveCompression Property (TagCriteria Object)**

Sets the archive compression value for which the TagRecordset query searches.

**Syntax**

object.**ArchiveCompression***[ = Byte]*

**Parameters**

None

**ArchiveCompression Property (TagFields Object)**

Determines whether the TagRecordset query returns the ArchiveCompression field.

**Syntax**

object.**ArchiveCompression***[ = Boolean]*

**Parameters**

None

**ArchiveCompressionSet Property (TagCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**ArchiveCompressionSet***[ = Boolean]*

**Parameters**

None

**ArchiveCompressionTimeout Property (Tag Object)**

Gets or sets the value of the Archive Compression Timeout. This is the amount of time after which a value will be written to the archive regardless of compression.

**Syntax**

object.**ArchiveCompressionTimeout***[ = Long]*

**Parameters**

None

**ArchiveCompressionTimeout Property (TagCriteria Object)**

Sets the Archive Compression Timeout value to search for in the TagRecordset query.

**Syntax**

object.**ArchiveCompressionTimeout***[ = Long]*

**Parameters**

None

**ArchiveCompressionTimeout Property (TagFields Object)**

Determines whether the TagRecordset query returns the *ArchiveCompressionTimeout* field.

**Syntax**

object.**ArchiveCompressionTimeout***[ = Boolean]*

**Parameters**

None

**ArchiveCompressionTimeoutSet Property (TagCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**ArchiveCompressionTimeoutSet***[ = Boolean]*

**Parameters**

None

**ArchiveCompressionTimeoutUpdated Property (Tag Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**ArchiveCompressionTimeoutUpdated***[ = Boolean]*

**Parameters**

None

**ArchiveCompressionUpdated Property (Tag Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**ArchiveCompressionUpdated***[ = Boolean]*

**Parameters**

None

**ArchiveCreateOfflineArchives Property (Archives Object)**

Returns whether a Data Archiver should accept data before starting an old archive.

**ArchiveDeadbandPercentRange Property (Tag Object)**

Returns or sets the deadband in percent of engineering unit range for archive compression for the Tag. Changes to tag properties are not committed until you call the WriteRecordset method of the TagRecordset object.

**Syntax**

object.**ArchiveDeadbandPercentRange***[ = Single]*

**Parameters**

None

**ArchiveDeadbandPercentRange Property (TagCriteria Object)**

Sets the archive compression deadband for which the TagRecordset query searches.

**Syntax**

object.**ArchiveDeadbandPercentRange***[ = Single]*

**Parameters**

None

**ArchiveDeadbandPercentRange Property (TagFields Object)**

Determines whether the TagRecordset query returns the ArchiveDeadbandPercentRange field.

**Syntax**

object.**ArchiveDeadbandPercentRange***[ = Boolean]*

**Parameters**

None

**ArchiveDeadbandPercentRangeSet Property (TagCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**ArchiveDeadbandPercentRangeSet***[ = Boolean]*

**Parameters**

None

**ArchiveDeadbandPercentRangeUpdated Property (Tag Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**ArchiveDeadbandPercentRangeUpdated***[ = Boolean]*

**Parameters**

None

**ArchiveMaxMemoryBufferSize Property (Archives Object)**

Returns memory size of Data Archiver queues.

**ArchiveUseClientLoginUser Property (Archives Object)**

Returns whether a Data Archiver requires users to enter a user name and password details in the client application.

**Remarks**

Set to FALSE to require a username and password.

**ArchiverClustered Property (Server Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**ArchiverClustered**

**Parameters**

None

**Remarks**

ArchiverClustered is a read-only property of type Boolean.

**ArchiverDemoMode Property (Server Object)**

Returns whether the server is currently running in demo-license mode. Demo mode is restricted to 100 tags and a single user connection.

**Syntax**

object.**ArchiverDemoMode**

**Parameters**

None

**Remarks**

ArchiverDemoMode is a read-only property of type Boolean.

**Archives Property (Server Object)**

Returns the Archives Object for the current server. The Archives Property does not return the Archives Object unless the authenticated user is a member of the Historian archive Administrators group.

**Syntax**

object.**Archives**

**Parameters**

None

**Remarks**

Archives is a read-only property of type Archives.

Example

```
Dim MyServer As New iHistorian_SDK.Server

            Dim MyArchives As iHistorian_SDK.Archives ' Connect to the local server

            If Not MyServer.Connect("", "", "")

            Then   err.Raise 1, , "Failed to connect to the local server" End If

            ' We are connected, try to instantiate the Archives object Set MyArchives =

 MyServer.Archives

            If MyArchives Is Nothing Then   err.Raise 1, , "User is not able to manage archives" End If
```

### Archiving Property (Archives Object)

Returns the Archiving status of the current archive.

**Syntax**

object.**Archiving*[ = Boolean]***

**Parameters**

None

### ArchivingOptions Property (Archives Object)

Exposes the ArchivingOptions collection to control the behavior of the Historian Archiver.

**Table 268. Archiving Options**

| Option | Description |
|---|---|
| ArchiveSizeIncrement | Sets the size increments of the archive file (Default=100MB). This option is specific to a data store. |
| ArchiveDefaultPath | Specifies the default path that the archives are created in. This option is specific to a data store. |
| ArchiveActiveHours | Specifies the hours before now that the data becomes Read-Only. This option is specific to data store. |
| ArchiveDefaultSize | Specifies the create archive default size in MB. This option is specific to a data store. |
| ArchiveAutomaticCreate | Specifies whether the system automatically creates archives when all empty archives are filled. This option is specific to a data store. |
| ArchiveAutomaticFreeSpace | The amount of free space required on the drive of the default path after creating an archive of default size. This option is specific to a data store. |
| ArchiveOverwriteOld | Determines whether or not old archives are cleared and re-used when no other empty space is available. This option is specific to a data store. |
| ArchiveBackupPath | Specifies the default path for backup archives. This option is specific to a data store. |
| ArchiveBaseName | Specifies the default string archive names should be based on. This option is specific only to a data store. |
| ArchiveBaseFileName | Returns a default string that an archive file names must be based on. This option is specific to a data store. |
| ArchiveCreateOfflineArchives | Returns whether a Data Archiver should accept data before starting an old archive. This option is specific to a data store. |

**Table 268. Archiving Options (continued)**

| Option | Description |
|---|---|
| ArchiveUseClientLoginUser | Returns whether a Data Archiver requires users to enter a user name and password details in the client application. This option is specific to a data store. |
| ArchiveAllowDataOverwrites | Returns whether or not a Data Archiver should allow overwriting of an existing data. This option is specific to a data store. |
| ArchiveDuration | Number of Units of time an archive can hold. Defined by ArchiveDurationType. This option is specific to a data store. |
| ArchiveTotalDuration | Number of Hours that the archives for a DataStore can span. (This is only meaningful for ScadaBuffer DataStores) This option is specific to a data store. |
| ArchiveDurationType | The type of archive duration.<br><br>• ArchiveDurationBySize<br>• ArchiveDurationDaily<br>• ArchiveDurationHourly<br><br>This option is specific to a data store. |
| ArchiveUseCaching | Indicates whether the archive should use caching or not. Set it to 1 to use caching and to 0 if you do not want to use caching. This option is specific to a data store. |
| ArchiveCacheSize | This is for internal use. |
| ArchiverServerMemoryLoad | How much server memory the Historian Data Archive is consuming. This option is specific to the data archiver. |
| ArchiverEquipmentDelimiter | This is for internal use. |

**Table 268. Archiving Options (continued)**

| Option | Description |
|---|---|
| ArchiverServerMemoryLoad | How much server memory the Historian Data Archive is consuming. This option is specific to the data archiver. |
| ArchiverEquipmentDelimiter | This is for internal use. |
| MessageOnDataUpdate | How much server memory the Historian Data Archive is consuming. This option is specific to the data archiver. |
| ArchiveMaxMemoryBufferSize | The maximum buffer memory (MB). This option is specific to the data archiver. |
| ArchiverTargetPrivateBytes | The memory size of the archiver. If it is 0, it means that it is managed by server. Otherwise, enter a number in megabytes of memory. This option is specific to the data archiver. |
| CollectorIdleTime | The number of seconds before a collector is considered idle. If this number is set to 270 seconds and no data for any tags is received for 270 seconds then the collector is considered idle. This option is specific to the data archiver. |
| MaximumQueryIntervals | Specifies the maximum number of samples per tag that Historian can return from a non-raw data query. Use this setting to throttle query results for non-raw data queries. This setting does not apply to filtered data queries or raw data queries. This option is specific to the data archiver. |
| MaximumQueryTime | Specifies the maximum time that a data or message query can take before it is terminated. Use this setting to limit query time and provide balanced read access to the archiver. This applies to all query types. This option is specific to the data archiver. |

**Syntax**

object.**ArchivingOptions**(OptionName) *[ = Variant]*

**Parameters**

Name **OptionName** Data type **String** Description **The name of an archiving option**.

**Example**

```
MyServer.DataStores.DataStoreUpdate("MYDATASTORE", True, "", "") Then


maintain auto recovery files


MyServer.MaintainAutoRecoveryFiles =TRUE
```

**AreaCount Property (OPCBrowse Object)**

Returns the number of Areas under a browse position. When a browse operation occurs, the server returns all areas and sources under the BrowsePosition. AreaCount gives you the number of Areas, which you can use to iterate the AreaNames and FullAreaNames arrays.

**Syntax**

object.**AreaCount**

**Parameters**

None

**Remarks**

AreaCount is a read-only property returned as type Long.

**AreaNames Property (OPCBrowse Object)**

Returns an area name for an AE server. See FullAreanames property for details.

**Syntax**

object.**AreaNames**(Index)

**Parameters**

Index. Integer. The index into the area nae array

**Remarks**

AreaNames is a read-only String property returned as a variant for script compatibility.

### Areas Property (OPCFilters Object)

Returns a list of the Areas selected for collection. This list is only applied when isAreaFiltering (true) has been called.

**Syntax**

object.**Areas**

**Parameters**

None

### AuditMessage Property (Server Object)

Returns or sets the E-signature audit message. Setting the audit message to an empty string will use the system default message.

**Syntax**

object.**AuditMessage***[ = String]*

**Parameters**

None

### AutoReconnectRateSeconds Property (OPCFilters Object)

If AutoReconnect logic is enabled (isAutoReconnectOn(true)) this property represents the rate in seconds that the Alarm Collector with attempt to determine if the server is still alive.

**Syntax**

object.**AutoReconnectRateSeconds***[ = Long]*

**Parameters**

None

## B

### BrowsePosition Property (OPCBrowse Object)

Returns the current value of the browse position in the hierarchy.

**Syntax**

object.**BrowsePosition***[ = Variant]*

**Parameters**

None

**Remarks**

This is used for subsequent browse calls to get the leaf under the current position.

**BrowseRoot Property (OPCBrowse Object)**

Returns the root name of the OPC area hierarchy.

**Syntax**

object.**BrowseRoot**

**Parameters**

None

**Remarks**

BrowseRoot is a read-only property of type String.

## Property Reference C-D

**CalcType Property (Tag Object)**

Returns or sets the CalcType for the tag. The CalcType is used to determine whether the tag is a Python Expression tag or a Raw tag.

| Name | Description | Value |
|---|---|---|
| RawTag | A tag is not a Python Expression Tag | 0 |
| PythonExprTag | A tag that uses a Python Expression to transform raw data into derived values. | 2 |

**Syntax**

```
object.CalcType [=ihTagCalcType]
```

**Parameters**

None

### CalcType Property (TagCriteria Object)

Returns or sets the CalcType for the tag. The CalcType is used to determine whether the tag is a Python Expression tag or a Raw tag.

| Name | Description | Value |
|---|---|---|
| RawTag | A tag is not a Python Expression Tag | 0 |
| PythonExprTag | A tag that uses a Python Expression to transform raw data into derived values. | 2 |

#### Syntax

```
object.CalcType [=ihTagCalcType]
```

#### Parameters

None

### CalcType Property (TagFields Object)

Determines whether the CalcType field should be returned in the TagRecordset query.

#### Syntax

```
object.CalcType [ = Boolean]
```

#### Parameters

None

### CalcTypeList Property (Tags Object)

Returns a list of valid calculation types available on a tag in Historian.

#### Syntax

```
object.CalcType
```

#### Parameters

None

### CalcTypeSet Property (TagCriteria Object)

A flag to indicate whether or not the CalcType property has been set or not.

**Syntax**

```
object.CalcTypeSet [ = Boolean]
```

**Parameters**

None

### CalcTypeUpdated Property (Tag Object)

A flag to indicate whether or not the CalcType property has been set.

**Syntax**

```
object.CalcTypeUpdated [ = Boolean]
```

**Parameters**

None

### Calculation Property (Tag Object)

Returns or sets the calculation for the Tag. Changes to tag properties are not committed until you call the WriteRecordset method of the TagRecordset object.

**Syntax**

```
object.Calculation[ = String]
```

**Parameters**

None

### Calculation Property (TagCriteria Object)

Sets the calculation to search for in the TagRecordset query.

**Syntax**

```
object.Calculation[ = String]
```

**Parameters**

None

### Calculation Property (TagFields Object)

Determines whether the TagRecordset query returns the Calculation field.

**Syntax**

```
object.Calculation[ = Boolean]
```

**Parameters**

None

### CalculationDependencies Property (Tag Object)

Returns a collection of tagnames that the calculation is dependent upon. Use the CalculationDependencies property with unsolicited tags only.

**Syntax**

```
object.CalculationDependencies
```

**Parameters**

None

**Remarks**

CalculationDependencies is a read-only property of type TagDependencies.

### CalculationDependencies Property (TagFields Object)

Determines whether the CalculationDependencies field should be returned in the TagRecordset query.

**Syntax**

```
object.CalculationDependencies[ = Boolean]
```

**Parameters**

None

### CalculationDependenciesUpdated Property (Tag Object)

Returns whether or not the Calculation Dependencies have been updated. & Boolean true if the dependencies have been updated, false otherwise.

**Syntax**

```
object.CalculationDependenciesUpdated
```

**Parameters**

None

### CalculationExecutionTime Property (Tag Object)

Returns the average time, in milliseconds, it takes to execute a calculation formula for the tag.

**Syntax**

```
object.CalculationExecutionTime
```

**Parameters**

None

**Remarks**

CalculationExecutionTime is a read-only property of type Long.

### CalculationExecutionTime Property (TagFields Object)

Determines whether the CalculationExecutionTime field should be returned in the TagRecordset query.

**Syntax**

```
object.CalculationExecutionTime[ = Boolean]
```

**Parameters**

None

### CalculationMode Property (DataCriteria Object)

Returns or sets the type of calculation to perform on archive data retrieved in the DataRecordset query. CalculationMode only applies to a Calculated SamplingMode.

The table below defines the available Calculation Modes:

| Name | Description | Value |
|---|---|---|
| Average | Retrieves the time-weighted average for each calculation interval. | 1 |

| Name | Description | Value |
|---|---|---|
| StandardDeviation | Retrieves the time-weighted standard deviation for each calculation interval. | 2 |
| Total | Retrieves the time- weighted rate total for each calculation interval. A rate total would be appropriate for totalizing a continuous measurement.<br><br>Time weighting is necessary due to the fact that compressed data is not evenly spaced in time. A factor must be applied to the totalized value to convert into the appropriate engineering units. Given the fact that this is a rate total, a base rate of Units/Day is assumed. If the actual units of the continuous measurement were Units/Minute, it would be necessary to multiply the results by 1440 Minutes / Day to convert the totalized number into the appropriate engineering units. | 3 |
| Minimum | Retrieves the minimum value for each calculation interval. | 4 |
| Maximum | Retrieves the maximum value for each calculation interval. | 5 |
| Count | Retrieves the count of archive values found within each calculation interval. | 6 |
| RawAverage | Retrieves the arithmetic average of archive values for each calculation interval. | 7 |
| RawStandardDeviation | Retrieves the arithmetic standard deviation of archive values for each calculation interval. | 8 |
| RawTotal | Retrieves the arithmetic total of archive values for each calculation interval. | 9 |
| MinimumTime | Retrieves the timestamp of the minimum value found within each calculation interval. It may be a raw or an interpolated value. The minimum must be a good data quality sample. | 10 |

| Name | Description | Value |
|---|---|---|
| MaximumTime | Retrieves the timestamp of the maximum value found within each calculation interval. It may be a raw or an interpolated value. The maximum must be a good data quality sample. | 11 |
| TimeGood | Retrieves the amount of time (in milliseconds) during the interval when the data is of good quality and the filter condition is met. | 12 |
| StateCount | Retrieves the amount of time a tag has transitioned to another state from a previous state during a time interval. | 13 |
| OPCAnd | Retrieves the OROPCQAND, bit-wise AND operation of all the 16-bit OPC qualities of the raw samples stored in the specified interval. | 16 |
| OPCOr | Retrieves the OPCQOR, bit-wise OR operation of all the 16-bit OPC qualities of the raw samples stored in the specified interval. | 16 |

**Syntax**

```
object.CalculationMode[ = iHistorian_SDK.ihCalculationMode]
```

**Parameters**

None

### CalculationModeList Property (Data Object)

This function returns a list of the available Calculation modes for a data request.

**Syntax**

```
object.CalculationModeList
```

**Parameters**

None

### CalculationModeSet Property (DataCriteria Object)

A flag to indicate whether this property has been set or not.

**Syntax**

```
object.CalculationModeSet[ = Boolean]
```

**Parameters**

None

### CalculationSet Property (TagCriteria Object)

A flag to indicate whether this property has been set or not.

**Syntax**

```
object.CalculationSet[ = Boolean]
```

**Parameters**

None

### CalculationUpdated Property (Tag Object)

A flag to indicate whether this property has been set or not.

**Syntax**

```
object.CalculationUpdated[ = Boolean]
```

**Parameters**

None

### CanAdministerConfiguration Property (Archives Object)

Returns whether the current user is authorized to perform configuration changes to Archives or Archiving options.

**Syntax**

```
object.CanAdministerConfiguration
```

**Parameters**

None

**Remarks**

CanAdministerConfiguration is a read-only property of type Boolean.

**Example**

```
' Check Security If Not MyServer.Archives.CanAdministerConfiguration Then   err.Raise 1, "Security",
"You Are Not Authorized To Make Configuration Changes" End If
```

### CanAdministerConfiguration Property (Collectors Object)

Returns whether the current user is authorized to perform configuration changes to Collectors.

**Syntax**

```
object.CanAdministerConfiguration
```

**Parameters**

None

**Remarks**

CanAdministerConfiguration is a read-only property of type Boolean.

**Example**

```
' Check Security If Not MyServer.Archives.CanAdministerConfiguration Then   err.Raise 1, "Security",
                   "You Are Not Authorized To Make Configuration Changes" End If
```

### CanAdministerSecurity Property (Server Object)

Returns whether the current user is authorized to perform tag security changes.

**Syntax**

```
object.CanAdministerSecurity
```

**Parameters**

None

**Remarks**

CanAdministerSecurity is a read-only property of type Boolean.

**Example**

```
Dim MyServer As New iHistorian_SDK.Server ' Connect to the local server If Not MyServer.Connect("", "",
 "") Then
err.Raise 1, , "Failed to connect to local server" End If If Not
 MyServer.Tags.CanAdministerConfiguration Then
err.Raise 1, , "Cannot administer security" End If
```

### CanAdministerSecurity Property (Tags Object)

Returns whether the current user is authorized to security configuration changes.

#### Syntax

`object.CanAdministerSecurity`

#### Parameters

None

#### Remarks

CanAdministerSecurity is a read-only property of type Boolean.

### CanAudit Property (Server Object)

Returns whether E-signatures are enabled on the hardware key.

#### Syntax

`object.CanAudit`

#### Parameters

None

#### Remarks

CanAudit is a read-only property of type Boolean.

### CanCalculate Property (Server Object)

Returns whether the Calculation collector is enabled on the hardware key.

#### Syntax

`object.CanCalculate`

#### Parameters

None

**Remarks**

CanCalute is a read-only property of type Boolean.

### CanRead Property (Data Object)

Returns whether the current user is authorized to read data from the server.

**Syntax**

```
object.CanRead
```

**Parameters**

None

**Remarks**

CanRead is a read-only property of type Boolean.

**Example**

```
' Check Security If Not MyData.CanRead Then   err.Raise 1, , "You are not authorized to read data from this server" End If
```

### CanRead Property (Messages Object)

Returns whether the current user is authorized to read messages from the server.

**Syntax**

```
object.CanRead
```

**Parameters**

None

**Remarks**

CanRead is a read-only property of type Boolean.

**Example**

```
' Check Security If Not MyData.CanRead Then   err.Raise 1, , "You are not authorized to read messages from this server" End If
```

### CanReplicate Property (Server Object)

Returns whether ServerToServer collector is enabled on the hardware key.

#### Syntax

```
object.CanRelpicate
```

#### Parameters

None

#### Remarks

CanReplicate is a read-only property of type Boolean.

### CanSupportRedundantCollectors Property (Server Object)

Returns whether Collector Redundancy is enabled on the hardware key.

#### Syntax

```
object.CanSupportRedundantCollectors
```

#### Parameters

None

#### Remarks

CanSupportRedundantCollectors is a read-only property of type Boolean.

### CanWrite Property (Data Object)

Returns whether the current user is authorized to write data to the server.

#### Syntax

```
object.CanWrite
```

#### Parameters

None

#### Remarks

CanWrite is a read-only property of type Boolean.

Example

```
' Check Security If Not MyData.CanWrite Then err.Raise 1, , "You are not authorized to write data to this server" End If
```

### CanWrite Property (Messages Object)

Returns whether the current user is authorized to write messages to the Server.

#### Syntax

```
object.CanWrite
```

#### Parameters

None

#### Remarks

CanWrite is a read-only property of type Boolean.

Example

```
' Check Security If Not MyMessages.CanWrite Then err.Raise 1, , "You are not authorized to write data to this server" End If
```

### CollectionDisabled Property (Tag Object)

Gets or sets whether the collection status of this tag.

#### Syntax

```
object.CollectionDisabled[ = Boolean]
```

#### Parameters

None

### CollectionDisabled Property (TagFields Object)

Determines whether the CollectionDisabled field should be returned in the TagRecordset query.

#### Syntax

```
object.CollectionDisabled[ = Boolean]
```

**Parameters**

None

### CollectionDisabledUpdated Property (Tag Object)

A flag to indicate whether this property has been set or not.

**Syntax**

```
object.CollectionDisabledUpdated[ = Boolean]
```

**Parameters**

None

### CollectionInterval Property (Tag Object)

Returns or sets the collection interval (in milliseconds) for a Tag. For polled collection, the interval schedules evenly spaced samples of the data from the data source. For unsolicited collection, the collection interval establishes the MINIMUM time allowed in between reports of unsolicited values from the data source. Changes to tag properties are not committed until you call the WriteRecordset method of the TagRecordset object.

**Syntax**

```
object.CollectionInterval[ = Long]
```

**Parameters**

None

### CollectionInterval Property (TagCriteria Object)

Sets the collection interval to search for in the TagRecordset Query.

**Syntax**

```
object.CollectionInterval[ = Long]
```

**Parameters**

None

### CollectionInterval Property (TagFields Object)

Determines whether the TagRecordset query returns the CollectionInterval field.

**Syntax**

```
object.CollectionInterval[ = Boolean]
```

**Parameters**

None

### CollectionIntervalSet Property (TagCriteria Object)

A flag to indicate whether this property has been set or not.

**Syntax**

```
object.CollectionIntervalSet[ = Boolean]
```

**Parameters**

None

### CollectionIntervalUpdated Property (Tag Object)

A flag to indicate whether this property has been set or not.

**Syntax**

```
object.CollectionIntervalUpdated[ = Boolean]
```

**Parameters**

None

### CollectionOffset Property (Tag Object)

Returns or sets the collection offset (in milliseconds) for a tag. The collection offset only applies to polled type collection as used to schedule sampling of data at a specific time of day. The collection offset is applied from midnight to determine the time of the first sample. The collection interval is then used to schedule subsequent samples.

For example, with a collection interval of 60,000ms and a collection offset of 30,000ms, data would be collected one per minute on the half minute. To collect data once per day at a specific time, the collection interval should be set to one day, 1440 * 60 * 1000 = 86,400,000ms, and the collection offset should be set to the time in milliseconds from midnight (for example, 6am = 6 * 3600 * 1000 = 21,600,000 ms).

Changes to tag properties are not committed until you call the WriteRecordset method of the TagRecordset object.

**Syntax**

`object.`**`CollectionOffset`***`[ = Long]`*

**Parameters**

None

**CollectionOffset Property (TagCriteria Object)**

Sets the collection offset to search for in the TagRecordset query.

**Syntax**

`object.`**`CollectionOffset`***`[ = Long]`*

**Parameters**

None

**CollectionOffset Property (TagFields Object)**

Sets Determines whether the TagRecordset query returns the CollectionOffset field.

**Syntax**

`object.`**`CollectionOffset`***`[ = Boolean]`*

**Parameters**

None

**CollectionOffsetSet Property (TagCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

`object.`**`CollectionOffset`***`[ = Boolean]`*

**Parameters**

None

**CollectionOffsetUpdated Property (Tag Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

```
object.CollectionOffsetupdated[ = Boolean]
```

**Parameters**

None

### CollectionType Property (Tag Object)

Returns or sets the type of collection used to acquire data for the Tag. Changes to tag properties are not committed until you call the WriteRecordset method of the TagRecordset object.

The following table lists the available types of collection:

| Name | Description | Value |
|---|---|---|
| Unsolicited | New values are reported by the source. | 1 |
| Polled | New values are polled from the source. | 2 |

**Syntax**

```
object.CollectionType[ = ihCollectionType]
```

**Parameters**

None

### CollectionType Property (TagCriteria Object)

Sets the collection type to search for in the TagRecordset query.

**Syntax**

```
object.CollectionType[ = ihCollectionType]
```

**Parameters**

None

### CollectionType Property (TagFields Object)

Determines whether the TagRecordset query returns the CollectionType field.

#### Syntax

```
object.CollectionType[ = Boolean]
```

#### Parameters

None

### CollectionTypeList Property (Tags Object)

Returns a list of the valid Collection types available in Historian.

#### Syntax

```
object.CollectionTypeList
```

#### Parameters

None

### CollectionTypeSet Property (TagCriteria Object)

A flag to indicate whether this property has been set or not.

#### Syntax

```
object.CollectionTypeSet[ = Boolean]
```

#### Parameters

None

### CollectionTypeUpdated Property (Tag Object)

A flag to indicate whether this property has been set or not.

#### Syntax

```
object.CollectionTypeUpdated[ = Boolean]
```

#### Parameters

None

### CollectorCompression Property (Tag Object)

Returns or sets whether collector compression is enabled for the Tag. Specify the deadband in the CollectorDeadbandPercentRange property.

Changes to tag properties are not committed until you call the WriteRecordset method of the TagRecordset object.

**Syntax**

```
object.CollectorCompression[ = Boolean]
```

**Parameters**

None

### CollectorCompression Property (TagCriteria Object)

Sets the collector compression status to search for in the TagRecordset query.

Changes to tag properties are not committed until you call the WriteRecordset method of the TagRecordset object.

**Syntax**

```
object.CollectorCompression[ = Boolean]
```

**Parameters**

None

### CollectorCompression Property (TagFields Object)

Determines whether the TagRecordset query returns the CollectorCompression field.

**Syntax**

```
object.CollectorCompression[ = Boolean]
```

**Parameters**

None

### CollectorCompressionSet Property (TagCriteria Object)

A flag to indicate whether this property has been set or not.

**Syntax**

```
object.CollectorCompressionSet[ = Boolean]
```

**Parameters**

None

**CollectorCompressionTimeout Property (Tag Object)**

Gets or sets the value of the Collector Compression Timeout. This is the amount of time after which a value will be sent to the Archiver regardless of compression.

**Syntax**

```
object.CollectorCompressionTimeout[ = Long]
```

**Parameters**

None

**CollectorCompressionTimeout Property (TagCriteria Object)**

Sets the Collector Compression Timeout value to search for in the TagRecordset query.

**Syntax**

```
object.CollectorCompressionTimeout[ = Long]
```

**Parameters**

None

**CollectorCompressionTimeout Property (TagFields Object)**

Determines whether the TagRecordset query returns the CollectorCompressionTimeout field.

**Syntax**

```
object.CollectorCompressionTimeout[ = Boolean]
```

**Parameters**

None

**CollectorCompressionTimeoutSet Property (TagCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

```
object.CollectorCompressionTimeoutSet[ = Boolean]
```

**Parameters**

None

### CollectorCompressionTimeoutUpdated Property (Tag Object)

A flag to indicate whether this property has been set or not.

**Syntax**

```
object.CollectorCompressionTimeoutUpdated[ = Boolean]
```

**Parameters**

None

### CollectorCompressionUpdated Property (Tag Object)

A flag to indicate whether this property has been set or not.

**Syntax**

```
object.CollectorCompressionTimeoutUpdated[ = Boolean]
```

**Parameters**

None

### CollectorCompressionUpdated Property (Tag Object)

A flag to indicate whether this property has been set or not.

**Syntax**

```
object.CollectorCompressionTimeoutUpdated[ = Boolean]
```

**Parameters**

None

### CollectorConditionEventList Property (OPCFilters Object)

Returns a list of the Condition Event Categories available for filtering on the Alarm Collector.

**Syntax**

```
object.CollectorConditionEventList
```

**Parameters**

None

**CollectorDeadbandPercentRange Property (Tag Object)**

Returns or sets the deadband in percent of engineering unit range for collector compression for the Tag. You must enable CollectorCompression before setting the CollectorDeadbandPercentRange.

Changes to tag properties are not committed until you call the WriteRecordset method of the TagRecordset object.

**Syntax**

```
object.CollectorDeadbandPercentRange[ = Single]
```

**Parameters**

None

**CollectorDeadbandPercentRange Property (TagCriteria Object)**

Sets the collector compression deadband to search for in the TagRecordset query.

**Syntax**

```
object.CollectorDeadbandPercentRange[ = Single]
```

**Parameters**

None

**CollectorDeadbandPercentRange Property (TagFields Object)**

Determines whether the TagRecordset query returns the CollectorDeadbandPercentRange field.

**Syntax**

```
object.CollectorDeadbandPercentRange[ = Boolean]
```

**Parameters**

None

**CollectorDeadbandPercentRangeSet Property (TagCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

```
object.CollectorDeadbandPercentRangeSet[ = Boolean]
```

**Parameters**

None

### CollectorDeadbandPercentRangeUpdated Property (Tag Object)

A flag to indicate whether this property has been set or not.

**Syntax**

```
object.CollectorDeadbandPercentRangeUpdated[ = Boolean]
```

**Parameters**

None

### CollectorGeneral1 Property (Tag Object)

Returns or sets the general1 (or spare) configuration field for the tag. Changes to tag properties are not committed until the WriteRecordset method of the TagRecordset object is called.

**Syntax**

```
object.CollectorGeneral1[ = String]
```

**Parameters**

None

**Example**

```
MyTag.CollectorGeneral1 = "Model2:123"
```

### CollectorGeneral1 Property (TagCriteria Object)

Sets the general1 (spare) field value to search for in the TagRecordset query.

**Syntax**

```
object.CollectorGeneral1[ = String]
```

**Parameters**

None

**Example**

```
With MyRecordset.Criteria .TagName = "*.F_CV" .CollectorGeneral1 = "Model" End With
```

### CollectorGeneral1 Property (TagFields Object)

Determines whether the CollectorGeneral1 field should be returned in the TagRecordset query.

**Syntax**

```
object.CollectorGeneral1[ = Boolean]
```

**Parameters**

None

**Example**

```
With MyRecordset.Fields .Clear .TagName = True .Description = True .CollectorGeneral1
= True .CollectorGeneral2 = True .CollectorGeneral3 = True .CollectorGeneral4 =
True .CollectorGeneral5 = True End With
```

### CollectorGeneral1Set Property (TagCriteria Object)

A flag to indicate whether this property has been set or not.

**Syntax**

```
object.CollectorGeneral1Set[ = Boolean]
```

**Parameters**

None

### CollectorGeneral1Updated Property (Tag Object)

A flag to indicate whether this property has been set or not.

**Syntax**

```
object.CollectorGeneral1Updated[ = Boolean]
```

**Parameters**

None

### CollectorGeneral2 Property (Tag Object)

Returns or sets the general2 (or spare) configuration field for the tag. Changes to tag properties are not committed until you call the WriteRecordset method of the TagRecordset object.

**Syntax**

```
object.CollectorGeneral2[ = String]
```

**Parameters**

None

### CollectorGeneral2 Property (Tagcriteria Object)

Sets the general2 (spare) field value to search for in the TagRecordset query.

**Syntax**

```
object.CollectorGeneral2[ = String]
```

**Parameters**

None

### CollectorGeneral2 Property (TagFields Object)

Determines whether the CollectorGeneral2 field should be returned in the TagRecordset query.

**Syntax**

```
object.CollectorGeneral2[ = Boolean]
```

**Parameters**

None

**Example**

```
With MyRecordset.Fields .Clear .TagName = True .Description = True .CollectorGeneral1 =
 True .CollectorGeneral2 = True
.CollectorGeneral3 = True .CollectorGeneral4 = True .CollectorGeneral5 = True End With
```

**CollectorGeneral2Set Property (TagCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

`object.`**`CollectorGeneral2`***`[ = Boolean]`*

**Parameters**

None

**CollectorGeneral2Updated Property (Tag Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

`object.`**`CollectorGeneral2`***`Updated[ = Boolean]`*

**Parameters**

None

**CollectorGeneral3Updated Property (Tag Object)**

Returns or sets the general3 (or spare) configuration field for the tag. Changes to tag properties are not committed until you call the WriteRecordset method of the TagRecordset object.

**Syntax**

`object.`**`CollectorGeneral3`***`[ = String]`*

**Parameters**

None

**CollectorGeneral3 Property (TagCriteria Object)**

Sets the general3 (spare) field value to search for in the TagRecordset query.

**Syntax**

`object.`**`CollectorGeneral3`***`[ = String]`*

**Parameters**

None

### CollectorGeneral3 Property (TagFields Object)

Determines whether the CollectorGeneral3 field should be returned in the TagRecordset query.

**Syntax**

`object.`**`CollectorGeneral3`***`[ = Boolean]`*

**Parameters**

None

**Example**

```
With MyRecordset.Fields .Clear .TagName = True .Description = True .CollectorGeneral1 = True
.CollectorGeneral2 = True .CollectorGeneral3 = True .CollectorGeneral4 = True .CollectorGeneral5 = True
 End With
```

### CollectorGeneral3Set Property (TagCriteria Object)

A flag to indicate whether this property has been set or not.

**Syntax**

`object.`**`CollectorGeneral3Set`***`[ = Boolean]`*

**Parameters**

None

### CollectorGeneral3Updated Property (Tag Object)

A flag to indicate whether this property has been set or not.

**Syntax**

`object.`**`CollectorGeneral3Updated`***`[ = Boolean]`*

**Parameters**

None

**CollectorGeneral4 Property (Tag Object)**

Returns or sets the general4 (or spare) configuration field for the tag. Changes to tag properties are not committed until you call the WriteRecordset method of the TagRecordset object.

**Syntax**

```
object.CollectorGeneral4[ = String]
```

**Parameters**

None

**CollectorGeneral4 Property (TagCriteria Object)**

Sets the general4 (spare) field value to search for in the TagRecordset query.

**Syntax**

```
object.CollectorGeneral4[ = String]
```

**Parameters**

None

**CollectorGeneral4 Property (TagFields Object)**

Determines whether the CollectorGeneral4 field should be returned in the TagRecordset query.

**Syntax**

```
object.CollectorGeneral4[ = Boolean]
```

**Parameters**

None

**Example**

```
With MyRecordset.Fields .Clear .TagName = True .Description = True .CollectorGeneral1 = True
.CollectorGeneral2 = True .CollectorGeneral3 = True .CollectorGeneral4 = True .CollectorGeneral5 = True
 End With
```

### CollectorGeneral4Set Property (TagCriteria Object)

A flag to indicate whether this property has been set or not.

#### Syntax

```
object.CollectorGeneral4Set[ = Boolean]
```

#### Parameters

None

### CollectorGeneral4Updated Property (Tag Object)

A flag to indicate whether this property has been set or not.

#### Syntax

```
object.CollectorGeneral4Updated[ = Boolean]
```

#### Parameters

None

### CollectorGeneral5 Property (Tag Object)

Returns or sets the general5 (or spare) configuration field for the tag. Changes to tag properties are not committed until you call the WriteRecordset method of the TagRecordset object.

#### Syntax

```
object.CollectorGeneral5[ = String]
```

#### Parameters

None

### CollectorGeneral5 Property (TagCriteria Object)

Sets the general5 (spare) field value to search for in the TagRecordset query.

#### Syntax

```
object.CollectorGeneral5[ = String]
```

#### Parameters

None

**CollectorGeneral5 Property (TagFields Object)**

Determines whether the CollectorGeneral5 field should be returned in the TagRecordset query.

**Syntax**

```
object.CollectorGeneral5[ = Boolean]
```

**Parameters**

None

**CollectorGeneral5Set Property (TagCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

```
object.CollectorGeneral5Set[ = Boolean]
```

**Parameters**

None

**CollectorName Property (Tag Object)**

Returns or sets the name of the Collector responsible for collecting data for the Tag. Changes to tag properties are not committed until you call the WriteRecordset method of the TagRecordset object.

**Syntax**

```
object.CollectorName[ = String]
```

**Parameters**

None

**CollectorName Property (TagFields Object)**

Determines whether the CollectorName field should be returned in the TagRecordset query.

**Syntax**

```
object.CollectorName[ = Boolean]
```

**Parameters**

None

### CollectorNameSet Property (TagCriteria Object)

A flag to indicate whether this property has been set or not.

**Syntax**

```
object.CollectorNameSet[ = Boolean]
```

**Parameters**

None

### CollectorNameUpdated Property (Tag Object)

A flag to indicate whether this property has been set or not.

**Syntax**

```
object.CollectorNameUpdated[ = Boolean]
```

**Parameters**

None

### CollectorOptions Property (Collector Object)

Exposes the CollectorOptions collection to control the behavior of a specific collector.

The following table describes each of the options:

**Table 269. Options**

| Option | Capability |
|---|---|
| CollectorCanBrowseSource | TRUE if the collector supports browse. |
| CollectorMaxMemoryBufferSize | Specifies the size in megabytes of the memory buffer assigned to the store and forward function. |
| Source Address | Yes |
| CollectorMaxDiskBufferSize | Specifies the minimum free disk space that must be available on the computer and cannot be used for store and forward function. |

**Table 269. Options (continued)**

| Option | Capability |
|---|---|
| CollectorRateOutputAddress | Specifies the address in the source database into which the collector will write the current value of the events per minute output. |
| CollectorStatusOutputAddress | Specifies the address in the source database into which the collector will write the current value of the collector status (running, stopped, unknown, etc.). |
| CollectorAdjustForTimeDifference | TRUE if you want to adjust collector timestamps to match the server clock. |
| CollectorStartDelay | A delay, in seconds, after collector startup before collection starts. |
| CollectorSourceTimesLocal | If you are using source timestamps, set this to TRUE if the source timestamps are in local time. |
| CollectorMakeTagChangesOnline | Set to TRUE if you want tag changes to take effect without restarting collector. |

**Syntax**

object.**CollectorOptions**(OptionName) *[ = Variant]*

**Parameters**

OptionName - Variant - The name of a collection

### CollectorSimpleEventList Property (OPCFilters Object)

Returns a list of the Simple Event Categories available for filtering on the Alarm Collector.

**Syntax**

```
object.CollectorSimpleEventList
```

**Parameters**

None

### CollectorSupportsAreaFiltering Property (OPCFilters Object)

Returns whether or not Area Filtering is supported by the Alarm Collector. This can only be determined if the collector is connected to the Alarm Server.

**Syntax**

```
object.CollectorSupportsAreaFiltering
```

**Parameters**

None

### CollectorSupportsCategoryFiltering Property (OPCFilters Object)

Returns whether or not Category Filtering is supported by the Alarm Collector. This can only be determined if the collector is connected to the Alarm Server.

**Syntax**

```
object.CollectorSupportsCategoryFiltering
```

**Parameters**

None

### CollectorSupportsCategoryFiltering Property (OPCFilters Object)

Returns whether or not Category Filtering is supported by the Alarm Collector. This can only be determined if the collector is connected to the Alarm Server.

**Syntax**

```
object.CollectorSupportsCategoryFiltering
```

**Parameters**

None

### CollectorSupportsEventFiltering Property (OPCFilters Object)

Returns whether or not Event Filtering is supported by the Alarm Collector. This can only be determined if the collector is connected to the Alarm Server.

**Syntax**

```
object.CollectorSupportsEventFiltering
```

**Parameters**

None

**CollectorSupportsSeverityFiltering Property (OPCFilters Object)**

Returns whether or not Severity Filtering is supported by the Alarm Collector. This can only be determined if the collector is connected to the Alarm Server.

**Syntax**

```
object.CollectorSupportsSeverityFiltering
```

**Parameters**

None

**CollectorSupportsSourceFiltering Property (OPCFilters Object)**

Returns whether or not Source Filtering is supported by the Alarm Collector. This can only be determined if the collector is connected to the Alarm Server.

**Syntax**

```
object.CollectorSupportsSourceFiltering
```

**Parameters**

None

**CollectorTrackingEventList Property (OPCFilters Object)**

Returns a list of the Tracking Event Categories available for filtering on the Alarm Collector.

**Syntax**

```
object.CollectorTrackingEventList
```

**Parameters**

None

**CollectorType Property (Collector Object)**

Returns or sets Collector Type property for the Collector.

**Syntax**

object.**CollectorType***[ = String]*

**Parameters**

None

**Example**

```
' Print Out The CollectorType configured for the collector

 Debug.Print MyCollector.CollectorType
```

**CollectorType Property (Tag Object)**

Returns or sets the type of Collector responsible for collecting data for the Tag. Changes to tag properties are not committed until you call the WriteRecordset method of the TagRecordset object.

**Syntax**

object.**CollectorType***[ = String]*

**Parameters**

None

**CollectorType Property (TagCriteria Object)**

Sets the collector type to search for in the TagRecordset query.

**Syntax**

object.**CollectorType***[ = String]*

**Parameters**

None

**CollectorType Property (TagFields Object)**

Determines whether the CollectorType field should be returned in the TagRecordset query.

**Syntax**

object.**CollectorType***[ = Boolean]*

**Parameters**

None

**CollectorTypeList Property (Collectors Object)**

Returns a list of Collector Types available. These types may be used in other SDK functions.

**Syntax**

object.**CollectorTypeList**

**Parameters**

None

**Example**

'Get a List of the Available Collector Types Dim ListArray As Variant Dim I As Integer ListArray = MyCollectors.CollectorTypeList For I = LBound(ListArray) To UBound(ListArray) Debug.Print "Historian has a " + ListArray(I, 2) + " collector" Next I

**CollectorTypeSet Property (TagCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object..**CollectorTypeSet***[ = Boolean]*

**Parameters**

None

**CollectorTypeUpdated Property (Tag Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**CollectorTypeUpdated***[ = Boolean]*

**Parameters**

None

**Examples**

Body text goes here.

| Column Header | Column Header |
|---|---|
| Table text | Table text |

| Column Header | Column Header |
| --- | --- |
| Table text | Press Tab to add rows. |

### CollectorVersion Property (Collector Object)

Returns the collector version. This version information is available for collectors of version 5.0 and later

#### Syntax

`Object.CollectorVersion`

#### Parameters

None

#### Sample Code

```
Public ConnectedServer As iHistorian_SDK.Server

Dim MyCollector As iHistorian_SDK.Collector

Dim CollectorName As String

Dim ServerName As String

Private Sub BTNGetVersion_Click()

On Error GoTo errc

    If Not ConnectedServer Is Nothing Then

        ConnectedServer.Disconnect

        Set ConnectedServer = Nothing

        Set ConnectedServer = CreateObject("iHistorian_SDK.Server")

    Else

        Set ConnectedServer = CreateObject("iHistorian_SDK.Server")

    End If

    If Not ConnectedServer.Connect(ServerName, "", "") Then

        MsgBox "Connect To Server: " + ServerName + " Failed."

    Else

         Set MyCollector = ConnectedServer.Collectors.Item(CollectorName)

        If MyCollector.Running = True Then

            MsgBox "Collector version: " + MyCollector.CollectorVersion

        Else

            MsgBox "Collector version: Unknown"

        End If
```

```
    End If
Exit Sub
errc:
    MsgBox Err.Number
End Sub
```

**Comment Property (Collector Object)**

Returns or sets the Comment property for this collector.

**Syntax**

object.**Comment***[ = String]*

**Parameters**

None

**Comment Property (DataComments Object)**

Returns the comment associated with the DataValue.

**Syntax**

object.**Comment**

**Parameters**

None

**Remarks**

Comment is a read-only property of type Variant.

**Comment Property (Tag Object)**

Returns or sets the comment associated with the Tag. Changes to tag properties are not committed until you call the WriteRecordset method of the TagRecordset object.

**Syntax**

object.**Comment***[ = String]*

**Parameters**

None

**Comment Property (TagCriteria Object)**

Sets the comment to search for in the TagRecordset query.

**Syntax**

object.**Comment***[ = String]*

**Parameters**

None

**Comment Property (TagFields Object)**

Determines whether the Comment field should be returned in the TagRecordset query.

**Syntax**

object.**Comment***[ = Boolean]*

**Parameters**

None

**CommentSet Property (TagCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**CommentSet***[ = Boolean]*

**Parameters**

None

**CommentTimestamp Property (DataComments Object)**

Returns the time that the comment was added to the archive.

**Syntax**

object.**CommentTimestamp**

**Parameters**

None

**Remarks**

CommentTimestamp is a read-only property of type Date.

**Example**

```
Debug.Print "Comment added at: " + _   Format$(MyValue.Comme
```

### CommentSet Property (TagCriteria Object)

A flag to indicate whether this property has been set or not.

**Syntax**

object.**CommentSet***[ = Boolean]*

**Parameters**

None

### CommentUpdated Property (Tag Object)

A flag to indicate whether this property has been set or not.

**Syntax**

object.**CommentUpdated***[ = Boolean]*

**Parameters**

None

### Comments Property (DataFields Object)

Determines whether the Comments field should be returned in the DataRecordset query.

**Syntax**

object.**Comments***[ = Boolean]*

**Parameters**

None

### Comments Property (DataValue Object)

Returns the collection of comments that applies to the DataValue.

**Syntax**

object.**Comments**

**Parameters**

None

Remarks

Comments is a read-only property of type Collection.

**CommentsUpdated Property (DataValue Object)**

This property is for internal use only.

**Syntax**

object.**Comments**Updated

**Parameters**

None

**ComputerName Property (Collector Object)**

Returns or sets the name of the computer that the collector executes on.

**Syntax**

object.**ComputerName***[ = String]*

**Parameters**

None

**Remarks**

ComputerName is a read-only property of type String.

**ConditionCollectionCompareValue Property**

Returns or Contains compare value against the value of the triggertag.

**Syntax**

object.**ConditionCollectionCompareValue** [ = String]

**Parameters**

None

**ConditionCollectionCompareValue Property**

Returns or Contains compare value against the value of the triggertag.

**Syntax**

object.**ConditionCollectionCompareValue** [ = String]

**Parameters**

None

**ConditionCollectionComparison Property**

Contains the condition collection for a tag.

**Syntax**

object.**ConditionCollectionComparison** [ = Enum]

**Parameters**

None

**ConditionCollectionEnabled Property**

Sets the condition collection for a tag.

**Syntax**

object.**ConditionCollectionEnabled** [ = Boolean]

**Parameters**

None

**ConditionCollectionMarkers Property**

Returns or Sets if the end of the markers to be stored when the condition becomes FALSE. If FALSE, a bad data marker is not inserted when the condition becomes false.

**Syntax**

object.**ConditionCollectionMarkers** [ = Boolean]

**Parameters**

None

**ConditionCollectionTriggertag Property**

Contains a trigger tag from the list of tags associated with the collector.

**Syntax**

object.**ConditionCollectionTriggertag** [ = String]

**Parameters**

None

**Connected Property (Server Object)**

Returns the authentication status of the current server connection. True means you are connected to the server.

**Syntax**

object.**Connected**

**Parameters**

None

**Remarks**

Connected is a read-only property of type Boolean.

**ConnectionOptions Property (Server Object)**

Returns the collection of options for the current connection. To set individual options refer to the following table.

| TimeOption | Converts timestamps to and from the local time zone | String = Local |
|---|---|---|
| | Converts timestamps to and from the server time zone. | String = Server |
| | | |
| | Converts Timestamps To/From Explicit. The time-zone is specified in the TimeZoneBias option. | String = Explicit |

| TimeOption | Converts timestamps to and from the local time zone | String = Local |
|---|---|---|
| TimeZoneBias | Converts timestamps to and from a specific offset from GMT (minutes) | Long |
| AdjustDayLightSavings | Converts timestamps by adjusting for daylight savings | Boolean |

**Syntax**

object.**ConnectionOptions(OptionName)** ***[ = Variant]***

**Parameters**

| Name | Data Type | Description |
|---|---|---|
| OptionName | Variant | The name of a connection option |

None

**Remarks**

ConnectionOptions is a read-only property of type Collection of Options.

**Count Property (TagDependencies Object)**

This function returns the current number of Dependent Tags configured.

**Syntax**

object.**Count**

**Parameters**

None

**Count Property (DataStores Object)**

Gets the number of items in the collection.

**Syntax**

object.**Count**

**Parameters**

None

**Remarks**

Count is a read-only property of type Collection.

**Count Property (EnumeratedStates Object)**

Returns the number of items in the EnumeratedStates collection.

**Syntax**

object.**Count**

**Parameters**

None

**Remarks**

Long

**Criteria Property (DataRecordset Object)**

Returns the Criteria object used to identify which DataValues the Historian server returns when a DataRecordset query is executed.

The following procedure describes the steps involved in specifying criteria of a DataRecordset Query.

1. Specify the sampling mode.

   A sampling mode must be set to determine how values are retrieved from the Historian archive. Depending on the sampling mode, you may also need to modify the Direction and NumberOfSamples. The following table lists the available Sampling Modes and their explanations.

| Name | Description | Value |
|---|---|---|
| CurrentValue | Retrieves the current value. The time frame criteria are ignored. | 1 |
| Interpolated | Retrieves evenly spaced interpolated values based on NumberOfSamples and the time frame criteria. | 2 |

| Name | Description | Value |
|---|---|---|
| Trend | Retrieves the raw minimum and raw maximum value for each specified interval. Use the Trend sampling mode to maximize performance when retrieving data points for plotting. A start time, end time, and an interval or number of samples must be specified. | 3 |
| RawByTime | Retrieves raw archive values (compressed) based on time frame criteria. | 4 |
| RawByNumber | Retrieves raw archive values (compressed) based on the StartTime criteria, the NumberOfSamples, and Direction criteria.<br><br>NOTE: The EndTime criteria is ignored for this Sampling mode. | 5 |
| Calculated | Retrieves evenly spaced calculated values based on NumberOfSamples, the time frame criteria, and the CalculationMode criteria. | 6 |

2. Supply tags for query. You can specify tags in one of two ways:
    a. Supply a Tagmask that searches for all tags that match the mask and apply the remaining criteria to retrieve data. The mask can include wildcards.
    b. Supply an array of strings representing the list of tags to retrieve data for by using the Tags() property of the DataCriteria object.
    a. Supply time frame for query.

        To supply a time frame for the query, set the StartTime and EndTime properties or set the StartTimeShortcut and EndTimeShortcut properties.
3. Specify the Calculation Mode.

For the calculated SamplingMode, you must also specify the Calculation Mode.

The following table describes the available calculation modes.

**Table 270. Enumeration: ihCalculationMode**

| Name | Description | Value |
|---|---|---|
| Equal | Filter condition is True, when the FilterTag is equal to the comparison value. | 1 |
| NotEqual | Filter condition is True, when the FilterTag is NOT equal to the comparison value. | 2 |
| LessThan | Filter condition is True, when the FilterTag is less than the comparison value. | 3 |
| GreaterThan | Filter condition is True, when the FilterTag is greater than the comparison value. | 4 |
| LessThanEqual | Filter condition is True, when the FilterTag is less than or equal to the comparison value. | 5 |
| GreaterThanEqual | Filter condition is True, when the FilterTag is greater than or equal to the comparison value. | 6 |

The filter eliminates archive data from retrieval or calculations that occur within the time period that the specified filter criterion is True. FilterMode determines exactly how changes in state of the FilterTag are interpreted for retrieval of data.

| Name | Description | Value |
|---|---|---|
| ExactTime | Retrieves data for the exact times that the filter condition is True. | 1 |
| BeforeTime | Retrieves data from the time of the last False filter condition up to the time of the True condition. | 2 |

| Name | Description | Value |
|---|---|---|
| AfterTime | Retrieves data from the time of the True filter condition up to the time of next False condition. | 3 |
| BeforeAndAfterTime | Retrieves data from the time of the last False filter condition up to the time of next False condition. | 4 |

**Syntax**

object.**Count**

**Parameters**

None

**Remarks**

Criteria is a read-only property of type DataCriteria.

**Criteria Property (MessageRecordset Object)**

Use this property to specify which Messages to return from the Historian server when a MessageRecordset query executes.

A Topic may be optionally supplied to limit the Message queries to specific message topics. The following table provides a list of message topics.

| Name | Description | Value |
|---|---|---|
| Connections | Provides messages about client connections. | 1 |
| Security | Provides messages and alerts about login attempts and attempts to access restricted resources. | 2 |
| ConfigurationAudit | Provides messages about configuration changes and audit trail. | 3 |
| ServiceControl | Provides messages and alerts regarding startup and shutdown of the system and specific services. | 4 |

| Name | Description | Value |
|---|---|---|
| Performance | Provides messages and alerts regarding system and archive performance. | 5 |

You can also optionally supply a time frame for the query. To do this, either set the StartTime and EndTime properties or set the StartTimeShortcut and EndTimeShortcut properties, or a combination of both.

All other fields of the MessageCriteria are evaluated by exact match if set.

**Syntax**

object.**Criteria**

**Parameters**

None

**Remarks**

Criteria is a read-only property of type MessageCriteria.

**Criteria Property (TagRecordset Object)**

Use this property to specify which Tags to return from the Historian server when a TagRecordset query is executed.

You can supply the Tagname and Description as a mask including wildcards to return all matching tags. All other fields of the TagCriteria object are tested for exact match if they are supplied or set.

**Syntax**

object.**Criteria**

**Parameters**

None

**Remarks**

Criteria is a read-only property of type TagCriteria.

**CurrentValue Property (Tag Object)**

Returns the current value of the Tag as a DataValue object. The DataValue object contains properties for the Value, TimeStamp, and DataQuality of the current value.

**Syntax**

object.**CurrentValue**

**Parameters**

None

**Remarks**

CurrentValue is a read-only property of type DataValue.

**D**

**Data Property (Server Object)**

Returns the Data object for the current server. Use the Data object to build queries for collected data.

**Syntax**

object.**Data**

**Parameters**

None

**DataStores Property (Server Object)**

Returns the collection of the available data stores.

**Syntax**

object.**DataStores**

**Parameters**

None

**Remarks**

DataStores

**DataQuality Property (DataFields Object)**

Determines whether the DataQuality field should be returned in the DataRecordset query.

**Syntax**

object.**DataQuality*[ = Boolean]***

**Parameters**

None

### DataQuality Property (DataValue Object)

Returns or sets the quality of the DataValue.

The table below defines the possible values for DataQuality:

| Name | Data Type | Description |
|---|---|---|
| Good | The quality of the value is GOOD. | 1 |
| Bad | The quality of the value is BAD. | 2 |
| Unknown | The quality of the value is UN-KNOWN. | 3 |

**Syntax**

object.**DataQuality*[ = ihDataQuality]***

**Parameters**

None

### DataQualityUpdated Property (DataValue Object)

This property is for internal use only.

**Syntax**

object.**DataQualityUpdated***[ = Boolean]*

**Parameters**

None

### DataStore Property (Archive Object)

Returns the data store name of the specified Archive.

**Syntax**

object.DataStore

**Parameters**

None

**Remarks**

String

### DataQualityUpdated Property (DataValue Object)

This property is for internal use only.

**Syntax**

object.**DataQualityUpdated***[ = Boolean]*

**Parameters**

None

### DataTimestamp Property (DataComments Object)

Returns the timestamp of the DataValue the comment is associated with.

**Syntax**

object.DataTimestamp

**Parameters**

None

**Remarks**

DataTimeStamp is a read-only property of type Date.

**Example**

```
Debug.Print "Commented point at: " + _ Format$(MyValue.Com
```

### DataType Property (DataComments Object)

Returns the data type of the associated data.

**Syntax**

object.**DataType**

**Parameters**

None

### DataType Property (Tag Object)

Returns or sets the data type of the Tag. Changes to tag properties are not committed until you call the WriteRecordset method of the TagRecordset object.

The table below lists the available data types.

| Name | Description | Value |
|---|---|---|
| Scaled | Tag scaled between high engineering units and low engineering units. | 1 |
| Float | Tag stored as single precision floating point. | 2 |
| DoubleFloat | Tag stored as double precision floating point. | 3 |
| Integer | Tag stored as 2-byte integer. | 4 |
| DoubleInteger | Tag stored as 4-byte integer. | 5 |
| FixedString | Tag stored as fixed length UNICODE string. | 6 |
| VariableString | Tag stored as variable length UNICODE string. | 7 |
| Blob | Tag stored as binary large object. | 8 |

**Syntax**

object.**DataType***[ = ihDataType]*

**Parameters**

None

### DataType Property (DataComments Object)

Sets the data type to search for in the TagRecordset query.

**Syntax**

object.**DataType***[ = ihDataType]*

**Parameters**

None

**DataType Property (TagFields Object)**

Determines whether the DataType field should be returned in the TagRecordset query.

**Syntax**

object.**DataType** *= Boolean]*

**Parameters**

None

**DataTypeList Property (Tags Object)**

Returns the list of Historian data types.

**Syntax**

object.**DataTypeList**

**Parameters**

None

**Remarks**

DataTypeList is a read-only property of type Variant.

**DataTypeSet Property (TagCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**DataTypeSet*[ = Boolean]***

**Parameters**

None

**DataTypeUpdated Property (Tag Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**DataTypeUpdated*[ = Boolean]***

**Parameters**

None

**DefaultCollectionInterval Property (Collector Object)**

Returns or sets the Default Collection Interval property for the Collector.

**Syntax**

object.**DefaultCollectionInterval***[ = Long]*

**Parameters**

None

**Example**' Print Out The DefaultCollectionInterval configured for the collector Debug.Print

**DefaultCollectionInterval Property (Collector Object)**

Returns or sets the Default Collection Interval property for the Collector.

**Syntax**

object.**DefaultCollectionInterval***[ = Long]*

**Parameters**

None

**Example**

```
' Print Out The DefaultCollectionInterval configured for the collector Debug.Print
```

**DefaultCollectionType Property (Collector Object)**

Returns or sets the Default Collection Type for the given collector. This can be either Polled or Solicited.

**Syntax**

object.**DefaultCollectionType***[ = iHistorian_SDK.ihCollectionType]*

**Parameters**

None

**Example**

```
' Print Out The DefaultCollectionType configured for the collectorDebug.Print

MyCollector.DefaultCollectorAbsoluteDeadband
```

### DefaultCollectorAbsoluteDeadbanding Property (Collector Object)

Returns or sets the Default Collector Absolute Deadbanding for the Collector. This specifies if the compression applied to newly added tags from the collector use absolute or relative compression.

**Syntax**

object.**DefaultCollectorAbsoluteDeadbanding***[ = Boolean]*

**Parameters**

None

**Example**

```
' Print Out The DefaultCollectorAbsoluteDeadbanding configured for the collector DebugPr
```

### DefaultCollectorCompression Property (Collector Object)

Returns or sets whether the collector uses Compression by default.

**Syntax**

object.**DefaultCollectorCompression***[ = Boolean]*

**Parameters**

None

**Example**

```
' Print Out The DefaultCollectorCompression configured for the collector Debug.Print MyColl
```

### DefaultCollectorCompressionDeadband Property (Collector Object)

Returns or sets the Default Collector Compression Deadband for the given collector. This is default compression value to be applied to newly added tags from the collector.

**Syntax**

object.**DefaultCollectorCompressionDeadband***[ = Single]*

**Parameters**

None

**Example**

```
' Print Out The DefaultCollectorCompressionDeadband configured for the collector Debug.Print MyC
```

### DefaultCollectorCompressionTimeout Property (Collector Object)

Returns or sets the Default Collector Compression Timeout for the Collector. This specifies the amount of time after which a value will be sent to the archiver by the collector even if it would have been compressed.

**Syntax**

object.**DefaultCollectorCompressionTimeout***[ = Long]*

**Parameters**

None

**Example**

```
' Print Out The DefaultCollectorCompressionTimeout configured for the collector Debug
```

### DefaultLoadBalancing Property (Collector Object)

Returns or sets whether the collector uses Load Balancing by default.

**Syntax**

object.**DefaultLoadBalancing***[ = Boolean]*

**Parameters**

None

**Example**

```
' Print Out The DefaultLoadBalancing configured for the collector Debug.Print MyCo
```

### DefaultServer Property (ServerManager Object)

Returns or sets a reference to the default Server Object.

**Syntax**

object.**DefaultServer**

**Parameters**

None

### DefaultSpikeInterval Property (Collector Object)

Returns or sets the Default Spike Interval property for the Collector.

**Syntax**

object.**DefaultSpikeInterval***[ = Long]*

**Parameters**

None

**Example**

```
' Print Out The DefaultSpikeInterval configured for the collector Debug.Print

 MyCollector.DefaultSpikeInter
```

### DefaultSpikeLogic Property (Collector Object)

Returns or sets the Default Collector Spike Logic property for the Collector.

**Syntax**

object.**DefaultSpikeLogic***[ = Boolean]*

**Parameters**

None

**Example**

```
' Print Out The DefaultSpikeLogic configured for the collector Debug.Print

 MyCollector.DefaultSpikeLogic
```

### DefaultSpikeMultiplier Property (Collector Object)

Returns or sets the Default Spike Multiplier property for the Collector.

**Syntax**

object.**DefaultSpikeMultiplier***[ = Double]*

**Parameters**

None

**Example**

```
' Print Out The DefaultSpikeMultiplier configured for the collector Debug.Pri
```

### DefaultTagPrefix Property (Collector Object)

Returns or sets the Default Tag Prefix property for the Collector.

**Syntax**

object.**DefaultTagPrefix***[ = String]*

**Parameters**

None

**Example**

```
' Print Out The DefaultTagPrefix configured for the collector Debug
```

### DefaultTimestampType Property (Collector Object)

Returns or sets the Default Timestamp Type for the given collector. The Timestamps may be assigned by either the Data Source or the Collector itself.

**Syntax**

object.**DefaultTimestampType***[ = iHistorian_SDK.ihTimeStampType]*

**Parameters**

None

**Example**

```
' Print Out The DefaultTimestampType configured for the collector Debug.Print
 MyCollector.DefaultTimestampType
```

### Definition Property (UserCalcFunction Object)

Returns the definition of the specified UserCalcFunction.

**Syntax**

object.**Definition***[ = String]*

**Parameters**

None

### Description Property (Collector Object)

Returns or sets the description for the given collector.

**Syntax**

object.**Description***[ = String]*

**Parameters**

None

**Example**

```
' Print Out The Description configured for the collector Debug.P
```

### Description Property (EnumeratedSets Object)

Returns or sets the description or comments for an enumerated set.

**Syntax**

object.**Description**

**Parameters**

None

**Returns**

String

### Description Property (EnumeratedState Object)

Returns or defines the description or comments for a state.

**Syntax**

object.**Description**

**Parameters**

None

**Remarks**

String. Returns the description or comments for each state.

**Description Property (Tag Object)**

Returns or sets the description of the Tag. Changes to tag Properties are not committed until you call the WriteRecordset method of the TagRecordset object.

**Syntax**

object.**Description*[ = String]***

**Parameters**

None

**Description Property (TagCriteria Object)**

Sets the description or description mask to search for in the TagRecordset query. The description may include wildcard characters.

**Syntax**

object.**Description*[ = String]***

**Parameters**

None

**Description Property (TagFields Object)**

Determines whether the Description field should be returned in the TagRecordset query.

**Syntax**

object.**Description*[= Boolean]***

**Parameters**

None

**Description Property (UserDefinedTypeFields Object)**

Returns or sets the description or comments for a field in a User Defined Type.

**Syntax**

object.**Description**

**Parameters**

None

**Returns**

String

**DescriptionSet Property (TagCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**DescriptionSet***[ = Boolean]*

**Parameters**

None

**DescriptionUpdated Property (Tag Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**DescriptionUpdated***[ = Boolean]*

**Parameters**

None

**Direction Property (DataCriteria Object))**

Sets the direction (forward or backward) of sampling data from the archive by the DataRecordset query. This parameter is only used for "RawByNumber" Sampling Mode.

The following table details the available sampling directions.

| Name | Description | Value |
|---|---|---|
| Forward | Data sampled from the start time forward in time. | 1 |
| Backward | Data sampled from the start time backwards in time. | 2 |

**Syntax**

object.**Direction***[ = iHistorian_SDK.ihSamplingDirection]*

**Parameters**

None

**DirectionSet Property (DataCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**DescriptionSet*[ = Boolean]***

**Parameters**

None

## Property Reference E-F

**EndTime Property (Archive Object)**

Returns the end time of the specified archive. This represents the latest timestamp for any tag contained in the archive.

**Syntax**

object.**EndTime**

**Parameters**

None

**Remarks**

EndTime is a read-only property of type Date.

**EndTime Property (DataCriteria Object)**

Returns or sets the end of the time range to retrieve data for in the DataRecordset query.

**Syntax**

object.**EndTime***[ = Date]*

**Parameters**

None

**EndTime Property (MessageCriteria Object)**

Establishes the end time of the MessageRecordset query.

**Syntax**

object.**EndTime***[ = Date]*

**Parameters**

None

**EndTimeSet Property (DataCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**EndTimeSet***[ = Boolean]*

**Parameters**

None

**EndTimeSet Property (MessageCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**EndTimeSet***[ = Boolean]*

**Parameters**

None

**EndTimeShortcut Property (DataCriteria Object)**

Returns or sets the end of the time range to retrieve data for in the DataRecordset query. Shortcuts are strings that define absolute or relative times. The time shortcuts are the following:

| Value | Description |
|---|---|
| (N)ow | The current time (absolute). |
| (T)oday | Today at midnight (absolute). |
| (Y)esterday | Yesterday at midnight (absolute). |
| (D)ays | Number of Days (relative). |

| Value | Description |
|---|---|
| (M)in | Number of Minutes (relative). |
| (H)our | Number of Hours (relative). |
| (W)eek | Number of Weeks (relative). |
| (BOM) | Beginning of this month at Midnight (absolute). |
| (EOM) | Last Day of this month at Midnight (absolute). |
| (BOY) | First Day of this year at Midnight (absolute). |
| (EOY) | Last Day of this year at Midnight (absolute). |

**Syntax**

object.**EndTimeShortcut***[ = String]*

**Parameters**

None

**Example**

```
With MyRecordset.Criteria   .Tagmask = "*.F_CV"   ' Start two days Ago at 8am in the morning

.StartTimeShortcut = "Today-2d+8h"   ' End this morning at 8am   .EndTimeShortcut = "Today+8h" End With
```

**EndTimeShortcut Property (MessageCriteria Object)**

Establishes the end time of the MessageRecordset query by specifying a time shortcut string versus a date. The time shortcuts are the following:

| Value | Description |
|---|---|
| (N)ow | The current time (absolute). |
| (T)oday | Today at midnight (absolute). |
| (Y)esterday | Yesterday at midnight (absolute). |
| (D)ays | Number of Days (relative). |
| (M)in | Number of Minutes (relative). |
| (H)our | Number of Hours (relative). |
| (W)eek | Number of Weeks (relative). |

| Value | Description |
|---|---|
| (BOM) | Beginning of this month at Midnight (absolute). |
| (EOM) | Last Day of this month at Midnight (absolute). |
| (BOY) | First Day of this year at Midnight (absolute). |
| (EOY) | Last Day of this year at Midnight (absolute). |

**Syntax**

object.**EndTimeShortcut***[ = String]*

**Parameters**

None

**Example**

```
' Set A End Time For Now MyMessages.Criteria.EndTimeShortcut = "Now"
```

**EndTimeShortcutSet Property (MessageCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**EndTimeShortcutSet***[ = Boolean]*

**Parameters**

None

**EngineeringUnits Property (Tag Object)>**

Returns or sets the engineering unit description of the Tag. Changes to tag properties are not committed until you call the WriteRecordset method of the TagRecordset object.

**Syntax**

object.**EngineeringUnits***[ = String]*

**Parameters**

None

**EngineeringUnits Property (TagCriteria Object)**

Sets the engineering unit description to search for in the TagRecordset query.

**Syntax**

object.**EngineeringUnits***[ = String]*

**Parameters**

None

**EngineeringUnits Property (TagFields Object)**

Determines whether the EngineeeringUnits field should be returned in the TagRecordset query.

**Syntax**

object.**EngineeringUnits***[ = Boolean]*

**Parameters**

None

**EngineeringUnitsSet Property (TagCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**EngineeringUnits***[ = Boolean]*

**Parameters**

None

**EngineeringUnitsUpdated Property (Tag Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**EngineeringUnitsUpdated***[ = Boolean]*

**Parameters**

None

**ErrorList Property (Server Object)**

Returns a collection of error messages accumulated during the current session of the Server object. Each message includes a timestamp indicating the time the error was generated and each includes an error message translated into the locale of the client, when possible.

**Syntax**

object.**ErrorList**

**Parameters**

None

**Remarks**

ErrorList is a read-only property of type Collection of String.

**Example**

```
ErrorTrap: Dim I As Long ' Loop through the ErrorList and print to the debug window
For I = 1 To MyServer.ErrorList.count Debug.Print MyServer.ErrorList(I) Next I
```

## F

### FailoverOnBadQuality Property (Collector Object)

Returns or sets whether the collector should automatically failover when the watchdog tag quality goes to BAD.

**Syntax**

object.**FailoverOnBadQuality***[ = Boolean]*

**Parameters**

None

### FailoverOnCollectorStatus Property (Collector Object)

Returns or sets whether the collector should automatically failover when the currently active collector status goes to UNKNOWN.

**Syntax**

object.**FailoverOnCollectorStatus***[ = Boolean]*

**Parameters**

None

**FailoverOnNonZeroValue Property (Collector Object)**

Returns or sets whether the collector should automatically failover when the watchdog tag value goes to non-zero.

**Syntax**

object.**FailoverOnNonZeroValue***[ = Boolean]*

**Parameters**

None

**FailoverOnValue Property (Collector Object)**

Returns or sets whether the collector should automatically failover based on the value of the watchdog tag. Set the FailoverOnNonZeroValue property or FailoverOnValueUnchanged property to indicate the exact criteria.

**Syntax**

object.**FailoverOnValue***[ = Boolean]*

**Parameters**

None

**FailoverOnValueUnchanged Property (Collector Object)**

Returns or sets whether the collector should automatically failover when the watchdog tag value remains unchanged. Use the WatchdogValueMaxUnchangedPeriod property to specify the time period.

**Syntax**

object.**FailoverOnValueUnchanged***[ = Boolean]*

**Parameters**

None

**Fields Property (DataRecordset Object)**

Returns the DataFields object which contains the information on which fields are returned in the DataRecordSet

**Syntax**

object.**Fields**

**Parameters**

None

**Fields Property (MessageRecordset Object)**

Returns the Fields object that is used to identify which Message fields to return from the Historian server when a MessageRecordset query executes.

**Syntax**

object.**Fields**

**Parameters**

None

**Remarks**

Fields is a read-only property of type MessageFields.

**Fields Property (TagRecordset Object)**

Returns the Fields object that is used to identify which Tag fields to return from the Historian server when a TagRecordset query executes.

**Syntax**

object.**Fields**

**Parameters**

None

**Remarks**

Fields is a read-only property of type TagFields.

**FileName Property (Archive Object)**

Returns or sets the file name of the specified archive. The filename must be specified in the context of the Historian server drives and directories.

Changing the file name of an archive moves the archive from one file location to another.

**Syntax**

object.**FileName***[ = String]*

**Parameters**

None

**Example**

```
Dim I As Long ' List The FileName and FileSize Of Each Archive With MyArchives
For I = 1 To .Item.count     Debug.Print .Item(I).Name, .Item(I).FileName, .Item(I).FileSizeCurrent
Next I End With ' Move The Archive To A New Directory,
Path Context Of Server On Error GoTo ErrorTrap MyArchive.FileName = NewArchiveFilename ErrorTrap:
Debug.Print "
```

### FileSizeCurrent Property (Archive Object)

Returns the number of MB used in the specified archive file. This number is less than or equal to the file size on disk.

Changing the file name of an archive moves the archive from one file location to another.

**Syntax**

object.**FileSizeCurrent**

**Parameters**

None

**Remarks**

FileSizeCurrent is a read-only property of type Long.

**Example**

```
Dim I As Long ' List The FileName and FileSizeCurrent Of Each Archive With MyArchives
For I = 1 To .Item.count     Debug.Print .Item(I).Name, .Item(I).FileName, .Item(I).FileSizeCurrent
Next I End With
```

### FileSizeTarget Property (Archive Object)

Returns or sets the target file size for the specified archive file in MB.

**Syntax**

object.**FileSizeTarget***[ = Long]*

**Parameters**

None

**Remarks**

FileSizeCurrent is a read-only property of type Long.

**Example**

```
Debug.Print "Target File Size for archive " & MyArchive.FileSizeTarget
```

### FilterComparisonMode Property (DataCriteria Object)

Returns or sets the type of comparison to be made on the FilterComparisonValue to apply appropriate filter to the DataRecordset query. If a FilterTag and FilterComparisonValue are supplied the query attempts to filter out time periods from the results where the filter condition is False. The FilterComparisonMode defines how archive values for the FilterTag should be compared to the FilterComparisonValue to establish the state of the filter condition.

The table below defines the available Filter Comparison Modes:

| Value | Description | Value |
|---|---|---|
| Equal | Filter condition is True when the FilterTag is equal to the comparison value. | 1 |
| NotEqual | Filter condition is True when the FilterTag is NOT equal to the comparison value. | 2 |
| LessThan | Filter condition is True when the FilterTag is less than the comparison value. | 3 |
| GreaterThan | Filter condition is True when the FilterTag is greater than the comparison value. | 4 |

| Value | Description | Value |
|---|---|---|
| LessThanEqual | Filter condition is True when the FilterTag is less than or equal to the comparison value. | 5 |
| GreaterThanEqual | Filter condition is True when the FilterTag is greater than or equal to the comparison value. | 6 |

**Syntax**

object.**FilterComparisonMode***[ = iHistorian_SDK.ihFilterComparisonMode]*

**Parameters**

None

**FilterComparisonModeList Property (Data Object)**

This function returns a list of the available Filter Comparison modes for a data request.

**Syntax**

object.**FilterComparisonModeList**

**Parameters**

None

**FilterComparisonModeSet Property (DataCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**FilterComparisonModeSet***[ = Boolean]*

**Parameters**

None

**FilterComparisonValue Property (DataCriteria Object)**

Sets the value to compare the filter tag with when applying the appropriate filter to the DataRecordset query.

**Syntax**

object.**FilterComparisonValue***[ = Variant]*

**Parameters**

None

**FilterComparisonValue Property (DataCriteria Object)**

Sets the value to compare the filter tag with when applying the appropriate filter to the DataRecordset query.

**Syntax**

object.**FilterComparisonValue***[ = Variant]*

**Parameters**

None

**FilterComparisonValueSet Property (DataCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**FilterComparisonValueSet***[ = Boolean]*

**Parameters**

None

**FilterMode Property (DataCriteria Object)**

Returns or sets the type of time filter applied to the DataRecordset query. If a FilterTag and FilterComparisonValue are supplied the query attempts to filter time periods from the results where the filter condition equals False. The FilterMode defines how time periods before and after transitions in the filter condition should be handled. For example, "AfterTime" indicates that the filter condition should be True starting at the timestamp of the archive value that triggered the True condition and leading up to the timestamp of the archive value that triggered the False condition.

Use the filter query to directly retrieve data for a particular lot number or batch of material. In this scenario, you must know whether the batch number was written into the archive at the beginning of the batch or at the end of the batch. If the batch number is written at the

beginning of the batch, use the "AfterTime" filter mode. If the batch number is written at the end of the batch, use the "BeforeTime" filter mode.

The table below defines the available Filter Modes:

| Name | Description | Value |
|---|---|---|
| ExactTime | Retrieves data for the exact times that the filter condition is True. | 1 |
| BeforeTime | Retrieves data from the time of the last False filter condition up until the time of the True condition. | 2 |
| AfterTime | Retrieves data from the time of the True filter condition up until the time of next False condition | 3 |
| BeforeAndAfterTime | Retrieves data from the time of the last False filter condition up until the time of next False condition. | 4 |

**Syntax**

object.**FilterMode***[ = iHistorian_SDK.ihFilterMode]*

**Parameters**

None

**FilterModeList Property (Data Object)**

This function returns a list of the available Filtering modes for a data request.

**Syntax**

object.**FilterModeList**

**Parameters**

None

**FilterModeSet Property (DataCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**FilterModeSet***[ = Boolean]*

**Parameters**

None

**FilterTag Property (DataCriteria Object)**

Returns or sets the tagname used to define the filter applied to the DataRecordset query. You can only specify a single tag (no wildcards).

**Syntax**

object.**FilterTag***[ = String]*

**Parameters**

None

**FilterTagSet Property (DataCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**FilterTagSet***[ = Boolean]*

**Parameters**

None

**FullAreaNames Property (OPCBrowse Object)**

Returns the fully-qualified area name for an AE server. When a browse operation occurs, the server returns all areas and sources under the BrowsePosition. The areas are akin to nodes (they can be set to the next browse position to traverse the tree). Sources are like leaves, and do not have children.

After a browse, the areas and leaves are populated in arrays, and the LeafName and AreaName properties allow you to access those arrays by index. The fully-qualified area name should be used to set the Browse Position. The Areanames property is used to display the individual areas.

**Syntax**

object.**FullAreaNames**(Index)

**Parameters**

Index - Integer - The index into the full area name array.

**Remarks**

FullAreaNames is a read-only String property returned as a variant for script compatibility.

**FullLeafNames Property (OPCBrowse Object)**

Returns the fully-qualified leaf name (source) for an AE server. When a browse operation occurs, the server returns all areas and sources under the BrowsePosition. The areas are akin to nodes (they can be set to the next browse position to traverse the tree). Sources are like leaves, and do not have children.

After a browse, the areas and leaves are populated in arrays, and the LeafName and AreaName properties allow you to access those arrays by index.

**Syntax**

object.**FullLeafNames**(Index)

**Parameters**

Index - Integer - The index into the full leaf name array.

**Remarks**

FullLeafNames is a read-only String property returned as a variant for script compatibility.

## Property Reference G-H

**G**

**General1 Property (Collector Object)**

Returns or sets the general fields of the specified collector. The general fields control the behavior of specific collector types. Refer to the documentation of a specific collector type to interpret the meaning of the general collector fields for that collector type.

**Syntax**

object.**General1***[ = String]*

**Parameters**

None

**Example**

```
MyCollector.General1 = "CanUseOPCTime=True"
```

### General2 Property (Collector Object)

Returns or sets the general fields of the specified collector. The general fields control the behavior of specific collector types. Refer to the documentation of a specific collector type to interpret the meaning of the general collector fields for that collector type.

**Syntax**

object.**General2***[ = String]*

**Parameters**

None

### General3 Property (Collector Object)

Returns or sets the general fields of the specified collector. The general fields control the behavior of specific collector types. Refer to the documentation of a specific collector type to interpret the meaning of the general collector fields for that collector type.

**Syntax**

object.**General3***[ = String]*

**Parameters**

None

### General4 Property (Collector Object)

Returns or sets the general fields of the specified collector. The general fields control the behavior of specific collector types. Refer to the documentation of a specific collector type to interpret the meaning of the general collector fields for that collector type.

**Syntax**

object.**General4***[ = String]*

**Parameters**

None

### General5 Property (Collector Object)

Returns or sets the general fields of the specified collector. The general fields control the behavior of specific collector types. Refer to the documentation of a specific collector type to interpret the meaning of the general collector fields for that collector type.

**Syntax**

object.**General4***[ = String]*

**Parameters**

None

## H

### Handle Property (Server Object)

Returns the internal identifier for the current server.

**Syntax**

object.**Handle**

**Parameters**

None

**Remarks**

Handle is a read-only property of type Long.

### HeartbeatOutputAddress Property (Collector Object)

Returns or sets the address in the data source that the collector sends heartbeat information to. The collector will write a 1 to this address once per minute.

**Syntax**

object.**HeartbeatOutputAddress***[ = String]*

**Parameters**

None

**Example**

```
'Print Out The Heartbeat Output Address configured for the collector DebugPrint

 MyCollector.HeartbeatOutputAddress
```

**HiEngineeringUnits Property (Tag Object)**

Returns or sets the high engineering unit range of the Tag. Changes to tag properties are not committed until you call the WriteRecordset method of the TagRecordset object.

**Syntax**

object.**HiEngineeringUnits***[ = Double]*

**Parameters**

None

**HiEngineeringUnits Property (TagCriteria Object)**

Sets the high engineering unit range to search for in the TagRecordset query.

**Syntax**

object.**HiEngineeringUnits***[ = Double]*

**Parameters**

None

**HiEngineeringUnits Property (TagFields Object)**

Determines whether the HiEngineeringUnits field should be returned in the TagRecordset query.

**Syntax**

object.**HiEngineeringUnits***[ = Boolean]*

**Parameters**

None

**HiEngineeringUnitsSet Property (TagCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**HiEngineeringUnitsSet***[ = Boolean]*

**Parameters**

None

**HiEngineeringUnitsUpdated Property (Tag Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**HiEngineeringUnitsUpdated***[ = Boolean]*

**Parameters**

None

**HiScale Property (Tag Object)**

Sets the high scale value used for input scaling on the Tag. Changes to tag properties are not committed until you call the WriteRecordset method of the TagRecordset object.

**Syntax**

object.**HiScale***[ = String]*

**Parameters**

None

**HiScale Property (TagCriteria Object)**

Sets the high input scale range value to search for in the TagRecordset query.

**Syntax**

object.**HiScale***[ = String]*

**Parameters**

None

**HiScale Property (TagFields Object)**

Determines whether the HiScale field should be returned in the TagRecordset query.

**Syntax**

object.**HiScale***[ = Boolean]*

**Parameters**

None

**HiScaleSet Property (TagCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**HiScaleSet***[ = Boolean]*

**Parameters**

None

**HiScaleUpdated Property (Tag Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**HiScaleUpdated***[ = Boolean]*

**Parameters**

None

**HighPart Property (QueryModifiers Object)**

Returns or sets the HighPart of the query modifier. Use the Server.CriteriaFromStrings method to determine the HighPart and LowPart of a query modifier.

**Syntax**

object.**HighPart**

**Parameters**

None

**Remarks**

Long

**HighSeverity Property (OPCFilters Object)**

Gets/Sets the High Severity filter in the Alarm Collector. Only events with a severity less than or equal to HighSeverity will be collected.

**Syntax**

object.**HighSeverity***[ = Long]*

**Parameters**

None

# Property Reference I-J

**I**

**Id Property (DataStore Object)**

Returns or sets the ID (GUID) of a data store. This is a read-only property.

**Syntax**

object.**Id**

**Parameters**

None

**Remarks**

String. This is a read-only property.

**ImportErrors Property (Alarms Object)**

Returns a list of Import Error messages.

**Syntax**

object.**ImportErrors**

**Parameters**

None

**ImportErrors Property (DataRecordset Object)**

Returns a list of Import Error messages.

**Syntax**

object.**ImportErrors**

**Parameters**

None

**ImportErrors Property (MessageRecordset Object)**

Returns a list of Import Errors.

**Syntax**

object.**ImportErrors**

**Parameters**

None

**ImportErrors Property (TagRecordset Object)**

Returns a list of tag import errors.

**Syntax**

object.**ImportErrors**

**Parameters**

None

**InputScaling Property (Tag Object)**

Returns or sets whether input scaling is enabled for the Tag. Changes to tag properties are not committed until you call the WriteRecordset method of the TagRecordset object.

**Syntax**

object.**InputScaling***[ = Boolean]*

**Parameters**

None

**InputScaling Property (TagCriteria Object)**

Sets the input scaling to search for in the TagRecordset query.

**Syntax**

object.**InputScaling***[ = Boolean]*

**Parameters**

None

**InputScaling Property (TagFields Object)**

Determines whether the InputScaling field should be returned in the TagRecordset query.

**Syntax**

object.**InputScaling***[ = Boolean]*

**Parameters**

None

**InputScalingSet Property (TagCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**InputScalingSet***[ = Boolean]*

**Parameters**

None

**InputScalingUpdated Property (Tag Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**InputScalingUpdated***[ = Boolean]*

**Parameters**

None

**IsSystem Property (DataStore Object)**

Indicates or sets that data store is the single-system data store and therefore, cannot be modified.

**Syntax**

object.**IsSystem**

**Parameters**

None

**InterfaceAbsoluteDeadband Property (Tag Object)**

Returns or sets the Absolute Deadband value for this tag to be used in the Collector.

**Syntax**

object.**InterfaceAbsoluteDeadband***[ = Double]*

**Parameters**

None

**InterfaceAbsoluteDeadband Property (TagCriteria Object)**

Sets the InterfaceAbsoluteDeadband to search for in the TagRecordset query.

**Syntax**

object.**InterfaceAbsoluteDeadband***[ = Double]*

**Parameters**

None

**InterfaceAbsoluteDeadband Property (TagFields Object)**

Determines whether the InterfaceAbsoluteDeadband field should be returned in the TagRecordset query.

**Syntax**

object.**InterfaceAbsoluteDeadband***[ = Double]*

**Parameters**

None

**InterfaceAbsoluteDeadbandSet Property (TagCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**InterfaceAbsoluteDeadbandSet***[ = Boolean]*

**Parameters**

None

**InterfaceAbsoluteDeadbandUpdated Property (Tag Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**InterfaceAbsoluteDeadbandUpdated***[ = Boolean]*

**Parameters**

None

**InterfaceAbsoluteDeadbanding Property (Tag Object)**

Returns or sets whether this tag is using Absolute Deadbanding in the Collector or not.

**Syntax**

object.**InterfaceAbsoluteDeadbanding***[ = Boolean]*

**Parameters**

None

**InterfaceAbsoluteDeadbanding Property (TagCriteria Object)**

Sets the InterfaceAbsoluteDeadbanding to search for in the TagRecordset query.

**Syntax**

object.**InterfaceAbsoluteDeadbanding***[ = Boolean]*

**Parameters**

None

**InterfaceAbsoluteDeadbanding Property (TagFields Object)**

Determines whether the InterfaceAbsoluteDeadbanding field should be returned in the TagRecordset query.

**Syntax**

object.**InterfaceAbsoluteDeadbanding***[ = Boolean]*

**Parameters**

None

**InterfaceAbsoluteDeadbandingSet Property (TagCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**InterfaceAbsoluteDeadbandingSet***[ = Boolean]*

**Parameters**

None

**InterfaceAbsoluteDeadbandingUpdated Property (Tag Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**InterfaceAbsoluteDeadbandingUpdated***[ = Boolean]*

**Parameters**

None

**IsActiveRedundantCollector Property (Collector Object)**

Returns whether this collector is currently the active collector in a redundant set of collectors.

**Syntax**

object.**IsActiveRedundantCollector**

**Parameters**

None

**Remarks**

IsActiveRedundantCollector is a read-only property of type Boolean.

**IsAudited Property (Data Object)**

Returns whether or not this data has an Audit Trail associated with it.

**Syntax**

object.**IsAudited**

**Parameters**

None

### IsAudited Property (Server Object)

Returns or sets whether the point verification feature is enabled. With point verification enabled, the system will prompt for a username and password upon any request to change data or configuration.

**Syntax**

object.**IsAudited***[ = Boolean]*

**Parameters**

None

### IsCurrent Property (Archive Object)

Returns whether or not the specified archive is the newest archive where new data currently flows into.

**Syntax**

object.**IsCurrent**

**Parameters**

None

**Remarks**

IsCurrent is a read-only property of type Boolean.

Example

```
'Find The Current Archive, Then Initiate A Backup With MyArchives

For I = 1 To .Item.count If .Item(I).IsCurrent Then

If Not .Item(I).Backup(BackupFilename) Then err.Raise 1, "Backup", "Backup Failed" End If

End If   Next I End With
```

**IsDefaultServer Property (Server Object)**

Returns and/or determines if the current server is the default server. You can create an instance of the Server object and connect it without supplying a server name, username, or password. In this case, the default server and associated user information establishes the default connection.

**Syntax**

object.**IsDefaultServer**

**Parameters**

None

**IsDeleted Property (DataValue Object)**

This function returns a Boolean which indicates whether or not this DataValue has been deleted or not.

**Syntax**

object.**IsDeleted**

**Parameters**

None

**IsDeleted Property (Tag Object)**

Returns whether or not this tag has been deleted.

**Syntax**

object.**IsDeleted**

**Parameters**

None

**Modified Property (DataValue Object)**

This function returns Boolean which indicates whether or not any of this DataValue's fields have been modified.

**Syntax**

object.**IsModified**

**Parameters**

None

**IsModified Property (Tag Object)**

Returns whether or not this tag has been modified.

**Syntax**

object.**IsModified**

**Parameters**

None

**IsNew Property (DataComments Object)**

Returns whether or not this Comment is new or not.

**Syntax**

object.**IsNew**

**Parameters**

None

**IsNew Property (DataValue Object)**

This function returns a Boolean which indicates whether or not this DataValue is new.

**Syntax**

object.**IsNew**

**Parameters**

None

**IsNew Property (Message Object)**

Returns whether this is a new message or not.

**Syntax**

object.**IsNew**

**Parameters**

None

**IsNew Property (Tag Object)**

Tag.IsNew indicates whether this is a newly created tag. Returns true if the tag's name does not exist in the server.

**Syntax**

object.**IsNew**

**Parameters**

None

**Item Property (Archives Object)**

Returns a specific Archive object contained in the Archives object. The Item collection can loop through all Archives, or reference a specific Archive object directly by archive name.

The Archive Item collection automatically populates when the Archives object is instantiated.

**Syntax**

object.**Item**

**Parameters**

None

**Remarks**

Item is a read-only property of type Collection of Archives.

**Item Property (Collectors Object)**

Returns a specific Collector object contained in the Collectors Object collection. The Item collection can loop through all Collectors, or reference a specific Collector object directly by Collector name.

The Collector Item collection automatically populates when the Collectors object is instantiated.

**Syntax**

object.**Item**

**Parameters**

None

**Remarks**

Item is a read-only property of type Collection.

### Item Property (DataRecordset Object)

Returns a specific DataValue object contained in the DataRecordset object. The Item collection can loop through all DataValues, or reference a specific DataValue object directly by Tagname and Timestamp.

The DataRecordset (and thus the Item collection) is populated with DataValues in three ways. The first is to set up query criteria and call the QueryRecordset method. The second is to import a list of DataValues using the Import method. The last way is to use the Add method to add new DataValues.

**Syntax**

object.**Item**(Tagname)

**Parameters**

Tagname - Variant - The name of the tag whose values to retrieve.

**Remarks**

Item is a read-only property of type Collection of DataValue.

### Item Property (DataStores Object)

Returns a specific data store object contained in the DataStores Object collection. The Item collection can loop through all DataStores, or reference a specific DataStore object directly by DataStore name.

**Syntax**

object.Item() As Collection

**Parameters**

None

**Remarks**

Item is a read-only property of type Collection.

**Item Property (EnumeratedStates Object)**

Returns an individual item from the EnumeratedStates collection.

**Parameters**

None

**Remarks**

Enumerated State.

**Item Property (MessageRecordset Object)**

Returns a specific Message object contained in the MessageRecordset object. The Item collection can loop through all Messages or reference a specific Message object directly by the timestamp.

The MessageRecordset (and thus the Item collection) populates with the Message in two ways: set up query criteria and call the QueryRecordset method, or use the Add method to add new Messages. Item is a read-only property of type Collection of Message.

**Syntax**

object.**Item**

**Parameters**

None

**Item Property (TagDependencies Object)**

This function attempts to find the given Tagname within the collection of currently configured tags.

**Syntax**

object.**Item**(Tagname)

**Parameters**

Tagname - Variant - Name of the tag to locate.

**Item Property (TagRecordset Object)**

Returns a specific Tag object contained in the TagRecordset object. The Item collection can loop through all Tags, or reference a specific Tag object directly by the tag name.

The TagRecordset (and thus the Item collection) populates with Tags in three ways:

The first is to set up query criteria and call the QueryRecordset method. The second is to import a list of Tags using the Import method. The last way is to use the Add method to add new Tags.

**Syntax**

object.**Item**

**Parameters**

None

**Remarks**

Item is a read-only property of type Collection of Tag.

**Item Property (UserDefinedTypeFields Object)**

Returns an individual item from the multiple fields of the User Defined Type.

**Syntax**

object.item(Index)

**Parameters**

Index - Variant - The index has to be returned.

**Remarks**

UserDefinedTypeField.

# Property Reference K-L

## L

### LastBackup Property (Archive Object)

Returns the date and time of the last archive backup.

**Syntax**

object.**LastBackup**

**Parameters**

None

**Remarks**

`LastBackup` is a read-only property of the type Date.

Example

```
Dim I As Long ' List The Start and End Time Of Each Archive With MyArchives For I = 1 To .Item.count
Debug.Print .Item(I).Name, .Item(I).StartTime, .Item(I).LastBackup Next I End With
```

### LastBackupUser Property (Archive Object)

Returns username of the person who performed the last backup of the archive.

**Syntax**

object.**LastBackupUser**

**Parameters**

None

**Remarks**

LastBackupUser is a read-only property of type String.

Example

```
Debug.Print "Modified By: " + MyArchive.LastBackupUser
```

### LastError Property (Archive Object)

Returns the last error message encountered by the Archive object. To see a complete list of messages, refer to the ErrorList property. When possible, the system translates messages into the locale of the client.

**Syntax**

object.**LastError**

**Parameters**

None

**Remarks**

LastError is a read-only property of type String.

**LastError Property (Archives Object)**

Returns the last error message encountered by the Archives object. To see a complete list of messages, refer to the ErrorList property. When possible, the system translates messages into the locale of the client.

**Syntax**

object.**LastError**

**Parameters**

None

**Remarks**

LastError is a read-only property of type String.

**LastError Property (Collector Object)**

Returns the last error message encountered by the Collector object. To see a complete list of messages, refer to the ErrorList property. When possible, the system translates messages into the locale of the client.

**Syntax**

object.**LastError**

**Parameters**

None

**Remarks**

LastError is a read-only property of type String.

**LastError Property (Collectors Object)**

Returns the last error message encountered by the Collectors Object. To see a complete list of messages, refer to the ErrorList property. When possible, the system translates messages into the locale of the client.

**Syntax**

object.**LastError**

**Parameters**

None

**Remarks**

LastError is a read-only property of type String.

**LastError Property (DataRecordset Object)**

Returns the last error message encountered by the DataRecordset object. To see a complete list of messages, refer to the ErrorList property. When possible, the system translates messages into the locale of the client.

**Syntax**

object.**LastError**

**Parameters**

None

**Remarks**

LastError is a read-only property of type String.

**LastError Property (DataStores Object)**

Returns the last error message encountered by the Collectors Object.

**Syntax**

object.**LastError**

**Parameters**

None

**Remarks**

LastError is a read-only property of type String.

**LastError Property (EnumeratedSets Object)**

Returns the last error message encountered by the EnumeratedSets object. To see a complete list of messages, refer to the ErrorList Property. When possible, the system translates messages into the locale of the client.

**Syntax**

object.**LastError()**

**Parameters**

None

**Remarks**

String. Indicates whether the string contains an error or not. This is a read-only property.

**LastError Property (EnumeratedSets Object)**

Returns the last error message encountered by the EnumeratedSets object. To see a complete list of messages, refer to the ErrorList Property. When possible, the system translates messages into the locale of the client.

**Syntax**

object.**LastError()**

**Parameters**

None

**Remarks**

String. Indicates whether the string contains an error or not. This is a read-only property.

**LastError Property (Messages Object)**

Returns the last error message encountered by the EnumeratedSets object. To see a complete list of messages, refer to the ErrorList Property. When possible, the system translates messages into the locale of the client.

**Syntax**

object.**LastError()**

**Parameters**

None

**Remarks**

`LastError` is a read-only property of type String.

**LastError Property (Messages Object)**

Returns the last error message encountered by the EnumeratedSets object. To see a complete list of messages, refer to the ErrorList Property. When possible, the system translates messages into the locale of the client.

**Syntax**

object.**LastError**

**Parameters**

None

**Remarks**

`LastError` is a read-only property of type String.

**LastError Property (Server Object)**

Returns the last error message encountered by the Server object. To see a complete list of messages, refer to the ErrorList property. When possible, the system translates messages into the locale of the client.

**Syntax**

object.**LastError***[ = String]*

**Parameters**

None

**LastError Property (ServerManager Object)**

Returns a text description of the last error encountered by the ServerManager object.

**Syntax**

object.**LastError**

**Parameters**

None

**Remarks**

`LastError` is a read-only property of type String.

**LastError Property (Tag Object)**

Contains the last error string generated.

**Syntax**

object.**LastError***[ = String]*

**Parameters**

None

**LastError Property (TagRecordset Object)**

Returns the last error message encountered by the TagRecordset object. To see a complete list of messages, refer to the ErrorList property. When possible, the system translates messages into the locale of the client.

**Syntax**

object.**LastError**

**Parameters**

None

**Remarks**

`LastError` is a read-only property of type String.

**LastError Property (Tags Object)**

Returns the last error message encountered by the Tags Object. To see a complete list of messages, refer to the ErrorList property. When possible, the system translates messages into the locale of the client.

**Syntax**

object.**LastError**

**Parameters**

None

**Remarks**

`LastError` is a read-only property of type String.

**LastModified Property (Tag Object)**

Returns the time the tag configuration of this Tag was last modified.

**Syntax**

object.**LastModified**

**Parameters**

None

**Remarks**

`LastModified` is a read-only property of type Date.

**LastModified Property (TagCriteria Object)**

Sets the last modified time to search for in the TagRecordset query.

**Syntax**

object.**LastModified***[ = Date]*

**Parameters**

None

**LastModified Property (TagFields Object)**

Determines whether the LastModified field should be returned in the TagRecordset query.

**Syntax**

object.**LastModified***[ = Boolean]*

**Parameters**

None

**LastModified Property (UserDefinedTypes Object)**

Returns the time when the User Defined Type was last modified.

**Syntax**

object.**LastModified**

**Parameters**

None

**Returns**

`LastModified` is a read-only property of type Date.

**LastModifiedSet Property (TagCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**LastModified***[ = Boolean]*

**Parameters**

None

**LastModifiedUser Property (Tag Object)**

Returns the user who last modified this Tag configuration.

**Syntax**

object.**LastModifiedUser**

**Parameters**

None

**Remarks**

`LastModifiedUser` is a read-only property of type Date.

**LastModifiedUser Property (TagCriteria Object)**

Sets the last modified user to search for in the TagRecordset query.

**Syntax**

object.**LastModifiedUser***[=string]*

**Parameters**

None

**LastModifiedUser Property (TagFields Object)**

Determines whether the LastModifiedUser field should be returned in the TagRecordset query.

**Syntax**

object.**LastModifiedUser***[=Boolean]*

**Parameters**

None

**LastModifiedUser Property (UserDefinedType Object)**

Returns the last user that modified the User Defined Type.

**Syntax**

object.**LastModifiedUser**

**Parameters**

None

**Returns**

`LastModifiedUser` is a read-only property of type String.

**LastModifiedUserSet Property (TagCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**LastModifiedUser***[=Boolean]*

**Parameters**

None

**LeafCount Property (OPCBrowse Object)**

Returns the number of leaves under a browse position. When a browse operation occurs, the server returns all areas and sources under the BrowsePosition. LeafCount gives you the number of leaves, which you can use to iterate the LeafNames and FullLeafNames arrays.

**Syntax**

object.**LeafCount**

**Parameters**

None

**Returns**

`LeafCount` is a read-only property of type String.

**LeafNames Property (OPCBrowse Object)**

Returns the display leaf name (source) for an AE server. See "FullLeafnames" property for more information.

**Syntax**

object.**LeafNames**(Index)

**Parameters**

Index - Integer - The index into the leaf name array.

**Returns**

`LeafNames` is a read-only String property returned as a variant for script compatibility.

**LicensedTags Property (Server Object)**

Returns the maximum number of tags that you can configure on the Server.

**Syntax**

object.**LicensedTags**(Index)

**Parameters**

None

**Remarks**

`LicensedTags` is a read-only property of type Long.

**LicensedUsers Property (Server Object)**

Returns the maximum number of users that may be connected to the Server at one time.

**Syntax**

object.**LicensedUsers**

**Parameters**

None

**Remarks**

`LicensedUser` is a read-only property of type Long.

**LoEngineeringUnits Property (Tag Object)**

Returns or sets the low engineering unit range of the Tag. Changes to tag properties are not committed until you call the WriteRecordset method of the TagRecordset object.

**Syntax**

object.**LoEngineeringUnits***[ = Double]*

**Parameters**

None

**LoEngineeringUnits Property (TagCriteria Object)**

Sets the low engineering unit range to search for in the TagRecordset query.

**Syntax**

object.**LoEngineeringUnits***[ = Double]*

**Parameters**

None

**LoEngineeringUnits Property (TagFields Object)**

Determines whether the LoEngineeringUnits field should be returned in the TagRecordset query.

**Syntax**

object.**LoEngineeringUnits***[ = Boolean]*

**Parameters**

None

**LoEngineeringUnitsSet Property (TagCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**LoEngineeringUnitsSet***[ = Boolean]*

**Parameters**

None

**LoEngineeringUnitsUpdated Property (Tag Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**LoEngineeringUnitsUpdated***[ = Boolean]*

**Parameters**

None

**LoScale Property (Tag Object)**

Returns or sets the low scale value used for input scaling on the Tag. Changes to tag properties are not committed until you call the WriteRecordset method of the TagRecordset object.

**Syntax**

object.**LoScale***[ = String]*

**Parameters**

None

**LoScale Property (TagCriteria Object)**

Sets the low input scale range value to search for in the TagRecordset query.

**Syntax**

object.**LoScale***[ = String]*

**Parameters**

None

**LoScale Property (TagFields Object)**

Determines whether the LoScale field should be returned in the TagRecordset query.

**Syntax**

object.**LoScale***[ = Boolean]*

**Parameters**

None

**LoScale Property (TagCriteria Object)**

Determines whether the LoScale field should be returned in the TagRecordset query.

**Syntax**

object.**LoScaleSet***[ = Boolean]*

**Parameters**

None

**LoScaleUpdated (Tag Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**LoScaleUpdated***[ = Boolean]*

**Parameters**

None

**LoadBalancing Property (Tag Object)**

Returns or sets the status of data collection load balancing for the Tag. Load balancing is used for polled type collection to evenly distribute data collection load over available sampling times. This is sometimes called "Phase Shifting". Changes to tag properties are not committed until you call the WriteRecordset method of the TagRecordset object.

**Syntax**

object.**LoadBalancing***[ = Boolean]*

**Parameters**

None

**LoadBalancing Property (TagCriteria Object)**

Sets the load balancing to search for in the TagRecordset query.

**Syntax**

object.**LoadBalancing***[ = Boolean]*

**Parameters**

None

**LoadBalancing Property (TagFields Object)**

Determines whether the LoadBalancing field should be returned in the TagRecordset query.

**Syntax**

object.**LoadBalancing***[ = Boolean]*

**Parameters**

None

**LoadBalancingSet Property (TagCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**LoadBalancingSet***[ = Boolean]*

**Parameters**

None

**LoadBalancingUpdated Property (Tag Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**LoadBalancingUpdated***[ = Boolean]*

**Parameters**

None

**LowPart Property (QueryModifiers Object)**

Returns or sets the LowPart of query modifier. Use the Server.CriteriaFromStrings method to determine the HighPart and LowPart of a query modifier.

**Syntax**

object.**LowPart**

**Parameters**

None

**Remarks**

Long

**LowSeverity Property (OPCFilters Object)**

Gets/Sets the Low Severity filter in the Alarm Collector. Only events with a severity higher than or equal to LowSeverity will be collected.

**Syntax**

object.**LowSeverity***[ = Long]*

**Parameters**

None

# Property Reference M-N

## M

### MaintainAutoRecoveryFiles Property (Server Object)

Returns or sets whether auto recovery files are maintained by the server. Auto recovery files are periodic backups of the current .IHA and .IHC and protect against data loss due to ungraceful server shutdown.

**Syntax**

object.**MaintainAutoRecoveryFiles***[ = Boolean]*

**Parameters**

None

### MajorVersion Property (Server Object)

Returns the server or client major version number (1, 2, 3, and so on).

**Syntax**

object.**MajorVersion***([API])*

**Parameters**

| Name | Data Type | Description |
|---|---|---|
| API | Boolean | When True, the server version is returned. When False, the client version is returned. The default is False. |

**Remarks**

`MajorVersion` is a read-only property of type Long.

### Master Property (TagRecordset Object)

Returns the Master tag upon which bulk changes to tag configuration may be performed. When calling the WriteRecordset method, and supplying the "UserMasterTag" parameter, all

changes made to the Master tag will be replicated to each tag in the TagRecordset whose Selected property is set to True.

**Syntax**

object.**Master***([ClearUpdates])*

**Parameters**

| Name | Data Type | Description |
|---|---|---|
| ClearUpdates | Boolean | Whether to clear all pending updates in the master tag (optional, default = False). |

**Remarks**

`Master` is a read-only property of type Tag.

Example

```
Dim Recordset As TagRecordset Dim Master As Tag ' Get a new TagRecordset Set Recordset =
 MyServer.Tags.NewRecordset '
Fill in criteria, get all tags with collector compression on With Recordset.Criteria
 .Tagname = "*"   .CollectorCompression = True End With Recordset.QueryRecordset
Set Master = Recordset.Master ' Set Compression Percent to 0.5% Master.CollectorDeadbandPercentRange =
 0.5 '
Select all tags in the recordset Recordset.SelectAll ' Commit Changes If Not
 Recordset.WriteRecordset(True)
Then   MsgBox "Failed To Commit Tag Changes" End If
```

## MaxReturnCells Property (DataRecordset Object)

Sets the Maximum number of cells to return in the DataRecordSet.

**Syntax**

object.**MaxReturnCells***([size])*

**Parameters**

| Name | Data Type | Description |
|---|---|---|
| Size | Long | Number of cells to return. |

### MaximumQueryIntervals Property (Server Object)

Returns or sets the maximum query intervals enforced by the server. This property restricts the maximum number of samples per tag that the server can return for non-raw data queries. Filtered and raw data queries are not throttled.

**Syntax**

object.**MaximumQueryIntervals***([Long])*

**Parameters**

None

### MaximumQueryTime Property (Server Object)

Returns or sets the maximum query time enforced by the server. The property restrict query times and provides balanced access to the server.

**Syntax**

object.**MaximumQueryTime***([Long])*

**Parameters**

None

### MessageNumber Property (Message Object)

Returns the message number of the Message. A message number is a unique identifier for each Message required for national language support.

**Syntax**

object.**MessageNumber***[Long]*

**Parameters**

None

**Remarks**

MessageNumber is a read-only property of type Long.

### MessageNumber Property (MessageCriteria Object)

Establishes which message number the MessageRecordset query will filter by. A message number is a unique identifier for each Message required for national language support.

**Syntax**

object.**MessageNumber***[Long]*

**Parameters**

None

**MessageNumber Property (MessageFields Object)**

Determines whether the MessageNumber field should be returned in the MessageRecordset query.

**Syntax**

object.**MessageNumber***[=Boolean]*

**Parameters**

None

**MessageNumberSet Property (MessageCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**MessageNumberSet***[=Boolean]*

**Parameters**

| Option | Description |
|---|---|
| MessageOnDataUpdate | Determines whether data updates generate a message. |

**MessageOptions Property (Messages Object)**

Exposes the MessageOptions collection to control the behavior of the Historian message generation and archiving.

If the property is set to True, all applications connected to the archiver will cause the archiver to generate a message on a data update, not just the application that set the

property. For instance, if you set the property with the SDK and the File collector updates data (a single value) a message will be generated by the archiver.

An example of a message that appears in Historian Administrator after you change a specified tag property through the SDK would be:

```
bsmith(MY_DOMAIN\bsmith) wrote 96 to tag aa at 01/10/2003 04:13:49.861 PM, Replaced 94.000
```

The message topic type would be Security.

The following table describes each of the messaging options:

**Syntax**

object.**MessageOptions**(OptionName)*[=Variant]*

**Parameters**

| Name | Data Type | Description |
|---|---|---|
| OptionName | String | The name of the message option. |

**Remarks**

MessageOptions is a read-only property of type Collection of options.

Example

```
ConnectedServer.Messages.MessageOptions("MessageOnDataUpdate") = True
```

### MessageString Property (Message Object)

Returns the translated text of the message, including any substitutions. Messages generally include translated fixed text and variable substitutions such as timestamps, usernames, and tagnames.

**Syntax**

object.**MessageString***[String]*

**Parameters**

None

### MessageString Property (MessageCriteria Object)

Establishes text search criteria the MessageRecordset query filters by. The MessageString may be any sub-string of the message. You should not include wildcard characters.

**Syntax**

object.**MessageString***[=String]*

**Parameters**

None

**MessageString Property (MessageFields Object)**

Determines whether the MessageString field should be returned in the MessageRecordset query.

**Syntax**

object.**MessageString***[=Boolean]*

**Parameters**

None

**MessageStringSet Property (MessageCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**MessageStringSet***[=Boolean]*

**Parameters**

None

**Messages Property (Server Object)**

Returns the Messages Object for the current Server. Use the Messages Object to build queries for message data.

**Syntax**

object.**Messages**

**Parameters**

None

### MessageOptions Property (Server Object)

Returns the server or client minor version number (0, 1, 2, and so on).

#### Syntax

object.**MessageVersion**([API])

#### Parameters

| Name | Data Type | Description |
|---|---|---|
| API | Boolean | When True, the server version is returned. When False, the client version is returned. The default is False. |

#### Remarks

MinorVersion is a read-only property of type Long.

## N

### Name Property (Archive Object)

Returns the name of the specified Archive.

#### Syntax

object.**Name**

#### Parameters

None

### Name Property (Collector Object)

Returns the name of the specified Collector.

#### Syntax

object.**Name**

#### Parameters

None

**Remarks**

Name is a read-only property of type String.

**Name Property (DataStore Object)**

Returns the name of data store.

**Syntax**

object.**Name**

**Parameters**

None

**Remarks**

String.

**Name Property (Option Object)**

The name parameter of this option.

**Syntax**

object.**Name***[ = String]*

**Parameters**

None

**Name Property (UserCalcFunction Object)**

Returns the name of the specified UserCalcFunction.

**Syntax**

object.**Name***[ = String]*

**Parameters**

None

**NodeFilter Property (OPCBrowse Object)**

Some OPCAE Servers allow you to set a filter for the browse. The Areas Returned by the Browse operation will be only those that match the filter criterion. If this property is not set, no filtering takes place.

**Syntax**

object.**NodeFilter***[ = Variant]*

**Parameters**

None

**Remarks**

Nodefilter is a read/write property of type Variant for script compatibility.

### NumberOfElements Property (Tag Object)

Returns or sets the number of elements of a tag to specify if it is an array. If the value of the NumberOfElements is -1 then it is an array tag. Changes to tag properties are not committed until you call the WriteRecordset method of the TagRecordset object.

**Syntax**

object.**NumberOfElements***[ = Long]*

**Parameters**

None

### NumberOfElements Property (TagCriteria Object)

Sets the NumberOfElements property to search for in the TagRecordset query. For example you can set to -1 if you want only array tags to be returned or set to 0 if you want non-array tags.

**Syntax**

object.**NumberOfElements***[ = Long]*

**Parameters**

None

### NumberOfElements Property (TagFields Object)

Determines whether the NumberOfElements field should be returned in the TagRecordset query.

**Syntax**

object.**NumberOfElements***[ = Boolean]*

**Parameters**

None

**NumberOfFields Property (UserDefinedType Object)**

Returns the number of fields in the User Defined Type.

**Syntax**

object.**NumberOfFields**

**Parameters**

Integer

**NumberOfSamples Property (DataCriteria Object)**

Returns or sets the number of samples to retrieve from the archive in the DataRecordset query. Samples will be evenly spaced within the time range defined by start time and end time for most sampling modes. For the "RawByNumber" sampling mode the NumberOfSamples determines the maximum number of values to retrieve. For the "RawByTime" sampling mode, the NumberOfSamples is ignored.

**Syntax**

object.**NumberOfFields***[ = Long]*

**Parameters**

None

**NumberOfSamplesSet Property (DataCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**NumberOfSamplesSet***[ = Long]*

**Parameters**

None

### NumStateNames Property (EnumeratedState Object)

Returns the number of states in the set.

**Syntax**

object.**NumStateNames**

**Parameters**

None

**Remarks**

Long

## Property Reference O-P

### P

### PercentGood Property (DataValue Object)

Returns the percentage of time the value was of good quality during the interval the DataValue applies to. Does not apply to **RawByNumber** and **RawByTime** sampling modes.

**Syntax**

object.**PercentGood**

**Parameters**

None

**Remarks**

PercentGood is a read-only property of type Single.

Example

```
If MyValue.PercentGood < 50 Then   err.Raise 1,, "We do not have enough good data" End If
```

### PerformanceDSStatistics (Archives Object)

Exposes performance statistics for the data store. These statistics are updated periodically.

The Product server and new current values are automatically reported to the client and updated in the PerformanceDSStatistics collection.

**Syntax**

object.**PerformanceDSStatistics**

**Parameters**

None

**Remarks**

PerformanceDSStatistics is a read-only property of type Collection of Variant.

**PerformanceDSStatistics (Archives Object)**

Exposes performance statistics for the data stores. These statistics are updated periodically.

The server and new current values are automatically reported to the client and updated in the PerformanceDSStatistics collection.

The following table describes each of the performance statistics and their purpose.

**DataStores Performance Statistics**

| Statistic | Description |
|---|---|
| ArchiveTotalEvents | Total number of data points received by the data store.<br><br>This option is specific only to data stores. |
| ArchiveTotalOutOf-Order | Total number of data points received by the server out of time order with the current value.<br><br>This option is specific only to data stores. |
| ArchiveAverageEventRate | Average events processed per minute.<br><br>This option is specific only to data stores. |
| ArchiveMinimum-EventRate | Minimum events processed per minute.<br><br>This option is specific only to data stores. |

| Statistic | Description |
|---|---|
| ArchiveMaximum-EventRate | Maximum events processed per minute.<br><br>This option is specific only to data stores. |
| ArchiveWriteCache-HitRatio | Hit Ratio To Write Cache.<br><br>This option is specific only to data stores. |
| ArchiveAverageCom-pressionRatio | Average compression ratio for all tags.<br><br>This option is specific only to data stores. |
| ArchiveMinimum-CompressionRatio | Minimum compression ratio for any tag.<br><br>This option is specific only to data stores. |
| ArchiveTotalFailed-Writes | Total number of refused write attempts by any connection.<br><br>This option is specific only to data stores. |
| ArchiveTotalMes-sages | Total number of non-alert messages generated.<br><br>This option is specific only to data stores. |
| ArchiveTotalAlerts | Total number of alert messages generated.<br><br>This option is specific only to data stores. |
| ArchiveFreeSpace | Amount of free space (MB) in the current archive.<br><br>This option is specific only to data stores. |
| ArchiveSpaceCon-sumptionRate | Current free space consumption rate (MB/Day).<br><br>This option is specific only to data stores. |
| ArchivePredicted-DaysToFull | Predicted days to archive full.<br><br>This option is specific only to data stores. |
| ArchiveSpaceEffi-ciency | Efficiency of archive space utilization.<br><br>This option is specific only to data stores. |
| ArchiveAverageEven-tRateArray | An array of archiver received rates used to make the trend.<br><br>This option is specific only to data stores. |

| Statistic | Description |
|---|---|
| ArchiveAverage-AlarmRate | Displays the rate at which Historian is receiving alarms and events data.<br><br>This option is specific only to data stores. |
| ArchiveTotalAlarm | The rate at which Historian is receiving data.<br><br>This option is specific only to data stores. |

**Syntax**

object.**PerformanceDSStatistics**

**Parameters**

None

**Remarks**

PerformanceDSStatistics is a read-only property of type Collection of Variant.

**PerformanceStatistics (Archives Object)**

Exposes performance statistics for the Product Archiver. These statistics are updated periodically.

The Product server and new current values are automatically reported to the client and updated in the PerformanceStatistics collection.

**Syntax**

object.**PerformanceStatistics**

**Parameters**

None

**Remarks**

PerformanceStatistics is a read-only property of type Collection of Variant.

**PerformanceStatistics Property (Archives Object)**

Exposes performance statistics for the Historian Archiver. These statistics are updated periodically. The Historian server and new current values are automatically reported to the client and updated in the PerformanceStatistics collection.

The following table describes each of the performance statistics and their purpose.

**Archives Performance Statistics**

| Statistic | Description |
|---|---|
| ArchiveTotalOutOf-Order | Total number of data points received by the server out of time order with the current value.<br><br>This option is specific only to data stores. |
| ArchiveAverageEventRate | Average events processed per minute.<br><br>This option is specific only to data stores. |
| ArchiveMinimum-EventRate | Minimum events processed per minute.<br><br>This option is specific only to data stores. |
| ArchiveMaximum-EventRate | Maximum events processed per minute.<br><br>This option is specific only to data stores. |
| ArchiveWriteCache-HitRatio | Hit Ratio To Write Cache.<br><br>This option is specific only to data stores. |
| ArchiveAverageCompressionRatio | Average compression ratio for all tags.<br><br>This option is specific only to data stores. |
| ArchiveMinimum-CompressionRatio | Minimum compression ratio for any tag.<br><br>This option is specific only to data stores. |
| ArchiveTotalFailed-Writes | Total number of refused write attempts by any connection.<br><br>This option is specific only to data stores. |
| ArchiveTotalMessages | Total number of non-alert messages generated.<br><br>This option is specific only to data stores. |
| ArchiveTotalAlerts | Total number of alert messages generated. |

| Statistic | Description |
|---|---|
| | This option is specific only to data stores. |
| ArchiveFreeSpace | Amount of free space (MB) in the current archive.<br><br>This option is specific only to data stores. |
| ArchiveSpaceConsumptionRate | Current free space consumption rate (MB/Day).<br><br>This option is specific only to data stores. |
| ArchivePredictedDaysToFull | Predicted days to archive full.<br><br>This option is specific only to data stores. |
| ArchiveSpaceEfficiency | Efficiency of archive space utilization.<br><br>This option is specific only to data stores. |
| ArchiveAverageEventRateArray | An array of archiver received rates used to make the trend.<br><br>This option is specific only to data stores. |
| ArchiveAverageAlarmRate | Displays the rate at which Historian is receiving alarms and events data.<br><br>This option is specific only to data stores. |
| ArchiveTotalAlarm | The rate at which Historian is receiving data.<br><br>This option is specific only to data stores. |

**Syntax**

object.**PerformanceStatistics**[AsLocalizedString])

**Parameters**

| Name | Data Type | Description |
|---|---|---|
| StatisticName | String | Name of the statistic to retrieve (read-only). |
| AsLocalizedString | Boolean | Whether to return the statistic value(s) as a localized string (optional, default = True, read-only). |

**Remarks**

PerformanceStatistics is a read-only property of type Collection of Variant.

Example

```
Dim TheStatistic As Variant

' Get The Overall Compression Performance Statistic

TheStatistic = MyArchives.PerformanceStatistics("ArchiveAverageCompressionRatio")
```

**PerformanceStatistics Property (Collector Object)**

Exposes performance statistics for the selected collector.

The following table describes each of the performance statistics and their purpose.

**Collector Performance Statistics**

Statistic Description

| CollectorTotal-EventsCollected | Total Values Collected |
|---|---|
| CollectorTotal-EventsReported | Total Values Reported To Server. |
| CollectorOutOfOrder-Events | Total Events Out Of Timestamp Order. |
| CollectorAverage-EventRate | Average Events Reported / Minute. |
| CollectorMinimum-EventRate | Minimum Events Reported / Minute. |
| CollectorMaximum-EventRate | Maximum Events Reported / Minute. |
| CollectorReportRatio | Ratio Of Events Collected To Events Reported. |
| CollectorCompres-sionPercent | The percent of data compressed by the collector. |
| CollectorMisses | The data that are missed by the collector due to missing scheduled collection times. Also called Overruns. |

**Syntax**

object.**PerformanceStatistics**(StatisticName,[AsLocalizedString])

**Parameters**

| Name | Data Type | Description |
|---|---|---|
| StatisticName | Variant | Name of the statistic to retrieve (read-only). |
| AsLocalizedString | Boolean | Whether to return the statistic value(s) as a localized string (optional, default = True, read-only). |

Example

```
Dim TheStatistic As Variant

' Get The Overall Compression Performance Statistic

TheStatistic = MyCollector.PerformanceStatistics("CollectorEventRate")
```

**PrimaryUsername Property (DataComments Object)**

Returns the user who added the comment to the archive.

**Syntax**

object.**PrimaryUsername**

**Parameters**

None

**Remarks**

PrimaryUsername is a read-only property of type String.

**Example**

```
Debug.Print "Comment added by: " + MyValue.Comments(1).PrimaryUsername
```

**PrincipalCollector Property (Collector Object)**

The name of the collector this collector is backing up. If the principal collector is not responding this collector is next in line to become active.

**Syntax**

object.**PrincipalCollector***[ = String]*

**Parameters**

None

**PropertyList Property (Archives Object)**

This function returns a list of the supported Archive properties.

**Syntax**

object.**PropertyList**

**Parameters**

None

**PropertyList Property (Collectors Object)**

Returns a list of the available collector properties. This is a complete list across all collector types. A given collector may ignore certain properties if they do not apply.

**Syntax**

object.**PropertyList**

**Parameters**

None

**PropertyList Property (Data Object)**

This function returns a list of the available Properties of a Data value.

**Syntax**

object.**PropertyList**([ExcludeRaw], [ExcludeAlarms])

**Parameters**

| Name | Data Type | Description |
|---|---|---|
| ExcludeRaw | Boolean | Whether to exclude the PercentGood property (op tional, default = False). |
| ExcludeAlarms | Boolean | Whether to exclude the Alarm Message and Alarm ID properties (optional, default = False). |

**PropertyList Property (Messages Object)**

Returns a list of available Properties for a message.

**Syntax**

object.**PropertyList**

**Parameters**

None

**PropertyList Property (Tags Object)**

Returns a list of valid Properties for a Tag.

**Syntax**

object.**PropertyList**

**Parameters**

None

**QueryModifiers Property (QueryModifiers Object)**

Returns or sets the QueryModifiers to be used during the DataRecordset query.

**Syntax**

object.**QueryModifiers**

```
Set resQueryModifiers ConnectedServer.CriteriaFromStrings(txtQueryModifier.Text)

 .Criteria.QueryModifiers = resQueryModifiers.QueryModifiers
```

# Property Reference Q-R

**R**

**RawValue Property (Option Object)**

The raw numeric value of this option.

**Syntax**

object.**RawValue***[ = Double]*

**Parameters**

None

**ReadOnly Property (Archive Object)**

Returns or sets the ReadOnly status of the specified archive. If you attempt to set the current archive to ReadOnly (the archive receiving newest data), an error is generated.

**Syntax**

object.**ReadOnly***[ = Boolean]*

**Parameters**

None

**Example** `' Set Archive To ReadOnly MyArchive.ReadOnly = True ' Reset Archive ReadOnly Status MyArchive.ReadOnly = False`

**ReadSecurityGroup Property (Tag Object)**

RReturns or sets the name of the security group controlling the reading of data for the Tag. Changes to tag properties are not committed until you call the WriteRecordset method of the TagRecordset object.

**Syntax**

object.**ReadSecurityGroup***[ = String]*

**Parameters**

None

**ReadSecurityGroup Property (TagCriteria Object)**

Returns or sets the ReadOnly status of the specified archive. If you attempt to set the current archive to ReadOnly (the archive receiving newest data), an error is generated.

**Syntax**

object.**ReadSecurityGroup***[ = String]*

**Parameters**

None

**Example** `With MyRecordset.Criteria .TagName = "*.F_CV" .ReadSecurityGroup = "Power Users" End With`

**ReadSecurityGroup Property (TagFields Object)**

Determines whether the ReadSecurityGroup field should be returned in the TagRecordset query.

**Syntax**

object.**ReadSecurityGroup***[ = Boolean]*

**Parameters**

None

**Example**

```
With MyRecordset.Fields .Clear .TagName = True .Description = True .ReadSecurityGroup = True .WriteSecurityGroup = True .AdministratorSecurityGroup = True End With
```

### ReadSecurityGroupSet Property (TagCriteria Object)

A flag to indicate whether this property has been set or not.

**Syntax**

object.**ReadSecurityGroup***[ = Boolean]*

**Parameters**

None

### ReadSecurityGroupUpdated Property (Tag Object)

A flag to indicate whether this property has been set or not.

**Syntax**

object.**ReadSecurityGroupUpdated***[ = Boolean]*

**Parameters**

None

### RedundancyEnabled Property (Collector Object)

Returns or sets whether the collector is part of a redundant configuration.

**Syntax**

object.**RedundancyEnabled***[ = Boolean]*

**Parameters**

None

#### Running Property (Collector Object)

Returns or sets the running state of the specified Collector. Use this property to pause and resume collection without restarting and stopping the collector.

**Syntax**

object.**Running***[ = Boolean]*

**Parameters**

None

Example

```
' Resume the collector MyCollector.Running = True

         ' Pause the collector MyCollector.Running = False
```

## Property Reference S-T

### S

#### SamplingDirectionList Property (Data Object)

This function returns a list of the available Sampling directions for a data request.

**Syntax**

object.**SamplingDirectionList**

**Parameters**

None

#### SamplingInterval Property (DataCriteria Object)

Returns or sets the interval (in milliseconds) between samples from the archive in the DataRecordset query. For the "RawByNumber" and "RawByTime" sampling modes, the SamplingInterval is ignored. This property can be used in place of the NumberofSamples property.

**Syntax**

object.**SamplingInterval***[=Long]*

**Parameters**

None

**SamplingIntervalSet Property (DataCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**SamplingIntervalSet***[=Boolean]*

**Parameters**

None

**SamplingMode Property (DataCriteria Object)**

Returns or sets the mode of sampling data from the archive by the DataRecordset query.

The following table details the available sampling modes:

| Name | Description | Value |
|---|---|---|
| CurrentValue | Retrieves the current value, the time frame criteria are ignored. | 1 |
| Interpolated | Retrieves evenly spaced interpolated values based on NumberOfSamples and the time frame criteria. | 2 |
| Trend | Retrieves the raw minimum and raw maximum value for each specified interval. Use the Trend sampling mode to maximize performance when retrieving data points for plotting. A start time, end time, and an interval or number of samples must be specified. | 3 |
| RawByTime | Retrieves raw archive values (compressed) based on time frame criteria. | 4 |
| RawByNumber | Retrieves raw archive values (compressed) based on the StartTime criteria, the NumberOfSamples, and Direction criteria. Note the EndTime criteria is ignored for this Sampling mode. | 5 |

| Name | Description | Value |
|---|---|---|
| Calculated | Retrieves evenly spaced calculated values based on NumberOfSamples, the time frame criteria, and the CalculationMode criteria. | 6 |
| Lab | The Lab sampling mode only returns the collected values, without any interpolation of the value. The collected value is repeated for each interval until there is a change in the raw data sample's value.<br><br>Lab sampling is most often used to create a step chart rather than a smooth curve.<br><br>Use Lab sampling instead of interpolated if you only want true collected values returned. The Lab sampling mode is generally not useful on highly compressed data. Use interpolated sampling instead. | 7 |
| InterpolatedtoRaw | When you request interpolated data, you specify an interval or number of samples. If the actual stored number of raw samples is greater than required, you will get interpolated data as described above. If the actual number of stored samples are less than the required, then you will get the raw samples. In this way, the needs of trending detail and application load are balanced.<br><br>This mode is best used when querying compressed data because the Data Archiver can switch to the more efficient raw data query. | 8 |
| TrendtoRaw | TrendtoRaw retrieves raw data between a given intervals when the actual data | 9 |

| Name | Description | Value |
|---|---|---|
| | samples are fewer than the requested number of samples. | |
| LabtoRaw | LabtoRaw is an extension to Lab mode of sampling and similar to Interpolated-toRaw mode where you will be switched to raw data or lab when the actual data samples are fewer than the requested samples. | 10 |
| RawByFilterToggle | RawByFilterToggle returns filtered time ranges. The values returned are 0 and 1. If the value is 1, then the condition is true and 0 means false.<br><br>This sampling mode is used with the time range and filter tag condition. The result starts with a starting time stamp and ends with an ending timestamp. | 11 |

**Syntax**

object.**SamplingMode***[=iHistorian_SDK.ihSamplingMode]*

**Parameters**

None

**SamplingModeList Property (Data Object)**

This function returns a list of the available Sampling Modes for Data.

**Syntax**

object.**SamplingModeList***([ExcludeRaw])*

**Parameters**

| Name | Data Type | Description |
|---|---|---|
| ExcludeRaw | Boolean | Whether to exclude raw sampling modes (optional, default = False). |

### SamplingModeSet Property (DataCriteria Object)

A flag to indicate whether this property has been set or not.

**Syntax**

object.**SamplingModeSet***[ = Boolean]*

**Parameters**

None

### SecondaryUsername Property (DataComments Object)

Returns the user who signed the comment when it was added to the archive.

**Syntax**

object.**SecondaryUsername**

**Parameters**

None

**Remarks**

SecondaryUserName is a read-only property of type String.

**Example**

```
Debug.Print "Comment signed by:"+MyValue.Comments(1).SecondaryUsername
```

### SecurityGroupList Property (Server Object)

Returns the list of security groups from the current server. Security groups determine the system functions permitted by users within the group.

**Syntax**

object.**SecurityGroupList**

**Parameters**

None

**Remarks**

SecurityGroupList is a read-only property of type Variant.

### SecurityGroupLocation Property (Server Object)

Returns or sets the location of the security groups used by the server. The Historian server can either use security groups assigned to the local server computer ("Local" location) or groups configured for the domain ("Domain" location). Note, changing the SecurityGroupLocation requires a server restart to take effect.

#### Syntax

object.**SecurityGroupLocation***[ = String]*

#### Parameters

None

### Selected Property (Tag Object)

Returns whether or not this tag is selected.

#### Syntax

object.**Selected***[ = Boolean]*

#### Parameters

None

### Server Property (Archive Object)

Returns the server associated with the Archive object.

#### Syntax

object.**Server**

#### Parameters

None

#### Remarks

Server is a read-only property of type Server.

#### Example

```
Dim ServerName As String ' Get The Name Of The Server ServerName = MyArchive.Server.ServerName
```

### Server Property (Archives Object)

Returns the Server associated with the Archives object.

#### Syntax

object.**Server**

#### Parameters

None

#### Remarks

Server is a read-only property of type Server.

#### Example

```
Dim ServerName As String ' Get The Name Of The Server ServerName = MyArchives.Server.ServerName
```

### Server Property (Collector Object)

Returns the Server associated with the Collector Object.

#### Syntax

object.**Server**

#### Parameters

None

#### Remarks

Server is a read-only property of type Server.

#### Example

```
Dim ServerName As String ' Get The Name Of The Server ServerName = MyCollector.Server.ServerName
```

### Server Property (Collectors Object)

Returns the Server associated with the Collectors object.

#### Syntax

object.**Server**

**Parameters**

None

**Remarks**

Server is a read-only property of type Server.

**Example**

```
Dim ServerName As String ' Get The Name Of The Server ServerName = MyCollectors.Server.ServerName
```

### Server Property (Data Object)

Returns the server associated with the Data object.

**Syntax**

object.**Server**

**Parameters**

None

**Remarks**

Server is a read-only property of type Server.

**Example**

```
Dim ServerName As String ' Get The Name Of The Server ServerName = MyDataServer.Server.ServerName
```

### Server Property (DataRecordset Object)

Returns the Server associated with the DataRecordset object.

**Syntax**

object.**Server**

**Parameters**

None

**Remarks**

Server is a read-only property of type Server.

**Example**

```
Dim ServerName As String ' Get The Name Of The Server ServerName = MyRecordset.Server.ServerName
```

### Server Property (DataStores Object)

Returns the Server associated with the DataRecordset object.

**Syntax**

object.**Server() As Server**

**Parameters**

None

**Remarks**

Server is a read-only property of type Server.

### Server Property (MessageRecordset Object)

Returns the Server associated with the DataRecordset object.

**Syntax**

object.**Server**

**Parameters**

None

**Remarks**

Server is a read-only property of type Server.

**Example**

```
Dim ServerName As String

              ' Get the name of the server

              ServerName = MyRecordset.Server.ServerName
```

### Server Property (Messages Object)

Returns the Server associated with the Messages object.

**Syntax**

object.**Server**

**Parameters**

None

**Remarks**

Server is a read-only property of type Server.

**Example**

```
Dim ServerName As String

              ' Get The Name Of The Server

              ServerName = MyMessages.Server.ServerName
```

### Server Property (TagRecordset Object)

Returns the Server associated with the TagRecordset object.

**Syntax**

object.**Server**

**Parameters**

None

**Remarks**

Server is a read-only property of type Server.

**Example**

```
Dim ServerName As String

              ' Get the name of the server

              ServerName = MyRecordset.Server.ServerName
```

### Server Property (Tags Object)

Returns the Server associated with the Tags Object.

**Syntax**

object.**Server**

**Parameters**

None

**Remarks**

Server is a read-only property of type Server.

**Example**

```
Dim ServerName As String

              ' Get the name of the server

              ServerName = MyTags.Server.ServerName
```

**ServerName Property (Server Object)**

Returns the name of the computer that the server is running on.

**Syntax**

object.**ServerName***[ = String]*

**Parameters**

None

**Remarks**

Server is a read-only property of type String.

**ServerName Property (Server Object)**

Returns the name of the computer that the server is running on.

**Syntax**

object.**ServerName***[ = String]*

**Parameters**

None

**Remarks**

Server is a read-only property of type String.

**ServerTime Property (Server Object)**

Returns the current time on the connected server. The system then converts the time from UTC into formatted time based on the ConnectionOptions (TimeOption) setting.

**Syntax**

object.**ServerTime**

**Parameters**

None

**Servers Property (ServerManager Object)**

Maintains a list of registered servers on the client. To connect a listed server, use the Connect Method of a selected Server Object.

**Syntax**

object.**Servers**

**Parameters**

None

**SetName Property (EnumeratedSets Object)**

Returns or defines the name of an Enumerated Set.

**Syntax**

object.**SetName**

**Parameters**

None

**Returns**

String

**SimpleEvents Property (OPCFilters Object)**

Returns the list of Simple Event Categories being filtered on the Alarm Collector. This list is only applied if isSimpleEventsOn(true) has been called.

**Syntax**

object.**SimpleEvents**

**Parameters**

None

**SourceAddress Property (Tag Object)**

Returns or sets the address used to identify the Tag in the data source. Changes to tag properties are not committed until you call the WriteRecordset method of the TagRecordset object.

**Syntax**

object.**SourceAddress***[ = String]*

**Parameters**

None

**SourceAddress Property (TagCriteria Object)**

Sets the tag source address to search for in the TagRecordset query.

**Syntax**

object.**SourceAddress***[ = String]*

**Parameters**

None

**SourceAddress Property (TagFields Object)**

Determines whether the SourceAddress field should be returned in the TagRecordset query.

**Syntax**

object.**SourceAddress***[ = Boolean]*

**Parameters**

None

**SourceAddressSet Property (TagCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**SourceAddressSet***[ = Boolean]*

**Parameters**

None

**SourceAddressUpdated Property (Tag Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**SourceAddressUpdated***[ = Boolean]*

**Parameters**

None

**Sources Property (OPCFilters Object)**

Returns a list of the Sources selected for collection. This list is only applied when isSourceFiltering (true) has been called.

**Syntax**

object.**Sources***[ = Boolean]*

**Parameters**

None

**SpikeLogic Property (Tag Object)**

Returns or sets whether this tag uses the spike logic compression algorithm.

**Syntax**

object.**SpikeLogic***[ = Boolean]*

**Parameters**

None

**SpikeLogic Property (TagCriteria Object)**

Sets the spike logic to search for in the TagRecordset query.

**Syntax**

object.**SpikeLogic***[ = Boolean]*

**Parameters**

None

**SpikeLogic Property (TagFields Object)**

Determines whether the SpikeLogic field should be returned in the TagRecordset query.

**Syntax**

object.**SpikeLogic***[ = Boolean]*

**Parameters**

None

**SpikeLogicOverride Property (Tag Object)**

Returns or sets whether this tag should use the its own Spike Logic setting or the default Collectors setting.

**Syntax**

object.**SpikeLogicOverride***[ = Boolean]*

**Parameters**

None

**SpikeLogicOverride Property (TagCriteria Object)**

Sets the spike logic override to search for in the TagRecordset query.

**Syntax**

object.**SpikeLogicOverride***[ = Boolean]*

**Parameters**

None

**SpikeLogicOverride Property (TagFields Object)**

Determines whether the SpikeLogicOverride field should be returned in the TagRecordset query.

**Syntax**

object.**SpikeLogicOverride***[ = Boolean]*

**Parameters**

None

### SpikeLogicOverrideSet Property (TagCriteria Object)

A flag to indicate whether this property has been set or not.

**Syntax**

object.**SpikeLogicOverrideSet***[ = Boolean]*

**Parameters**

None

### SpikeLogicOverrideUpdated Property (Tag Object)

A flag to indicate whether this property has been set or not.

**Syntax**

object.**SpikeLogicOverrideUpdated***[ = Boolean]*

**Parameters**

None

### SpikeLogicSet Property (TagCriteria Object)

A flag to indicate whether this property has been set or not.

**Syntax**

object.**SpikeLogicSet***[ = Boolean]*

**Parameters**

None

### SpikeLogicUpdated Property (Tag Object)

A flag to indicate whether this property has been set or not.

**Syntax**

object.**SpikeLogicUpdated***[ = Boolean]*

**Parameters**

None

### StartTime Property (Archive Object)

Returns the start time of the specified archive. This represents the earliest timestamp for any sample contained in the archive.

**Syntax**

object.**StartTime**

**Parameters**

None

### StartTime Property (DataCriteria Object)

Returns or sets the start of the time range to retrieve data for in the DataRecordset query.

**Syntax**

object.**StartTime***[ = Date]*

**Parameters**

None

### StartTime Property (MessageCriteria Object)

Establishes the start time of the MessageRecordset query.

**Syntax**

object.**StartTime***[ = Date]*

**Parameters**

None

### StartTimeSet Property (DataCriteria Object)

A flag to indicate whether this property has been set or not.

**Syntax**

object.**StartTimeSet***[ = Boolean]*

**Parameters**

None

### StartTimeSet Property (MessageCriteria Object)

A flag to indicate whether this property has been set or not.

**Syntax**

object.**StartTimeSet***[ = Boolean]*

**Parameters**

None

### StartTimeShortcut Property (DataCriteria Object)

Returns or sets the start of the time range to retrieve data for in the DataRecordset query. Shortcuts are strings that define absolute or relative times.

The time shortcuts are the following:

| Value | Description |
|---|---|
| (N)ow | The current time (absolute). |
| (T)oday | Today at midnight (absolute). |
| (Y)esterday | Yesterday at midnight (absolute). |
| (D)ays | Number of Days (relative). |
| (M)in | Number of Minutes (relative). |

| Value | Description |
|---|---|
| (H)our | Number of Hours (relative). |
| (W)eek | Number of Weeks (relative). |
| (BOM) | Beginning of this month at Midnight (absolute). |
| (EOM) | Last Day of this month at Midnight (absolute). |
| (BOY) | First Day of this year at Midnight (absolute). |
| (EOY) | Last Day of this year at Midnight (absolute). |

**Syntax**

object.**StartTimeShortcut***[ = String]*

**Parameters**

None

**Example**

```
 With MyRecordset.Criteria   .Tagmask = "*.F_CV"   ' Start two days Ago at 8am in the morning
.StartTimeShortcut = "Today-2d+8h"   ' End this morning at 8am
.EndTimeShortcut = "Today+8h" End With
```

### StartTimeShortcut Property (MessageCriteria Object)

Establishes the start time of the MessageRecordset Query by specifying a time shortcut string versus a date.

The time shortcuts are the following:

| Value | Description |
|---|---|
| (N)ow | The current time (absolute). |
| (T)oday | Today at midnight (absolute). |
| (Y)esterday | Yesterday at midnight (absolute). |
| (D)ays | Number of Days (relative). |
| (M)in | Number of Minutes (relative). |

| Value | Description |
|---|---|
| (H)our | Number of Hours (relative). |
| (W)eek | Number of Weeks (relative). |
| (BOM) | Beginning of this month at Midnight (absolute). |
| (EOM) | Last Day of this month at Midnight (absolute). |
| (BOY) | First Day of this year at Midnight (absolute). |
| (EOY) | Last Day of this year at Midnight (absolute). |

**Syntax**

object.**StartTimeShortcut***[ = String]*

**Parameters**

None

**Example**

```
'Set A Start Time For Two Days Ago

MyMessages.Criteria.StartTimeShortcut = "-2d"
```

### StartTimeShortcutSet Property (MessageCriteria Object)

A flag to indicate whether this property has been set or not.

**Syntax**

object.**StartTimeShortcutSet***[ = Boolean]*

**Parameters**

None

### State Property (DataStore Object)

Returns or sets the state of data store. For example, Running.

**Syntax**

object.**State**

**Parameters**

None

Remarks

`ihDataStoreState`

**State Property (DataStore Object)**

Returns or sets the state of data store. For example, Running.

**Syntax**

object.**State**

**Parameters**

None

Remarks

`ihDataStoreState`

**State Property (DataStore Object)**

Returns or sets the state of data store. For example, Running.

**Syntax**

object.**State**

**Parameters**

None

Remarks

`ihDataStoreState`

**StateLowRawValue Property (EnumeratedState Object)**

Returns or defines the minimum value of a state when you are using a range of values for a state. For example, if your state value is from 0 to 5 and it is defined as ON, then this would be 0.

**Syntax**

object.**StateLowRawValue**

**Parameters**

None

Remarks

`Variant. Returns the low raw value of the state.`

**StateValue Property (QueryModifiers Object)**

Returns or sets the state value to be used when querying data with the StateCount and StateTime calculation modes.

**Syntax**

object.**StateValue = value**

**Status Property (Archive Object)**

Returns the status of the specified Archive. This status is typically Current or Active.

**Syntax**

object.**Status**

**Parameters**

None

Remarks

Status is a read-only property of type String.

Example

```
' Print Out The Archive Status Debug.Print MyArchive.Status
```

**Status Property (Collector Object)**

Returns the status of the specified Collector. This status is typically Running or Stopped.

**Syntax**

object.**Status**

**Parameters**

None

Remarks

Status is a read-only property of type String.

Example

```
' Print Out The Collector Status Debug.Print MyCollector.Status
```

**StepValue Property (Tag Object)**

Returns or sets whether this tag is using Step Value or not.

**Syntax**

object.**StepValue***[ = Boolean]*

**Parameters**

None

**StepValue Property (TagCriteria Object)**

Sets the Step Value value to search for in the TagRecordset query.

**Syntax**

object.**StepValue***[ = Boolean]*

**Parameters**

None

**StepValue Property (TagFields Object)**

Determines whether the StepValue field should be returned in the TagRecordset query.

**Syntax**

object.**StepValue***[ = Boolean]*

**Parameters**

None

**StepValueSet Property (TagCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**StepValueSet***[ = Boolean]*

**Parameters**

None

**StepValueUpdated Property (Tag Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**StepValueUpdated***[ = Boolean]*

**Parameters**

None

**StorageType Property (DataStore Object)**

Returns the type of data store. For example, ScadaBufferStore or Historical data store.

**Syntax**

object.**StorageType**

**Parameters**

None

**Remarks**

ihDataStorageType

**StoreMilliseconds Property (Tag Object)**

Returns or sets the time resolution of the Tag. Changes to tag properties are not committed until you call the WriteRecordset method of the TagRecordset object.

**Syntax**

object.**StoreMilliseconds***[ = Boolean]*

**Parameters**

None

**StoreMilliseconds Property (TagCriteria Object)**

Sets the time resolution to search for in the TagRecordset query.

**Syntax**

object.**StoreMilliseconds***[ = Boolean]*

**Parameters**

None

**StoreMilliseconds Property (TagFields Object)**

Determines whether the StoreMilliseconds field should be returned in the TagRecordset query.

**Syntax**

object.**StoreMilliseconds***[ = Boolean]*

**Parameters**

None

**StoreMillisecondsSet Property (TagCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**StoreMilliseconds***[ = Boolean]*

**Parameters**

None

**StoreMillisecondsUpdated Property (Tag Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**StoreMillisecondsUpdated***[ = Boolean]*

**Parameters**

None

**StoreOPCQuality Property (Server Object)**

Returns or sets whether the server stores the raw (16-bit) OPC quality flags.

**Syntax**

object.**StoreOPCQuality***[ = Boolean]*

**Parameters**

None

**Strict Client Authentication Property**

Returns or sets whether to enforce strict client authentication. Setting this value to TRUE will disallow connections from clients older than version 4.7.

**Syntax**

object.**StrictClientAuthentication**

**Parameters**

None

**Strict Collector Authentication (Archives Object)**

Returns or sets whether to enforce strict client authentication. Setting this value to TRUE will disallow connections from clients older than version 4.7.

**Syntax**

object.**StrictCollectorAuthentication**

**Parameters**

None

**StringLength Property (Tag Object)**

Returns or sets the fixed string length associated with a fixed string type Tag. Changes to tag properties are not committed until you call the WriteRecordset method of the TagRecordset object.

**Syntax**

object.**StringLength***[ = Byte]*

**Parameters**

None

**StringLength Property (TagFields Object)**

Determines whether the StringLength field should be returned in the TagRecordset query.

**Syntax**

object.**StringLength***[ = Boolean]*

**Parameters**

None

**StringLengthSet Property (TagCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**StringLengthSet***[ = Boolean]*

**Parameters**

None

**StringLengthUpdated Property (Tag Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**StringLengthUpdated***[ = Boolean]*

**Parameters**

None

**SubscribeStatus Property (Archives Object)**

Subscribes to changes in Archive status and configuration. The Historian server publishes messages to any client signed up for Archive status changes as modifications are committed to the Archive Database. Setting the SubscribeStatus to True signs up for status and configuration changes on all archives.

Status and configuration changes are reported asynchronously to the client through the Status_Received event of the Archives object.

**Syntax**

object.**SubscribeStatus***[ = Boolean]*

**Parameters**

None

**Example**

```
' Subscribe To Archive Status Updates
          MyArchives.SubscribeStatus = True
          ' Unsubscribe To Archive Status Updates
          MyArchives.SubscribeStatus = False
```

**SubscribeStatus Property (Collectors Object)**

Subscribes to changes in Collector status and configuration. The Historian server publishes messages to any client signed up for Collector status changes as modifications are committed to the Collector database. Setting the SubscribeStatus to True will sign up for status and configuration changes on all Collectors.

Status and configuration changes are reported asynchronously to the client through the Status_Received event of the Collectors object.

**Syntax**

object.**SubscribeStatus***[ = Boolean]*

**Parameters**

None

**Example**

```
' Subscribe To Collector Status Updates
              MyCollectors.SubscribeStatus = True
              ' Unsubscribe To Collector Status Updates
              MyCollectors.SubscribeStatus = False
```

**Substitutions Property (Message Object)**

Returns the variable text of a message such as timestamps, usernames, and tagnames.

**Syntax**

object.**Substitutions**

**Parameters**

None

**Remarks**

Substitutions is a read-only property of type String.

**Example**

```
 Dim MySubstitutions As Collection Dim I As Integer Set MySubstitutions = MyMessage.Substitutions
For I = 1 To MySubstitutions.count   Debug.Print MySubstitutions(I) Next I
```

### Substitutions Property (MessageFields Object)

Determines whether the Substitutions should be returned in the MessageRecordset query.

**Syntax**

object.**Substitutions***[ = Boolean]*

**Parameters**

None

**Example**

```
 Dim MyMessages As iHistorian_SDK.MessageRecordset Set MyMessages = GetServer.Messages.NewRecordset
 With
MyMessages.Fields   .Topic = True   .TimeStamp = True   .MessageString = True   .Substitutions = True
 End With
```

## T

### Tagmask Property (DataCriteria Object)

Returns or sets a mask of tags to retrieve data for in the DataRecordset query. The tag mask may include wildcard characters including the "*" and "?".

**Syntax**

object.**Tagmask***[ = String]*

**Parameters**

None

**T**

**TagmaskSet Property (DataCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**TagmaskSet***[ = Boolean]*

**Parameters**

None

**Tagname Property (Tag Object)**

Returns the Tagname property of the Tag. Tagname is a read-only property.

**Syntax**

object.**Tagname***[ = String]*

**Parameters**

None

**Remarks**

Tagname is a read-only property of type String.

**Tagname Property (Tag Object)**

Returns the Tagname property of the Tag. Tagname is a read-only property.

**Syntax**

object.**Tagname***[ = String]*

**Parameters**

None

**Remarks**

Tagname is a read-only property of type String.

**Tagname Property (TagCriteria Object)**

Sets the Tagname or tag mask to search for in the TagRecordset query. The Tagname may include wildcard characters.

**Syntax**

object.**Tagname***[ = String]*

**Parameters**

None

**Tagname Property (TagFields Object)**

Determines whether the Tagname field should be returned in the TagRecordset query.

**Syntax**

object.**Tagname***[ = Boolean]*

**Parameters**

None

**TagnameSet Property (TagCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**TagnameSet***[ = Boolean]*

**Parameters**

None

**TagnameUpdated Property (Tag Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**TagnameUpdated***[ = Boolean]*

**Parameters**

None

**Tags Property (DataCriteria Object)**

Returns or sets an array of Tagnames to retrieve data for in the DataRecordset query.

**Syntax**

object.**Tags***[ = Variant]*

**Parameters**

None

**Tags Property (DataRecordset Object)**

Exposes the list of Tagnames associated with the collection of DataValue results maintained in the DataRecordset object. The Item property takes a tagname or tag index as a parameter to expose the collection of DataValues for that tag. The first collection of DataValues in the Item property.

**Syntax**

object.**Tags**

**Parameters**

None

Remarks

Tags is a read-only property of type Collection of String.

**Tags Property (Server Object)**

Returns the Tags object for the current server, but only if the authenticated user is a member of the Historian tag administrators group.

**Syntax**

object.**Tags**

**Parameters**

None

**TagsSet Property (DataCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**TagsSet***[ = Boolean]*

**Parameters**

None

**TimeStamp Property (DataFields Object)**

Determines whether the Timestamp field should be returned in the DataRecordset query.

**Syntax**

object.**TimeStamp***[ = Boolean]*

**Parameters**

None

**TimeStamp Property (DataValue Object)**

Returns the timestamp of the DataValue in a localized time.

**Syntax**

object.**TimeStamp**

**Parameters**

None

**Remarks**

TimeStamp is a read-only property of type Date.

**TimeStamp Property (Message Object)**

Returns the time the Message was created.

**Syntax**

object.**TimeStamp***[ = Date]*

**Parameters**

None

**Remarks**

TimeStamp is a writeable property of type Date.

### TimeStamp Property (MessageFields Object)

Determines whether the TimeStamp field should be returned in the MessageRecordset query.

**Syntax**

object.**TimeStamp***[ = Boolean]*

**Parameters**

None

### TimeStampType Property (Tag Object)

Returns or sets the type of time stamping applied to data at collection time.

| Name | Description | Value |
|---|---|---|
| Source | New values take timestamp from data source. | 1 |
| Collector | New values are timestamped by Collector. | 2 |

Changes to tag properties are not committed until you call the WriteRecordset method of the TagRecordset object.

**Syntax**

object.**TimeStampType***[ = ihTimeStampType]*

**Parameters**

None

### TimeStampType Property (TagCriteria Object)

Sets the timestamp type to search for in the TagRecordset query.

**Syntax**

object.**TimeStampType***[ = Boolean]*

**Parameters**

None

**TimeStampType Property (TagFields Object)**

Determines whether the TimeStampType field should be returned in the TagRecordset query.

**Syntax**

object.**TimeStampType***[ = Boolean]*

**Parameters**

None

**TimeStampTypeSet Property (TagCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**TimeStampTypeSet***[ = Boolean]*

**Parameters**

None

**TimeStampTypeUpdated Property (Tag Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**TimeStampTypeUpdated***[ = Boolean]*

**Parameters**

None

**TimeZoneBias Property (Tag Object)**

Returns or sets the time zone bias for the tag. Time zone bias is used to indicate the natural time zone of the tag expressed as an offset from GMT in minutes. Changes to tag properties are not committed until you call the WriteRecordset method of the TagRecordset object.

**Syntax**

object.**TimeZoneBias***[ = Long]*

**Parameters**

None

**TimeZoneBias Property (TagCriteria Object)**

Sets the time zone bias to search for in the TagRecordset query.

**Syntax**

object.**TimeZoneBias***[ = Long]*

**Parameters**

None

**TimeZoneBias Property (TagFields Object)**

Determines whether the TimeZoneBias field should be returned in the TagRecordset query.

**Syntax**

object.**TimeZoneBias***[ = Boolean]*

**Parameters**

None

**TimeZoneBiasUpdated Property (Tag Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**TimeZoneBiasUpdated***[ = Boolean]*

**Parameters**

None

**TimeZoneBiasSet Property (TagCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**TimeZoneBiasSet***[ = Boolean]*

**Parameters**

None

### TimestampTypeList Property (Tags Object)

Returns a list of the valid Timestamp types available in Historian.

**Syntax**

object.**TimestampTypeList**

**Parameters**

None

### Topic Property (Message Object)

Returns the Topic Number of the Message.

**Syntax**

object.**Topic***[ = ihMessageTopic]*

**Parameters**

None

**Remarks**

Topic is a writeable property of type ihMessageTopic.

**Example**

```
If MyMessage.Topic = ihMessageTopic.General Then   Debug.Print "This is a General Message" End If
```

### Topic Property (MessageCriteria Object)

Establishes which Topic the MessageRecordset query will filter by.

**Syntax**

object.**Topic***[ = ihMessageTopic]*

**Parameters**

None

**Remarks**

Topic is a writeable property of type ihMessageTopic.

**Example**

```
MyMessages.Criteria.Topic = ihMessageTopic.General
```

### Topic Property (MessageFields Object)

Determines whether the Topic field should be returned in the MessageRecordset query.

**Syntax**

object.**Topic***[ = Boolean]*

**Parameters**

None

**Remarks**

Topic is a writeable property of type ihMessageTopic.

**Example**

```
Dim MyMessages As iHistorian_SDK.MessageRecordset Set MyMessages = GetServer.Messages.NewRecordset With
 MyMessages.Fields   .Topic = True End With
```

### TopicList Property (Messages Object)

Returns a list of available Topics.

**Syntax**

object.**TopicList***([IncludeMessages], [IncludeAlerts])*

**Parameters**

| Name | Data Type | Description |
|---|---|---|
| IncludeMessages | Boolean | Indicates whether Message topics should be included in the returned list |

| Name | Data Type | Description |
|---|---|---|
| IncludeAlerts | Boolean | Indicates whether Alert topics should be included in the returned list |

### TopicName Property (Message Object)

Returns the Topic Name of the Message.

**Syntax**

object.**TopicName**

**Parameters**

None

**Remarks**

Topic is a writeable property of type String.

**Example**

```
Debug.Print "This is a " + MyMessage.TopicName + " message."
```

### TopicSet Property (MessageCriteria Object)

A flag to indicate whether this property has been set or not.

**Syntax**

object.**TopicSet***[ = Boolean]*

**Parameters**

None

### TrackingEvents Property (OPCFilters Object)

Returns the list of Tracking Event Categories being filtered on the Alarm Collector. This list is only applied if isTrackingEventsOn(true) has been called.

**Syntax**

object.**TrackingEvents**

**Parameters**

None

# Property Reference U-V

**U**

**UserName Property (Message Object)**

Returns the username who generated the message, or who the message is associated with.

**Syntax**

object.**UserName***[ = String]*

**Parameters**

None

**Remarks**

UserName is a writeable property of type String.

**UserName Property (MessageFields Object)**

Determines whether the UserName field should be returned in the MessageRecordset query.

**Syntax**

object.**UserName***[ = Boolean]*

**Parameters**

None

**UserName Property (Server Object)**

Returns the name of the currently authenticated user for the server.

**Syntax**

object.**UserName**

**Parameters**

None

**UsernameSet Property (MessageCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**UserNameSet***[ = Boolean]*

**Parameters**

None

**V**

**Value Property (DataFields Object)**

Determines whether the Value field should be returned in the DataRecordset query.

**Syntax**

object.**Value***[ = Boolean]*

**Parameters**

None

**Value Property (DataValue Object)**

Returns or sets the value of the DataValue.

**Syntax**

object.**Value***[ = Variant]*

**Parameters**

None

**Value Property (Option Object)**

The value parameter of this option expressed as a localized string.

**Syntax**

object.**Value***[ = Variant]*

**Parameters**

None

#### Version Property (Server Object)

Returns the server or client version number.

**Syntax**

object.**Version***([API])*

**Parameters**

| Name | Data Type | Description |
|---|---|---|
| API | Boolean | When True, the server version is returned. When False, the client version is returned. The default is False. '<br><br>When connecting to an Historian v1.0 server and passing True, Historian returns "Unknown" for the version number. |

**Remarks**

Version is a read-only property of type String.

## Property Reference W-Z

### W

#### WatchdogTag Property (Collector Object)

Returns or sets the collector redundancy watchdog tag name.

**Syntax**

object.**WatchdogTag***[ = String]*

**Parameters**

None

#### WatchdogValueMaxUnchangedPeriod Property (Collector Object)

Returns or sets the time in seconds when the collector should automatically failover if a raw sample is not received for the watchdog tag from the primary collector. Use with FailoverOnValueUnchanged property.

**Syntax**

object.**WatchdogValueMaxUnchangedPeriod***[ = Long]*

**Parameters**

None

**WriteSecurityGroup Property (Tag Object)**

Sets the name of the security group controlling the writing of data for the Tag. Changes to tag properties are not committed until you call the WriteRecordset method of the TagRecordset object.

**Syntax**

object.**WriteSecurityGroup***[ = String]*

**Parameters**

None

**WriteSecurityGroup Property (TagCriteria Object)**

Sets the write security group to search for in the TagRecordset query.

**Syntax**

object.**WriteSecurityGroup***[ = String]*

**Parameters**

None

**WriteSecurityGroup Property (TagFields Object)**

Determines whether the WriteSecurityGroup field should be returned in the TagRecordset query.

**Syntax**

object.**WriteSecurityGroup***[ = Boolean]*

**Parameters**

None

**WriteSecurityGroupSet Property (TagCriteria Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**WriteSecurityGroupSet***[ = Boolean]*

**Parameters**

None

**WriteSecurityGroupUpdated Property (Tag Object)**

A flag to indicate whether this property has been set or not.

**Syntax**

object.**WriteSecurityGroupUpdated***[ = Boolean]*

**Parameters**

None

## Method Reference A-B

**APIStatusToString Method (Server Object)**

Converts a status ID into a string.

**Syntax**

object.**APIStatusToString**(nStatus, bHRESULT)

**Table 271. Parameters**

| Name | Data Type | Description |
|---|---|---|
| nStatus | Long | The status ID to be converted. |
| bHRESULT | Boolean | If true, then the status ID is an HRESULT; otherwise, it is an APIStatus. |

**Returns**

String. The translation of the status code.

**Add Method (Archives Object)**

Adds a new archive with the specified characteristics to the Historian server. You can also use this method to add an existing archive file to the archives collection to restore an archive.

Specify the FileLocation in the context of the Historian server machine drive and directory structure. If the specified FileLocation points to an existing valid Historian archive, the method effectively re-registers this archive and automatically loads it. Use this method when you return a previously unloaded archive to the system.

Since the Add request is synchronous, it immediately creates a new Archive object on the Historian server and another one in the Archives collection on the client.

**Syntax**

object.**Add**(ArchiveName As String, FileLocation As String, FileSize As Long, ArchiveDataStoreName As String)

**Table 272. Parameters**

| Name | Data Type | Description |
|---|---|---|
| ArchiveName | String | Name of the archive to add (Read-Only). |
| FileLocation | String | Fully qualified name of the archive file (Read-Only). |
| FileSize | Long | Size of archive in MB (Read-Only). |
| ArchiveDataStore-Name | String | Name of the archive data store the archive belongs to. This is an optional parameter. |

**Returns**

Archive. The newly added Archive object or Nothing.

### Add Method (Collectors Object)

Adds a new collector with the specified name to the Historian server. The collector name must be unique in a given Historian server.

Since the Add request is synchronous, it immediately creates a new Collector Object on the Historian server and another one in the Collectors collection on the client.

**Syntax**

object.**Add**(CollectorName)

**Table 273. Parameters**

| Name | Data Type | |
|---|---|---|
| CollectorName | String | Name of the new collector to add. (Read-only) |

**Returns**

Collector. Returns a reference to the newly created Collector object.

### Add Method (DataRecordset Object)

Adds a new blank DataValue record to the current DataRecordset for the specified tag. The DataValue is written to the server when the WriteRecordset method of the DataRecordset object is called. You can optionally specify the value of the record or specify it after creation by operating on the DataValue object.

**Syntax**

object.**Add**(Tagname, TimeStamp, [InValue])

**Table 274. Parameters**

| Name | Data Type | Definition |
|---|---|---|
| Tagname | String | Tag to which you are adding a new data value (read-only). |
| TimeStamp | Date | Timestamp to add value at (read-only). |
| InValue | DataValue | New value (optional). |

**Returns**

DataValue. Returns a reference to the newly created data value.

### Add Method (DataStores Object)

Adds a data store.

**Syntax**

object.**Add**(MyDataStoreName,MyDefaultDS, MyDataStoreType, MyDataStoreDescription)

**Table 275. Parameters**

| Name | Data Type | |
|---|---|---|
| DataStoreName | String | Indicates the name of the data store. |
| IsDefault | Boolean | Indicates whether the data store is the default data store. |
| DataStoreType | String | Indicates the type of the data store: Historical, User, or SCADA buffer. |
| Description | String | The description provided for the data store. |

**Returns**

DataStore.

### Add Method (EnumeratedSets Object)

Adds an enumerated set to the Enumerated Sets collection. The enumerated set is not added to the Historian Server until the SaveSet Method is called.

**Syntax**

object.**Add**(MySet)

**Table 276. Parameters**

| Name | Data Type | Description |
|---|---|---|
| MySet | EnumeratedSet | The set that has to be added. |

**Returns**

None.

### Add Method (EnumeratedStates Object)

Adds an enumerated state to the enumerated set collection. The enumerated state is not added to the Historian Server until the SaveSet Method is called.

**Syntax**

Object.**Add**(MyState)

**Table 277. Parameters**

| Name | Data Type | Description |
|---|---|---|
| MyState | EnumeratedState | The state that has to be added. |

**Returns**

None.

### Add Method (MessageRecordset Object)

Adds a new blank message record to the current MessageRecordset. The message is written to the server when the WriteRecordset method of the MessageRecordset object is called.

**Syntax**

object.**Add**

**Parameters**

None

**Returns**

The empty message that was added.

### Add Method (TagDependencies Object)

Adds a dependent Tag to the current calculation. The calculation will be triggered when the value of this tag changes.

**Syntax**

object.**Add**(Tagname)

**Table 278. Parameters**

| Name | Data Type | Description |
|---|---|---|
| Tagname | String | Name of the tag to add |

**Returns**

Boolean. Whether the Add operation succeeded.

### Add Method (TagRecordset Object)

Adds a new blank tag record to the current TagRecordset. The tag is written to the server when the WriteRecordset method of the TagRecordset object is called.

#### Syntax

object.**Add**(Tagname)

**Table 279. Parameters**

| Name | Data Type | Description |
|---|---|---|
| Tagname | String | Name of the tag to add |

#### Returns

Returns a reference to the newly created tag.

### Add Method (UserDefinedtypeFields Object)

Adds multiple fields to the User Defined Type. The User Defined Type is not added to the Historian Server until the SaveSet Method is called.

#### Syntax

object.**Add**(MyField)

**Table 280. Parameters**

| Name | Data Type | Description |
|---|---|---|
| MyField | UserDefinedType-Field | The field to be added. |

#### Returns

None.

### AddComment Method (DataValue Object)

Adds a comment to the current DataValue. If you supply a secondary username and password, the SDK authenticates the username and password before the committing the comment.

#### Syntax

object.**AddComment**(Comment, [DataTypeHint], [SecondaryUser], [SecondaryPassword])

**Table 281. Parameters**

| Name | Data Type | Description |
|---|---|---|
| Comment | Variant | The comment to add. |
| DataTypeHint | String | The name of the data type for the comment (optional, read-only). |
| SecondaryUser | String | Supervisor's username to sign comment (optional, read-only). |
| SecondaryPassword | String | Supervisor's password to sign comment (optional, read-only). |

**Returns**

Boolean. Returns True if the AddComment Method operation succeeded.

### AddEx Method (Archives Object)

Adds a new archive with the specified characteristics to the Historian server. You can also use this method to add an existing archive file to the archives collection to restore an archive.

Specify the FileLocation in the context of the Historian server machine drive and directory structure. If the specified FileLocation points to an existing valid Historian archive, the method effectively re-registers this archive and automatically loads it. Use this method when you return a previously unloaded archive to the system.

Since the Add request is synchronous, it immediately creates a new Archive object on the Historian server and another one in the Archives collection on the client.

The window pointed to by hWnd is kept alive (messages processed) while the archive is being added.

**Syntax**

object.**AddEx**(ArchiveName As String, FileLocation As String, FileSize As Long, hWnd As Long, ArchiveDataStoreName As String)

**Table 282. Parameters**

| Name | Data Type | Description |
|---|---|---|
| ArchiveName | String | Name of the archive to add (Read-Only). |
| FileLocation | String | Fully qualified name of the archive file (Read-Only). |
| FileSize | Long | Size of archive in MB (Read-Only). |
| hWnd | Long | The window handle to keep alive. |
| ArchiveDataStore-Name | String | Name of the archive data store the archive belongs to. This is an optional parameter. |

**Returns**

Archive. The newly added Archive object or Nothing.

### AddServer Method (ServerManager Object)

Registers a new server to the list of registered servers on the client. It makes an attempt to authenticate the connection based on the username and password supplied. If a username and password are not supplied, it assumes the user to be the currently authenticated domain user.

If it cannot authenticate the connection, because of incorrect user information or an invalid ServerName, the AddServer method returns False and does not register the supplied ServerName on the client.

**Syntax**

object.**AddServer**(ServerName, [UserName], [Password], [ConnectionTimeout])

**Table 283. Parameters**

| Name | Data Type | Description |
|---|---|---|
| ServerName | String | Computer name of the Historian server (read-only). |
| UserName | String | Username to authenticate (optional, read-only). |
| Password | String | Password to authenticate (optional, read-only). |
| Connection-Timeout | Long | The maximum length of time clients should wait for messages from the server before concluding the server is unavailable (optional, read-only). |

**Returns**

Boolean. Returns True if the AddServer operation succeeded.

**Example**

```
Dim MyManager As New iHistorian_SDK.ServerManager

Dim MyServer As iHistorian_SDK.Server

' Try to add the new server

If MyManager.AddServer("USGB014") Then

Set MyServer = MyManager.Servers("USGB014") Else

err.Raise 1, , "Failed to authenticate new server" End If
```

### AlarmAttributesRecordSet Method (Alarms Object)

This function returns a list of all the Vendor Attributes in the Alarm Archiver.

**Syntax**

object.**AlarmAttributesRecordSet**

**Parameters**

None.

**Returns**

*AlarmAttributes*. The AlarmAttributes object contains the list of vendor attributes.

### AlarmRecordSet Method (Alarms Object)

This function is used to query alarms or events from the archiver. This function takes one parameter, an AlarmOpenRecordSetInfo object, which specifies the query. The following sections describe how to populate this object to perform an Alarm or Event query.

**AlarmOpenRecordSetInfo Parameter**

This object contains details on what to select, and any filters for the query. After creating the object, you first specify which fields to select, and which fields to sort (SelectFields), and then specify any criteria/filter to apply (AlarmCriteria).

Create the object: `Dim myAlarmOpenRecordSetInfo As New AlarmOpenRecordSetInfo`

**Alarm Fields**

The following is a list of the alarm fields that can be returned in the recordset, or used to sort or filter the recordset. These are used in the SelectFields and AlarmCriteria object (see below for more details).

**Table 284.**

| Option | Data Type | Description |
|---|---|---|
| AlarmID | Long | The unique ID of the alarm or event in the Historian alarm database. |
| ItemID | String | The OPC ItemID of the alarm. This contains the source address of the data access tag the alarm is associated with. This could contain a NULL value if the alarm is not associated with a tag. |
| Source | String | This is the unique identifier used by the OPC AE Collector for the alarm or event. |
| DataSource | String | The collector interface name associated with the alarm or event. |
| Tagname | String | The Historian Tag Name associated with the alarm. The tag name will be NULL unless the tag is also collected by Historian. |
| EventCategory | String | The OPC event category of the alarm or event. |
| ConditionName | String | The OPC condition of the alarm. This does not apply to event data. This, combined with the Source, comprises an alarm. |
| SubCondition-Name | String | The OPC sub-condition of the alarm. This does not apply to event data. This is the state of the alarm. |
| StartTime | Date | The start time or time stamp of the alarm or event. |
| EndTime | Date | The end time of the alarm. This does not apply to event data. |
| AckTime | Date | The time the alarm was acknowledged. This does not apply to event data. |

**Table 284. (continued)**

| Option | Data Type | Description |
|---|---|---|
| Timestamp | Date | The time stamp of the alarm or event. |
| Message | String | The message attached to the alarm or event. |
| Acked | Boolean | Stores the acknowledgement status of the alarm. If the alarm is acknowledged, this will be set to TRUE. |
| Severity | Long | The severity of the alarm or event. This is stored as an integer value with a range of 1-1000. |
| Actor | String | The operator who acknowledged the alarm, or caused the tracking event. |
| Quality | String | The quality of the alarm or event. |

**AlarmOpenRecordSetInfo.SelectFields**

Each of the fields above can be selected, or used to sort the alarm recordset. Fill in the AlarmOpenRecordSetInfo.SelectFields for each field, according to the following table.

| Option | Data Type | Description |
|---|---|---|
| Select | Boolean | True to select the field. |
| OrderBy | Boolean | True to order by the field. |
| OrderPriority | Integer | Relative priority compared to other field order priorities. Highest first. Ties are random. |
| Descending | Boolean | True to sort descending. OrderBy must also be True. |

**Example**

Set up SelectFields to select the AlarmId and ItemId. Order by the AlarmId, then the ItemId.

```
myAlarmOpenRecordSetInfo.SelectFields.AlarmId.Select = TRUE

myAlarmOpenRecordSetInfo.SelectFields.AlarmId.OrderBy = TRUE

myAlarmOpenRecordSetInfo.SelectFields.AlarmId.OrderPriority = 10

myAlarmOpenRecordSetInfo.SelectFields.ItemId.Select = TRUE
```

```
myAlarmOpenRecordSetInfo.SelectFields.ItemId.OrderBy = TRUE

myAlarmOpenRecordSetInfo.SelectFields.ItemId.OrderPriority = 20
```

The SelectFields object has one other field, AllFields, which can be used to select all fields.

```
SelectFields.AllFields = TRUE
```

**Choose Query Type**

For all queries, the type of query should be specified. If it is not specified, the default is to return a record set containing the events. The query type is specified by setting the AlarmOpenRecordSetInfo.AlarmCriteria.AlarmType object.

```
Query Alarms: myAlarmOpenRecordSetInfo.AlarmType = ihALARM_CONDITION

Query Events: myAlarmOpenRecordSetInfo.AlarmType = ihALARM_TRACKING

Query Historical Alarm Transitions: myAlarmOpenRecordSetInfo.AlarmType = ihALARM_CONDITION_HIST
```

**Add filters**

Similar to the SelectFields object, each field listed above (Alarm Fields) can have criteria associated with it. This criteria is used to filter the record set, so for example, you can ask for all alarms with severity greater than 500. The type of Criteria that can be applied is dependant on the Data Type of the field. The following table lists the available criteria:

| Field Data Type | Criteria Types |
|---|---|
| Integer/Long | Min, Max, Equal, NotEqual |
| Float | Min, Max, Equal, NotEqual |
| String | StringMask |
| Quality | Min, Max, Equal, NotEqual |

For example, the Severity field is of type Long. Therefore, Min, Max, Equal, or NotEqual criteria can be specified for the severity. To query where severity is greater than 500:

```
myAlarmOpenRecordSetInfo.AlarmCriteria.SeverityCriteria.Min = 500
```

**Specify Max Records**

An optional criteria, MaxRecords, can be set to set the upper limit size of the recordset.

Example: Return the first 10 records:

```
myAlarmOpenRecordSetInfo.AlarmCriteria.MaxRecords = 10
```

**Note:**
To avoid having random rows returned, if you specify a MaxRecords criteria, you should generally also specify an Order By (see SelectFields above).

**Syntax**

object.**AlarmRecordSet**(theAlarmOpenRecordSetInfo)

**Table 285. Parameters**

| Name | Data Type | Description |
|---|---|---|
| theAlarmOpenRecordSetInfo | Variant | See above for details. |

**Returns**

AlarmRecordset. An AlarmRecordset object, populated according the details specified in the AlarmOpenRecordSetInfo parameter.

**Example**

```
Dim MyServer As iHistorian_SDK.Server

Set MyServer = GetServer

Dim myAlarmOpenRecordSetInfo As AlarmOpenRecordSetInfo

' Select the last 100 alarms in the alarm history table

myAlarmOpenRecordSetInfo.SelectFields.AllFields = True

myAlarmOpenRecordSetInfo.SelectFields.Timestamp.OrderBy = True

myAlarmOpenRecordSetInfo.SelectFields.Timestamp.Descending = True

myAlarmOpenRecordSetInfo.AlarmCriteria.AlarmType = ihALARM_CONDITION_HIST

myAlarmOpenRecordSetInfo.AlarmCriteria.MaxRecords = 100

Dim myAlarmRecordSet As AlarmRecordSet

Set myAlarmRecordSet = MyServer.Alarms.AlarmRecordSet(myAlarmOpenRecordSetInfo)
```

## AllFields Method (DataFields Object)

Specifies that all DataFields should be returned in the DataRecordset query.

**Syntax**

Syntax object.**AllFields**

**Parameters**

None.

**Returns**

None.

### AllFields Method (MessageFields Object)

Specifies that all MessageFields should be returned in the MessageRecordset query.

**Syntax**

object.**AllFields**

**Parameters**

None.

**Returns**

None.

### AllFields Method (TagFields Object)

Sets all fields for retrieval in the TagRecordset query.

**Syntax**

object.**AllFields**

**Parameters**

None.

**Returns**

None.

### Backup Alarms Method

Backs up or saves a copy of the alarms to an offline file.

**Syntax**

object.**BackupAlarms**

**Table 286. Parameters**

| Name | Data Type | Description |
|---|---|---|
| BackupFile | String | The file that stores the backup alarm data. |
| StartTime | Date | Start time of the alarm data. |
| EndTime | Date | End time of the alarm data. |

**Returns**

Boolean. Returns TRUE if the alarms are backed up.

**Example**

```
Dim Status As ihStatus

Dim ReturnStatus As Boolean

Dim AlarmsStartTime As ihTimeStruct

Dim AlarmsEndTime As ihTimeStruct Status = ihSTATUS_FAILED ReturnStatus = False

AlarmsStartTime = Date_To_UTC(StartTime) AlarmsEndTime = Date_To_UTC(EndTime)

On Error GoTo errc

Status = ihBackupAlarms(MyServer.Handle, BackupFile, AlarmsStartTime, AlarmsEndTime, 0)

If Status <> ihSTATUS_OK Then err.Raise 1, , "Error while performing Backup alarms [" +

 ErrorDescription(Status) ReturnStatus = True

BackupAlarms = ReturnStatus errc:

zLastError = "Backup Alarms >> " + err.Description

BackupAlarms = ReturnStatusEnd Function
```

## Backup Method (Archive Object)

Performs an online backup of the specified archive by stopping inbound data flow and copying an image of the specified archive into the file identified by the BackupFileName parameter.

When the online backup operation completes, the archive returns to normal operation and accepts inbound data flow.

**Syntax**

object.**Backup**(BackupFileName As String, Optional DataStoreName As String)

**Table 287. Parameters**

| Name | Data Type | Description |
|---|---|---|
| BackupFileName | String | Fully qualified name of the backup file (read-only). |
| DataStoreName | String | Name of the data store to which the backup file belongs. This is an optional parameter. |

**Returns**

Boolean. Returns whether or not the BackupFile operation succeeded.

**Example**

```
Dim BackupFilename As String

BackupFilename = ArchiverLauncher.Archives & "\TheCurrentArchive.ZIP"

Set MyArchives = GetServer.Archives

' Find The Current Archive, Then Initiate A Backup

With MyArchives

  For I = 1 To .Item.count

    If .Item(I).IsCurrent Then

      If Not .Item(I).Backup(BackupFilename) Then

        err.Raise 1, "Backup", "Backup Failed"

      End If

    End If

  Next I

End With
```

## BackupEx Method (Archive Object)

Performs an online backup of the specified archive by stopping inbound data flow and copying an image of the specified archive into the file identified by the BackupFileName parameter.

When the online backup operation completes, the archive returns to normal operation and accepts inbound data flow.

The Ex method will process messages for a client window while waiting for the backup to complete (so clients do not appear to be frozen.

**Syntax**

object.**BackupEx**(BackupFileName As String, hWnd As Long, DataStoreName As String)

**Table 288. Parameters**

| Name | Data Type | Description |
|---|---|---|
| BackupFileName | String | Fully qualified name of the backup file (read-only) |
| hWnd | Long | The handle of the window to keep alive. |
| DataStoreName | String | Name of the data store. This is an optional parameter. |

**Returns**

Boolean. Returns whether or not the BackupFile operation succeeded.

**Example**

```
Dim BackupFilename As String
BackupFilename = ArchiverLauncher.Archives & "\TheCurrentArchive.ZIP" Set MyArchives =
 GetServer.Archives
' Find The Current Archive, Then Initiate A Backup
With MyArchives
  For I = 1 To .Item.count
    If .Item(I).IsCurrent Then
      If Not .Item(I).Backup(BackupFilename)
        Then err.Raise 1, "Backup", "Backup Failed"
      End If
    End If
  Next I
End With
```

## Browse Method (OPCBrowse Object)

Populates the Browse Object with sources and areas for the current browse position.

**Syntax**

object.**Browse**(theServer, theCollector)

**Table 289. Parameters**

| Name | Data Type | Description |
|---|---|---|
| theServer | Variant | The Server Object for the Historian Server. |
| theCollector | Variant | The Collector Object for the Collector to Browse. |

**Returns**

Boolean. Returns whether or not the Browse operation succeeded.

### BrowseCollector Method (TagRecordset Object)

Browses a collector for its available tags.

**Syntax**

object.**BrowseCollector**(CollectorName, AdditionsOnly, SourceFilter, DescriptionFilter, [BrowsePosition], [Recursive])

**Table 290. Parameters**

| Name | Data Type | Description |
|---|---|---|
| CollectorName | String | The name of the collector to browse. |
| AdditionsOnly | Boolean | Browse only tags that are additions to the Historian server. |
| SourceFilter | String | The tag source address filter. |
| DescriptionFilter | String | The tag description filter. |
| BrowsePosition | String | The browse position when performing an OPC hierarchical browse (optional, default = ""). |
| Recursive | Boolean | Whether to perform an OPC hierarchical browse (optional, default = False). |

**Returns**

Boolean. Success/Failure.

### BrowseTags Method (Collector Object)

BrowseTags is available for collection in the data source. The tags can then be added to the archiver using the WriteRecordSet Method.

**Syntax**

object.**BrowseTags**([AdditionsOnly], [SourceFilter], [DescriptionFilter], [BrowsePosition], [Recursive])

**Table 291. Parameters**

| Name | Data Type | Description |
|---|---|---|
| AdditionsOnly | Boolean | Browse only tags that are additions to the Historian server (optional, default = True). |
| SourceFilter | String | The tag source address filter (optional, default = ""). |
| DescriptionFilter | String | The tag description filter (optional, default = ""). |
| BrowsePosition | String | The browse position when performing an OPC hierarchical browse (optional, default = ""). |
| Recursive | Boolean | Whether to perform an OPC hierarchical browse (optional, default = False). |

**Returns**

TagRecordset. Returns a reference to the tag record that resulted from browsing the selected collector.

**Example**

```
Dim MyNewTags As iHistorian_SDK.TagRecordset

' Request The Collector To Browse Its Tag Source

Set MyNewTags = MyCollectors.Item("USIM031_OPC1").BrowseTags

' Modify Tag Records In Recordset To Add Additional Configuration Information

' Go Ahead And Add New Tags To System

If Not MyNewTags.WriteRecordset Then

  err.Raise 1, , "TagRecordset.WriteRecord failed: " + MyNewTags.LastError

End If
```

# Method Reference C-D

## C

### Clear Method (DataCriteria Object)

Clears any previously supplied criteria for the DataRecordset query. Use this method to initialize the DataRecordset query.

**Syntax**

object.**Clear**

**Parameters**

None.

**Returns**

None.

### Clear Method (DataFields Object)

Clears all DataFields from being returned in the DataRecordset query. Use this method to initialize the DataRecordset query.

**Syntax**

object.**Clear**

**Parameters**

None.

**Returns**

None.

### Clear Method (Enumerated Sets Object)

Applies to:

Clears all the elements from the EnumeratedSets collection object and create a new instance of the EnumeratedSets collection.

**Syntax**

object.**Clear**

**Parameters**

None.

**Returns**

None.

**Clear Method (EnumeratedStates Object)**

Applies to:

Clears all the states from the enumerated set collection.

**Syntax**

object.**Clear**

**Parameters**

None.

**Returns**

None.

**Clear Method (MessageCriteria Object)**

Clears all previously supplied criteria for the MessageRecordset query. Use this method to initialize the MessageRecordset query.

**Syntax**

object.**Clear**

**Parameters**

None.

**Returns**

None.

**Clear Method (MessageFields Object)**

Clears all previously supplied criteria for the MessageRecordset query. Use this method to initialize the MessageRecordset query.

**Syntax**

object.**Clear**

**Parameters**

None.

**Returns**

None.

### Clear Method (TagCriteria Object)

Clears previously entered criteria for the TagRecordset query. Use this method to initialize the TagRecordset query.

**Syntax**

object.**Clear**

**Parameters**

None.

**Returns**

None.

### Clear Method (TagFields Object)

Clears all fields for retrieval in the TagRecordset query.

**Syntax**

object.**Clear**

**Parameters**

None.

**Returns**

None.

**Returns**

### Clear Method (UserDefinedTypeFields Object)

Applies to:

Clears all the fields from the User Defined Type.

**Syntax**

object.**Clear**

**Parameters**

None.

**Returns**

None.

**Returns**

### ClearRecordset Method (DataRecordset Object)

Clears specific records from the DataRecordset without deleting them from the Historian server.

You can specify a tagname to remove all DataValues for a specific tag, or supply a specific timestamp to remove a single DataValue.

**CAUTION:**
You can clear the entire DataRecordset by omitting both the tagname and timestamp.

**Syntax**

object.**ClearRecordset**([Tagname], [TimeStamp])

**Table 292. Parameters**

| Name | Data Type | Description |
|---|---|---|
| Tagname | String | Tag to clear records for (optional, read-only). |

**Table 292. Parameters (continued)**

| Name | Data Type | Description |
|---|---|---|
| TimeStamp | Date | Timestamp to clear records for (optional, read-only). |

**Returns**

Boolean. Returns True if the ClearRecordset operation succeeded.

### ClearRecordset Method (TagRecordset Object)

Clears specific records from the TagRecordset without deleting them from the Historian server. You can specify a tagname to delete a specific record.

> **CAUTION:**
> You can clear the entire TagRecordset by omitting the tag name.

**Syntax**

object.**ClearRecordset**([Tagname])

**Table 293. Parameters**

| Name | Data Type | Description |
|---|---|---|
| Tagname | String | Name of tag to remove from record set (read-only, optional). |

**Returns**

Boolean. Success/Failure.

### CloseAlarms Method (Alarms Object)

This function sends a special alarm request to close alarms on a specific collector before a specific date. This is useful to close out any alarms that are stuck mid-lifecycle.

**Syntax**

object.**CloseAlarms**(endDate, theCollector)

**Table 294. Parameters**

| Name | Data Type | Description |
|---|---|---|
| endDate | Date | The date from which to close alarms. Alarms before this date are closed. |
| theCollector | String | Close alarms associated with this collector/datasource.<br><br>**Note:**<br>If your alarm collector is linked to a data collector, pass in the data collector name. |

**Returns**

None.

### CloseArchive Method (Archive Object)

Closes the specified archive for new data flow. You can perform this operation on the current archive only (the one accepting the newest data). Once an archive is closed, the Historian server looks to the next available empty archive to re-commence data storage.

If no empty archive is found, the method creates one or overwrites the oldest loaded archive. The archiving options, ArchiveAutomaticCreate, ArchiveAutomaticFreeSpace, ArchiveDefaultSize, and ArchiveOverWriteOld, control this behavior.

If ArchiveAutomaticCreate is running and enough free space exists on the drive of the ArchiveDefaultPath, the method creates a new archive of default size after it creates an archive of default size.

If the method cannot create a new archive, it overwrites the oldest archive based on ArchiveOverWriteOld being on. If ArchiveOverWriteOld is not on, the Historian archiver shuts down and the collectors start buffering data.

**Syntax**

object.**CloseArchive**(DataStoreName As String)

**Table 295. Parameters**

| Name | Data Type | Description |
|---|---|---|
| DataStoreName | String | The name of the data store the archive belongs to. This is an optional parameter. |

**Returns**

Boolean. Returns whether or not the CloseArchive operation succeeded.

### CollectorHasBackup Method (Collectors Object)

Returns whether a collector has a backup collector.

**Syntax**

object.**CollectorHasBackup**(CollectorName)

**Table 296. Parameters**

| Name | Data Type | Description |
|---|---|---|
| CollectorName | String | Name of the collector to determine if a backup exists. |

**Returns**

Boolean. True if the Collector has a backup, False otherwise.

**Example**

```
Dim MyCollectors As iHistorian_SDK.Collectors

Set MyCollectors = MyServer.Collectors

Dim HasBackup As Boolean

HasBackup = MyCollectors.CollectorHasBackup("SimulationCollector")
```

### CommitImport Method (Alarms Object)

Writes the alarms/events acquired via the Import method to the Historian Archiver.

**Syntax**

object.**CommitImport**

**Parameters**

None

**Returns**

Boolean. Returns whether or not the Import operation succeeded.

### Connect Method (Server Object)

Initiates a connection to the current server. Calling the Connect method on a currently connected session re-authenticates the user.

If a ServerName has been set, the method authenticates the user by the username and password supplied, if any. If they are not supplied, it authenticates the user by the currently authenticated domain user. If a ServerName has not been set, it uses the ServerName registered as the Default Server. If a username and password are not supplied, it uses the username and password registered with the default server. If neither establishes a username, it authenticates the user by the currently authenticated domain user.

**Syntax**

object.**Connect**([ServerName], [UserName], [Password])

**Table 297. Parameters**

| Name | Data Type | Description |
|---|---|---|
| ServerName | String | Table text |
| UserName | String | Username to authenticate (optional). |
| Password | String | Password to authenticate (optional). |

**Returns**

Boolean. Returns whether or not the Connect operation succeeded.

**Example**

```
Dim MyServer As New iHistorian_SDK.Server

' Connect to the default server using default user

If Not MyServer.Connect Then

err.Raise 1, , "Failed to authenticate on server " + MyServer.ServerName

End If

' Connect to the default server using specific user

If Not MyServer.Connect("Fred", "000") Then
```

```
err.Raise 1, , "Failed to authenticate on server " + MyServer.ServerName

End If

' Connect to specific server using specific user

If Not MyServer.Connect("USGB014", "Fred", "000") Then

err.Raise 1, , "Failed to authenticate on server " + MyServer.ServerName

End If
```

### ConvertShortcutToTime Method (Server Object)

Converts a time shortcut to a date. It converts the time from UTC into formatted time based on the ConnectionOptions (TimeOption) settings listed below.

| Value | Description |
|---|---|
| (N)ow | The current time (absolute). |
| (T)oday | Today at midnight (absolute). |
| (Y)esterday | Yesterday at midnight (absolute). |
| (D)ays | Number of Days (relative). |
| (M)in | Number of Minutes (relative). |
| (H)our | Number of Hours (relative). |
| (W)eek | Number of Weeks (relative). |
| (BOM) | Beginning of this month at Midnight (absolute). |
| (EOM) | Last Day of this month at Midnight (absolute). |
| (BOY) | First Day of this year at Midnight (absolute). |
| (EOY) | Last Day of this year at Midnight (absolute). |

#### Syntax

object.**ConvertShortcutToTime**(Shortcut)

**Table 298. Parameters**

| Name | Data Type | Description |
|---|---|---|
| Shortcut | String | Date of shortcut to convert (read-only). |

#### Returns

Date. Returns date converted from string shortcut.

### CopyTo Method (Tag Object)

Copies the properties of the given tag to the specified destination tag.

#### Syntax

object.**CopyTo**(TargetTag)

**Table 299. Parameters**

| Name | Data Type | Description |
|---|---|---|
| TargetTag | Variant | The destination tag. |

#### Returns

None.

### CriteriaFromStrings Method (QueryModifiers Object)

Applies to:

Returns the sampling mode, calculation mode and query modifiers associated with the input CriteriaString.

#### Syntax

object.**CriteriaFromStrings**(CriteriaString As String)

**Table 300. Parameters**

| Name | Data Type | Description |
|---|---|---|
| CriteriaString | String | Indicates the criteria string. For example, #ONLYGOOD. |

#### Returns

DataCriteria

## D

### DataStoreUpdate Method (DataStores Object)

Updates the changes made to the data store settings.

**Syntax**

object.**MyDataStores.Item**(DataStoreName)

**Table 301. Parameters**

| Name | Data Type | Description |
|---|---|---|
| DataStoreName | String | Name of the data store that should be updated. |
| IsDefault | Boolean | Indicates whether the data store is the default data store. |
| Description | String | Description of the data store. This is an optional parameter. |
| StorageType | String | Indicates whether the storage type is historical or SCADA buffer. This is an optional parameter. |

**Returns**

Boolean. Returns TRUE if the data store has been updated and FALSE if there is an error in updating the data store.

### Delete Method (Archive Object)

Deletes the specified archive on the Historian server. This is a synchronous operation that executes immediately when you call the Delete Method.

**Syntax**

object.**Delete**(DataStoreName As String)

**Table 302. Parameters**

| Name | Data Type | Description |
|---|---|---|
| DataStoreName | String | Name of the data store that contains the archive. This is an optional parameter. |

**Returns**

Boolean. Returns whether or not the Delete operation succeeded.

### Delete Method (Archives Object)

Attempts to delete an Archive from the Historian server.

#### Syntax

object.**Delete**(ArchiveName As String, ArchiveDataStoreName As String)

**Table 303. Parameters**

| Name | Data Type | Description |
|---|---|---|
| ArchiveName | String | Name of the archive to delete. |
| ArchiveDataStore-Name | String | Name of the archive data store the archive belongs to. This is an optional parameter. |

#### Returns

Boolean. Returns whether or not the Delete operation succeeded.

### Delete Method (Collector Object)

Deletes the specified collector on the Historian Server.

> **CAUTION:**
> The default option also deletes all tags marked with this collector as their source (CollectorName Property) at the same time it deletes the collector.

#### Syntax

object.**Delete**([DeleteTags])

**Table 304. Parameters**

| Name | Data Type | Description |
|---|---|---|
| DeleteTags | Boolean | Deletes collector tags when deleting the collector. (optional, default = True) |

#### Returns

Boolean. Whether the Delete was successful.

### Delete Method (Collectors Object)

Removes an existing collector with the specified name from the Historian server.

**Syntax**

object.**Delete**(CollectorName, [DeleteTags])

**Table 305. Parameters**

| Name | Data Type | Description |
|---|---|---|
| CollectorName | String | Name of the collector to Delete. |
| DeleteTags | Boolean | Should the tags from this collector be deleted as well? |

**Returns**

Boolean. Success/Failure.

**Example**

```
Dim MyCollectors As iHistorian_SDK.Collectors

Set MyCollectors = MyServer.Collectors

MyCollectors.Delete "OPC Collector"
```

### Delete Method (DataRecordset Object)

Marks the specified tag in the current TagRecordset for deletion. If the specified tag does not exist in the current TagRecordset, the method adds it. If the tag is not found, the Delete method fails.

In either case, the method does not delete this tag on the Historian server until the WriteRecordset Method of the TagRecordset object is called.

**Syntax**

object.**Delete**(Tagname, TimeStamp)

**Table 306. Parameters**

| Name | Data Type | Description |
|---|---|---|
| Tagname | String | Name of the tag to delete (read-only). |
| TimeStamp | Date | Tag timestamp (optional). |

**Returns**

Boolean. Returns True if successful.

**Delete Method (DataStores Object)**

Deletes the data store.

**Syntax**

object.**Delete**(DataStoreName As String)

**Table 307. Parameters**

| Name | Data Type | Description |
|---|---|---|
| DataStoreName | String | Indicates the name of the data store that has to be deleted. |

**Returns**

Boolean. Returns TRUE if the data store has been deleted and FALSE if there is an error in deleting the data store.

**Delete Method (DataValue Object)**

Deletes the DataValue from the archive. Commit the Delete operation by calling the WriteRecordset method of the DataRecordset object.

**Syntax**

object.**Delete**

**Parameters**

None

**Returns**

Boolean. Returns whether or not the Delete operation succeeded.

**Delete Method (EnumeratedSets Object)**

Deletes the specified set from the Historian Server. This is a synchronous operation that is executed immediately when you call the Delete method.

Syntax

object.**Delete**(setName)

**Table 308. Parameters**

| Name | Data Type | Description |
|---|---|---|
| SetName | Variant | The name of the set that has to be deleted. |

**Returns**

Boolean. Returns True if the method has been deleted

### Delete Method (EnumeratedStates Object)

Deletes the specified state from the enumerated set. This is a synchronous operation that is executed immediately when you call the Delete method.

**Syntax**

object.**Delete**(StateName)

**Table 309. Parameters**

| Name | Data Type | Description |
|---|---|---|
| StateName | Variant | The name of the state that has to be deleted. |

**Returns**

Boolean. Returns True if the state has been deleted or False if not.

### Delete Method (Tag Object)

Deletes the specified tag on the Historian Server. This is a synchronous operation that executes immediately when you call the Delete method. You can choose to delete a tag permanently from the Historian Server by passing an additional parameter as True or False.

**Note:**
This method is called within the WriteRecordset method. For more information, refer to the Sample Code section.

**Syntax**

object.**Delete**((Optional) DeletePermanent)

**Table 310. Parameters**

| Name | Data Type | Description |
|---|---|---|
| DeletePermanent | Boolean | (Optional) Pass TRUE to permanently delete a tag. |

**Returns**

Boolean. Returns whether or not the Delete operation succeeded.

### Delete Method (TagRecordset Object)

Marks the specified tag in the current TagRecordset for deletion. If the specified tag does not exist in the current TagRecordset, the method adds it. If the tag is not found, the Delete method fails. In either case, the method does not delete this tag on the Historian server until the WriteRecordset method of the TagRecordset object is called.

**Syntax**

object.**Delete**(Tagname)

**Table 311. Parameters**

| Name | Data Type | Description |
|---|---|---|
| Tagname | String | Name of the tag to delete (read-only). |

**Returns**

Boolean. Returns whether or not the Delete operation succeeded.

### Delete Method (UserDefinedTypeFields Object)

Deletes a field from the User Defined Type. This is a synchronous operation that is executed immediately when you call the Delete method.

**Syntax**

object.**Delete**(FieldName)

**Table 312. Parameters**

| Name | Data Type | Description |
|---|---|---|
| FieldName | Variant | The name of the state that has to be deleted. |

**Returns**

Boolean. Returns TRUE if the state has been deleted.

### DeleteEx Method (Archive Object)

Deletes the specified archive on the Historian server. This is a synchronous operation that executes immediately when you call the Delete Method. The window associated with the handle passed in is kept alive (messages processed) while the operation takes place.

**Syntax**

object.**DeleteEx**(hWnd As Long, DataStoreName As String)

**Table 313. Parameters**

| Name | Data Type | Description |
|---|---|---|
| hWnd | Long | The handle of the window to keep alive. |
| DataStoreName | String | Name of the data store that contains the archive. This is an optional parameter. |

**Returns**

Boolean. Returns whether or not the Delete operation succeeded.

### DeleteEx Method (Archives Object)

Attempts to delete an Archive from the Historian server. Keeps the window with handle hwnd alive by processing its messages while waiting for the deletion to occur.

**Syntax**

object.**DeleteEx**(ArchiveName As String, hWnd As Long, ArchiveDataStoreName As String)

**Table 314. Parameters**

| Name | Data Type | Description |
|---|---|---|
| ArchiveName | String | Name of the archive to delete |
| hWnd | Long | The window handle to keep alive. |
| ArchiveDataStore-Name | String | Name of the archive data store the archive belongs to. This is an optional parameter. |

**Returns**

Boolean. Success/Failure.

**Disconnect Method (Server Object)**

Disconnects the currently authenticated connection to the Server.

**Syntax**

object.**Disconnect**

**Parameters**

None

**Returns**

None.

# Method Reference E-H

**E**

**Export Method (Alarms Object)**

Exports the contents of the AlarmRecordSet into the specified file.

**Syntax**

object.**Export**(AlmRS, RSInfo, FileName, FileFormat)

**Table 315. Parameters**

| Name | Data Type | Description |
|---|---|---|
| AlmRS | AlarmRecordSet | Returns the requested records (writeable). |
| RSInfo | AlarmOpenRecord-SetInfo | Contains descriptions of the desired records (read-only). |
| FileName | String | Fully qualified export filename (read-only). |
| FileFormat | ihFileFormat | File format of the export file (read-only). |

**Returns**

Boolean. Returns whether or not the Export operation succeeded.

**Export Method (DataRecordset Object)**

Exports the contents of the current DataRecordset into the specified file.

**Syntax**

object.**Export**(FileName, FileFormat)

**Table 316. Parameters**

| Name | Data Type | Description |
|---|---|---|
| FileName | String | Fully qualified export filename (read-only. |
| FileFormat | ihFileFormat | File format of the export file (read-only). |

**Returns**

Boolean. Returns whether or not the Export operation succeeded.

**Example**

```
' Export File From Existing Query Results

If Not MyRecordset.Export("C:\Temp\DataReport.RPT", ihFileFormat.Report) Then

Err.Raise 1, , "Error Exporting File: " + MyRecordset.LastError

End If
```

### Export Method (EnumeratedSets Object)

Applies to:

Exports the contents of the EnumeratedSets collection into the specified file.

The following file formats are supported:

| Name | Description | Value |
|---|---|---|
| CSV | File is imported/exported as comma separated values. | 1 |
| XML | File is imported/exported as XML. | 2 |

Imported files follow a specific format that contains specific keywords. With CSV, the first row of the file establishes the fields in the file and their positions. With XML reports, the files describe the format of the data.

**Syntax**

object. **Export**(FileName, FileFormat)

**Table 317. Parameters**

| Name | Data Type | Description |
|---|---|---|
| FileName | String | Fully qualified export file name (read-only). |
| FileFormat | ihFileFormat | File format of the export file (read-only). |

**Returns**

Boolean. Returns True if the sets have been exported successfully.

**Export Method (MessageRecordset Object)**

Exports the contents of the current MessageRecordset into the specified file.

The following file formats are supported. Exported files follow a specific format containing specific keywords. With CSV and tabular reports, the first row of the file establishes the fields in the file and their positions. With XML, the file itself describes the format of the data.

| Name | Description | Value |
|---|---|---|
| CSV | File is imported/exported as comma separated values. | 1 |
| XML | File is imported/exported as XML. | 2 |
| Report | File is exported as a columnar report | 3 |

**Syntax**

object.**Export**(FileName, FileFormat)

**Table 318. Parameters**

| Name | Data Type | Description |
|---|---|---|
| FileName | String | Fully qualified export file name (read-only). |
| FileFormat | ihFileFormat | File format of the export file (read-only). |

**Returns**

Boolean. Success/Failure.

**Export Method (TagRecordset Object)**

Exports the contents of the current TagRecordset into the specified file.

The following file formats are supported:

| Name | Description | Value |
|---|---|---|
| CSV | File is imported/exported as comma separated values. | 1 |
| XML | File is imported/exported as XML. | 2 |
| Report | File is exported as a columnar report | 3 |

Imported files follow a specific format that contains specific keywords. With CSV and tabular reports, the first row of the file establishes the fields in the file and their positions. With XML reports, the files describe the format of the data. Prepare a file for import by exporting it with the desired fields for import.

**Syntax**

object.**Export**(FileName, FileFormat)

**Table 319. Parameters**

| Name | Data Type | Description |
|---|---|---|
| FileName | String | Fully qualified export file name (read-only). |
| FileFormat | ihFileFormat | File format of the export file (read-only). |

**Returns**

Returns whether or not the Export operation succeeded.

**Example**

```
' Export file from existing query results

Set MyRecordset = MyServer.Tags.NewRecordset

MyRecordset.Criteria.Tagname = "*"

MyRecordset.Fields.AllFields

MyRecordset.QueryRecordset

If Not MyRecordset.Export(Path & "TagReport.RPT", ihFileFormat.Report) Then

err.Raise 1, , "Error exporting file: " & MyRecordset.LastError

End If
```

### Export Method (UserDefinedType Object)

Applies to:

Exports the contents of the User Defined Type into the specified file.

The following file formats are supported:

| Name | Description | Value |
|---|---|---|
| CSV | File is imported/exported as comma separated values. | 1 |
| XML | File is imported/exported as XML. | 2 |

Imported files follow a specific format that contains specific keywords. With CSV, the first row of the file establishes the fields in the file and their positions. With XML reports, the files describe the format of the data.

#### Syntax

object. **Export**(FileName, FileFormat)

**Table 320. Parameters**

| Name | Data Type | Description |
|---|---|---|
| FileName | String | Fully qualified export file name (read-only). |
| FileFormat | ihFileFormat | File format of the export file (read-only). |

#### Returns

Boolean. Returns TRUE if the User Defined Type is exported successfully.

## G

### GetCurrentValue Method (Collector Object)

Returns the current value for a given tag.

#### Syntax

object.**GetCurrentValue**(SourceAddress, ErrorMessage, CurrentValue)

**Table 321. Parameters**

| Name | Data Type | Description |
|---|---|---|
| SourceAddress | String | The tag source address. |
| ErrorMessage | String | The error message string encountered, if any. |
| CurrentValue | String | A DataValue object representing the current tag value. |

**Returns**

Boolean. Succeeded/Failed. **Example**

```
Dim OPC1 As iHistorian_SDK.Collector

Dim MyTagValue As Variant

Dim MyErrorMessage As String

' Request The Collector To Get the Current Value for a Tag

Set OPC1 = MyCollectors.Item("USIM031_OPC1")

OPC1.GetCurrentValue "OPCTag1", MyErrorMessage, MyTagValue
```

### GetFilters Method (OPCFilters Object)

Returns the current set of Filters configured for MyCollector.

**Syntax**

object.GetFilters(MyServer, MyCollector)

**Table 322. Parameters**

| Name | Data Type | Description |
|---|---|---|
| MyServer | Variant | The Historian server connection. |
| MyCollector | MyCollector | The Collector object to acquire the Filter information for. |

**Returns**

Boolean true on success, false otherwise

### GetLastError Method (Server Object)

Returns the last error message encountered by the Server object. To see a complete list of messages, refer to the ErrorList property. When possible, the system translates messages into the locale of the client.

**Syntax**

object.**GetLastError**

**Parameters**

None

**Returns**

String. The last error message encountered.

#### GetPrimaryCollectorName Method (Collectors Object)

Returns the name of the Primary Collector for a set of redundantly configured Collectors.

**Syntax**

object.**GetPrimaryCollectorName**(CollectorName)

**Table 323. Parameters**

| Name | Data Type | Description |
|---|---|---|
| CollectorName | String | The name of the Collector. |

**Returns**

String. The primary collector name.

## Method Reference I-L

### I

#### Import Method (Alarms Object)

This function imports the alarms/events in the specified file into this Alarms object.

**CAUTION:**
Any previously imported alarms will be discarded.

**Syntax**

object.Import(FileName, FileFormat)

**Table 324. Parameters**

| Name | Data Type | Description |
|---|---|---|
| FileName | String | Fully qualified import filename (read-only). |
| FileFormat | Long | File format of the import file (read-only). |

**Returns**

Boolean. Returns whether or not the Import operation succeeded.

### Import Method (DataRecordset Object)

Imports the specified file into the current DataRecordset. If the DataRecordset contains items when the Import Method is invoked, the method first clears current DataRecordset before it imports the specified file.

**Syntax**

object.**Import**(FileName, FileFormat)

**Table 325. Parameters**

| Name | Data Type | Description |
|---|---|---|
| FileName | String | Fully qualified import filename (read-only). |
| FileFormat | Long | File format of the import file (read-only). |

**Returns**

Boolean. Returns whether or not the Import operation succeeded.

**Example**

```
' Get A New Recordset

Set MyRecordset = MyData.NewRecordset

' Import The File

If Not MyRecordset.Import("C:\Temp\ImportData.CSV", ihFileFormat.CSV) Then

Err.Raise 1, , "Error Importing File:" + MyRecordset.LastError

End If
```

```
' Commit Data

If Not MyRecordset.WriteRecordset Then

Err.Raise 1, , "Error Committing File: " + MyRecordset.LastError

End If
```

**ImportMethod (EnumeratedSets Object)**

Imports the specified file into the EnumeratedSets collection. The following file formats are supported:

| Name | Description | Value |
|---|---|---|
| CSV | File is imported/exported as comma separated values. | 1 |
| XML | File is imported/exported as XML. | 2 |

Imported files follow a specific format that contains specific keywords. With CSV files, the first row of the file establishes the fields in the file and their positions. With XML reports, the files describe the format of the data.

**Syntax**

object. **Import**(FileName, FileFormat, Server)

**Table 326. Parameters**

| Name | Data Type | Description |
|---|---|---|
| FileName | String | Fully qualified export file name (read-only). |
| FileFormat | ihFileFormat | File format of the export file (read-only). |
| Server | Server | Server from which the sets are imported. |

**Returns**

Boolean. Returns True if the sets have been imported successfully.

**Import Method (MessageRecordset Object)**

Attempts to import a list of Messages from a file.

**Syntax**

object.**Import**(FileName, FileFormat)

**Table 327. Parameters**

| Name | Data Type | Description |
|---|---|---|
| FileFormat | ihFileFormat | The format of the file specified in FileName |
| FileName | String | The name of the file to import. |

**Returns**

Boolean true if the Import succeeded, false otherwise.

### Import Method (TagRecordset Object)

Imports the specified file into the current TagRecordset. If the TagRecordset contains items when the Import method is invoked, the method first clears the current TagRecordset before importing the specified file. After you have imported a file, call the WriteRecordset method of the TagRecordset object to save the data to the Historian server. The following file formats are supported:

| Name | Description | Value |
|---|---|---|
| CSV | File is imported/exported as comma separated values. | 1 |
| XML | File is imported/exported as XML. | 2 |
| Report | File is exported as a columnar report. | 3 |

Imported files follow a specific format that contains specific keywords. With CSV and tabular reports, the first row of the file establishes the fields in the file and their positions. With XML reports, the files describe the format of the data.

**Note:**
Prepare a file for import by exporting it with the desired fields for import.

**Syntax**

object.**Import**(FileName, FileFormat)

**Table 328. Parameters**

| Name | Data Type | Description |
|---|---|---|
| FileName | String | Fully qualified import file name (read-only). |

**Table 328. Parameters (continued)**

| Name | Data Type | Description |
|---|---|---|
| FileFormat | Long | File format of the import file (read-only). |

**Returns**

Returns whether or not the Import operation succeeded.

**Example**

```
' Get a new recordset
Set MyRecordset = MyTags.NewRecordset
' Import the file
If Not MyRecordset.Import(Path & "ImportTags.csv", ihFileFormat.CSV) Then err.Raise 1, , "Error
 importing file: " & MyRecordset.LastError
End If
' Commit Data
If Not MyRecordset.WriteRecordset Then
err.Raise 1, , "Error committing file: " & MyRecordset.LastError
End If
```

## Import Method (UserDefinedType Object)

Applies to:

Imports the specified file into the User Defined Type collection. The following file formats are supported:

| Name | Description | Value |
|---|---|---|
| CSV | File is imported/exported as comma separated values. | 1 |
| XML | File is imported/exported as XML. | 2 |

Imported files follow a specific format that contains specific keywords. With CSV files, the first row of the file establishes the fields in the file and their positions. With XML reports, the files describe the format of the data.

**Syntax**

object. **Import**(FileName, FileFormat, Server)

**Table 329. Parameters**

| Name | Data Type | Description |
|---|---|---|
| FileFormat | ihFileFormat | File format of the import file (read-only). |
| FileName | String | Fully qualified export file name (read-only). |
| Server | Server | Server from which the sets are imported. |

**Returns**

Boolean. Returns TRUE if the type imports successfully.

### InitiateFailover Method (Collectors Object)

Manually tries to initiate a Failover to a redundantly configured collector.

**Syntax**

object.**InitiateFailover**(CollectorName)

**Table 330. Parameters**

| Name | Data Type | Description |
|---|---|---|
| CollectorName | String | Name of the collector to fail-over. |

**Returns**

Collector. Returns Success/Failure.

**Example**

```
Dim MyCollectors As iHistorian_SDK.Collectors

Set MyCollectors = MyServer.Collectors

MyCollectors.InitiateFailover "SimulationCollector"
```

## L

### LastError Method (Alarms Object)

This function returns the last error message generated by this object.

**Syntax**

object.**LastError**

**Parameters**

None

**Returns**

String. The last error message encountered.

### LoadUserCalcLibrary Method (Server Object)

Retrieves the user calculation library from the Server object. This library contains all of the user-created functions and subroutines available for calculations on this Server. The calculation library will be returned as an array of UserCalcFunction objects. If no functions exist in the library, UserCalcFunctions should be set to Empty

**Syntax**

object.**LoadUserCalcLibrary**(UserCalcFunctions)

**Table 331. Parameters**

| Name | Data Type | Description |
|---|---|---|
| UserCalcFunctions | Variant | Resulting array of UserCalcFunction objects. |

**Returns**

Boolean. Returns whether or not the calculation library was loaded successfully.

**Example**

```
Dim MyServer As New iHistorian_SDK.Server

Dim MyUserCalcFunctions() As iHistorian_SDK.UserCalcFunction

Dim MyNewFunction As New iHistorian_SDK.UserCalcFunction

' Connect to the local server

If Not MyServer.Connect("", "", "") Then

err.Raise 1, , "Failed to connect to the local server" End If

' Load the calculation library

MyServer.LoadUserCalcLibrary MyUserCalcFunctions

' Create a new function

MyNewFunction.Name = "Sum"

MyNewFunction.Definition = "Function Sum(a, b)" & vbCrLf & "Sum = a + b" & vbCrLf & "End Function"

' Add it to the loaded library

If IsArray(MyUserCalcFunctions) Then
```

```
ReDim Preserve MyUserCalcFunctions(UBound(MyUserCalcFunctions) + 1) Else
ReDim MyUserCalcFunctions(0) End If
Set MyUserCalcFunctions(UBound(MyUserCalcFunctions)) = MyNewFunction
' Save the changes to the calculation library
If Not MyServer.SaveUserCalcLibrary(MyUserCalcFunctions) Then err.Raise 1, , "Failed to save the
 calculation library"
End If
```

# Method Reference M-P

## M

### ManageServerDialog Method (ServerManager Object)

Displays a window to manage server connection information on the client. This window allows you to add and remove new connections, and to modify the default username and password. This method optionally returns the name of the server last selected in the window.

#### Syntax

object.**ManageServerDialog**([SelectedServer])

**Table 332. Parameters**

| Name | Data Type | Description |
|---|---|---|
| SelectedServer | String | Name of the selected Historian Server (optional, read/write). |

#### Returns

None.

#### Example

```
Dim MyManager As New iHistorian_SDK.ServerManager
Dim MyServer As iHistorian_SDK.Server
Dim SelectedServer As String
' Show the manage server window
MyManager.ManageServerDialog SelectedServer
' If a server was selected, get the server
```

```
If Trim(SelectedServer) <> "" Then

Set MyServer = MyManager.Servers(SelectedServer) End If
```

**N**

**NewRecordset Method (Data Object)**

Returns a new DataRecordset object to subsequently build a query for tag data from the Historian server. It is the responsibility of the developer to release the DataRecordset object when processing has been completed. DataRecordset objects may be re-used by re-executing a query with new criteria set through the DataCriteria of the DataRecordset object.

You must also use a new DataRecordset object to add new data points to the system and delete existing data points from the system. Changes are not committed until calling the WriteRecordset method of the DataRecordset object.

**Syntax**

object.**NewRecordset**

**Parameters**

None

**Returns**

DataRecordset. A new, empty, DataRecordset object.

**NewRecordset Method (Messages Object)**

Returns a new MessageRecordset object to build a query for messages and alerts from the Historian server message archive. You must release the MessageRecordset object when processing completes. You can re-use a MessageRecordset object by re-executing a query with new criteria set through the MessageCriteria of the MessageRecordset object.

To add new messages, you must use a new MessageRecordset object. Call the WriteRecordset method of the MessageRecordset object to commit changes to the archiver.

**Syntax**

object.**NewRecordset**

**Parameters**

None

**Returns**

MessageRecordset. Returns the newly created MessageRecordset object.

### NewRecordset Method (Tags Object)

Returns a new TagRecordset object to build a query for tag information. You must terminate the TagRecordset object when processing completes. You can re-use the TagRecordset objects by re-executing a query with new criteria set through the TagCriteria of the TagRecordset object.

You must also use a new TagRecordset object to add new tags to the system and to delete tags from the system. Call the WriteRecordset Method of the TagRecordset object to commit changes to the archiver.

**Syntax**

object.**NewRecordset**

**Parameters**

None.

**Returns**

TagRecordset. Returns a reference to the newly created Recordset Object.

### Purge Alarms By Id Method

Purges a single alarm as identified by its alarm ID.

**Syntax**

object.**AlarmIds**

**Table 333. Parameters**

| Name | Data Type | Description |
|---|---|---|
| AlarmIds | Long | Alarm ID of the alarm. |

**Returns**

Boolean. Returns TRUE if the alarm is purged.

**Example**

```
Dim Status As ihStatus

            Dim ReturnStatus As Boolean

            Dim Alarms() As Long

            Dim NumberOfAlarms As Long

            Dim i As Long

            Status = ihSTATUS_FAILED ReturnStatus = False

            NumberOfAlarms = (UBound(AlarmIds) + 1)

            ReDim Alarms(0 To NumberOfAlarms - 1) As Long

            On Error GoTo errc

            For i = 0 To NumberOfAlarms - 1

            Alarms(i) = AlarmIds(i)

            Next i

            Status = ihPurgeAlarmsById(MyServer.Handle, Alarms(0), NumberOfAlarms, 0)

            If Status <> ihSTATUS_OK Then err.Raise 1, , "Error Purging alarms by Id[" +
 ErrorDescription(Status) + "," + CS ReturnStatus = True

            PurgeAlarmsById = ReturnStatus errc:

            zLastError = "Purge Alarms By Id>> " + err.Description

            PurgeAlarmsById = ReturnStatus

            End Function
```

### Purge Alarms Method

Purges or deletes the alarms from the Archiver.

#### Syntax

object.**PurgeAlarms**

**Table 334. Parameters**

| Name | Data Type | Description |
|---|---|---|
| BackupFile | String | The file that stores the purged alarm data. |
| ShouldZipAlarms | Boolean | Indicates whether the alarms should be zipped into file or not. |
| StartTime | Date | Start time of the alarms. |
| EndTime | Date | End time of the alarms. |

#### Returns

Boolean. Returns TRUE if the alarms are purged.

**Example**

```
Dim Status As ihStatus
                        Dim AlarmsStartTime As ihTimeStruct Dim AlarmsEndTime As ihTimeStruct Dim
 ShouldZip As ihBoolean
                        Dim ReturnStatus As Boolean
                        Status = ihSTATUS_FAILED ReturnStatus = False
                        AlarmsStartTime = Date_To_UTC(StartTime) AlarmsEndTime = Date_To_UTC(EndTime)
                        If ShouldZipAlarms = True Then
                        ShouldZip = ihTRUE Else
                        ShouldZip = ihFALSE End If
                        On Error GoTo errc
                        Status = ihPurgeAlarms(MyServer.Handle, BackupFile, ShouldZip, AlarmsStartTime,
 AlarmsEndTime, 0)
                        If Status <> ihSTATUS_OK Then err.Raise 1, , "Error Purging alarms [" +
 ErrorDescription(Status) + "," + CStr(St


                        ReturnStatus = True PurgeAlarms = ReturnStatus errc:
                        zLastError = "Purge Alarms>> " + err.Description
                        PurgeAlarms = ReturnStatus
                        End Function
```

# Method Reference Q-T

## Q

### QueryArray Method (Archives Object)

This function returns a list of properties of an Archive found on the Historian server.

**Syntax**

object.**QueryArray**(ArchiveName, Params, SortAscending, ArrayOrientation, ArraySize, ReturnCount, ReturnArray)

**Table 335. Parameters**

| Name | Data Type | Description |
|---|---|---|
| ArchiveName | String | Name of the Archive to return information on. |
| Params | Variant | A list of the parameters to retrieve on the specified Archive. |
| SortAscending | Boolean | Sorting preference for the returned list of properties. |
| ArrayOrientation | Integer | The desired orientation of the returned array. |
| ArraySize | Long | The desired size of the returned array. |
| ReturnCount | Long | The number of rows returned in ReturnArray |
| ReturnArray | Variant | A returned array which contains the requested properties. |

**Returns**

Boolean. Success/Failure.

### QueryArray Method (Collectors Object)

This function returns a list of properties for a set of Collectors.

**Syntax**

object.**QueryArray**(CollectorName, Params, SortAscending, ArrayOrientation, ArraySize, ReturnCount, ReturnArray)

**Table 336. Parameters**

| Name | Data Type | Description |
|---|---|---|
| CollectorName | String | Name of the Collector to return information on. |
| Params | Variant | A list of the parameters to retrieve from the Collectors. |

**Table 336. Parameters (continued)**

| Name | Data Type | Description |
|---|---|---|
| SortAscending | Boolean | Sorting preference for the returned list of properties. |
| ArrayOrientation | Integer | The desired orientation of the returned array. |
| ArraySize | Long | The desired size of the returned array. |
| ReturnCount | Long | The number of rows returned in ReturnArray |
| ReturnArray | Variant | A returned array which contains the requested properties. |

**Returns**

Boolean. Success/Failure.

### QueryArray Method (DataRecordset Object)

This function returns an array of data records from the Historian server.

**Syntax**

object.**QueryArray**(Params, SortAscending, ArrayOrientation, ArraySize, ReturnCount, ReturnArray)

**Table 337. Parameters**

| Name | Data Type | Description |
|---|---|---|
| Params | Variant | A list of the parameters to retrieve. |
| SortAscending | Boolean | Sorting preference for the returned list of properties. |
| ArrayOrientation | Integer | The desired orientation of the returned array. |
| ArraySize | Long | The desired size of the returned array. |
| ReturnCount | Long | The number of rows returned in ReturnArray |
| ReturnArray | Variant | A returned array which contains the requested properties. |

**Returns**

Boolean. Success/Failure.

### QueryArray Method (MessageRecordset Object)

This function returns an array of Messages from the Historian server.

**Syntax**

object.**QueryArray**(Params, SortAscending, ArrayOrientation, ArraySize, ReturnCount, ReturnArray)

**Table 338. Parameters**

| Name | Data Type | Description |
|---|---|---|
| Params | Variant | A list of the parameters to retrieve. |
| SortAscending | Boolean | Sorting preference for the returned list of properties. |
| ArrayOrientation | Integer | The desired orientation of the returned array. |
| ArraySize | Long | The desired size of the returned array. |
| ReturnCount | Long | The number of rows returned in ReturnArray |
| ReturnArray | Variant | A returned array which contains the requested properties. |

**Returns**

Boolean. Success/Failure.

### QueryArray Method (TagRecordset Object)

This function returns an array of Tags from the Historian server.

**Syntax**

object.**QueryArray**(Params, SortAscending, ArrayOrientation, ArraySize, ReturnCount, ReturnArray)

**Table 339. Parameters**

| Name | Data Type | Description |
|---|---|---|
| Params | Variant | A list of the parameters to retrieve. |
| SortAscending | Boolean | Sorting preference for the returned list of properties. |
| ArrayOrientation | Integer | The desired orientation of the returned array. |
| ArraySize | Long | The desired size of the returned array. |
| ReturnCount | Long | The number of rows returned in ReturnArray. |
| ReturnArray | Variant | A returned array which contains the requested properties. |

**Returns**

Boolean. Success/Failure

### QueryRecordset Method (DataRecordset Object)

Executes the DataValue query based on the fields and criteria specified.

**Syntax**

object.**QueryRecordset**

**Parameters**

None

**Returns**

Boolean. Returns whether or not the QueryRecordset operation succeeded.

**Example**

```
Dim I As Integer

Dim J As Integer

Dim K As Integer

Dim strComment$

Dim lngInterval As Long
```

```
Dim TagCount As Integer

Dim strDataQuality As String

Dim iDataRecordset As iHistorian_SDK.DataRecordset

Dim iDataValue As iHistorian_SDK.DataValue

Dim lEndTime&, lStartTime&, lNumSamples&

Dim lNumSeconds, lNumSamplesPerSecond

On Error GoTo Error_Handle

If CheckConnection = True Then

If lbTags.Text = "" Then

MsgBox "No Tag Selected", vbOKOnly, "SDK Sample" Exit Sub

End If

Set iDataRecordset = ConnectedServer.Data.NewRecordset

'reset lstValues.Clear

'build query

With iDataRecordset

'filter code

If txtFilterTag.Text <> "" Then

.Criteria.FilterTagSet = True

.Criteria.FilterTag = txtFilterTag.Text

.Criteria.FilterComparisonModeSet = True

'comparison mode

Select Case cboComparisonMode.Text

Case Is = "Equal"

.Criteria.FilterComparisonMode = 1

Case Is = "NotEqual "

.Criteria.FilterComparisonMode = 2

Case Is = "LessThan"

.Criteria.FilterComparisonMode = 3

Case Is = "GreaterThan"

.Criteria.FilterComparisonMode = 4

Case Is = "LessThanEqual"

.Criteria.FilterComparisonMode = 5

Case Is = "GreaterThanEqual"

.Criteria.FilterComparisonMode = 6

End Select

.Criteria.FilterModeSet = True

'filter mode
```

```
Select Case cboFilterMode.Text

Case Is = "ExactTime"

.Criteria.FilterMode = 1

Case Is = "BeforeTime"

.Criteria.FilterMode = 2

Case Is = "AfterTime"

.Criteria.FilterMode = 3

Case Is = "BeforeAndAfterTime"

.Criteria.FilterMode = 4

End Select

.Criteria.FilterComparisonValue = txtFilterValue.Text

End If

.Criteria.Tagmask = lbTags.Text

.Criteria.StartTime = dtStartTime.Value

.Criteria.EndTime = dtEndTime.Value

'get sample mode

Select Case cboSampleMode.Text

Case Is = "Interpolated"

.Criteria.SamplingMode = 2 'interpolated

Case Is = "Raw By Number"

.Criteria.SamplingMode = 5 'raw by number

Case Is = "Raw By Time"

.Criteria.SamplingMode = 4 'raw by time

Case Is = "Current Value"

.Criteria.SamplingMode = 1 'current value

Case Is = "Calculated"

.Criteria.SamplingMode = 6 'calculation

Case Is = "Trend"

.Criteria.SamplingMode = 3 'trend

End Select

If .Criteria.SamplingMode = 5 Then

'if raw by number get direction

If optDirectionForward.Value = True Then

.Criteria.Direction = 1 'forward

Else

.Criteria.Direction = 2 'backward

End If
```

```
End If

'if calculation get calc mode

If .Criteria.SamplingMode = 6 Then

Select Case cboCalculationMode.Text

Case Is = "Average"

.Criteria.CalculationMode = 1 'average

Case Is = "Standard Deviation"

.Criteria.CalculationMode = 2 'standard deviation

Case Is = "Total"

.Criteria.CalculationMode = 3 'total

Case Is = "Minimum"

.Criteria.CalculationMode = 4 'minimum

Case Is = "Maximum"

.Criteria.CalculationMode = 5 'maximum

Case Is = "Count"

.Criteria.CalculationMode = 6 'count

Case Is = "Raw Average"

.Criteria.CalculationMode = 7 'raw average

Case Is = "Raw Standard Deviation"

.Criteria.CalculationMode = 8 'raw standard deviation

Case Is = "Raw Total"

.Criteria.CalculationMode = 9 'raw total

Case Is = "Minimum Time"

.Criteria.CalculationMode = 10 'minimum time

Case Is = "Maximum Time"

.Criteria.CalculationMode = 11 'maximum time

Case Is = "Time Good"

.Criteria.CalculationMode = 12 'time good

End Select

End If

If optSamplingByNumber.Value = True Then

.Criteria.NumberOfSamples = Int(txtNumSamples.Text) Else

Select Case cboTimeUnits.Text

Case Is = "Milliseconds"

.Criteria.SamplingInterval = Int(txtInterval.Text) Case Is = "Seconds"

.Criteria.SamplingInterval = Int(txtInterval.Text) * 1000

Case Is = "Minutes"
```

```
.Criteria.SamplingInterval = Int(txtInterval.Text) * 60000

Case Is = "Hours"

.Criteria.SamplingInterval = Int(txtInterval.Text) * 3600000

Case Is = "Days"

.Criteria.SamplingInterval = Int(txtInterval.Text) * 86400000

End Select

End If

.Fields.AllFields

VB.Screen.MousePointer = vbHourglass 'wait wait wait lStartTime = Timer

'do query

If Not .QueryRecordset Then lEndTime = Timer

MsgBox "Query Failed..." & Chr(13) & iDataRecordset.LastError

VB.Screen.MousePointer = vbDefault

Exit Sub

End If

lEndTime = Timer VB.Screen.MousePointer = vbDefault lNumSamples = 0

TagCount = iDataRecordset.Item(1).Count

For I = 1 To iDataRecordset.Tags.Count

For J = 1 To iDataRecordset.Item(I).Count

Set iDataValue = iDataRecordset.Item(I).Item(J) Select Case iDataValue.DataQuality

Case Is = 1

strDataQuality = "Good" Case Is = 2

strDataQuality = "Bad" Case Is = 3

strDataQuality = "Unknown"

Case Else

strDataQuality = "ERROR" End Select

strComment = ""

For K = 1 To iDataValue.Comments.Count

strComment = strComment & " " & iDataValue.Comments(K).Comment

Next K

lstValues.AddItem Format(iDataValue.TimeStamp, "MM/dd/yyyy hh:mm:ss") & _

Space(10) & CStr(iDataValue.Value) & vbTab & strDataQuality & vbTab & strComment lNumSamples =

 lNumSamples + 1

Next J

Next I

End With

lNumSeconds = lEndTime - lStartTime lNumSamplesPerSecond = lNumSamples If lNumSeconds > 0 Then
```

```
lNumSamplesPerSecond = lNumSamples / lNumSeconds
End If
txtReadTime.Caption = lNumSamples & " returned in " & lEndTime - lStartTime & " seconds (" &
 lNumSamplesPerSec
Caption = "Output values for " & lbTags.Text & " (" & TagCount & ")" VB.Screen.MousePointer = vbDefault
 'done
Else
MsgBox "Not Connected"
End If
Exit Sub
Error_Handle:
VB.Screen.MousePointer = vbDefault
Select Case Err.Number
Case Is = 6 'overflow
MsgBox "Error Number: " & Err.Number & Chr(13) & "Description: " & Err.Description & Chr(13) & "Check
 numb
Case Is = 13 'type mismatch
MsgBox "Error Number: " & Err.Number & Chr(13) & "Description: " & Err.Description & Chr(13) & "Check
 numb
Case Is = 91
MsgBox "Error Number: " & Err.Number & Chr(13) & "Check connection to server"
Case Else
MsgBox "Error Number: " & Err.Number & Chr(13) & Err.Description, vbOKOnly, "Error"
End Select
```

## QueryRecordset Method (MessageRecordset Object)

Executes the message query based on the fields and criteria specified.

### Syntax

object.**QueryRecordset**

### Parameters

None

### Returns

Boolean. Success/Failure.

### Example

```
Dim MyRecordset As iHistorian_SDK.MessageRecordset

Dim I As Integer

Dim J As Integer

Dim lEndTime&, lStartTime&, lNumMessages& Dim lNumSeconds&, lNumMessagesPerSecond&

If CheckConnection = True Then

' Get A New Recordset

Set MyRecordset = ConnectedServer.Messages.NewRecordset

' Return Timestamp, Message Number, and Message String Fields In Query

With MyRecordset.Fields

.TimeStamp = True

.MessageNumber = True

.MessageString = True

End With

' Query messages for given start and end time

With MyRecordset.Criteria

.Clear

.StartTime = dtMessageStart.Value

.EndTime = dtMessageEnd.Value

If txtMessageContains.Text <> "" Then

.MessageString = txtMessageContains.Text

End If

End With

VB.Screen.MousePointer = vbHourglass ' wait wait wait

' Run Query lStartTime = Timer

If Not MyRecordset.QueryRecordset Then lEndTime = Timer

MsgBox "Query Failed..." & Chr(13) & MyRecordset.LastError

VB.Screen.MousePointer = vbDefault

Exit Sub

End If

lEndTime = Timer

' reset output list lstRetrievedMessages.Clear

' display all messages lNumMessages = 0

For J = 1 To MyRecordset.Item.Count

lstRetrievedMessages.AddItem Format(MyRecordset.Item(J).TimeStamp, "MM/dd/yyyy hh:mm:ss") & _

vbTab & MyRecordset.Item(J).MessageString lNumMessages = lNumMessages + 1

Next J
```

```
' calculate performance statistics lNumSeconds = lEndTime - lStartTime lNumMessagesPerSecond =
 lNumMessages If lNumSeconds > 0 Then
lNumMessagesPerSecond = lNumMessages / lNumSeconds
End If
VB.Screen.MousePointer = vbDefault
txtReadTime.Caption = lNumMessages & " returned in " & lEndTime - lStartTime & " seconds (" &
 lNumMessagesP
End If
```

## QueryRecordset Method (TagRecordset Object)

Executes the tag query based on the fields and criteria specified.

### Syntax

object.**QueryRecordset**

### Parameters

None.

### Returns

Boolean. Returns whether the QueryRecordset operation succeeded.

### Example

```
Dim MyTags As iHistorian_SDK.Tags
Dim MyRecordset As iHistorian_SDK.TagRecordset
Dim lStartTime&, lEndTime&, lNumSeconds&, lNumTagsPerSecond& Dim I As Integer
On Error GoTo errc lbTags.Clear
' If we are connected to server
If CheckConnection = True Then
' Query all the tagnames
Set MyTags = ConnectedServer.Tags
Set MyRecordset = MyTags.NewRecordset
MyRecordset.Criteria.Tagname = txtTagMask.Text
If txtDescriptionMask.Text <> "" Then
MyRecordset.Criteria.Description = txtDescriptionMask.Text
End If MyRecordset.Fields.Clear
MyRecordset.Fields.Tagname = True
VB.Screen.MousePointer = vbHourglass
lStartTime = Timer
```

```
If Not MyRecordset.QueryRecordset Then Err.Raise 1, , "Tag Query Failed: " + MyRecordset.LastError

lEndTime = Timer

For I = 1 To MyRecordset.Item.Count

lbTags.AddItem MyRecordset.Item(I).Tagname

Next I

VB.Screen.MousePointer = vbDefault

' Calculate performance statistics

lNumSeconds = lEndTime - lStartTime

lNumTagsPerSecond = MyRecordset.Item.Count

If lNumSeconds > 0 Then

lNumTagsPerSecond = MyRecordset.Item.Count / lNumSeconds

End If

lblAddTime.Caption = MyRecordset.Item.Count & " tags returned in " & lEndTime - lStartTime & " seconds

 (" & lN MyRecordset.ClearRecordset

Set MyRecordset = Nothing

Else

MsgBox "Not connected"

End If

Exit Sub errc:

MsgBox "TestBrowseTags >> " + Err.Description
```

### QueryTagAlias Method (TagRecordset Object )

This function returns a list of tag aliases for the tag name passed in. Pass in the current name of the tag and this function will return if there are any previous names for the tag due to tag rename.

Syntax object.**QueryTagAlias**(TagNames(), NumberofTags, TagAlias(), NoOfAliases)

**Table 340. Parameters**

| Name | Data Type | Description |
|---|---|---|
| TagNames() | String | Name of the tags to return information on. |
| NumberofTags | Long | Total count of tag names passed in. This must be equal to 1. |
| TagAlias() | String | List of 0 or more tag aliases. |
| NoOfAliases | Integer | The number of tag aliases returned. |

**Returns**

Boolean. TRUE if successful. FALSE otherwise.

### QueryUserDefinedType Method (UserDefinedType Object)

Queries the Historian Server and loads all the User Defined Types that match the query mask.

**Syntax**

object.**QueryUserDefinedType**(Server, QueryMask)

**Table 341. Parameters**

| Name | Data Type | Description |
|---|---|---|
| Server | Server | The reference to the Historian Server object. |
| QueryMask | String | A mask string that can be used to search for UserDefinedType on the Historian Server. The string can include wild-card characters like "*" and "?". |

**Returns**

Boolean. Indicates whether the query was successful.

## R

### Reload Method (Collector Object)

Causes a re-calculation of tags to occur for a specified tag period. You can reload all tags for the time period or specify specific tags that you want to reload.

**Note:**
This method is only supported by ServerToServer and Calculation collectors.

**Syntax**

object.**Reload**(StartTime, EndTime, [Tags])

**Table 342. Parameters**

| Name | Data Type | Description |
|---|---|---|
| StartTime | Date | The time that you want the reload to begin at (read-only). |
| EndTime | Date | The time that you want the reload to end at (read-only). |
| Tags | Variant | The specified tag names either in an array of tag names, or in a tagRecordset object for tags that you want to reload (optional). |

**Returns**

Boolean. Returns whether or not the Reload operation succeeded.

### Remove Method (TagDependencies Object)

Removes a Tag dependency from the current calculation.

> **CAUTION:**
> If no Tagname is specified, **all** the Tag dependencies will be removed.

**Syntax**

object.**Remove**([Tagname])

**Table 343. Parameters**

| Name | Data Type | Description |
|---|---|---|
| Tagname | Variant | Name of the tag to remove (optional), default = ""). |

**Returns**

Boolean. Whether the Remove operation succeeded.

### RemoveServer Method (ServerManager Object)

Removes the specified server from the list of registered servers on the client.

**Syntax**

object.**RemoveServer**(ServerName)

**Table 344. Parameters**

| Name | Data Type | Description |
|---|---|---|
| ServerName | String | Computer name of the Historian server (read/write). |

**Returns**

Boolean. Whether the server was successfully removed from the list.

### Rename Method (Tags Object)

Use this method to rename tag names. You can rename a tag permanently by passing an additional parameter as TRUE or FALSE.

**Note:**
This method is called within the WriteRecordset method. For more information, refer to the Sample Code section.

**Syntax**

object.**Rename**(newTagName, (Optional) RenamePermanent)

**Table 345. Parameters**

| Name | Data Type | Description |
|---|---|---|
| newTagName | String | Name of the new tag to rename. |
| RenamePermanent (Optional) | Boolean | Pass TRUE to permanently rename a tag. |

**Returns**

Boolean. Returns whether or not the Rename operation succeeded.

### Restore Alarms Method

Restores the alarms.

**Syntax**

object.**RestoreAlarms**

**Table 346. Parameters**

| Name | Data Type | Description |
|---|---|---|
| RestoreFileName | String | The name of the file to be restored to the absolute path. |

**Returns**

Boolean. Returns TRUE if the alarms have been restored.

**Example**

```
Dim Status As ihStatus

Dim ReturnStatus As Boolean Status = ihSTATUS_FAILED ReturnStatus = False

On Error GoTo errc

Status = ihRestoreAlarms(MyServer.Handle, RestoreFileName, 0)

If Status <> ihSTATUS_OK Then err.Raise 1, , "Error in restoring the alarms [" +

 ErrorDescription(Status) + "," ReturnStatus = True

RestoreAlarms = ReturnStatus errc:

zLastError = "Purge Alarms>> " + err.Description

RestoreAlarms = ReturnStatus

End Function
```

## S

### SaveSet Method (EnumeratedSets Object)

Saves the Enumerated Set that has been passed into the Historian Server.

Syntax

object.**SaveSet**(SetTosave)

**Table 347. Parameters**

| Name | Data Type | Description |
|---|---|---|
| SetToSave | EnumeratedSet | The set that is passed in to be saved. |

**Returns**

None.

### SaveSet Method (UserDefinedType Object)

Saves the UserDefinedType that has been passed into the Historian Server.

**Syntax**

object.**SaveSet**(Handle, MySet, Datatype)

**Table 348. Parameters**

| Name | Data Type | Description |
|---|---|---|
| Handle | Long | Server handle. |
| MySet | UserDefinedType | The UserDefinedType to be saved. |
| DataType | ihDatatype | The data type of the UserDefinedType |

**Returns**

Boolean. Returns TRUE if the UserDefinedType is saved.

### SaveToCollectorProperty Method (OPCFilters Object)

Saves the current filter configuration in the Collector object.

**Syntax**

object.**SaveToCollectorProperty**(Collector)

**Table 349. Parameters**

| Name | Data Type | Description |
|---|---|---|
| Collector | Variant | The Collector object in which to save the filtering information. |

**Returns**

Boolean true on success, false otherwise.

### SaveUserCalcLibrary Method (Server Object)

Saves the given user calculation library to the Server object. This will update the set of user-created functions and subroutines available for calculations on this Server. The new calculation library must be passed in as an array of UserCalcFunctions.

This list of functions will replace any existing user calculation library on the Server. To save an empty calculation library, pass a non-array value such as Empty.

#### Syntax

object.**SaveUserCalcLibrary**(UserCalcFunctions)

**Table 350. Parameters**

| Name | Data Type | Description |
|---|---|---|
| UserCalcFunctions | Variant | Array of UserCalcFunction objects. |

#### Returns

Boolean. Returns whether or not the calculation library was saved successfully.

#### Example

```
Dim MyServer As New iHistorian_SDK.Server

Dim MyUserCalcFunctions() As iHistorian_SDK.UserCalcFunction

Dim MyNewFunction As New iHistorian_SDK.UserCalcFunction

' Connect to the local server

If Not MyServer.Connect("", "", "") Then

err.Raise 1, , "Failed to connect to the local server" End If

' Load the calculation library

MyServer.LoadUserCalcLibrary MyUserCalcFunctions

' Create a new function

MyNewFunction.Name = "Sum"

MyNewFunction.Definition = "Function Sum(a, b)" & vbCrLf & "Sum = a + b" & vbCrLf & "End Function"

' Add it to the loaded library

If IsArray(MyUserCalcFunctions) Then

ReDim Preserve MyUserCalcFunctions(UBound(MyUserCalcFunctions) + 1) Else

ReDim MyUserCalcFunctions(0) End If

Set MyUserCalcFunctions(UBound(MyUserCalcFunctions)) = MyNewFunction

' Save the changes to the calculation library

If Not MyServer.SaveUserCalcLibrary(MyUserCalcFunctions) Then err.Raise 1, , "Failed to save the

 calculation library"

End If
```

### SelectAll Method (TagRecordset Object)

Selects each tag in the current TagRecordset. Use in conjunction with the Master tag to perform bulk update operations on the selected tags of the TagRecordset.

**Syntax**

object.**SelectAll**

**Parameters**

None.

**Returns**

None.

**Example**

```
' Query for tags
Set MyRecordset = MyServer.Tags.NewRecordset
MyRecordset.QueryRecordset
' Select all tags
MyRecordset.SelectAll
' Update the HiEngineeringUnits for all tags
MyRecordset.Master.HiEngineeringUnits = 300
' Commit changes
MyRecordset.WriteRecordset
```

### SetFields Method (DataRecordset Object)

Sets the DataValue fields to return from the Historian server when a DataRecordset query is executed.

**Syntax**

object.**SetFields**(Params)

**Table 351. Parameters**

| Name | Data Type | Description |
|---|---|---|
| Params | Variant | Array of DataValue fields to set. |

**Returns**

Boolean. Success / Failure

### SetFields Method (MessageRecordset Object)

Set a list of desired Fields to be returned for the messages.

**Syntax**

object.**SetFields**(Params)

**Table 352. Parameters**

| Name | Data Type | Description |
|---|---|---|
| Params | Variant | Array of Message fields |

**Returns**

Boolean. Success / Failure

### SetFields Method (TagRecordset Object)

Sets the Fields to retrieve in the TagRecordSet.

**Syntax**

object.**SetFields**(Params)

**Table 353. Parameters**

| Name | Data Type | Description |
|---|---|---|
| Params | Variant | The array of field values to set. |

**Returns**

Boolean. Success / failure.

### SetNames Method (EnumeratedSets Object)

This function returns an array of names of all the loaded Enumerated Sets within the Historian server object.

**Syntax**

object.**SetNames**()

**Parameters**

None.

**Returns**

String Array.

### ShowErrorListDialog Method (Server Object)

Displays a window that details errors messages accumulated during the current session of the Server object. Each message is timestamped at the time the error generated and includes an error message translated into the locale of the client when possible.

**Syntax**

object.**ShowErrorListDialog**

**Parameters**

None.

**Returns**

None

**Example**

```
ErrorTrap:

' On error display the error list window

If MyServer.ErrorList.count > 0 Then

MyServer.ShowErrorListDialog

End If
```

### SubscribeAlerts Method (Messages Object)

Subscribes to alert messages reported by the Historian Server. As the server receives alerts, the Historian server publishes messages to any client signed up for alerts.

- To subscribe to messages of specific types, supply a topic.
- To subscribe to all topics, do not supply a topic. Subscribe all is the default.
- Subscribe to individual topics by making multiple calls to the SubscribeAlerts with different topics.
- To subscribe to all topics, call SubscribeAlerts and pass 0.
- To **unsubscribe** to a specific topic or all topics, call SubscribeAlerts and supply the Subscribe parameter set to False.

The Alert_Received event of the Messages object reports alert messages to the client asynchronously.

**Syntax**

object.**SubscribeAlerts**(Topic, Subscribe)

**Table 354. Parameters**

| Name | Data Type | Description |
|---|---|---|
| Topic | ihMessageTopic | Topics of Alerts to subscribe to (optional, default = All). |
| Subscribe | Boolean | Flag to subscribe / unsubscribe to alerts (optional, default = True). Set to False to un-subscribe. |

**Returns**

Boolean. Returns whether or not the SubscribeAlerts operation succeeded.

### SubscribeChanges Method (Tags Object)

Subscribes to changes in tag configuration. The Historian server publishes messages to any client signed up for tag configuration changes as modifications are saved to the tag database.

The system reports tag configuration messages asynchronously to the client through the ChangeReceived event of the Tags Object.

To **unsubscribe** to a specific tag, or all tags, call SubscribeChanges and supply the Subscribe parameter set to False.

**Syntax**

object.**SubscribeChanges**(Tagname, Subscribe)

**Table 355. Parameters**

| Name | Data Type | Description |
|---|---|---|
| Tagname | String | Name of tag to Subscribe for configuration changes (read-only). |
| Subscribe | Boolean | Flag to Subscribe/Unsubscribe to tag changes (default = True). |

**Returns**

Boolean. Returns whether or not the SubscribeChanges operation succeeded.

### SubscribeData Method (Data Object)

Subscribes to changes in the current value of a specific tag. The Historian server publishes messages as new values are received to any client signed up for current value changes. To qualify as a new current value, any new data point must have a newer timestamp than the previously established current value. Values received by the Historian server have passed a deadband check by the collector reporting the data. Since new current values have not been compressed, however, values reported as current may not exactly match those that reach the archive.

You can use the MinimumElapsedTime, in milliseconds, to throttle the rate at which current values are reported to a specific server connection. If the same tag is subscribed twice, the last supplied MinimumElapsedTime is used. If MinimumElapsedTime is not supplied, or is zero, the system reports all current values.

The system reports current values asynchronously to the client through the DataReceived event of the Data object.

To unsubscribe to a specific tag, or all tags, call SubscribeData and set the Subscribe parameter to False.

#### Syntax

object.**SubscribeData**(Tagname, MinimumElapsedTime, Subscribe)

**Table 356. Parameters**

| Name | Data Type | Description |
|---|---|---|
| Tagname | String | Name of the tag to Subscribe for current values (read-only). |
| MinimumElapsed-Time | MinimumElapsed-Time | Minimum elapsed time (ms) between values (read-only). |
| Subscribe | Boolean | Flag to subscribe/unsubscribe to current values (read-only). |

#### Returns

Boolean. Operation success / fail

### SubscribeMessages Method (Messages Object)

Subscribes to messages reported by the Historian Server. The Historian Server publishes messages as messages are received to any client signed up for messages. As an option, you may supply a topic to subscribe only to messages of specific types. If you do not supply a topic, all topics are subscribed. You can subscribe to individual topics by making multiple calls to the SubscribeMessages method with different topics, or pass 0 to subscribe to all topics.

The system reports alert messages asynchronously to the client through the Message_Received event of the Messages object.

To unsubscribe to a specific topic, or all topics, call SubscribeMessages and set the Subscribe parameter to False.

**Syntax**

object.**SubscribeMessages**(Topic, Subscribe)

**Table 357. Parameters**

| Name | Data Type | Description |
|---|---|---|
| Topic | ihMessageTopic | Topics of messages to Subscribe to (optional, default = True). |
| Subscribe | Boolean | Flag to unsubscribe to a specific topic (optional, default = True). |

**Returns**

Boolean. Operation success / fail

### Substitutions Method (MessageFields Object)

Determines whether the Substitutions should be returned in the MessageRecordset query.

**Syntax**

object.**Substitutions**

**Parameters**

None

**Returns**

None

**Example**

```
Dim maMyMessages As iHistorian_SDK.MessageRecordset

Set MyMessages = GetServer.Messages.NewRecordset With MyMessages.Fields

.Topic = True

.TimeStamp = True

.MessageString = True

.Substitutions = True

End With
```

**T**

**TestCalculation Method (Tag Object)**

Runs the calculation currently stored in the Calculation property. This calculation will be run with a current time of "Now". The results of the calculation will be stored in the Value and DataQuality parameters.

If an error occurs during the test, a description of the error will be stored in the ErrorMessage parameter.

**Syntax**

object.**TestCalculation**(Value, DataQuality, ErrorMessage)

**Table 358. Parameters**

| Name | Data Type | Description |
|---|---|---|
| Value | Variant | Value of the calculation result. |
| DataQuality | String | Quality of the calculation result. |
| ErrorMessage | String | Description of any error that occurred during the test. |

**Returns**

Boolean. Operation success / fail

**TranslateMessage Method (Server Object)**

Returns a translated message or prompt based on the current locale and the MessageNumber specified. If no translation is available for the current locale, the method uses the default message.

To insert context specific information into the generic message string, use substitutions. For example, the generic message string:

```
Connection To Server:[1]

Failed With Error Number: [2]
```

requires two substitutions, the first being the server, and the second being the error number.

The substituted message then reads:

```
Connection To Server: USGB014

Failed With Error Number: 65535.
```

**Syntax**

object.**TranslateMessage**(MessageNumber, DefaultMessage, Substitutions)

**Table 359. Parameters**

| Name | Data Type | Description |
|---|---|---|
| MessageNumber | Long | Message or prompt number to translate (read-only). |
| DefaultMessage | String | Default message to translate (optional, read-only). |
| Substitutions | ParamArray | Ordered substitutions into message (optional, read-only). |

**Returns**

String. The translated message text.

**Example**

```
Dim MyServer As New iHistorian_SDK.Server

Dim MyPrompt As String

' Connect to the default server

If Not MyServer.Connect Then

err.Raise 1, , "Failed to authenticate on server " + MyServer.ServerName

End If
```

```
' Translate the prompt from the connected server

MyPrompt = MyServer.TranslateMessage(549, "User: [1]", MyServer.Username)

' Prompt is translated to "User: Fred"
```

# Method Reference U-Z

## U

### UnSelectAll Method (TagRecordset Object)

Applies to:

Clears all selections in the current TagRecordset. See the SelectAll method and the Master Tag Property.

#### Syntax

object.**UnSelectAll**

#### Parameters

None

#### Returns

None

#### Example

```
' Clear any current selection MyRecordset.UnSelectAll
```

### UserCalcFunctionsFromString Method (Server Object)

Applies to:

Converts the given string to an array of user calculation functions. The string is assumed to have been generated from a call to UserCalcFunctionsToString.

If no functions exist in the string, UserCalcFunctions will be set to Empty.

#### Syntax

object.**UserCalcFunctionsFromString**(FuncStr, UserCalcFunctions)

**Table 360. Parameters**

| Name | Data Type | Description |
|---|---|---|
| FuncStr | String | String to convert. |
| UserCalcFunctions | Variant | Resulting array of UserCalcFunction objects. |

**Returns**

Boolean. Conversion successful / failed.

### UserCalcFunctionsToString Method

Applies to:

Converts the given array of user calculation functions to a string.

To convert an empty set of functions, pass a non-array value such as Empty.

**Syntax**

object.**UserCalcFunctionsToString**(UserCalcFunctions, FuncStr)

**Table 361. Parameters**

| Name | Data Type | Description |
|---|---|---|
| UserCalcFunctions | Variant | Array of UserCalcFunction objects. |
| FuncStr | String | String resulting from conversion |

**Returns**

Boolean. Conversion successful / failed.

## W

### WriteArray Method (DataRecordset Object)

Applies to:

Attempt to write a set of Data to the Historian archiver.

> **Note:**
> This function is not fully implemented yet and will always return `false current DataRecordset` before it imports the specified file.

**Syntax**

object.**WriteArray**(DataArray)

**Table 362. Parameters**

| Name | Data Type | Description |
|---|---|---|
| DataArray | Variant | Data Array to be written to the Historian archiver. |

**Returns**

Boolean. Write succeeded / failed.

### WriteRecordset Method (DataRecordset Object)

Applies to:

Saves changes made to the DataRecordset object to the Historian server. If DataValues have not changed, the method does not write DataValues to the server.

**Syntax**

object.**WriteRecordset**

**Parameters**

None

**Returns**

Boolean. Write succeeded / failed.

### WriteRecordset Method (MessageRecordset Object)

Applies to:

Saves changes made to the MessageRecordset object to the Historian server. Only new messages are written to the server.

**Syntax**

object.**WriteRecordset**

**Parameters**

None

**Returns**

Boolean. Write succeeded / failed.

### WriteRecordset Method (TagRecordset Object)

Applies to:

Saves changes made to the TagRecordset Object to the Historian server. If tags have not changed, they are not re-written to the server.

**Syntax**

object.**WriteRecordset**([UseMasterTag])

**Table 363. Parameters**

| Name | Data Type | Description |
|---|---|---|
| UseMasterTag | Boolean | Whether to apply Master tag changes to all tags (read-only optional). |

**Returns**

Boolean. Write succeeded / failed.

## X

### XML Method (DataRecordset Object)

Applies to:

Returns an XML document fragment representing the DataValues and DataFields contained in the current DataRecordset.

**Syntax**

object.**XML**([XMLHeader], [StartIndex], [EndIndex])

**Table 364. Parameters**

| Name | Data Type | Description |
|---|---|---|
| XMLHeader | String | XML to include before the DataRecordset XML (optional). |
| StartIndex | Long | Index of first tag to include in XML (optional). |
| EndIndex | Long | Index of last tag to include in XML (optional). |

**Returns**

String. An XML document fragment string.

**Example**

```
Dim Recordset As DataRecordset

' Get A New Data Recordset

Set Recordset = MyServer.Data.NewRecordset

' Fill In Criteria, Get One Tag For Yesterday

With Recordset.Criteria

  .Tagmask = "MyNode.OneTag.F_CV"

  .StartTime = DateAdd("d", -1, Now)

  .EndTime = Now

End With

' Fill In Fields, Timestamp and Value

With Recordset.Fields

  .TimeStamp = True

  .Value = True End With

Recordset.QueryRecordset

' Print XML to Debug Window

Debug.Print Recordset.XML
```

## XML Method (MessageRecordset Object)

Applies to:

Returns an XML document fragment representing the Messages and MessageFields contained in the current MessageRecordset.

**Syntax**

object.**XML**([XMLHeader], [StartIndex], [EndIndex])

**Table 365. Parameters**

| Name | Data Type | Description |
|---|---|---|
| XMLHeader | String | XML to include before message XML (read-only, optional). |
| StartIndex | Long | Index of first message to Include in XML (read-only, optional). |
| EndIndex | Long | Index of last message to include in XML (read-only, optional). |

**Returns**

String. An XML document fragment string.

### XML Method (TagRecordset Object)

Applies to:

Returns an XML document fragment representing the Tags and TagFields contained in the current TagRecordset.

**Syntax**

object.**XML**([XMLHeader], [StartIndex], [EndIndex])

**Table 366. Parameters**

| Name | Data Type | Description |
|---|---|---|
| XMLHeader | String | XML to include before the TagRecordset XML (read-only, optional). |
| StartIndex | Long | Index of first tag to Include in XML (read-only, optional). |
| EndIndex | Long | Index of last tag to include in XML (read-only, optional). |

**Example**

```
Dim Recordset As TagRecordset

' Get a new tag recordset

Set Recordset = MyServer.Tags.NewRecordset

' Fill in criteria, aet all tags

With Recordset.Criteria

.Tagname = "*"

End With

' Fill in fields, get tagname and description

With Recordset.Fields

  .Tagname = True

  .Description = True

  End With

Recordset.QueryRecordset

' Print XML to debug window

Debug.Print Recordset.XML
```

# Event Reference A-Z

## A

### AlertReceived Event (Messages Object)

The AlertReceived event fires each time an alert is reported to a client. This event fires only for those alert topics that you subscribed to by using the SubscribeAlerts method of the Messages object.

#### Syntax

**AlertReceived**(NewAlert)

**Table 367. Parameters**

| Name | Data Type | Description |
|---|---|---|
| NewAlert | Message | The Alert reported by the Historian server (read-only). |

#### Example

```
' It is necessary to declare Messages "WithEvents" Dim WithEvents MyMessages As iHistorian_SDK.Messages

 Attribute MyMessages.VB_VarHelpID = -1
```

```
Private Sub Form_Load()
' Subscribe To All Alert Topics
If Not MyMessages.SubscribeAlerts(ihMessageTopic.Security, True) Then
Err.Raise 1, , "Failed To Subscribe" End If
End Sub
' Event Procedure
Private Sub MyMessages_Alert_Received(NewMessage As Message)
' We Just Received A New Alert
With NewMessage
Debug.Print "Received Alert " + CStr(.MessageNumber) + _ " " + .MessageString
End With
End Sub
```

## C

### ChangeReceived Event (Tags Object)

Fires each time a tag configuration change is reported to a client. Subscribe to data changes using the current data received event. This event fires only for tags that you subscribed to by using the SubscribeChanges method of the Tags object. You can subscribe to tag changes on a per tag basis or you can pass "" to subscribe to changes on any tag.

#### Syntax

**ChangeReceived**(ChangedTag)

**Table 368. Parameters**

| Name | Data Type | Description |
|---|---|---|
| ChangedTag | Tag | Tag configuration change reported by the server (read-only). |

## D

### DataReceived Event (Data Object)

Fires each time a changed value is reported to a client. This event fires only for tags that you subscribed to by using the SubscribeData method of the Data object.

#### Syntax

**DataReceived**(Tagname, Value)

**Table 369. Parameters**

| Name | Data Type | Column Header |
|---|---|---|
| Tagname | String | Tag with the current value change reported by the server (read-only). |
| Value | DataValue | Value reported by the server (read-only). |

**Example**

```
' It is necessary to declare Data "WithEvents" Dim WithEvents MyData As iHistorian_SDK.Data Attribute

 MyData.VB_VarHelpID = -1

Private Sub Form_Load()

' Subscribe To Changes

If Not MyData.SubscribeData("USGB014.FIC101.F_CV", 5000, True) Then

Err.Raise 1, , "Failed To Subscribe" End If

End Sub

' Event Procedure

Private Sub MyData_DataReceived(Tagname As String, DataValue As DataValue)

' We Just Received a Current Value Update

With DataValue

Debug.Print "Received Current Value For " + Tagname + " " + _

" At " + Format$(.TimeStamp) + " With Value of " + _ CStr(.Value)

End With

End Sub
```

**M**

### MessageReceived Event (Messages Object)

) Fires each time a message is reported to a client. This event fires only for message topics that you subscribed to by using the SubscribeMessages method of the Messages object.

**Syntax**

**MessageReceived**(NewMessage)

**Table 370. Parameters**

| Name | Data Type | Description |
|---|---|---|
| NewMessage | Message | Message reported by the Historian Server (read-only). |

## S

### StatusReceived Event (Archives Object)

Fires each time a tag configuration change is reported to a client. This event fires only if the SubscribeStatus property of the Archives object is set to true.

#### Syntax

**StatusReceived**(ChangedArchive)

**Table 371. Parameters**

| Name | Data Type | Description |
|---|---|---|
| ChangedArchive | Archive | Reported change for Historian server archive (read-only). |

#### Example

```
' It is necessary to declare Archives "WithEvents" Dim WithEvents MyArchives As iHistorian_SDK.Archives

 Attribute MyArchives.VB_VarHelpID = -1

Private Sub Form_Load()

' Subscribe To Changes

MyArchives.SubscribeStatus = True

End Sub

' Event Procedure

Private Sub MyArchives_StatusReceived(ChangedArchive As Archive)

' We Just Received An Archive Status Update

With ChangedArchive

Debug.Print "Received Update For " + .Name + " (" + .FileName + ")" End With

End Sub
```

### StatusReceived Event (Collectors Object)

Fires each time a Collectors' status or configuration change is reported to a client. This event fires only if the SubscribeStatus property of the Collectors object is set to True.

#### Syntax

**StatusReceived**(ChangedCollector)

**Table 372. Parameters**

| Name | Data Type | Description |
|---|---|---|
| ChangedCollector | Collector | Reported change for collector (read-only). |

**Example**

```
' It is necessary to declare Collectors "WithEvents"

Dim WithEvents MyCollectors As iHistorian_SDK.Collectors

Attribute MyCollectors.VB_VarHelpID = -1

Private Sub Form_Load()

' Subscribe To Changes

MyCollectors.SubscribeStatus = True

End Sub
```

**T**

### TagnameChangeReceived Event (Tags Object)

Occurs each time a tag is renamed. This event occurs only for tags that you subscribed to by using the SubscribeChanges method of the Tags object. You can subscribe to tag changes on a per tag basis or you can pass "" to subscribe to changes on any tag.

**Syntax**

**TagnameChangeReceived**(ChangedTag, oldTag)

**Table 373. Parameters**

| Name | Data Type | Description |
|---|---|---|
| ChangedTag | Tag | Tag rename change reported by the server (read-only). |
| oldTag | String | Old tag name reported by the server (read-only). |

# Chapter 12. Historian Utility Suite

## Alerting System

The Alerting System is a reliable and efficient solution designed to promptly notify users via email when a particular service becomes unavailable or experiences an unexpected interruption. This system aims to minimize downtime and ensure timely response to service disruptions, enabling users to take appropriate actions to mitigate the impact on their operations. By providing real-time alerts, it empowers users to stay informed about service interruptions and promptly address any issues.

Key features of this utility include:

- **Service Monitoring**: The alerting system continuously monitors the availability and status of the targeted service, whether it's a custom service or an application service like Historian Data Archiver(x64), or any other critical service.
- **Customizable Alert Conditions**: Alerting system is driven by a config file, which give the flexibility to use by any product. Users can define for alerts for specific services under which they want to receive alert emails. This system accommodates two categories of the services, in case of Historian, user can specify IH and Collectors, since the system do a search a 'contains' condition of the given string. It will monitor all the
- **Real-Time Email Notifications**: When the system detects a service stop or meets the predefined alert conditions, it immediately triggers an email notification to the designated user. The email includes relevant service name in the subject and the body.
- **User Management**: The alerting system allows users to manage their notification preferences and frequency settings in milli seconds to monitor services. They can change recipients to receive alerts, add or remove email addresses.
- **Dump Creation & Uploading to FTP**: A configurable attribute will allow the alerting system to create a dump when a service stops. uploads the mini dumps to preconfigured ftp sites, which reduces the manual errors during the setting to create dumps and ftp upload. Let's assume a user starts the alerting system, and after a few hours, a service get down, then the system automatically creates mini dumps and automatically uploads to the ftp mentioned for analysis.

**Note:**
The ServiceRunCheck.exe utility should be run with Administrator privileges. The ServiceRunCheck.exe.config, also provided with this utility, can be edited in any text editor (such as Notepad). In this file you can update details like user names, passwords, smtp clients, and so on.

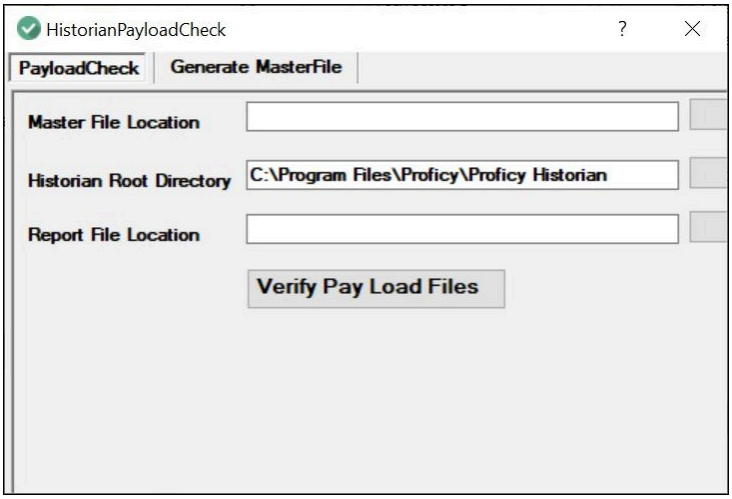
For more information, see article ID 000068757 on the GE Vernova Support center: https://digitalsupport.ge.com/s/article/Historian-Alerting-System-Utility?language=en_US

# Version Validator

The Version Validator utility reports if all the .exes and .dlls from a given folder are of correct version with the correct last modified date. This Utility aids the user to identify any mismatches of binaries.

**Note:**
The HistorianPayLoadCheck.exe utility should be run with Administrator privileges, in order to generate master files.

This utility has two tabs: **PayLoadCheck** and **Generate Masterfile** as shown in the following figure:




For more information on how to use, see article ID 000068787 on the GE Vernova Support center: https://digitalsupport.ge.com/s/article/Historian-Version-Validator-Utility?language=en_US

# DiagnostiX Tool

The DiagnostiX is a powerful software solution designed to streamline the support and development process by providing a comprehensive understanding of the customer environment. This tool helps Support teams and Developers tackle customer issues, enabling them to self-solve obvious problems and significantly reducing the time and effort spent on resolution.

Key features and benefits include:

- **Environment Snapshot**: This tool offers a holistic view of the customer environment, capturing essential details such as operating system, hardware specifications, software versions, network configuration, and other relevant parameters. This comprehensive snapshot eliminates the need for back-and-forth communication to gather information, saving valuable time for both support teams and developers.

- **Self-Solve Capabilities**: This tool goes beyond just identifying issues; it also provides built-in solutions for common problems. Through a knowledge base of known issues and their resolutions, the tool offers step-by-step guidance to support teams and developers, enabling them to self-solve straightforward issues efficiently. This not only saves time but also empowers the teams to handle a broader range of customer concerns independently.
- **Enhanced Collaboration**: This tool acts as a collaborative platform, facilitating seamless communication between support teams and developers. It provides a centralized repository for sharing diagnostic reports, annotated screenshots, log files, and other relevant artifacts. This ensures that all stakeholders have access to the same information, fostering better collaboration and reducing the back-and-forth communication that often delays issue resolution.
- **Early Resolution and Customer Satisfaction**: With its comprehensive insights and self-solve capabilities, This tool empowers support teams to resolve customer issues at an earlier stage. By addressing problems swiftly, customer satisfaction levels soar, as frustrations are minimized, and the time taken for issue resolution is significantly reduced. Ultimately, this tool helps support teams deliver exceptional customer service and enhance overall customer experience.

**Note:**
The DiagnostiX.exe utility should be run with Administrator privileges. The DiagnostiX.exe.config, also provided with this utility, can be edited in any text editor (such as Notepad). In this file you can update details like user names, passwords, smtp clients, and so on.

For more information, see article ID 000068924 on the GE Vernova Support center: https://digitalsupport.ge.com/s/article/DiagnostiX-Historian-Diagnostic-Report-Tool

# Chapter 13. Collector Tool Kit

## Collector Toolkit Overview

### Overview

The Collector Toolkit lets you write programs that integrate tightly with Proficy Historian and let you leverage the same configuration tools, redundancy schemes, and health monitoring as collectors that ship with Proficy Historian. A custom collector is a collector developed using the Collector Toolkit. It collects data and messages from a data source and writes them to a Data Archiver. Using the Collector Toolkit, you can create custom collectors that:

- Collect data and messages from any data source
- Perform collector compression and buffer collected data
- Report data and messages to a local or remote Data Archiver

Custom collectors can be developed to function much like the standard OPC and iFIX collectors that come with the Proficy Historian product. The collected data can be used in any application that connects to Proficy Historian.

The toolkit enables development of programs that collect data at the current time. It is not suitable for developing migration programs, file import programs, SQL import programs or other programs that produce data which has timestamps in the past. Use other Proficy Historian toolkits to accomplish these task.

The toolkit supports pre-processing raw data with Python Expression Tags during collection, provided that you enable this. For details on how to use these tags, refer to the Python Expression Tags in *Data Collectors General* .

## Prerequisites

### Prerequisites

This topic covers prerequisites for using the Collector Toolkit in Windows. To use the Collector Toolkit in Linux, consult Installing and Configuring the Collector Toolkit for Linux *(on page 1900)*.

To create custom collectors using the Collector Toolkit:

- You must have Visual Studio 2010 and Historian 6.0 SP1 or higher installed and configured on your machine.
- You must ensure that you have administrative rights and open Visual Studio in Administrative mode.
- The collectors developed based on the Collector Toolkit must be written in C++.

**Note:**

- Both the computer the collector is running on and the Data Archiver it connects to must be Historian 6.0SP1 or higher. Also, Historian Administrator should be 6.0 SP1 or newer.
- You can create custom collectors using Collector Toolkit on both 32-bit and 64-bit collectors.

### Enabling Python Expression Tags with the Collector Toolkit

To enable Python Expression Tags with the Collector Toolkit:

1. Open the registry in regedit.
2. Navigate to the key created for the specific collector type for which you want expression use enabled. The collector type is the value of `ServiceName` that is specified in the method call to `CCollectorDelegator::InitializeCollector`.
    - If you are creating a 64 bit collector, this is under `HKEY_LOCAL_MACHINE\SOFTWARE\Intellution, Inc.\iHistorian\Services`.
    - If you are creating a 32 bit collector, this is under `HKEY_LOCAL_ MACHINE\SOFTWARE\Wow6432Node\Intellution, Inc.\iHistorian\Services`.
    - For example, if you are creating a 32 bit collector with a `ServiceName` of `RabbitMQCollector` with the toolkit, go to `HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Intellution, Inc.\iHistorian\Services\RabbitMQCo`.
3. Under that key, add a string variable with the following values:
    - Name: ServiceExtensions
    - Type: REG_SZ
    - Data: PythonExpressions
4. Ensure that the Python Expression Extension binary dependencies (casablanca100.dll, python34.dll, PythonExpressionExtension.dll) are in the same location as the collector executable.

Refer to the *Python Expression Tags* for details on how to use these tags.

# Installing the Collector Toolkit with Historian

## About Installation

This topic covers installing the Collector Toolkit with Historian in Windows. To use the Collector Toolkit in Linux, see Installing and Configuring the Collector Toolkit for Linux *(on page 1900)*.

Before you install the Collector Toolkit, verify that Microsoft Visual Studio 2010 is already installed on your computer. The Collector Toolkit is one of the standard options available during the installation of Proficy Historian. User developed Collectors built with the Collector Toolkit will require a client license in the same manner as other API programs. Run the install on the computer on which you want to develop your collector and select the Collector Toolkit option at the prompt. Refer to the Getting Started with Historian guide for the complete installation procedure

## Creating the Custom Collector Using the Wizard

You can create a custom collector using the Visual Studio wizard or the sample application available as part of the installation. If you want to create a collector using the wizard, follow the below procedure.

The toolkit, by default, gets saved in the same installation folder as Historian in the Program files folder of your computer. For example, `C:\Program Files\Proficy\Proficy Historian \CollectorToolkit\CollectorToolkit`. The example here indicates that Historian is installed in C drive. If Historian is installed in D drive, the location changes accordingly.

1. Navigate to the folder where the toolkit utility is saved. For example, `C:\Program Files \Proficy\Proficy Historian\CollectorToolkit\CollectorToolkit` in the case of a 32-bit collector and `C:\Program Files\Proficy\Historian\x86\CollectorToolkit` in case of a 64-bit collector.
2. Double-click the `CollectorToolkit.sln` file. A Microsoft Visual Studio window appears in Administrator mode.
3. Right-select the project in the Solution Explorer pane and select one of the following: **Build Solution** or **Rebuild Solution**.

   An output window appears indicating whether the Build or Rebuild has been successful. If you see the message, `Build succeeded or Rebuild All succeeded`, it indicates that the wizard has been activated. If the wizard is successful, the `CollectorToolkit.ico` and `CollectorToolkit.vsz` files appear in the wizards' folder in Visual Studio.
4. Proceed to create a custom collector.

# Installing and Configuring the Collector Toolkit for Linux

## Installing and Configuring the Collector Toolkit for Linux

The toolkit is compiled and tested on CentOS 7 and Ubuntu 14. The minimum system requirements are:

- g++(GCC) 4.8.x
- glibc2.17+

**Installing the Collector Tool kit for Linux**

The Collector Toolkit is delivered as a *tar* file. There are four different artifacts to choose from: static and shared library versions of debug and non-debug (release) versions. Choose whichever of the artifacts best fits your needs.

Unzip the tar file at a location where you want to install it. There are 4 folders at the top level.

- **shared**: The **shared** folder contains an example collector using the Collector Toolkit. A sample **Make** file is also provided. This example works as is. Your can use this as a template to create your own custom collector.
- **lib**: The `lib` folder contains two libraries: `libihAPI` and `libihCollectorDelegator`.
- **include**: The `include` folder contains include files that you would need to include in your custom collector. `ihAPI.h` is optional. It needs to be included only if you are making use of any System APIs directly in your custom collector.
- **bin**: The `bin` folder contains some pre-compiled binaries. `Random` is the binary for the Simulation Collector. `RandomValueSimulator` is the sample collector built using the example code in the `shared` folder.

**Configuring the Collector Tool Kit**

1. Update the following key in the `HistorianServers.reg` file: `[HKEY_LOCAL_MACHINE\Software\Intellution, Inc.\iHistorian\Services\<ServiceName>`.
   <ServiceName> must match the service name passed to the `InitializeCollector()` function in `main()` in your custom collector.
2. Update the following two properties in the `HistorianServers.reg` file.
   - `HistorianNodeName`: Update this to your Historian Server's name or IP Address. The custom collector tries to connect to the Historian Server specified by this property.
   - `InterfaceName`: Change this to the interface name of your choice. The custom collector uses this to identify itself in Historian Admin Console.

# Configuring Custom Collector Wizard

## Configuring a Custom Collector using the Wizard

You can create a custom collector using the Visual Studio wizard or the sample application available as part of the installation.

To create a collector using the wizard, follow the below procedure.

1. Navigate to the folder where the toolkit utility is saved. For example, `C:\Program Files \Proficy\ProficyHistorian\CollectorToolkit\CollectorToolkit`. The toolkit, by default, gets saved in the same installation folder as Historian in the Program files folder of your computer. The example here indicates that Proficy Historian is installed on the C drive. If Proficy Historian is installed in D drive, the location changes accordingly.
2. Double-click the `CollectorToolkit.sln` file. A Microsoft Visual Studio window appears in Administrator mode.
3. Right-select the project in the Solution Explorer pane and select one of the following: **Build Solution** or **Rebuild Solution**.

   An output window appears indicating whether the Build or Rebuild has been successful.

   - If you see the message, `Build succeeded or Rebuild All succeeded`, it indicates that the wizard has been activated. If the wizard is successful, the `CollectorToolkit.ico` and `CollectorToolkit.vsz` files appear in the wizards' folder in Visual Studio.
   - Ensure that the `CollectorToolkit.vsz` and `CollectorToolkit.ico` files exist in the above corresponding Operating system locations.
   - If you are working in 64 bit operating system, open `CollectorToolkit.vsz` in Notepad and replace the following line: `Param="ABSOLUTE_PATH = C:\Program Files\Proficy\Proficy Historian\CollectorToolkit\CollectorToolkit"` with `Param="ABSOLUTE_PATH = C:\Program Files\Proficy\Proficy Historian\x86\CollectorToolkit\CollectorToolkit"`. This enables you to open the wizard correctly.
4. Proceed to Creating a custom collector as described below.

## Creating a Custom Collector

**Note:**

The Collector Toolkit runtime for 32-bit collectors uses `ihCollector-DelegatorN.dll` and `ihCollector-DelegatorN.lib`, where *N* is a version number, which changes from version to version.

The Collector Toolkit runtime for 64-bit collectors uses `ihCollector-DelegatorN_x64.dll` and `ihCollector-DelegatorN_x64.lib`, where *N* is a version number, which changes from version to version.

1. Open Microsoft Visual Studio 2010 in Administrator Mode.
2. Open a new project by navigating to **File > New > Project**.
   The New Project page appears.
3. In the Installed Templates section, select **Wizards**.
   The Collector Toolkit Wizard window appears.
4. Enter the following information in the New Project page.
   - Provide the name of the collector in the Name field.
   - Provide the location where you want to save the Collector in the Location field. Select **Browse...** to navigate to the desired location.
   - Provide the solution name in the Solution name field. A solution name is the name of the Collector solution.
   - Select the Create directory for solution check box to create a directory for the collector.
5. Select **OK** to proceed.
   The Welcome to the Historian Collector Toolkit Wizard page appears.
6. Select **Next** to proceed.
   The Application Settings page appears. The options that you see on the Application Setting page are set by default. These are the recommended settings and should not be changed.
7. Select **Finish**

   The Microsoft Visual Studio appears in Administrator mode with the source code framework for the custom collector. Write the custom logic in the Skeletal Methods generated through the Wizard for the respective functionality. Refer to About Interfaces *(on page 1903)* for a description of the methods.

# Changing Historian Server Name

## Changing the Historian Server Name Using Registry

By default, custom collector tries to connect to the local Historian server. If you want the collector to connect to a remote Historian server, you must change the `HistorianNodeName` registry key located at:

- On a 64 bit machine, this is under `HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Intellution, Inc.\iHistorian\Services\<CollectorName>\`
- On a 32 bit collector, this is under `\HKEY_LOCAL_MACHINE\SOFTWARE\Intellution, Inc.\iHistorian\Services\<CollectorName>\`

# Working with collector Interfaces

## About Interfaces

A collector is an interface through which data can be sent to the Data Archiver.

The custom collector is a 32 bit windows executable, where you need to implement only the collector specific functionality. The toolkit provides the code that schedules collection, performs compression and buffering, and writes data to the Data Archiver.

The following is a list to provide better understanding.

### Interfaces Enabling Basic Functionality

- `ihCollectorToolkitPreInitialize`
- `ihCollectorToolkitInitialize`
- `ihCollectorToolkitInitializeCompleted`
- `ihCollectorToolkitDefinitionInitialize`
- `ihCollectorToolkitReset`
- `ihCollectorToolkitPropertyUpdate`
- `ihCollectorGetTags`
- `ihCollectorToolkitShutdown`

### Interfaces Enabling Advanced Functionality

These interfaces are further categorized as follows.

The following methods are used to perform operations on Polled tags:

- `ihCollectorToolkitPolledInit`
- `ihCollectorToolkitPolledInitCompleted`
- `ihCollectorToolkitASyncStartReading`
- `ihCollectorToolkitASyncAddTag`
- `ihCollectorToolkitASyncDeleteTag`
- `ihCollectorToolkitASyncReload`
- `ihCollectorToolkitDataCallback`

The following methods are used to perform operations on Hierarchical and Miscellaneous functions:

- `ihCollectorToolkitGetTagsHierarchical`
- `ihCollectorToolkitDestroyGroupData`
- `ihCollectorToolkitWriteValue`
- `ihCollectorToolkitDebugModeChange`
- `ihCollectorToolkitMiscReload`
- `ihCollectorToolkitGetData`

### Collector Development

You can develop a collector that uses polled reads or unsolicited reads or both. Polled reads are grouped within the shell are present to collector specific code a list of tags to be polled. The collector specific code can do whatever grouping is necessary or efficient for that data source. Unsolicited tags are simply passed on by one to the collector specific code at collector start up and on configuration change. It is up to the collector specific code to group tags and report data changes into the collector shell.

You can support tag browse or you can hard code responses to browse requests and require the tags to be created in some other way such as the Microsoft Excel tag import.

Throughout development, consider what information would be useful to see in the collector logs to troubleshoot issues and use the debug mode functionality to control what messages should appear. Your collector specific code will be notified of changes to debug mode in the `ihCollectorToolkitDebugModeChange` function.

Collection can be paused and resumed without the collector `exe` being restarted. Pause and resume are relayed to the collector specific code in the `ihCollectorToolkitReset` function.

Your collector can choose to support a heartbeat, status, and rate functionality via the `ihCollectorToolkitWriteValue` function. In this case the collector would write into the data source in addition to the reading done during collection. The toolkit will call the write function once per minute and you will need to write to the data source.

Your collector can react to configuration changes sent to collector specific code in the `ihCollectorToolkitPropertyUpdate` function. These are changes to the collector itself in the General1-5 fields not changes to the tag list

Possibly the most complex design decision is if you should support on-the-fly configuration changes or require collector restart. Adding or removing tags to collection or changing collection intervals for active tags is not always possible for all data sources. Consider this as an advanced feature.

# Custom Collector Design

## Design topics for Creating Custom Collectors

Among the first design decisions to be made with a collector is to map the data source's timestamps, values, and qualities to the Proficy Historian types.

**Timestamps**

You can let the toolkit assign timestamps to polled or unsolicited data or you can provide the timestamps with unsolicited data. If you choose to provide timestamps, you must provide them in the GMT+0 time zone. Proficy Historian offers microsecond resolution on timestamps, which should be sufficient for most applications.

**Values**

The Proficy Historian has a finite set of data types. If your data source has other types that cannot be converted then do not collect those tags. If there is any scaling or adjustments to the data, consider if those should be done in your collector specific code or via input scaling in the toolkit or calculation tags in the Data Archiver. Also, consider what the value means if the data quality is not good. In most parts of Proficy Historian, if you set the data quality to bad then the value will be ignored.

**Quality**

This can be the most important design decision of your collector. Your design needs to indicate what data is meaningful and should be used in trends and reports and what data is not meaningful. If the Historian contains tags from other data sources, you should be consistent with other collectors so that the retrieval program need to be aware of where the data came from.

Once you have a design for timestamp, value, and quality, you should think about how best to group your tags for the most efficient collection. The toolkit will pass a list of polled and unsolicited tags to your code at startup and their intervals. For polled tags, you can mostly ignore the tags because you will be sent a list later when it is time to poll the data. For unsolicited tags, you need to build your own groups and notify the toolkit when data changes.

Adding Browsing functionality is optional but is recommended to have. Think about how your code will respond to browse requests and how you will filter the tagnames and descriptions.

Your code does not need to do anything for collector compression or for store and forward of data.

You collector can utilize the same log file mechanism as other collectors but you should decide what you should write to help troubleshoot the collector.

Finally, you should consider how your collector should react to tag changes made while the collector is running. Tags can be added or deleted or properties like Collection interval or Deadband can be modified.

# Backward Compatibility of the Collector Toolkit

## Backward Compatibility of the Collector Toolkit

From time to time, the runtime library files of the Collector Toolkit (namely `ihCollectorDelegator.dll` and its variants) undergo changes to introduce new features or fix defects.

In some cases, changes are introduced that break backward compatibility. This means that the new runtime library does not work with custom collectors developed with past versions of the Toolkit. In such cases, a new version number appears in the names of the DLL files to indicate that they should not replace previous releases of the runtime library.

In other cases, a new runtime library, with changes, still works with custom collectors developed with past versions of the Toolkit. If the replacement library has the same name as the replaced library, you can safely replace earlier versions of the runtime library with the new library.

The Collector Toolkit includes the latest version of header files and import libraries (`.h` and `.lib` files), but no previous versions of these files. However, the Toolkit ships all previously released runtime DLLs that are necessary for custom collectors developed in the past to run. When there have been successive releases of a DLL that are backward compatible (indicated by their shared name), only the latest release is shipped.

If you want to develop new custom collectors, you should acquire and use the latest version of the Collector Toolkit. If you want to maintain or fix defects in an existing custom collector, you can use the version of the Toolkit with which the collector was developed. Alternately, you may want to try a more recent version. However, in such a case, some changes to the source code may be necessary if the Toolkit contains changes that break backward compatibility.

# Custom Collector Toolkit Interface Technical Reference

## Custom Collector Toolkit Interface Technical Reference

The Collector Toolkit interfaces help in writing collectors with basic functionality such as initializing collectors, adding data and tags, the others help in writing advanced features such as store and forward functions, heartbeat, buffering, queuing. The following are the various interfaces in detail:

**ihCollectorToolkitASyncAddTag**

The shell will call this function at start up and on change to indicate a tag that should be collected. The collector specific code should keep all information about the tags and use it later to deliver the collected data.

**Prototype**

```
int ihInterfaceTKASyncAddTag(ihInterfaceTKASyncTagInfo *ASyncTag,
int IsCollectorStarting)
```

**Returns**

TRUE on success. FALSE if the tag could not be added. For example, if the tag has an invalid address you can return an error here or report bad data later after `ihCollectorToolkitAsyncStartReading` is called.

**Parameters**

| Name | Data Type | Description |
|---|---|---|
| `ASyncTag` | `ihInterfaceTKASyncTagInfo` | Refer to the structure description. |
| `IsCollectorStarting` | `int` | TRUE if this function is being called as part of collector start-up. FALSE if this is being called because a tag was added or modified while the collector was running. |

**ihCollectorToolkitASyncDeleteTag**

Use this tag to update information to the Collector when a tag is deleted so that it will stop collecting data from the source. For example, if you delete a tag or stop collection in the administrator then the shell will call this function.

**Prototype**

```
int ihCollectorToolkitASyncDeleteTag(int tagId)
```

**Returns**

TRUE on success; your collector specific code should always return TRUE.

**Parameters**

| Name | Data Type | Description |
|---|---|---|
| `tagId` | Int | Tag identifier; the tag name is not passed into this function. You would have to match the tag identifier with a magnate passed in `ihCollectorToolkitASyncAddTag`. |

**ihCollectorToolkitASyncInit**

The collector shell will call this to indicate the start of a list of tags to be collected asynchronously. The tags will be delivered via the `ihCollectorToolkitASyncAddTag` function and the collector specific code should keep track of all tag information.

**Prototype**

```
int ihInterfaceTKASyncInit(void);
```

**Returns**

TRUE on success; your collector specific code should always return TRUE.

**ihCollectorToolkitASyncInitCompleted**

The collector shell will call this method to indicate the end of a list of tags to be collected asynchronously.

**Prototype**

```
int ihInterfaceTKASyncInitCompleted(void)
```

**Returns**

TRUE on success; your collector specific code should always return TRUE.

**ihCollectorToolkitASyncReload**

This function is used to perform a recalculation on all unsolicited tags in the custom collector. Custom collector must implement the necessary changes to read data from the source and send values via the `ihCollectorToolkitDataCallback`. The time frame for recalculation is given in the input parameters `StartTime` and `EndTime`.

For more information on the methods of unsolicited tags, refer the methods listed for unsolicited tags in About Interfaces *(on page 1903)*.

**Prototype**

```
int ihCollectorToolkitASyncReload(ihTKTimeStruct TimeStruct *StartTime, ihTKTimeStruct TimeStruct
 *EndTime)
```

**Returns**

Always, TRUE.

**Parameters**

| Name | Data Type | Description |
|---|---|---|
| `StartTime` | `ihTKTimeStruct` | Reserved |
| `EndTime` | `ihTKTimeStruct` | Reserved |

### ihCollectorToolkitASyncStartReading

The collector shell will call this method to indicate that the collector should start reading data for asynchronous tags from the source and report any data changes to the shell.

**Prototype**

```
int CustomCollector:: ihCollectorToolkitASyncStartReading(void)
```

**Returns**

TRUE on success; your collector specific code should return TRUE.

### ihCollectorToolkitDataCallback

This callback is used when unsolicited tags get the data and asynchronously writes them into the Historian. Custom collector receives the data from unsolicited tags and sends it to the `ihCollectorToolkitDataCallback`. This call back function internally writes the data into the Historian.

**Prototype**

```
voidihCollectorToolkitDataCallback(void *Param)
```

**Returns**

Void.

**Parameters**

| Name | Data Type | Description |
|---|---|---|
| `ASyncTag` | `ihInterfaceTKASyncData` | Refer to the structure description. |

**Example**

You can use this function as shown in the following example to create a thread using which the unsolicited tags

data is written into the Historian.

```
/// Summary;
/// Unsolicited dedicated thread corresponding method.
Here all unsolicited tags get data from source(usually, way should be source
notification to historian) and sends to historian
/// </summary>
/// <param name="param">;Collector instance&lt;/param>
UINT RandomValueSimulator::TKAsyncReadFunc(void* param)
{
   POSITION pos = NULL;
   RandomValueSimulator* pColl = (RandomValueSimulator*) param;
   while (TRUE)
   {
     if (g_DoAsyncRead)
     {
         CSingleLock lock(&;g_Sync, TRUE);
         if (!g_AsyncTags.IsEmpty())
         {
           // gets each unsolicited tag into ihInterfaceTKAsyncTagInfo object
            if (!pos)
              pos = g_AsyncTags.GetHeadPosition();
              ihInterfaceTKASyncTagInfo* tag = g_AsyncTags.GetNext(pos);
              int numTags = 1;
              ihInterfaceTKDataInfo data;
              memset(&;data, 0, sizeof(ihInterfaceTKDataInfo));
              data.Tag = pColl->;TKStrdup(tag->;Tag);
              data.DataProp.ValueDataType = tag->;DataType;
              data.DataProp.TimeStamp = pColl->;TKGetSystemTime();
              // gets data for selected tag
              pColl->;ihCollectorToolkitGetData(0, 0, 0, numTags, NULL, &;data);
              unsigned long collectionTime = (unsigned long)time(0);
              int tagId = tag->;TagId;
              ihInterfaceTKASyncData asyncData;
              memset(&;asyncData, 0, sizeof(ihInterfaceTKASyncData));
```

```
            asyncData.NumValues = numTags;

            asyncData.TagIds = &;tagId;

            asyncData.Values = &;data.DataProp;

            asyncData.CollectionTimes = &;collectionTime;

            // sends to historian

            pColl->;ihCollectorToolkitDataCallback(&;asyncData);

            }

            else

            {

              pos = NULL;

               }

            }

            else

            {

              pos = NULL;

            }

            long sleepTimeInMs = 500 + (rand() % 46) * 100;

            // sleep from 0.5 to 5 seconds

            Sleep(sleepTimeInMs);

      }

      return 0;

}
```

### ihCollectorToolkitDefinitionInitialize

Use this interface to specify what labels you want to appear on the collector configuration page of the administrator. The administrator will show 5 edit boxes for General 1 to General 5 but with this function you can control what text is show next to each edit box.

All custom collectors share the same set of labels.

#### Prototype

```
void ihCollectorToolkitDefinitionInitialize(ihCollectorToolkitDefProperties* CustomPropertyDesc)
```

#### Returns

Void.

#### Parameters

| Name | Data Type | Description |
|---|---|---|
| `CustomPropertyDesc` | `ihCollectorToolkitDefProperties` | Pointer to a Custom Collector's specific properties General 1-5 definitions. |

**ihCollectorToolkitDestroyGroupData**

Reserved for future use. Your collector specific code should call the `CCollectorDelegator` function.

**Prototype**

```
void ihCollectorToolkitDestroyGroupData(int NumTags, void *Data, void *Misc)
```

**Returns**

Void.

**Parameters**

| Name | Data Type | Description |
|---|---|---|
| `NumTags` | `int` | Reserved |
| `Data` | `void` | Reserved |
| `Misc` | `void` | Reserved |

**ihCollectorToolkitDebugModeChange**

The collector calls this method when a change is detected. 0 indicates only errors and important information should be written to log. 255 is the highest setting for all debug information to display.

**Prototype**

```
void ihCollectorToolkitDebugModeChange(int DoDebug)
```

**Returns**

Void.

**Parameters**

| Name | Data Type | Description |
|---|---|---|
| `DoDebug` | `int` | Zero or non zero debug level. |

**ihCollectorToolkitGetData**

This is the function called by the collector shell each time the collector should poll data from the data source. The shell will pass a list of tags by tag id to be read. The tagids were provided to the collector specific code in the `PolledAddTag` call. The string tagnames, if needed, are available in the `ihInterfaceTKDataInfo` structure. The collector should gather the data and fill in the remaining fields of the `ihInterfaceTKDataInfo`.

**Prototype**

```
void ihCollectorToolkitGetData(int MinInterval, int AnySourceTimes, int CreatePermGroup, int NumTags,
int *TagIds, ihInterfaceTKDataInfo *Data)
```

**Returns**

Void.

**Parameters**

| Name | Data Type | Description |
|---|---|---|
| `MinInterval` | `int` | The minimum configured collection interval of the tags being passed in for polling. |
| `AnySourceTimes` | `Int` | TRUE if any of the tags to be collected are configured for Time assigned by source. This would mean the collector specific code has to do additional work to provide a timestamp with the data. If this is FALSE, then the collector shell will provide the timestamp. |
| `CreatePermGroup` | `Int` | FALSE if this is a one time read such at start up or collector redundancy fail over. In this case, you can choose to not make a permanent collection group inside your code. If this is TRUE, then this is a normal scheduled polled read and you should |

| Name | Data Type | Description |
|---|---|---|
| | | consider making a permanent group in your code for greater efficiency. |
| `NumTags` | `Int` | Number of tags to be read. |
| `TagIds` | `Int` | List of tags to be read, identified by tag ID. The tag names are in the data parameter, if required. |
| `Data` | `ihInterfaceTKDataInfo` | Pointer to the Proficy Historian Collector Data Properties. This contains some information from the shell and then the collector specific code needs to fill in the collected values. |

**Example**

Populating data for a single tag.

```
ihInterfaceTKDataInfodata;

memset(&data,0, sizeof(ihInterfaceTKDataInfo));

data.Tag= L”computername.OperatorName”

data.DataProp.ValueDataType= ihTKFloat;       // correct data type- same as tag type

data.DataProp.TimeStamp= pColl->TKGetSystemTime(); //10/01/2014 07:05:00

data.DataProp.Value = 10;   // value that matches the data type.

data.DataProp.Quality = ihTKOPCGood;
```

### ihCollectorToolkitGetTagsHierarchical

Use this interface to retrieve and represent the data in a hierarchical manner.

**Prototype**

```
void ihCollectorToolkitGetTagsHierarchical(wchar_t* BrowsePosition, wchar_t* NodeFilter,

ihInterfaceTKHierarchicalBrowseResponse* Response)
```

**Returns**

Void.

**Parameters**

| Name | Data Type | Description |
|---|---|---|
| `BrowsePosition` | `wchar_t*` | Position of browsing for specific Tag. |
| `NodeFilter` | `wchar_t*` | Node filtering required for tag to get hierarchical information. |
| `Response` | `ihTKHierarchicalBrowseResponse` | Hierarchical Groups of tags browse response. |

### ihCollectorGetTags

Use this interface to browse tags from source. The shell will give some criteria and the collector specific code should fill in the result.

Tags can be skipped when they exist in the data source but should not be collected into Proficy Historian. For example, if a tag has a data type that is not compatible with Proficy Historian, you would not return it and you can increment the number skipped.

**Prototype**

```
void ihCollectorToolkitGetTags(ihInterfaceTKTagRecordset *Tags, ihInterfaceTKCfgInfo *Cfg, wchar_t
 *BrowsePosition,
ihTKBoolean Recursive, wchar_t *DescriptionMask, wchar_t *SourceAddressMask, int CanReadASync,
int DoMsgPump, int *NumSkipped
```

**Returns**

Void.

**Parameters**

| Name | Data Type | Description |
|---|---|---|
| `Tags` | `ihInterfaceTKRecordset` | Pointer to a Proficy Historian Tags Record set containing browse criteria and where you can return the tags. |

| Name | Data Type | Description |
|---|---|---|
| `Cfg` | `ihInterfaceTKCfgInfo` | Pointer to Proficy Historian Collector Configuration Information. Use this if necessary to help determine the browse results. |
| `BrowsePosition` | `wchar_t` | Pointer to a position of browsing for specific Tag. |
| `Recursive` | `ihTKBoolean` | TRUE if the browse should be recursive in collector specific code. Applies only if your data source has a tree of available tags. |
| `DescriptionMask` | `wchar_t` | * or a description mask that you should use when determining tags to return. |
| `SourceAddressMask` | `wchar_t` | * or a source address mask that you should use when determining tags to return. |
| `CanReadASync` | `int` | Non-zero if your collector is capable of asynchronous reads. Typically, this parameter does not affect browse results. |
| `DoMsgPump` | `int` | 0 should be ignored. |
| `NumSkipped` | `int` | Use this to return any tags that were skipped because they did not match the description or source address mask or had an invalid data type. |

**ihCollectorToolkitInitializeCompleted**

When this method is called by the collector shell, it indicates that the initialization is complete.

**Prototype**

```
ihCollectorToolkitInitializeCompleted(void)
```

**Returns**

FALSE on success; collector should return FALSE.

### ihCollectorToolkitInitialize

Use this interface to initialize the collector by using information in the `ihInterfaceTKCfgInfo` and `ihInterfaceTKPreCfgInfo` structures. Your code also needs to save away the callback pointers passed into this function for use later with unsolicited reads and browses.

**Prototype**

```
int ihCollectorToolkitInitialize(ihInterfaceTKCfgInfo *Cfg, ihInterfaceTKPreCfgInfo *PreCfg, wchar_t
 *ErrorMsg,
int ErrorMsgSize,wchar_t *RegKeyName, ihCollectorToolkitCollectorCallbacks *Callbacks, int DoDebug)
```

**Returns**

TRUE if initialization was successful. Otherwise, FALSE. If initialization failed, you can return a text string error message in the `ErrorMsg` parameter and that message will be entered into the collector log.

**Parameters**

| Name | Data Type | Description |
|---|---|---|
| `Cfg` | `ihInterfaceTKCfgInfo` | Pointer to the Proficy Historian Collector Configuration Information. |
| `PreCfg` | `ihInterfaceTKPreCfgInfo` | Pointer to the Collector pre-Configuration Information. |
| `ErrorMsg` | `wchar_t` | Pointer to an Initialization Error Message, if any. Any error would be reported to collector log. |
| `ErrorMsgSize` | `int` | Error Message Size. This is the maximum length that your error message size can be. |

| Name | Data Type | Description |
|---|---|---|
| `RegKeyName` | `wchar_t` | Pointer to a Registry Key Name. Will be filled in with the location of your collector's registry information. |
| `Callbacks` | `ihTKCollectorCallbacks` | Pointer to callbacks to functions inside the collector shell. Save these function pointers and call them at the appropriate time, such as to deliver unsolicited data into the shell. |
| `DoDebug` | `int` | This value will be zero if the collector is not in debug mode and non-zero if debug mode is enabled. Save this value and use it to log more or less information to your collector log. |

**ihCollectorToolkitMiscReload**

Reserved for future use. Your collector specific code should simply call the function in the `CCollectorDelegator`.

**Prototype**

```
int ihCollectorToolkitMiscReload(ihTKTimeStruct *StartTime, ihTKTimeStruct *EndTime)
```

**Returns**

TRUE; your collector specific code should return TRUE.

**Parameters**

| Name | Data Type | Description |
|---|---|---|
| `*StartTime` | `ihInterfaceTKTimeStruct` | Reserved |
| `*EndTime` | `ihTKTimeStruct` | Reserved |

### ihCollectorToolkitPolledAddTag

The collector shell calls this to indicate the tag will be used later in polled collection. The collector specific code should keep all information given as it will be used in the `ihInterfaceGetData` call.

The polled tag add notification is done via `ihCollectorToolkitPolledAddTag` and data is sent to Historian via `ihCollectorToolkitGetData`. The frequency of asking for data for polled tag is based on the tag collection interval.

**Prototype**

```
int ihCollectorToolkitPolledAddTag(ihInterfaceTKPolledTagInfo *PolledTag, int IsCollectorStarting)
```

**Returns**

TRUE on success; your collector specific code should return TRUE.

**Parameters**

| Name | Data Type | Description |
|---|---|---|
| `PolledTag` | `ihInterfaceTKPolledTagInfo` | Pointer to a Historian Collector Polled Tag Information. |
| `IsCollectorStarting` | `int` | Parameter to provide information about collector state. |

### ihCollectorToolkitPreInitialize

Use this interface to provide the toolkit with some of the capabilities of the collector. Here, you can specify if you support multiple instances, if you support unsolicited reads, and if your collector supports browsing.

Information filled into the `ihInterfaceTKCfgInfo` structure will later be passed into the `ihCollectorToolkitInitialize` function.

**Prototype**

```
void ihCollectorToolkitPreInitialize(ihInterfaceTKPreCfgInfo *PreCfg, ihInterfaceTKCfgInfo *Cfg)
```

**Returns**

Void.

**Parameters**

| Name | Data Type | Description |
|---|---|---|
| `PreCfg` | `ihInterfaceTKPreCfgInfo` | Pointer to Collector Configuration Information. |
| `Cfg` | `ihInterfaceTKCfgInfo` | Pointer to additional Collector Configuration Information. |

**ihCollectorToolkitPropertyUpdate**

This function is called when the user changes any collector property in Historian Administrator such as the General1 -5 properties. Your collector specific code can decide what changes are meaningful.

**Prototype**

```
void ihCollectorToolkitPropertyUpdate(ihInterfaceTKCfgInfo *Cfg)
```

**Returns**

Void.

**Parameters**

| Name | Data Type | Description |
|---|---|---|
| `Cfg` | `ihInterfaceTKCfgInfo` | Current values of Collector configuration information. |

**ihCollectorToolkitPolledInit**

The collector shell calls this method at start up and on change to indicate the start of a list of tags to be polled, as the collector runs. The actual polling takes place in the `ihInterfaceGetData` call.

**Prototype**

```
int ihCollectorToolkitPolledInit(void);
```

**Returns**

FALSE on success; your collector specific code should return FALSE.

**ihCollectorToolkitPolledInitCompleted**

The collector shell will call this to indicate the end of the polled tag list. This informs the source to begin its initialization.

**Prototype**

```
int ihCollectorToolkitPolledInitCompleted(void);
```

**Returns**

FALSE on success; your collector specific code should return FALSE.

### ihCollectorToolkitPropertyUpdate

This function is called when the user changes any collector property in Historian Administrator such as the General1 -5 properties. Your collector specific code can decide what changes are meaningful.

**Prototype**

```
void ihCollectorToolkitPropertyUpdate(ihInterfaceTKCfgInfo *Cfg)
```

**Returns**

Void.

**Parameters**

| Name | Data Type | Description |
|---|---|---|
| `Cfg` | `ihInterfaceTKCfgInfo` | Current values of Collector configuration information. |

### ihCollectorToolkitPolledDeleteTag

The shell calls this function to notify the collector specific code that a tag will no longer be polled.

**Prototype**

```
int ihCollectorToolkitPolledDeleteTag(int tagId);
```

**Returns**

TRUE on success; your collector specific code should return TRUE.

**Parameters**

| Name | Data Type | Description |
|---|---|---|
| `TagId` | `int` | TagId that is being deleted. The tag name is not passed into this function. Match the TagId with the information passed in the `ihCollector-ToolkitPolledAddTag`. |

### ihCollectorToolkitPolledInitCompleted

The collector shell will call this to indicate the end of the polled tag list. This informs the source to begin its initialization.

**Prototype**

```
int ihCollectorToolkitPolledInitCompleted(void);
```

**Returns**

FALSE on success; your collector specific code should return FALSE.

### ihCollectorToolkitReset

The toolkit will call this function when the collector is being paused or shutdown. Proficy Historian stores collector information before the Collector is shut down.

**Prototype**

```
void ihCollectorToolkitReset(int Shutdown)
```

**Returns**

Void.

**Parameters**

| Name | Data Type | Description |
|---|---|---|
| `Shutdown` | `int` | 0 if the collector pauses and non-zero if the collector is shutting down. |

### ihCollectorToolkitStartReading

When the shell calls this method, it indicates the collector specific code to start polled and unsolicited reading from the source.

**Prototype**

```
int ihCollectorToolkitStartReading(void)
```

**Returns**

TRUE on success; your collector specific code should return TRUE.

### ihCollectorToolkitShutdown

The collector shell will call this function when the collector executable (`.exe`) is shutting down.

**Prototype**

```
void ihInterfaceTKShutdown(void);
```

**Returns**

Void.

### ihCollectorToolkitTagChange

The toolkit will call this function when the properties of a tag are changed in Client Tools, Historian Administrator, or other external tools. The tag and what properties have changed are indicated in the tag props and changed fields.

This function is called on tag property change and is not involved with data changes.

Your collector should decide how to react to tag property changes or should ignore the changes.

**Prototype**

```
int ihCollectorToolkitTagChange(int TagId, iHTKTagProperties *TagProps, iHTKTagFields *ChangedFields
```

**Returns**

int (TRUE/FALSE).

**Parameters**

| Name | Data Type | Description |
|---|---|---|
| `TagId` | `int` | TagID that has changed. |
| `TagProps` | `ihTKTagProperties` | Pointer to a Proficy Historian Tag Property structure that indicates the new tag property values. |
| `ChangedFields` | `ihTKTagFields` | Pointer to information about which tag properties have changed. |

### ihCollectorToolkitGetEnumeratedSets

The toolkit calls this function based on the value configured under the `SynchInterval` registry key value. This function is called every *x* minutes, where *x* is the value configured for the registry key.

Collectors implemented using the Collector Toolkit have to implement this method to fetch the `EnumeratedSet` from their source.

**Prototype**

```
int ihCollectorToolkitGetEnumeratedSets(ihTKEnumeratedSetRecordSet *EnumeratedSetRecordSet)
```

**Returns**

int (TRUE/FALSE).

**Parameters**

| Name | Data Type | Description |
|---|---|---|
| `EnumeratedSetRecordSet` | `ihTKEnumeratedSetRecordSet` | Pointer to an `EnumeratedSetRecordSet` structure which should be added to Historian Server. |

**ihCollectorToolkitWriteValue**

The toolkit will call this function once per minute if the user has configured heartbeat/status/rate tags. If your collector does not support this feature you can hard code a return value and not do the write.

**Prototype**

```
void ihCollectorToolkitWriteValue(wchar_t *Tag, wchar_t *SValue, double DValue
```

**Returns**

Void.

**Parameters**

| Name | Data Type | Description |
|---|---|---|
| `Tag` | `wchar_ t` | Pointer to a source address in the data source to write to. |
| `doubled Value` | `DValue` | Numeric value to be written, if the Tag is a numeric tag. For example, a report rate would be written to a rate tag or a 1 would be written to a heartbeat tag. |

| Name | Data Type | Description |
|---|---|---|
| `SValue` | `SValue` | The string value to be written if tag source address is a string tag. For example, the string Running would be written to a status tag. |

**What is a Helper Method?**

Helper Methods are used to allocate memory for variables while developing Collectors. Using these helper methods will prevent cross boundary issues. The following are the Helper Methods:

| Name | Description |
|---|---|
| `wchar_t*TKStrdup(const wchar_t* string)` | Use this method to allocate memory on the heap, assign values and returns. |
| `voidTKFree(void* pointer)` | Use this method to free up pointer memory. |
| `void*TKMalloc(_In_ size_t _Size)` | Use this method to allocate memory. |

# Custom Structure Technical Reference

## What is a Structure?

A Structure is a user defined data type. Structures are used in Interface Methods to represent data types that are compatible with Historian and allow the custom collectors to interact with Historian.

**Note:**
The abbreviation, TK stands for Toolkit.

For developing basic Custom Collectors, you should have knowledge about the following structures:

- `ihInterfaceTKPreCfgInfo`, `ihInterfaceTKCfgInfo` - Maintains collector configuration information.
- `ihCollectorToolKitDefProperties` - Maintains collectors specific properties.
- `ihInterfaceTKDataInfo` - Maintains data information.
- `ihInterfaceTKRecordset` - Maintains tag `recordset`.
- `ihTKTagProperties` - Maintains tag properties.
- `ihTKTagFields` - Maintains tag fields details
- `ihTKTagCriteria` - Maintains tag criteria details.

- `ihTKDataType` - Maintains tag data type
- `ihInterfaceTKHierarchicalBrowseResponse` - Maintains hierarchical browse response information.
- `ihInterfaceTKASyncTagInfo` - Maintains Unsolicited tag information.
- `ihInterfaceTKPolledTagInfo` - Maintains Polled tag information.

The following are few more structures that are used for developing Custom Collectors:

- `ihTKTagRecordset`
- `ihTKBlobData`
- `ihTKHiddenValue`
- `ihTKRawQuality`
- `ihTKQuality`
- `ihTKCommentData`
- `ihTKComments`
- `ihTKDataProperties`
- `ihInterfaceTKASyncData`
- `ihTKGetDataType`
- `ihTKStatus`
- `ihTKMessageTopic`
- `ihTKCollectorCallbacks`
- `ihTKTimeStruct`
- `ihTKQualityStatus`
- `ihTKQualitySubStatus`
- `ihTKDataType`
- `ihTKInterfaceType`
- `ihTKCollectionType`
- `ihTKTimeStampType`
- `ihTKTimeResolution`
- `ihTKTagId`
- `ihTKConditionCollectionComparison`
- `ihTKAlarmInterfaceProperties`
- `ihTKHierarchicalBrowseResponse`

## Custom Collector Toolkit Structure Reference

Following are the custom Collector Toolkit structure references:

**ihInterfaceTKPreCfgInfo**

The `ihInterfaceTKPreCfgInfo` structure maintains the pre-configuration information of a collector.

### Definition

```
typedef struct ihInterfaceTKPreCfgInfo {
ihTKInterfaceType InterfaceType;
int MultipleInstancesAllowed;
int MinimumInterval;
int MaxTagsPerRead;
int CanReadASync;
int CanBrowseSource;
int CanSourceTimestamp;
int ForceInputScaling;
int NeedMsgPump;
float ForcedScaleHI;
float ForcedScaleLO;
int ForceAsyncSource;
int ForcePolledSource;
int AyncAllowAdjustWhenSource;
int PolledAllowAdjustWhenSource;
int DontWriteASyncOfflineValue;
int DontWritePolledOfflineValue;
int DoesReloadMode;
int DoesLagTimes;
ihTKTagFields ReloadTagFields;
ihTKTagFields NotifyTagFields;
int IsHistoricalCollector;
int CanBrowseHierarchical;
wchar_t ReloadModeName[100];
TKCallBackFunctionCalcLibraryTag * CalcLibraryTagCallback;
void * CalcLibraryTagCallbackParam;
int CanSendOPCQuality;
} ihInterfaceTKPreCfgInfo;
```

### Parameters

| Name | Description |
|---|---|
| `InterfaceType` | Indicates the collector responsible for collecting data for the tag. |

| Name | Description |
|---|---|
| `MultipleInstancesAllowed` | Indicates whether multiple instances are allowed. |
| `MinimumInterval` | Indicates the minimum interval the collector uses. |
| `MaxTagsPerRead` | Indicates the maximum tags per read. |
| `CanReadASync` | Indicates whether Async tags can be read. |
| `CanBrowseSource` | If True, this column indicates that the collector is capable of browsing its source for tags. |
| `CanSourceTimestamp` | Indicates whether the data source is capable of providing timestamps along with the data. |
| `ForceInputScaling` | Forces input scaling to be used. |
| `NeedMsgPump` | Indicates whether a message pump is required. |
| `ForcedScaleHI` | Forces to use Hi Scaling. |
| `ForcedScaleLO` | Forces to use Low scaling. |
| `ForceAsyncSource` | Forces to use Async source. |
| `ForcePolledSource` | Forces to use polled source. |
| `AsyncAllowAdjustWhenSource` | Forces Async to allow adjust when source. |
| `PolledAllowAdjustWhenSource` | Forces to polled to allow adjust when source. |
| `DontWriteASyncOfflineValue` | Forces not to write Async offline value. |
| `DontWritePolledOfflineValue` | Forces not to write polled offline values. |
| `DoesReloadMode` | Uses the reload mode. |
| `DoesLagTimes` | Defines the lag times. |
| `IsHistoricalCollector` | Indicates whether it is a historical collector. |
| `CanBrowseHierarchical` | Indicates whether a hierarchical browse can be done. |
| `ReloadModeName` | Reload mode name. |
| `CalcLibraryTagCallback` | Call back function for calculation tag. |
| `CalcLibraryTagCallbackParam` | Calculation tag call back parameters |
| `CanSendOPCQuality` | Indicates whether OPC Quality can be sent. |

### ihInterfaceTKCfgInfo

The `ihInterfaceTKCfgInfo` structure maintains the configuration information of the collectors.

#### Definition

```
typedef struct ihInterfaceTKCfgInfo {
int CanBrowseSource;
int CanSourceTimestamp;
int ForcePolled;
int DoOnFly;
wchar_t* CustomProp1;
wchar_t* CustomProp2;
wchar_t* CustomProp3;
wchar_t* CustomProp4;
wchar_t* CustomProp5;
wchar_t StatusOutputAddress[500];
wchar_t RateOutputAddress[500];
wchar_t HeartbeatOutputAddress[500];
wchar_t InterfaceName[500];
wchar_t HistorianNodeName[1024];
wchar_t LogFilePath[500];
wchar_t BufferFilePath[500];
uint32 MinimumDiskFreeBufferSize;
uint32 MaximumMemoryBufferSize;
int ShouldAdjustTime;
int ShouldQueueWrites;
int SourceTimeInLocalTime;
volatile int32_t CollectionDelay;
wchar_t DefaultTagPrefix[200];
uint32_t DefaultCollectionInterval;
ihTKCollectionType DefaultCollectionType;
ihTKTimeStampType DefaultTimeStampType;
int DefaultLoadBalancing;
int DefaultCollectorCompression;
float DefaultCollectorCompressionDeadband;
uint32_t DefaultCollectorCompressionTimeout;
int ReloadMode;
volatile int32_t DisableOnTheFlyTagChanges;
```

```
uint32_t ReadLagTime;

int32_t MaxHistoricalRecoverySeconds;

int32_t HistoricalOverrunThresholdSecs;

int DontStartupPolling;

int DefaultSpikeLogic;

float DefaultSpikeMultiplier;

uint32_t DefaultSpikeInterval;

volatile int32_t DataRecoveryQueueEnabled;

int DefaultAbsoluteDeadbanding;

double DefaultAbsoluteDeadband;

int RedundancyEnabled;

wchar_t RedundancyPrincipalCollector[500];

volatile int32_t RedundancyIsActiveCollector;

wchar_t RedundancyPrimaryCollector[500];

uint32_t SyncThreadInterval;

} ihInterfaceTKCfgInfo;
```

**Parameters**

| Name | Description |
|---|---|
| `CanBrowseSource` | If True, this column indicates that the collector is capable of browsing its source for tags. |
| `CanSourceTimestamp` | Indicates whether the data source is capable of providing timestamps along with the data. |
| `ForcePolled` | If you set as true then collection type will be forced to be `ihPolled`. |
| `DoOnFly` | Indicates whether to enable or disable on the fly changes. |
| `CustomProp1` | The general (or spare) configuration fields for the collector. The `CollectorGeneral1` column is not user defined; it is different for each collector. |
| `CustomProp2` | The general (or spare) configuration fields for the collector. The `CollectorGeneral2` column is not user defined; it is different for each collector. |

| Name | Description |
|---|---|
| `CustomProp3` | The general (or spare) configuration fields for the collector. The `CollectorGeneral3` column is not user defined; it is different for each collector. |
| `CustomProp4` | The general (or spare) configuration fields for the collector. The `CollectorGeneral4` column is not user defined; it is different for each collector. |
| `CustomProp5` | The general (or spare) configuration fields for the collector. The `CollectorGeneral5` column is not user defined; it is different for each collector. |
| `StatusOutputAddress[500]` | The address in the source database into which the collector writes the status signal output. |
| `RateOutputAddress[500]` | An address or tag name in the data source into which the collector writes the current value of the events per minute output. |
| `HeartbeatOutputAddress[500]` | The address in the source database into which the collector writes the heartbeat signal output. |
| `InterfaceName[500]` | The name of the collector. The collector name is unique in a given Historian Server. |
| `HistorianNodeName[1024]` | The name of the `DataArchiver` server where we need to store our data. |
| `LogFilePath[500]` | Path name to specify the location of the log files. |
| `BufferFilePath[500]` | Path name to specify the location of the buffer files. |
| `MinimumDiskFreeBufferSize` | The minimum size (in MB) of disk buffer for buffering outgoing data. |
| `MaximumMemoryBufferSize` | The maximum size of memory buffer (in MB) for buffering outgoing data. |
| `ShouldAdjustTime` | If the data source supplies the timestamps, the `ShouldAdjustTime` value is False. If the collector supplies the timestamps, the `ShouldAdjustTime` value is True. |
| `ShouldQueueWrites` | Indicates whether queue writes are allowed or not. |

| Name | Description |
|---|---|
| `SourceTimeInLocalTime` | Applicable only for data source timestamps. Indicates whether the timestamps use local time. If `SourceTime-InLocalTime` is False, then UTC time is used. |
| `CollectionDelay` | The length of time, in seconds, that the collector should delay collection at start-up (to allow data source time to initialize). |
| `DefaultTagPrefix[200]` | The prefix that is automatically added to all tag names added by the specified collector. |
| `DefaultCollectionInterval` | The collection interval, in milliseconds, for tags added by the collector. |
| `ihTKCollectionType`<br><br>`DefaultCollectionType` | Type of collection used to acquire data for tags added by the collector:<br><br>• Unsolicited - The collector accepts data from the source whenever the source presents the data.<br>• Polled - The collector acquires data from a source on a periodic schedule determined by the collector. |
| `ihTKTimeStampType`<br><br>`DefaultTimeStampType` | The type of time stamping applied to data samples at collection time for tags added by the collector:<br><br>• Source - The source delivers the timestamp along with the data sample.<br>• Collector - The collector delivers the timestamp along with the collected data. |
| `DefaultCollectorCompression` | Indicates whether the default collector compression is enabled for tags added by the collector. |
| `DefaultCollectorCompression-Deadband` | The default collector compression deadband for tags added by the collector. |
| `DefaultCollectorCompression-Timeout` | Indicates the default collector compression time out value. |
| `DisableOnTheFlyTagChanges` | Indicates whether a user can perform on the fly changes to this tag. When enabled (True), you can |

| Name | Description |
|---|---|
| | make changes to this tag without having to restart the collector.<br><br>If this option is disabled (False), any changes you make to this tag does not affect collection until you restart the collector. |
| `DefaultSpikeLogic` | Indicates whether the Spike Logic is enabled. |
| `DefaultSpikeMultiplier` | Indicates the default Spike Logic multiplier. |
| `DefaultSpikeInterval` | Indicates the default Spike Logic interval. |
| `DataRecoveryQueueEnabled` | Indicates whether the Recovery queue is enabled. |
| `DefaultAbsoluteDeadbanding` | Indicates if the absolute deadband is enabled. |
| `RedundancyEnabled` | Indicates that the collector redundancy is enabled. |
| `RedundancyPrincipalCollec-tor[500]` | Indicates the principal collector. |
| `RedundancyIsActiveCollector` | Indicates that the current collector is active. |
| `RedundancyPrimaryCollec-tor[500]` | Indicates the principal collector. |
| `SyncThreadInterval` | If greater than 0, indicates that the collector is capable of synchronizing `EnumeratedSets` populated by `ihCol-lectorToolkitGetEnumeratedSets` implementation.<br><br>`ihCollectorToolkitGetEnumeratedSets` is fired every *x* minutes, where *x* is the value of `SyncThreadInterval`. |

### ihCollectorToolKitDefProperties

The `ihCollectorToolkitDefProperties` structure maintains the Custom Collectors specific properties General1-5 definitions.

#### Definition

```
typedef struct {
wchar_t* InterfaceDefName;
int32_t InterfaceType;
unsigned char IsSystemInterface;
```

```
wchar_t* General1Description;
wchar_t* General2Description;
wchar_t* General3Description;
wchar_t* General4Description;
wchar_t* General5Description;
} ihCollectorToolkitDefProperties;
```

**Parameters**

| Name | Description |
|---|---|
| `InterfaceDefName` | The name of the Custom Collector. |
| `InterfaceType` | `ihTKCustom`- specifies the custom collector type. |
| `IsSystemInterface` | Boolean. If true, specifies whether it is a System collector or a custom collector. Default - false. |
| `General1Description` | Collector general-1 property description. |
| `General2Description` | Collector general-2 property description. |
| `General3Description` | Collector general-3 property description. |
| `General4Description` | Collector general-4 property description. |
| `General5Description` | Collector general-5 property description. |

The parameters `General1Description`, `General2Description` to `General5Description`, specifies the collector specific properties descriptions which are not listed among the general collector properties. Using Collector Toolkit we can add up to 5 collector specific properties descriptions.

### ihInterfaceTKDataInfo

The `ihInterfaceTKDataInfo` structure maintains data properties of a collector.

**Definition**

```
typedef struct ihInterfaceTKDataInfo {
wchar_t
*ArchiveTagName;
wchar_t *Tag;
```

```
ihTKDataProperties DataProp;

ihTKAPIStatus ErrorStatus;

ihTKGetDataType GetDataType;

wchar_t *ErrorMessage;

uint32_t CollectionTime;

wchar_t *CustomProp1;

wchar_t *CustomProp2;

wchar_t *CustomProp3;

wchar_t *CustomProp4;

wchar_t *CustomProp5;

} ihInterfaceTKDataInfo;
```

**Parameters**

| Name | Description |
|---|---|
| `*ArchiveTagName` | Tag name of the archive. |
| `*Tag` | Name of the tag. |
| `DataProp` | Properties of the data. |
| `ErrorStatus` | Status of the error. |
| `GetDataType` | Data type. |
| `*ErrorMessage` | Error message. |
| `CollectionTime` | Collection time. |
| `*CustomProp1` | `CustomProp1` property description. |
| `*CustomProp2` | `CustomProp2` property description. |
| `*CustomProp3` | `CustomProp3` property description. |
| `*CustomProp4` | `CustomProp4` property description. |
| `*CustomProp5` | `CustomProp5` property description. |

**Example**

Populating data for a single tag.

```
ihInterfaceTKDataInfodata;

memset(&data,0, sizeof(ihInterfaceTKDataInfo));

data.Tag= L”computername.OperatorName”
```

```
data.DataProp.ValueDataType= ihTKFloat;// correct data type- same as tag type

data.DataProp.TimeStamp = pColl->TKGetSystemTime(); //10/01/2014 07:05:00

data.DataProp.Value = 10; // value that matches the data type.

data.DataProp.Quality = ihTKOPCGood;
```

### ihTKTagProperties

The `ihTKTagProperties` structure maintains the properties of a Historian tag.

#### Definition

```
typedef struct ihTKTagProperties {wchar_t* Tagname; wchar_t*

Description; wchar_t*

EngineeringUnits; wchar_t*

Comment; ihTKDataType

DataType;

unsigned char FixedStringLength;

wchar_t* InterfaceName; wchar_t*

SourceAddress; ihTKCollectionType

CollectionType; uint32_t

CollectionInterval; uint32_t

CollectionOffset; ihTKBoolean

LoadBalancing; ihTKTimeStampType

TimeStampType; double

HiEngineeringUnits;

double LoEngineeringUnits;

ihTKBoolean InputScaling;

double HiScale;

double LoScale;

ihTKBoolean InterfaceCompression;

float InterfaceDeadbandPercentRange;

ihTKBoolean ArchiveCompression;

float ArchiveDeadbandPercentRange;

wchar_t* CustomProp1;

wchar_t* CustomProp2;

wchar_t* CustomProp3;

wchar_t* CustomProp4;

wchar_t* CustomProp5;

wchar_t* ReadSecurityGroup;
```

```
wchar_t* WriteSecurityGroup;

wchar_t* AdministratorSecurityGroup;

ihTKTimeStruct LastModified;

wchar_t* LastModifiedUser;

ihTKInterfaceType InterfaceType;

ihTKBoolean ObsoleteField;

int32_t UTCBias;

uint32_t NumberOfCalculationDependencies;

wchar_t** CalculationDependencies;

uint32_t AverageCollectionTime;

ihTKBoolean CollectionDisabled;

uint32_t ArchiveCompressionTimeout;

uint32_t InterfaceCompressionTimeout;

ihTKBoolean SpikeLogic;

ihTKBoolean SpikeLogicOverride;

ihTKBoolean InterfaceAbsoluteDeadbanding;

double InterfaceAbsoluteDeadband;

ihTKBoolean ArchiveAbsoluteDeadbanding;

double ArchiveAbsoluteDeadband;

ihTKBoolean StepValue;

ihTKTimeResolution TimeResolution;

ihTKBoolean ConditionCollectionEnabled;

wchar_t* ConditionCollectionTriggerTag;

ihTKConditionCollectionComparison ConditionCollectionComparison;

wchar_t* ConditionCollectionCompareValue;

ihTKBoolean ConditionCollectionMarkers;

ihTKTagId TagId;

} ihTKTagProperties;
```

### Parameters

| Name | Description |
|---|---|
| `Tagname` | Tagname property of the tag. |
| `Description` | User description of the tag. |
| `EngineeringUnits` | Engineering units description of the tag. |

| Name | Description |
|---|---|
| `Comment` | User comment associated with the selected tag. |
| `DataType` | The data type returned in this column is the data type that is defined in Historian Administrator application. |
| `FixedStringLength` | Zero unless the data type is `FixedString`. If the data type is `FixedString`, this number represents the maximum length of the string value. |
| `InterfaceName` | Name of the collector responsible for collecting data for the specified tag. |
| `SourceAddress` | Address used to identify the tag at the data source. For `iFIXsystems`, this is `NTF` (`Node.Tag.Field`). |
| `CollectionType` | Types of collection used to acquire data for the tag. |
| `CollectionInterval` | The time interval, in milliseconds, between readings of data from this tag. |
| `CollectionOffset` | The time shift from midnight, in milliseconds, for collection of data from this tag. |
| `LoadBalancing` | Indicates whether the data collector should automatically shift the phase of sampling to distribute the activity of the processor evenly over the polling cycle. This is sometimes called Phase Shifting. |
| `TimeStampType` | The type of time stamping applied to data samples at collection time: |

| Name | Description |
|---|---|
| | • Source: The source delivers the time-stamp along with the data sample.<br>• Collector: The collector delivers the timestamp along with the collected data. |
| `HiEngineeringUnits` | The high end of the engineering units range. Used only for scaled data types and input scaled tags. |
| `LoEngineeringUnits` | The low end of the engineering units range. Used only for scaled data types and input scaled tags. |
| `InputScaling` | Indicates whether the measurement should be converted to an engineering unit's value. When set to False, the measurement is interpreted as a raw measurement.<br><br>With input scaling set to True, the system converts the value to engineering units by scaling the value between the Hi and Lo Scale. If `InputScaling` is not enabled, the system assumes that the measurement is already converted into engineering units. |
| `HiScale` | The high end value of the input scaling range used for the tag. |
| `LoScale` | The low end value of the input scaling range used for the tag. |
| `InterfaceCompression` | Indicates whether collector compression is enabled for the tag.<br><br>Collector compression applies a smoothing filter to incoming data by ignoring incremental changes in values that fall within a deadband centered around the last collected value. The collector passes (to the archiver) |

| Name | Description |
|---|---|
| | any new value that falls outside the deadband and then centers the deadband around the new value. |
| `InterfaceDeadbandPercentRange` | The current value of the compression deadband. |
| `ArchiveCompression` | Indicates whether the archive collector compression is enabled for the tag. |
| `ArchiveDeadbandPercentRange` | The current value of the archive compression deadband. |
| `CustomProp1` | The general (or spare) configuration fields for the collector. The `CustomProp1` column is not user-defined; it is different for each collector. |
| `CustomProp2` | The general (or spare) configuration fields for the collector. The `CustomProp2` column is not user-defined; it is different for each collector. |
| `CustomProp3` | The general (or spare) configuration fields for the collector. The `CustomProp3` column is not user-defined; it is different for each collector. |
| `CustomProp4` | The general (or spare) configuration fields for the collector. The `CustomProp4` column is not user-defined; it is different for each collector. |
| `CustomProp5` | The general (or spare) configuration fields for the collector. The `CustomProp5` column is not user-defined; it is different for each collector. |
| `ReadSecurityGroup` | The name of the Windows security group controlling the reading of data for the tag. |

| Name | Description |
|---|---|
| `WriteSecurityGroup` | The name of the Windows security group controlling the writing of data for the tag. |
| `AdministratorSecurityGroup` | The name of the Windows security group responsible for controlling configuration changes for the tag. |
| `LastModified` | The date and time that the tag configuration for this tag was last modified. The time structure includes milliseconds. |
| `LastModifiedUser` | The user name of the Windows user who last modified this tag's configuration. |
| `InterfaceType` | The collector responsible for collecting data for the tag. |
| `ObsoleteField` | |
| `UTCBias` | The time zone bias for the tag. Time zone bias is used to indicate the natural time zone of the tag expressed as an offset from UTC (Universal Time Coordinated) in minutes.<br><br>UTC is the international time standard, the current term for what was commonly referred to as Greenwich Mean Time (GMT). |
| `ArchiveCompressionTimeout` | Indicates the maximum amount of time from the last stored point before another point is stored, if the value does not exceed the archive compression deadband.<br><br>The data archiver treats the incoming sample after the timeout occurs as if it exceeded compression. It then stores the pending sample. |

| Name | Description |
|---|---|
| `InterfaceCompressionTimeout` | Indicates the maximum amount of time the collector will wait between sending samples to the archiver. |
| `SpikeLogic` | Indicates whether the Spike Logic is enabled for the tag. |
| `SpikeLogicOverride` | Indicates whether the Spike Logic setting for this tag overrides the collector. |
| `InterfaceAbsoluteDeadbanding` | Indicates if absolute collector deadband is enabled for this tag. |
| `InterfaceAbsoluteDeadband` | Indicates the value for absolute collector deadband. |
| `ArchiveAbsoluteDeadbanding` | Indicates if absolute archive deadband is enabled for this tag. |
| `ArchiveAbsoluteDeadband` | Indicates the value for absolute archive deadband. |
| `StepValue` | Indicates if the `StepValue` property is enabled for the tag. |
| `TimeResolution` | Indicates the timestamp resolution in seconds, milliseconds, or microseconds. |
| `ConditionCollectionEnabled` | Indicates whether condition-based collection is enabled. |
| `ConditionCollectionTriggerTag` | Tag condition-based collection trigger tag. |
| `ConditionCollectionComparison` | Tag condition-based collection comparison operator. |
| `ConditionCollectionCompareValue` | Tag condition-based collection compare value. |
| `ConditionCollectionMarkers` | Indicates whether to employ condition-based collection markers. |
| `TagId` | `TagID` associated with this tag. |

### ihTKTagFields

The `ihTKTagFields` structure maintains the information that should be provided to Historian while creating tags.

#### Definition

```
typedef struct ihTKTagFields {
ihTKBoolean    AllFields;
ihTKBoolean    Tagname;
ihTKBoolean    Description;
ihTKBoolean    EngineeringUnits;
ihTKBoolean    Comment;
ihTKBoolean    DataType;
ihTKBoolean    FixedStringLength;
ihTKBoolean    InterfaceName;
ihTKBoolean    SourceAddress;
ihTKBoolean    CollectionType;
ihTKBoolean    CollectionInterval;
ihTKBoolean    CollectionOffset;
ihTKBoolean    LoadBalancing;
ihTKBoolean    TimeStampType;
ihTKBoolean    HiEngineeringUnits;
ihTKBoolean    LoEngineeringUnits;
ihTKBoolean    InputScaling;
ihTKBoolean    HiScale;
ihTKBoolean    LoScale;
ihTKBoolean    InterfaceCompression;
ihTKBoolean    InterfaceDeadbandPercentRange;
ihTKBoolean    ArchiveCompression;
ihTKBoolean    ArchiveDeadbandPercentRange;
ihTKBoolean    CustomProp1;
ihTKBoolean    CustomProp2;
ihTKBoolean    CustomProp3;
ihTKBoolean    CustomProp4;
ihTKBoolean    CustomProp5;
ihTKBoolean    ReadSecurityGroup;
ihTKBoolean    WriteSecurityGroup;
ihTKBoolean    AdministratorSecurityGroup;
```

```
ihTKBoolean    ObsoleteField;
ihTKBoolean    LastModified;
ihTKBoolean    LastModifiedUser;
ihTKBoolean    InterfaceType;
ihTKBoolean    TimeResolution;
ihTKBoolean    UTCBias;
ihTKBoolean    AverageCollectionTime;
ihTKBoolean    CalculationDependencies;
ihTKBoolean    CollectionDisabled;
ihTKBoolean    ArchiveCompressionTimeout;
ihTKBoolean    InterfaceCompressionTimeout;
ihTKBoolean    SpikeLogic;
ihTKBoolean    SpikeLogicOverride;
ihTKBoolean    InterfaceAbsoluteDeadbanding;
ihTKBoolean    InterfaceAbsoluteDeadband;
ihTKBoolean    ArchiveAbsoluteDeadbanding;
ihTKBoolean    ArchiveAbsoluteDeadband;
ihTKBoolean    StepValue;
ihTKBoolean    MaxTagsToRetrieve;
ihTKBoolean    ConditionCollectionEnabled;
ihTKBoolean    ConditionCollectionTriggerTag;
ihTKBoolean    ConditionCollectionComparison;
ihTKBoolean    ConditionCollectionCompareValue;
ihTKBoolean    ConditionCollectionMarkers;
ihTKBoolean    TagId;
} ihTKTagFields;
```

### Parameters

| Name | Description |
|---|---|
| `AllFields` | Tag request contains all fields. |
| `Tagname` | Tag request contains `Tagname`. |
| `Description` | Tag request contains `Description`. |
| `EngineeringUnits` | Tag request contains `EngineeringUnits`. |
| `Comment` | Tag request contains `Comment`. |
| `DataType` | Tag request contains `DataType`. |

| Name | Description |
|---|---|
| `FixedStringLength` | Tag request contains `FixedStringLength`. |
| `InterfaceName` | Tag request contains `InterfaceName`. |
| `SourceAddress` | Tag request contains `SourceAddress`. |
| `CollectionType` | Tag request contains `CollectionType`. |
| `CollectionInterval` | Tag request contains `CollectionInterval`. |
| `CollectionOffset` | Tag request contains `CollectionOffset`. |
| `LoadBalancing` | Tag request contains `LoadBalancing`. |
| `TimeStampType` | Tag request contains `TimeStampType`. |
| `HiEngineeringUnits` | Tag request contains `HiEngineeringUnits`. |
| `LoEngineeringUnits` | Tag request contains `LoEngineeringUnits`. |
| `InputScaling` | Tag request contains `InputScaling`. |
| `HiScale` | Tag request contains `HiScale`. |
| `LoScale` | Tag request contains `LoScale`. |
| `InterfaceCompression` | Tag request contains `InterfaceCompression`. |
| `InterfaceDeadbandPercentRange` | Tag request contains `InterfaceDeadbandPer-centRange`. |
| `ArchiveCompression` | Tag request contains `ArchiveCompression`. |
| `ArchiveDeadbandPercentRange` | Tag request contains `ArchiveDeadbandPer-centRange`. |
| `CustomProp1` | Tag request contains `CustomProp1`. |
| `CustomProp2` | Tag request contains `CustomProp2`. |
| `CustomProp3` | Tag request contains `CustomProp3`. |
| `CustomProp4` | Tag request contains `CustomProp4`. |
| `CustomProp5` | Tag request contains `CustomProp5`. |
| `ReadSecurityGroup` | Tag request contains `ReadSecurityGroup`. |
| `WriteSecurityGroup` | Tag request contains `WriteSecurityGroup`. |

| Name | Description |
|---|---|
| `AdministratorSecurityGroup` | Tag request contains `AdministratorSecurityGroup`. |
| `ObsoleteField` | Tag request contains `ObsoleteField`. |
| `LastModified` | Tag request contains `LastModified`. |
| `LastModifiedUser` | Tag request contains `LastModifiedUser`. |
| `InterfaceType` | Tag request contains `InterfaceType`. |
| `TimeResolution` | Tag request contains `TimeResolution`. |
| `UTCBias` | Tag request contains `UTCBias`. |
| `AverageCollectionTime` | Tag request contains `AverageCollectionTime`. |
| `CalculationDependencies` | Tag request contains `CalculationDependencies`. |
| `CollectionDisabled` | Tag request contains `CollectionDisabled`. |
| `ArchiveCompressionTimeout` | Tag request contains `ArchiveCompressionTimeout`. |
| `SpikeLogic` | Tag request contains `SpikeLogic`. |
| `SpikeLogicOverride` | Tag request contains `SpikeLogicOverride`. |
| `InterfaceAbsoluteDeadbanding` | Tag request contains `InterfaceAbsoluteDeadbanding`. |
| `InterfaceAbsoluteDeadband` | Tag request contains `InterfaceAbsoluteDeadband`. |
| `ArchiveAbsoluteDeadbanding` | Tag request contains `ArchiveAbsoluteDeadbanding`. |
| `ArchiveAbsoluteDeadband` | Tag request contains `ArchiveAbsoluteDeadband`. |
| `StepValue` | Tag request contains `StepValue`. |
| `MaxTagsToRetrieve` | Tag request contains `MaxTagsToRetrieve`. |

| Name | Description |
|---|---|
| `ConditionCollectionEnabled` | Tag request contains `ConditionCollection-Enabled`. |
| `ConditionCollectionTriggerTag` | Tag request contains `ConditionCollection-TriggerTag`. |
| `ConditionCollectionComparison` | Tag request contains `ConditionCollection-Comparison`. |
| `ConditionCollectionCompareValue` | Tag request contains `ConditionCollection-CompareValue`. |
| `ConditionCollectionMarkers` | Tag request contains `ConditionCollection-Markers`. |
| `TagId` | Tag request contains `TagId` |

### ihTKTagCriteria

The `ihTKTagCriteria` structure maintains the information required while retrieving data from Historian.

#### Definition

```
typedef struct ihTKTagCriteria {
wchar_t* TagnameMask;
uint32_t NumberOfTags;
wchar_t** TagnameArray;
wchar_t* DescriptionMask;
wchar_t* EngineeringUnits;
wchar_t* Comment;
ihTKDataType DataType;
unsigned char FixedStringLength;
wchar_t* InterfaceName;
wchar_t* SourceAddress;
ihTKCollectionType CollectionType;
uint32_t CollectionInterval;
uint32_t CollectionOffset;
ihTKBoolean  LoadBalancing;
ihTKTimeStampType TimeStampType;
double HiEngineeringUnits;
double LoEngineeringUnits;
```

```
ihTKBoolean InputScaling;

double HiScale;

double LoScale;

ihTKBoolean InterfaceCompression;

float InterfaceDeadbandPercentRange;

ihTKBoolean ArchiveCompression;

float ArchiveDeadbandPercentRange;

wchar_t* CustomProp1;

wchar_t* CustomProp2;

wchar_t* CustomProp3;

wchar_t* CustomProp4;

wchar_t* CustomProp5;

wchar_t* ReadSecurityGroup;

wchar_t* WriteSecurityGroup;

wchar_t* AdministratorSecurityGroup;

ihTKTimeStruct LastModified;

wchar_t* LastModifiedUser;

ihTKInterfaceType InterfaceType;

ihTKBoolean ObsoleteField;

int32_t UTCBias;

uint32_t AverageCollectionTime;

ihTKBoolean CollectionDisabled;

uint32_t ArchiveCompressionTimeout;

uint32_t InterfaceCompressionTimeout;

ihTKBoolean SpikeLogic;

ihTKBoolean SpikeLogicOverride;

ihTKBoolean InterfaceAbsoluteDeadbanding;

double InterfaceAbsoluteDeadband;

ihTKBoolean ArchiveAbsoluteDeadbanding;

double ArchiveAbsoluteDeadband;

ihTKBoolean SourceAddressIsMask;

ihTKBoolean StepValue;

int32_t MaxTagsToRetrieve;

ihTKTimeResolution TimeResolution;

ihTKBoolean ConditionCollectionEnabled;

wchar_t* ConditionCollectionTriggerTag;

ihTKConditionCollectionComparison ConditionCollectionComparison;
```

```
wchar_t* ConditionCollectionCompareValue;

ihTKBoolean ConditionCollectionMarkers;

ihTKTagId TagId;

} ihTKTagCriteria;
```

**Parameters**

| Name | Description |
|---|---|
| `TagnameMask` | Tag name mask criteria. |
| `NumberOfTags` | Number of tags for a tag. |
| `TagnameArray` | Tag name array of a tag. |
| `DescriptionMask` | Tag description mask criteria. |
| `EngineeringUnits` | Tag engineering units criteria. |
| `Comment` | Tag comment criteria. |
| `DataType` | Tag data type criteria. |
| `FixedStringLength` | Tag fixed string length criteria. |
| `InterfaceName` | Tag interface name. |
| `SourceAddress` | Tag source address criteria. |
| `CollectionType` | Tag collection type criteria. |
| `CollectionInterval` | Tag collection interval criteria. |
| `CollectionOffset` | Tag collection offset criteria. |
| `LoadBalancing` | Tag load balancing enabled criteria. |
| `TimeStampType` | Tag timestamp type criteria. |
| `HiEngineeringUnits` | Tag high engineering unit criteria. |
| `LoEngineeringUnits` | Tag low engineering unit criteria. |
| `InputScaling` | Tag input scaling enabled criteria. |
| `HiScale` | Tag high scale value criteria. |
| `LoScale` | Tag low scale value criteria. |
| `InterfaceCompression` | Tag interface compression. |

| Name | Description |
|---|---|
| `InterfaceDeadbandPercentRange` | Tag interface deadband percentage range. |
| `ArchiveCompression` | Tag archive compression enabled criteria. |
| `CustomProp1` | Tag custom property 1. |
| `CustomProp2` | Tag custom property 2. |
| `CustomProp3` | Tag custom property 3. |
| `CustomProp4` | Tag custom property 4. |
| `CustomProp5` | Tag custom property 5. |
| `ReadSecurityGroup` | Tag read security group name criteria. |
| `WriteSecurityGroup` | Tag write security group name criteria. |
| `AdministratorSecurityGroup` | Tag administrator security group name criteria. |
| `LastModified` | Tag last modified time criteria. |
| `LastModifiedUser` | Tag last modified user name criteria. |
| `InterfaceType` | Tag interface type. |
| `ObsoleteField` | Tag obsolete field. |
| `UTCBias` | Tag UTC bias criteria. |
| `AverageCollectionTime` | Tag average collection time. |
| `CollectionDisabled` | Tag collection disabled criteria. |
| `ArchiveCompressionTimeout` | Tag archive-compression time out criteria. |
| `InterfaceCompressionTimeout` | Tag interface compression time out. |
| `SpikeLogic` | Tag spike logic enabled criteria. |
| `SpikeLogicOverride` | Tag spike logic override enabled criteria. |
| `InterfaceAbsoluteDeadbanding` | Tag interface absolute deadband. |
| `ArchiveAbsoluteDeadbanding` | Tag archive compression absolute deadband criteria. |

| Name | Description |
|---|---|
| `SourceAddressIsMask` | Whether to interpret source address criteria as a wildcard expression. |
| `StepValue` | Tag step value enabled criteria. |
| `MaxTagsToRetrieve` | Maximum number of tags to retrieve per query. |
| `TimeResolution` | Tag time resolution criteria. |
| `ConditionCollectionEnabled` | Tag condition-based collection enabled criteria. |
| `ConditionCollectionTriggerTag` | Tag condition-based collection trigger tag criteria. |
| `ConditionCollectionComparison` | Tag condition-based collection comparison operator criteria. |
| `ConditionCollectionCompareValue` | Tag condition-based collection compare value criteria. |
| `ConditionCollectionMarkers` | Tag condition-based collection markers enabled criteria. |
| `TagId` | TagId criteria. |

### ihTKDataType

The `ihTKDataType` structure contains the data types supported by Historian.

#### Definition

```
typedefenum ihTKDataType {

ihTKDataTypeUndefined = 0,

ihTKScaled,

ihTKFloat,

ihTKDoubleFloat,

ihTKInteger,

ihTKDoubleInteger,

ihTKFixedString,

ihTKVariableString,

ihTKBlob,ihTKTime,
```

```
ihTKInt64,

ihTKUInt64,

ihTKUInt32,

ihTKUInt16,

ihTKMaxDataType}ihTKDataType;
```

**Parameters**

| Name | Description |
|---|---|
| `ihTKDataTypeUndefined` | An Undefined data type. |
| `ihTKScaled` | A single precision (32-bit) floating-point type. |
| `ihTKFloat` | A single precision (32-bit) floating-point type. |
| `ihTKDoubleFloat` | A double precision (64-bit) floating-point type. |
| `ihTKInteger` | A short, signed integral type (16-bit). |
| `ihTKDoubleInteger` | A long, signed integral type (32-bit). |
| `ihTKFixedString` | A fixed-length UNICODE string The length is determined by `Tag.FixedStringLengthtag property`. |
| `ihTKVariableString` | A variable-length UNICODE string. |
| `ihTKBlob` | An unstructured, binary data type. |
| `ihTKTime` | A date-time type (64-bit), capable of storing one DateTime instance value. |
| `ihTKInt64` | A __int64-bit quad integer. |
| `ihTKUInt64` | A __int64-bit unsigned quad integer. |
| `ihTKUInt32` | A long, unsigned integral type (32-bit). |
| `ihTKUInt16` | A short, unsigned integral type (16-bit). |
| `ihTKMaxDataType` | A max data type. |

**ihInterfaceTKASyncTagInfo**

The `ihInterfaceTKASyncTagInfo` structure contains unsolicited tag information.

### Definition

```
typedef struct ihInterfaceTKASyncTagInfo {
int TagId;
wchar_t *ArchiveTagName;
wchar_t *Tag;
int Interval;
double DeadbandPct;
ihTKDataType DataType;
int UseSourceTimeStamp;
wchar_t *CustomProp1;
wchar_t *CustomProp2;
wchar_t *CustomProp3;
wchar_t *CustomProp4;
wchar_t *CustomProp5;
} ihInterfaceTKASyncTagInfo;
```

### Parameters

| Name | Description |
|---|---|
| `TagId` | TagId of the tag to be stored. |
| `*ArchiveTagName` | Tag name of the corresponding TagId stored in the archiver. |
| `*Tag` | Tag name used as primary key. |
| `Interval` | Collection interval of the tag. |
| `DeadbandPct` | Deadband percentage used for compression. |
| `DataType` | Datatype of the tag. |
| `UseSourceTimeStamp` | Enables you to specify to use source time stamp or collector timestamp. |
| `*CustomProp1` | The general (or spare) configuration fields for the collector. The CustomProp1 column is not user-defined; it is different for each collector. |

| Name | Description |
|---|---|
| `*CustomProp2` | The general (or spare) configuration fields for the collector. The CustomProp2 column is not user-defined; it is different for each collector. |
| `*CustomProp3` | The general (or spare) configuration fields for the collector. The CustomProp3 column is not user-defined; it is different for each collector. |
| `*CustomProp4` | The general (or spare) configuration fields for the collector. The CustomProp4 column is not user-defined; it is different for each collector. |
| `*CustomProp5` | The general (or spare) configuration fields for the collector. The CustomProp5 column is not user-defined; it is different for each collector. |

### ihInterfaceTKPolledTagInfo

The `ihInterfaceTKPolledTagInfo` structure maintains the polled tag information of the collectors.

#### Definition

```
typedef struct ihInterfaceTKPolledTagInfo {
int TagId;
wchar_t *ArchiveTagName;
wchar_t *Tag;
int Interval;
int Offset;
double DeadbandPct;
ihTKDataType DataType;
int UseSourceTimeStamp;
wchar_t *CustomProp1;
wchar_t *CustomProp2;
wchar_t *CustomProp3;
wchar_t *CustomProp4;
```

```
wchar_t *CustomProp5;

} ihInterfaceTKPolledTagInfo;
```

**Parameters**

| Name | Description |
|---|---|
| `TagId` | TagId of the tag to be stored. |
| `*ArchiveTagName` | Tag name of the corresponding tag ID stored in the archiver. |
| `*Tag` | Tag name used as primary key. |
| `Interval` | Collection interval of the tag. |
| `Offset` | Offset from midnight (in seconds) to force sampling at a specific time of day. |
| `DeadbandPercentage` | Deadband percentage used for compression. |
| `ihTKDataType DataType` | Datatype of the tag. |
| `UseSourceTimeStamp` | Enables you to specify to use source time stamp or collector timestamp. |
| `*CustomProp1` | The general (or spare) configuration fields for the collector. The CollectorGeneral1 column is not user-defined; it is different for each collector. |
| `*CustomProp2` | The general (or spare) configuration fields for the collector. The CollectorGeneral2 column is not user-defined; it is different for each collector. |
| `*CustomProp3` | The general (or spare) configuration fields for the collector. The CollectorGeneral3 column is not user-defined; it is different for each collector. |
| `*CustomProp4` | The general (or spare) configuration fields for the collector. The CollectorGeneral4 col- |

| Name | Description |
|---|---|
| | umn is not user-defined; it is different for each collector. |
| `*CustomProp5` | The general (or spare) configuration fields for the collector. The CollectorGeneral5 column is not user-defined; it is different for each collector. |

### ihTKTagRecordset

The `ihTKTagRecordset` structure maintains Historian tags' record set.

#### Definition

```
typedef struct ihTKTagRecordset {
ihTKTagFields RequestedFields;
ihTKTagCriteria Criteria;
ihTKTagFields CriteriaFields;
uint32_t NumberOfRecords;
ihTKTagPropertiesPtr TagRecords;
} ihTKTagRecordset;
```

#### Parameters

| Name | Description |
|---|---|
| `RequestedFields` | Tag fields structure, which can be used to retrieve the specific fields of tags. |
| `Criteria` | Criteria structure, which can be used to query the tags based on the fields that are set. |
| `CriteriaFields` | Tag fields structure that need to be populated based on query criteria. For example, if user wants to query the tags based on Tag names then `CriteriaFields.Tagname` must set to true. |
| `NumberOfRecords` | Number of `TagRecords`. |

| Name | Description |
|---|---|
| `TagRecords` | Array of `TagRecords`. Each `TagRecord` is a tag property of each tag. |

### ihTKBlobData

The `ihTKBlobData` structure contains some memory that can support data in any format such as XML or CSV.

#### Definition

```
typedefstruct ihTKBlobData {
void*Blob;
uint32_tBlobSize;
}ihTKBlobData;
```

#### Parameters

| Name | Description |
|---|---|
| `Blob` | Blob data. |
| `BlobSize` | Blob size. |

### ihTKHiddenValue

The `ihTKHiddenValue` structure represents the mapping of supported data types Historian and Microsoft, the operating system.

#### Definition

```
typedef union ihTKHiddenValue {
short Integer;
int32_t DoubleInteger;
float Float;
double DoubleFloat;
ihTKBlobData Blob;
wchar_t* String;
ihTKTimeStruct Time;
int64_t Int64;
uint64_t UInt64;
uint32_t UInt32;
uint16_t UInt16;
```

```
} ihTKHiddenValue;
```

**Parameters**

| Name | Description |
|---|---|
| `Integer` | Single integer. |
| `DoubleInteger` | Double integer. |
| `Float` | Float. |
| `DoubleFloat` | Double float. |
| `Blob` | Blob. |
| `String` | String. |
| `Time` | Timestamp structure with seconds and nanoseconds. |
| `Int64` | Quad integer. |
| `UInt64` | Unsigned quad integer. |
| `UInt32` | Unsigned double integer. |
| `UInt16` | Unsigned short. |

### ihTKRawQuality

The `ihTKRawQuality` structure maintains the raw quality types of the tag data.

**Definition**

```
typedef struct ihTKRawQuality {
ihTKBoolean Deleted;
ihTKBoolean Replaced;
unsigned char QualityStatus;
unsigned char QualitySubStatus;
ihTKBoolean OPCQualityValid;
unsigned short OPCQuality;
} ihTKRawQuality;
```

**Parameters**

| Name | Description |
|---|---|
| `Deleted` | Deleted tag data. |
| `Replaced` | Replaced tag data. |
| `QualityStatus` | QualityStatus of the tag data. |
| `QualitySubStatus` | QualitySubStatus of the tag data. |
| `OPCQualityValid` | OPCQualityValid. |
| `OPCQuality` | OPCQuality. |

### ihTKQuality

The `ihTKQuality` structures determines the percentage of good data in the raw quality.

#### Definition

```
typedefunion ihTKQuality {
ihTKRawQuality RawQuality;
float PercentGood;
}ihTKQuality;
```

#### Parameters

| Name | Description |
|---|---|
| `ihTKRawQuality` | RawQuality. |
| `ihTKQuality` | PercentGood. |

### ihTKCommentData

The `ihTKCommentData` structure maintains the comment about Comment Data in the tag, if any.

#### Definition

```
typedefstruct ihTKCommentData {
wchar_t*DataTypeHint;
ihTKBlobDataCommentData;
}ihTKCommentData;
```

#### Parameters

| Name | Description |
|---|---|
| `DataTypeHint` | Data type. |
| `CommentData` | Comment data. |

### ihTKComments

The `ihTKComments` structure maintains the information about comment data.

#### Definition

```
typedef struct ihTKComments {
ihTKTimeStruct StoredOnTimeStamp;
ihTKTimeStruct CommentTimeStamp;
wchar_t* SuppliedUsername;
wchar_t* Username;
ihTKCommentData CommentData;
} ihTKComments;
```

#### Parameters

| Name | Description |
|---|---|
| `StoredOnTimeStamp` | Stored time (Set by Archiver). |
| `CommentTimeStamp` | Timestamp. |
| `SuppliedUsername` | Supplied username (optionally given in `ih-CommentAdd`). |
| `Username` | NT user name of writer. (Set by Archiver). |
| `CommentData` | Comment Data. |

### ihTKDataProperties

The `ihTKDataProperties` structure represents the properties of the data.

#### Definition

```
typedef struct ihTKDataProperties {
ihTKTimeStruct TimeStamp;
ihTKDataType ValueDataType;
ihTKValue Value;
ihTKQuality Quality;
```

```
unsigned char NumberOfComments;

ihTKCommentsPtr Comments;

} ihTKDataProperties;
```

**Parameters**

| Name | Description |
|---|---|
| `TimeStamp` | Time stamp of the data. |
| `ValueDataType` | Data type. |
| `Value` | Data value. |
| `Quality` | Quality. |
| `NumberOfComments` | Number of comments, if any. |
| `Comments` | Array of comments. |

### ihInterfaceTKASyncData

The `ihInterfaceTKASyncData` structure represents the unsolicited tags' data. `ihInterfaceTKASyncData` can be defined as an array of pointers if user want to send multiple data for multiple tags in one callback. This callback doesn't return errors, but will log any errors in writing to either the DA log or the collector specific log.

**Definition**

```
typedef struct ihInterfaceTKASyncData {

int NumValues; int *TagIds;

ihTKDataProperties *Values;

uint32_t *CollectionTimes;

} ihInterfaceTKASyncData;
```

**Parameters**

| Name | Description |
|---|---|
| `NumValues` | Number of data properties. |
| `*TagIds` | Tag ids. |
| `*Values` | Array of data properties. |
| `*CollectionTimes` | Collected Times. |

#### Example

Adding data to `ihInterfaceTKASyncData` structure.

```
uint32_tcollectionTime = (uint32_t)time(0);

int tagId = tag->TagId;

ihInterfaceTKASyncData asyncData;

memset(&asyncData,0, sizeof(ihInterfaceTKASyncData));

asyncData.NumValues= 1; asyncData.TagIds = &tagId;

asyncData.Values = &data.DataProp;

asyncData.CollectionTimes= &collectionTime;

pColl->ihCollectorToolkitDataCallback(&asyncData);
```

### ihTKGetDataType

The `ihTKGetDataType` structure indicates the different ways data is collected from source.

#### Definition

```
typedef enum ihTKGetDataType {

ihTKGetDataTimed = 0,

ihTKGetDataAsync,

ihTKGetDataDemand

} ihTKGetDataType;
```

#### Parameters

| Name | Description |
|---|---|
| `ihTKGetDataTimed` | Polled data. The collector acquires data from a source on a periodic schedule determined by the collector. |
| `ihTKGetDataAsync` | Asynchronous data, the collector accepts data from the source whenever the source presents the data. |
| `ihTKGetDataDemand` | Data on demand, the collector accepts data from source whenever there is a demand for data. |

### ihTKStatus

The `ihTKStatus` structure contains the error statuses of Historian.

### Definition

```
typedef enum ihTKStatus {

ihTKSTATUS_OK = 0,

ihTKSTATUS_FAILED = -1,

ihTKSTATUS_API_TIMEOUT = -2,

ihTKSTATUS_NOT_CONNECTED = -3,

ihTKSTATUS_INTERFACE_NOT_FOUND = -4,

ihTKSTATUS_NOT_SUPPORTED = -5,

ihTKSTATUS_DUPLICATE_DATA = -6,

ihTKSTATUS_NOT_VALID_USER = -7,

ihTKSTATUS_ACCESS_DENIED = -8,

ihTKSTATUS_WRITE_IN_FUTURE = -9,

ihTKSTATUS_WRITE_ARCH_OFFLINE = -10,

ihTKSTATUS_ARCH_READONLY = -11,

ihTKSTATUS_WRITE_OUTSIDE_ACTIVE = -12,

ihTKSTATUS_WRITE_NO_ARCH_AVAIL = -13,

ihTKSTATUS_INVALID_TAGNAME = -14,

ihTKSTATUS_LIC_TOO_MANY_TAGS = -15,

ihTKSTATUS_LIC_TOO_MANY_USERS = -16,

ihTKSTATUS_LIC_INVALID_LIC_DLL = -17,

ihTKSTATUS_NO_VALUE = -18,

ihTKSTATUS_DUPLICATE_INTERFACE = -19,

ihTKSTATUS_NOT_LICENSED = -20,

ihTKSTATUS_CALC_CIRC_REFERENCE = -21,

ihTKSTATUS_BACKUP_EXCEEDED_SPACE = -22,

ihTKSTATUS_INVALID_SERVER_VERSION = -23,

ihTKSTATUS_DATA_RETRIEVAL_COUNT_EXCEEDED= -24,

ihTKSTATUS_DELETEDATA_OUTSIDE_ACTIVE = -25,

ihTKSTATUS_ALARM_ARCHIVER_UNAVAILABLE  = -26,

ihTKSTATUS_ARGUMENT_INVALID = -27,

ihTKSTATUS_ARGUMENT_NULL = -28,

ihTKSTATUS_ARGUMENT_OUT_OF_RANGE = -29,

ihTKSTATUS_MAX_ERROR_NUM = -30,

} ihTKStatus;
```

### Parameters

| Name | Description |
|---|---|
| `ihTKSTATUS_FAILED` | Generic failure. |
| `ihTKSTATUS_API_TIMEOUT` | Server machine name not found, or server found but Archiver service is not running. |
| `ihTKSTATUS_NOT_ CONNECTED` | Not currently connected to a Historian server. |
| `ihTKSTATUS_INTERFACE_ NOT_FOUND` | Interface not found. |
| `ihTKSTATUS_NOT_SUPPORTED` | Reserved. |
| `ihTKSTATUS_DUPLICATE_DATA` | WriteData was called with `error_on_replace = TRUE` and the supplied data would have overwritten the existing data. |
| `ihTKSTATUS_NOT_VALID_USER` | Server found, but invalid username or password. |
| `ihTKSTATUS_ACCESS_DENIED` | Access is denied by the Historian server. Check user name/password or security group membership. |
| `ihTKSTATUS_WRITE_IN_FUTURE` | Write time stamp is too far in the future. |
| `ihTKSTATUS_WRITE_ARCH_OFFLINE` | There is no Archiver to hold the write time stamp. |
| `ihTKSTATUS_ARCH_READONLY` | The archive to hold the write time stamp is marked as read-only. |
| `ihTKSTATUS_WRITE_OUTSIDE_ACTIVE` | The write time stamp is before the active hours (now - "data is read only after") setting. |
| `ihTKSTATUS_WRITE_NO_ARCH_-AVAILABLE` | No archive is available to hold the write time stamp. |
| `ihTKSTATUS_INVALID_TAGNAME` | Tagname used is not valid. Tagname does not exist in the Historian server. |
| `ihTKSTATUS_LIC_TOO_MANY_TAGS` | Exceeded tag license count on the server. |
| `ihTKSTATUS_LIC_TOO_MANY_USERS` | Exceeded user license count on the server. |
| `ihTKSTATUS_LIC_INVALID_LIC_DLL` | An invalid license DLL is installed. |
| `ihTKSTATUS_NO_VALUE` | No value has been passed to the function. |
| `ihTKSTATUS_NOT_LICENSED` | Your installation of Historian is not licensed. |

| Name | Description |
|---|---|
| `ihTKSTATUS_CALC_CIRC_REFERENCE` | Reserved. |
| `ihTKSTATUS_DUPLICATE_INTERFACE` | Reserved. |
| `ihTKSTATUS_BACKUP_EXCEEDED_SPACE` | Reserved. |
| `ihTKSTATUS_INVALID_SERVER_VERSION` | You are attempting to use this API on an invalid version of Historian. |
| `ihTKSTATUS_DATA_RETRIEVAL_COUNT_EXCEEDED` | You tried to request too many samples in one read request. |
| `ihTKSTATUS_INVALID_PARAMETER` | Generic failure when an invalid value is passed into the user API. |

### ihTKMessageTopic

The `ihTKMessageTopic` structure posts the status messages of Historian.

#### Definition

```
typedef enum ihTKMessageTopic {
ihTKMessageTopicUndefined=0,
ihTKConnections,
ihTKConfigurationAudit,
ihTKGeneral,
ihTKServiceControl,
ihTKPerformance,
ihTKSecurity,
ihTKMessageTopicMax,
ihTKAllTopics=10000,
ihTKAlertTopics, ihTKMessageTopics,
} ihTKMessageTopic;
```

#### Parameters

| Name | Description |
|---|---|
| `ihTKConnections` | Connection related messages. |
| `ihTKConfigurationAudit` | Audit messages. |
| `ihTKGeneral` | General messages. |

| Name | Description |
|---|---|
| `ihTKServiceControl` | Service control messages. |
| `ihTKPerformance` | Performance related messages. |
| `ihTKSecurity` | Security related messages. |
| `ihTKMessageTopicMax` | Must be the last one after the basic topics and before the combinations. |
| `ihTKAlertTopics` | Any alerts. |

### ihTKCollectorCallbacks

The `ihTKCollectorCallbacks` structure controls the way the Async tag's functions are performed.

#### Definition

```
typedef struct ihTKCollectorCallbacks {
TKCallBackFunctionNoParams *ShutdownFunc;
TKCallBackFunctionOneParam *LogMsgFunc;
TKCallBackFunctionOneParam *DataFunc;
TKCallBackFunctionOneLongParam *AsyncOverrunFunc;
TKCallBackFunctionOneLongParam *ChangeInterfaceControlFunc;
TKCallBackFunctionTwoLongParams *ReconnectFunc;
TKCallBackFunctionMessageAdd *AddMessageCallback;
//CallbackFunctionAlarmNotification *AlarmNotificationFunc;
TKCallBackFunctionGetTimeOffset *GetTimeOffsetFunc;
} ihTKCollectorCallbacks;
```

#### Parameters

| Name | Description |
|---|---|
| `*ShutdownFunc` | When the Collector shuts down, this callback will be called from the toolkit. |
| `*LogMsgFunc` | When user wants to log any information, this call back is used. |
| `*DataFunc` | When data needs to be added, this call back is used. |
| `*AsyncOverrunFunc` | This callback is called for data overruns. |

| Name | Description |
|---|---|
| `*ReconnectFunc` | When user needs to reconnect to Historian, this call-back is used. |
| `*AddMessageCallback` | When user wants to add a message to Historian, this callback is used. |
| `*GetTimeOffsetFunc` | When time offsets need to be adjusted, this callback is used. |

### ihTKTimeStruct

The `ihTKTimeStruct` structure contains the time value in seconds and Nanoseconds.

#### Definition

```
typedef struct ihTKTimeStruct {
uint32_t Seconds;
uint32_t Nanoseconds;
} ihTKTimeStruct
```

#### Parameters

| Name | Description |
|---|---|
| `Seconds` | The time value in seconds. |
| `Nanoseconds` | The time value in nanoseconds. |

### ihTKQualityStatus

The `ihTKQualityStatus` structure defines the quality of the incoming data value that Historian stores.

#### Definition

```
typedef enum ihTKQualityStatus {
ihTKOPCBad = 0,
ihTKOPCUncertain,
ihTKOPCNA, ihTKOPCGood,
} ihTKQualityStatus;
```

#### Parameters

| Name | Description |
|---|---|
| `Bad` | The quality of the associated data value is bad. There is low or no confidence in the associated data value. |
| `Uncertain` | There is uncertainty about the associated data value. |
| `NA` | The associated data value is unused. |
| `Good` | The quality of the associated data value is good. |

### ihTKQualitySubStatus

The `ihTKQualitySubStatus` structure indicates the reasons for the quality of associated data value.

#### Definition

```
typedef enum ihTKQualitySubStatus {
ihTKOPCNonspecific = 0,
ihTKOPCConfigurationError,
ihTKOPCNotConnected,
ihTKOPCDeviceFailure,
ihTKOPCSensorFailure,
ihTKOPCLastKnownValue,
ihTKOPCCommFailure,
ihTKOPCOutOfService,
ihTKScaledOutOfRange,
ihTKOffLine, ihTKNoValue,
ihTKCalculationError,
ihTKConditionCollectionHalted,
ihTKCalculationTimeout
} ihTKQualitySubStatus;
```

#### Parameters

| Name | Description |
|---|---|
| `Nonspecific` | The quality of the data value due to a non-specific status. |
| `ConfigurationError` | The quality of the data value due to a configuration error. |

| Name | Description |
|---|---|
| `NotConnected` | The quality of the data value due to a non-connectivity. |
| `DeviceFailure` | The quality of the data value due to device failure. |
| `SensorFailure` | The quality of the data value due to sensor failure. |
| `LastKnownValue` | The quality of the data value from the last known value. |
| `CommFailure` | The quality of the data value due to sensor failure. |
| `OutOfService` | The quality of the data value due to an out of service status. |
| `ScaledOutOfRange` | The quality of the data value due to the value being out of range. |
| `OffLine` | The quality of the data value due to the source being offline. |
| `NoValue` | The quality of the data value if the source does not provide a value. |
| `CalculationError` | The quality of the data value due to a calculation error. |
| `ConditionCollectionHalted` | The quality of the data due to halting the collection. |
| `CalculationTimeout` | The quality of the data value due to calculation time-out |

### ihTKDataType

The `ihTKDataType` structure contains the data types supported by Historian.

#### Definition

```
typedef enum ihTKDataType {

ihTKDataTypeUndefined = 0,
```

```
ihTKScaled,
ihTKFloat,
ihTKDoubleFloat,
ihTKInteger,
ihTKDoubleInteger,
ihTKFixedString,
ihTKVariableString,
ihTKBlob,
ihTKTime,
ihTKInt64,
ihTKUInt64,
ihTKUInt32,
ihTKUInt16,
ihTKMaxDataType
} ihTKDataType;
```

**Parameters**

| Name | Description |
|---|---|
| `ihTKDataTypeUndefined` | An Undefined data type. |
| `ihTKScaled` | A single precision (32-bit) floating-point type. |
| `ihTKFloat` | A single precision (32-bit) floating-point type. |
| `ihTKDoubleFloat` | A double precision (64-bit) floating-point type. |
| `ihTKInteger` | A short, signed integral type (16-bit). |
| `ihTKDoubleInteger` | A long, signed integral type (32-bit). |
| `ihTKFixedString` | A fixed-length UNICODE string The length is determined by `Tag.FixedStringLengthtag` property. |
| `ihTKVariableString` | A variable-length UNICODE string. |
| `ihTKBlob` | An unstructured, binary data type. |
| `ihTKTime` | A date-time type (64-bit), capable of storing one `Date-Time` instance value. |
| `ihTKInt64` | A __int64-bit quad integer. |
| `ihTKUInt64` | A __int64-bit unsigned quad integer. |

| Name | Description |
|---|---|
| `ihTKUInt32` | A long, unsigned integral type (32-bit). |
| `ihTKUInt16` | A short, unsigned integral type (16-bit). |
| `ihTKMaxDataType` | A max data type. |

### ihTKInterfaceType

The `ihTKInterface` structure contains the different interfaces that can interact with Historian. For example, `ihTKCustom` is a custom collector type generated by Toolkit.

#### Definition

```
typedef enum ihTKInterfaceType {
ihTKInterfaceUndefined=0,
ihTKIFix,
ihTKRandom,
ihTKOPC,
ihTKFile,
ihTKIFixLabData,
ihTKManualEntry,
ihTKOther,
ihTKCalcEngine,
ihTKServerToServer,
ihTKPI,
ihTKOPCAE,
ihTKCIMPE,
ihTKPIDistributor,
ihTKCIMME,
ihTKPerfTag,
ihTKCustom
} ihTKInterfaceType;
```

#### Parameters

| Name | Description |
|---|---|
| `ihTKIFix` | Interface for iFix_collector. |
| `ihTKRandom` | Interface for Simulation collector. |

| Name | Description |
|---|---|
| `ihTKOPC` | Interface for OPC DA collector. |
| `ihTKFile` | Interface for File collector. |
| `ihTKIFixLabData` | Interface for Deprecated. |
| `ihTKManualEntry` | Interface for Deprecated. |
| `ihTKOther` | Interface for Deprecated. |
| `ihTKCalcEngine` | Interface for Calculation collector. |
| `ihTKServerToServer` | Interface for ServerToServer collector. |
| `ihTKPI` | Interface for OSI PI collector. |
| `ihTKOPCAE` | Interface for OPC A&E collector. |
| `ihTKCIMPE` | Interface for Native CIMPLICITY collector. |
| `ihTKPIDistributor` | Interface for OSI PI distributor. |
| `ihTKCIMME` | Interface for Proficy Machine Edition collector. |
| `ihTKCustom` | Interface for Custom collector. |

### ihTKCollectionType

The `ihTKCollectionType` structure indicates the types of data collection methods Historian supports.

#### Definition

```
typedef enum ihTKCollectionType {
ihTKUnsolicited=1, ihTKPolled
} ihTKCollectionType;
```

#### Parameters

| Name | Description |
|---|---|
| `ihTKUnsolicited` | Asynchronous data collection based on incoming data value changes. |
| `ihTKPolled` | Periodic data collection based on a configured interval. |

### ihTKTimeStampType

The `ihTKTimeStampType` structure stores timestamp for data according to source time or the collector time.

**Definition**

```
typedef enum ihTKTimeStampType {
ihTKSource = 1, ihTKInterface,
} ihTKTimeStampType;
```

**Parameters**

| Name | Description |
|---|---|
| `ihTKSource` | Stores the timestamp from data source. |
| `ihTKInterface` | Stores the timestamp from the collector based on the host computer clock. |

### ihTKTimeResolution

The `ihTKTimeResolution` structure contains the time resolution that Historian supports.

**Definition**

```
typedef enum ihTKTimeResolution {
ihTKSeconds = 0,
ihTKMilliseconds,
ihTKMicroseconds,
ihTKNanoseconds
} ihTKTimeResolution;
```

**Parameters**

| Name | Description |
|---|---|
| `ihTKSeconds` | Time up to a resolution of 1 second. |
| `ihTKMilliseconds` | Time up to a resolution of 1 millisecond (1-thousandth of a second). |
| `ihTKMicroseconds` | Time up to a resolution of 1 microsecond (1-millionth of a second). |
| `ihTKNanoseconds` | Time up to a resolution of 1 nanosecond (1-billionth of a second). |

### ihTKTagId

The `ihTKTagId` structure indicates the GUID of a tag.

**Definition**

```
typedef struct { uint32_t
Data1; unsigned short
Data2; unsigned short
Data3;
unsigned char Data4[ 8 ];
} ihTKTagId;
```

### ihTKConditionCollectionComparison

The `ihTKConditionCollectionComparison` structure eliminates storing the values that are not within the defined range.

**Definition**

```
typedef enum ihTKConditionCollectionComparison
{
ihTKConditionComparisonUndefined = 0,
ihTKConditionComparisonEqual,
ihTKConditionComparisonLessThan,
ihTKConditionComparisonLessThanEqual,
ihTKConditionComparisonGreaterThan,
ihTKConditionComparisonGreaterThanEqual,
ihTKConditionComparisonNotEqual
}ihTKConditionCollectionComparison;
```

**Parameters**

| Name | Description |
|---|---|
| `ihTKConditionComparisonEqual` | Equality comparison operator. |
| `ihTKConditionComparisonLessThan` | Value less than the comparison operator. |
| `ihTKConditionComparisonLessThanEqual` | Value less than or equal comparison operator. |
| `ihTKConditionComparisonGreaterThan` | Value greater than comparison operator. |
| `ihTKConditionComparisonGreaterThanEqual` | Value greater than or equal comparison operator. |

| Name | Description |
| --- | --- |
| `ihTKConditionComparisonNotEqual` | Inequality comparison operator. |

### ihTKAlarmInterfaceProperties

#### Definition

```
typedef struct ihTKAlarmInterfaceProperties {
ihTKBoolean SupportsEventFiltering;
ihTKBoolean SupportsCategoryFiltering;
ihTKBoolean SupportsSourceFiltering;
ihTKBoolean SupportsAreaFiltering;
ihTKBoolean SupportsSeverityFiltering;
int NumSimpleEvents;
wchar_t** SimpleEventList;
int NumTrackingEvents;
wchar_t** TrackingEventList;
int NumConditionEvents;
wchar_t** ConditionEventList;
} ihTKAlarmInterfaceProperties;
```

#### Parameters

| Name | Description |
| --- | --- |
| `SupportsEventFiltering` | Indicates if it supports event filtering. |
| `SupportsCategoryFiltering` | Indicates if it supports category filtering. |
| `SupportsSourceFiltering` | Indicates if it supports source filtering. |
| `SupportsAreaFiltering` | Indicates if it supports area filtering. |
| `SupportsSeverityFiltering` | Indicates if it supports severity filtering. |
| `NumSimpleEvents` | Number of simple events. |
| `SimpleEventList` | Simple events. |
| `NumTrackingEvents` | Number of tracking events. |
| `TrackingEventList` | Tracking events. |
| `NumConditionEvents` | Number of condition events. |
| `ConditionEventList` | Condition events. |

#### ihTKHierarchicalBrowseResponse

The `ihTKHierarchicalBrowseResponse` structure represents the tags in a hierarchical way.

**Definition**

```
typedef struct ihTKHierarchicalBrowseResponse {
int NodeCount;
wchar_t**      NodeNames;
wchar_t**      FullNodeNames;
int LeafCount;
wchar_t**      LeafNames;
wchar_t**      FullLeafNames;
} ihTKHierarchicalBrowseResponse;
```

**Parameters**

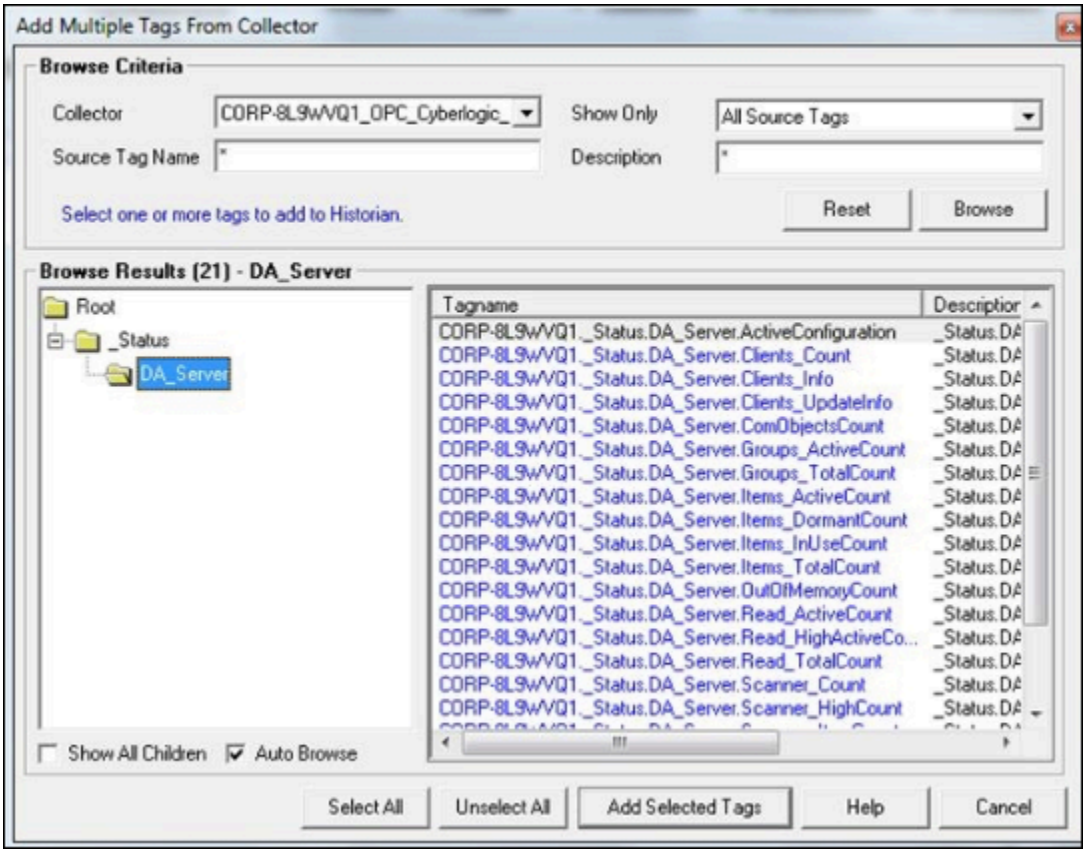
| Name | Description |
|---|---|
| `NodeCount` | Number of elements in the node. |
| `NodeNames` | Names of all the node names. |
| `FullNodeNames` | Complete node name. |
| `LeafCount` | Number of leaf elements. |
| `LeafNames` | Names of all the leaf elements. |
| `FullLeafNames` | Complete leaf names. |

# Hierarchical Custom Controller Browsing

## Browsing Custom Controller in a Hierarchy

Hierarchical browsing can be developed using the Collector Toolkit. Hierarchical browsing enables you to browse custom collectors in a hierarchical manner if your server supports hierarchical organization of tags in a tree structure.

To browse for Custom Collector tags in a hierarchy:




1. Browse your data source for new tags.
2. From the Collector list, select the custom collector you wish to browse. A hierarchical tree appears in the Browse Results window.
3. To limit the displayed tags to only those that are not collected, from the Show Only list select **Source Tags Not Collected**.
4. To limit the displayed tags to match a tag name or tag description, enter the value to match in the **Source Tag Name** or **Description** text boxes.
5. Navigate to the node in the tree you want to browse, and then select Browse. The tags within the selected portion of the Custom Collector tag hierarchy will be displayed.
    a. To browse automatically, select the **AutoBrowse** check box. The available tags will be displayed in the Browse Results window whenever a node is selected in the tree.
    b. To show all child elements within a hierarchy, enable the **Show All Children** check box. All tags at, or below the hierarchical level of the selected node in the tree will be displayed in the Browse Results window.
6. Select the tag or tags you want to add to Historian, and select **Add Selected Tags**. Collected tags will appear in black in the tag list.

**Note:**

- The Browse option for the Custom Collector will not return all items that reside in the Server configuration. Items that may not get returned include, but are not limited to, unsupported data types and user-defined items in some Simulation Servers. Occasionally, items that do not appear in the browse can still be added manually using the Add Tag Manually option.
- If you are browsing and adding tags with the Custom Collector, note that some Custom Collectors do not support data blocks with a length greater than 1. These Servers can choose to show the first item in an array in the browse rather than show them all. For example, an OPC Server may contain 3000 analog values datablock:1 to datablock:3000, but would only show datablock:1.
- If you want to archive data from poll records of a length greater than 1, it is recommended that you use the Excel Add-In to configure a large block of tags - including the missing items - and add the tags.
- If you are unable to browse items on your server containing forward slashes ( / ), you may have to change the default separator in your Custom Collector configuration. To do so, you will need to open the Windows Registry Editor and edit the `HKEY_LOCAL_MACHINE\SOFTWARE\Intellution, Inc\iHistorian\Services\[Collector Name]\OPCBrowseTreeSep key` (where [*Collector Name*] is the name of your Custom collector) and change the string value to a character not present among your Server item IDs. Typical values include |, !, or &. Create this key, if it does not already exist.
- If you are unable to browse readable items in your server with the Custom collector, you may need to change the browse access mask used by the collector. To do so, you will need to open the Windows Registry Editor and edit the [`HKEY_LOCAL_MACHINE\SOFTWARE\Intellution, Inc.\iHistorian\Services\[Collector Name`] (where [*Collector Name*] is the name of your Custom collector) and add DWORD key "OPCBrowseAccessRightsMask"=dword:00000003. Valid values are 0, 1, 2, 3 with 1 being the default. Use 0 or 3 if you are unable to browse readable items. Creating or changing the value takes effect on the next browse attempt and does not require a collector restart.

## Developing Hierarchical Browsing using Collector Toolkit

The following callback is used to develop hierarchical browsing using the Collector Toolkit:

```
ihCollectorToolkitGetTagsHierarchical(wchar_t* BrowsePosition, wchar_t* NodeFilter,

ihTKHierarchicalBrowseResponse* Response);
```

The following are the parameters in this callback:

| Name | Description |
|---|---|
| `BrowsePosition` | The place where the current node is selected. |
| `NodeFilter` | The delimiter using which hierarchical node differentiation is made |
| Response structure | Contains the number of nodes/leaves for a particular node. |

To define the hierarchical tree format as illustrated in the Add Multiple Tags from Collector figure above, Historian provides the following information in three responses, provided as a sample here. Use the information in the following three responses to browse the tags and add the required tags.

As a first response Historian returns:

```
ihInterfaceGetTagsHierarchical()
```

The following are the parameter values:

```
BrowsePosition "ihHierarchicalBrOwSeRoOt"

NodeFilter: "" (By default, this uses a “/”)

Response = NULL

NodeCount 1 int

NodeNames "_Status" wchar_t *

FullNodeNames "/_Status" wchar_t *

ihInterfaceGetTags()

BrowsePosition “ihHierarchicalBrOwSeRoOt” wchar_t*
```

Second response:

```
ihInterfaceGetTagsHierarchical()

BrowsePosition "/_Status"

NodeNames "DA_Server"

FullNodeNames "/_Status/DA_Server"

ihInterfaceGetTags()

BrowsePosition “/_Status”
```

Third response:

```
ihInterfaceGetTagsHierarchical()

BrowsePosition "/_Status/DA_Server"

BrowsePosition "/_Status/DA_Server"
```

## Collector Initialization Callbacks

The following are the callbacks used for a collector when it is initialized:

- `ihCollectorToolkitPreInitialize`
- `ihCollectorToolkitInitialize`
- `ihCollectorToolkitInitializeCompleted`

When a collector is shutting down, `ihCollectorToolkitShutdown` method is called and the collector performs all the necessary steps before it is completely shut down.

### Example

The following sample program helps you understand the `ihCollectorToolkitPreInitialize` function.

Historian expects Custom Collector Pre-Initialization information from the user. For example, custom collector can browse tags, collector type and so on.

```
/// </summary>

/// <param name="PreCfg">Collector Pre - Configuration information structure</param>

/// <param name="Cfg">Collector Configuration information structure</param>

void RandomValueSimulator::ihCollectorToolkitPreInitialize(ihInterfaceTKPreCfgInfo *PreCfg, ihInterfaceTKCfgInfo *Cfg)

{

// Initializes General1-5 values, here General1,2 were initialized with 1000, 60.

Cfg->CustomProp1 = TKStrdup(_T("1000"));

Cfg->CustomProp2 = TKStrdup(_T("60"));

// Historian follows representation of the collectors in the form of ComputerName_CollectorName, Tags in the form of

 ComputerName.TagName.

// In the following section custom collector is trying to get the computer name.

CString ComputerName, IP;

TKGetHostNameAndIP(ComputerName, IP);

if (ComputerName.GetLength() >; 0)

{

ComputerName.Append(_T("."));

wcscpy_s(Cfg->;DefaultTagPrefix, ComputerName.GetBuffer());
```

```
}

RandPreCfg = *PreCfg;

RandPreCfg.CanSendOPCQuality = TRUE; // to send OPC Quality

RandPreCfg.InterfaceType = ihTKCustom; // interface type

RandPreCfg.MultipleInstancesAllowed = FALSE;

RandPreCfg.MinimumInterval = RandMinimumInterval;

RandPreCfg.MaxTagsPerRead = MaxTagsPerGroup;

RandPreCfg.CanReadASync = TRUE;// to read tags Asynchronously

RandPreCfg.CanBrowseSource = TRUE;// you can browse the collector

RandPreCfg.CanSourceTimestamp = TRUE;// collector sends data containing source time stamp or server time stamp

RandPreCfg.ForceInputScaling = FALSE;

RandPreCfg.NeedMsgPump = FALSE;

RandPreCfg.ForcedScaleLO = 0.0; // Low engineering unit

RandPreCfg.ForcedScaleHI = (float) RAND_MAX;// High engineering unit

RandPreCfg.DoesReloadMode = FALSE;

RandPreCfg.DoesLagTimes = FALSE;

RandPreCfg.CanBrowseHierarchical = TRUE;

*PreCfg = RandPreCfg;

}
```

For the `ihCollectorToolkitInitialize` function:

```
Collector Initialization

/// </summary>

/// <param name="Cfg">Collector Configuration information</param>

/// <param name="PreCfg">Collector Pre-Configuration Information</param>

/// <param name="ErrorMsg">Error Message while initializing</param>

/// <param name="ErrorMsgSize">Size of Error Message</param>

/// <param name="RegKeyName">modification required registry keys, if any </param>

/// <param name="Callbacks">Callbacks of all the asynchronous methods</param>

/// <param name="DoDebug">debug param</param>

/// <returns>status of the method call</returns>

int RandomValueSimulator::ihCollectorToolkitInitialize(ihInterfaceTKCfgInfo *Cfg,

ihInterfaceTKPreCfgInfo *PreCfg, wchar_t *ErrorMsg, int ErrorMsgSize, wchar_t *RegKeyName, ihCollectorToolkitCallback

 *Callbacks, int DoDebug)

{

// updates configuration information to collector

ihCollectorToolkitPropertyUpdate(Cfg);
```

```
TKFree(TagPrefix);

// Historian follows representation of Tags in the form of ComputerName.TagName.

TagPrefix = TKStrdup(Cfg->DefaultTagPrefix);

Cfg->DoOnFly = 1;

// Initializes error msg to ""(NULL)

ErrorMsg = TKStrdup(_T(""));

srand((unsigned) time(NULL));

RandCfg = *Cfg;

g_Callbacks = Callbacks;

return(TRUE);

}
```

## Polled Tag Callbacks

For polled tags we use different callback functions at various stages.

**Step 1**

At collector start up, `ihCollectorToolkitPolledInit` and `ihCollectorToolkitPolledInitCompleted` callbacks are triggered and the details of all the Polled tags for the given collector is returned.

**Step 2**

If a polled tag is added, `ihCollectorToolkitPolledAddTag` method is called and the details of the tag is entered in the collector tag list.

**Step 3**

If a polled tag is deleted, `ihCollectorToolkitPolledDeleteTag` method is called and the tag details are removed from the collector tag list.

### Example

The following sample helps you understand the given callback functions.

The collectors maintains the polled tag details in the local Cache as:

```
map<int, ihInterfaceTKPolledTagInfo> TagIdToTagInfoMap;
```

This map is filled using this sample example:

```
/// Polled tags initialization started

/// </summary>

/// <returns>TRUE/FALSE</returns>
```

```
int RandomValueSimulator::ihCollectorToolkitPolledInit(void)

{

TagIdToTagInfoMap.clear();

return CCollectorDelegator::ihCollectorToolkitPolledInit();

}


---------------------------------------------------------------------


/// <summary>

/// Historian updating the source saying that, polled tags initialization successful.

Are there any initialization from source for polled tags?

/// <summary>

/// <returns>TRUE/FALSE</returns>

int RandomValueSimulator::ihCollectorToolkitPolledInitCompleted(void)

{

return CCollectorDelegator::ihCollectorToolkitPolledInitCompleted();

}


---------------------------------------------------------------------


/// <summary>

/// Adds polled tag to the historian from source

/// </summary>

/// <param name="PolledTag">ihInterfaceTKPolledTagInfo instance</param>

/// <param name="IsCollectorStarting">collector status</param>

/// <returns>TRUE/FALSE</returns>

int RandomValueSimulator::ihCollectorToolkitPolledAddTag(ihInterfaceTKPolledTagInfo

*PolledTag, int IsCollectorStarting)

{

TagIdToTagInfoMap.insert(std::pair<int, ihInterfaceTKPolledTagInfo>

(PolledTag-<TagId, *PolledTag));

return CCollectorDelegator::ihCollectorToolkitPolledAddTag(PolledTag,

IsCollectorStarting);

}


---------------------------------------------------------------------
```

```
/// <summary>
/// Deleted polled tag from Historian from Client Tools/non web admin/external tools
/// <summary>
/// <param name="tagId">Tag Identification</param>
/// <returns>TRUE/FALSE</returns>
int RandomValueSimulator::ihCollectorToolkitPolledDeleteTag(int tagId)
{
TagIdToTagInfoMap.erase(TagIdToTagInfoMap.find(tagId));
return CCollectorDelegator::ihCollectorToolkitPolledDeleteTag(tagId);
}
```

## Unsolicited Tags Callbacks

For unsolicited tags we use different callback functions at various stages.

**Step 1**

At collector start up, `ihCollectorToolkitASyncInit` and `ihCollectorToolkitASyncInitCompleted` callbacks are triggered and the details of all the unsolicited tags for the given collector is returned.

**Step 2**

If an unsolicited tag is added, `ihCollectorToolkitASyncAddTag` method is called and the details of the tag is entered in the collector tag list.

**Step 3**

If an unsolicited tag is deleted, `ihCollectorToolkitASyncDeleteTag` method is called and the tag details are removed from the collector tag list.

### Example

The following sample helps you understand the given callback functions. The below sample code creates a thread for simulating the Unsolicited tag's collector behavior.

```
//This structure is the list of Unsolicited Tags.
struct AsyncTagList : public CList<ihInterfaceTKASyncTagInfo*, ihInterfaceTKASyncTagInfo*>
{
virtual ~AsyncTagList()
{
FreeAll();
}
```

```
void AddTag(ihInterfaceTKASyncTagInfo* tag)

{

ihInterfaceTKASyncTagInfo* info = new ihInterfaceTKASyncTagInfo;//need to allocate in delegator

memcpy(info, tag, sizeof(ihInterfaceTKASyncTagInfo));

AddTail(info);

}

void FreeAll()

{

while (!IsEmpty())

delete RemoveHead();

}

};

AsyncTagList g_AsyncTags;


--------------------------------------------------------------------------------


/// <summary>

/// Initializes unsolicited tags, by creating dedicated thread

/// <summary>

/// <returns></returns>

int    RandomValueSimulator::ihCollectorToolkitASyncInit(void)

{

       CSingleLock lock(&g_Sync, TRUE);

       g_AsyncTags.FreeAll();

       if (!g_AsyncThread)

              g_AsyncThread = AfxBeginThread(TKAsyncReadFunc, this);

       return TRUE;

}


--------------------------------------------------------------------------------


/// <summary>

/// unsolicited tags initialization completed

/// <summary>

/// <returns>TRUE/FALSE</returns>

int    RandomValueSimulator::ihCollectorToolkitASyncInitCompleted(void)

{
```

```
	return TRUE;
}


--------------------------------------------------------------------------------


/// <summary>
/// Custom Collector Initialization completed and is ready to read data from Source for unsolicited tag
/// <summary>
/// <returns>TRUE/FALSE</returns>
int    RandomValueSimulator::ihCollectorToolkitASyncStartReading(void)
{
	InterlockedExchange(&g_DoAsyncRead, TRUE);
	return TRUE;
}


--------------------------------------------------------------------------------


/// <summary>
/// Adds unsolicited tag to the historian
/// <summary>
/// <param name="ASyncTag">ihInterfaceTKAsyncTagInfo pointer</param>
/// <param name="IsCollectorStarting">Current Status of the Collector</param>
/// <returns>TRUE/FALSE</returns>
int    RandomValueSimulator::ihCollectorToolkitASyncAddTag(ihInterfaceTKASyncTagInfo *ASyncTag, int
 IsCollectorStarting)
{
	CSingleLock lock(&g_Sync, TRUE);
	g_AsyncTags.AddTag(ASyncTag);
	return TRUE;
}


--------------------------------------------------------------------------------


/// <summary>
/// From Clients tools/non-web admin/custom tools, if user deletes a tag, historian updates custom collector saying
 that tag got deleted.
/// So that, custom collector stops collecting data from source for that tag
```

```
/// <summary>
/// <param name="tagId">Tag Identifier</param>
/// <returns>TRUE/FALSE</returns>
int    RandomValueSimulator::ihCollectorToolkitASyncDeleteTag(int tagId)
{
       CSingleLock lock(&g_Sync, TRUE);
       POSITION pos = g_AsyncTags.GetHeadPosition();
       while (pos)
       {
              ihInterfaceTKASyncTagInfo * tagInfo = g_AsyncTags.GetAt(pos);
              if (tagInfo->TagId == tagId)
              {
                     g_AsyncTags.RemoveAt(pos);
                     return TRUE;
              }
              g_AsyncTags.GetNext(pos);
       }
       return TRUE;
}

--------------------------------------------------------------------------------

/// <summary>
/// This method called by historian, if it needs to perform any calculations on source data. This method is only
 usefull in "Calculation collector" way of collection for unsolicited tags
/// <summary>
/// <param name="StartTime"> start time for reload</param>
/// <param name="EndTime"> end time for reload</param>
/// <returns>TRUE/FALSE</returns>
int RandomValueSimulator::ihCollectorToolkitASyncReload(ihTKTimeStruct *StartTime, ihTKTimeStruct *EndTime)
{
       return CCollectorDelegator::ihCollectorToolkitASyncReload(StartTime, EndTime);
}

--------------------------------------------------------------------------------

/// <summary>
```

```
/// Unsolicited dedicated thread corresponding method. Here all unsolicited tags get data from source(usually, way
 should be source notification to historian) and sends to historian
/// <summary>
/// <param name="param"> Collector instance </param>
UINT RandomValueSimulator::TKAsyncReadFunc(void* param)
{
	POSITION pos = NULL;
	RandomValueSimulator* pColl = (RandomValueSimulator*) param;
	while (TRUE)
	{
		if (g_DoAsyncRead)
		{
			CSingleLock lock(&g_Sync, TRUE);
			if (!g_AsyncTags.IsEmpty())
			{
				// gets each unsolicited tag into ihInterfaceTKAsyncTagInfo object
				if (!pos)
					pos = g_AsyncTags.GetHeadPosition();
				ihInterfaceTKASyncTagInfo* tag = g_AsyncTags.GetNext(pos);
				int numTags = 1;
				ihInterfaceTKDataInfo data;
				memset(&data, 0, sizeof(ihInterfaceTKDataInfo));
				data.Tag = pColl->TKStrdup(tag->Tag);
				data.DataProp.ValueDataType = tag->DataType;
				data.DataProp.TimeStamp = pColl->TKGetSystemTime();
				// gets data for selected tag
				pColl->ihCollectorToolkitGetData(0, 0, 0, numTags, NULL, &data);
				unsigned long collectionTime = (unsigned long)time(0);
				int tagId = tag->TagId;
				 ihInterfaceTKASyncData asyncData;
				memset(&asyncData, 0, sizeof(ihInterfaceTKASyncData));
				asyncData.NumValues = numTags;
				asyncData.TagIds = &tagId;
				asyncData.Values = &data.DataProp;
				asyncData.CollectionTimes = &collectionTime;
				// sends to historian
				pColl->ihCollectorToolkitDataCallback(&asyncData);
```

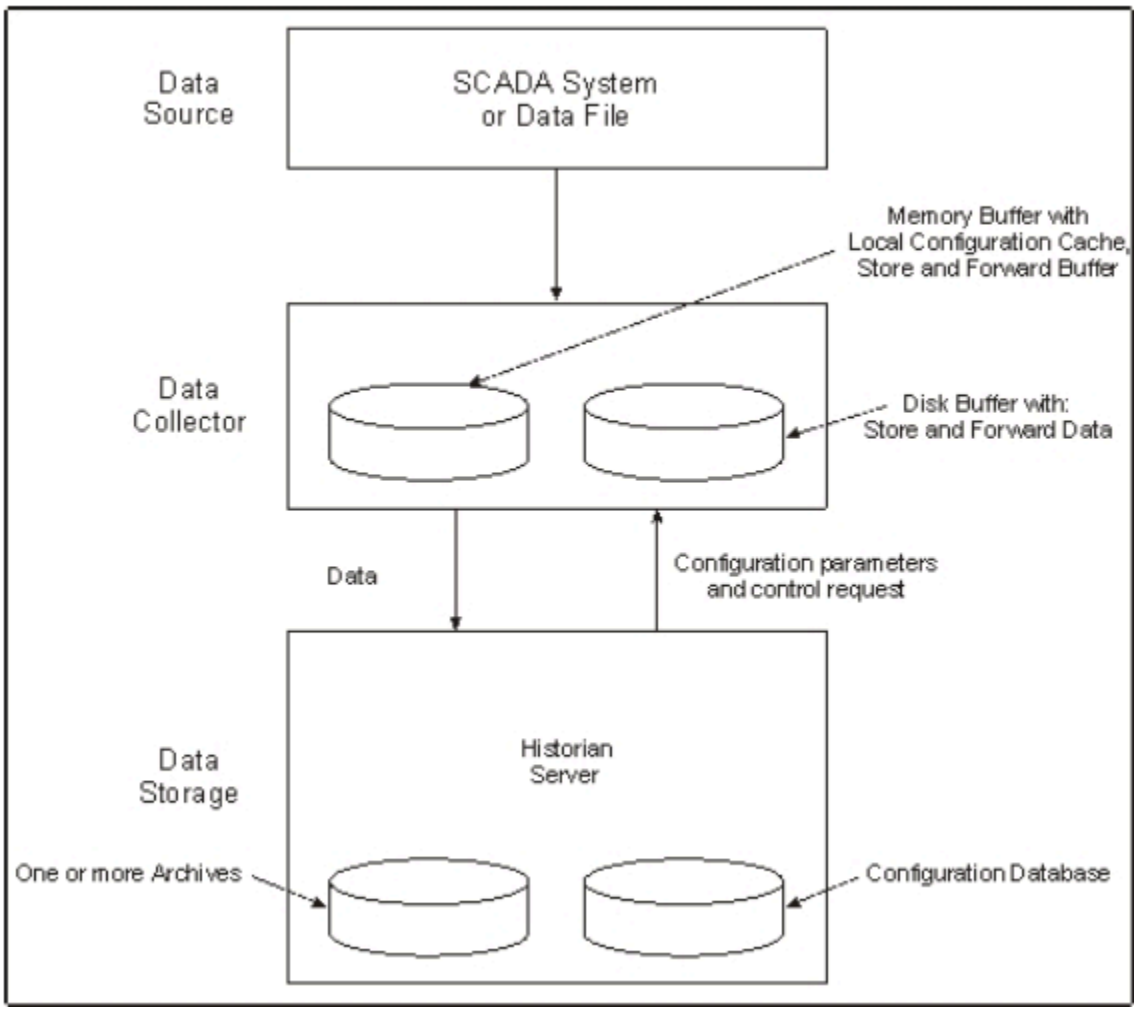
```
                    }
                    else
                    {
                          pos = NULL;
                    }
             }
             else
             {
                    pos = NULL;
             }
             long sleepTimeInMs = 500 + (rand() % 46) * 100; // sleep from 0.5 to 5 seconds
             Sleep(sleepTimeInMs);
      }
      return 0;
}
```

# Chapter 14. Data Collectors - General

## Data Collectors Overview

### About Historian Data Collectors

A data collector gathers data from a data source on a schedule or event basis, processes it, and forwards it to the Historian server or a cloud destination for archiving. The following image shows the data flow in a typical Historian system from a data source to the archive.




The following table provides a list of collectors, their usage, and whether each of them is toolkit-based and consumes a client access license (CAL).

| Collector Name | Description | Is Toolkit-Based? | Consumes a CAL? |
|---|---|---|---|
| The Calculation collector *(on page 2098)* | Performs data calculations on values stored in the archiver. | No | Yes |
| The CygNet collector *(on page 2167)* | Collects data from a CygNet server. | Yes | Yes |
| The File collector *(on page 2181)* | Imports CSV and XML files into Historian. | No | No |
| The HAB collector *(on page 2208)* | Collects data from Habitat. | Yes | |
| The iFIX Alarms and Events collector collector *(on page 2251)* | Collects alarms and events data from iFIX. | No | No |
| The iFIX collector *(on page 2251)* | Collects tag data from iFIX. | No | No |
| The MQTT collector *(on page 2298)* | Collects data published to a topic using an MQTT broker. | Yes | Yes |
| The MQTT Sparkplug B collector *(on page 2312)* | Enables you to connect as a Primary Host application and collect data based on the MQTT Sparkplug B specification (Sparkplug 3.0.0). Specifically, you can subscribe to a topic that was published using the Sparkplug B payload and collect time-series data in a structured and understandable format. | Yes | Yes |

| Collector Name | Description | Is Toolkit-Based? | Consumes a CAL? |
|---|---|---|---|
| The ODBC collector *(on page 2341)* | Collects data from an application based on an ODBC driver. | Yes | Yes |
| The OPC Classic Alarms and Events collector *(on page 2390)* | Collects data from an OPC Classic Alarms and Events server (such as CIMPLICITY). | No | No |
| The OPC Classic DA collector *(on page 2356)* | Collects data from an OPC Classic Data Access (DA) server (such as CIMPLICITY). | | |
| The OPC Classic HDA collector *(on page 2374)* | Collects data from an OPC Classic Historical Data Access (HDA) server (such as CIMPLICITY). | Yes | Yes |
| The OPC UA Data Access (DA) collector *(on page 2426)* | Collects data from an OPC UA DA server (such as CIMPLICITY). | Yes | Yes |
| The OSI PI collector *(on page 2443)* | Collects data from an OSI PI server. | No | No |
| The Python collector *(on page 2463)* | Run Python scripts on tag values and stores them in Historian | No | No |
| The Server-to-Server collector *(on page 2499)* | Collects data from a Historian server and sends it to another Historian server. | No | Yes |
| The Server-to-Server distributor *(on page 2554)* | Collects data from a smaller Historian server and sends it to a larger, centralized Historian | No | Yes |

| Collector Name | Description | Is Toolkit-Based? | Consumes a CAL? |
|---|---|---|---|
| | server or a cloud destination. | | |
| The Simulation collector *(on page 2558)* | Generates random numbers and string patterns for testing/demonstration purposes. | No | No |
| The Windows Performance collector *(on page 2562)* | Collects Windows performance counter data. | Yes | Yes |
| The Wonderware collector *(on page 2566)* | Collects data from a Wonderware Historian 2014 R2 server. | Yes | Yes |

Data collectors use a specific data acquisition interface that match the data source type, such as iFIX Easy Data Access (EDA) or OPC 1.0 or 2.0 (Object Linking and Embedding for Process Control). For more information, see Supported Acquisition Interfaces *(on page 1998)*. The Simulation collector generates random numeric and string data. The File collector reads data from text files.

**Limitations:** When failover occurs from a primary collector to a secondary collector (or vice versa), there will be some data loss as the collector tries to connect to the source to fetch the data.

## Bi-Modal Cloud Data Collectors

Collectors can send data to an on-premises Historian server as well as cloud destinations such as Google Cloud, Azure IoT Hub, AWS Cloud, and Predix Cloud. Therefore, these collectors are called bid-modal collectors. The following collectors, however, are not bi-modal collectors; they can send data only to an on-premises Historian server:

- The File collector
- The HAB collector
- The iFIX Alarms and Events collector
- The Calculation collector
- The Server-to-Server distributor
- The OSI PI Distributor
- The OPC Classic Alarms and Events collector
- The Python collector

When you create a collector instance, you can choose whether you want the collector to send data to an on-premises Historian server or a cloud destination. You can create multiple instances of the same collector, and configure each of them to send data to a different destination.

**Note:**
Bi-modal Collectors support up to Transport Layer Security (TLS) 1.2.

The Predix cloud destination (via a secure web socket) supports APM, Automation, or Brilliant Manufacturing Cloud subscription. The Collector Toolkit is updated as well. Hence, a custom collector created using the toolkit has the same capabilities.

There are a few differences in the working of a bi-modal collector based on whether the destination is Historian or cloud. Following table explains the key differences.

| Functionality | Destination - Historian | Destination - Cloud |
|---|---|---|
| HISTORIANNODENAME registry key | Contains the destination Historian Server's name/ IP Address. | Contains the cloud destination settings as well as proxy historian server name or IP if applicable (configServer).<br><br>Cloud destination format:<br>`CloudDestinationAddress\|configServer\|IdentityIssuer\|ClientID\|ClientSecret\|ZoneID\|Proxy\|proxyUser\|proxyPassword` |
| Mapping Source Tags<br><br>with Destination Tags<br><br>(Add Tags) | You must map tags in Historian Server to Data Source tags using one of the Admin tools (VB Admin/Web Admin). The data gets stored in IHA files and the Tag configurations are stored in IHC files. | As it is not possible to map tags in the Cloud with tags in the Data Source, user must select if mapping should be done through Historian (works as a proxy) or through Offline Configuration File at the time of installation.<br><br>If the user selects Historian, then tags will be created in the Cloud which in turn may have been mapped through one of the Admin tools (VB Admin/Web Admin).<br><br>If Offline Configuration File is selected, the user must provide an XML configuration file containing tag configurations that need to be cre- |

| Functionality | Destination - Historian | Destination - Cloud |
|---|---|---|
| | | ated in the Cloud for mapping them with the Source tags. |
| Other Tag Management Operations such as Delete, Rename, Data cleaning | It is possible to do all tag management operations. | No tag management operations are allowed. After you update the offline tag configuration file, or after you specify the tags using Historian Administrator, the changes are reflected automatically (without the need to restart the collector). |
| Data Type support | All standard data types are supported. | All other data types, excepting arrays, enums and User defined types (UDT), BLOB, are supported. |

## Data Collector Software Components

Data Collector software consists of four main components:

- Data Collector Program

  Executable data collection program for the type of collector. For example, `ihFileCollector.exe`.

- Local Tag Cache

  Cache of configuration information that permits the collector to perform collection even when the archiver is not present at start-up (`*.cfg`).

- Local Outgoing Data Buffer

  Buffer of the data sent to the server that the server has not yet confirmed receiving.

- Historian API

  Interface that connects the collector to the Historian Server for configuration, data flow, and control functions.

## Supported Windows versions for Data Collectors

The following table displays the supported Windows processor versions (32-bit or 64-bit) for the Historian data collectors.

| Collector Name | 32-bit | 64-bit |
|---|---|---|
| The Calculation collector | Yes | Yes |
| The CygNet collector | No | Yes |
| The File collector | Yes | Yes |
| The HAB collector | No | Yes |
| The iFIX Alarms and Events collector collector | Yes | Yes |
| The iFIX collector | Yes | Yes |
| The OPC Classic Alarms and Events collector | Yes | Yes |
| The OPC DA collector | Yes | Yes |
| The OPC Classic HDA collector | No | Yes |
| The OPC UA Data Access (DA) collector | No | Yes |
| The OSI PI collector (API / SDK) | Yes | Yes |
| The OSI PI distributor | Yes | Yes |
| The Python collector | Yes | Yes |
| The Server-to-Server collector | Yes | Yes |
| The Server-to-Server distributor | Yes | Yes |
| The Simulation collector | Yes | Yes |
| The Windows Performance collector | Yes | Yes |
| The Wonderware Collector | No | Yes |
| The ODBC collector | No | Yes |
| The MQTT collector | No | Yes |

## Data Collector Functions

A Historian Data Collector performs the following functions:

- Connects to the data source using a specific data acquisition interface, such as EDA, OPC 1.0, or OPC2.0.
- Groups tags by collection interval for efficient polling.

- Reads data as frequently as 10 times/sec, depending on the configuration parameters of individual tags. An OPC Collector configured for unsolicited collection can read data as frequently as 1 millisecond (or 1000 times/second).
- Scales the collected value to the EGU Range.
- Compresses collected data based on a deadband specified on a tag by tag basis, and forwards only values that exceed the deadband to the Historian Server for final compression and archiving.
- Automatically stores data during a loss of connection to the server and forwards that data to the server after the connection is restored.

### Common Collector Functions

Each collector performs some functions common to all types of collectors (except the File collector). These functions are:

- Maintains a local cache of tag information to sustain collection while the server connection is down.
- Automatically discovers available tags from a data source and presents them to Historian Administrator.
- Buffers data during loss of connection to the server and forwards it to the server when the connection is restored.
- Automatically adjusts timestamps, if enabled, for synchronizing collector and archiver timestamps.
- Supports both collector and device timestamp, where applicable.
- Schedules data polling for polled collection.
- Performs first level of data compression (collector compression).
- Responds to control requests, such as pause/resume collection.

After collecting and processing information, a collector forwards the data to the Historian Server, which optionally performs final compression and stores the information in the Archive Database. The Archive Database consists of one or more files, each of which contains a specific time period of historical data. For more information on Historian Server architecture, refer to System Architecture *(on page 69)*.

### File collector Functions

The File collector imports files in either CSV (Comma Separated Variables) or XML (Extensible Markup Language) format. Since this is basically a file transfer operation, a File collector does not perform the typical collector functions of data polling, browsing for tags, pause/resume collection, data compression, or storing/forwarding of data on loss of server connection. A File collector, however, is an extremely useful tool for importing and configuring tags, for bulk updating of tag parameters and messages, and for importing data from all types of systems.

## Supported Acquisition Interfaces

This section provides a list of specific Data Collectors and the associated Data Acquisition Interfaces (protocols or ways in which data is input).

| Data Collector | Data Acquisition Interface |
|---|---|
| iFIX Data Collector | EDA data acquisition interface |
| Machine Edition View Data Collector | Point Management API Interface |
| OPC Data Collector | OPC data acquisition interface |
| OPC Classic Alarms and Events collector | The OPC Alarms and Events server |
| CygNet collector | SQL Server ODBC Driver Interface |
| File Data Collector | CSV or XML file import |
| Simulation Data Collector | Random pattern of data |
| Calculation collector | Calculations performed on data already in the server |
| Server-to-Server Collector | Data and messages collected from one Historian Server (source) to another Historian Server (destination) |
| OPC UA Data Access (DA) collector | OPC Data Acquisition interface for Microsoft Windows |
| Wonderware Data Collector | SQL Server ODBC Driver Interface |
| ODBC collector | SQL Server ODBC Driver Interface |

## Best Practices for Working with Data Collectors

Here are some best practices that enable the collectors collect and store the most accurate data:

- Synchronize the Windows clock for the following computers:
  - Source Data Archiver of a Server-to-Server Collector
  - Computer on which the Data Collector is running
  - Destination Data Archiver
- Turn on the Data Recovery Mode option for Historical Collectors such as Server-to-Server and Calculation collectors. This ensures that most gaps in data collection due to the unavailability of

source Archiver or in the case of collector not running are automatically filled in the next time the collector runs.

- If you are using polled collection with Calculation or Server-to-Server Collector, ensure that you have at least one uncompressed polled tag so that the polled data is frequently sent to the destination Archiver. This ensures that the bad data marker sent when a collector shuts down has an accurate time stamp that reflects the time of shutdown of the collector.

# About Installing Historian Data Collectors

When you install collectors, the required binaries are downloaded. In addition, if iFIX/CIMPLICITY are installed on the same machine as the collectors, instances of the following collectors are created:

- The iFIX collector
- The iFIX Alarms & Events collector
- The OPC Classic Data Access collector for CIMPLICITY
- The OPC Classic Alarms and Events collector for CIMPLICITY

If an iFIX collector instance created in version 9.0 exists, after you upgrade collectors, another instance of the iFIX collector is created. Because of this, the Remote Collector Manager (RCM) will not work correctly. Therefore, if you want to use RCM, you must delete one of the instances. If needed, you can manually create another instance of the iFIX collector using Configuration Hub or the RemoteCollectorConfigurator utility. This is applicable to the iFIX Alarms and Events collector as well.

**Note:**
If you want to upgrade collectors earlier than version 7.1, additional registries that you create manually are deleted. Therefore, we recommend that you back them up, uninstall the collectors, and then install the latest version.

## Install Collectors Using the Installer

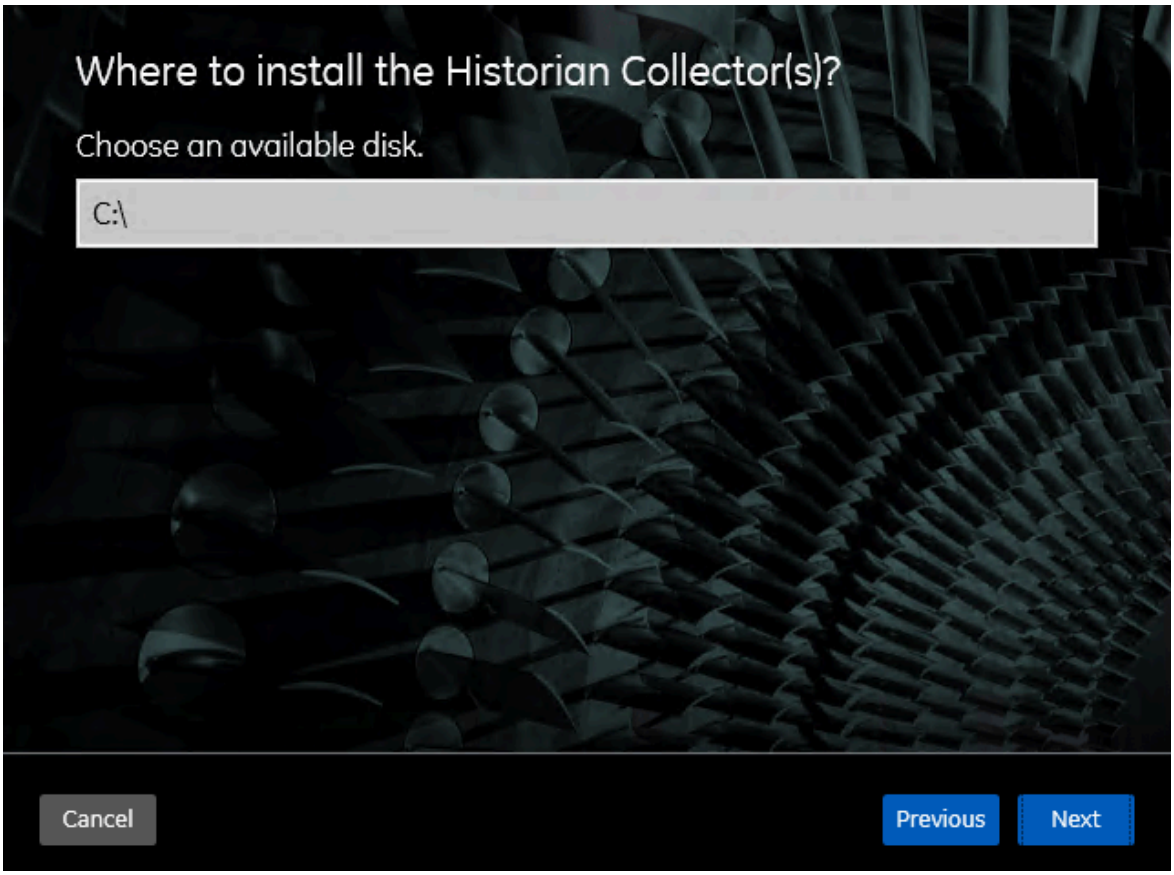
After you install collectors, the following artefacts will be available:

- **Executable files:** These files are required to add a collector instance.
- **Instances of the following collectors:**
  - The iFIX collector
  - The iFIX Alarms & Events collector
  - The OPC Classic Data Access collector for CIMPLICITY
  - The OPC Classic Alarms and Events collector for CIMPLICITY

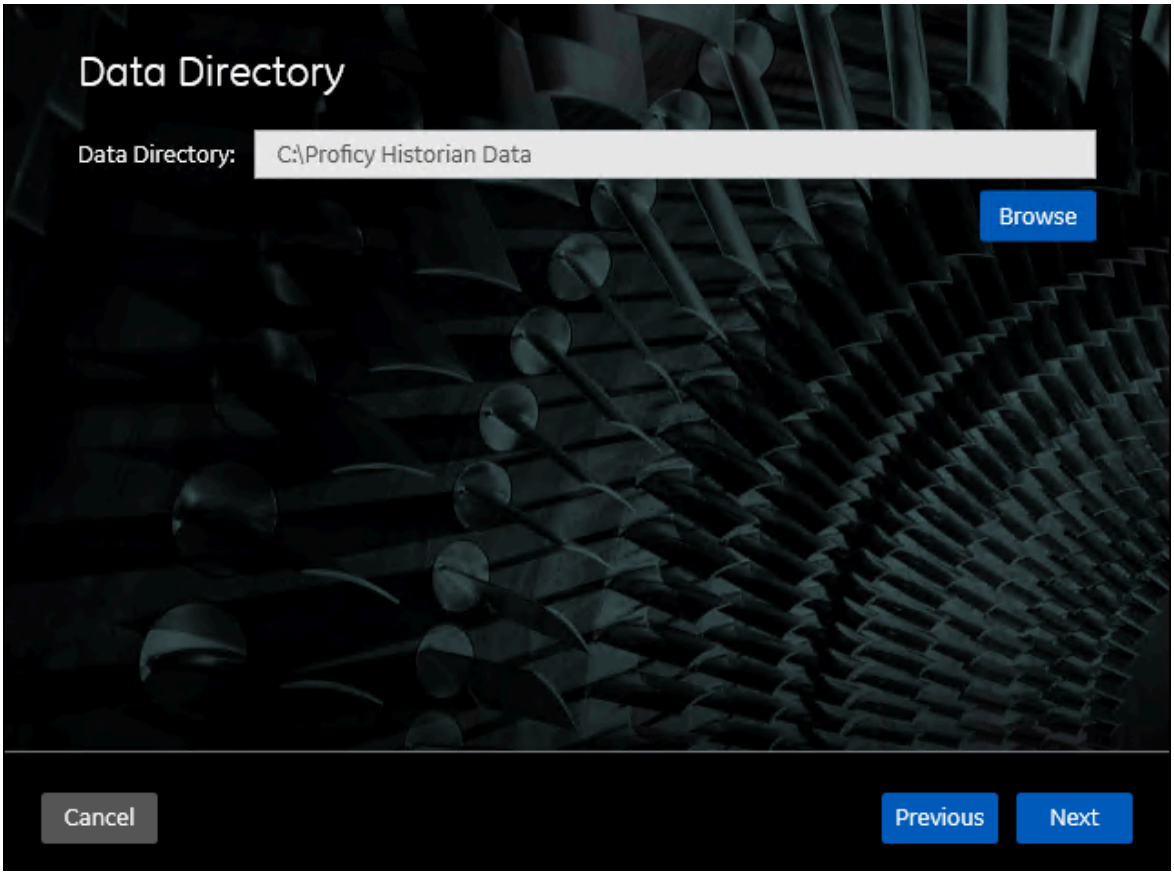
These instances will be created only if iFIX and/or CIMPLICITY are installed on the same machine as the collectors.

- **The Remote Collector Management agent:** Provides the ability to manage collectors remotely.

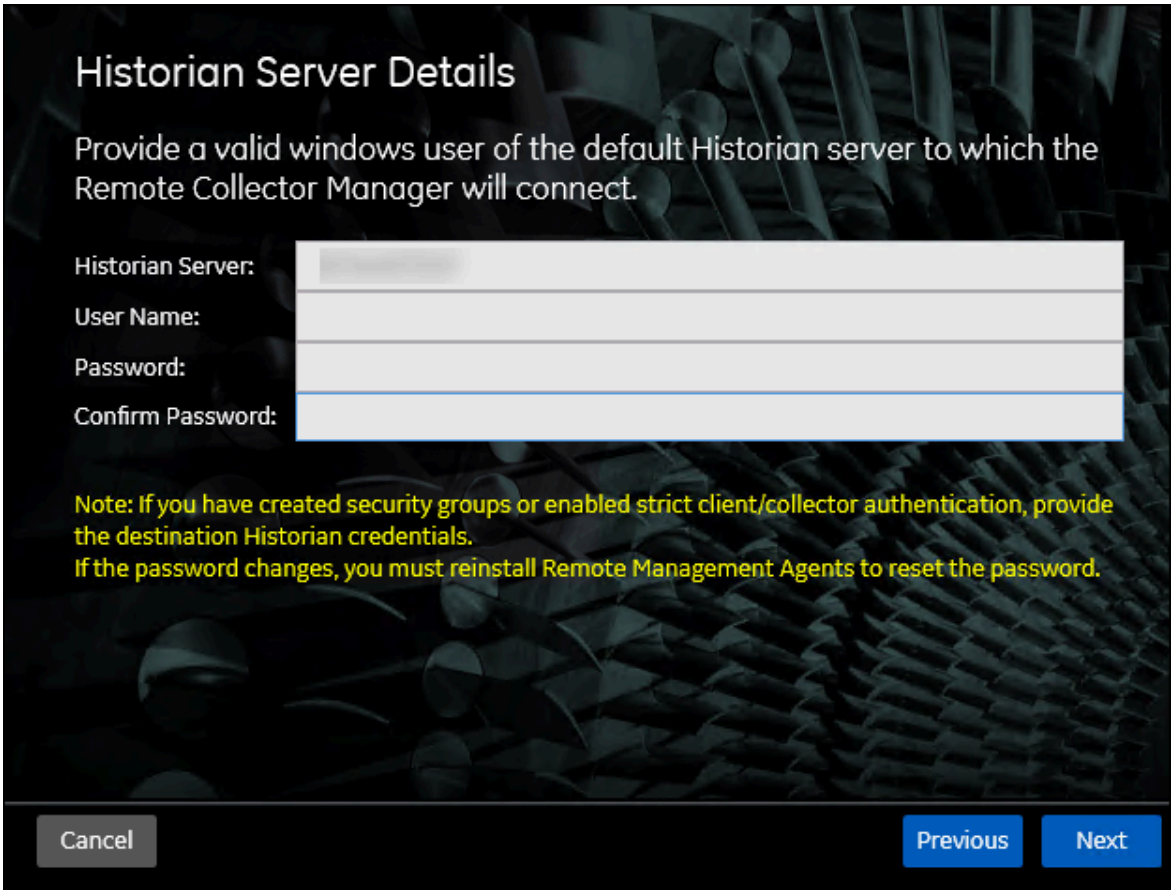
1. Run the `InstallLauncher.exe` file.
2. Select **Install Collectors**.
   The welcome page appears.
3. Select **Next**.
   The license agreement appears.
4. Select the **Accept** check box, and then select **Next**.
   The default installation drive appears.




5. If needed, modify the installation drive, and then select **Next**.
   The data directory page appears.




6. If needed, change the folder for storing the collector log files, and then select **Next**.
   The destination Historian server page appears.




7. Provide the credentials of the Windows user account of the destination Historian server to which you want Remote Management Agent to connect.

   These details are required for Remote Collector Manager to connect to Historian to manage the collectors remotely. If you are installing collectors on same machine as the Historian server, and if strict collector authentication is disabled, you need not provide these details; by default, the machine name of the local Historian server is considered. If, however, they are installed on different machines, or if strict collector authentication is enabled, you must provide the credentials of the Historian server user.

8. Select **Next**.
   The **You are ready to install** page appears.
9. Select **Install**.
   The installation begins.
10. When you are prompted to reboot your system, select **Yes**.

The collector executable files are installed in the following folder: `<installation drive>:\Program Files (x86)\GE Digital\<collector name>`. The iFIX collectors are installed in the following folder: `C:\Program Files\GE\iFIX`. The following registry paths are created:

- `HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\ GE Digital\iHistorian\Services\<collector type>`
- `HKEY_LOCAL_MACHINE\SOFTWARE\GE Digital\iHistorian\Services\<collector type>`

In addition, if iFIX and/or CIMPLICITY are installed on the same machine as the collectors, instances of the following collectors are created:

- The iFIX collector
- The iFIX Alarms & Events collector
- The OPC Classic Data Access collector for CIMPLICITY
- The OPC Classic Alarms and Events collector for CIMPLICITY

## Installing a Collector at a Command Prompt

After you install collectors and Remote Management Agent, the following artefacts will be available:

- **Executable files:** These files are required to add a collector instance.
- **Instances of the following collectors:**
  - The iFIX collector
  - The iFIX Alarms & Events collector
  - The OPC Classic Data Access collector for CIMPLICITY
  - The OPC Classic Alarms and Events collector for CIMPLICITY

  These instances will be created only if iFIX and/or CIMPLICITY are installed on the same machine as the collectors.
- **The Remote Collector Management agent:** Provides the ability to manage collectors remotely.

Using Configuration Hub, you will then add a collector instance and begin using the collector.
This topic describes how to install collectors at a command prompt. You can also install them using the installer *(on page 143)*.

1. Navigate to the `Collectors` folder in the installation folder.
2. At a command prompt, enter:

   ```
   Collectors_Install.exe -s RootDrive=<value> DestinationServerName=<value>
   DataPath="<value>" UserName1=<value> Password=<value>
   ```

| Parameter | Description | Default Value |
|---|---|---|
| RootDrive | The installation drive for the collectors. | `C:\` |
| DataPath | The folder for storing the collector log files. | `C:\Proficy Historian Data` |
| DestinationServerName | The host name of the destination Historian server to which you want collectors to send data.<br><br>This is required for Remote Collector Manager to connect to Historian to manage the collectors remotely. If you are installing collectors on the same machine as the Historian server, and if strict collector authentication is disabled, you need not provide the server name; by default, the machine name of the local Historian server is considered. If, however, they are installed on different machines, or if strict collector authentication is enabled, you must provide the credentials of the Historian server user. | local host name |
| UserName1 | The username of the Windows user of the destination Historian server. A value is required only if the destination Historian server and collectors are on different machines. | |

| Parameter | Description | Default Value |
|---|---|---|
| Password | The password of the Windows user of the destination Historian server. A value is required only if the destination Historian server and collectors are on different machines. | |

For example: `Collectors_Install.exe -s RootDrive=C:\ DestinationServerName=myservername DataPath="C:\Proficy Historian Data" UserName1=user123 Password=xyz123`

3. Restart the machine. If you uninstall a collector or install another one before restarting the machine, an error may occur.

The collector executable files are installed. In addition, if iFIX and/or CIMPLICITY are installed on the same machine as the collectors, instances of the following collectors are created:

- The iFIX collector
- The iFIX Alarms & Events collector
- The OPC Classic Data Access collector for CIMPLICITY
- The OPC Classic Alarms and Events collector for CIMPLICITY

1. Ensure that the Windows user that you have specified while installing collectors is added to the iH Security Admins and iH Collector Admins groups.
2. Enable trust for a client certificate for Configuration Hub.
3. Enable trust for a self-signed certificate on Chrome *(on page 97)*.
4. Import an issuer certificate.

You are now ready to use Configuration Hub. To add and manage collector instances, you can use Configuration Hub or Remote Collector Management. For instructions specific to setting up the iFIX collector and the iFIX Alarms and Events collector, refer to Working with iFIX Collectors.

# Upgrade Collectors

- Ensure that you are attempting to upgrade the collectors that were installed using the Historian Installation Package. If you previously installed the collectors using the Proficy Installer, such as the iFIX Proficy Installer, you must upgrade the collectors using the Proficy Installer only.
- If an iFIX collector instance created in version 9.0 exists, after you upgrade collectors, another instance of the iFIX collector is created. Because of this, the Remote Collector Manager (RCM)

will not work correctly. Therefore, if you want to use RCM, you must delete one of the instances. If needed, you can manually create another instance of the iFIX collector using Configuration Hub or the RemoteCollectorConfigurator utility. This is applicable to the iFIX Alarms and Events collector as well.

- For collectors earlier than version 7.1, additional registries that you create manually are deleted. Therefore, we recommend that you back up them, uninstall the collectors, and then install the latest version.

Install the collectors *(on page 142)*.
The collectors are upgraded to the latest version.

# Sending Data to Cloud

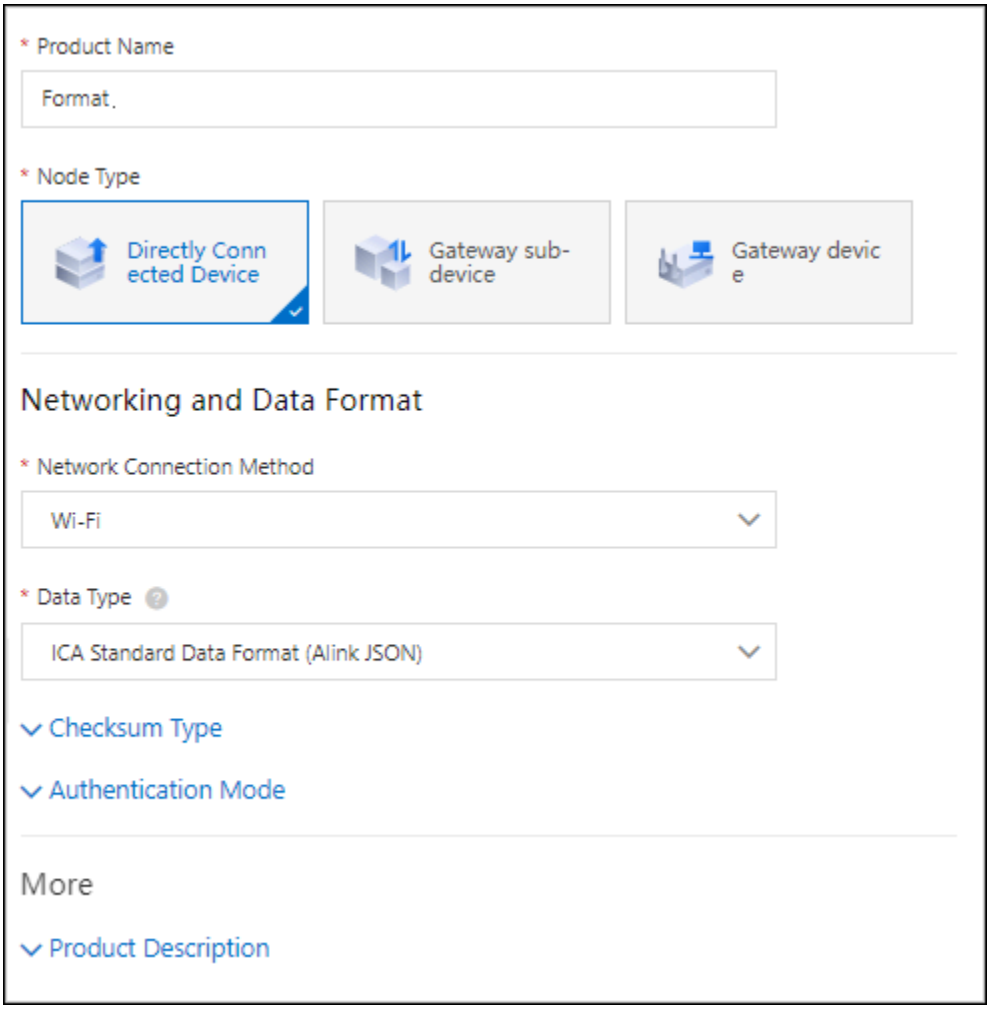
## Send Data to Alibaba Cloud

Generate a password using the utility. While generating the password, use the same algorithm that you will use to connect to Alibaba Cloud.

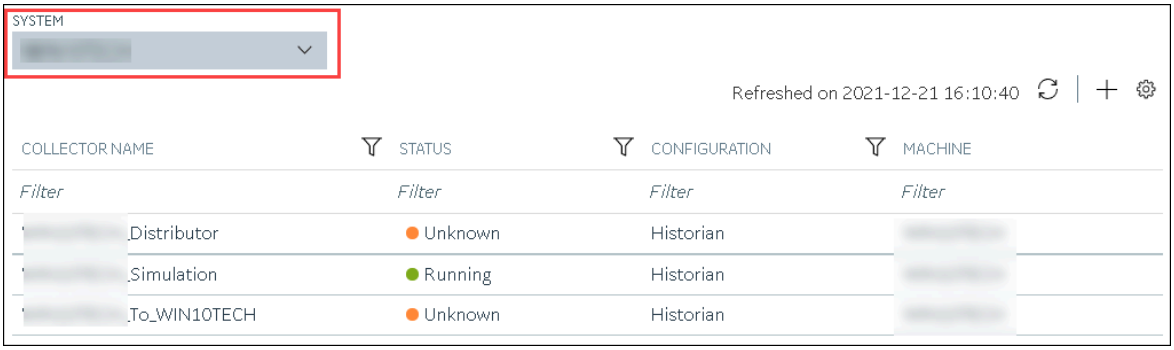
To send data to Alibaba Cloud, you can choose any of the following collectors:

- The iFIX collector
- The MQTT collector
- The ODBC collector
- The OPC Classic DA collector
- The OPC Classic HDA collector
- The OPC UA DA collector
- The OSI PI collector
- The Python Collector
- The Server-to-Server collector
- The Simulation collector
- The Windows Performance collector
- The Wonderware collector

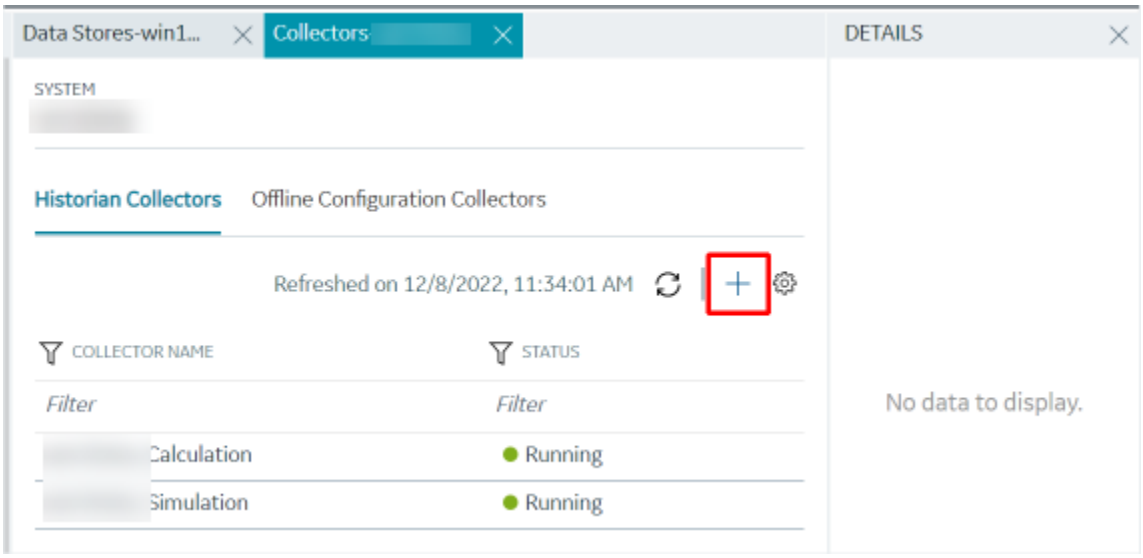
1. Access Alibaba IoT Platform console.
2. Create a product. When you do so:
    - In the **Node Type** field, select **Directly Connected Device**.
    - In the **Network Connection Method** field, select **Wi-Fi**.
    - In the **Data Type** field, select **ICA Standard Data Format**.




3. Note down the region ID for the region you have selected. For a list of region IDs, refer to https://www.alibabacloud.com/help/doc-detail/40654.htm.
4. Access the product certificate, and note down the product secret and product key values.
5. Create a device.
6. Access Configuration Hub *(on page 336)*.
7. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**. A list of collectors in the system appears.
8. If needed, select the system in which you want to add a collector instance.

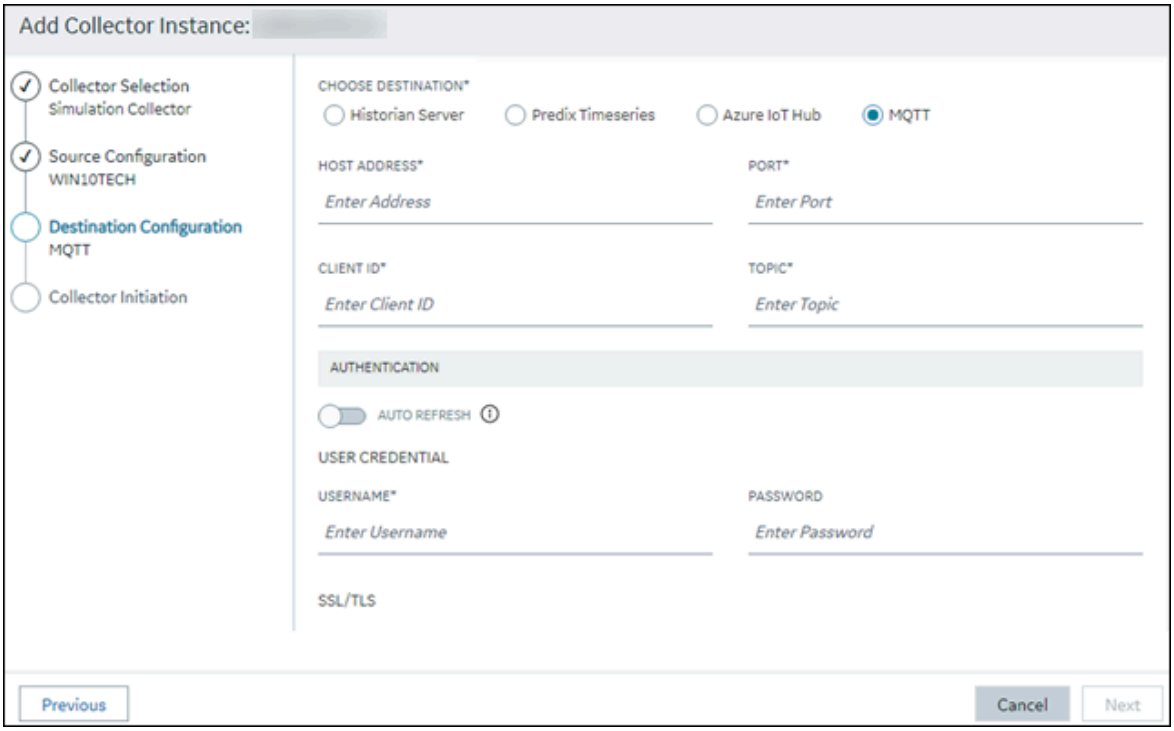



9. In the upper-right corner of the main section, select +.




   The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.

10. In the **COLLECTOR TYPE** field, select a collector type, and then select **Get Details**.
    The **INSTALLATION DRIVE** and **DATA DIRECTORY** fields are disabled and populated.
11. Select **Next**.
    The **Source Configuration** section appears, populating the hostname of the collector machine.
12. As needed, enter values in the available fields, and then select **Next**.
    The **Destination Configuration** section appears.

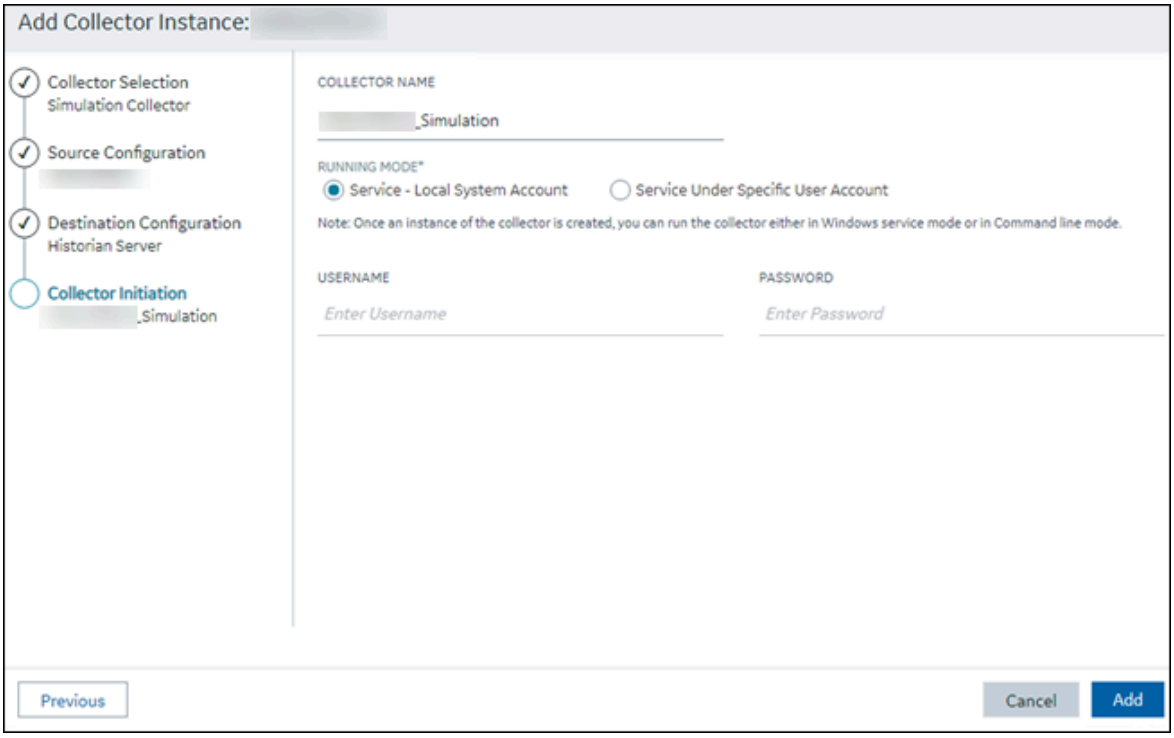



13. In the **CHOOSE DESTINATION** field, select **MQTT**, and then provide values as described in the following table.

| Field | Description |
|---|---|
| HOST ADDRESS | Enter a value in the following format: `<product name>.iot-as-mqtt.<region ID>.aliyuncs.com`. A value is required.<br><br>For example: `a23dr53dwrt.iot-as-mqtt.cn-shanghai.aliyuncs-.com` |
| PORT | Enter `1883`. A value is required. |
| CLIENT ID | Enter a value in the following format: `<device name>\|securemode=<value>,signmethod=<algorithm name>`. A value is required.<br>◦ For securemode, enter `2` for direct TLS connection, or enter `3` for direct TCP connection.<br>◦ For signmethod, specify the signature algorithm that you want to use. Valid values are hmacmd5, hmacsha1, hmacsha256, and sha256. You must use the same algorithm to generate the password.<br><br>For example: `MyDevice\|securemode=3,signmethod=hmacsha1` |

| Field | Description |
|---|---|
| TOPIC | Enter a value in the following format: `/sys/<product name>/<device name>/thing/event/property/post`. A value is required.<br><br>For example: `/sys/a23dr53dwrt/MyDevice/thing/event/property/post` |
| USERNAME | Enter a value in the following format: `<device name><product name>`. A value is required.<br><br>For example: `MyDevicea23dr53dwrt` |
| PASSWORD | Enter the password that you have generated. A value is required. |
| CHOOSE CONFIGURATION | Select the type of the configuration to specify the tags whose data you want to collect. Select one of the following options:<br>◦ **Historian Configuration**: Select this option if you want to add the tags manually using Historian Administrator *(on page 659)*. If you select this option, the **CONFIGURATION HISTORIAN SERVER** field appears.<br>◦ **Offline Configuration**: Select this option if you want to provide the tag names using the offline configuration *(on page 2039)* file instead of adding tags manually. By default, this file is located in the following location: `<installation folder of Historian>\GE Digital\<collector name>` |

14. Select **Next**.
    The **Collector Initiation** section appears.




15. Enter a collector name.

    The value that you enter:

    - Must be unique.
    - Must not exceed 15 characters.
    - Must not contain a space.
    - Must not contain special characters except a hyphen, period, and an underscore.

16. In the **RUNNING MODE** field, select one of the following options.

    - **Service - Local System Account**: Select this option if you want to run the collector as a Windows service using the credentials of the local user (that is, the currently logged-in user). If you select this option, the **USERNAME** and **PASSWORD** fields are disabled.
    - **Service Under Specific User Account**: Select this option if you want to run the collector as a Windows service using a specific user account. If you select this option, you must enter values in the **USERNAME** and **PASSWORD** fields. If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:
        - iH Security Admins
        - iH Collector Admins
        - iH Tag Admins

    If you choose the first option, you can also configure the collector to start automatically when you start the computer, or, in the case of iFIX collectors, whenever you start iFIX.

17. Select **Add**.
    The collector instance is created.
18. Specify the tags for which you want to collect data.
    - If you have selected **Historian Configuration** in the **CHOOSE CONFIGURATION** field, specify the tags manually *(on page 357)*.
    - If you have selected **Offline Configuration** in the **CHOOSE CONFIGURATION** field, specify the tags using the offline configuration file *(on page 2039)*.

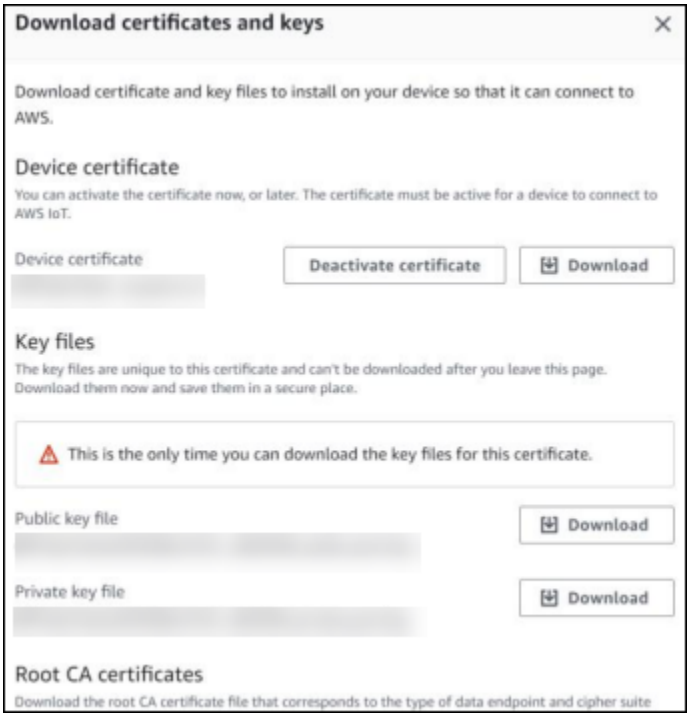
    The collector begins sending Historian data to the device that you have created.

## Send Data to AWS IoT Core

To send data to an AWS IoT Code, you can choose any of the following collectors:

- The iFIX collector
- The MQTT collector
- The ODBC collector
- The OPC Classic DA collector
- The OPC Classic HDA collector
- The OPC UA DA collector
- The OSI PI collector
- The Python Collector
- The Server-to-Server collector
- The Simulation collector
- The Windows Performance collector
- The Wonderware collector

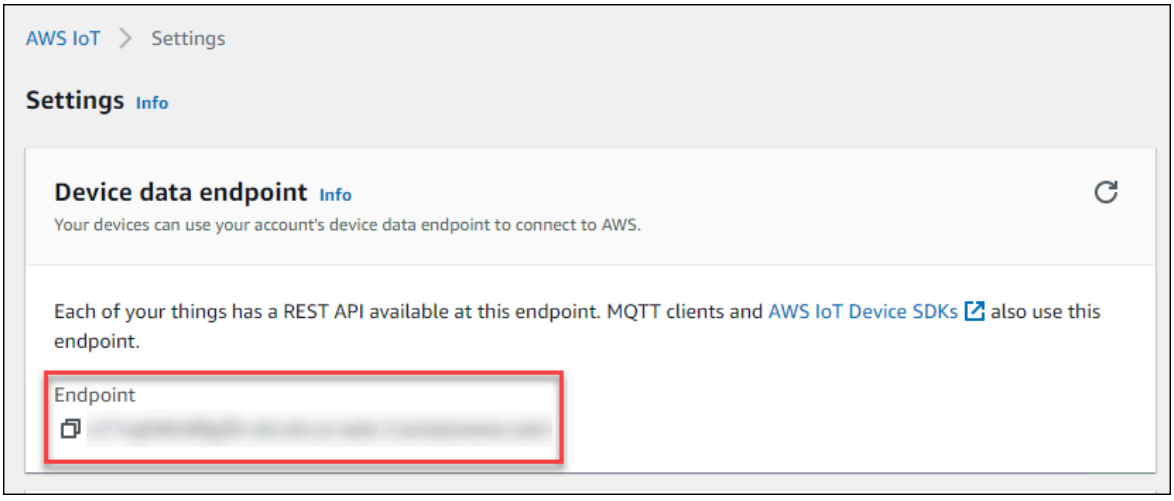
1. Access the **AWS Management Console** page.
2. Search and select **IoT Core**.
   The **AWS IoT** page appears.
3. Create a policy allowing the permissions that you want to grant on your device (for example, iot:Connect, iot:Publish, iot:Subscribe, iot:Receive). For the resource, provide the topic name. If, however, you want to use all topics, enter *.
4. Create a thing, linking it with the policy that you have created.
5. Download the certificates and key files for the device to communicate. In addition, download the root CA certificate.




**Important:**

This is mandatory, and it is the only time you can download the certificates.

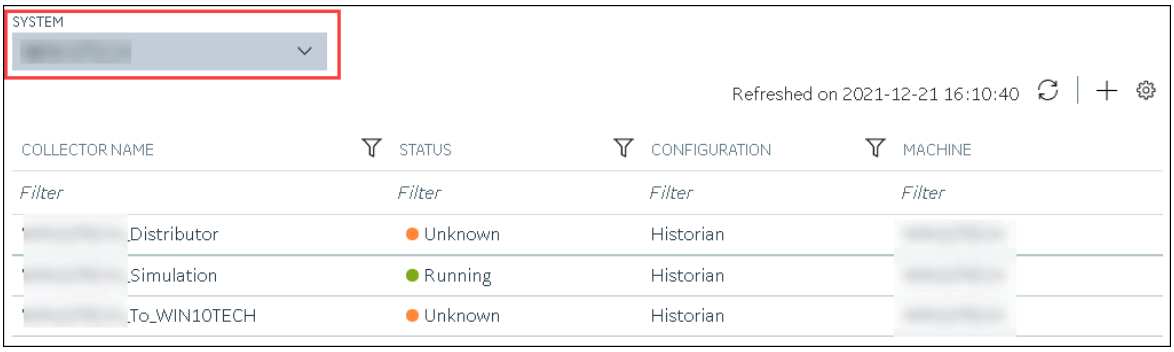
6. In the left navigation pane, select **Settings**.
7. Make a note of the endpoint that appears.



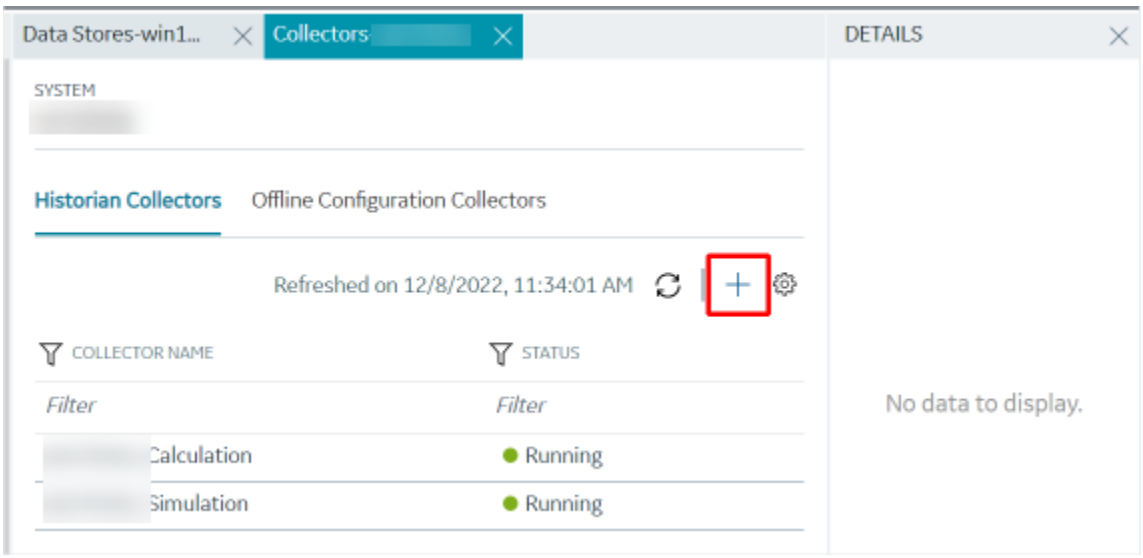

8. Access Configuration Hub *(on page 336)*.
9. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.

   A list of collectors in the system appears.

10. If needed, select the system in which you want to add a collector instance.

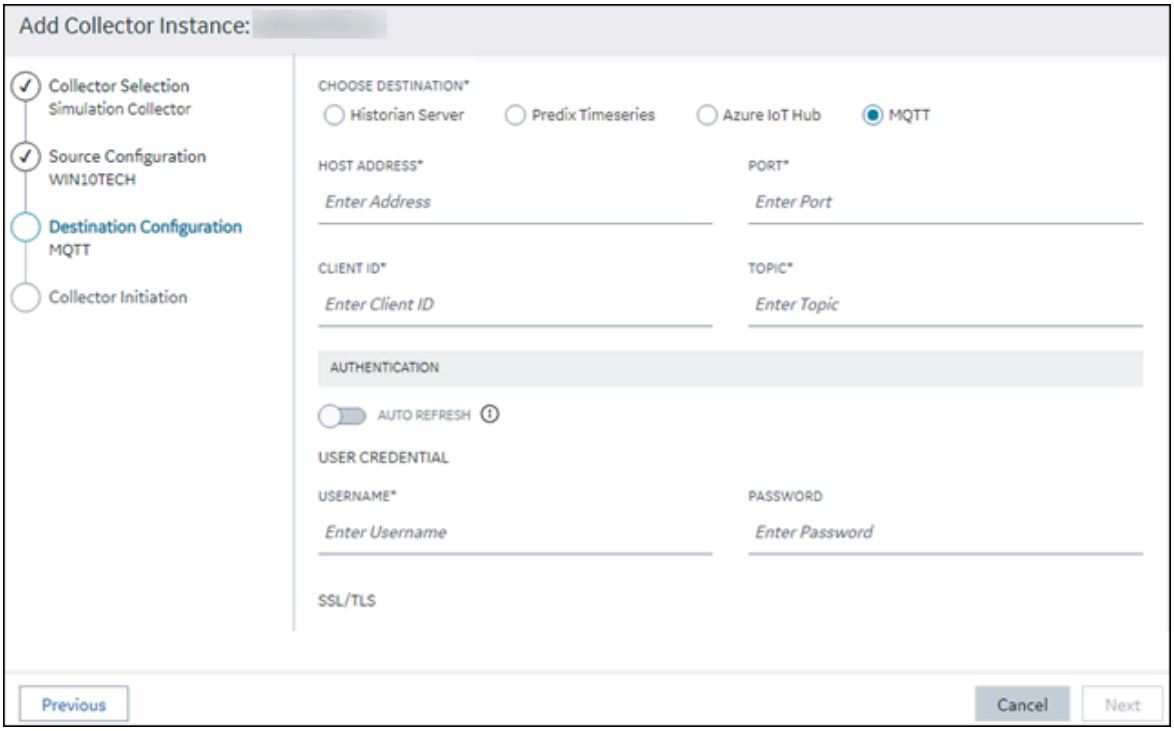



11. In the upper-right corner of the main section, select +.




The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.

12. In the **COLLECTOR TYPE** field, select a collector type (except the File collector and the Server-to-Server collector), and then select **Get Details**.
    The **INSTALLATION DRIVE** and **DATA DIRECTORY** fields are disabled and populated.
13. Select **Next**.
    The **Source Configuration** section appears, populating the hostname of the collector machine.
14. As needed, enter values in the available fields, and then select **Next**.
    The **Destination Configuration** section appears.



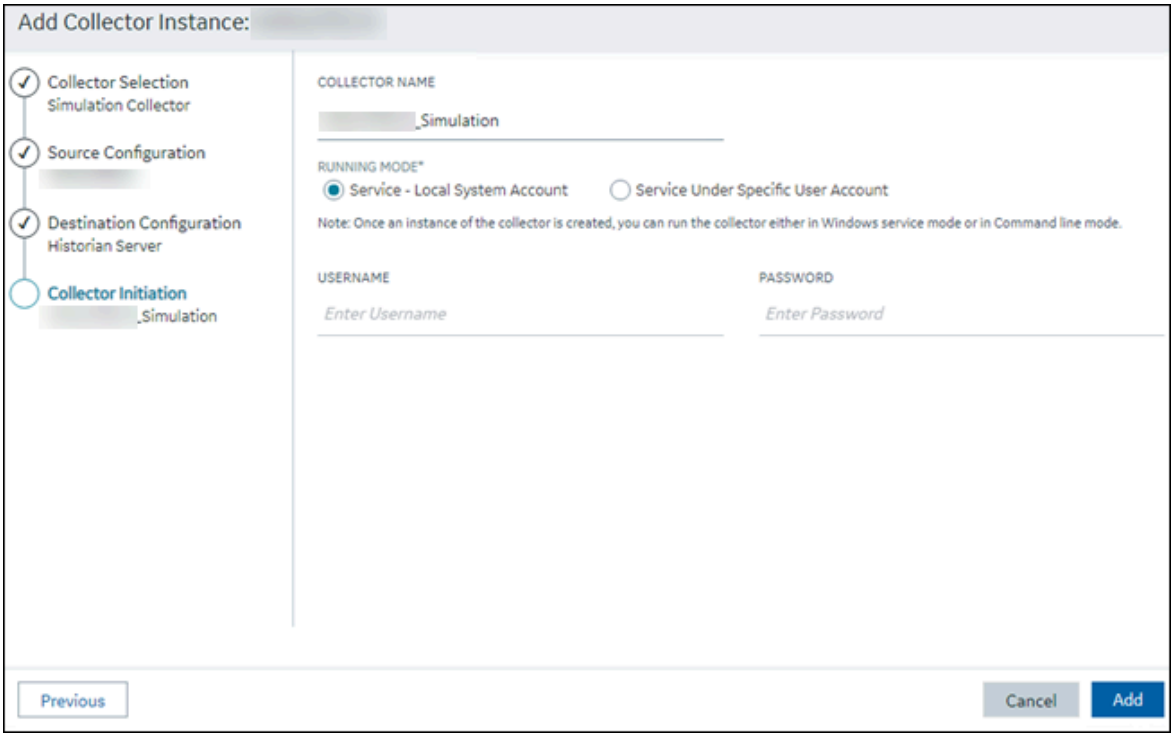

15. In the **CHOOSE DESTINATION** field, select **MQTT**, and then provide values as described in the following table.

| Field | Description |
|---|---|
| HOST ADDRESS | Enter the endpoint that you have noted down. A value is required. |
| PORT | Enter 8883. A value is required. |
| CLIENT ID | Enter the thing name. A value is required. |
| TOPIC | Enter the MQTT topic to which you want the collector to publish data. A value is required. For information on topic names, refer to https://docs.aws.amazon.com/iot/latest/developer-guide/topics.html. |
| USERNAME | Enter any value. Since we will use a certificate-based authentication, username and password will not be used; however, you must still enter a value. |

| Field | Description |
|---|---|
| PASSWORD | Enter any value. Since we will use a certificate-based authentication, username and password will not be used; however, you must still enter a value. |
| CA SERVER ROOT FILE | Enter the path of the root CA certificate file that you have downloaded. |
| CLIENT CERTIFICATE | Enter the path of the device certificate that you have downloaded. |
| PRIVATE KEY FILE | Enter the path of the private key file that you have downloaded. |
| PUBLIC KEY FILE | Enter the path of the public key file that you have downloaded. |
| CHOOSE CONFIGURATION | Select the type of the configuration to specify the tags whose data you want to collect. Select one of the following options:<br>◦ **Historian Configuration**: Select this option if you want to add the tags manually using Historian Administrator *(on page 659)*. If you select this option, the **CONFIGURATION HISTORIAN SERVER** field appears.<br>◦ **Offline Configuration**: Select this option if you want to provide the tag names using the offline configuration *(on page 2039)* file instead of adding tags manually. By default, this file is located in the following location: `<installation folder of Historian>\GE Digital\<collector name>` |

16. Select **Next**.

    The **Collector Initiation** section appears.




17. Enter a collector name.

    The value that you enter:

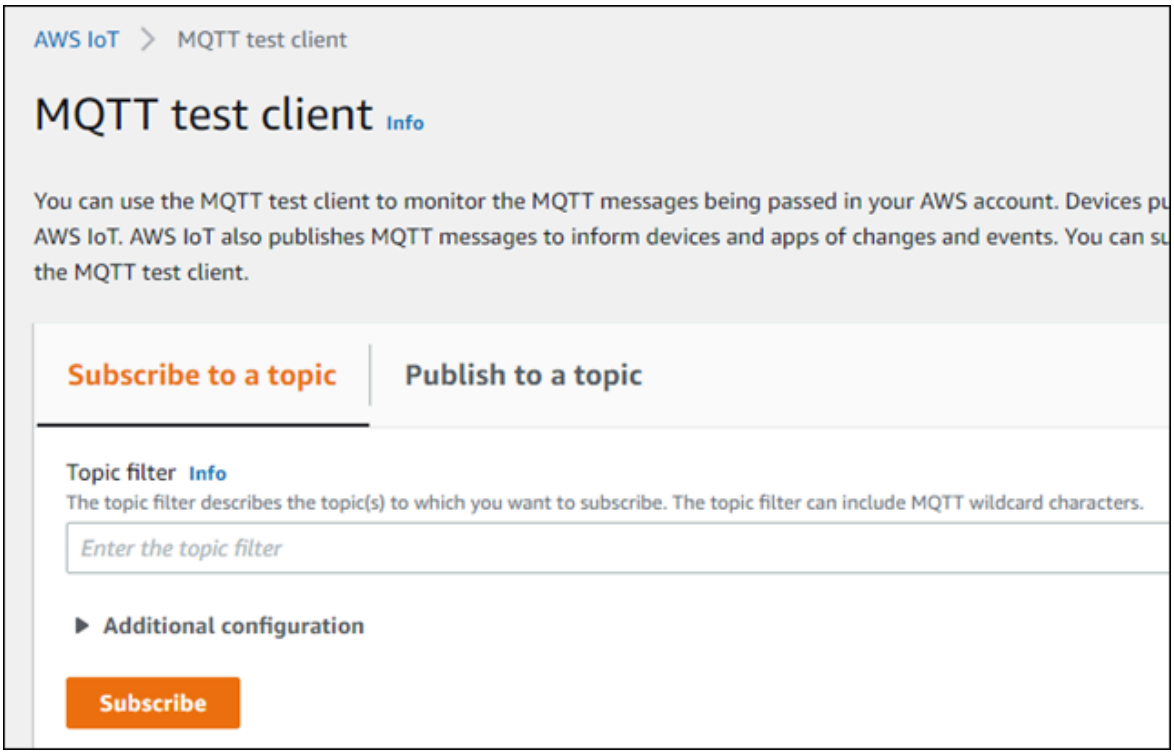
    - Must be unique.
    - Must not exceed 15 characters.
    - Must not contain a space.
    - Must not contain special characters except a hyphen, period, and an underscore.

18. In the **RUNNING MODE** field, select one of the following options.
    - **Service - Local System Account**: Select this option if you want to run the collector as a Windows service using the credentials of the local user (that is, the currently logged-in user). If you select this option, the **USERNAME** and **PASSWORD** fields are disabled.
    - **Service Under Specific User Account**: Select this option if you want to run the collector as a Windows service using a specific user account. If you select this option, you must enter values in the **USERNAME** and **PASSWORD** fields. If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:
      - iH Security Admins
      - iH Collector Admins
      - iH Tag Admins

    If you choose the first option, you can also configure the collector to start automatically when you start the computer, or, in the case of iFIX collectors, whenever you start iFIX.

19. Select **Add**.
    The collector instance is created.
20. Specify the tags for which you want to collect data.
    - If you have selected **Historian Configuration** in the **CHOOSE CONFIGURATION** field, specify the tags manually *(on page 357)*.
    - If you have selected **Offline Configuration** in the **CHOOSE CONFIGURATION** field, specify the tags using the offline configuration file *(on page 2039)*.

    The collector begins sending Historian data to the thing that you have created.
21. Access AWS IoT Core, and in the left pane, select **Test**.
    The **MQTT test client** page appears.




22. Subscribe to the topic to which the collector is publishing data, and then select **Subscribe**.
    The messages received from the topic appear, indicating that the collector is sending data to the AWS IoT device.
    AWS supports a payload of maximum 128 KB. Therefore, if the message size is greater than 128 KB, create a registry key named CloudMaxSamplesPerMsg for the collector instance, and decrease the value to 700 or less. If, however, you want to send more data in a message, we recommend that you create another collector instance and send data to another thing resource in AWS.

> **Tip:**
> To find out the message size, modify the collector instance *(on page 634)* and set the log level to 3 or more.

23. Create a VPC destination or an HTTP destination for the messages.
24. Monitor the data that you have collected.

## Send Data to Azure Cloud in the Key-Value Format

To send data to an Azure IoT Hub device, you can choose any of the following collectors:

- The iFIX collector
- The MQTT collector
- The ODBC collector
- The OPC Classic DA collector
- The OPC Classic HDA collector
- The OPC UA DA collector
- The OSI PI collector
- The Python Collector
- The Server-to-Server collector
- The Simulation collector
- The Windows Performance collector
- The Wonderware collector

This topic describes how to send data in the key-value format. In this format, the message size is bigger because names of the tag properties are repeated. However, it provides clarity to novice users. For example: `{"body":`

```
[{"tagname":"Azure_Iot_simulation_tag_1","epochtime":1629730936000,"tagvalue":7129.124023438,"quality":3},
{"tagname":"Azure_Iot_simulation_tag_2","epochtime":1629730936000,"tagvalue":123.3738924567,"quality":3}] ,"messa
```

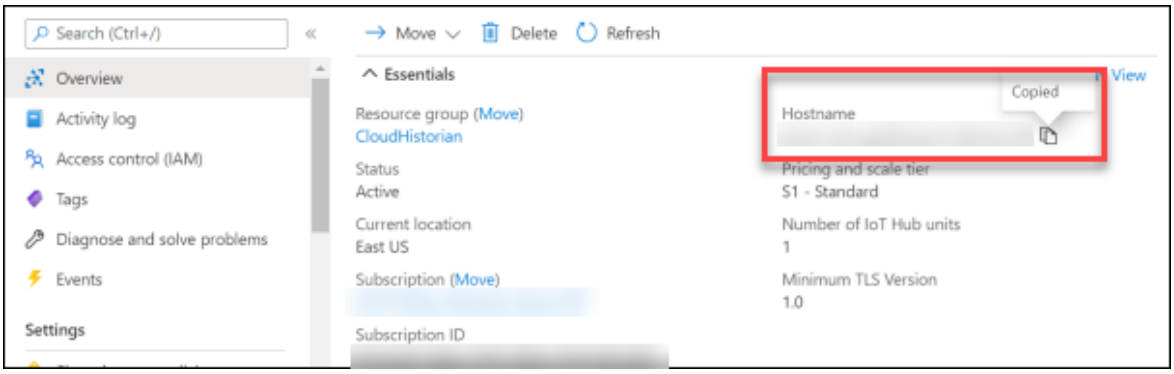
You can also send data in the KairosDB format *(on page 608)*.

> **Note:**
> Data in Azure IoT Hub is stored for maximum seven days, after which it is deleted from the hub. Therefore, you must consume the data within seven days. Based on your requirement, you can store it in a relevant Azure storage. You can then use Azure functions or streaming analytics to analyse the data.

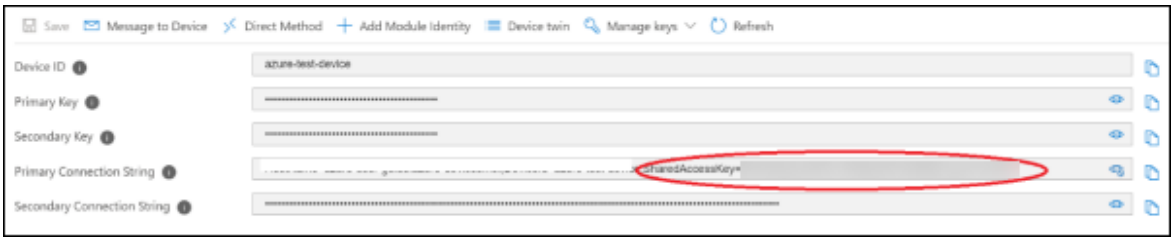
1. Create Azure IoT Hub.

> **Tip:**
> To choose the correct Azure IoT Hub tier based on your data throughput, refer to https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-devguide-quotas-throttling. For guidance on choosing the appropriate subscription, refer to https://azure.microsoft.com/en-us/pricing/details/iot-hub/

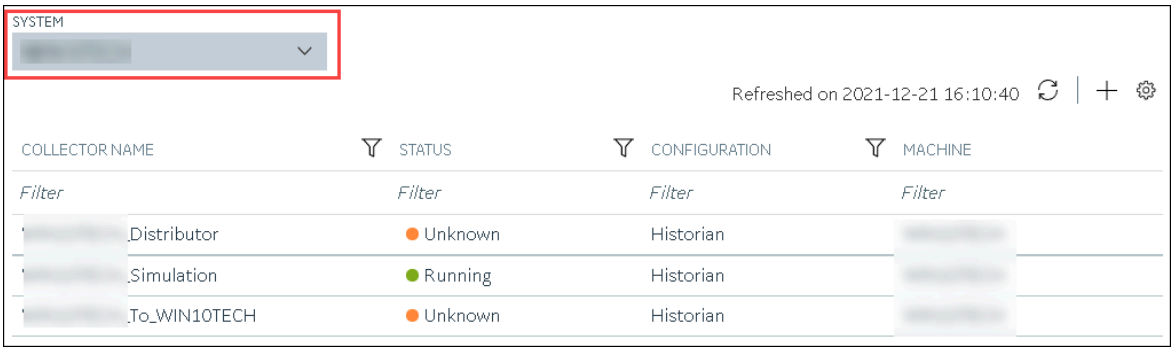
2. After you create Azure IoT Hub, select **Go to resource**, and then note down the hostname:



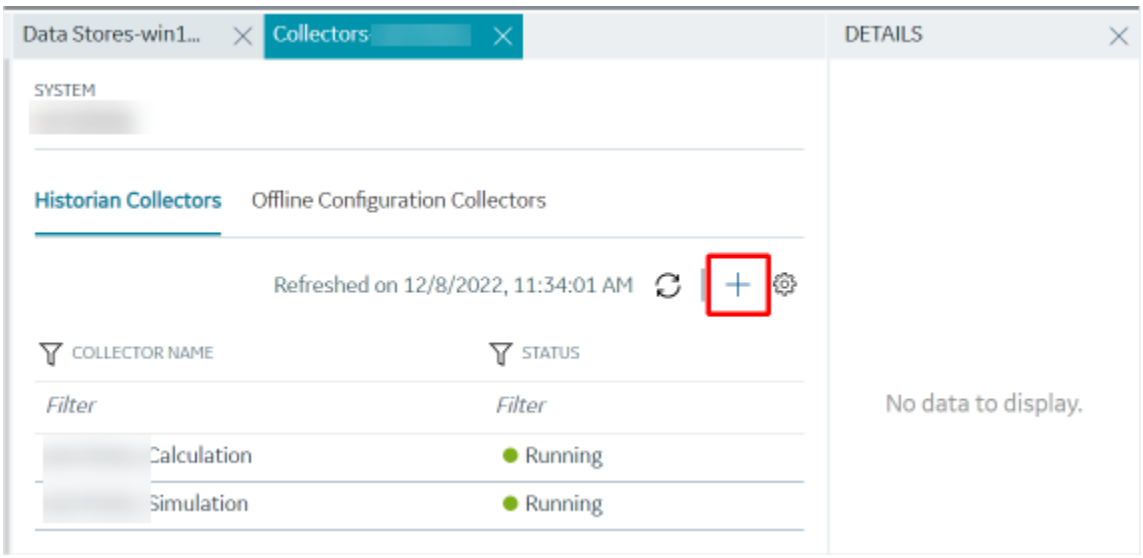

3. Create devices in Azure IoT Hub to group related tag information; thus mapping a collector instance to a device. We recommend that you create one device per collector instance. Ensure that the device is running.
   When you create a device, use the following guidelines to choose the authentication type:
   - **Symmetric Key**: Select this option if you want to use a Shared Access Signature (SAS) authentication.
   - **X.509 Self-Signed**: Select this option if you want to create self-signed certificates using OpenSSL. We recommend that you use these certificates only for testing purposes. For instructions, refer to https://docs.microsoft.com/en-us/azure/iot-hub/tutorial-x509-self-sign.
   - **X.509 CA Signed**: Select this option if you want to use CA-signed certificates
4. If you have selected **Symmetric Key** in the previous step, select the link in the **Device ID** column, and note down the shared access key value.



5. Access Configuration Hub *(on page 336)*.
6. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.
   A list of collectors in the default system appears.
7. If needed, select the system in which you want to add a collector instance.
8. If needed, select the system in which you want to add a collector instance.

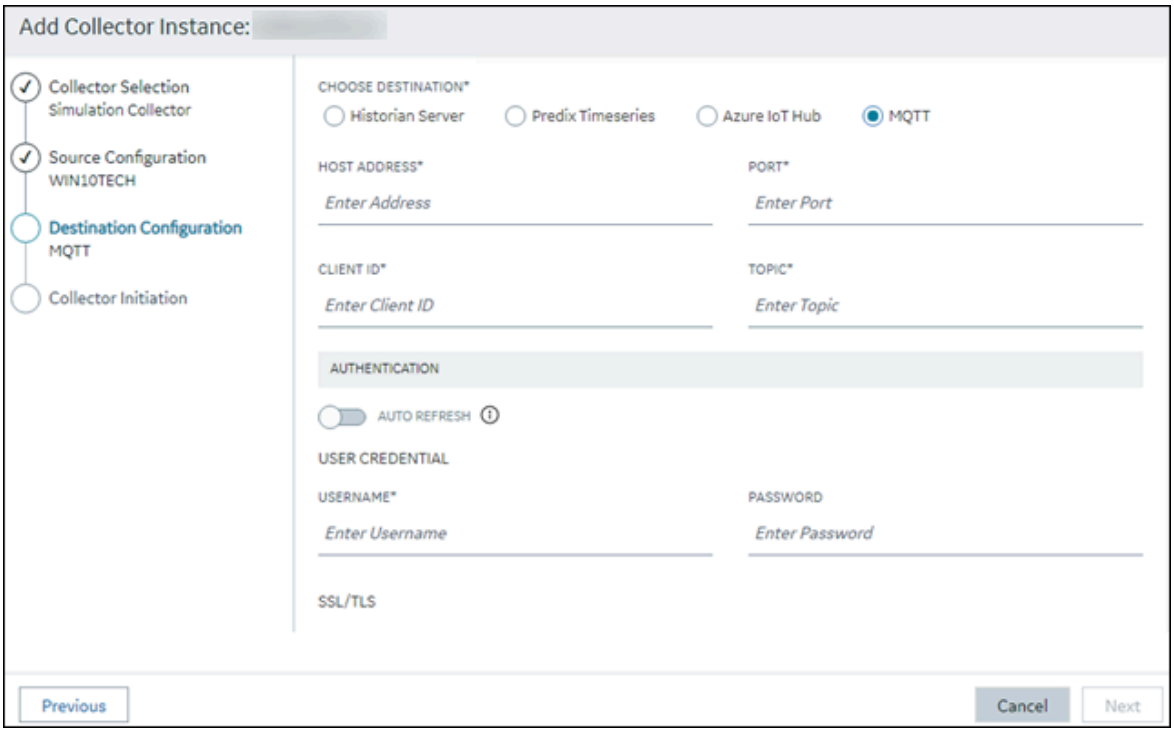



9. In the upper-right corner of the main section, select +.




   The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.
10. In the **COLLECTOR TYPE** field, select a collector type, and then select **Get Details**.
    The **INSTALLATION DRIVE** and **DATA DIRECTORY** fields are disabled and populated.
11. Select **Next**.
    The **Source Configuration** section appears.
12. As needed, enter values in the available fields.
13. Select **Next**.

The **Destination Configuration** section appears.




14. Select **MQTT**, and provide values as described in the following table.

| Field | Description |
|---|---|
| **HOST ADDRESS** | Enter the host name of the resource that you have noted down in step 2. A value is required and must be in the following format: `<Azure IoT Hub name>.azure-devices.net` |
| **PORT** | Enter `8883`. |
| **CLIENT ID** | Enter the ID of the device that you created in step 3. A value is required and must be unique for an MQTT broker. |
| **TOPIC** | Enter `devices/<device ID>/messages/events`. |
| **AUTO REFRESH** | Indicates whether you want to automatically create/refresh the SAS authentication token when it expires. |

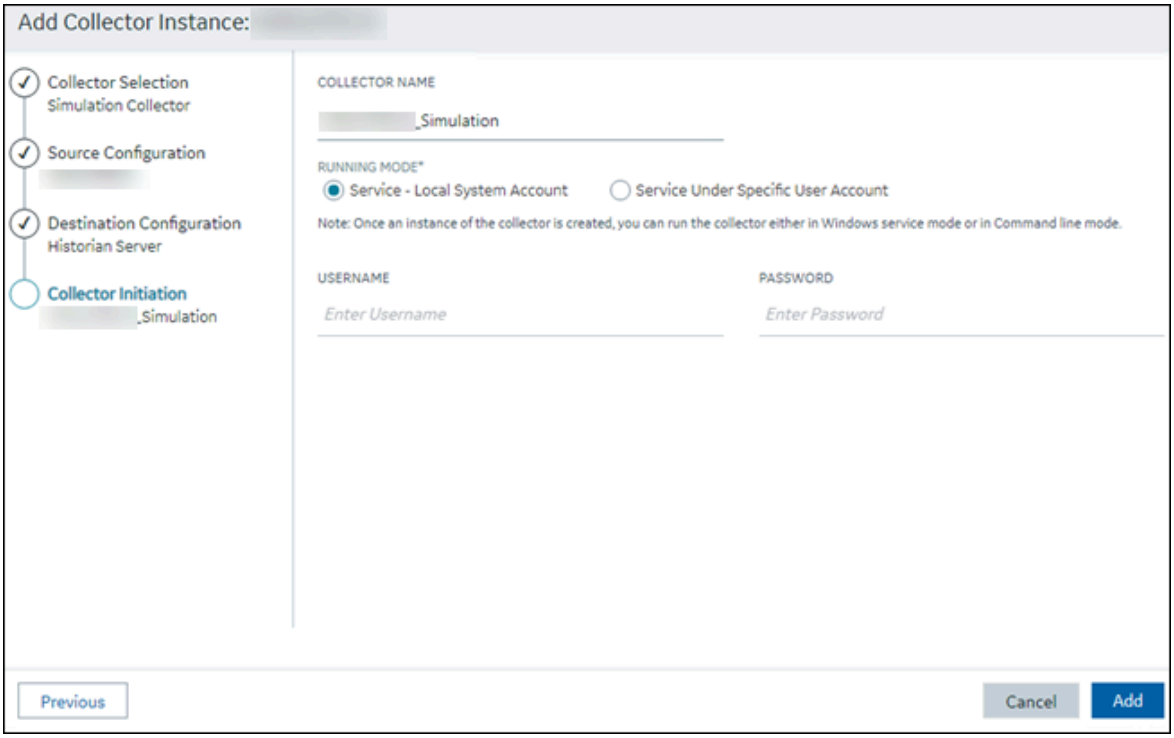
| Field | Description |
|---|---|
| | ◦ If you switch the toggle off, you must manually provide the token as soon as it expires.<br>◦ If you switch the toggle on, you must provide the shared access key that you have noted down in step 4. And, you can leave the **PASSWORD** field blank.<br>This is applicable only if you have selected **Symmetric Key** in step 3. |
| **USERNAME** | Enter a value in the following format: `<host name or IP address>/<device ID>/?api-version=2018-06-30` |
| **PASSWORD** | Enter the SAS token. This is applicable only if you have selected **Symmetric Key** in step 3 and if you have switched off the **AUTO REFRESH** toggle.<br><br>For instructions on generating a SAS token, refer to https://docs.microsoft.com/en-us/azure/cognitive-services/translator/document-translation/create-sas-tokens?tabs=Containers. |
| **DEVICE SHARED KEY** | Enter the shared access key value that you noted down in step 4. A value is required. This is applicable only if you have selected **Symmetric Key** in step 3 and if you have switched the **AUTO REFRESH** toggle on. |
| **CA SERVER ROOT FILE** | Enter the path of the CA server root file that you want to use. You can find the file here: https://github.com/Azure-Samples/IoTMQTTSample/blob/master/IoTHubRootCA_Baltimore.pem. |
| **CLIENT CERTIFICATE** | Enter the path to the client certificate. A value is required. This is applicable only if you have selected one of these options in step 3: |

| Field | Description |
|---|---|
| | ◦ **X.509 Self-Signed**: If you have selected this option, you can generate the certificate using OpenSSL.<br>◦ **X.509 CA Signed**: If you have selected this option, you would receive the certificate from CA. |
| **PRIVATE KEY FILE** | Enter the complete path to the private key file. A value is required. This is applicable only if you have selected one of these options in step 3:<br>◦ **X.509 Self-Signed**: If you have selected this option, you can generate the key file using OpenSSL.<br>◦ **X.509 CA Signed**: If you have selected this option, you would receive the key file from CA. |
| **PUBLIC KEY FILE** | Enter the path to the public key file. This is applicable only if you have selected one of these options in step 3:<br>◦ **X.509 Self-Signed**: If you have selected this option, you can generate the key file using OpenSSL.<br>◦ **X.509 CA Signed**: If you have selected this option, you would receive the key file from CA. |
| **CHOOSE CONFIGURATION** | The type of the configuration to specify the tags whose data you want to collect. Select one of the following options:<br>◦ **Historian Configuration**: Select this option if you want to add the tags manually *(on page 357)*. If you select this option, the **CONFIGURATION HISTORIAN SERVER** field appears.<br>◦ **Offline Configuration**: Select this option if you want to provide the tag names using the offline configuration *(on page* |

| Field | Description |
|---|---|
| | *2039)* file instead of adding tags manually. By default, this file is located in the following location: <*installation folder of Historian*>\GE Digital\<*collector name*> |
| **CONFIGURATION HISTORIAN SERVER** | The host name of the machine from which you want to access Historian Administrator to add the tags manually for the collector. This field appears only if you have selected **Historian Configuration** in the **CHOOSE CONFIGURATION** field. |

15. Select **Next**.

    The **Collector Initiation** section appears.




16. Enter a collector name.

    The value that you enter:

    - Must be unique.
    - Must not exceed 15 characters.

- Must not contain a space.
- Must not contain special characters except a hyphen, period, and an underscore.

17. In the **RUNNING MODE** field, select one of the following options.
    - **Service - Local System Account**: Select this option if you want to run the collector as a Windows service using the credentials of the local user (that is, the currently logged-in user). If you select this option, the **USERNAME** and **PASSWORD** fields are disabled.
    - **Service Under Specific User Account**: Select this option if you want to run the collector as a Windows service using a specific user account. If you select this option, you must enter values in the **USERNAME** and **PASSWORD** fields. If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:
        - iH Security Admins
        - iH Collector Admins
        - iH Tag Admins

    If you choose the first option, you can also configure the collector to start automatically when you start the computer, or, in the case of iFIX collectors, whenever you start iFIX.
18. Select **Add**.
    The collector instance is created.
19. Specify the tags for which you want to collect data.
    - If you have selected **Historian Configuration** in the **CHOOSE CONFIGURATION** field, specify the tags manually *(on page 357)*.
    - If you have selected **Offline Configuration** in the **CHOOSE CONFIGURATION** field, specify the tags using the offline configuration file *(on page 2039)*.

    The collector begins sending Historian data to the Azure IoT Hub device that you have created.

## Send Data to Google Cloud

1. Download the Google root CA certificate from https://pki.google.com/roots.pem.
2. Create public/private key pairs. Use OpenSSL only for testing purposes.

To send data to a Google Cloud device, you can choose any of the following collectors:

- The iFIX collector
- The MQTT collector
- The ODBC collector
- The OPC Classic DA collector
- The OPC Classic HDA collector

- The OPC UA DA collector
- The OSI PI collector
- The Python Collector
- The Server-to-Server collector
- The Simulation collector
- The Windows Performance collector
- The Wonderware collector

1. Access Google Cloud Platform.
2. Create a project. Note down the project ID.
3. Create a registry.

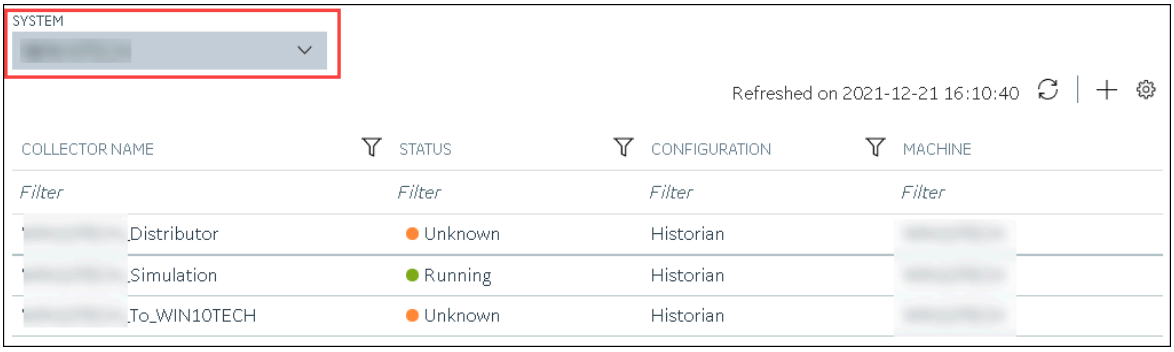
   When you create the registry:
   - Use the MQTT protocol.
   - You can choose to provide a CA certificate.

   Note down the registry ID and the region values.
4. Add a device to the registry.

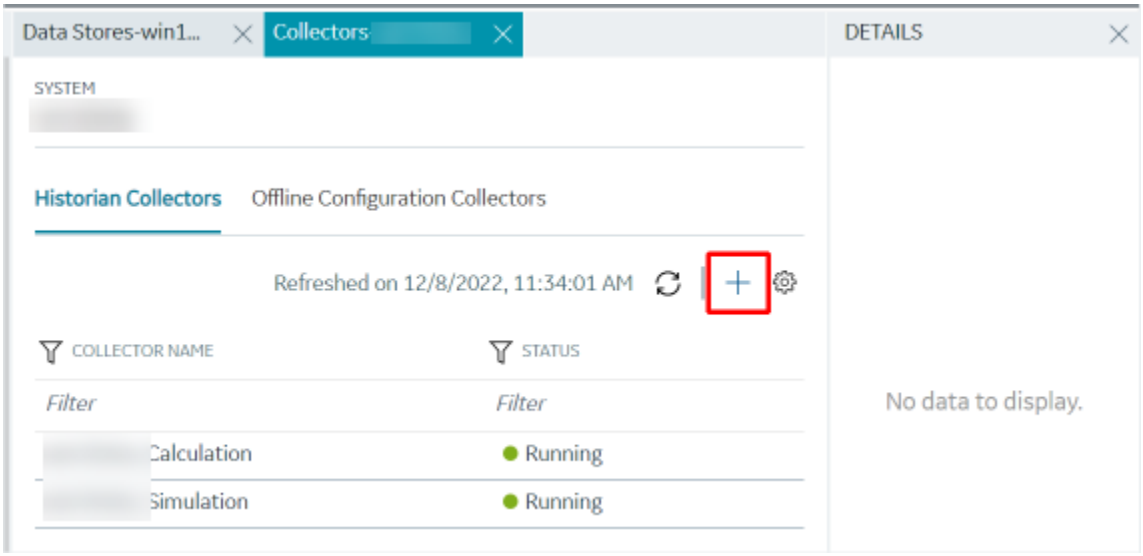
   When you add the device:
   - Allow device communication.
   - Upload the public key or enter the details manually.

   Note down the device ID.
5. Access Configuration Hub *(on page 336)*.
6. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.

   A list of collectors in the default system appears.
7. If needed, select the system in which you want to add a collector instance.
8. If needed, select the system in which you want to add a collector instance.

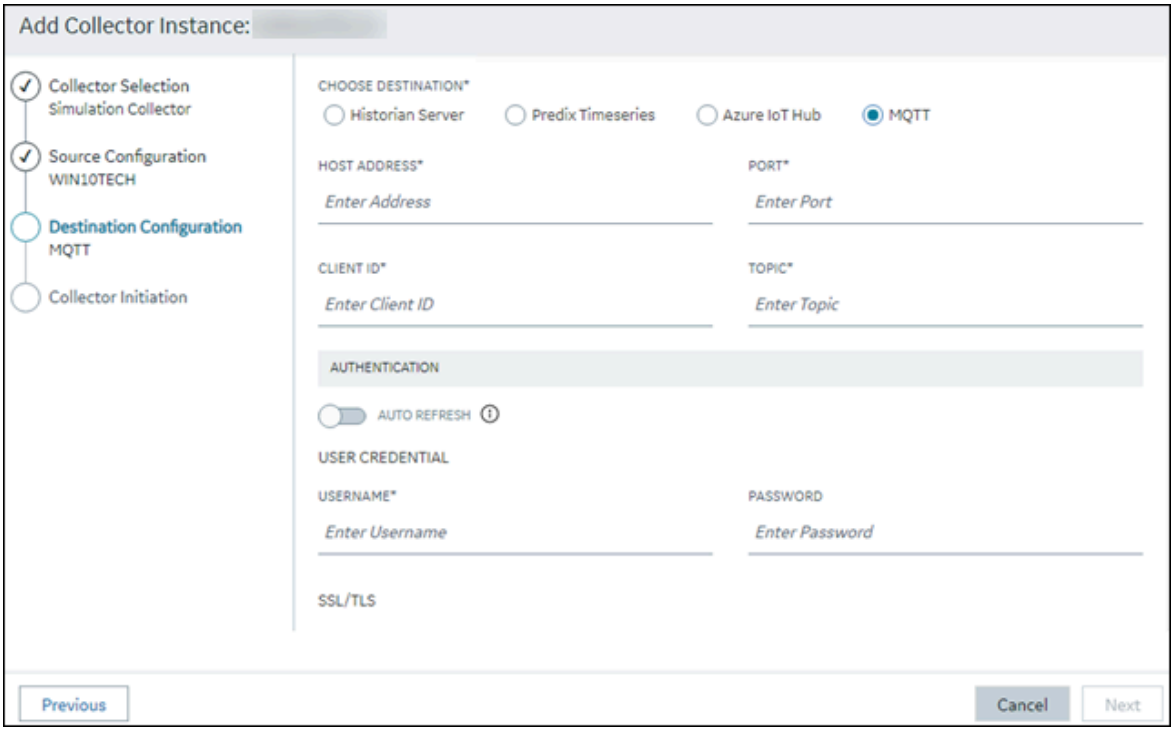



9. In the upper-right corner of the main section, select +.




The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.

10. In the **COLLECTOR TYPE** field, select a collector type, and then select **Get Details**.
    The **INSTALLATION DRIVE** and **DATA DIRECTORY** fields are disabled and populated.
11. Select **Next**.
    The **Source Configuration** section appears.
12. As needed, enter values in the available fields.
13. Select **Next**.
    The **Destination Configuration** section appears.




14. Select **MQTT**, and provide values as described in the following table.

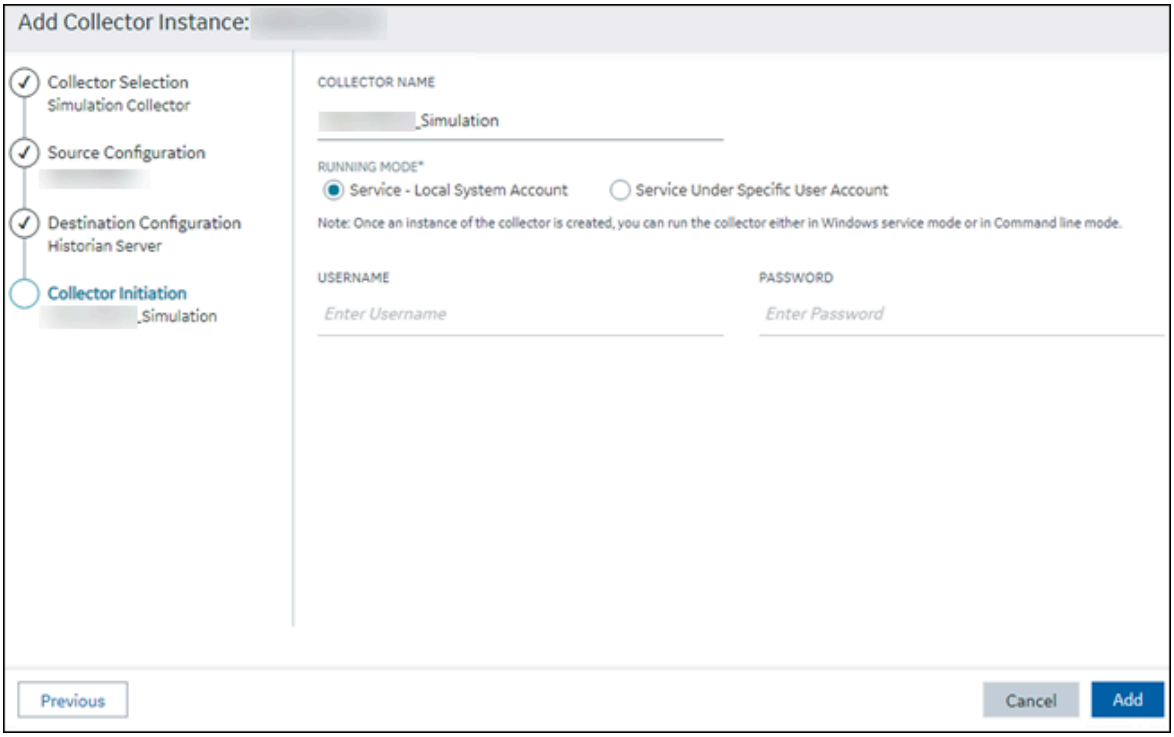
| Field | Description |
|---|---|
| **HOST ADDRESS** | Enter `mqtt.googleapis.com`. A value is required. |
| **PORT** | Enter `8883` or `443`. |
| **CLIENT ID** | Enter the ID of the device that you created in the following format: `projects/<project ID>/locations/<cloud region>/registries/<registry ID>/devices/<device ID>`. For example: `projects/mygcpproject/locations/asia-east1/registries/testmqttgcpiot/devices/gcptesting`<br><br>A value is required and must be unique for an MQTT broker. |
| **TOPIC** | Enter `devices/<device ID>/events`. |
| **AUTO REFRESH** | Indicates whether you want to automatically refresh the authentication token when it expires. If you switch the toggle off, you must manually |

| Field | Description |
|---|---|
| | provide the token as soon as it expires. Google Cloud accepts only those tokens that expire in 24 hours or less; therefore, we recommend that you switch the toggle on. |
| **USERNAME** | Enter any value. This value is not used, but only if you enter a value, you can proceed. |
| **PASSWORD** | If you have switched the **AUTO REFRESH** toggle on, leave this field blank. Historian generates a JSON Web Token (JWT) and uses it automatically. |
| **CA SERVER ROOT FILE** | Enter the path of the Google root CA certificate that you have downloaded. |
| **CLIENT CERTIFICATE** | Enter the path to the client certificate. |
| **PRIVATE KEY FILE** | Enter the complete path to the private key file. A value is required. |
| **PUBLIC KEY FILE** | Enter the path to the public key file. A value is required. |
| **CHOOSE CONFIGURATION** | The type of the configuration to specify the tags whose data you want to collect. Select one of the following options:<br>◦ **Historian Configuration**: Select this option if you want to add the tags manually *(on page 357)*. If you select this option, the **CONFIGURATION HISTORIAN SERVER** field appears.<br>◦ **Offline Configuration**: Select this option if you want to provide the tag names using the offline configuration *(on page 2039)* file instead of adding tags manually. By default, this file is located in the following location: `<installation folder of Historian>\GE Digital\<collector name>` |

| Field | Description |
|---|---|
| **CONFIGURATION HISTORIAN SERVER** | The host name of the machine from which you want to access Historian Administrator to add the tags manually for the collector. This field appears only if you have selected **Historian Configuration** in the **CHOOSE CONFIGURATION** field. |

15. Select **Next**.

    The **Collector Initiation** section appears.




16. Enter a collector name.

    The value that you enter:

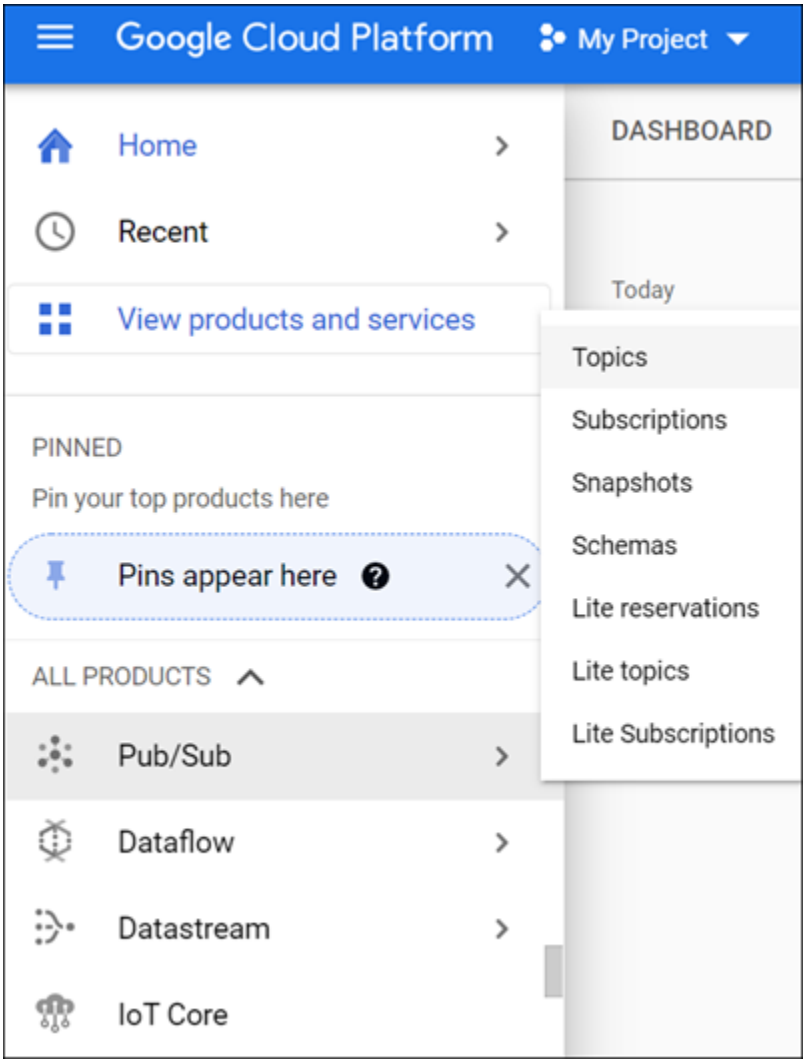
    - Must be unique.
    - Must not exceed 15 characters.
    - Must not contain a space.
    - Must not contain special characters except a hyphen, period, and an underscore.

17. In the **RUNNING MODE** field, select one of the following options.

- **Service - Local System Account**: Select this option if you want to run the collector as a Windows service using the credentials of the local user (that is, the currently logged-in user). If you select this option, the **USERNAME** and **PASSWORD** fields are disabled.
- **Service Under Specific User Account**: Select this option if you want to run the collector as a Windows service using a specific user account. If you select this option, you must enter values in the **USERNAME** and **PASSWORD** fields. If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:
  - iH Security Admins
  - iH Collector Admins
  - iH Tag Admins

If you choose the first option, you can also configure the collector to start automatically when you start the computer, or, in the case of iFIX collectors, whenever you start iFIX.

18. Select **Add**.
    The collector instance is created.
19. Access Google Cloud Platform, and select **Pub/Sub > Topics**.




20. Select **Messages > PULL**.
    Messages published to the topic that you have created appear. These messages contain the data sent by the collector instance. You can verify that the message content is correct by selecting **Message body**.

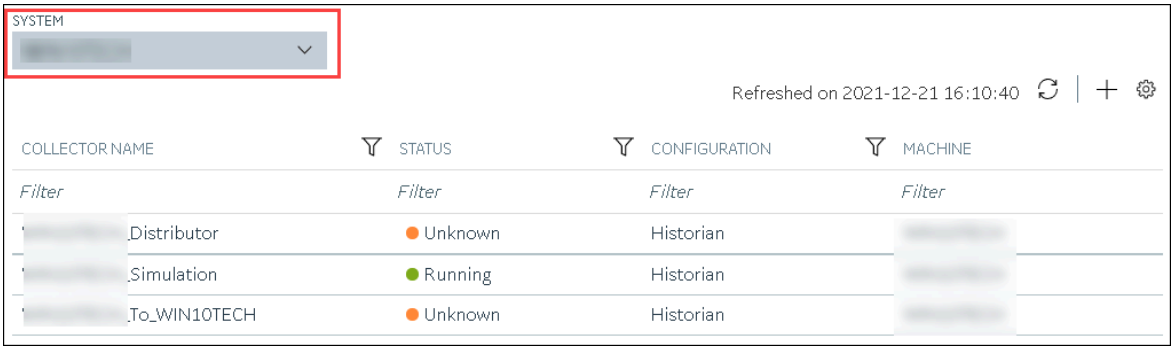
## Send Data to Predix Cloud

To send data to Predix Cloud, you can choose any of the following collectors:

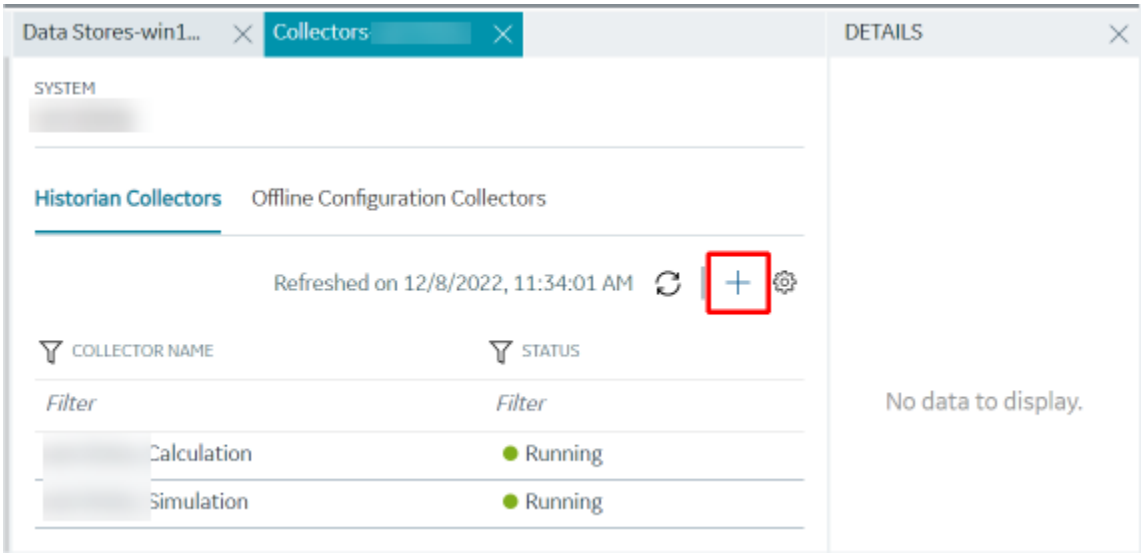
- The iFIX collector
- The MQTT collector
- The ODBC collector
- The OPC Classic DA collector
- The OPC Classic HDA collector

- The OPC UA DA collector
- The OSI PI collector
- The Python Collector
- The Server-to-Server collector
- The Simulation collector
- The Windows Performance collector
- The Wonderware collector

1. Register with the Timeseries service or any UAA service that you want to use. Note down the destination address, URI, client ID, client secret, and the zone ID that you have provided.
2. Access Configuration Hub *(on page 336)*.
3. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**. A list of collectors in the system appears.
4. If needed, select the system in which you want to add a collector instance.

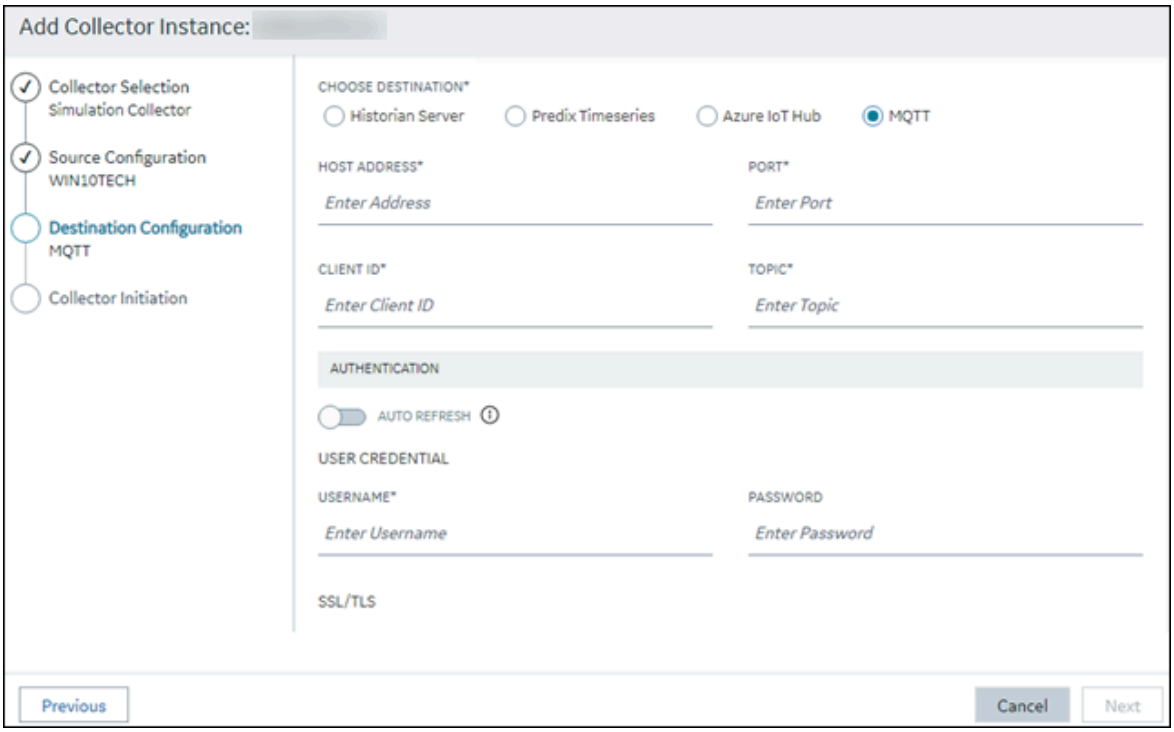



5. In the upper-right corner of the main section, select +.



The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.

6. In the **COLLECTOR TYPE** field, select a collector type (except the File collector and the Server-to-Server collector), and then select **Get Details**.
   The **INSTALLATION DRIVE** and **DATA DIRECTORY** fields are disabled and populated.
7. Select **Next**.
   The **Source Configuration** section appears, populating the hostname of the collector machine.
8. As needed, enter values in the available fields, and then select **Next**.
   The **Destination Configuration** section appears.




9. In the **CHOOSE DESTINATION** field, select **Predix Timeseries**, and then provide values as described in the following table.

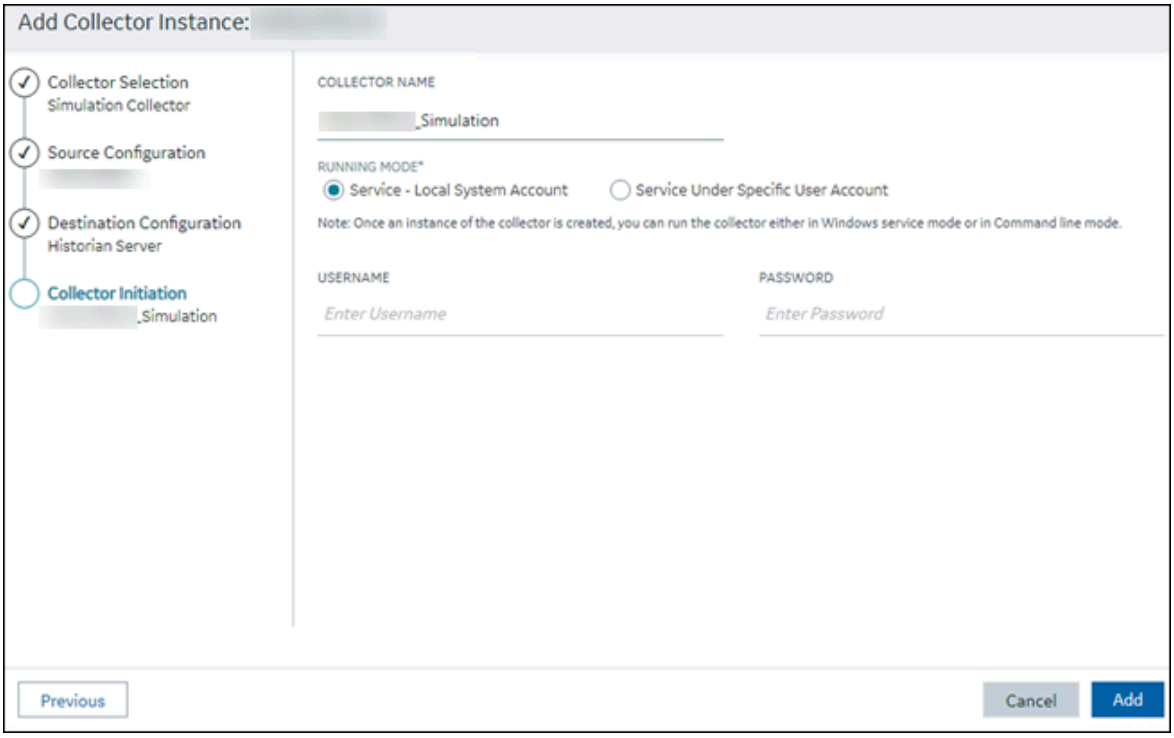
| Field | Description |
|---|---|
| **CLOUD DESTINATION ADDRESS** | The URL of a data streaming endpoint exposed by the Predix Time Series instance to which you want to send data. Typically, it starts with “wss://”. This value is used as part of the interface name and default tag prefix of the collector. Your Predix Time Series administrator can provide this URL. |

| Field | Description |
|---|---|
| **IDENTITY ISSUER** | The URL of an authentication endpoint for the collector to authenticate itself and acquire necessary credentials to stream to the Predix Time Series. In other words, this is the issuer ID of the Proficy Authentication instance that you want to use to connect to Predix Time Series. Typically, it starts with https:// and ends with "/oauth/token". |
| **CLIENT ID** | Identifies the collector when interacting with Predix Time Series. This is equivalent to the username in many authentication schemes. The client must exist in the Proficy Authentication instance identified by the identity issuer, and the system requires that the `timeseries.zones. {ZoneId}.ingest` and `timeseries.zones.{ZoneId}.query` authorities are granted access to the client for the Predix Zone ID specified. Your Predix Time Series administrator can provide this information. |
| **CLIENT SECRET** | The secret to authenticate the collector. This is equivalent to the password in many authentication schemes. |
| **ZONE ID** | Unique identifier of the instance to which the collector will send data. |
| **PROXY** | Identifies the URL of the proxy server to be used for both the authentication process and for sending data. If the collector is running on a network where proxy servers are used to access web resources outside of the network, then you must provide the proxy server settings. However, it does not affect the proxy server used by Windows when establishing secure connections. As a result, you must still configure the proxy settings for the Windows user account under which the collector service runs. |
| **PROXY USERNAME** | The username to connect to the proxy server. |
| **PROXY PASSWORD** | The password to connect to the proxy server. |
| **DATAPOINT ATTRIBUTES** | The attributes or parameters related to a datapoint that you want the collector to collect. Select **Add Attributes** to specify the attributes. You can add maximum five attributes for each collector instance. |

| Field | Description |
|---|---|
| **CHOOSE CONFIGURATION** | The type of the configuration to specify the tags whose data you want to collect. Select one of the following options:<br>◦ **Historian Configuration**: Select this option if you want to add the tags manually *(on page 357)*. If you select this option, the **CONFIGURATION HISTORIAN SERVER** field appears.<br>◦ **Offline Configuration**: Select this option if you want to provide the tag names using the offline configuration *(on page 2039)* file instead of adding tags manually. By default, this file is located in the following location: `<installation folder of Historian>\GE Digital\<collector name>` |
| **CONFIGURATION HISTORIAN SERVER** | The host name of the machine from which you want to access Historian Administrator to add the tags manually for the collector. This field appears only if you have selected **Historian Configuration** in the **CHOOSE CONFIGURATION** field. |

10. Select **Next**.

    The **Collector Initiation** section appears.



11. Enter a collector name.
    The value that you enter:
    - Must be unique.
    - Must not exceed 15 characters.
    - Must not contain a space.
    - Must not contain special characters except a hyphen, period, and an underscore.
12. In the **RUNNING MODE** field, select one of the following options.
    - **Service - Local System Account**: Select this option if you want to run the collector as a Windows service using the credentials of the local user (that is, the currently logged-in user). If you select this option, the **USERNAME** and **PASSWORD** fields are disabled.
    - **Service Under Specific User Account**: Select this option if you want to run the collector as a Windows service using a specific user account. If you select this option, you must enter values in the **USERNAME** and **PASSWORD** fields. If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:
      - iH Security Admins
      - iH Collector Admins
      - iH Tag Admins

    If you choose the first option, you can also configure the collector to start automatically when you start the computer, or, in the case of iFIX collectors, whenever you start iFIX.
13. Select **Add**.
    The collector instance is created.
14. Specify the tags for which you want to collect data.
    - If you have selected **Historian Configuration** in the **CHOOSE CONFIGURATION** field, specify the tags manually *(on page 357)*.
    - If you have selected **Offline Configuration** in the **CHOOSE CONFIGURATION** field, specify the tags using the offline configuration file *(on page 2039)*.

    The collector begins sending Historian data to Predix Timeseries.

## Protocols and Port Numbers

The following table provides a list of protocols that are available to send data to Azure IoT Hub, guidelines on which protocol to choose, and the port number that each protocol uses.

| Protocol | When to Use | Port Number |
|---|---|---|
| HTTP | Use this protocol if the data that you want to send is not large and/or the default ports for the other protocols are not available. | 80 |
| MQTT | MQTT is lightweight compared to AMQP, and is widely used. Use this protocol if you want to send data using low bandwidth and/or you do not want to connect to multiple devices using the same connection. | 8883 |
| AMQP | AMQP is more reliable compared to other protocols. It sends data in batches, and hence, the network traffic is less compared to that of MQTT. Use this protocol if you want to send a large amount of data from multiple collectors frequently. | 5671 |
| MQTT over web sockets | MQTT is lightweight compared to AMQP, and is widely used. In addition, communication using web sockets is more reliable and secure. Use this protocol if you want to send data using low bandwidth and securely. | 443 |
| AMQP over web sockets | AMQP is more reliable compared to other protocols. It sends data in batches, and hence, the network traffic is less compared to that of MQTT. In addition, communication using web sockets is more reliable and secure. Use this protocol if you want to send a large amount of data from multiple collectors frequently and securely. | 443 |

# Offline Collector Configuration

## Offline Configuration for Collectors

Offline Configuration helps you to define the configuration properties of a collector (Taglist, Tag properties, and collector interface properties) in XML format. This feature is particularly useful when collectors connect to a cloud destination.

When collectors connect to both Historian server and cloud, you can add tags, set tag configuration properties, and collector interface properties.

The path to the Offline Tag Configuration file is provided in the collector registry as: OfflineTagConfigurationFile (Data Type: String). This key contains the path to offline configuration file. For Server to Server Collector the default value is: `C:\Program Files (x86)\GE Digital\Historian Server to Server Collector\Config\S2S_Offline_Config.xml`.

The following tag properties are required for Cloud collector tags:

Mandatory properties for non-historical collector (PI to Cloud, OPCUA to Cloud etc)

**Tagname**, **Data Type** and **Source Address**

Mandatory properties for historical collector (Server to Cloud)

**Tagname**, **Data Type**, **Source Address** and **CalculationDependency**

All other properties are not mandatory.

## Creating Offline Configuration XML file

It is recommended that you add the Collector property section above the Tag property section in your offline configuration XML file.

1. Add the following collector interface properties to the top of your configuration XML file.
   The following is an example for the Server to Server Collector interface properties:

   ```
   <Import>
   <Collectors>
   <Collector Name="<Collector Name>">
   <InterfaceType>ServerToServer</InterfaceType>
   <InterfaceGeneral1>10</InterfaceGeneral1>
   ......
   </Collector>
   </Collectors>
   ```

2. Add your TagList and Tag properties to your XML file.

   ```
   <Collectors>
   ...
   </Collectors>
   ```

```
<TagList Version="1.0.71">


<Tag>
<Tagname>simCollector1</Tagname>
<SourceAddress>Result = CurrentValue("SJC1GEIP05.Simulation00002")</SourceAddress>
...
</Tag>


<Tag>
<Tagname>simCollector2</Tagname>
<SourceAddress>Result = CurrentValue("SJC1GEIP05.Simulation00002")</SourceAddress>
...
</Tag>
...
</TagList>
</Import>
```

3. Add the closing `</Import>` tag to the end of your XML file.

## Collector Interface Properties

The collector interface properties are written in the following formal in the XML file.

```
<Import>
<Collectors>
<Collector Name="<Collector Name>">
<InterfaceType>ServerToServer</InterfaceType>
<InterfaceGeneral1>10</InterfaceGeneral1>
......
</Collector>
</Collectors>
</Import>
```

where *<Collector Name>* is the collector name found in the ServerToServerCollector.shw file.

You can configure the following properties:

| Property Name | Possible Values | Example |
|---|---|---|
| InterfaceType | ServerToServer, PI, Custom | `<InterfaceType>ServerToServer</InterfaceType>` |
| DefaultTagPrefix | Any tag prefix name | <DefaultTagPrefix>OfflineCloud</DefaultTagPrefix> |
| CanBrowseSource | Yes, No | `<CanBrowseSource>Yes</CanBrowseSource>` |
| CanSourceTimestamp | Yes, No | `<CanSourceTimestamp>Yes</CanSourceTimestamp>` |
| MinimumDiskFreeBuffer-Size | Size in MB | `<MinimumDiskFreeBufferSize>150</MinimumDiskFree-BufferSize>` |
| MaximumMemory-BufferSize | Size in MB | `<MaximumMemoryBufferSize>200</MaximumMemoryBuffer-Size>` |
| ShouldAdjustTime | Yes, No | `<ShouldAdjustTime>Yes</ShouldAdjustTime>` |
| ShouldQueueWrites | Yes, No | `<ShouldQueueWrites>No</ShouldQueueWrites>` |
| SourceTimeInLocalTime | Yes, No | `<SourceTimeInLocalTime>No</SourceTimeInLocalTime>` |
| CollectionDelay | Time in seconds | `<CollectionDelay>2</CollectionDelay>` |
| DefaultCollectionInterval | Time in milliseconds | `<DefaultCollectionInterval>1000</DefaultCollection-Interval>` |
| DefaultCollectionType | Polled, Unsolicited | `<DefaultCollectionType>Unsolicited</DefaultCollec-tionType>` |
| DefaultTimeStampType | Source, Collector | `<DefaultTimeStampType>Source</DefaultTimeStampType>` |
| DefaultLoadBalancing | Yes, No | `<DefaultLoadBalancing>No</DefaultLoadBalancing>` |
| DefaultCollectorCom-pression | Yes, No | `<DefaultCollectorCompression>No</DefaultCollector-Compression>` |
| DefaultCollectorCom-pressionDeadband | Double type value | `<DefaultCollectorCompressionDeadband>0.00000</De-faultCollectorCompressionDeadband>` |
| DisableOnTheFlyTag-Change | Yes, No | `<DisableOnTheFlyTagChange>No</DisableOnTheFlyTag-Change>` |
| DefaultCollectorCom-pressionTimeout | Time in milliseconds | `<DefaultCollectorCompressionTimeout>0</DefaultCol-lectorCompressionTimeout>` |

| Property Name | Possible Values | Example |
|---|---|---|
| DefaultSpikeLogic | Yes, No | `<DefaultSpikeLogic>Yes</DefaultSpikeLogic>` |
| DefaultSpikeMultiplier | Any numeric value | `<DefaultSpikeMultiplier>4</DefaultSpikeMultiplier>` |
| DefaultSpikeInterval | Any numeric value | `<DefaultSpikeInterval>5</DefaultSpikeInterval>` |
| DataRecoveryQueueEnabled | Yes, No | `<DataRecoveryQueueEnabled>No</DataRecoveryQueueEnabled>` |
| DefaultAbsoluteDeadbanding | Yes, No | `<DefaultAbsoluteDeadbanding></DefaultAbsoluteDeadbanding>` |
| DefaultAbsoluteDeadband | Double type value | `<DefaultAbsoluteDeadband>0.00000</DefaultAbsoluteDeadband>` |
| RedundancyEnabled | Yes, No | `<RedundancyEnabled>No</RedundancyEnabled>` |
| RedundancyPrincipalCollector | | `<RedundancyPrincipalCollector></RedundancyPrincipalCollector>` |
| RedundancyIsActiveCollector | Yes, No | `<RedundancyIsActiveCollector>No</RedundancyIsActiveCollector>` |
| InterfaceGeneral1 | Customized for each collector | `<InterfaceGeneral1>10</InterfaceGeneral1>` |
| InterfaceGeneral2 | Customized for each collector | `<InterfaceGeneral2>4</InterfaceGeneral2>` |
| InterfaceGeneral3 | Customized for each collector | `<InterfaceGeneral3>3.188.87.41</InterfaceGeneral3>` |
| InterfaceGeneral4 | Customized for each collector | `<InterfaceGeneral4></InterfaceGeneral4>` |
| InterfaceGeneral5 | Customized for each collector | `<InterfaceGeneral5></InterfaceGeneral5>` |

## Tag List and Tag Properties

The format of Source Address for the Server to Cloud Collector is typically of format: Result = CurrentValue("SJC1GEIP05.Simulation00002"). For other collectors the format of the source address should be the value expected by the source server, for example, OPC and PI Collector tags typically use the Tag source item id for the SourceAddress.

Server to Cloud Tag Example:

```
<Tag>
<Tagname>simCollector2</Tagname>
<SourceAddress>Result = CurrentValue("SJC1GEIP05.Simulation00002")</SourceAddress>
...
</Tag>
```

OPC Collector Tag Example:

```
<Tag>
        <Tagname>simCollector2</Tagname>
        <SourceAddress>Channel1.Device1.Tag1</SourceAddress>
        ...
    </Tag>
```

You can configure the following properties:

| Property Name | Possible Values | Example |
|---|---|---|
| Tagname | Any name | `<Tagname>sim-Tag1</Tagname>` |
| Description | Description of tag | `<Description>sim-Tag1</Description>` |
| EngineeringUnits | Unit of value | `<EngineeringUnit-s>Centigrade</Engi-neeringUnits>` |
| Comment | Comment of tag | `<Comment>sim-Tag1</Comment>` |
| DataType | SingleFloat, SingleInteger, DoubleFloat, FixedString, Vari-ableString, Scaled, Byte, Boolean, DoubleInteger, Unsigned-SingleInteger, UnsignedDoubleInteger, QuadInteger, Un-signedQuadInteger, Blob, Time, Array, MultiField | `<DataType>Single-Float</DataType>` |
| FixedStringLength | | `<FixedStringLength></FixedStringLength>` |
| InterfaceName | | `<InterfaceName></In-terfaceName>` |

| Property Name | Possible Values | Example |
|---|---|---|
| SourceAddress | Tag source address | `<SourceAddress>Result = CurrentValue("SJC1GEIP05.Simulation00002")</SourceAddress>` |
| CollectionType | Polled, Unsolicited | `<CollectionType>Unsolicited</CollectionType>` |
| CollectionInterval | Interval of collection. Unit depends on TimeResolution. | `<CollectionInterval>2</CollectionInterval>` |
| CollectionOffset | Time in seconds | `<CollectionOffset>0</CollectionOffset>` |
| LoadBalancing | Yes, No | `<LoadBalancing>No</LoadBalancing>` |
| TimeStampType | Source, Collector | `<TimeStampType>Source</TimeStampType>` |
| HiEngineeringUnits | Any numeric value | `<HiEngineeringUnits>200000.00</HiEngineeringUnits>` |
| LoEngineeringUnits | Any numeric value | `<LoEngineeringUnits>0</LoEngineeringUnits` |
| InputScaling | Yes, No | `<InputScaling>Yes</InputScaling>` |

| Property Name | Possible Values | Example |
|---|---|---|
| HiScale | Any numeric value | `<HiScale>32767.00</HiScale>` |
| LoScale | Any numeric value | `<LoScale>0</LoScale>` |
| SpikeLogic | Yes, No | `<SpikeLogic>Yes</SpikeLogic>` |
| SpikeLogicOverride | Yes, No | `<SpikeLogicOverride>Yes</SpikeLogicOverride>` |
| InterfaceCompression | Yes, No | `<InterfaceCompression>Yes</InterfaceCompression>` |
| InterfaceDeadbandPercentRange | Any double type value | `<InterfaceDeadbandPercentRange>0</InterfaceDeadbandPercentRange>` |
| InterfaceCompressionTimeout | Time in milliseconds | `<InterfaceCompressionTimeout>0</InterfaceCompressionTimeout>` |
| InterfaceAbsoluteDeadband | Any double type value | `<InterfaceAbsoluteDeadband>0</InterfaceAbsoluteDeadband>` |
| InterfaceAbsoluteDeadbanding | Yes, No | `<InterfaceAbsoluteDeadbanding>No</InterfaceAbsoluteDeadbanding>` |
| ConditionCollectionEnabled | Yes, No | `<ConditionCollectionEnabled>No</ConditionCollectionEnabled>` |

| Property Name | Possible Values | Example |
|---|---|---|
| ConditionCollectionTriggerTag | Name of tag | `<ConditionCollectionTriggerTag>simTag1</ConditionCollectionTriggerTag>` |
| ConditionCollectionComparison | = , EQ , < , LT , <= , LE , > , GT , >= , GE , != , NE | `<ConditionCollectionComparison>EQ</ConditionCollectionComparison>` |
| ConditionCollectionCompareValue | Any numeric value | `<ConditionCollectionCompareValue>0</ConditionCollectionCompareValue>` |
| ConditionCollectionMarkers | Yes, No | `<ConditionCollectionMarkers>No</ConditionCollectionMarkers>` |
| NumberOfElements | | `<NumberOfElements></NumberOfElements>` |
| UserDefinedTypeName | | `<UserDefinedTypeName></UserDefinedTypeName>` |
| CalcType | Raw, Analytic, PythonExpr | <CalcType>Raw</CalcType> |
| TimeResolution | Seconds, Milliseconds, Microseconds | `<TimeResolution>Seconds</TimeResolution>` |
| InterfaceGeneral1 | Customized for each collector | `<InterfaceGeneral1>10</InterfaceGeneral1>` |

| Property Name | Possible Values | Example |
|---|---|---|
| InterfaceGeneral2 | Customized for each collector | `<InterfaceGeneral2>4</InterfaceGeneral2>` |
| InterfaceGeneral3 | Customized for each collector | `<InterfaceGeneral3>3.188.87.41</InterfaceGeneral3>` |
| InterfaceGeneral4 | Customized for each collector | <InterfaceGeneral4></InterfaceGeneral4> |
| InterfaceGeneral5 | Customized for each collector | `<InterfaceGeneral5></InterfaceGeneral5>` |

## About Updating Tag Properties Dynamically

When you add or delete tags for a collector, or when you modify the properties of the tags or the collector using the offline configuration file, the changes are reflected without the need to restart the collector.

The following conditions apply for the changes to reflect automatically:

- This is applicable only to bi-modal collectors.
- The changes are reflected in 90 seconds.

**Note:**
If you do not want tag properties to be updated dynamically for a collector, perform the following steps:

1. Access the offline configuration file for the collector. By default, this file is available in the following location: `<installation folder of Historian>\GE Digital \<collector name>`. For example, for the simulation collector, the path to the file is `C:\Program Files (x86)\GE Digital\Historian Simulation Collector \Config\Sim_Offline_Config.xml`.

2. Inside the `<collector>` element, for the `<DisableOnTheFlyTagChanges>` parameter, enter Yes, and save the file. By default, the value is No.

   **Tip:**
   To verify that the changes are saved, verify the .shw file for the collector.

### Troubleshooting

By default, the offline configuration file is continuously monitored for any changes in the tag names and/or properties, and the changes are reflected dynamically. If, however, the changes are not reflected, you can create the following registry key to fix the issue: `OfflineTagForceCheckDuration`

## Cloud Collector Specific Registry Configuration

Various registry keys are available for modifying the default behavior of cloud collector. These keys can be added to the specific cloud collectors for altering the default behavior.

For a Server-to-Server collector, the keys will be added to: HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\GE Digital\iHistorian\Services\ServerToServerCollector

**ZIP compression**: Zip compression is available for Cloud collectors so that the JSON Payload can be compressed with a reduced network bandwidth.

Following registry key is available for compression:

**`JsonPayloadGzipCompression:`** Use this registry key to reduce the network usage while the data is transferred from collector to cloud.

Default value: 0, means no compression (if registry key does not exist)

Valid values: 1 to 9. 1 is minimal compression and 9 is maximum compression.

Data Type: DWORD

**Registry key for Controlling Max send queue size**

`CloudMaxOutstandingMsgs`: Use this registry key to configure the maximum send queue size.

Default value: 512

Valid values: 24 to 512

Data Type: DWORD

Registry key to save messages in failed queues

`FailedMsgQueueEnabled`: Use this registry key to save messages in failed queues as backup. This is done when any message fails for any unexpected reason.

Default: 0 (Disabled). (if no registry key)

Valid Values: 0 (Disabled) or 1(Enabled)

Data Type: Binary

# Working with Tags

## Understanding Tag names

Historian tag names vary according to the type of collector. By default, the tag name is the source address prepended with a string.

It is recommended that tag names use only characters available for folders and file names to avoid the problems (limitations) with some clients and filtering. You can use tag names that contain the characters & and + in the Non-Web Historian Administrator.

### iFIXCollector Tagnames

The format of Historian tag names for an iFIX collector generally is

```
Node.Tag.Field
```

where

- `Node`, by default, is the name of the SCADA node, the data source. This field is configurable, however.
- `Tag` is the database tag.
- `Field` is the database field.

Examples of typical tag names:

```
NODE8.WATER-_SWITCH.F_CV

NODE2.MASH_LEVEL.B_CUALM

USGBS1.FIC101.F_CV

USGBS1.FT102.A_LAALM
```

where

- `NODE8`, `NODE2`, and `USGBS1` are the names of the iFIX SCADA nodes.
- `WATER_SWITCH`, `MASH_LEVEL`, `FIC101`, and FT102 are the names of the database tags.
- `4F_CV` means single floating point, current value.
- `B_CUALM` means current alarm status.
- `A_LAALM` means analog input, latched alarm.

### OPC or Simulation Collector Tagnames

The format of Historian tags for an OPC or Simulation collector is:

```
ComputerName.ItemID
```

where

- `ComputerName`, by default, is the name of the machine on which the collector is installed. This field is configurable, however.
- `ItemID` is the data point being polled.

### Calculation collector Tagnames

There is no specified format for Historian tags for a Calculation collector. We recommend that you select a consistent naming convention so the tags are easily and clearly identifiable. You should avoid using spaces and other special characters or reserved words used in SQL or VBScript. This applies to any tag being used as a source tag in the formula.

### Server-to-Server Collector Tagnames

There is no specified format for Historian tags for a Server-to-Server Collector. We recommend that you select a consistent naming convention so the tags are easily and clearly identifiable. Server-to-Server Collectors allow you to specify a prefix to add to the Server-to-Server Collector tags.

**Note:**
In cases where you want the destination tag's data to be a duplicate of the source tag, then the tag name would be identical. This is especially important so that all messages and alerts are included.

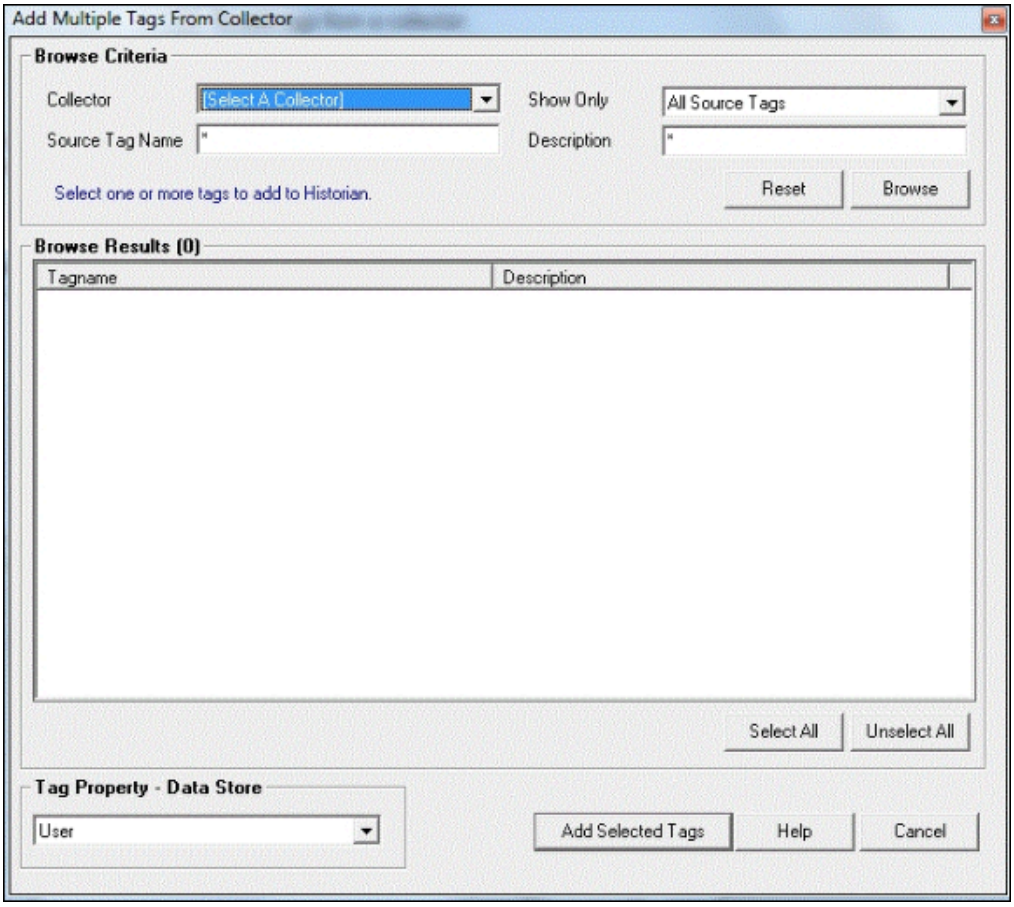
# Adding Tags from a Collector

## Add Multiple Tags from a Collector Using Historian Administrator

This topic describes how to add multiple tags from a collector using Historian Administrator. You can also add tags using the Web Admin console *(on page 2055)*.

**Note:**
When you add a tag, do not enter a leading space or a trailing space in the tag name.

1. Select the **Add Tags From Collector** link in the **Tag Maintenance** page.
   The **Add Multiple Tags from Collector** window appears.



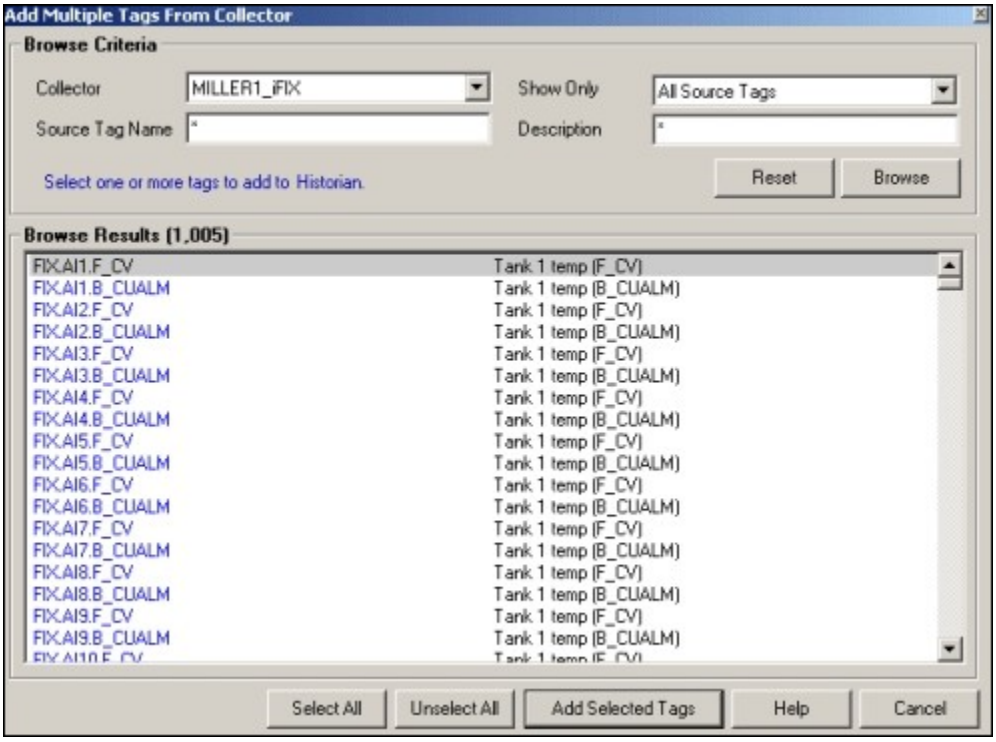

2. Select a collector from the **Collector** drop-down list.

> **Note:**
> If you add tags from a File collector, the file you import specifies the collector to which the tags are assigned. Those tags are then returned by a browse of the specified collector. It is also possible to leave the assignment blank. If the file does not specify a Collector Name for a tag, the tag is added with no collector name.
>
> You cannot browse a Calculation collector through the **Add Tags From Collector** window in Historian Administrator.

3. In the **Show Only** field, select either **All Source Tags** or **Source Tags Not Collected**.
   If you select the second option, the browse returns only the tags that are not currently included for collection.

   If a Historian tag name is different from its source address tag name, the source tag is displayed in the returned list even if you browse the collector using the **Show Source Tags Not Collected** criterion. Collection on the same source address using a unique tag name is allowed.
4. In the **Source Tag name** and **Description** fields, you can optionally enter masks for the browse, using standard Windows wildcard characters.
5. Select **Browse** to initiate the search or **Reset** to start over.

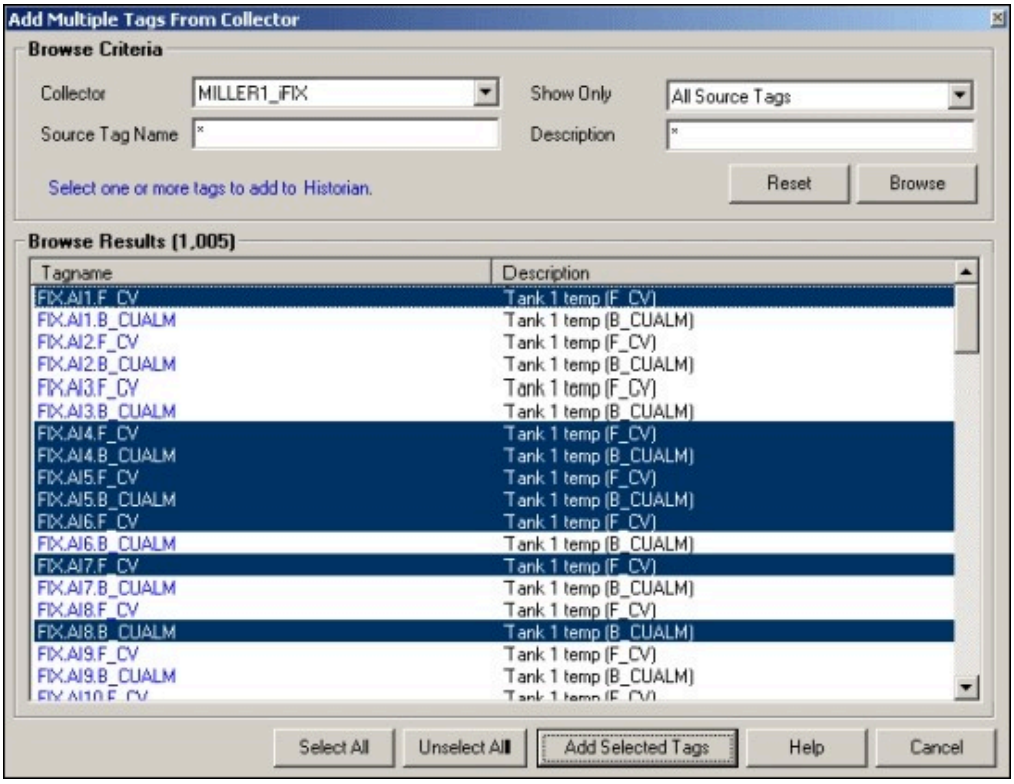
The browse returns a list of tags, as shown in the following figure. In the Historian Non-Web Administrator, a tag that is currently collected appears in black type. A tag that is not currently collected appears in blue type.




## Adding Uncollected Tags from a Collector

1. Select the required tags. Selecting a tag selects its tag name and description.
   you can select:
   - a single tag by selecting on the name of the tag.
   - multiple tags by pressing the **Ctrl** key and selecting the tags.
   - a contiguous group by selecting the first tag, pressing the **Shift** key, and selecting the last tag of the group.
   - all tags by selecting **Select All** at the bottom of the page.

   To clear all tags, select **Unselect All**.




2. Select **Add Selected Tags**. The selected tags are added to the Historian Tag Database.
   The **Tag Maintenance** page appears. The lower left portion of the page displays a list of all tag names added to the Historian Tag Database.

## Add Tags from a Collector Using the Web Admin Console

This topic describes how to add tags from a collector using the Web Admin console. You can also add tags using Historian Administrator *(on page 2052)*.

> **Note:**
> When you add a tag, do not enter a leading space or a trailing space in the tag name.

1. Select the plus sign in the **Tag** page.
2. Select **Add Tags from Collector**.
   The **Add Tags from Collector** window appears.
3. Select the collector from the **Collector name** list.
   For collectors with hierarchical browsing, expand the folder to select the desired tags. The > symbol indicates that you need to navigate further within the folder.

4. Enter the **Source Tag Name** or select **Browse**.
   The list of available tags is displayed.
5. Select tags.
   You can select a single tag and or select multiple tags. To select a series of tags, press and hold the **Shift** key and select the series.
6. Select **Add** to add the tags. To add all the tags, select **Add All**.
7. Select **Preview** to preview the selected tag details.
8. Select **Add Selected Tags** to add the selected tags from the collectors.

## Adding Tags for Collectors with Hierarchical Browsing

1. Select the plus sign in the **Tag** page.
2. Select **Add Tags from Collector**.
   The **Add Tags from Collector** window appears.
3. Select the collector from the **Collector name** list.
4. Enter the **Source Tag Name** or select **Browse**.
   The list of available tags is displayed.
5. Select tags.
   You can select a single tag and or select multiple tags. To select a series of tags, press and hold the **Shift** key and select the series.
6. Select **Add** to add the tags. To add all the tags, select **Add All**.
7. Select **Preview** to preview the selected tag details.
8. Select **Add Selected Tags** to add the selected tags from the collectors.

## Manually Adding Tags

Typically, you add new tags to the Historian by browsing the data source or by bulk importing a group of tags with the Excel Add-In tool. Occasionally, you may need to add a single tag manually. You can add tags manually:

- for creating a calculation tag.
- for holding values that are added using the Excel Add-In or custom SDK application.
- for testing purposes.

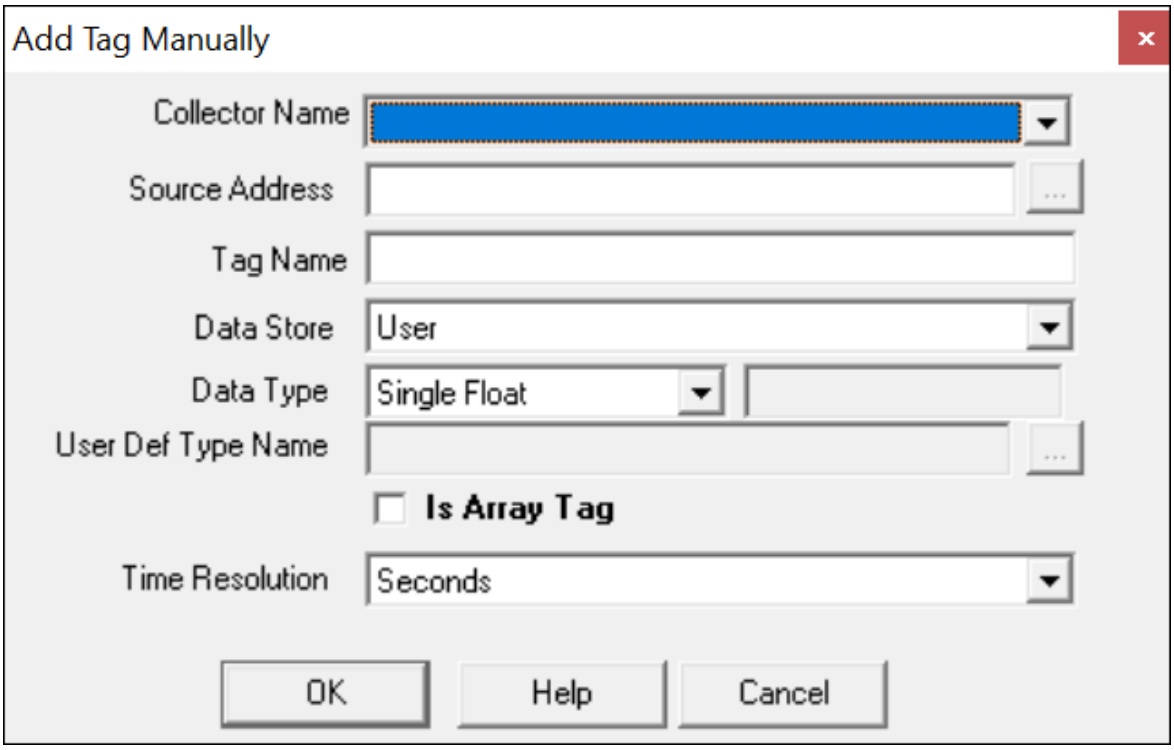
For example, if you are currently not connected to a collector, the browse is slow, or is not supported, and if you want to configure tags associated with that collector, you can add them manually.

When adding a tag manually, you must manually configure the fields for the tag that the **Add Tag Manually** window does not include. For instance, when you add a tag manually, the **Data Type** field does not automatically populate after you select a **Source Address**. You must manually set the data type from the **Collection** section after you add the tag. Use caution when selecting the data type. If you select the wrong data type, you most likely will get incorrect data or you could even lose data. It does not use the collector default settings, such as those for archive and collector compression, as it would with the browse and pick.

## Add a Tag Using Historian Administrator

1. Add an instance of the collector *(on page 357)* that you want to use to collect the tag data.
2. Create a data store *(on page 852)* in which you want to store the tag data.

1. Access Historian Administrator.
2. Select **Tags > Add Tag Manually**.
   The **Add Tag Manually** window appears.




3. Enter values in the available fields as described in the following table.

| Field | Description |
|---|---|
| **Collector Name** | Select the collector that you want to use to collect data of the tag. |
| **Source Address** | Enter the |
| **Tag Name** | Enter a name for the tag. |
| **Data Store** | Select the type of the data store in which you want to store the tag data. |
| **Data Type** | Select the data type of the tag. If you select **MultiField**, the **User Def Type Name** field is enabled. |
| **User Def Type Name** | For a tag of the multi-field data type, select the data type of the tag that you have defined. |
| **Is Array Tag** | Select this check box if the tag stores an array of data. |
| **Time Resolution** | |
| **Time Adjustment** | **Important:** If you manually add a Server-to-Server tag, ensure that you set the **Time Adjustment** field for the tag to the **Adjust for Source Time Difference** option, after you add the tag. The Time Adjustment field is located in the **Advanced** section in the **Tag Maintenance** page. This field only applies to Server-to-Server tags that use a polled collection type. |

4. Select **OK**.
   The tag is added.

## Add a Tag Using the Web Admin Console

The dynamic collector update feature ensures that any modifications done to the tag configuration do not affect all the tags in a collector. Only the tags that stop data collection will record zero data and bad quality without restarting the collector. In other words, the tags that do not stop data collection do not record bad data samples to the collection.

Whenever you add tags, delete tags, or modify certain tag properties, the following collectors reload only the modified tag(s) without restarting the collectors:

- OPC Collector
- iFIX collector
- Calculation collector
- Simulation Collector
- Server to Server Collector
- PI Collector
- PI Distributor

For a tag to stop and restart the collection without restarting the collector, you must enable the **On-line Tag Configuration Changes** option in the **Advanced** section of the **Collector Maintenance** page. By default, the **On-line Tag Configuration Changes** option is enabled.

If you disable the **On-line Tag Configuration Changes** option, any changes you make to the tags do not affect collection until after you restart the collector. To restart the collector, you must stop and start the collector service or executable. Restarting the collector stops and restarts the tag(s) collection and records bad data samples to the collection.

All the collector configuration changes done within a 30 second time frame are batched up to let you update/modify a small set of tags at a time to collect the modified data faster.

**Note:**
It is recommended that you disable the **On-line Tag Configuration Changes** option while updating large sets of tags at the same time, and restart the collector after modification.

Follow these steps to add a new tag manually from the Web Administrator:

1. Select the link in the Tag Details page and select **Add Tags Manually**.
   The **Add Tag** window appears.
2. Select a collector from the drop-down list in the **Collector Name** field.

This associates the new tag with a specific collector.

3. Enter the **Source Address** and **Tag Name** in the appropriate fields.
4. Select the data store in the **Data Store** field.
5. Select a **Data Type** from the drop-down list.
6. For fixed string data types only, enter a value in the field adjacent to the **Data Type** field.
7. Select Seconds, Milliseconds, or Microseconds in the **Time Resolution** field.
8. Select the **Is Array Tag** option, if the tag is an Array Tag.
9. Select **Add** to add the tag.

> **Important:**
> If you manually add a Server-to-Server tag, ensure that on the **Tag Maintenance** page **Advanced** tab, you set the **Time Adjustment** field to **Adjust for Source Time Difference** , after you add the tag. Note that, this field only applies to Server-to-Server tags that use a polled collection type.

## Add a Source Address Using the Web Admin Console

1. Select a collector from the drop-down list in the **Collector Name** field.
   This associates the new tag with a specific collector.
2. Enter the **Source Address** or select the **Browse** button.
   The **Add Tags from Collectors** window appears.
3. Select the tag you want to associate to source address. You can select only one tag.
4. Select **OK**.
   The source address of the tag will be added.

## Copy a Tag

If you want to create a copy of an existing tag with all the same properties, use the **Copy Tag** function. **Copy Tag** works only for individual tags. You cannot copy multiple tags at once.

1. To copy a tag using Historian Administrator:
   a. Select a tag to copy from the tag list.
   b. Select the **Copy Tag** link in the **Tag Maintenance** page.
      The **Copy Tag** window appears.
   c. Type a new tag name.
   d. Select **OK** to copy the tag and all the associated tag properties.

2. To copy a tag using the Web Admin console:

    a. Select the **Copy Tag** icon in the **Tag** page.
    The **Copy Tag** window appears.

    b. Type a new tag name.

    c. Select **OK** to copy the tag and all the associated tag properties.

## Search for Tags

1. To search for tags using Historian Administrator:

    a. In the **Tag Maintenance** page, select the **Search Historian Tag Database** link.
    The Search **Historian Tag Database** window appears.

    b. Enter a tag search mask in the **Tag Mask** field, using * and ? wildcard characters.
    You can search for a description instead of a tag by entering a mask in the **Description** field.

    If you leave both fields blank, the search returns all tags.

    c. Select the name of the collector in the **Collector** field.

    d. Enter the maximum number of tags the search should return.
    If you leave this field blank, your search will return all the tags available in the Historian Tag Database.

    e. Select **OK**.
    The **Tag Maintenance** page appears.

> **Note:**
> Prior to performing any maintenance, it is recommended to adhere to the accepted practice of performing a backup of your Historian archive. It is also recommended that you use the Excel Add-In to export your tag configuration for all tags. It is recommended that you export tags associated with each collector on a separate worksheet.

f. If you want to change the default tag properties, select the name of the tag.

g. Enter the properties of the tag in the appropriate fields.

h. Select **Update** to save your entries.

> **CAUTION:**
> Multi-select and property change will affect all selected tags. Changing the collector type for the Calculation and Server-to-Server tags will result in a loss of the calculation formula.

2. To search for tags using the Web Admin console:

   a. Select **Advanced Search** in the **Tags** page.
   The **Advanced Search** window appears.

   b. In the **Step 1** section, select the tag criteria from the list.

   c. Enter or select the **Tag Criteria Value**.
   If you leave the fields blank, the search returns all the available tags.

   d. Select **Add Criteria** and choose **Collector Name** as criteria.

   e. Select the collector from the list.

   f. Select **Find Tags**.
   All the tags that satisfy the query criteria are displayed in the **Step 2** section.

   g. Select the tags and select **Apply** to return the list of tags to the parent **Tags** page.

   h. If you want to change the default tag properties, select the name of the tag from the list and edit the properties in the **Tag Editor** section.

i. Select **Update** to save your entries.

> **CAUTION:**
> Multi-select and property change will affect all selected tags. Changing the collector type for the Calculation and Server-to-Server tags will result in a loss of the calculation formula.

## Remove Tags

Deleting a tag only removes it from the browse list; the data remains intact in the Archive and can be queried by tag name. It is recommended that you export your tag configuration before and after tag modifications.

1. To remove a tag using Historian Administrator:

    a. On Historian Administrator **Main** page, select the **Tags** link on the toolbar.
    The **Tag Maintenance** page appears.

    b. Select the name of the tag you want to remove.
    To remove multiple tags:
    - Select a tag to highlight it.
    - Select multiple tags by pressing the **Control** key and selecting the individual tags.
    - Select contiguous tags by pressing the **Shift** key and selecting the first and last tags of the sequence.

    c. Select the **Delete** button.
    The **Delete Tag** window appears.

    d. Select either **Remove Tag from System** or **Stop Data Collection** and select **OK**.
    If you want to stop collection temporarily and resume collection later for a specified time, you can disable collection for that tag instead. To do this, select the tag on the **Tag Maintenance** page, select **Collection**, and then select the **Disable** option for the **Collection** field.

2. To remove a tag using the Web Admin console:

    a. On Historian Administrator **Main** page, select the **Tags** link on the toolbar.
    The **Tag Maintenance** page appears.

    b. Select the name of the tag you want to remove.

To remove multiple tags:

- Select a tag to highlight it.
- Select multiple tags by pressing the **Control** key and selecting the individual tags.
- Select contiguous tags by pressing the **Shift** key and selecting the first and last tags of the sequence.

c. Select the **Delete** button.
The **Delete Tag** window appears.

d. Select either **Remove Tag from System** or **Stop Data Collection** and select **OK**.

## Browse a Data Source for New Tags

To browse for new tags, your collector must be running. If it is not running, start the collector.

The most common way to add tags to an Historian Database is to browse the data source for new tags.

**Note:**
Performing large tag browses in Historian Administrator may cause your session to time out. Use the browse filter criteria to return a smaller list. In the Non-Web Administrator, if your OPC server supports hierarchical organization of your tags, see Adding Tags for Collectors with Hierarchical Browsing *(on page 2056)* to speed browse sessions.

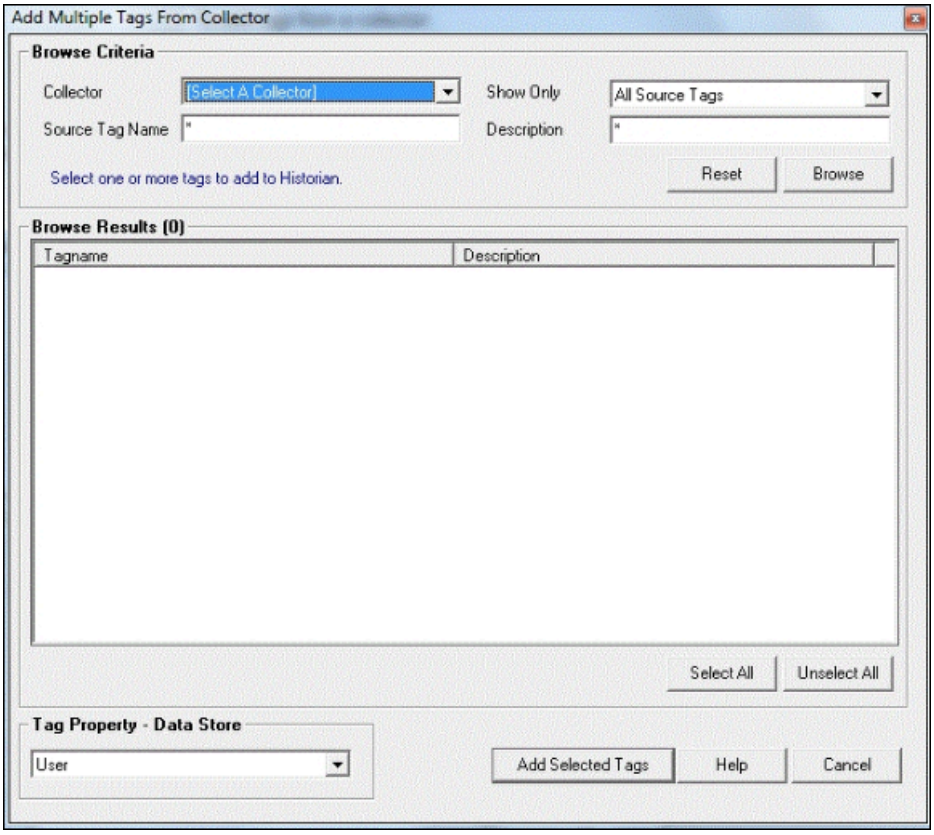
1. To browse for new tags using Historian Administrator:

   a. Open the Historian Non-Web Administrator.

   b. Select the **Collectors** link from the toolbar.
   The **Collector Maintenance** page appears.

   c. If you have multiple collectors listed, select a collector from the **Collectors** list.

   d. To browse for new tags from your data source, select **Add Tags** at the bottom of the page.
   The **Add Multiple Tags From Collector** window appears.



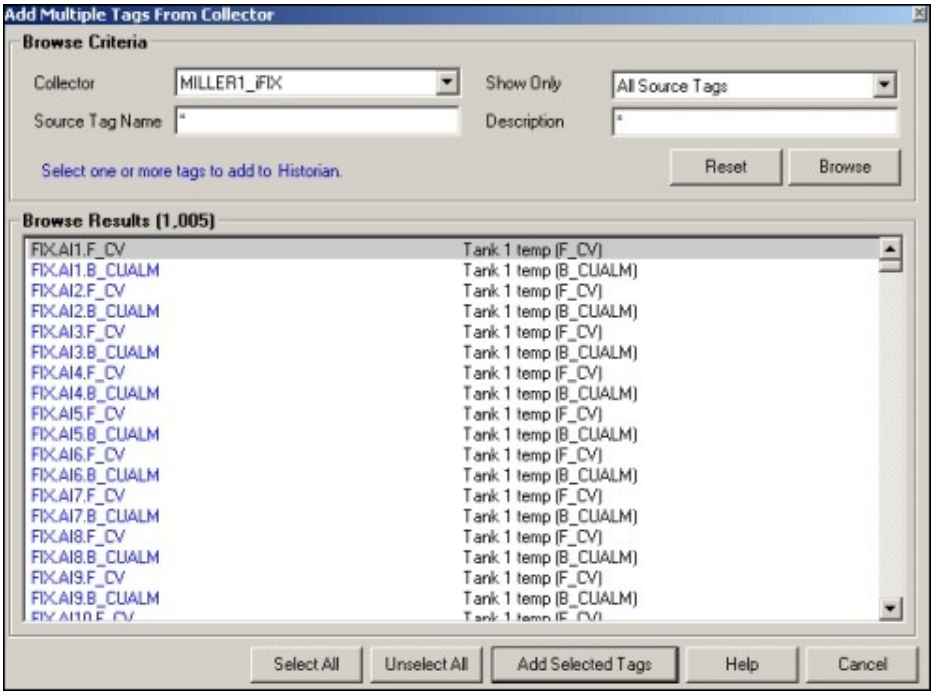

e. In the **Show Only** field, select either **All Source Tags** or **Source Tags Not Collected**.
If you select the second option, the browse returns only the tags that are not currently included for collection.

If a Historian tagname is different from its source address tagname, the source tag is displayed in the returned list even if you browse the collector using the **Show Source Tags Not Collected** criterion. Collection on the same source address using a unique tagname is allowed.

A check mark beside a tag indicates that the tag is currently being collected. Absence of a mark indicates that the tag is not currently being collected.

f. In the **Source Tagname** and **Description** fields, you can optionally enter masks for the browse, using standard Windows wildcard characters.

g. Select **Browse** to initiate the search or **Reset** to start over.
The browse returns a list of tags, as shown in the following figure. In the Historian Non-Web Administrator, a tag that is currently collected appears in black type. A tag that is not currently collected appears in blue type.




See also Add Multiple Tags from a Collector Using Historian Administrator *(on page 2052)*.

2. To browse for new tags using the Web Admin console:

   a. Open the Historian Web Administrator.

   b. Select the **Collectors** link from the toolbar.
   The **Collector** page appears.

   c. If you have multiple collectors listed, select a collector from the **Collectors** list.

   d. To browse for new tags from your data source, go to the **Tags** page from the **Configuration** page.

   e. Select **Add Tags** at the bottom of the page and select **Add Tag from Collector**.
   The **Add Tags from Collector** window appears.

   f. Select single or multiple tags, select on **Add Selected Tags** button.
   See also Add Tags from a Collector Using the Web Admin Console *(on page 2055)*.

## About Configuring Collector Options

If you are using Historian Administrator, configure collector options as follows:

- Start at the **Collector Maintenance** page in Historian Administrator. You can access the Collector Maintenance page in several ways:
    - Select the Collectors link in any of the major pages in Historian Administrator.
    - Select the collector name in the Collectors section of Historian Administrator Main page. This displays the page with the specific collector already selected.

The Collector Maintenance page lists all connected collectors at the left of the page. The right-side displays parameter values, in several tabs, for the collector you select by selecting on a name in the list.

- To make a change in a configurable parameter, enter the value in the appropriate field and select **Update**. For more information, refer to *Historian Administrator*.

If you are using the Web Admin console,

- Go to the **Collectors** page. This displays the list of available collectors and their status. You can edit the collector configurable parameters.
- From the **Collector Maintenance page** or the **Collector page**, select the various tabs to display parameters of various types for the specific collector you have selected.

  For more information, refer to *Historian Web Administrator Help*.

  .

## About Collector Redundancy

Historian includes support for collector redundancy, which decreases the likelihood of lost data due to software or hardware failures. Implementing collector redundancy ensures that collection of your data remains uninterrupted. Collector redundancy makes use of two or more collectors, gathering data from a single source. For more information, refer to the About Collector Redundancy *(on page 975)*.

**Note:**
Use Polled tags only as watchdog tags.

## Collect Vendor Attributes

Data sources are often customized to include information specific to a site's installation in the form of vendor attributes. This customization adds data not covered by the OPC or OPCAE specifications and may or may not require storage in the Historian Archive. As a result, Historian Administrator provides a means to specify which vendor attributes will be collected from any given data source. A maximum of 10 vendor attributes can be collected.

1. To collect vendor attributes using Historian Administrator:

    a. In the Historian Administrator **Main** page, select the **Collectors** link in the toolbar.
    The **Collector Maintenance** page appears.

    b. Select **Configuration**.

    c. To collect all vendor attributes from the data source, select **All**.

    d. To collect up to 10 selected vendor attributes from the data source:
        i. Select the **Selected** option.
        ii. Select **Add** to add a vendor attribute to collect from the data source.

        The **Vendor Attributes** window appears.

    e. In the **Vendor Attribute** field, enter the vendor attribute you wish to collect from the data source and select **OK**.
    The vendor attribute appears in the list box on the **Collector Configuration** page.

    f. To remove a vendor attribute, select it in the list box and select **Remove**.

    g. Select **Update** to apply your changes.

2. To collect vendor attributes using the Web Admin console:

    a. In the Historian Dashboard, select the **Details** button in the Collectors section.
    The **Collect Statistics** window appears.

    b. Select the **Configure** button.
    The **Collector Configuration** page appears.

    c. To collect all vendor attributes from the data source, select **All**.

    d. To collect up to 10 selected vendor attributes from the data source:
        i. Select the **Selected** option.
        ii. Select **Add** to add a vendor attribute to collect from the data source.

        The **Vendor Attributes** window appears.

    e. In the **Vendor Attribute** field, enter the vendor attribute you wish to collect from the data source and select **OK**.
    The vendor attribute appears in the list box on the **Collector Configuration** page.

f. To remove a vendor attribute, select it in the list box and select **Remove**.

g. Select **Update** to apply your changes.

## Collector Spare Configuration

Spare configuration enables you to add additional configuration to the tag using the **Spare 1** to **Spare 5** fields in all the collectors except in a Server-to-Server collector, Server-to-Server distributor, and an OSI PI distributor.

- In case of an OSI PI distributor, data is read from the Historian tag displayed in the **Source Address** field and sent to the OSI PI tag name displayed in the **Spare 1** field. To control the source and destination tags, change the **Source Address** and **Spare 1** values. You can add or update values in the remaining spare fields.
- In case of Server-to-Server collector and Server-to-Server distributor, you can update the **Spare 1** to **Spare 4** values, but the **Spare 5** field is used only for internal purposes. Therefore, do not update the **Spare 5** field.

# Data Collector Operation and Troubleshooting

## Data Collector File Locations

Historian data collector files are installed in the following default directories:

- **Executables**
  - OPC, Simulator, File, Calculation, and Server-to-Collectors

    ```
    c:\Program Files\Proficy\Historian\Server
    ```

  - iFIXCollector

    ```
    C:\Program Files (x86)\GE\iFIX
    ```

- **LOG files and SHOW files**

  ```
  C:\Historian Data\LogFiles (*.log, *.shw)
  ```

- Buffer files and local tag cache

  ```
  C:\Historian Data\BufferFiles (*.msq, *.cfg)
  ```

## Pause or Resume Data Collection for All Tags

1. To pause or resume data collection using Historian Administrator:
    a. Open Historian Administrator.
    b. Select the **Collectors** link.
    The **Collector Maintenance** page appears.
    c. From the list of collectors at the left of the page, select the collector you want to pause or resume.
    d. Select the appropriate running status from the **Collection Status** section in the **General** section.
    No data buffering occurs while collection is paused.
    e. Select **Update** to activate your change.
2. To pause or resume data collection using the Web Admin console:
    a. Go to the **Collectors Configuration** page.
    b. From the list of collectors at the left of the page, select the collector you want to pause or resume.
    c. Select **Pause** or **Resume**.
    No data buffering occurs while collection is paused.
3. To pause or resume data collection by accessing the local collector machine:
    a. On the **Start** menu, select **Run**.
    b. Type `services.msc` and select **OK**.
    The **Services** window appears.
    c. Select the **Historian** (*collector type*) **Collector** and select **Start** or **Stop**.
    d. Select **Close**.
4. To pause data collection from the Collector icon:

a. Select the Windows **Start** button, select Historian 6.0 from the **Programs** menu, and select the appropriate icon for your collector.
   The process should start and an application icon display should appear in minimized form on the lower toolbar.

b. To stop the collector, maximize the icon from the toolbar, type `s`, and press **Enter**.

## Pause Data Collection for a Subset of Tags

1. To pause data collection for a subset of tags using Historian Administrator:

   a. From Historian Administrator **Main** page, select the **Tags** link.
      The **Tag Maintenance** page appears.

   b. Select one or more tags in the **Tags** section.
      The right-hand column displays current parameters.

   c. Locate the **Collection** field in the **Collection** section.

   d. Select the **Disabled** option.

   e. Select **Update**.

2. To pause data collection using the Web Admin console:

   a. Go to the **Tags** page from the **Configuration** page.

   b. Select one or more tags in the **Tags** section.
      The **Tag Editor** section displays current parameters.

   c. Locate the **Collection** field in the **Collection** section.

   d. Disable the option.

   e. Select **Update**.
      Data collection stops for those tags. To re-activate collection for those tags, select **Enabled** for the **Collection** option and select **Update**.

> **Note:**
> If the collector is configured to allow online changes, making configuration changes such as the above may cause bad data samples to be recorded as the collector internally restarts. Disable online configuration changes and restart the collector manually if you want to avoid this behavior.

## Modify User Privileges for Starting a Collector

To start any collector, a user must have Power User privileges at a minimum. Typically, a user from the Administrator group starts a collector. If running as a service, you can use the local system account. If a user is not a Power User or Administrator, for instance, and you still want to allow that user to start a collector, you can override the user permissions setting in the Windows Registry.

The following example shows how to change the user permissions for a collector (the iFIX collector). While these steps outline the procedure for changing the user permissions for the iFIX collector, note that you must perform these steps individually for each collector that you want to allow the user to start.

To change the permissions in the Windows Registry, you must be an Administrator. After you change the permissions, you can exit the Registry Editor, allow the user to log in again, and then allow that user to restart the collector.

1. On the **Start** menu, select **Run**.
2. Type `regedt32` and select **OK**.
   The Registry Editor appears.
3. Navigate to the following registry key:

   ```
   HKEY_LOCAL_MACHINE\SOFTWARE\Intellution,Inc.\iHistorian\Services\iFixCollector
   ```

4. Right-select the **iFIX Collector** folder and select **Permissions**.
   The **Permissions for iFIXCollector** window appears.
5. Select the **Users** group in the top portion of the window.
6. Select the **Allow** check box from the **Full Control** permissions, in the bottom portion of the window.
7. Select **Apply**.

## About Monitoring Data Collector Performance Statistics

You can evaluate data collector performance by observing information displayed or recorded in the following pages or files:

- Historian Administrator **Main** page and Historian **Messages** page.

  For a detailed description of the parameter/option fields and the Alerts and Message Search Windows, refer to *Using Historian Administrator* manual.

- LOG and SHOW files on the data collector local machine.

  LOG (`.log`) files are historical journals of every event affecting operation of the collector. When you troubleshoot a problem in a collector, examining the log files is the best place to begin. The default path for LOG and SHW files is `C:\Proficy Historian Data\LogFiles`. The highest number is the most recent.

  SHOW (`.shw`) files allow you to examine the current configuration of a data collector. This file also details version and system configuration affecting the specific collector. The default path for LOG and SHW files is `C:\Proficy Historian Data\LogFiles`.

  If you are upgrading from a previous version of Historian, then the Archives, LogFiles, and BufferFiles destination paths will remain unchanged.

  Historian periodically checks for `Archives`, `Bufferfiles`, and `Logfiles` folder disk space availability. If the available disk space is less than configured, then Historian Data Archiver may shutdown.

- Event Viewer on the Historian Server and on the collector local machine.

  The Windows Event Viewer logs all system events of interest to an administrator or developer. Each event has an identifying icon, such as Information, Warning, or Error. Select an item to display more detail about the event. Use this information to determine when and why a server fault occurred and when satisfactory operation was restored.

- Historian tag using the source address of the Rate Output Address, Status Output Address, or heartbeat Output Address.

## Disabling Rebroadcasting for Historian Data Archiver

The Historian Data Archiver service rebroadcasts collector status reports to all connected clients to ensure they are immediately notified of any changes to their configuration. If you have many collectors (20 or more) pointing to a single Historian archive, you may experience network congestion.

1. Start the Windows Registry Editor.
2. Navigate to the `HKEY_LOCAL_MACHINE\SOFTWARE\Intellution, Inc\iHistorian\Services\DataArchiver` key

3. Create a new DWORD called `RouteInterfaceStatusMsg` and set it to `0` (to enable rebroadcasting, set this value to `1`).
4. Close the Windows Registry Editor.
5. Restart the Historian Data Archiver service.

> **Note:**
> If rebroadcasting is disabled in the Historian Data Archiver service, any programs subscribed to collector status changes using `Collectors.SubscribeStatus = true` will no longer automatically receive collector status messages. To receive collector status messages, periodically read the `Collector.Status` property instead.

## Troubleshooting Tag Configuration

To troubleshoot the tag configuration:

- View the Collector SHOW (`DataCollector.shw`) file in `C:\Historian Data\LogFiles\xxxCollector.shw` to review the current Data Collector configuration.
- View the `DataCollector.LOG` file in `C:\Proficy Historian Data\LogFiles\xxxCollector-01.log` to detect operational errors with Tags.
- If you are using aggregate data, such as average, minimum, or maximum, use a chart display such as the trend option or the Excel Add-In to sample and then observe how data is stored.
- If you are not certain about what is happening, turn off Collector Compression and Archive Compression and observe how raw data values flow into the archive. Be sure to export the tag configuration first so you can return to the original configuration.

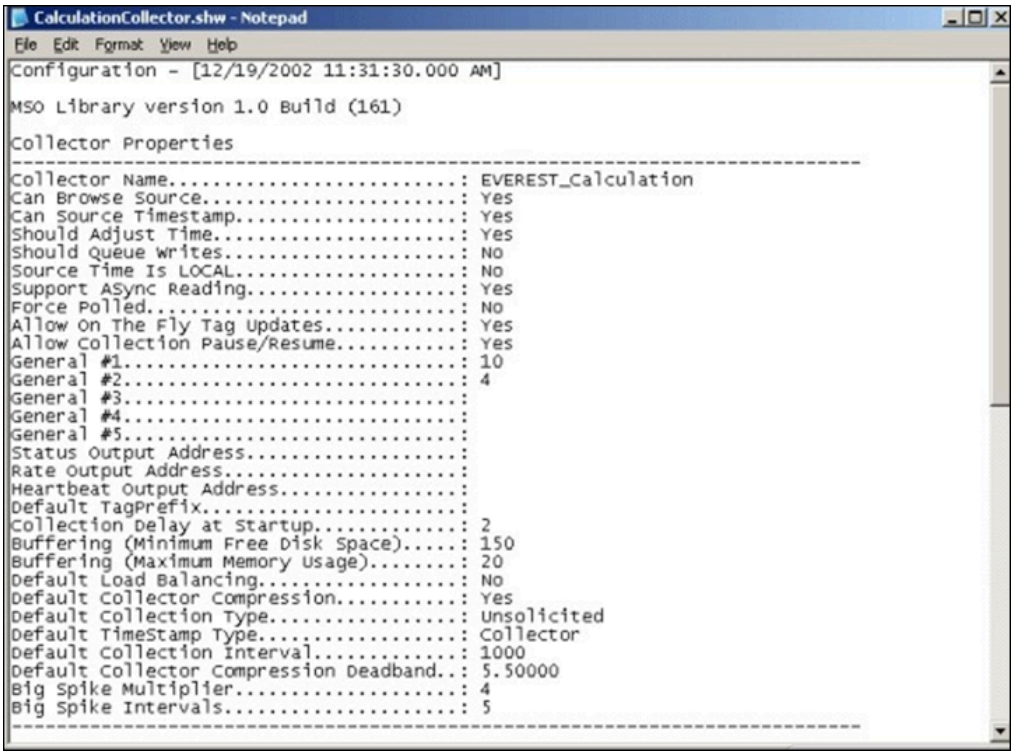
> **Note:**
> If you are upgrading from a previous version of Historian, then the `Archives`, `LogFiles`, and `BufferFiles` destination paths will remain unchanged. By default, `C:\Program Files\Proficy\Proficy Historian\Logfiles\`.

## Reviewing the Active Collector Configuration

You can view the active collector configuration via the Historian Administration too, and you can also view the local Data collector SHOW (`.shw`) file.

1. In Historian Administrator Main page, select the **Collectors** link. The Collector Maintenance page appears.
   The **Collector Maintenance** page appears.

2. Select a data collector in the left column. The right column fills with current configuration data.
3. Examine the current configuration settings to verify that they are appropriate.
   You can also review the active configuration by examining the local Data Collector SHOW(.SHW) File, as shown below:




4. Scan the contents of the file and verify that the configuration parameters are correct for the application.
   If any of the values are not appropriate:

   a. Go back Historian Administrator.

   b. Select the **Collectors** link to display the **Collector Maintenance** page.

   c. Select a collector.

   d. Access the various sections and enter new values as appropriate.

   e. Select **Update**.
      The new values are stored in the Data Collector SHW file after 30 seconds.

## Collector and Archive Compression

It is possible for collected tags with no compression to appear in Historian as if the collector or archive compression options are enabled. If collector compression occurs, you will notice an increase in the percentage of the Compression value from 0% in the **Collectors** panel of the **System Statistics** page in Historian Administrator. When archive compression occurs, you will notice the **Archive Compression** value and status bar change on the **System Statistics** page.

### Collector Compression

For all collectors, except the File collector, you may observe collector compression occurring for your collected data (even though it is not enabled) if bad quality data samples appear in succession. When a succession of bad data quality samples appears, Historian collects only the first sample in the series. No new samples are collected until the data quality changes. Historian does not collect the redundant bad data quality samples, and this is reflected in the Collector Compression percentage statistic.

For a Calculation or Server-to-Server Collector, you may possibly observe collector compression (even though it is not enabled) when calculations fail, producing no results or bad quality data.

### Archive Compression

If the **Archive Compression** value on the **System Statistics** page indicates that archive compression is occurring, and you did not enable archive compression for the tags, the reason could be because of internal statistics tags with archive compression enabled.

## Data Buffering

During normal operation, the Data Collector sends data and messages to the Historian Server using TCP/IP.The Server responds when it has successfully received the data.

Normal variations in response from the server can leave a small number of messages buffered in memory.When the collector loses its connection, or whenever the server cannot keep up with throughput, the data collector establishes a buffer. During such periods, the data collector continues to write data, caching it in the local file and memory buffer instead of writing it to the server. When the collector reestablishes the connection to the server, it forwards the stored data to the server, clearing the buffer as the server successfully receives the data.

If a collector writing to an archive loses its connection and the disk buffer becomes full, real-time collection does not begin immediately upon the re-established connection to the server. No data is collected from the time that the connection to the archive is re-established until approximately the time it takes for the buffer on the collector to clear.

**Note:**
The Data Buffering feature does not apply to File collectors. The File collector does not process incoming files when the connection to the server is down. When the connection is re-established, processing of incoming files resumes.

If there is not enough free space for a collector to create its buffer files on initial startup, the collector shuts down immediately and sends the following message to the Event Viewer:

```
"[datetime] MessageAdd -MDW_iFIX collector Buffering could not create buffer files. Shutting down."
```

If there is not enough free space for the collector to create its buffer files on startup, the collector shuts down and sends a message to the Event Viewer. The simplest way to prevent this from happening is to free up disk space to allow the collector to start. If this is not possible, you can edit the Registry to change the buffer size.

## Editing the Registry to Change the Buffer Size

1. Select **Run** from the **Start** menu.
   The **Run** window appears.
2. Type `Regedit` and press **Enter**.
   The **Registry Editor** appears.
3. Locate the ComputerName_OPC1 key, where ComputerName is the name of your computer. You can find this key here:
   ```
   HKEY_LOCAL_MACHINE\SOFTWARE\Intellution,Inc.\Historian\Services\OPCCollector\ComputerName_OPC1
   ```
4. Add a new DWORD value. Enter the name `MinimumDiskFreeBufferSize`, select the decimal option, and set the value to a small number such as 10 or 20. The value that you enter represents the number of MB of buffer space needed.
5. Start the collector.
6. If the Collector does not connect to the Historian Server, check the log file in the logfiles folder on the collector computer. If the log states that Historian "could not create buffer files" then repeat steps 1-5, this time using a smaller number.
7. Once the Collector connects to the Historian Server, the collector should appear in the Admin UI and you can readjust the buffer file size on the bottom of the collector's **General** section.

## Setting Up Services Recovery Actions in Windows

Windows allows you to set up recovery actions to take place if a service fails. If you are running Historian Collectors on Windows, set recovery actions to restart the service for your individual collectors.

To set recovery actions for a specific service:

1. Open the **Control Panel**.
2. Double-click the **Administrative Tools** icon.
3. Double-click the **Services** icon in the **Administrative Tools** window.
   The **Services** window appears.
4. Right-select the Service you want to set recovery options for.
5. Select **Properties**.
   The **Service Properties** window appears.
6. Select **Recovery**.
7. Select the recovery actions you want in the **First attempt failure**, **Second attempt failure** and Subsequent attempts failure fields.
   For more information on setting Services Recovery Actions, refer to the Microsoft Management Console online Help.

# Working with Python Expression Tags

## Python Expression Tags in Historian

A Python® expression is typically a single line of Python code which, when executed, evaluates to a single value. It can also be thought of as the right-hand side of a Python assignment operation.

Python Expression Tags are used in cases where you do not want to store a raw data value, but wish to store only derived or calculated values. Historian allows raw data to be pre-processed with Python expressions during collection, so that the data collected for the expression tags contain these derived values.

See www.python.org/ for complete Python documentation.

**Note:**
This functionality is available only for the Enterprise edition of Historian.

To use a Python Expression Tag in Historian, you must configure a tag as a Python Expression Tag using either Historian File collector or Historian Excel Add-in. You can then use the Python expression on this tag to pre-process raw data before the result is inserted into Historian.

The following collectors support Python Expression Tags:

- Simulation Data Collector
- iFix Data Collector
- OPC Data Collector
- OSI PI Collector
- Windows Performance Collector
- Collector Toolkit

To enable Python Expression support for toolkit-based collectors, copy the following files to the collector executable path:

- `python34.dll`
- `ihcpprest140_2_8.dll.dll`
- `PythonExpressionExtension.dll`

For 32-bit collectors, these files are located at `C:\Program Files\Proficy\Proficy Historian \x86\Server`. For 64-bit collectors, these files are located at `C:\Program Files\Proficy\Proficy Historian\x64\Server`.

## Supported Python Modules

This table lists the available Python modules. Within the approved modules below, certain Python classes, functions, and keywords may not be available for use, for security reasons. For example, the `exec` function cannot be used.

**Note:**
Python modules not listed in the table are not supported.

| Standard Python Modules | Documentation | Needs Importing |
|---|---|---|
| `builtins` | Standard Python documentation | No |
| `datetime` | Standard Python documentation | Yes |
| `math` | Standard Python documentation | Yes |
| `statistics` | Standard Python documentation | Yes |
| **Module part of Historian install** | Unit of Measure Python Module.pdf<br><br>(This document can also be found in the help directory for your Historian installation.) | Yes |

| Standard Python Modules | Documentation | Needs Importing |
|---|---|---|
| `uom` (unit of measure conversion module) | | |

## Constructing and Adding Python Expression Tags

To configure and use the Python Expression Tag in Historian:

1. Construct a Python Expression. *(on page 2080)*
2. Construct the JSON configuration. *(on page 2080)*
3. Add the Python Expression Tag to Historian. *(on page 2082)*
   For examples, see the Python Expression Tag Examples *(on page 2084)*.

## Constructing a Python Expression

1. Open a text editor.
2. Construct an expression in Python that takes raw data and produces a calculated value you want to store in Historian.
   - The raw data in your expression is exposed in the Python script as an object with the following properties: `value`, `is_quality_good`, `is_quality_bad`, `timestamp`
   - You can give this object any name you want.
3. Ensure that the expression you create is a valid Python expression (that is, it evaluates to a single value).
   For example, we call our object `temp` and we use the `value` property. We create a simple, valid Python expression, taking raw data (`temp.value`) and create a calculated value.

   ```
   temp.value + math.pow(10,temp.value/70)
   ```

   To use this example, `temp.value` must be defined and the Python `math` module must be imported.

## Constructing the JSON Configuration

Construct a JSON configuration object that contains the Python expression in the same text editor. This allows you to specify Python modules that you require and to link the variables (parameters) used in the expression to a raw data source.

It is easier to edit your JSON in the Historian Excel Add-In, which you will use later to add the tag to Historian, see Adding a Python Expression Tag to Historian *(on page 2082)*.

The JSON needs to be *"minified"* and it needs to conform to a particular structure. **Example of JSON containing a Python Expression in "Minified" format**

```
{"imports":["math"],"script":"temp.value +

math.pow(10,temp.value/70)","parameters":[{"name":"temp","source":{"add
```

Here is an example of JSON structure in *"Beautified"* format. The **highlighted** entries are placeholders.

```
{
"imports":["<imported_module0>", "<imported_module1>",
"<imported_module2>",...], "script":"<python_expression>",
"parameters":[
{
"name":"<variable_name>", "source":{"address":"<source_address>",
"dataType":"<datatype_of_parameter_value_field>"}
},...
]
}
```

After `"imports"`, list the Python modules you need to import for your Python expression. See Supported Python Modules *(on page 2079)*.

After `"script"`, enter the python expression that you created in Constructing a Python Expression *(on page 2080)*.

Provide values for the following parameters:

| Parameter | Description |
|---|---|
| name | The variable name you gave to the object representing the raw data which will be pre-processed by the expression you created. This variable is exposed in the Python script as an object with the following properties: `value`, `is_quality_good`, `is_quality_bad`, `timestamp`. |
| address | The source address used by the collector to access the raw data. This parameter is stipulated in the context of the chosen collector, which must be on the list of collectors *(on page 2078)* supporting Python Expression Tags. |
| dataType | The datatype of the `\value` field for the variable representing the raw data. This allows the Python expression to know the data type on which it is operating.<br><br>The datatype options are: `SingleFloat, DoubleFloat, SingleInteger, DoubleInteger, QuadInteger, UnsignedSingleInteger, UnsignedDoubleInteger, UnsignedQuadInteger,` |

| Para-meter | Description |
|---|---|
| | `FixedString, VariableString, Byte, Boolean`. These are the same Historian data types that are supported by the File collector. |

**Example of JSON containing a Python Expression in *"Beautified"* format**

This example uses the Simulation Collector. Your collector might use a different source address.

```
{
 "imports":["math"],
 "script":"temp.value+math.pow(10,temp.value/70)",
 "parameters":[{"name":"temp","source":
{"address":"Simulation00001","dataType":"SingleFloat" } } ]
}
```

It is important to check that your JSON is valid, since no validation will be performed on the JSON at tag creation.

**Minified JSON**

Once you have constructed this JSON, you need to format it as a *"minified"* string, so that it can be processed in later steps. Minified JSON has no newline characters or comments. There are tools which can help you minify JSON.

For our example, the minified JSON would look like this:

```
{"imports":["math"],"script":"temp.value + math.pow(10,temp.value/70)","parameters":
[{"name":"temp","source":{"address":"Simulation00001","dataType":"SingleFloat"}}]}
```

Pay attention to escape characters in your JSON. If your JSON contains a \ character, you need to escape it. So, \ becomes \\ (since \ is used to escape another \).

## Adding a Python Expression Tag to Historian

You can now add a Python Expression Tag to Historian. Ensure the following for your Python Expression Tag:

- `CalcType` is set to `"PythonExpr"`.
- `SourceAddress` contains the JSON configuration.

Adding the Python Expression Tag can be done using Historian File collector or Historian Excel Add-in in the same way that you would add a regular tag.

For more information on adding tags, refer to Historian File collector or Historian Excel Add-in.

- *File collector* (especially the *CSV File Formats* and *XML File Formats* sections)

  If you add your tag to the Historian via the File collector and using the CSV format, you must enclose the JSON in quotation marks (") to satisfy CSV requirements for a column value containing commas (,).

  This means that you will also need to escape any quotation marks (") in the JSON. That is, " becomes "" (as " is used to escape another ").

- *Using the Historian Excel Add-in*

  The Historian Excel Add-In has the advantage of being easiest to use for editing your JSON.

## Example of Adding a Tag Using the File collector

For this example, we choose to add a Python Expression tag to the Historian using the File collector to import a CSV file.

The file contents look like this:

```
[Tag]
Tagname,CollectorName,CalcType,SourceAddress,DataType,Description
ExampleTag,SimulationCollector,PythonExpr,
"{""imports"":[""math""],""script"":""temp.value +
math.pow(10,temp.value/70)"",""parameters"":[{""name"":""temp"",""source"":{""address"":""Simulation00001"",""dataType"
":""SingleFloat""}}]}",
SingleFloat,Python Expression Tag example
```

Note the following:

- The `CalcType` header is included and set to `PythonExpr`.
- The Source Address is set to the minified JSON created in the previous step.
- The `CollectorName` is set to `SimulationCollector`, which is a Simulation Collector. This collector is on the list of collectors supporting Python Expression Tags. Your collector might be called by a different name.

- The quotation marks within the JSON string are escaped with other quotation marks in the CSV file.

  For more information, refer to File collector > CSV file format.

## Viewing the Python Expression Tag

When viewing a Python Expression Tag using Historian Administrator, note that:

- The Source Address field contains the full applicable JSON configuration, which includes an indication of the source address. Changing this is not recommended.
- The **Browse** button for configuring the Source Address is disabled for Python Expression Tags.

## Python Expression Tag Examples

This section provides examples for using Python® Expression Tags.

### Using No Python Modules or Functions

In this example, we want to perform gross error detection on a signal `"Signal"` and clip the values to a range between 0 and 600.

#### Expression

To solve this, we create the following Python expression.

```
0 if Signal.value<0 else (600 if Signal.value>600 else Signal.value)
```

| | Python Datatype | Name | Description |
|---|---|---|---|
| Expression Inputs | `SingleFloat` | `Signal` | Represents the signal value. |
| Expression Result | `SingleFloat` | Not Applicable | Represents the resulting expression value, with extreme outliers clipped. |

**Python Modules to Import for this Expression:** None. (Only modules contained on the list of supported modules *(on page 2079)* are available for this expression.)

#### Constructing the JSON:

Using the created expression, we construct the following JSON. For more information, refer to Constructing the JSON Configuration *(on page 2080)*.

```
"script":"0 if Signal.value<0 else (600 if Signal.value>600 else Signal.value)", "parameters":[
                    {
                    "name":" Signal",
                    "source":{"address":"Simulation00001", "dataType":"SingleFloat"}
                    }
                    ]
                    }
```

Note that the `address` parameter is stipulated in the context of the chosen collector, which must be on the list of collectors *(on page 2078)* supporting Python Expression Tags.

In this example, we have used the Simulation Collector. Your collector might use a different source address.

We then transform the above into a minified JSON string:

```
{"script":"0 if Signal.value < 0 else (600 if Signal.value > 600 else Signal.value)","parameters":
                    [{"name":"Signal","source":{"address":"Simulation00001",dataType":"SingleFloat"}}]}
```

**Adding the Expression Tag to Historian**

For this example, we choose to add a Python Expression tag to the Historian using the File collector to import a CSV file. (We could also have added the tag via the Historian Excel Add-In.)

The file contents look like this:

```
[Tag]


 Tagname,CollectorName,CalcType,SourceAddress,DataType,DescriptionGEDSignalTag,SimulationCollector,PythonExpr,
                    "{""script"":""0 if Signal.value < 0 else (600 if Signal.value > 600 else
 Signal.value)"",""parameters"":


 [{""name"":""Signal"",""source"":{""address"":""Simulation00001"",""dataType"":""SingleFloat""}}]}",
                    SingleFloat,Python Expression Tag example
```

Note the following:

- The `CalcType` header is included and set to `PythonExpr`.
- The `Source Address` is set to the minified JSON created in the previous step.

- The `CollectorName` is set to `SimulationCollector`, which is a Simulation Collector. Your collector might be called by a different name.
- The quotation marks within the JSON string are escaped with other quotation marks.

We then import the file, following the instructions specified in File collector.

## Using a Bulit-In Python Function

In this example, we want to calculate the maximum of two temperature values to be collected.

### Expression

To solve this, we create the following Python expression:

```
max(ThermocoupleA.value, ThermocoupleB.value)
```

| | Python Datatype | Name | Description |
|---|---|---|---|
| Expression Inputs | `SingleFloat` | `ThermocoupleA` | Represents a temperature value. |
| | `SingleFloat` | `ThermocoupleB` | Represents a temperature value. |
| Expression Result | `SingleFloat` | Not Applicable | Represents the maximum of the two given temperature values |

**Python Modules to Import for this Expression:** None. A built-in Python module from the Python Standard Library is used. (Only modules contained on the list of supported modules *(on page 2079)* are available to this expression.)

### Constructing the JSON

Using the created expression, we construct the following JSON:

```
{
 "script":"max(ThermocoupleA.value,ThermocoupleB.value)",
 "parameters":[
  {
  "name":"ThermocoupleA",
  "source":{"address":"Simulation00001","dataType":"SingleFloat"}
  },
  {
  "name":"ThermocoupleB",
  "source":{"address":"Simulation00002","dataType":"SingleFloat"}
```

```
  }
 ]
}
```

Note that the `address` parameter is stipulated in the context of the chosen collector, which must be on the list of collectors *(on page 2078)* supporting Python Expression Tags. In this example, we have used the Simulation Collector. Your collector might use a different source address.

We then transform the above into a minified JSON string:

```
{"script":"max(ThermocoupleA.value,ThermocoupleB.value)","parameters":[{"name":"ThermocoupleA","source":
{"address":"Simulation00001","dataType":"SingleFloat"}},{"name":"ThermocoupleB","source":
{"address":"Simulation00002","dataType":"SingleFloat"}}]}
```

#### Adding the Expression Tag to Historian

For this example, we choose to add a Python Expression tag to the Historian using the Historian Excel Add-In. (We could also have added the tag by using via the File collector to import a CSV file. )

Ensure the following for your Python Expression Tag:

- The `CalcType` is set to `PythonExpr`.
- The `SourceAddress` contains the JSON configuration.
- The `CollectorName` is set to the name of the chosen collector, which must be on the list of collectors supporting Python Expression Tags. Your collector might be called by a different name.
- The quotation marks within the JSON string are escaped with other quotation marks in the CSV file.

Otherwise, the tag is added in exactly the same way as for a regular tag.

### Using A Python Standard Library Module

In this example we want to calculate a result based on a specific time range for an expression input. We set a supply voltage to zero within prescribed time ranges.

#### Expression

To solve this, we create the following Python expression:

```
0 if (SupplyVoltage.timestamp.astimezone().time() &gt;= datetime.time(18) and
SupplyVoltage.timestamp.astimezone().time() &lt;= datetime.time(20, 30)) else
SupplyVoltage.value
```

| | Python Datatype | Name | Description |
|---|---|---|---|
| Expression Inputs | `SingleFloat` | `SupplyVoltage` | Represents the value we wish to transform. |
| Expression Result | `datetime` | Not Applicable | Represents the resulting supply voltage, set to zero in the prescribed time ranges. |

**Python Modules to Import for this Expression:** `datetime` module. This module is shipped with Historian.

**Constructing the JSON**

Using the created expression, we construct the following JSON:

```
{
 "imports":["datetime"],
 "script":"0 if (SupplyVoltage.timestamp.astimezone().time() >= datetime.time(18) and
SupplyVoltage.timestamp.astimezone().time() <= datetime.time(20, 30)) else
SupplyVoltage.value","parameters":[
  {
  "name":" SupplyVoltage",
  "source":{"address":"Simulation00001", "dataType":"SingleFloat"}
  }
 ]
}
```

Note that the `address` parameter is stipulated in the context of the chosen collector, which must be on the list of collectors *(on page 2078)* supporting Python Expression Tags. In this example, we have used the Simulation Collector. Your collector might use a different source address.

We then transform the above into a minified JSON string:

```
{"imports":["datetime"],"script":"0 if (SupplyVoltage.timestamp.astimezone().time() >= datetime.time(18) and
SupplyVoltage.timestamp.astimezone().time() <= datetime.time(20, 30)) else
SupplyVoltage.value","parameters":[{"name":"SupplyVoltage","source":
{"address":"Simulation00001","dataType":"SingleFloat"}}]}
```

**Adding the Expression Tag to Historian**

For this example, we choose to add a Python Expression tag to the Historian using the Historian Excel Add-In. (We could also have added the tag via the File collector to import a CSV file.)

Ensure the following for your Python Expression Tag:

- The `CalcType` is set to `PythonExpr`.
- The `SourceAddress` contains the JSON configuration.
- The `CollectorName` is set to the name of the chosen collector, which must be on the list of collectors supporting Python Expression Tags. Your collector might be called by a different name.

Otherwise, the tag is added in exactly the same way as for a regular tag.

## Using A Python Standard Library Module

In this example we want to calculate a result based on a specific time range for an expression input. We set a supply voltage to zero within prescribed time ranges.

### Expression

To solve this, we create the following Python expression:

```
0 if (SupplyVoltage.timestamp.astimezone().time() &gt;= datetime.time(18) and
SupplyVoltage.timestamp.astimezone().time() &lt;= datetime.time(20, 30)) else
SupplyVoltage.value
```

| | Python Datatype | Name | Description |
|---|---|---|---|
| Expression Inputs | `SingleFloat` | `SupplyVoltage` | Represents the value we wish to transform. |
| Expression Result | `datetime` | Not Applicable | Represents the resulting supply voltage, set to zero in the prescribed time ranges. |

**Python Modules to Import for this Expression:** `datetime` module. This module is shipped with Historian.

### Constructing the JSON

Using the created expression, we construct the following JSON:

```
{
 "imports":["datetime"],
 "script":"0 if (SupplyVoltage.timestamp.astimezone().time() >= datetime.time(18) and
SupplyVoltage.timestamp.astimezone().time() <= datetime.time(20, 30)) else
SupplyVoltage.value","parameters":[
```

```
  {
  "name":" SupplyVoltage",
  "source":{"address":"Simulation00001", "dataType":"SingleFloat"}
  }
 ]
}
```

Note that the `address` parameter is stipulated in the context of the chosen collector, which must be on the list of collectors *(on page 2078)* supporting Python Expression Tags. In this example, we have used the Simulation Collector. Your collector might use a different source address.

We then transform the above into a minified JSON string:

```
{"imports":["datetime"],"script":"0 if (SupplyVoltage.timestamp.astimezone().time() >= datetime.time(18) and
SupplyVoltage.timestamp.astimezone().time() <= datetime.time(20, 30)) else
SupplyVoltage.value","parameters":[{"name":"SupplyVoltage","source":
{"address":"Simulation00001","dataType":"SingleFloat"}}]}
```

#### Adding the Expression Tag to Historian

For this example, we choose to add a Python Expression tag to the Historian using the Historian Excel Add-In. (We could also have added the tag via the File collector to import a CSV file.)

Ensure the following for your Python Expression Tag:

- The `CalcType` is set to `PythonExpr`.
- The `SourceAddress` contains the JSON configuration.
- The `CollectorName` is set to the name of the chosen collector, which must be on the list of collectors supporting Python Expression Tags. Your collector might be called by a different name.

Otherwise, the tag is added in exactly the same way as for a regular tag.

### Using A Historian Python Module

In this example we want to convert a temperature value from Fahrenheit to Celsius.

#### Expression

To solve this, we create the following Python expression:

```
uom.to_Celsius(Thermocouple.value, uom.Temperature.Fahrenheit)
```

| | Python Datatype | Name | Description |
|---|---|---|---|
| Expression Inputs | `SingleFloat` | `Thermocouple` | Represents the temperature value in degrees Fahrenheit |
| Expression Result | `SingleFloat` | Not Applicable | Represents the temperature value in degrees Celsius |

**Python Modules to Import for this Expression:** uom module. This Python module is shipped with Historian.

(Only modules contained on the list of supported modules *(on page 2079)* are available to this expression.)

**Constructing the JSON**

Using the created expression, we construct the following JSON:

```
{
 "imports":["uom"],
 "script":"uom.to_Celsius(Thermocouple.value, uom.Temperature.Fahrenheit)",
 "parameters":[
  {
  "name":" Thermocouple",
  "source":{"address":"Simulation00001", "dataType":"SingleFloat"}
  }
 ]
}
```

Note that the `address` parameter is stipulated in the context of the chosen collector supporting Python Expression Tags. In this example, we have used the **Simulation Collector**. Your collector might use a different source address.

We then transform the above into a minified JSON string:

```
{"imports":["uom"],"script":"uom.to_Celsius(Thermocouple.value,uom.Temperature.Fahrenheit)","parameters":
[{"name":"Thermocouple","source":{"address":"Simulation00001","dataType":"SingleFloat"}}]}
```

**Adding the Expression Tag to Historian**

For this example, we choose to add a Python Expression tag to the Historian using the File collector to import a CSV file. (We could also have added the tag via the Historian Excel Add-In.)

The file contents look like this:

```
[Tag]
Tagname,CollectorName,CalcType,SourceAddress,DataType,DescriptionConvertedTempTag,SimulationCollector,PythonExpr,
"{""imports"":[""uom""],""script"":""uom.to_Celsius(Thermocouple.value,uom.Temperature.Fahrenheit)"",
""parameters"":[{""name"":""Thermocouple"",""source"":{""address"":""Simulation00001"",""dataType"":
""SingleFloat""}}]}",
SingleFloat,Python Expression Tag example
Note the following: The CalcType header is included and set to PythonExpr.
```

Note the following:

- The `CalcType` header is included and is set to `PythonExpr`.
- The `SourceAddress` contains the JSON configuration.
- The `CollectorName` is set to the name of the chosen collector, which must be on the list of collectors supporting Python Expression Tags. Your collector might be called by a different name.
- The quotation marks within the JSON string are escaped with other quotation marks in the CSV file.

  For more information, see *File collector* > CSV File Formats.

We then import the file, following the instructions in the File collector section.

## Using Array/Table Lookup

In this example we want to translate a string representing order of magnitude into a corresponding numerical value using array/table lookup.

This example will be explained by means of a hypothetical collector called `PlantSensorCollector` that is a Python Expression enabled collector. The collector collects a source tag with address `TemperatureSetpoint` of type `VariableString`, having values `'Low', 'Medium', and 'High'`.

### Expression

To solve this, we create the following Python expression:

```
{'Low':100, 'Medium':400, 'High':800}.get(Setpoint.value, 0)
```

| | Python Datatype | Name | Description |
|---|---|---|---|
| Expression Inputs | `VariableString` | `Setpoint` | Represents the given ordinal string value we wish to transform. |
| Expression Result | `SingleFloat` | Not Applicable | Represents the numerical value corresponding to the ordinal string input. |

**Python Modules to Import for this Expression:** None. (Only modules contained on the list of supported modules *(on page 2079)* are available to this expression.)

**Constructing the JSON:** Using the created expression, we construct the following JSON:

```
{
                 "script":"{'Low':100,'Medium':400,'High':800}.get(Setpoint.value, 0)",
                 "parameters":[
                  {
                  "name":" Setpoint",
                  "source":{"address":"TemperatureSetpoint", "dataType":"VariableString"}
                  }
                 ]
                }
```

We then transform the above into a minified JSON string:

```
{"script":"{'Low':100,'Medium':400,'High':800}.get(Setpoint.value,0)","parameters":
                [{"name":"Setpoint","source":{"address":"TemperatureSetpoint","dataType":"VariableString"}}]}
```

**Adding the Expression Tag to Historian:** For this example, we choose to add a Python Expression tag to the Historian using the File collector to import a CSV file. (We could also have added the tag via the Historian Excel Add-In.)

The file contents look like this:

```
[Tag]Tagname,CollectorName,CalcType,SourceAddress,DataType,DescriptionNumericalTagDerivedFromOrdinalVal,PlantSensorColl
ector,PythonExpr,
                "{""script"":""{'Low':100,'Medium':400,'High':800}.get(Setpoint.value,0)"",""parameters"":


 [{""name"":""Setpoint"",""source"":{""address"":""TemperatureSetpoint"",""dataType"":""VariableString""}}]}",
                VariableString, Python Expression Tag example
```

Note the following:

- The `CalcType` is set to `PythonExpr`.
- The `SourceAddress` contains the JSON configuration.
- The `CollectorName` is set to the name of the chosen collector, which must be on the list of collectors supporting Python Expression Tags. Your collector might be called by a different name.
- The quotation marks within the JSON string are escaped with other quotation marks in the CSV file.

  For more information, see *File collector > CSV File Formats*.

We then import the file, following the instructions specified in *File collector*.

# Uninstall Collectors

If you want to uninstall collectors, delete all the instances of the collectors that you have created. You can do so using Configuration Hub *(on page 654)* or Remote Collector Management *(on page 817)*. The following instances of collectors (the ones that were created during the installation of collectors) are deleted automatically:

- The iFIX collector
- The iFIX Alarms & Events collector
- The OPC Classic Data Access collector for CIMPLICITY
- The OPC Classic Alarms and Events collector for CIMPLICITY

Even after you uninstall collectors and Web-based Clients, the corresponding Windows services and registry entries are not removed.

1. Select **Programs / Uninstall a Program** in Control Panel.
2. Select **Historian Collectors**, and then select **Uninstall**.
   The collectors are uninstalled.

# Troubleshooting Issues with Collectors

This topic contains solutions/workarounds to some of the common issues encountered with Configuration Hub. This list is not comprehensive. If the issue you are facing is not listed on this page, refer to Troubleshooting the Historian Server *(on page 298)*.

**Proxy Timeout Error While Adding a Collector Instance**

**Description**

When you attempt to add a collector, sometimes, a proxy error appears even if the error occurs because of an API timeout.

**Workaround**

1. Access the `historian-httpd.conf` file located at `C:\Program Files\GE \Operations Hub\httpd\conf\app-specific.d`.
2. Increase the timeout value (for example, 250). If the timeout parameter is not available, enter it as follows: `ProxyTimeout <value>`
3. Restart the Proficy Historian Tomcat Server and the Proficy Operations Hub Httpd Reverse Proxy services.

### Unable to Use the Windows login User and Password During the Destination Historian Server Configuration When Adding a Collector Instance

**Description**

During Collector instance creation, for some reason, if you are unable to use the Windows login user and password during the Destination Historian server configuration, you can perform the following workaround.

**Workaround**

- Edit the **Password** and **UserName** registries and add the predefined user, that is, <hostname>.admin and its password in the Collector Manager registry at `Computer \HKEY_LOCAL_MACHINE\SOFTWARE\GE DIGITAL\Historian Remote Management Agents\Collector Manager`.
- Check if the predefined user, that is, <hostname>.admin was added to the **Local users & Group**. If not, add the predefined user and its password to the **Local users & Group**. For more information on how to add a user to **Local users & Group**, refer to Adding Users to Windows Security Group.

### Receiving a Collector Configuration Error

**Description**

A single `ihConfigurationGetProperties[-2]` error appears in the `collector.LOG` file.

**Workaround**

The error most likely occurred as a result of the collector connecting and querying for changes in the tag database immediately, getting a timeout, and then immediately querying again and succeeding.

**Polled-based Collector performance improvements leading to "Load on source" and "Increase Overruns"**

**Description**

From Historian 2022 onwards, the performance of polled-based collectors has been enhanced by introducing a higher number of polled threads, set to a default of 4, compared to the previous default of 1. While this enhancement has a positive impact on the collector performance, it may cause a decrease in performance on the source side due to increased load and also cause potential overruns. Therefore, if you are facing any performance issue on the source side, you can modify the number of polled threads using the following workaround.

**Workaround**

1. Open Registry Editor.
2. Create **REG_NumPolledReadThreads** as a new **DWORD (32-bit) Value** in the following registry paths, and then change its value to **1**.

   **32-bit collector registry path for Proficy products**:

   ```
   Computer\HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Intellution,
   Inc.\iHistorian\Services
   ```

   **32-bit collector registry path**:

   ```
   Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\GE Digital
   \iHistorian\Services
   ```

   **64-bit collector registry path**:

   ```
   Computer\HKEY_LOCAL_MACHINE\SOFTWARE\GE Digital\iHistorian
   \Services
   ```

   For example,

   **iFIX Collector**

   ```
   Computer\HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Intellution,
   Inc.\iHistorian\Services\iFixCollector
   ```

   **Simulation Collector**

   ```
   Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\GE Digital
   \iHistorian\Services\SimulationCollector
   ```

**OPC Collector**

```
Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Intellution,
Inc.\Historian OPC Collector
```

3. Ensure to restart the corresponding collector.

# Chapter 15. The Calculation Collector

## Overview of the Calculation Collector

Using the Calculation collector, you can perform data calculations on values already in the archiver. It retrieves data from tags in the Historian archive, performs the calculation, and then stores the resulting values into new archive tags. You can then access these tags like you access any other Historian tag.

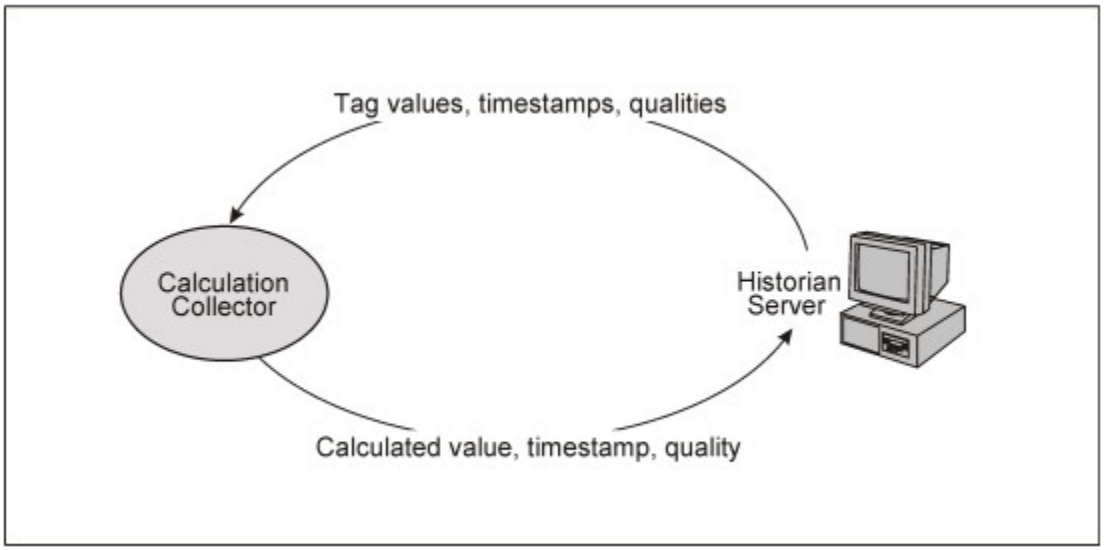
### Features of the Calculation Collector

The Calculation collector performs calculations on the following values:

- Current values of other Historian tags in the same archiver.
- Previous raw samples of other tags in the same archiver.
- Calculated values of other Historian tags in the same archiver, such as minimum, maximum, average, or standard deviation. You can specify a time range for these calculations or perform a filtered query. You can use the resulting single number in a calculation.
- Interpolated values of other Historian tags in the same archiver.
- Any data retrievable using a VBScript, file I/O, ADO, and so on.

### Data Flow

The following image shows the data flow in a Calculation collector. The output from a calculation is always a single Historian tag.



### Advantages of Using the Calculation Collector

- The Calculation collector can keep a history of the calculated values.
- It can perform thousands of calculations per second. Therefore, it is generally preferred to a VB SDK program performing the same functions.
- It can perform calculations on data stored in the following sources:
    - A SCADA database (such as iFIX)
    - A VB SDK program
    - A Historian collector (using input scaling)
    - By the Calculation collector
    - The Historian OLE DB provider
    - Reporting tools such as Crystal Reports
    - The Historian Excel Add-In

### Limitations

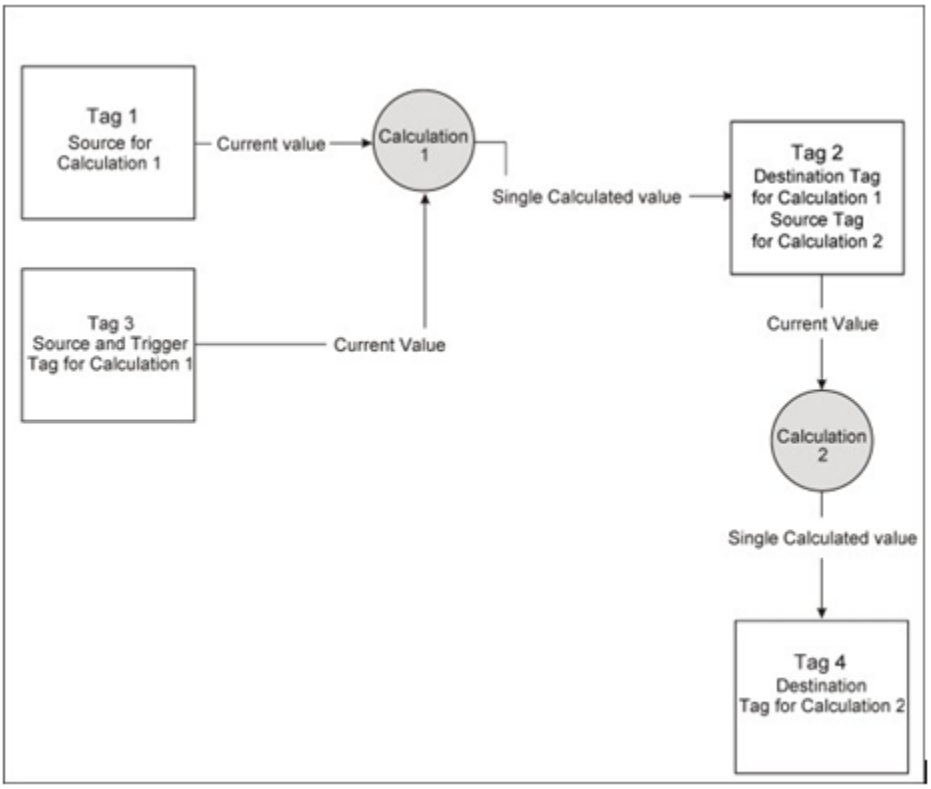
- Python Expression Tags are not supported for use together with the Calculation collector. The behavior of the Calculation collector is different from that of Python Expression Tags, which are used in cases where you do not want to store a raw data value, but wish to store only derived values. For more information, refer to Python Expression Tag Examples *(on page 2084)*.
- You cannot store specific numeric qualities or sub-qualities in a calculation formula. You can only set a good or bad data quality in the Calculation collector; that is you can store Good for NonSpecific (when quality > 0 in the calculation formula) or Bad for CalcFailed (when quality = 0 in the calculation formula).
- By design Calculation collector manages good quality values only.

## About the Tags Used by the Calculation Collector

The Calculation collector uses the following types of Historian tags in a calculation:

- **One or more source tags:** These tags are used in the calculation formula.
- **One or more trigger tags:** These tags trigger the calculation. Triggers can be polled (schedule-based) *(on page 2112)* or unsolicited (event-based) *(on page 2114)*.
- **A single destination tag:** This tag stores the result of the calculation. It is also called a calculation tag. It must be different from the source tags of the calculation. However, a destination tag can be a source tag for another calculation.

The following image shows how you can use the Calculation collector to perform a more complex calculation with several tags and multiple calculations resulting in one output tag. It also shows how the output of one calculation can be used as an input to another.




## Workflow for Using the Calculation Collector

After you have added an instance of the Calculation collector, to perform a calculation using the collector, you must perform the following steps:

| Step Number | Description | Notes |
|---|---|---|
| 1 | Create a tag administrative security group *(on page 291)*. In addition to defining the iH Tag Admins who have the power to create, modify, and remove tags, you can also define individual tag level security to protect sensitive tags. | This step is required to prevent a user from intentionally or unintentionally launching malicious VBScript code. For example, if a user connected to the data archiver created a calculation tag to file system object, that user could potentially delete all the files on your hard drive. To pre- |

| Step Number | Description | Notes |
|---|---|---|
| | | vent this, implement tag-level security. |
| 1 | Configure the Calculation collector *(on page 2101)*. | This step is required only if you want to change the default values for collector-specific parameters. |
| 2 | Create the source and destination tags *(on page 2099)*. To do so, you can add a tag manually using Historian Administrator *(on page 2057)* or the Web Admin console *(on page 2059)*. Or, you can copy a tag *(on page 2060)*.<br><br>**Note:**<br>You cannot browse a Calculation collector through the **Add Tags From Collector** window in Historian Administrator. | This step is required. |
| 3 | Assign a trigger to the destination tag *(on page 2114)*. | This step is required only if the tag is unsolicited (that is, event-based). |
| 4 | Define the calculation formula *(on page 2118)*. | This step is required. |

# Configure the Calculation Collector

1. Install the Historian server *(on page 104)* and collectors *(on page 142)*.
2. Create an instance of the collector using Configuration Hub *(on page 479)* or the RemoteCollectorConfigurator utility *(on page 797)*.

1. To configure the collector using Configuration Hub:

    a. Access Configuration Hub *(on page 336)*.

    b. Select **Collectors**, and then select the Calculation collector instance that you want to configure.
    The fields specific to the collector instance appear in the **DETAILS** section.

    c. Enter values as specified in the following table.

| Field | Description |
|---|---|
| Calculation Timeout (sec) | The maximum time a calculation must be performed before being terminated. The default value is 10 seconds. If the calculation takes longer, it is canceled, and a bad data quality sample is stored in the destination tag with a subquality, calculation error. |
| Max Recovery Time (hr) | The maximum time, in hours till now, that the collector will attempt to restore data. This is applicable only to event-based tags. The default value is 4 hours.<br><br>If you want to disable automatic calculation of the tag, set the value of this field to 0. |
| **MTLS Security** | Indicates whether you want to use Mutual TLS (MTLS) protocol to enforce a secure and strong authentication mechanism. |
| **MTLS Data Encryption** | Indicates whether you want to encrypt the data that the collector shares to the data archiver (DA). |
| For more information on how to enable MTLS Security, refer to Enable MTLS Security *(on page 632)*. | |

    d. As needed, enter values in the other sections common to all collectors *(on page 579)*.

    e. Restart the collector.

2. To configure the collector using Historian Administrator:

    a. Access Historian Administrator *(on page 823)*.

    b. On the **Main** page, in the **Collectors** section, double-click the collector that you want to configure. Alternatively, you can select **Collectors**, and then select the collector from the **Collectors** pane.

The **General** section of the collector appears, displaying the properties of the collector.

c. Select **Configuration**, and then enter values as described in the following table.

| Field | Description |
|---|---|
| Calculation Timeout (sec) | The maximum time a calculation must be performed before being terminated. The default value is 10 seconds. If the calculation takes longer, it is canceled, and a bad data quality sample is stored in the destination tag with a subquality, calculation error. |
| Max Recovery Time (hr) | The maximum time, in hours till now, that the collector will attempt to restore data. This is applicable only to event-based tags. The default value is 4 hours.<br><br>If you want to disable automatic calculation of the tag, set the value of this field to 0. |

d. Pause and resume the data collection. Or restart the collector.

The collector instance is configured.

# Recalculate Tag Values

## Recalculate Tag Values Using Configuration Hub

When the connection between the collector and the Historian server is lost, the collector buffers data. When the connection is lost, the buffered data is sent to the Historian server. When the buffered data arrives, the timestamps are earlier than the most recent calculation timestamp. Since the timestamp is earlier, polled calculations are not performed again with the new data (unlike the unsolicited calculations). Hence, it is possible that calculations performed for tags during and after the connection loss might not be entirely accurate. Therefore, after the Historian server restores a lost connection, you may want to manually recalculate tag values. The recalculated data will use the most accurate values in calculations.

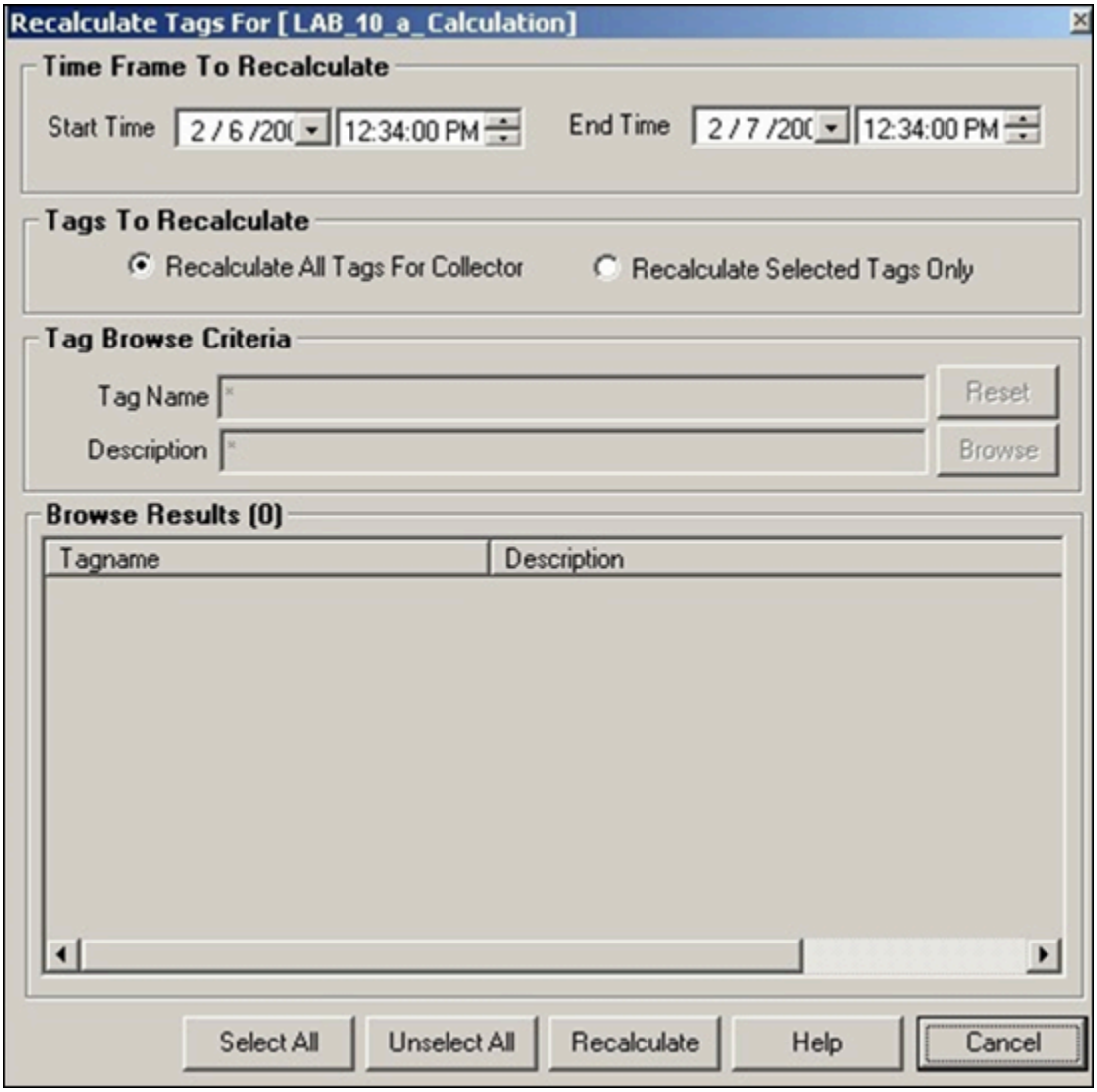
**Note:**
If you want to disable the automatic recalculation, set the **Max Recovery Time** to 0 in the **Configuration** section of the **Collector Maintenance** page of the collector.

## Recalculate Tag Values Manually

When the connection between the collector and the Historian server is lost, the collector buffers data. When the connection is lost, the buffered data is sent to the Historian server. When the buffered data arrives, the timestamps are earlier than the most recent calculation timestamp. Since the timestamp is earlier, polled calculations are not performed again with the new data (unlike the unsolicited calculations). Hence, it is possible that calculations performed for tags during and after the connection loss might not be entirely accurate.Therefore, after the Historian server restores a lost connection, you may want to manually recalculate tag values. The recalculated data will use the most accurate values in calculations.

**Note:**
If you want to disable the automatic recalculation, set the **Max Recovery Time** to 0 in the **Configuration** section of the **Collector Maintenance** page of the collector.

1. Access Historian Administrator *(on page 823)*.
2. On the **Main** page, in the **Collectors** section, double-click the collector whose tag values you want to recalculate. Alternatively, you can select **Collectors**, and then select the collector instance from the **Collectors** section.
   The **General** section of the collector appears, displaying the properties of the collector.
3. Select **Recalculate**.
   The **Recalculate Tags For <collector name>** window appears.




4. Provide values as described in the following table.

| Field | Description |
|---|---|
| **Start Time** | Enter the start date and time. Tag values calculated after this date and time will be recalculated. |
| **End Time** | Enter the end date and time. Tag values calculated until this date and time will be recalculated. |
| **Tags to Recalculate** | Specify whether you want to recalculate values for all the tags added in the collector or just the |

| Field | Description |
|---|---|
| | selected tags. If you select **Recalculate Selected Tags Only**, the **Tag Name** and **Description** fields are enabled. |
| **Tag Name** and **Description** | Enter the search criteria to filter out the tags whose values you want to recalculate. These fields are enabled only if you select **Recalculate Selected Tags Only**. After you select **Browse**, the search results appear in the **Browse Results** section. |

5. Select **Recalculate**.
   A message appears, asking you to confirm that you want to recalculate the data.
6. Select **OK**.
   The tags values are recalculated.

# Using the Calculation Collector

## Write Data to an Arbitrary Tag

If the tags that you want to specify are also used as trigger tags or source tags of a calculation, ensure that the Calculation collector does not read the tag data before you add the tag. To do so:

1. Access the ShouldPreReadData registry key under `HKEY_LOCAL_MACHINE\Software\Intellution, Inc.\iHistorian\Services\CalculationCollector\`.
2. Create a DWORD named `ShouldPreReadData`, and set its value to zero.
3. Restart the collector.

You can write data to an arbitrary tag in the Historian archive through the `AddData` function. This function is used in a calculation formula to write values, errors, time stamps and qualities of one or more tags to the Historian archive.

Use the following syntax to write data to an arbitrary tag:

```
errs = AddData(TagNames, Values, Timestamps, Qualities)
```

The following table provides information on the parameters.

| Parameter | Description |
|---|---|
| TagNames | Identifies the names of the tags. A value is required, and must exist in the archive to which you want to send the tag data. You can provide a single tag name or an array of tag names, enclosed in double quotation marks. |
| Values | Identifies the values of the tags. A value is required, and must be a single value or an array of values, depending on whether the tag name is a single name or an array. Values must be enclosed in double quotation marks. You can enter only a single value for each tag name. |
| Timestamp | Identifies the timestamp of the tag data. Enter an absolute time value, enclosed in double quotation marks ("24/12/20242:32:15 PM"). If you do not want to enter a value, enter NULL, and the current time will be considered. Do not enter a future timestamp. |
| Qualities | Identifies the quality of the tag data. Enter an integer from 0 to 100, with 0 indicating bad quality and 100 indicating good quality. |

**Tip:**
You can refer to Examples of Using the AddData Function *(on page 2107)*. For information on the status codes and their descriptions, refer to Status Codes of the AddData Function *(on page 2109)*.

## Examples of Using the AddData Function

This topic provides examples of using the AddData function.

### Writing a Single Tag Value

The following example is used to write a single value, the current time, and a good quality value to a single tag.

```
errs = AddData("Bucket Brigade.UInt4", 9, "Now", 100)
```

### Writing an Array of Tags

The following example is used to write an array of tags and their values.

```
Dim Tags(3)

Tags(0) = "Bucket Brigade.Boolean"
```

```
Tags(1) = "Bucket Brigade.Int4"

Tags(2) = "Bucket Brigade.Real8"

Tags(3) = "Bucket Brigade.String"


Dim Values(3)

Values(0) = True

Values(1) = 5

Values(2) = 172.3

Values(3) = "Hello, World"


errs = AddData(Tags, Values, Null, Null)
```

## Writing Timestamps and Qualities

The following example is used to write an array of tags and their values in addition to timestamp and quality values.

```
Dim Tags(3)

Tags(0) = "Bucket Brigade.Boolean"

Tags(1) = "Bucket Brigade.Int4"

Tags(2) = "Bucket Brigade.Real8"

Tags(3) = "Bucket Brigade.String"


Dim Values(3)

Values(0) = True

Values(1) = 5

Values(2) = 172.3

Values(3) = "Hello, World"


Dim Timestamps(3)

Timestamps(0) = "Today"

Timestamps(1) = "Now"

Timestamps(2) = "Yesterday"

Timestamps(3) = "24/12/2004 2:32:15 PM"


Dim Qualities(3)

Qualities(0) = 100

Qualities(1) = 0

Qualities(2) = 100
```

```
Qualities(3) = 0


Dim errs


errs = AddData(Tags, Values, Timestamps, Qualities)
```

## Status Codes of the AddData Function

This topic describes the status codes that appear when you use the AddData function. These status codes are stored in the `Errs` output variable as an array of status codes, one for each tag. The following table describes the status codes.

| Status Code | String | Description |
|---|---|---|
| 0 | `ihSTATUS_OK` | The values have been successfully written to the tags. |
| -1 | `ihSTATUS_FAILED` | The operation has failed. |
| -2 | `ihSTATUS_API_ TIMEOUT` | The operation has timed out while connecting to Historian. |
| -3 | `ihSTATUS_NOT_ CONNECTED` | The Calculation collector cannot connect to Historian. |
| -4 | `ihSTATUS_INTERFACE_ NOT_FOUND` | You cannot connect to the Calculation collector. |
| -5 | `ihSTATUS_NOT_ SUPPORTED` | The write operation is not supported. |
| -6 | `ihSTATUS_DUPLICATE_ DATA` | The write operation has created duplicate data in the archive. |
| -7 | `ihSTATUS_NOT_VALID_ USER` | The username used to connect to the Historian archive is not valid. |
| -8 | `ihSTATUS_ACCESS_ DENIED` | The username used to connect to Historian does not have write access to the archive. |
| -9 | `ihSTATUS_WRITE_IN_ FUTURE` | The timestamp that you have entered is set to a time in future. |

| Status Code | String | Description |
|---|---|---|
| -10 | ihSTATUS_WRITE_ ARCH_OFFLINE | The archive is currently offline. |
| -11 | ihSTATUS_ARCH_ READONLY | The archive is set to read-only. |
| -12 | ihSTATUS_WRITE_ OUTSIDE_ACTIVE | An attempt has been made to write data to a time before the archive has been created. |
| -13 | ihSTATUS_WRITE_NO_ ARCH_AVAIL | No archive is available for writing. |
| -14 | ihSTATUS_INVALID_ TAGNAME | The tag names that you have entered do not exist in the archive. |
| -15 | ihSTATUS_LIC_TOO_MANY_TAGS | You have attempted to add more tags than the current license allows. |
| -16 | ihSTATUS_LIC_TOO_ MANY_USERS | There are currently too many users connected to the archive. |
| -17 | ihSTATUS_LIC_ INVALID_LIC_DLL | The Historian license is expired or invalid. |
| -18 | ihSTATUS_NO_VALUE | You have not entered a tag value. |
| -19 | ihSTATUS_DUPLICATE_ INTERFACE | Two collectors with the same name exist. |
| -20 | ihSTATUS_NOT_ LICENSED | The Historian license is not activated. |
| -21 | ihSTATUS_CALC_CIRC_ REFERENCE | A circular reference has been entered in the calculation formula. |
| -22 | ihSTATUS_BACKUP_ EXCEEDED_SPACE | The archive has reached the Minimum Hard Drive Space setting, and no new archives are being created. |
| -23 | ihSTATUS_INVALID_ SERVER_VERSION | The archive is not compatible with the Calculation collector. |

| Status Code | String | Description |
|---|---|---|
| -24 | `ihSTATUS_DATA_ RETRIEVAL_-COUNT_ EXCEEDED` | There are too many data points to retrieve. |

# Creating Triggers

## Types of Triggers

You can create the following types of triggers for a calculation:

- **Polled or scheduled:** Used to trigger a calculation based on a scheduled time interval. For example, you can calculate the average value of tag data collected every hour.

  The polled type trigger functions the same as the other collectors. Although Historian internally optimizes calculation execution times, the data for polled tags is timestamped on the data collection interval. For example, if the calculation engine is unable to process the polled triggers as scheduled, the calculations will be executed later, but with data interpolated back to the scheduled time. If there are too many triggers to be processed, some triggers will be dropped and no samples are logged for that calculation time.

  For information on creating a polled trigger, refer to Create a Polled Trigger *(on page 2112)*.

- **Unsolicited or event-based:** Used to trigger a calculation based on an event. For example, you can calculate the average value of tag data when the data exceeds a certain value.

  When you set an event-based trigger, you must also set up a dependency list of one or more tags. Event-based triggers will keep calculations as up to date as possible. They are also useful when you want to do on-demand calculations. You can use a trigger tag that is written to by an external program or operation.

  If you want to perform raw sample replication you would use an event-based trigger. To retrieve data from a tag, use the formula:

  ```
  Result=CurrentValue("Tag1")+CurrentValue("Tag2")
  ```

  If you are using recovery mode, all referenced tags in an unsolicited calculation must be listed as trigger tags because recovery will be performed only for the configured trigger tags.

  Event-based triggers have a dependency list of trigger tags. The trigger fires whenever there is a data change for the trigger tag (for example, changes in the quality and value of a trigger tag). The

value of a trigger tag can change when the tag exceeds the collector compression (if you enabled collector compression).

The calculation is processed each time any tag in the dependency list changes. If you have multiple tags in the list and they change even one millisecond apart, then you will have multiple events, and the calculation formula will be processed for each.

However, the following actions do not trigger a calculation:

- Deletion of a tag that is in the dependency list.
- Re-addition of a tag in the dependency list.

The calculation is triggered at the same time as the timestamp of the sample in the trigger tag. The values of all other tags in the formula are interpolated forward to this time so that the timestamps of all input tags are the same. Even if these are sequential events, they have the same timestamp. The calculation time becomes the timestamp for the sample stored in the destination tag.

Event-based triggers have a collection interval. The Calculation collector notifies the archiver not to send notification of changes to trigger tags any faster than the collection interval setting.

For information on creating an unsolicited trigger, refer to Create an Unsolicited Trigger *(on page 2114)*.

## Create a Polled Trigger

1. To create a poller trigger using Configuration Hub:
    a. Access Configuration Hub *(on page 336)*.
    b. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**. A list of tags appears.
    c. From the list of tags, select the Calculation tag (say, tag 2).
    d. In the **Collection Options** section, select **Polled** from **Collection Type**, and specify.
    e. Set the **Collection Interval Value** and **Collection Offset** values. For example, if you want to set a trigger every day, set these values to 24 hours and 8 hours, respectively.
    f. In the upper left corner of the page select **Save**.
    The triggers are created.
2. To create a polled trigger using Historian Administrator:

a. Access Historian Administrator *(on page 823)*.
b. Select **Tags**.
c. In the **Tags** section, select the tag to which you want to apply the trigger.
d. In the **Collection** section, select **Polled from the Collection Type**.
e. Set the **Collection Interval** and **Collection Offset** values. For example, if you want to set a trigger every day, set these values to 24 hours and 8 hours, respectively. Depending on the trigger that you want to set, enter the appropriate VBScript code in the **Calculation** pane. For examples, refer to Examples of Scheduling Polled Triggers *(on page 2113)*.
f. Select **Update**.

## Examples of Scheduling Polled Triggers

You can schedule calculations using polled triggers, as shown in the following examples.

### Scheduling a Trigger every Monday

Since Monday is the second day of a week, enter the following VBScript code in the **Calculation** pane:

```
Dim curDate curDate=CurrentTime

IF (Weekday(curDate))=2 THEN

Result=50 <Place your calculation here>

END IF
```

Notice that the `CurrentTime` built-in function is used in this example instead of `Now`.

### Scheduling a Trigger on the First Day of Every Month

Enter the following VBScript code in the **Calculation** pane:

```
Dim curDate curDate=CurrentTime

IF (day(curDate))=1 THEN

Result=50 <Place your calculation here>

END IF
```

Notice that the `CurrentTime` built-in function is used in this example instead of `Now`.

### Scheduling a Trigger on the Last Day of Every Month

Enter the following VBScript code in the **Calculation** pane:

```
Dim curDate curDate=CurrentTime

IF (day(curDate))=1 THEN
```

```
Result=50 <Place your calculation here>

END IF
```

Notice that the `CurrentTime` built-in function is used in this example instead of `Now`.

### Example 5: Creating a Controlled Sequence of Polled Tags Using a Collection Offset

A controlled sequence is a calculation that is based on the result of another calculation. The following is an example of how to configure a controlled sequence of polled tags by using the collection offset. The collection offset is greater than 0 in this example.

A tag named `SumOfData` has a Collection Interval of 60 seconds and the Collection Offset of 0 milliseconds.

`SumofData` performs the first calculation:

```
Result = CurrentValue("DataTag1") + CurrentValue("DataTag2")
```

Another tag, named `CorrectedUnits`, uses a Collection Interval of 60 seconds, but a Collection Offset of 1000 milliseconds.

`CorrectedUnits` fires and performs another calculation based on the output of the first calculation:

```
Result = CurrentValue("SumOfData") *.0001
```

## Create an Unsolicited Trigger

This topic describes how to create an unsolicited trigger for a calculation, and how to set up a dependency list:

1. Create the tag that you want to use to create a trigger (say, tag1).
2. Create another tag in the Calculation collector whose value you want to update based on the trigger (say, tag2).
3. Ensure that the Calculation collector is running.

1. To create an unsolicited trigger using Configuration Hub:

    a. Access Configuration Hub *(on page 336)*.

    b. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**. A list of tags appears.

    c. From the list of tags, select the Calculation tag (say, tag 2).

d. In the **Collection Options** section, select **Unsolicited** from the **Collection Type**, and specify an interval.

e. Select Calculation Triggers and select .
The Calculation Triggers window appears.

f. Search for tags or select tags from the list as trigger tags.

g. You can remove the selected tags by selecting X from the Selected Tags list.

h. Select **Insert Trigger**.
The trigger tag count will be displayed.

i. In the upper left corner of the page select **Save**.
The triggers are created.

2. To create an unsolicited trigger using Historian Administrator:

a. Access Historian Administrator *(on page 823)*.

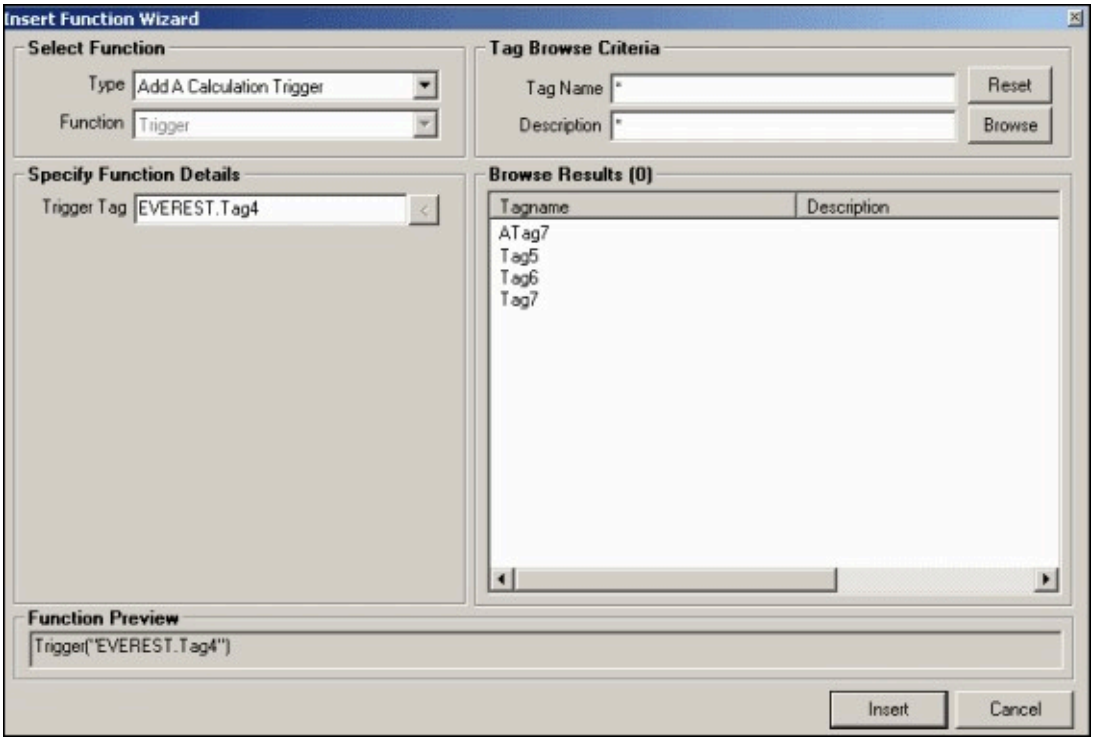
b. Select **Tags**, and then select the tag to which you want to apply the trigger.

c. In the **Collection** section, select **Unsolicited from the Collection Type**, and specify an interval.

d. Select **Calculation**.

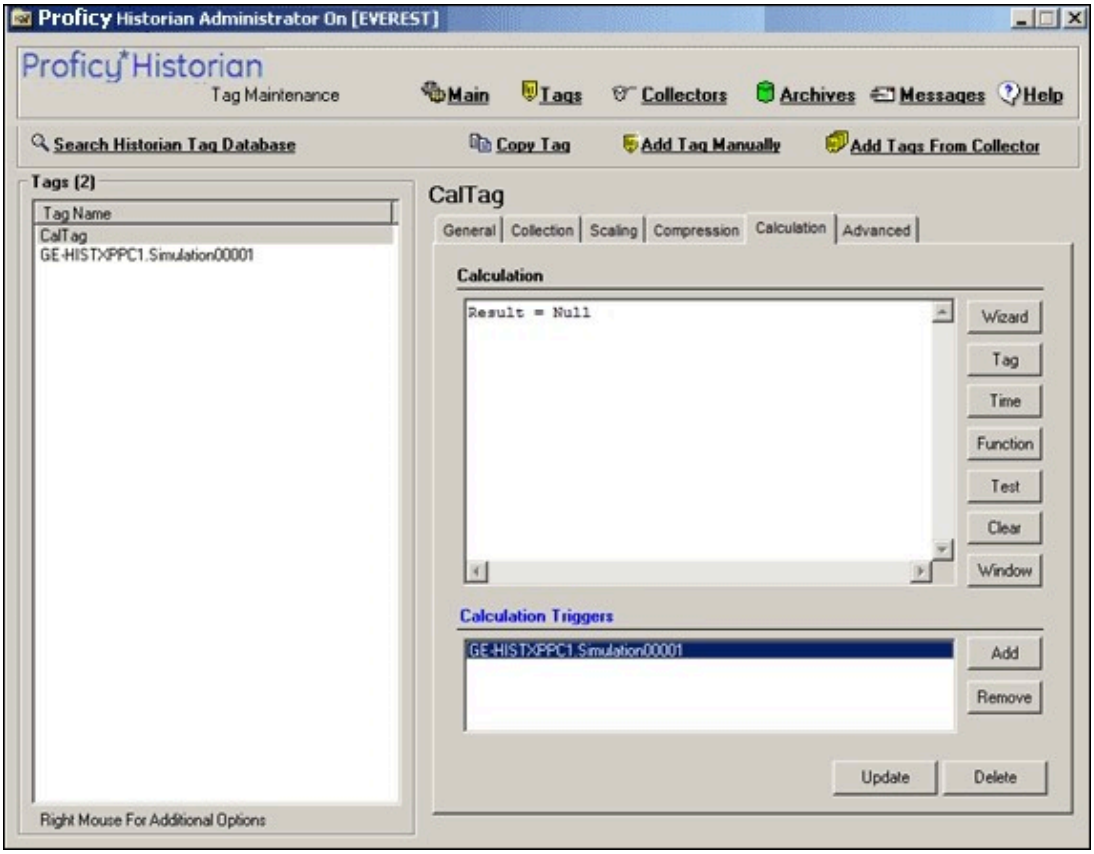
e. Select **Add** to add a trigger.
The **Insert Function** window appears.

f. In the **Trigger Tag** field, enter the name of the trigger tag that you want to use in the calculation.
The wizard automatically populates the **Trigger Tag** field and updates the **Function Preview** field, as shown in the following figure:




g. Select **Insert**.

   The trigger is added to the **Calculation Triggers** section.




In addition, the trigger list appears when using the **Insert Function** window to build a calculation formula for an event based tag.

h. Select **Update**.
The unsolicited trigger is created.

## Examples of Scheduling Unsolicited Triggers

### Example1: Using One Trigger Tag in a Formula

The following is an example of an event-based calculation with one trigger tag:

```
Result=CurrentValue("Tag2") + CurrentValue("Tag3")
```

You can configure `Tag1`, which is not in the formula, to be the calculation trigger for this example. `Tag2` and `Tag3` are not trigger tags. Trigger tags do not have to reside in the formula. There is no relation between formula tags and trigger tags. However, if you are planning to use recovery mode, you want all formula tags to be triggers.

### Example 2: Using Multiple Trigger Tags in a Formula

The following is an example of an event-based calculation with multiple trigger tags:

```
Result=CurrentValue("Tag1") + CurrentValue("Tag2")
```

Configure `Tag1` and `Tag2` to be the calculation triggers for this example.

### Example 3: Creating a Controlled Sequence of Unsolicited Tags Using Trigger Tags

A controlled sequence is a calculation that is based on the result of another calculation. The following is an example of how to create a controlled sequence of unsolicited tags using trigger tags. In this example, you create a calculation tag that is based on the result of another calculation tag.

For `CalcTag1`, the calculation is as follows:

```
Result = CurrentValue("TagA") + CurrentValue("TagB")
```

`TagA` and `TagB` are the calculation triggers for `CalcTag1`:

For `CalcTag2`, the calculation is as follows:

```
Result = CurrentValue("CalcTag1") * CurrentValue("TagC")
```

`CalcTag1` is the calculation trigger for `CalcTag2`.

# Calculation Formulas

## About Calculation Formulas

To perform a calculation using the Calculation collector, you must define the calculation formula. You can do so in one of the following ways:

- Using the Insert Function wizard *(on page 2123)*, which helps you use any of the built-in functions *(on page 2127)* orcreate your own function *(on page 2125)*.
- Entering the syntax of the formula directly in the form of a VBScript code *(on page 2121)*.

Before you create calculation formulas, refer to the general guidelines *(on page 2119)*.

There are two predefined global values called Result and Quality. These global values control the value and quality of the output sample. If the Result is not set in the formula, then no sample is stored.

## General Guidelines for Defining a Calculation Formula

This section provides guidelines that you must follow when defining a calculation formula.

**Identify Time Intensive Calculations**

Use the `Calculation Execution Time` property of each tag to identify time-intensive queries. In Historian Administrator, look for the **Execution Time** on the **Calculation** section for an estimate of how long, on average, it takes for the calculation per tag (starting from the time the collector was started).

You can also include that column when you export tags to Excel using the Excel Add-In feature. For information, refer to Exporting Tags *(on page 2746)*.

You can also include that column (AverageCollectionTime) when you query the ihTags table using the Historian OLE DB Provider. Sorting by this column will let you find them fast.

**Troubleshoot Issues with Large Configurations**

If the timestamps of your raw samples appear slightly old, do not assume that the collector has stopped working. It is possible that the collector is just running behind.

For instance, if you have a report rate of 15,000, but the newest raw sample that you see is 20-30 minutes old, wait for 1-2 minutes, and review the newest raw sample again. If the collector stopped, the newest raw sample will be unchanged. If it did change, then the engine is still running, but is lagging behind. If that happens, check if the collector overrun count is increasing. If yes, the collector is dropping samples, and you must decrease the load.

**Error Handling in VBScript**

Start each script with the On Error Resume Next statement so that errors are trapped. If you use this statement, the script runs even if a run-time error occurs. You can then implement error handling in your VBScript.

It is a good practice to include statements in your VBScript that catch errors when you run the script. If there is an unhandled error, a value of 0 with a bad data quality is stored. When you catch an error in the VBScript, consider including a statement in your calculation that sets the Quality=0 when the error occurs. (The 0 value means that the quality is bad.) If you do not specifically include this setting in your script, Historian stores a good data quality point (Quality=100), even if an error has occurred in your formula. If Quality=100 is not appropriate for your application, consider setting the quality to 0.

You cannot use the On Error `GoTo` Label statement for error handling, as it is not supported in VBScript. As a workaround, you can write code in the full Visual Basic language and then place it in a `.DLL` so that you

can call it from within your VBScript using the `CreateObject` function. For examples of calculations that use the `CreateObject` function, refer to Examples of Calculation Formulas *(on page 2138)*.

**Unsupported VBScript Functions**

You can use any VBScript syntax to build statements in a calculation formula with the exception of the following functions:

- `MsgBox`
- `InputBox`

**Milliseconds not Supported in VBScript**

The `CDate()` function does not support the conversion of a time string with milliseconds in it. Whenever you use the `CDate()` function, a literal time string, or a time string with a shortcut, do not specify milliseconds in the time criteria. Milliseconds are not supported in VBScript.

You cannot use milliseconds in times passed into built-in functions such as the `PreviousTime` and `NextValue` functions. For example, you cannot loop through raw samples with millisecond precision.

**Notes on VBScript Time Functions**

Using the VBScript time functions such as Now, Date, or Time can lead to unexpected results, especially in recalculation or recovery scenarios. To avoid these issues, use the `CurrentTime` built-in function provided by Historian, instead of Now, Date, or Time. For example, the VBScript Now is always the clock time of the computer and is likely not useful when recalculating or recovering data for times in the past. However, the "Now" time shortcut is equivalent to `CurrentTime` and can be used as input to the other built in functions.

**Using Quotation Marks in VBScript**

If you want to use quotation marks in a tag name, you must insert a double quotes for each quotation mark that you want to use, as required for proper VBScript syntax. For example, if you want to get the current value of a tag named TagCost"s, you must enter:

```
Result = CurrentValue("TagCost""s")
```

In this example, note the double quotation marks that appear before the letter s in the TagCost"s name in the formula.

**Avoiding Circular References in VBScript**

Do not use circular references in calculation formulas. For instance, if the tag name is `Calc1`, a formula with a circular reference would be `Result=CurrentValue("Calc1")`. Whether the tag is polled or unsolicited, you get a bad value back using the circular reference.

**Uninterrupted Object Method Calls**

Object method calls are not interrupted. It is possible to exceed the Calculation Timeout setting if you have a method call that takes a long time to execute. The Calculation Timeout error still occurs, but only after the method completes.

**Help for VBScript**

You can get detailed Help for VBScript by referencing the Microsoft documentation on the MSDN web site. A *VBScript User's Guide and Language Reference* is available here: http://msdn.microsoft.com/en-us/library/t0aew7h6.aspx

**Avoiding Deleted Tags**

You can reference a deleted tag in a calculation formula, without an error appearing. For instance, you could enter a formula such as `Result=CurrentValue("DeletedTag")`, where `DeletedTag` is the name of the deleted tag. You can do this because when you delete a tag, Historian removes deleted tags from the Tag Database (so you cannot browse for it), but it retains the data for that tag in the archive.

However, it is recommended that you do not reference deleted tag names in your calculation formulas, because if the archive files are removed with the data for the deleted tag, the calculation will not work properly.

## Create a Calculation Formula Using a VBScript Code

This topic describes how to create a calculation formula by entering a VBScript code. You can also create a calculation formula using the Insert Function wizard *(on page 2123)*.

**Important:**
If a tag contains bad data quality, you cannot store its value through a calculation formula. For example, if your VBScript includes: `Result = 7 Quality = 0`, Historian does not store the `7`, it stores `0`.

**Tip:**
For examples, refer to Examples of Scheduling Polled Triggers *(on page 2113)* and Examples of Scheduling Unsolicited Triggers *(on page 2117)*.

Create the tag that you want to use to store the calculation results. You can create the tag manually using Configuration Hub *(on page 473)*, Historian Administrator *(on page 2057)* or the Web Admin console *(on page 2059)*. Or, you can copy a tag *(on page 2060)*.

1. To create a calculation formula using Configuration Hub:
    a. Access Configuration Hub *(on page 336)*.
    b. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
    A list of tags appears.
    c. Select a Calculation tag, right-click or select more options, and then select **Calculation**.
    The calculation editor appears.
    d. Enter the VB script in the editor.
    e. To test the function, select .
    A message appears, stating whether the syntax is correct.
    f. In the upper-left corner of the page, select **Save**.
    The calculation formula is created.
2. To create a calculation formula using Historian Administrator:
    a. Access Historian Administrator *(on page 823)*.
    b. Select **Tags**, select the tag for which you want to create a calculation formula, and then select **Calculation**.
    c. In the **Calculation Triggers** section, select **Add**.
    The **Insert Function Wizard** window appears.
    d. Under **Select Function**, in the **Type** field, select **Add A Calculation Trigger**.
    e. Under **Tag Browse Criteria**, enter the search criteria to find the tag.
    The search results appear in the **Browse Results** section.
    f. Select the tag that you want to add, and then select **Insert**.
    g. In the **Calculation** field, enter the calculation formula using the VBScript syntax.
    h. To verify that the syntax is correct, select **Test**.
    A message appears, stating whether the syntax is correct.

## Create a Calculation Formula Using the Pre-built Functions

This topic describes how to create a calculation formula using the pre-built functions. You can also create a calculation formula using a VBScript code *(on page 2121)*.

For information on a list of the available types and associated functions, refer to Types of Functions Supported *(on page 2135)*. For information on each pre-defined function, refer to Built-In Functions *(on page 2127)*. In addition to the built-in functions, you can create your own customized functions *(on page 2136)*.

1. Create the tag that you want to use to store the calculation results. You can create the tag manually using Configuration Hub *(on page 473)*, Historian Administrator *(on page 2057)* or the Web Admin console *(on page 2059)*. Or, you can copy a tag *(on page 2060)*.
2. Access the advanced options of the collector, and then disable the **On-line Tag Configuration Changes** option using Historian Administrator *(on page 2057)*. In Configuration Hub *(on page 473)* this property is available in Collector Options section. If you do so, each time you update a calculation formula, the collector does not reload tags.

1. To create a calculation formula using Configuration Hub:

    a. Access Configuration Hub *(on page 336)*.

    b. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**. A list of tags appears.

    c. Select a calculation tag, right-click or select more options, and then select **Calculation**. The calculation editor appears.

    d. In the **CALCULATION** section, remove `Null` (and retain the `Result =`).

    e. In the Calculation editor, place the cursor in the location where you want to insert the pre-built function.

    f. In the **DETAILS** section, under **Pre-Built Functions**, in the **Function Type** field, select a function type.
    Depending on the function type you have selected, a list of functions appears in the **Function** field.

    g. In the **Function** field, select a function.
    Based on the selected function, the related fields will be displayed.

    h. Enter values in the other fields that appear after selecting a function.

i. In the **Input Tag** field, select , and then select the tag where applicable.
The function preview appears below the calculation editor.

j. Select **Insert Function**.
The function is inserted at the cursor position.

k. To test the function, select .
A message appears, stating whether the syntax is correct.

l. In the upper-left corner of the page, select **Save**.
The calculation formula is created.

2. If you want to create a calculation formula using Historian Administrator:

a. Access Historian Administrator *(on page 823)*.

b. Select **Tags**, select the tag for which you want to create a calculation formula, and then select **Calculation**.

c. In the **Calculation** section, remove `Null` (retain `Result =`).

> **Tip:**
> Avoid selecting other tags until you save your changes or you will lose your code changes.

d. Select **Wizard**.
The **Insert Function Wizard** window appears.

e. Under **Select Function**, select values in the available fields, and then select **Insert**.

f. If you want to perform an unsolicited (also called event-based) calculation, in the **Type** field, select **Add A Calculation Trigger**. Search and select the tag that you want to add, and then select **Insert**.
The calculation formula is created.

g. To verify that the syntax is correct, select **Test**.
A message appears, stating whether the syntax is correct.

## Create a User-Defined Function

This topic describes how to create your own function to use in a calculation formula. For more information, refer to User-defined Functions *(on page 2136)*. You can also use any of the built-in functions *(on page 2127)*.

Create the tag that you want to use to store the calculation results. You can create the tag manually using Configuration Hub *(on page 473)*, Historian Administrator *(on page 2057)* or the Web Admin console *(on page 2059)*. Or, you can copy a tag *(on page 2060)*.

1. To create a user-defined function using Configuration Hub:
    a. Access Configuration Hub *(on page 336)*.
    b. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
    A list of tags appears.
    c. Select a calculation tag, right-click or select more options, and then select **Calculation**.
    The calculation editor appears.
    d. In the **CALCULATION** section, remove `Null` (retain `Result =`).
    e. In the calculation editor, place the cursor in the location where you want to insert the user-defined function.
    f. In the **DETAILS** section, under **User Defined Functions**, select .
    The **Add/Edit User Defined Functions (UDF)** window appears.
    g. Select **Add New**.
    A function is created with the following naming convention: UserFunction<*number*>. The same function is used in the function editor. You can change the function name by entering the new name in the function editor and selecting **Update UDF**.
    h. Enter the VB script for the function. Or, if you want to use a pre-built function, select the function type and function in the respective fields under **Pre-Built Functions**. And, enter values in the other fields that appear after selecting a function.
    The function appears in the function preview below the function editor.
    i. Select **Insert Preview**.
    The function is included in the function editor at the cursor position.
    j. To test the function syntax, select .

A message appears, stating whether the function syntax is correct.

k. Select **Update UDF**.
The user-defined function is created.

l. To use the function in the calculation formula, select **Insert UDF**.
The function is inserted in the calculation formula at the cursor position.

m. To test the calculation formula, select .
A message appears, stating whether the syntax is correct.

n. In the upper-left corner of the page, select **Save**.
The calculation formula is created.

2. To create a user-defined function using Historian Administrator:

a. In Historian Administrator, select **Tags**, select the tag for which you want to create a calculation formula, and then select **Calculation**.

b. In the **Calculation** section, remove `Null` (retain `Result =`).

> **Tip:**
> Avoid selecting other tags until you save your changes or you will lose your code changes.

c. Select **Functions**.
The **User Defined Functions** window appears.

d. Select **New**.
The **Edit Function** window appears.

e. Define the function.
You can build formulas using the wizard, or create it manually by entering functions in the **Edit Function** box. For information, refer to User-defined Functions *(on page 2136)*.

f. Select **Syntax** to check for errors.

g. Select **Update**.
Your function appears in the list, and is available for use in other calculations as well.

h. To use the function, select **Insert Function**.
The function is inserted in your calculation formula.

## Built-in Functions

This topic describes the built-in functions that you can use to create a calculation formula. You can also create your own calculation function *(on page 2136)*.

**Note:**

- In this table, `Time` refers to the actual time; this time can include absolute and relative time shortcuts. Refer to the Date/Time Shortcuts *(on page 2138)* section for more information.
- You cannot control the timestamp of the stored sample. It is determined by the triggering tag or polling schedule.
- You cannot use microseconds for any of the built-in calculation functions.

  For all the functions that retrieve previous values, it is similar to performing a `RawByNumber` query with a count of 1 and direction of backward. A less-than operation (not less-than-or-equal-to) is used on the timestamp to get the sample. Similarly, for all the functions that retrieve next values, it is similar to performing a `RawByNumber` query with a count of 1 and direction of forward. A greater-than operation (not greater-than-or-equal-to) is used on the timestamp to get the sample.

| Function Name | Description |
|---|---|
| `CurrentValue(<tag name>)` | The value of the tag, interpolated to the calculation execution time. The `CurrentValue` function returns 0 if the quality is 0 (bad quality). This occurs if you initialized it to 0, or if a previous call failed. |
| `CurrentQuality(<tag name>)` | The current quality of the tag (0 for bad quality and 100 for good quality). |
| `CurrentTime` | The calculation execution time, which becomes the timestamp of the stored value.<br><br>For real time processing of polled tags, the calculation execution time is the time when the calculation is triggered. For unsolicited tags, the calculation execution time is the timestamp delivered with the subscription. |

| Function Name | Description |
|---|---|
| | **Note:**<br>When a calculation is performed, the timestamp of the result is the time that the calculation has begun, not the time that it completed.<br><br>For recovery of polled or unsolicited tags, the calculation execution time is the time when the calculation would have been performed if the collector were running. |
| `Previous-Value(<tag name>, Time)` | The tag value of the raw sample prior to the current time. |
| `Previous-Quality(<tag name>, Time)` | The quality of the tag (0 for bad quality and 100 for good quality) prior to the current time. |
| `Previous-GoodVal-ue(<tag name>, Time)` | The latest good value of the raw sample prior to the current time. |
| `Previous-GoodQual-ity(<tag name>, Time)` | The good quality of the raw sample prior to the current time. |
| `Previous-Time(<tag name>, Time)` | The timestamp of the raw sample prior to the current time. |
| `Previ-ousGood-Time(<tag name>, Time)` | The timestamp of the latest good quality of the raw sample prior to the current time. |
| `NextVal-ue(<tag name>, Time)` | The value of the raw sample after the current timestamp. |

| Function Name | Description |
|---|---|
| `NextQuality(<tag name>, Time)` | The quality of the tag (0 for bad quality and 100 for good quality) after the current time. |
| `NextTime(<tag name>, Time)` | The timestamp of the raw sample after the current timestamp. |
| `NextGoodValue(<tag name>, Time)` | The value of the good raw sample after the current time. |
| `NextGoodQuality(<tag name>, Time)` | The good quality of the raw sample after the current time. |
| `NextGoodTime(<tag name>, Time)` | The timestamp of the good raw sample after the current time. |
| `InterpolatedValue(<tag name>, Time)` | The tag value, interpolated to the time that you enter. |
| `Calculation` | Unfiltered calculated data query that returns a single value, similar to the Excel Add-In feature. For a list of the calculation mode, refer to Calculation Modes *(on page 1073)*. |
| `AdvancedCalculation` | Unfiltered calculated data query that returns a single value, similar to the Excel Add-In feature. For a list of the calculation mode, refer to Calculation Modes *(on page 1073)*. |
| `AdvancedFilteredCalculation` | Advanced Filtered calculated data query that returns a single value, similar to the Excel Add-In feature. |
| `FilteredCalculation` | Filtered calculated data query that returns a single value, similar to the Excel Add-In feature. |
| `LogMessage(string_message)` | Allows you to write messages to the Calculation collector or the Server-to-Server collector log file for debugging purposes. The collector log files are located in the `Historian\LogFiles` folder. |

| Function Name | Description |
|---|---|
| | **Note:** The `LogMessage` function is the only function that does not appear in the wizard. |
| `GetMultiFieldValue(Variable, <field name>)` | Returns the value of the field that you have specified. The variable contains the current value of all the fields of a multi-field tag. Before using this function, you must read the tag into a variable, using the `CurrentValue()` function. You can then use the `GetMultiFieldValue` function to access the value of the field.<br><br>The value of the field that you enter must be the same as the name of the field in the user defined type. If the field name is not found, a null value is returned. |
| `GetMultiFieldQuality(Variable, <field name>)` | Returns the quality (0 for bad quality and 100 for good quality) of the field that you have specified. The variable contains the current value of all the fields of a multi-field tag. Before using this function, you must read the tag into a variable, using the `CurrentValue()` function. You can then use the `GetMultiFieldValue` function to access the value of the field.<br><br>The value of the field that you enter must be the same as the name of the field in the user-defined type. If the field name is not found, a null value is returned.<br><br>If the user-defined type can store individual quality, you get the field quality. Otherwise, you get the sample quality. |
| `SetMultiFieldValue(Variable, <field name>, Value, Quality)` | Sets the value and the quality for the field that you have specified.<br><br>You can use this function to construct a multifield value containing values for each field, and then use the `result=` syntax to store the value in Historian. |

## Counting the Number of Bad Quality Samples

The following example shows how to loop through samples of a tag named C2 to count the number of bad quality samples.

```
Dim count, starttime, endtime, tagquality count=0

StartTime=CurrentTime EndTime=DateAdd("n",-1,StartTime) Do while StartTime>EndTime

TagQuality=PreviousQuality("C2",StartTime)
```

```
startTime=PreviousTime("C2",StartTime) IF TagQuality=0 THEN

count=count + 1

END IF loop Result=count
```

### Counting the Number of Collected Digital 1s For a Tag

The following example counts the number of collected digital 1s for a tag so that, for instance, you can determine how many times a pump is turned ON and OFF.

```
Dim count, starttime, endtime,tagquality,TagValue

count=0

StartTime=CurrentTime

EndTime=DateAdd("h",-1,StartTime)

On error resume next

Do while StartTime>=EndTime

TagValue=PreviousValue("FIX.DI.F_CV",StartTime)

TagQuality=PreviousQuality("FIX.DI.F_CV",StartTime)

startTime=PreviousTime("FIX.DI.F_CV",StartTime)

IF TagQuality=100 AND TagValue=1 then

count=count + 1

END IF

loop

Result=count
```

### Determining the Trigger When Using Multiple Trigger Tags

The following example shows how to determine which tag triggered the calculation, from a list of two possible trigger tags. The example compares the two trigger tags and determines which one has the newest raw sample. This method of getting the newest raw sample can also be used to determine if a remote collector is sending data or is disconnected from the server.

In this example, archive compression is disabled for both of these tags.

```
dim timetag1

dim timetag2

dim tag1

dim tag2


tag1 = "BRAHMS.AI1.F_CV"

tag2 = "BRAHMS.AI2.F_CV"
```

```
' Get the timestamp of the newest raw sample for tag1:
timetag1 = previousTime(tag1, CurrentTime)

' Get the timestamp of the newest raw sample for tag2:
timetag2 = previousTime(tag2, CurrentTime)

if timetag1 > timetag2 then
' If tag1 triggered me, then:
result = 1 else
' If tag2 triggered me, then:
result = 2
end if
```

## Using Array or Multifield Data in Calculation

You can create tags of arrays and multifield types and use the Calculation collector, Server-to-Server collector, Server-to-Server distributor with these tags.

### Arrays

To use the Array data as input to a calculation formula you can use the name of the array tag like "Array1" or the individual element of the array like "Array1[4]". For example, if you have an array tag "Array1" of floating point values and a calculation tag "FloatCalc1" of float data type, then you can use the array as input to calculate a float value.

```
result = currentvalue("Array1[4]")+5
```

You can use Calculation() function to read the array tag as shown in the following code.

```
Result = Calculation("Array1","Average","Now 1Minute","Now",Quality)
```

In this example, the calculation tag should be an array tag because the average of an array is an array, not a single value. Each element is averaged over the time range. Since an average of an integer or float array is a floating point value, the calculation tag must be a single or double float array.

If you want to find the minimum of array elements in a given time, then use vbscript code to compute and store the result in a Float tag as shown.

```
if CurrentValue("Array1[0]") < CurrentValue("Array1[1]") then
    Result = CurrentValue("Array1[0]")
```

```
else

    Result = CurrentValue("Array1[1]")

end if
```

#### Multifield

If you have a user-defined type "MySample" with fields "r;FloatVal" and "r;IntVal" you can create `Tag1` and use the value of one field in an Integer Calc Tag. The destination tag is not a multifield tag.

```
result = currentvalue("Tag1.IntVal")+5
```

### Storing Array or Multifield data in Calculation tags

#### Array

If your calculation tag is an array tag, then you can copy the entire array values into it. For example, you can copy the entire values from `Array1` into `Array2` using the given code.

```
result = CurrentValue("Array1")
```

You can take an array value collected from a field device and adjust the values before storing it in another array tag `Array2` using this code:

```
dim x

x=CurrentValue("Array1")

x(1) = x(1)+10

result = x
```

You can simply construct an array value inside your formula and store it in `Array2`, for example:

```
dim MyArray(2)' The 2 is the max index not the size

MyArray(0)=1

MyArray(1)=2

MyArray(2)=3 result = MyArray
```

#### Multifield

You can have the collector combine collected data into a multifield tag. Create a calculation `Tag1` using the user-defined Type "MySample," then use this formula to fill in the fields:

```
Dim InputValue, myval,x,y


' get the current value of another multifield tag

InputValue = CurrentValue("tag1")
```

```
' get the values of each of the fields
x = GetMultiFieldValue(InputValue, "IntVal")
y = GetMultiFieldValue(InputValue, "floatval")

' store the field values in this tag
SetMultiFieldValue myval,"IntVal",x,100
SetMultiFieldValue myval,"floatval",y,100
Result = myval
```

## Using Array or Multifield data to trigger calculation

### Array

You can use the array tag as a trigger tag for your float or array calculation tags. For example, you can use `Array1` as a trigger so that when it changes, the `"CalcArray1"` tag will be updated. You cannot use an individual array element such as `"Array1[3]"` as a trigger, you must use the entire array tag as the trigger tag.

### Multifield

You can use a multifield tag as a trigger tag by either using the tagname `"Tag1"` or tagname with the field name `"Tag1.FloatVal"`.

## Sending Array or Multifield data to a Remote Historian

### Array

You can use the Server to Server Collector or Server to Server Distributor to send array data to a destination Historian. If the destination Historian is version 6.0 or later, you can simply browse the tags and add them.

You cannot send an array to the older versions of archiver (Pre 6.0 versions) as these archivers will store the array tags as a blob data type in the destination and you will not be able to read them. However, you can send individual elements of an array to these archivers, for example, `result = currentvalue("Array1 [4]")`.

### Multifield

The destination needs to be Historian 6.0 or above to store a multifield tag but you can send individual fields to a pre Historian 6.0 archiver.

For multifield tags, you must create the User Defined Type manually at the destination

You can write an entire multified tag data sample in one write or you can create multiple tags in the destination, one for each field you want to copy. For example, if you have one tag `"Tag1"` with two fields `"FloatVal"` and `"IntVal"` on a source archiver, then you can create two tags (`"Tag1.FloatVal"` and `"Tag1.IntVal"`) on the destination.

**Note:**
If you change a field name or add or remove fields you must update your collection and your destination tags.

### Reading and writing a Multifield tag using MultiField functions

The following example shows how to read an entire multifield tag, using the `GetMultiFieldValue` function and to write the value to a field in another tag using the `SetMultiFieldValue` function.

```
Dim CurrMultifieldValue


' Read the value of a multi field tag into a variable

CurrMultifieldValue = CurrentValue("MyMultifieldTag")


' Read the field value of multifield tag into the temporary variable

F1 = GetMultiFieldValue(CurrMultifieldValue, "Temperature Field")


' Perform a calculation on the value

Celcius = (F1 32)/ 9* 5


' Set the calculated value to another field of the multifield tag

SetMultiFieldValue(CurrMultifieldValue, "Temperature Field Celcius", Celcius, 100)

result = CurrMultifieldValue
```

## Types of Functions Supported

The following table describes the types of actions supported. All the value functions return a single value.

| Type of Action | Available Functions for the Action |
|---|---|
| Insert a value | • Current value<br>• Previous value<br>• Next value<br>• Interpolated value |

| Type of Action | Available Functions for the Action |
|---|---|
| Insert a calculation | • Unfiltered calculation<br>• Filtered calculation |
| Insert a timestamp | • Time shortcut<br>• Previous value timestamp<br>• Next value timestamp<br>• Current time |
| Check data quality | • Current value quality<br>• Previous value quality<br>• Next value quality |
| Set data quality | • Set Quality Good<br>• Set Quality Bad |
| Add data value | Value |
| Insert a tag name | Tagname |
| Insert an alarm calculation | • Previous Alarm<br>• Next Alarm<br>• Get Alarm Property<br>• Set Alarm Property<br>• Add Event<br>• New Alarm<br>• Update Alarm<br>• Return to Normal |
| Insert a multifield operation | • GetMultiFieldValue<br>• GetMultiFieldQuality<br>• SetMultiFIeldValue |

## User-defined Functions

In addition to the built-in functions *(on page 2127)*, you can create custom calculation functions. After you create a custom calculation function, it is available for use with other calculations as well.

Functions are useful as shortcuts for large blocks of source code. By creating a function out of commonly used calculation formulas, you can save time and effort instead of typing a few lines of calculation formula every time you want to perform the same operation, it is compressed to a single line.

The syntax of a function is simple:

```
Function functionname (variable list)
   [calculation formulas]
End Function
```

The operations a function performs are contained within the Function / End Function statements. If you need to send data to the function a tag name, for example you simply create a variable in the function's parameters to receive the data. Multiple variables must be separated by commas. These variables exist only within the function.

The following is an example of a function. This function, named `checkValue()`, looks at a tag and assigns it an alarm if it is over a specified value.

### A Function to Assign an Alarm to a Tag Based on a Condition

The following function, named checkValue, assigns an alarm to a tag if the tag value reaches a specified value.

```
Function checkValue (tagname,sourcename,value)
  If CurrentValue(tagname) > value Then
  Set AlarmObj = new Alarm
  AlarmObj.SubConditionName = "HI"
  AlarmObj.Severity = 750
  AlarmObj.NewAlarm
  "alarmname", "Simulated", "tagname", "Now"
  checkValue = true
Else
  checkValue = false
  End If
End Function
```

If you want to use this function, enter the values for tag name, source name, and value, as shown in the following example:

```
alm_set = checkValue("DD098.FluidBalance","FluidBalance_ALM",5000)
```

In this example, if the value of the DD098.FluidBalance tag exceeds 5000, the function returns a true value, indicating that the alarm was set; the *alm_set* variable will be set to `true`. Otherwise, the *alm_set* variable will be set to `false`.

## Date/Time Shortcuts

The following table outlines the date/time shortcuts that you can use in calculation formulas.

**Table 374. Date/Time Shortcuts**

| Shortcut | Description |
|---|---|
| Now | Now (the time and date that you execute the query) |
| Today | Today at midnight |
| Yesterday | Yesterday at midnight |
| BOY | First day of year at midnight |
| EOY | Last day of year at midnight |
| BOM | First day of month at midnight |
| EOM | Last day of month at midnight |

**Relative Date/Time Shortcuts**

Optionally, you can add or subtract relative times to the following absolute times. You must use them in conjunction with the date/time shortcuts listed in the preceding table (for example, Today+5h+3min instead of 5h3min).

- Second
- Minute
- Hour
- Day
- Week

## Converting a Collected Value

The following code sample converts a temperature value from degrees Celsius to degrees Fahrenheit.

```
Result=CurrentValue("Temp F")*(9/5)+32
```

## Calculations Inside Formulas

The following code sample contains a calculation within a formula. In this case, we are taking the average of values of the tag `Simulation00001` over the previous hour. Typically, use a polled trigger to schedule the execution of the formula.

```
Result=Calculation("Simulation00001","Average","Now-1hour","Now",Quality)
```

## Conditional Calculation

The following code sample stores the value of a tag only if it is 100.

```
IF CurrentQuality("Simulation00001")=100 THEN

Result=CurrentValue("Simulation00001")

END IF
```

## Combining Tag Values and Assigning a Trigger

The following code sample adds current values of multiple tags using two calculation triggers.

```
Result=CurrentValue("SERVER1.Simulation00003")+CurrentValue("SERVER1.Simulation00006")
```

The calculation triggers used in the sample are SERVER1.Simulation0003 and SERVER1.Simulation0006. The calculation is triggered if the value of either Server1.Simulation0003 or Server1.Simulation0006 changes.

## Using CreateObject in a Formula

The following code sample reads data from another Historian Server using the Historian OLE DB provider, and stores it in a destination tag. When using this example, specify the username and password.

```
'connection and recordset variables

Dim Cnxn

Dim rsCurrentValueFromOtherServer

'open connection

Set Cnxn = CreateObject("ADODB.Connection")

'connect to default server using current username and password

'establish connection

Cnxn.Open "Provider=ihOLEDB.iHistorian.1;User Id=;Password="

'Create and open first Recordset using Connection execute

Set rsCurrentValueFromOtherServer = CreateObject("ADODB.Recordset")

'Get the value from the other server

Set rsCurrentValueFromOtherServer = Cnxn.Execute("select value from ihRawData

where SamplingMode=CurrentValue and tagname = Simulation00001")

'Set the result to the current value of other tag

Result=rsCurrentValueFromOtherServer("Value")

'Clean up

IF rsCurrentValueFromOtherServer.State = adStateOpen THEN
```

```
rsCurrentValueFromOtherServer.Close

END IF

IF Cnxn.State = adStateOpen THEN Cnxn.Close

END IF

Set rsCurrentValueFromOtherServer = Nothing

Set Cnxn = Nothing
```

## Using a File

The following code sample shows how to read and write text files during a calculation. You may have data in a file to use as input to a calculation, or you may want to write debug values to a text file instead of using the `LogMessage` function.

```
Dim filesys, writefile, count,readfile

'need to create a file system object since there is no

'file I/O built into VBScript

Set filesys = CreateObject("Scripting.FileSystemObject")

'open the text file, or create it if it does not exist

set readfile = filesys.OpenTextFile("C:\somefile.txt", 1, true)

'try to read from the file

IF readfile.AtEndOfLine <> true THEN

count= readfile.ReadAll

END IF

'add one to the number stored in the count count = count+1

'close the file for reading

readfile.Close

'open the same file but for writing

Set writefile= filesys.OpenTextFile("C:\somefile.txt", 2, true)

'write the updated count writefile.Write count

'close file for writing

writefile.Close

Result = count
```

## Converting a Number to a String

If your device and collector expose data as numeric codes, you can change to a string description. This examples also demonstrates that a calculation can output a string.

```
DIM X

x=CurrentValue ("tag1")
```

```
select case x

case 1

Result="one"

case 2

Result="two"

case else

Result="other"

End select
```

## Detecting Recovery Mode Inside a Formula

The following code sample detects the recovery mode or recalculation inside a formula. If there are individual tags, you do not want to perform a recovery.

```
Dim MAXDIFF, TimeDiff

'Maximum difference in timestamps allowed (Must be > 2,

'units = seconds) MAXDIFF = 10

'Calculate time difference

TimeDiff = DateDiff("s", CurrentTime(), Now)

'Compare times, if difference is < MAXDIFF seconds perform calc

If TimeDiff < MAXDIFF Then

'Place calculation to be performed here:

Result = CurrentValue("DENALI.Simulation00001") Else

'Place what is to be done when no calc is performed here

Result = Null

End If
```

## Looping Through Data Using the SDK

The following code sample uses the SDK to perform a query on a data set. It determines the minimum raw value over a one-hour time period.

```
on error resume next

Dim MyServer 'As Historian_SDK.Server

Dim I

Dim J

Dim K

Dim strComment

Dim lngInterval

Dim TagCount
```

```
Dim strDataQuality

Dim iDataRecordset

Dim iDataValue

Dim lEndTime, lStartTime, lNumSamples

Dim lNumSeconds, lNumSamplesPerSecond

Dim RawMin

'Instantiate The SDK

Set MyServer = CreateObject("iHistorian_SDK.Server")

'Attempt Connection

If Not MyServer.Connect("DENALI", "administrator","") Then

result = err.description

else

Set iDataRecordset = MyServer.Data.NewRecordset

'Find the number of samples.

'build query

With iDataRecordset

.Criteria.Tagmask = "EIGER.Simulation00001"

.Criteria.StartTime = DateAdd("h",-1,Now)

.Criteria.EndTime = Now

.Criteria.SamplingMode = 4 'RawByTime

.Criteria.Direction = 1 'forward

.Fields.AllFields

'do query

If Not .QueryRecordset Then

result = err.description

End If

'Some Large number so that real samples are less

RawMin = 1000000

For I = 1 To iDataRecordset.Tags.Count

For J = 1 To iDataRecordset.Item(I).Count

Set iDataValue = iDataRecordset.Item(I).Item(J)

' if the value is good data quality

if iDataValue.DataQuality = 1 then

if iDataValue.Value < RawMin then

rawMin = iDataValue.Value

end if

end if
```

```
lNumSamples = lNumSamples + 1

Next

Next

End With

End If

Result = RawMin

'Disconnect from server

MyServer.Disconnect
```

## Using an ADO Query

The following code sample uses a query combining Historian data with ADO data. In the example, you convert a collected value, number of barrels per day (`BarrelsUsedToday`), to a dollar amount. The code then obtains the price per barrel (`CostOfBarrel`) from the SQL server, and finally stores the total dollars in an integer tag (`TotalCostToday`).

You can also do this with a linked server and the Historian OLE DB provider, but this example maintains a history of the results.

```
Dim CostOfBarrel, BarrelsUsedToday, TotalCostToday

'Calculate the total number of barrels used over

'the previous 24hours.

BarrelsUsedToday = Calculation("BarrelsUsedTag","Total","Now 1Day","Now",Quality)

'Retrieve cost per barrel used

Dim SQLExpression

Dim Cnxn

Dim rsCurrentValue

SQLExpression = "SELECT Barrel_Cost AS Value1 FROM RawMaterial_Costs WHERE Barrel_Type = CrudeOil and

samplingmode = CurrentValue"

'open connection

Set Cnxn = CreateObject("ADODB.Connection")

'connect to default server using current username and password

'establish connection

Cnxn.Open "Provider=SQLOLEDB.1;User ID=sa; Password=;Initial Catalog=Northwind"

'Create and open first Recordset using Connection execute

Set rsCurrentValue = CreateObject("ADODB.Recordset")

'Get the value from the other server

Set rsCurrentValue= Cnxn.Execute(SQLExpression)

'Set the result to the current value of other tag
```

```
CostOfBarrel = rsCurrentValue("Value1")

'Clean up

If rsCurrentValue.State = adStateOpen then

rsCurrentValue.Close

End If

If Cnxn.State = adStateOpen then

Cnxn.Close

End If

Set rsCurrentValue = Nothing

Set Cnxn = Nothing

'Retrieve number of barrels used

BarrelsUsedToday = Calculation("BarrelsUsed","Count","Now 1Day","Now",Quality)

'Calculate total cost of barrels today

TotalCostToday = CostOfBarrel * BarrelsUsedToday
```

## Windows Performance Statistics Physical Memory Usage

The following code sample creates a formula that collects data reflecting private byte usage.

```
`Get a reference to the local data archiver process object

Set RawProc = GetObject("winmgmts:Win32_PerfRawdata_Perfproc_process.name='ihDataArchiver.'")

`Scale the virtual bytes number to a value within

`the tag's EGU range

result =RawProc.PrivateBytes *.001
```

## Windows Performance Statistics Virtual Memory Usage

The following code sample creates a formula that collects data reflecting virtual byte usage.

```
`Get a reference to the local data archiver process object

                Set RawProc = GetObject("winmgmts:Win32_PerfRawdata_Perfproc_process.name='ihDataArchiver.'")

`Scale the virtual bytes number to a value within the

`tag's EGU range

result =RawProc.VirtualBytes *.0001
```

## Determining Collector Downtime

The following code sample determines the amount of downtime, in seconds, that the Calculation collector has experienced over the last day. Downtime occurs when there are two consecutive bad quality data points for the pulse tag. If the last known data point for the pulse tag is bad quality, all the time between

its timestamp and the current time is regarded as downtime. In the following sample, the pulse tag is configured to be polled, with a collection interval of one day.

```
Dim pulseTag, totalDownTime, startTime, endTime
Dim prevTime, prevQuality, lastPrevTime, lastPrevQuality
pulseTag = "calcPulseTag"
totalDownTime = 0
endTime = CurrentTime()
startTime = DateAdd("d", -1, endTime)
lastPrevTime = curTime lastPrevQuality = 0
Do
  'get the timestamp and quality of the tag value previous to the last one we checked
  On Error Resume Next
  prevTime = PreviousTime(pulseTag, lastPrevTime)
  If Err.Number <> 0 Then
    'no more values for this tag exit gracefully
    Exit Do
End If
prevQuality = PreviousQuality(pulseTag, lastPrevTime)
'if we have two consecutive bad data points, add to the downtime
If prevQuality = 0 And lastPrevQuality = 0 Then
  If prevTime > startTime Then
    totalDownTime = totalDownTime + DateDiff("s", prevTime, lastPrevTime)
Else
    totalDownTime = totalDownTime + DateDiff("s", startTime, lastPrevTime)
End If
End If
  'store the timestamp and quality for comparison with the next values
lastPrevQuality = prevQuality
  lastPrevTime = prevTime
Loop While lastPrevTime > startTime
Result = totalDownTime
```

## Analyzing the Collected Data

The following code sample analyzes the collected data to determine the amount of time that a condition was true and had good quality in the last day.

```
Dim tagName, startTime, endTime

tagName = "testTag"

startTime = "Now 1Day"

endTime = "Now"

Result = CalculationFilter(tagName, "TotalTimeGood", startTime, endTime, 100, tagName, "AfterTime", "Equal", 1)
```

## Simulating Demand Polling

To simulate demand polling, create the following tags.

| Tag | Description |
|---|---|
| Polled Tag | A polled tag with a collection interval of the longest period you want between raw samples. Do not enable collector or archive compression. This tag should point to the same source address as the unsolicited tag. |
| Unsolicited Tag | An unsolicited tag with a 0 or 1 second collection interval. This tag ensures you will be notified whenever changes occur. This tag should point to the same source address as the polled tag. |
| Combined Tag | An unsolicited calculation tag that is triggered by either the polled tag or the unsolicited tag, and combines the raw samples of both into a single tag. Use a 0 or 1 second collection interval and use the following formula:<br>`dim timetag1`<br>`dim timetag2`<br>`dim tag1`<br>`dim tag2`<br>`Dim x`<br>`tag1 = "T20.di-1.F_CV"`<br>`tag2 = "t20.T20.DI-1.F_CV"`<br>`x = DateAdd("s", 1,CurrentTime) ' add 1 second to calc time`<br>`' Get the timestamp of the newest raw sample for tag1:`<br>`timetag1 = previousTime(tag1, x)`<br>`' Get the timestamp of the newest raw sample for tag2:`<br>`timetag2 = previousTime(tag2, x)`<br>`if timetag1 > timetag2 then`<br>`' If tag1 triggered me, then:`<br>`result = PreviousValue(tag1,CurrentTime)`<br>`else` |

| Tag | Description |
| --- | --- |
| | ```<br>' If tag2 triggered me, then:<br><br>result = PreviousValue(Tag1, CurrentTime)<br><br>end if<br>``` |

# Native Alarms and Events Functions

## About Native Alarms and Events Functions

In addition to using the calculation wizard to create calculation formulas within your calculation tag, you can enter functions manually. The following sections list the functions available, along with their usage.

## Retrieving and Setting Alarm Properties Manually

Alarm and event properties can also be set manually in the Calculation collector.

1. To retrieve alarm properties, append the property name to the alarm object, using the following syntax:

   ```
   variable = AlarmObj.Property
   ```

   **Example**

   The following example retrieves the alarm's severity and places it in a variable named *ALM_Severity*.

   ```
   ALM_Severity = AlarmObj.Severity
   ```

2. To set alarm properties manually, append the property name to the alarm object and supply a new value for the property using the following syntax:

   ```
   AlarmObj.Propertyname = "Property Name"
   ```

   > **Important:**
   > The `Set Alarm Property` function will not save changes to the alarm database. A call to `UpdateAlarm` must be made after the `Set Alarm Property` function is called.

   **Example**

   The following example sets an alarm's severity to 100, then updates the alarm in the Historian archive.

```
AlarmObj.Severity = 100

AlarmObj.UpdateAlarm "Now"
```

## Insert Calculation Functions Manually

In addition to using the Calculation Wizard to create your alarms and events calculations, you can also enter them manually. The following functions are available:

### NextAlarm

The `NextAlarm` function returns the next alarm for a tag or source on or after a given time stamp.

#### Syntax

```
set AlarmObj = NextAlarm (source, condition, timestamp)
```

#### Example Code

The following example will get the next alarm based on the properties of the current alarm object.

```
Set AlarmObj = NextAlarm (AlarmObj.Source, AlarmObj.ConditionName,

DateAdd("s", 1, AlarmObj.Timestamp))
```

### NextAlarmForTag

The `NextAlarmForTag` function is identical to the `NextAlarm` function, but takes a tag name as its input instead of a source.

#### Syntax

```
Set AlarmObj = NextAlarmForTag (tag name, condition, timestamp)
```

#### Example Code

The following example will get the next alarm for the tag `SYN4450_Flow` with a `LevelAlarm` condition on or after the current alarm's timestamp.

```
Set AlarmObj = NextAlarmForTag ("SYN4450_Flow", "LevelAlarm", AlarmObj.Timestamp)
```

### PreviousAlarm

The `PreviousAlarm` function returns the previous alarm for a tag or source on or before a given timestamp.

#### Syntax

```
set AlarmObj = PreviousAlarm (source, condition, timestamp)
```

#### Example Code

The following example will get the previous alarm based on the properties of the current alarm object.

```
Set AlarmObj = NextAlarm (AlarmObj.Source, AlarmObj.ConditionName, AlarmObj.Timestamp)
```

### PreviousAlarmForTag

The `PreviousAlarmForTag` function is identical to the `PreviousAlarm` function, but takes a tag name as its input instead of a source.

#### Syntax

```
Set AlarmObj = PreviousAlarmForTag (tag name, condition, timestamp)
```

#### Example Code

The following example will get the previous alarm for the tag `SYN4450_Flow` with a `LevelAlarm` condition on or before the current time.

```
Set AlarmObj = PreviousAlarmForTag ("SYN4450_Flow", "LevelAlarm", "Now")
```

### AddEvent

The `AddEvent` method will create a new event with the current alarm properties.

#### Syntax

```
AlarmObj.NewAlarm source, tag, time stamp
```

#### Example Code

The following example creates a new event for the `Simulation00001` tag on `Simulation` source with a severity of `50`, a message of `Test Message`, and the current time.

```
Set AlarmObj = new Alarm

AlarmObj.Severity = 50

AlarmObj.Message = "Test Message"

AlarmObj.AddEvent "Simulation", "Simulation00001", "Now"
```

### NewAlarm

The `NewAlarm` method will create a new alarm, based on the current alarm object properties.

#### Syntax

```
AlarmObj.NewAlarm source, condition, tag, time stamp
```

#### Example Code

The following example creates a new alarm for the `Simulation00001` tag on `Simulation` source with a severity of `50`, a condition of `Low Fluid Levels`, and the current time.

```
Set AlarmObj = new Alarm

AlarmObj.Severity = 50

AlarmObj.Message = "SomeMsg"

AlarmObj.NewAlarm "Simulation", "Low Fluid Levels", "Simulation00001", "Now"
```

### NextAlarm

The `NextAlarm` function returns the next alarm for a tag or source on or after a given time stamp.

#### Syntax

```
set AlarmObj = NextAlarm (source, condition, timestamp)
```

#### Example Code

The following example will get the next alarm based on the properties of the current alarm object.

```
Set AlarmObj = NextAlarm (AlarmObj.Source, AlarmObj.ConditionName,

DateAdd("s", 1, AlarmObj.Timestamp))
```

### GetVendorAttribute

The `GetVendorAttribute` method will get the value of the given vendor attribute on the current alarm object and place it into a supplied variable.

#### Syntax

```
variable = AlarmObj.GetVendorAttribute (Vendor_Attribute)
```

#### Example Code

The following example retrieves a vendor attribute called `Cause_Comment` from the alarm object, and places it into the `ALM_Cause_Comment` variable.

```
ALM_Cause_Comment = AlarmObj.GetVendorAttribute ("Cause_Comment")
```

### SetVendorAttribute

The `SetVendorAttribute` method sets the value of the given vendor attribute on the current alarm object.

#### Syntax

```
AlarmObj.SetVendorAttribute Vendor_Attribute, Value
```

#### Example Code

The following example sets values for the vendor attributes `Cause_Comment` and `Status_Code`, then updates the alarm in the Historian archive.

```
AlarmObj.SetVendorAttribute "Cause_Comment", "This alarm was caused by..."

AlarmObj.SetVendorAttribute "Status_Code", 5032

AlarmObj.UpdateAlarm "Now"
```

**Important:**
The `SetVendorAttribute` method will not save changes to the alarm database. A call to `UpdateAlarm` must be made after the `SetVendorAttribute` method is set.

### UpdateAlarm

The `UpdateAlarm` function updates the current alarm with whatever changes have been made to an alarm's properties.

#### Syntax

```
AlarmObj.UpdateAlarm timestamp
```

#### Example Code

```
AlarmObj.UpdateAlarm "Now"
```

### ReturnToNormal

The `ReturnToNormal` method sets the end time for the current alarm.

#### Syntax

```
AlarmObj.ReturnToNormal timestamp
```

#### Example Code

The following example sets the end time for the alarm to the current time.

```
AlarmObj.ReturnToNormal "Now"
```

# Data Input

## Calculation and Server-to-Server Collectors

The Calculation and Server-to-Server collectors have some unique behavior not found in other standard collectors. This section provides details about Recovery *(on page 2152)* and Manual Recalculation *(on page 2153)*.

## Recovery

This feature is unique to Calculation and Server-to-Server collectors. If the calculation engine is not running for a period of time, recovery makes it look like it was running. Recovery can also be used to fill in a hole of time where the collector was not able to communicate with the source archiver.

Recovery is applicable to both unsolicited and polled tags. Messages are also recovered. Comments are not recovered.

Normally, it is impossible to go back to the past and collect data. However, since these collectors are 'deriving' data instead of 'collecting' data, it is possible to recover past data, especially since the source of the derived data is archived in the Historian. It is important to understand that while recovery is possible in the calculation and Server-to-Server collectors, it only makes sense for certain types of calculation formulas.

Intended candidates for data recovery are formulas whose only inputs are Historian tags, since past data for these tags can be interpolated. Formulas that use data from external text files or from ADO via CreateObject will most likely not recover correct data because the inputs are not historized. If you are using these types of formulas, you should turn off recovery for the whole collector or insert VBScript code in the formula of individual tags to detect recovery. An example of this is given in the Historian documentation. A similar approach can be used to set a Max Recovery Time on a tag basis, overriding the collector wide setting.

Even calculation tags using only Historian tags as inputs have some caveats for recovery. If you are deriving calculated data from other calculated data, be sure to set up a trigger tag for each of the tags used in your formula. This way the tags will be processed in chain order. All tags are processed in time order.

The recovery logic is not intended to overcome polled collection overruns. If you configure too much collection, then you will get overruns.

You can control the amount of recovered data using Max Recovery Time configuration setting. You can turn off the recovery by setting it to zero.

## Manual Recalculation

The Manual re-calc/re-replicate option is often the best choice for generating past derived data.

**Note:**
If you perform a server-to-server recalculation on source and destination servers whose clocks are not synchronized, extra data points may appear and original data points may not be recalculated. To ensure this does not occur, ensure the time is synchronized on both source and destination servers.

**S2S/S2C collector Backfill procedure**

With the Recalculate feature you can recalculate all tags for the time period during and after the connection loss. The recalculated tags will use the most accurate values in calculations.

During the period of connection loss, the collector buffers the data. When the connection is restored, the buffered data is forwarded to the Historian Server. When the buffered data arrives, the timestamps show earlier time than the most recent calculation timestamp.

Since the timestamp is earlier, the polled calculations will not execute again with the new data but the unsolicited calculations will re-trigger. Therefore, it is possible that calculations performed for tags during and after the connection loss might be not be entirely accurate.

**Run S2C Backfill via Command line**

```
ihServerToServerCollector.exe RELOADFILENAME=[file location]

RELOADUSERNAME=[Username] <start time> <end time>
```

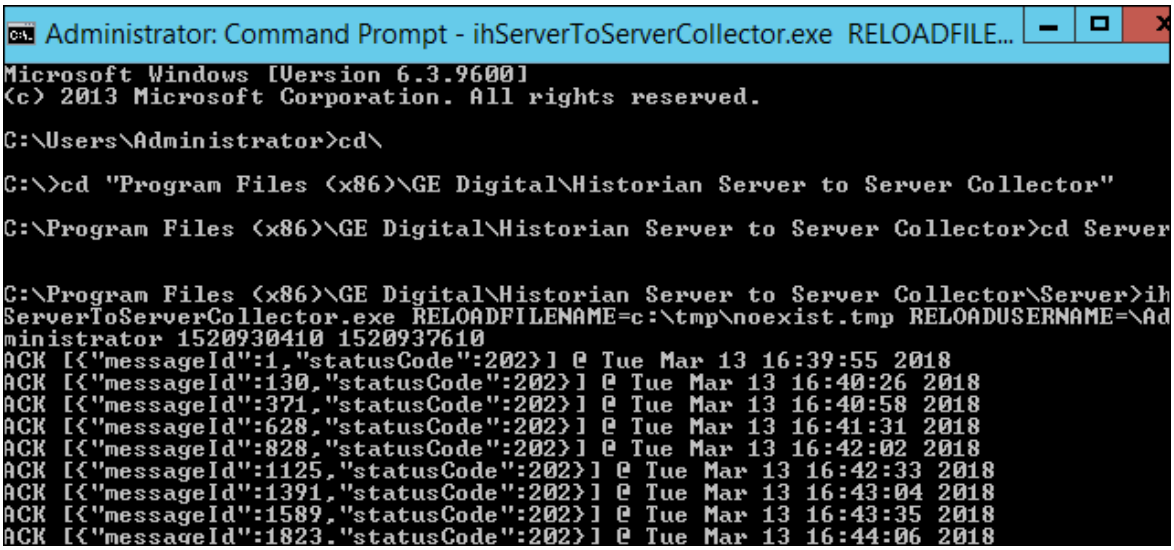
**Example**: `C:\Program Files\Proficy\Proficy Historian\x86\Server>ihServerToServerCollector.exe RELOADFILENAME=c:\taglist.txt RELOADUSERNAME=\Administrator 1516875659 1516875785`

For multi-instance support, the command requires the interface name as shown in the following example:

**Example for Multi-instance Support**: `C:\Program Files (x86)\GE Digital\Historian Server to Server Collector\Server>ihServerToServerCollector.exe RELOADFILENAME=c:\taglist.txt RELOADUSERNAME= \Administrator 1669462600 1669463200 REG=sekhartest05_To_PredixOFFSS`

See the following information about the parameters:

- RELOADFILENAME: This is an optional parameter. File name should be absolute path, this file consists of the tag names, for which Backfill should be performed, each tag should be separated by new line. Any discrepancies in the file/no file exists/parameter not provided leads to Backfill all the tags related to the collector at the current time. After the Backfill, file gets deleted.
- RELOADUSERNAME: This is an optional parameter. This username is used only when destination server is Historian for auditing purpose, and gets ignored when the destination is cloud.
- TIMESTAMP: This parameter accepts Start and end time in seconds in epoch format for which Backfill should happen. https://www.epochconverter.com/




**How Data Recovery works:**

- When the recovery logic is executed, the collector will setup subscriptions for all the trigger tags.
- Next, it will recover data. The collector first determines how long it has been since the last write. It compares the current time to data in the registry key `LastCalcRepWriteTime`, which stores the last time data was written to the archive. The collector compares this to the Max Recovery Time that is specified in the user settings and performs a raw data query on the shorter of these two periods. Then it will take the shorter of these two and do a raw data query for all trigger tags. It will then process the returned samples in sequential order based on time. For example, if the collector was shut down for 8 hours, but Max Recovery Time was 4 hours, only 4 hours of data would be recovered.
- Recovery is performed before real time processing. Once recovery is complete, it will start polling and processing subscriptions in real time. The subscriptions in real time are queued up till the recovery is done.

- Recovery logic will place an end-of-collection marker at the point in time where the collector was shut down. This end-of-collection marker may or may not be there once the recovery is complete. As part of recovery logic, if it calculates a data point exactly at that timestamp where the end-of-collection marker is there, then it will be overwritten with the calculated good data.
- The recovery logic does not write samples to trigger tags or tags that are just in the formula. It is intended to write samples to the calculation tags.
- Messages are added to the log file that indicate when entering and exiting recovery mode.

**Examples**

The examples below assume the following tag configuration.

- Machine 1:

  Runs Data Archiver, iFIX collector (Collector 1), and Calculation collectors.

- Machine 2:

  Runs iFIX collector (Collector 2), which collects and sends data to the archiver in Machine 1 (as a Remote Collector).

  TagA and TagB are the iFix tags coming from Collector1 and Collector2, respectively. Both of these tags are scanned at a 1-minute poll rate.

The following example demonstrates the recovery function for an unsolicited 1-minute interval calculation tag that has a simple current value function.

Create an event based 1-minute interval Calculation Tag (CalcTag1) in Machine 1 consisting of the following calculation: `Result=CurrentValue (TagA)`

Stop the Calculation collector for 5 minutes and then restart it to trigger data recovery for the 5-minute shutdown period. For the following example, the Calculation collector was stopped at 2002-12-27 17:05:36 and started at 2002-12-27 17:10:48.

Since there is no interruption for the iFIX collector, the raw data query for TagA results the following output:

***Raw Data Query for TagA during shutdown period***

114) 81 [2002-12-27 17:02:00:00000] Good NonSpecific

115) 72 [2002-12-27 17:03:00:00000] Good NonSpecific

116) 64 [2002-12-27 17:04:00:00000] Good NonSpecific

117) 56 [2002-12-27 17:05:00:00000] Good NonSpecific

118) 39 [2002-12-27 17:06:00:00000] Good NonSpecific

119) 31 [2002-12-27 17:07:00:00000] Good NonSpecific

120) 22 [2002-12-27 17:08:00:00000] Good NonSpecific

121) 14 [2002-12-27 17:09:00:00000] Good NonSpecific

122) 6 [2002-12-27 17:10:00:00000] Good NonSpecific

A raw data query for CalcTag1 during the shutdown period generates the following:

***Raw Data Query for CalcTag1 (before recovery)***

96) 81 [2002-12-27 17:02:00:00000] Good NonSpecific

97) 72 [2002-12-27 17:03:00:00000] Good NonSpecific

98) 64 [2002-12-27 17:04:00:00000] Good NonSpecific

99) 56 [2002-12-27 17:05:00:00000] Good NonSpecific

100) 0 [2002-12-27 17:05:36:00000] Bad OffLine

Note that an end-of-collection marker is placed at the shutdown point (that is, at 17:05:36) with a bad data quality.

Once the recovery is complete, this is what we see for the recovered CalcTag1. Note that data during the shutdown period is recovered completely. Compare this result set with the one for TagA. Both are the same.

*Raw Data Query for CalcTag1 (after recovery)*

96) 81 [2002-12-27 17:02:00:00000] Good NonSpecific

97) 72 [2002-12-27 17:03:00:00000] Good NonSpecific

98) 64 [2002-12-27 17:04:00:00000] Good NonSpecific

99) 56 [2002-12-27 17:05:00:00000] Good NonSpecific

100) 0 [2002-12-27 17:05:36:00000] Bad OffLine

101) 39 [2002-12-27 17:06:00:00000] Good NonSpecific

102) 31 [2002-12-27 17:07:00:00000] Good NonSpecific

103) 22 [2002-12-27 17:08:00:00000] Good NonSpecific

104) 14 [2002-12-27 17:09:00:00000] Good NonSpecific

105) 6 [2002-12-27 17:10:00:00000] Good NonSpecific

Also note that the end-of-collection marker is not overwritten by the recovery logic here. If it calculated a data point exactly at the end-of-collection marker, then it would have been overwritten by the calculated good value.

The following example demonstrates the recovery function for an unsolicited calculation tag that has multiple triggers.

Create an event based Calculation Tag (CalcTag2) in Machine 1 consisting of the following calculation:

```
Result=CurrentValue (TagA) + CurrentValue (TagB)
```

where TagA and TagB are both trigger tags, coming from Collector1 and Collector2 respectively. Set the collection offset of 5 seconds for TagA and 10 seconds for TagB, forcing the calculation to be performed twice per minute.

Stop the Calculation collector for 5 minutes, and then restart it to trigger data recovery for this 5-minutes shutdown period. For the following example, the Calculation collector was stopped at 02/18/2003 12:15:33 and started at 02/18/2003 12:21:53.

Since the iFIX collector was not interrupted, a raw data query for TagA and TagB values generates the following output:

***Raw Data Query for TagA during the shutdown period***

10) 13 [2003-02-18 12:10:05:00000] Good NonSpecific

11) 12 [2003-02-18 12:11:05:00000] Good NonSpecific

12) 11 [2003-02-18 12:12:05:00000] Good NonSpecific

13) 11 [2003-02-18 12:13:05:00000] Good NonSpecific

14) 10 [2003-02-18 12:14:05:00000] Good NonSpecific

15) 18 [2003-02-18 12:15:05:00000] Good NonSpecific

16) 17 [2003-02-18 12:16:05:00000] Good NonSpecific

17) 16 [2003-02-18 12:17:05:00000] Good NonSpecific

18) 16 [2003-02-18 12:18:05:00000] Good NonSpecific

19) 15 [2003-02-18 12:19:05:00000] Good NonSpecific

20) 14 [2003-02-18 12:20:05:00000] Good NonSpecific

21) 13 [2003-02-18 12:21:05:00000] Good NonSpecific

***Raw Data Query for TagB during the shutdown period***

10) 35 [2003-02-18 12:10:10:00000] Good NonSpecific

11) 34 [2003-02-18 12:11:10:00000] Good NonSpecific

12) 33 [2003-02-18 12:12:10:00000] Good NonSpecific

13) 32 [2003-02-18 12:13:10:00000] Good NonSpecific

14) 31 [2003-02-18 12:14:10:00000] Good NonSpecific

15) 31 [2003-02-18 12:15:10:00000] Good NonSpecific

16) 39 [2003-02-18 12:16:10:00000] Good NonSpecific

17) 38 [2003-02-18 12:17:10:00000] Good NonSpecific

18) 37 [2003-02-18 12:18:10:00000] Good NonSpecific

19) 36 [2003-02-18 12:19:10:00000] Good NonSpecific

20) 36 [2003-02-18 12:20:10:00000] Good NonSpecific

21) 35 [2003-02-18 12:21:10:00000] Good NonSpecific

A raw data query for CalcTag2 during the shutdown period generates the following:

*Raw Data Query for CalcTag2 (before recovery)*

12) 50 [2003-02-18 12:09:05:00000] Good NonSpecific

13) 50 [2003-02-18 12:09:10:00000] Good NonSpecific

14) 49 [2003-02-18 12:10:05:00000] Good NonSpecific

15) 48 [2003-02-18 12:10:10:00000] Good NonSpecific

16) 47 [2003-02-18 12:11:05:00000] Good NonSpecific

17) 46 [2003-02-18 12:11:10:00000] Good NonSpecific

18) 45 [2003-02-18 12:12:05:00000] Good NonSpecific

19) 44 [2003-02-18 12:12:10:00000] Good NonSpecific

20) 44 [2003-02-18 12:13:05:00000] Good NonSpecific

21) 43 [2003-02-18 12:13:10:00000] Good NonSpecific

22) 42 [2003-02-18 12:14:05:00000] Good NonSpecific

23) 41 [2003-02-18 12:14:10:00000] Good NonSpecific

24) 49 [2003-02-18 12:15:05:00000] Good NonSpecific

25) 49 [2003-02-18 12:15:10:00000] Good NonSpecific

26) 0 [2003-02-18 12:15:11:00000] Bad OffLine

Once data recovery is complete, this is what we see for the recovered data for CalcTag2. Note that data during the shutdown period is completely recovered:

***Raw Data Query for CalcTag2 (after recovery)***

12) 50 [2003-02-18 12:09:05:00000] Good NonSpecific

13) 50 [2003-02-18 12:09:10:00000] Good NonSpecific

14) 49 [2003-02-18 12:10:05:00000] Good NonSpecific

15) 48 [2003-02-18 12:10:10:00000] Good NonSpecific

16) 47 [2003-02-18 12:11:05:00000] Good NonSpecific

17) 46 [2003-02-18 12:11:10:00000] Good NonSpecific

18) 45 [2003-02-18 12:12:05:00000] Good NonSpecific

19) 44 [2003-02-18 12:12:10:00000] Good NonSpecific

20) 44 [2003-02-18 12:13:05:00000] Good NonSpecific

21) 43 [2003-02-18 12:13:10:00000] Good NonSpecific

22) 42 [2003-02-18 12:14:05:00000] Good NonSpecific

23) 41 [2003-02-18 12:14:10:00000] Good NonSpecific

24) 49 [2003-02-18 12:15:05:00000] Good NonSpecific

25) 49 [2003-02-18 12:15:10:00000] Good NonSpecific

26) 0 [2003-02-18 12:15:11:00000] Bad OffLine

27) 48 [2003-02-18 12:16:05:00000] Good NonSpecific

28) 56 [2003-02-18 12:16:10:00000] Good NonSpecific

29) 55 [2003-02-18 12:17:05:00000] Good NonSpecific

30) 54 [2003-02-18 12:17:10:00000] Good NonSpecific

31) 54 [2003-02-18 12:18:05:00000] Good NonSpecific

32) 53 [2003-02-18 12:18:10:00000] Good NonSpecific

33) 52 [2003-02-18 12:19:05:00000] Good NonSpecific

34) 51 [2003-02-18 12:19:10:00000] Good NonSpecific

35) 50 [2003-02-18 12:20:05:00000] Good NonSpecific

36) 50 [2003-02-18 12:20:10:00000] Good NonSpecific

37) 49 [2003-02-18 12:21:05:00000] Good NonSpecific

38) 48 [2003-02-18 12:21:10:00000] Good NonSpecific

39) 47 [2003-02-18 12:22:05:00000] Good NonSpecific

40) 46 [2003-02-18 12:22:10:00000] Good NonSpecific

The following example demonstrates the recovery function for an unsolicited calculation tag that has multiple triggers, but for which none of the triggers is in the formula.

TagA and TagB are the iFix tags coming from Collector1 and Collector2, respectively. Both tags are scanned at a 1-minute poll rate. This example uses two more iFix tags, TagC and TagD, coming from Collector1.

Create an event-based Calculation Tag (CalcTag3) in Machine 1 consisting of the following calculation:

```
Result=CurrentValue (TagA) + CurrentValue (TagB)
```

Make sure that the trigger tags for this calculation tag are TagC and TagD, which are not in the formula. Set the collection offset of 5 seconds for TagC and 10 seconds for TagD, forcing the calculation to be performed twice per minute.

Stop the Calculation collector for 5 minutes, and then restart it to trigger data recovery for this 5-minutes shutdown period. For the following example, the Calculation collector was stopped at 02/18/2003 02:24:37 and started at 02/18/2003 02:31:44.

Since the iFIX collector was not interrupted, a raw data query for TagA and TagB values generates the following output:

***Raw Data Query for TagA during shutdown period***

56) 13 [2003-02-18 14:21:05:00000] Good NonSpecific

57) 12 [2003-02-18 14:22:05:00000] Good NonSpecific

58) 11 [2003-02-18 14:23:05:00000] Good NonSpecific

59) 11 [2003-02-18 14:24:05:00000] Good NonSpecific

60) 10 [2003-02-18 14:25:05:00000] Good NonSpecific

61) 19 [2003-02-18 14:26:05:00000] Good NonSpecific

62) 18 [2003-02-18 14:27:05:00000] Good NonSpecific

63) 17 [2003-02-18 14:28:05:00000] Good NonSpecific

64) 16 [2003-02-18 14:29:05:00000] Good NonSpecific

65) 16 [2003-02-18 14:30:05:00000] Good NonSpecific

66) 15 [2003-02-18 14:31:05:00000] Good NonSpecific

*Raw Data Query for TagB during shutdown period*

141) 36 [2003-02-18 14:20:10:00000] Good NonSpecific

142) 36 [2003-02-18 14:21:10:00000] Good NonSpecific

143) 35 [2003-02-18 14:22:10:00000] Good NonSpecific

144) 34 [2003-02-18 14:23:10:00000] Good NonSpecific

145) 33 [2003-02-18 14:24:10:00000] Good NonSpecific

146) 32 [2003-02-18 14:25:10:00000] Good NonSpecific

147) 31 [2003-02-18 14:26:10:00000] Good NonSpecific

148) 31 [2003-02-18 14:27:10:00000] Good NonSpecific

149) 39 [2003-02-18 14:28:10:00000] Good NonSpecific

150) 38 [2003-02-18 14:29:10:00000] Good NonSpecific

151) 37 [2003-02-18 14:30:10:00000] Good NonSpecific

152) 36 [2003-02-18 14:31:10:00000] Good NonSpecific

A raw data query for CalcTag3 during the shutdown period generates the following:

*Raw Data Query for CalcTag3 (before recovery)*

6) 49 [2003-02-18 14:21:05:00000] Good NonSpecific

7) 49 [2003-02-18 14:21:10:00000] Good NonSpecific

8) 48 [2003-02-18 14:22:05:00000] Good NonSpecific

9) 47 [2003-02-18 14:22:10:00000] Good NonSpecific

10) 46 [2003-02-18 14:23:05:00000] Good NonSpecific

11) 45 [2003-02-18 14:23:10:00000] Good NonSpecific

12) 45 [2003-02-18 14:24:05:00000] Good NonSpecific

13) 44 [2003-02-18 14:24:10:00000] Good NonSpecific

14) 0 [2003-02-18 14:24:11:00000] Bad OffLine

A data query for the recovered CalcTag3 values once data recovery is complete generates the following. Note that data during the shutdown period is completely recovered:

***Raw Data Query for CalcTag3 (after recovery)***

6) 49 [2003-02-18 14:21:05:00000] Good NonSpecific

7) 49 [2003-02-18 14:21:10:00000] Good NonSpecific

8) 48 [2003-02-18 14:22:05:00000] Good NonSpecific

9) 47 [2003-02-18 14:22:10:00000] Good NonSpecific

10) 46 [2003-02-18 14:23:05:00000] Good NonSpecific

11) 45 [2003-02-18 14:23:10:00000] Good NonSpecific

12) 45 [2003-02-18 14:24:05:00000] Good NonSpecific

13) 44 [2003-02-18 14:24:10:00000] Good NonSpecific

14) 0 [2003-02-18 14:24:11:00000] Bad OffLine

15) 43 [2003-02-18 14:25:05:00000] Good NonSpecific

16) 42 [2003-02-18 14:25:10:00000] Good NonSpecific

17) 51 [2003-02-18 14:26:05:00000] Good NonSpecific

18) 50 [2003-02-18 14:26:10:00000] Good NonSpecific

19) 49 [2003-02-18 14:27:05:00000] Good NonSpecific

20) 49 [2003-02-18 14:27:10:00000] Good NonSpecific

21) 48 [2003-02-18 14:28:05:00000] Good NonSpecific

22) 56 [2003-02-18 14:28:10:00000] Good NonSpecific

23) 55 [2003-02-18 14:29:05:00000] Good NonSpecific

24) 54 [2003-02-18 14:29:10:00000] Good NonSpecific

25) 54 [2003-02-18 14:30:05:00000] Good NonSpecific

26) 53 [2003-02-18 14:30:10:00000] Good NonSpecific

27) 52 [2003-02-18 14:31:05:00000] Good NonSpecific

28) 51 [2003-02-18 14:31:10:00000] Good NonSpecific

29) 49 [2003-02-18 14:32:05:00000] Good NonSpecific

30) 49 [2003-02-18 14:32:10:00000] Good NonSpecific

31) 48 [2003-02-18 14:33:05:00000] Good NonSpecific

32) 47 [2003-02-18 14:33:10:00000] Good NonSpecific

# Troubleshoot Calculation Collector

## Troubleshooting Calculation collector

If you observer errors with Calculation collector, you should:

1. Examine the collector log files in the `Historian\LogFiles` folder and scan for errors in the log that may explain your problem.
2. Check the connection to the source.
   For example, if you lose connection to the source, all tags for that collector stop collecting. What that means is that even if you hard-code the formula results, the collector will not collect them. If you enter a formula such as `result=7` on a polled, 1-second tag in the Server-to-Server Collector, Historian does not log any raw samples to the destination computer. This action occurs even though the result is hardcoded.
3. Ensure that you are following the guidelines described in General Guidelines for Designing a Calculation Formula *(on page 2119)*.

> **Tip:**
> use the `LogMessage` function to include messages in your formula when certain points of a calculation executes so that you can isolate errors. Refer to Writing Messages to the Collector Log File for Debugging Purposes *(on page 2165)* for details.

## Unsupported Data Types for Calculation Tags

Calculation tags with Quad Integer and Unsigned Quad Integer data types return bad quality values due a limitation in Visual Basic (VB).

## Unsupported Calculations in Calculation collector

Calculation collector supports only the calculations performed using the current value calculation. It does not support other calculations due to a Visual Basic script limitation.

## Writing Messages to the Collector Log File for Debugging Purposes

If you want to include debugging messages after certain parts of your formula execute, you can you can use the LogMessage function when creating a formula in the Calculation pane. The syntax of this function is as follows:

```
LogMessage(message_string)
```

where `message_string` is the message that you want to appear in the log file for the Calculation or Server-to-Server Collector. If the `message_string` is not a string variable, use double quotes around the text for `message_string` value. For instance, a properly formatted text string with double quotes would appear like this:

```
LogMessage("This is a message")
```

**Note:**
The `LogMessage` function does not appear in the wizard.

## Importing Calculations with Line Breaks into Historian

You can import calculations with line breaks into Historian when you use the File collector or the Excel Add-in.

1. To create a line break in a `.CSV` file for the File collector, you insert the `*CR*` character sequence where you want each line break to occur.
2. To create a calculation with multiple lines in Excel, press **Alt+Enter** at the end of each line where you want a line break to occur within a cell.
   You can also use the `*CR*` character sequence to denote a line break in the formula.

   **Example of a .CSV File that Includes a Calculation with Multiple Lines**

   In the following example the bold ***CR*** characters identify where the line breaks occur in the calculation formula.

   ```
   [Tags]


   Tagname,Description,DataType,HiEngineeringUnits,LoEngineeringUnits,Calculation,
   CollectorType,CollectorName,CollectionType,CalculationDependencies
   CalTag16,Multiline calc tag sample,SingleFloat,35000,0, "'multiline calc comment*CR*IF CurrentQuality
   (^Fixlab15.simulation00001^)=100 THEN*CR*Result=CurrentValue(^Fixlab15.simulation00002^)
   *CR*END IF",Calculation,Fixlab15_Calculation,Unsolicited,Fixlab15.simulation00001
   ```

> **Important:**
> For this example to work, only include three line breaks: one after the word [Tags], one after the word `CalculationDependencies` on line 3, and one at the very end of the example. It is important that the last 4 lines of this example all appear on the same line in the actual `.CSV` file. The example only includes extra line breaks so that the text is more easily readable, and does not flow off the page.

## Recovery Mode

Recovery logic is activated when the Calculation collector and Historian Server reestablish connection after a connection loss. Recovery mode allows the collector to recover data when the connection between the collector and the server is reestablished. Recovery mode produces calculated values for time when the Calculation collector was not running.

When using recovery mode, all referenced tags in an unsolicited calculation must be listed as trigger tags. For more information, refer to About Recovery Mode *(on page 2503)*.

# Chapter 16. The CygNet Collector

## Overview of the CygNet Collector

The CygNet collector collects data from a CygNet server and stores it in the Historian server.

**Topology:** The CygNet collector supports a distributed model, where the CygNet server, the collector, and the Historian server are installed on different machines. Typically, however, the collector is installed on the same computer as the CygNet server and sends data to a remote Historian server.

**Features:**

- You can browse the source for tags and their attributes on a CygNet server that supports browsing.
- Both the polled as well as unsolicited data collection are supported. During unsolicited data collection, when changes to the CygNet tags are detected, they are forwarded to the Historian server. The collector duplicates raw samples from the CygNet server into the Historian data archive.
- The supported timestamp resolution is 100ms.
- The collector support manual recalculation of tag data.
- Floating point, integer, and string data are supported.

**Components:**

- System API
- Collector Toolkit

**How it works:**

1. When you browse for tags, the CygNet collector connects with CygNet using the CxScript64.dll API.
2. The collector queries the CygNet faculty service and point service associated with the Current Value service (CVS).
3. The CygNet point value is mapped to the Historian server as a collector tag.
4. The collector stores the Value Historian service (VHS) and points Uniform Data Codes as the tag source address.
5. Once all the CygNet points associated with the selected CVS/VHS have been mapped with the Historian tags, the collector begins querying from VHS to collect data from CygNet.

**Supported data types:**

The CygNet collector collects analog, digital, and string types of data. The following table provides the data types recommended for use with Historian.

| CygNet Data Types | Recommended Data Type in Historian |
|---|---|
| Analog | Float |
| Enumeration | Integer |
| Digital | Boolean |
| String | Variable String |

**Supported tag attributes:**

- Tagname
- Source Address
- Engineering Unit Description
- Data Type
- Hi Engineering Units and Lo Engineering Units (applicable only to analog and discrete data types)

**Note:**
Although you can browse and query some of these attributes, they are not displayed in the browse interface. These attributes are used when you add a tag, but are not visible to you, regardless of attributes available from the server.

# Configuration

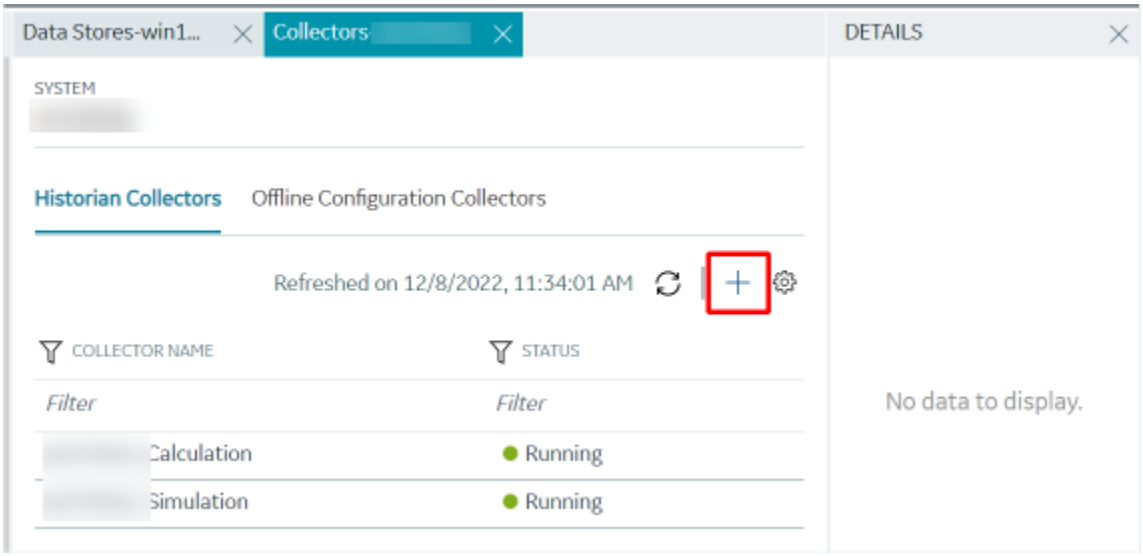
## Add and Configure a CygNet Collector

A CygNet collector collects data from a CygNet server and stores it in the Historian server. For more information, refer to Overview of the CygNet Collector *(on page 2167)*.

**Note:**
You cannot send data to a cloud destination using a CygNet collector.

This topic describes how to add a collector instance using Configuration Hub. You can also add a collector instance using the RemoteCollectorConfigurator utility *(on page 797)*, which does not require you to install Web-based Clients.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.
   A list of collectors in the default system appears.
3. In the upper-right corner of the main section, select +.




   The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.
4. In the **MACHINE NAME** field, select the machine in which you want to add a collector instance.
5. In the **COLLECTOR TYPE** field, select **Cygnet Collector**, and then select **Get Details**.
   The **INSTALLATION DRIVE** and **DATA DIRECTORY** fields are disabled and populated.
6. Select **Next**.
   The **Source Configuration** section appears.
7. In the **SERVER SITE** field, enter the host name or IP address of the CygNET server from which you want to collect data.
8. Select **Next**.
   The **Destination Configuration** section appears. Under **CHOOSE DESTINATION**, the **Historian Server** option is selected by default; the other options are disabled.
9. If you created security groups or enabled a strict client/collector authentication, enter the **USERNAME** and **PASSWORD** of the on-premises Historian server that you created during the installation of the collector.
   If you entered the **USERNAME**and **PASSWORD**, select **Test Connection**. This will help you to test if the Historian server that you are trying to connect is valid or if the credentials that you entered are valid.
   If the entered credentials are valid, a successful connection message appears.

10. Select **Next**.
    The **Collector Initiation** section appears. The **COLLECTOR NAME** field is populated with a value in the following format: `<Historian server name>_Cygnet`
11. If needed, modify the value in the **COLLECTOR NAME** field.
    The value that you enter:
    - Must be unique.
    - Must contain the string `Cygnet`.
    - Must not exceed 15 characters.
    - Must not contain a space.
    - Must not contain special characters except a hyphen, period, and an underscore.
12. In the **RUNNING MODE** field, select one of the following options.
    - **Service - Local System Account**: Select this option if you want to run the collector as a Windows service using the credentials of the local user (that is, the currently logged-in user). By default, this option is selected, and the **USERNAME** and **PASSWORD** fields are disabled.
    - **Service Under Specific User Account**: Select this option if you want to run the collector as a Windows service using a specific user account. If you select this option, you must enter values in the **USERNAME** and **PASSWORD** fields.

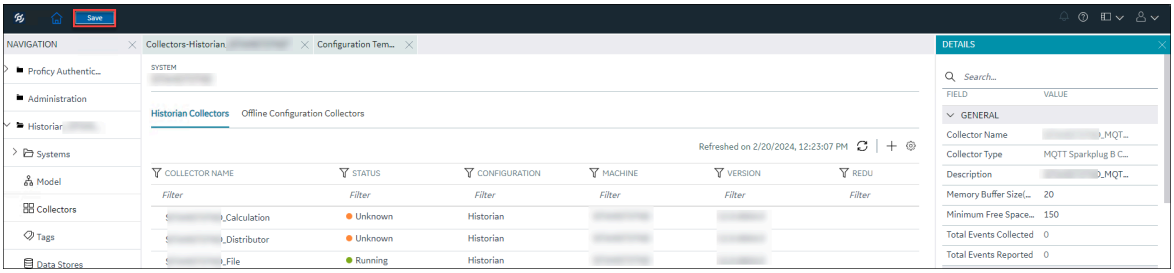
      If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:
      - iH Security Admins
      - iH Collector Admins
      - iH Tag Admins

    You can also configure the collector to start automatically when you start the computer.
13. Select **Add**.
    The collector instance is added. The fields specific to the collector section appear in the **DETAILS** section.
14. Under **COLLECTOR SPECIFIC CONFIGURATION**, configure values as described in the following table.

| Field | Description |
|---|---|
| **Recovery Time** | The maximum time, in hours, for which the collector will attempt to recover data after the collector is started or when connection between the collector and the CygNet server is re-established. This time is calculated as the duration between the current time and the last known write time. |

| Field | Description |
|---|---|
| | Continuous data collection is resumed only after the previous data has been recovered.<br><br>By default, this value is set to 0, which means data recovery is not attempted. The maximum value you can provide is 168 hours (that is, 7 days). |
| **Thread Count** | The maximum number of threads that you want to collector to use to query data from the CygNet server. |
| **CygNet Debug Mode** | The debug mode for the collector. You can enter a value between 0 and 255, where 0 turns off debugging and 255 enables detailed debugging (with query transactions).<br><br>**Note:**<br>Do not turn on debugging for a long period. If you do so, very large log files are created, which can consume a great deal of disk space. We recommend a maximum of 10 minutes. |
| **General Optimized** | Indicates whether you want to apply optimization on the tag data reads. |
| **Service-Site** | Identifies the CygNet site or data source from which the CygNet collector collects data. A value is required. |
| **MTLS Security** | Indicates whether you want to use Mutual TLS (MTLS) protocol to enforce a secure and strong authentication mechanism. |
| **MTLS Data Encryption** | Indicates whether you want to encrypt the data that the collector shares to the data archiver (DA). |
| For more information on how to enable MTLS Security, refer to Enable MTLS Security *(on page 632)*. | |

15. If needed, enter values in the other sections common to all collectors *(on page 579)*.
16. In the upper-left corner of the page, select **Save**.




The changes to the collector instance are saved.

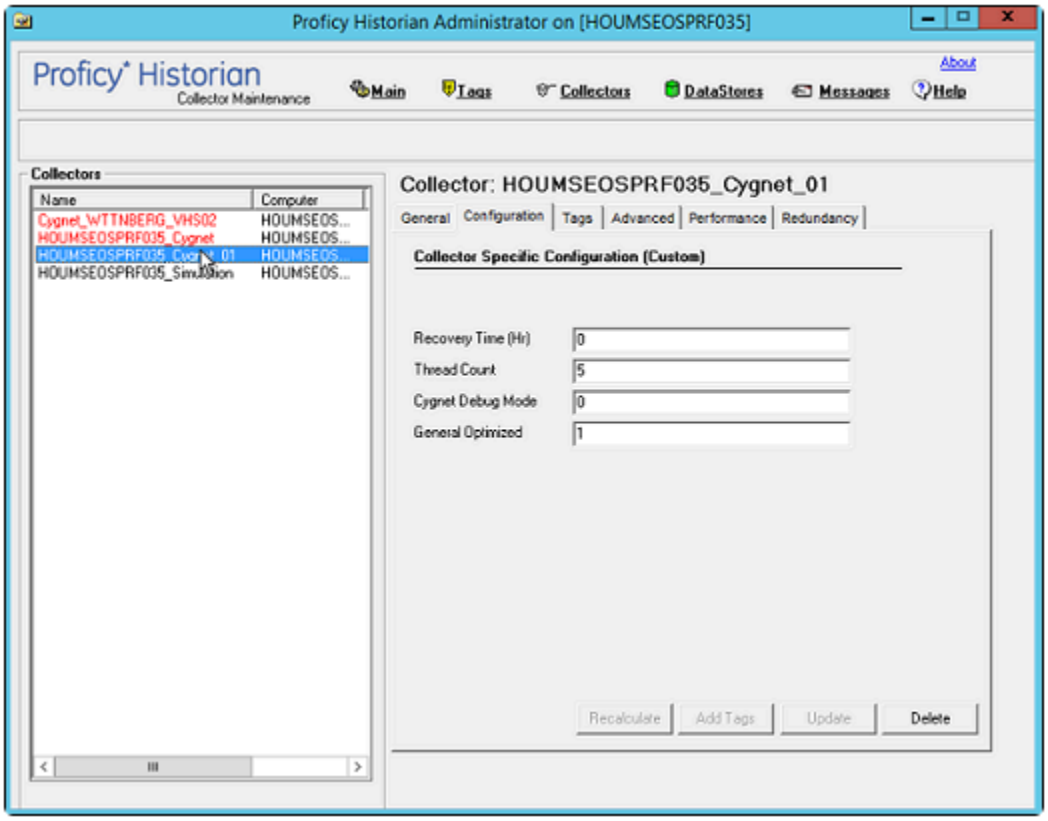
17. If needed, restart the collector.

Specify the tags *(on page 357)* whose data you want to collect using the collector.

## Configure the CygNet Collector Using Historian Administrator

1. Access Historian Administrator *(on page 823)*.
2. Select **Collectors**, and then select the CygNet collector instance that you want to configure.
3. Select **Configuration**.

   The **Configuration** section appears.




4. Enter values as specified in the following table.

| Field | Description |
|---|---|
| **Recovery Time** | The maximum time, in hours, for which the collector will attempt to recover data after the collector is started or when connection between the collector and the CygNet server is re-established. This time is calculated as the duration between the current time and the last known write time.<br><br>Continuous data collection is resumed only after the previous data has been recovered.<br><br>By default, this value is set to 0, which means data recovery is not attempted. The maximum value you can provide is 168 hours (that is, 7 days). |
| **Thread Count** | The maximum number of threads that you want to use to query data from the CygNet server. |
| **CygNet Debug Mode** | The debug mode for the collector. You can enter a value between 0 and 255, where 0 turns off debugging and 255 enables detailed debugging (with query transactions).<br><br>**Note:**<br>Do not turn on debugging for a long period. If you do so, very large log files are created, which can consume a great deal of disk space. We recommend maximum 10 minutes. |
| **General Optimized** | Indicates whether you want to apply optimization on the tag data reads. |
| **ServiceSite** | Identifies the CygNet site or data source from which the CygNet collector collects data. A value is required. |

**Note:**
You can also modify these parameters in the registry under HKEY_LOCAL_MACHINE \SOFTWARE\GE Digital\iHistorian\Services\CygNetCollector. The following table provides the mapping between the parameters in the Registry and the UI.

| Registry Parameters | Parameters in the UI |
|---|---|
| General2 | Recovery Time |
| General3 | Thread Count |
| General4 | CygNet Debug Mode |
| General5 | General Optimized |

For a list of the general parameters, refer to General Parameters of a Collector *(on page 814)*.

5. Select **Update**.
6. Restart the collector.
   The collector is configured.

# Specifying Tags for Data Collection

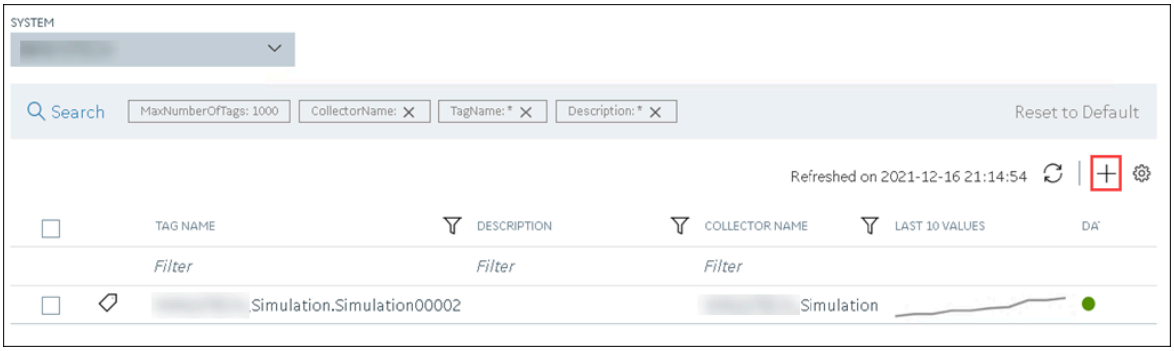
## Add Tags for the Data Store Using Configuration Hub

- Add the collector instance *(on page 357)* using which you want to collect data. Ensure that the collector is running.
- By default, the tag data is stored in the user data store, which is created automatically when you set up Configuration Hub. If, however, you want to store the data in a different data store, create it *(on page 370)*.

This topic describes how to specify the tags for which you want to collect data by browsing through the tags in the data source. For example, for an iFIX collector, if there are 1,00,000 tags in the iFIX server, you must specify the ones for which you want to collect data. Only then data is collected for those tags.

In addition to adding tags from the data source, you can create tags manually *(on page 473)*.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
3. Select +.

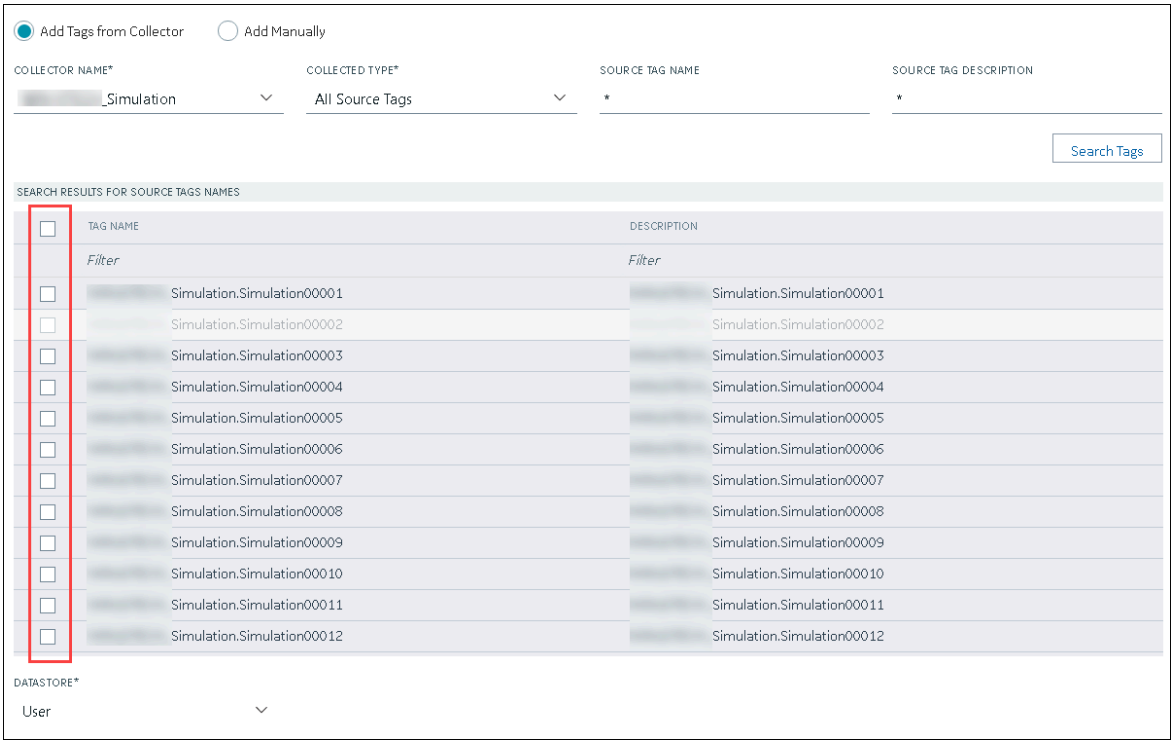



The **Add Tag-<system name>** page appears. The **Add Tags from Collector** option is selected by default.

4. Enter values as described in the following table.

| Field | Description |
|---|---|
| **COLLECTOR NAME** | Select the collector instance that you want to use to collect data. A value is required. |
| **COLLECTED TYPE** | Specify whether you want to browse through all the tags in the data source or only from the tags that you have not added yet. A value is required. |
| **SOURCE TAG NAME** | Enter the name of the tag (either completely or partially) to narrow down the search results. |
| **SOURCE TAG DESCRIPTION** | Enter the description of the tag (either completely or partially) to narrow down the search results. |

5. Select **Search Tags**.
   A list of tags that match *all* the criteria that you have specified appears. If a tag is already added, it is disabled.
6. Select the check box corresponding to each tag for which you want to collect data.




7. In the **DATA STORE** field, if you want to store the data in a different data store than the user data store, select the same.
8. Select **Add Tag**.
   Data collection begins for the selected tags.

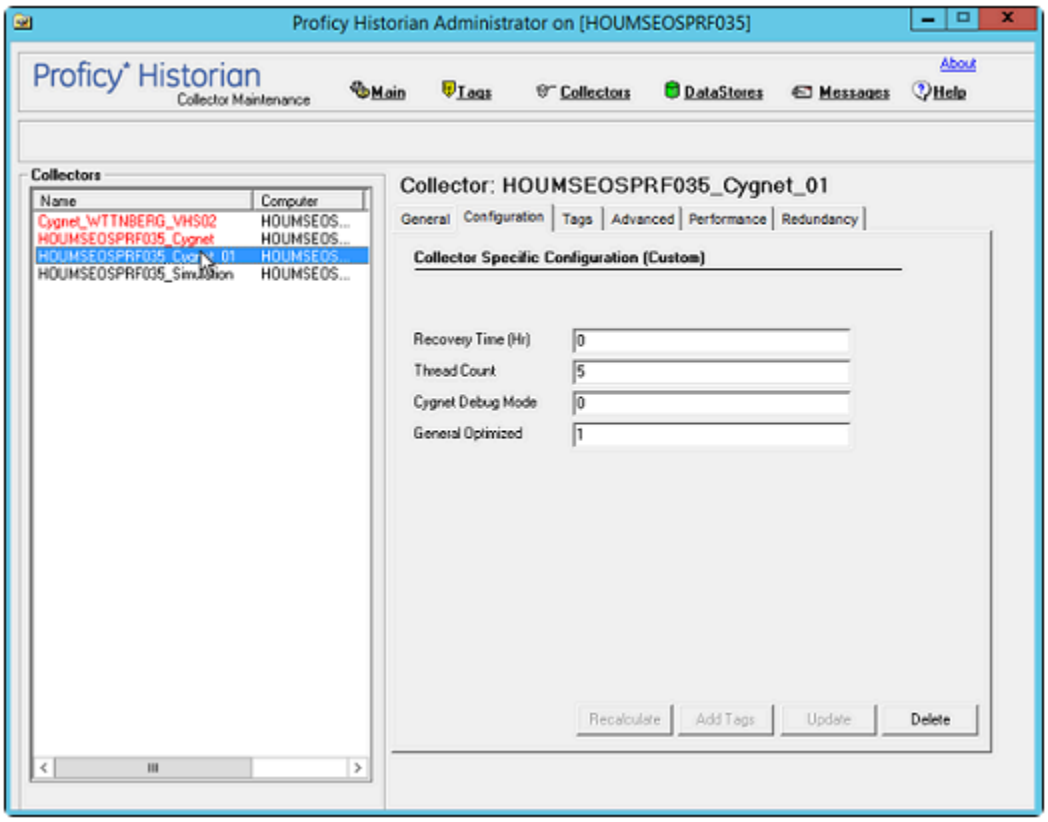
As needed, configure each tag by providing values for the tag properties. For information on the delta query modes, refer to Counter Delta Queries.

## Specify the Tags for Data Collection Using Historian Administrator

Install the CygNet ODBC driver on the collector machine, which is required for the CygNet collector to connect to the server. Note that the CygNet ODBC driver is not available with Historian.

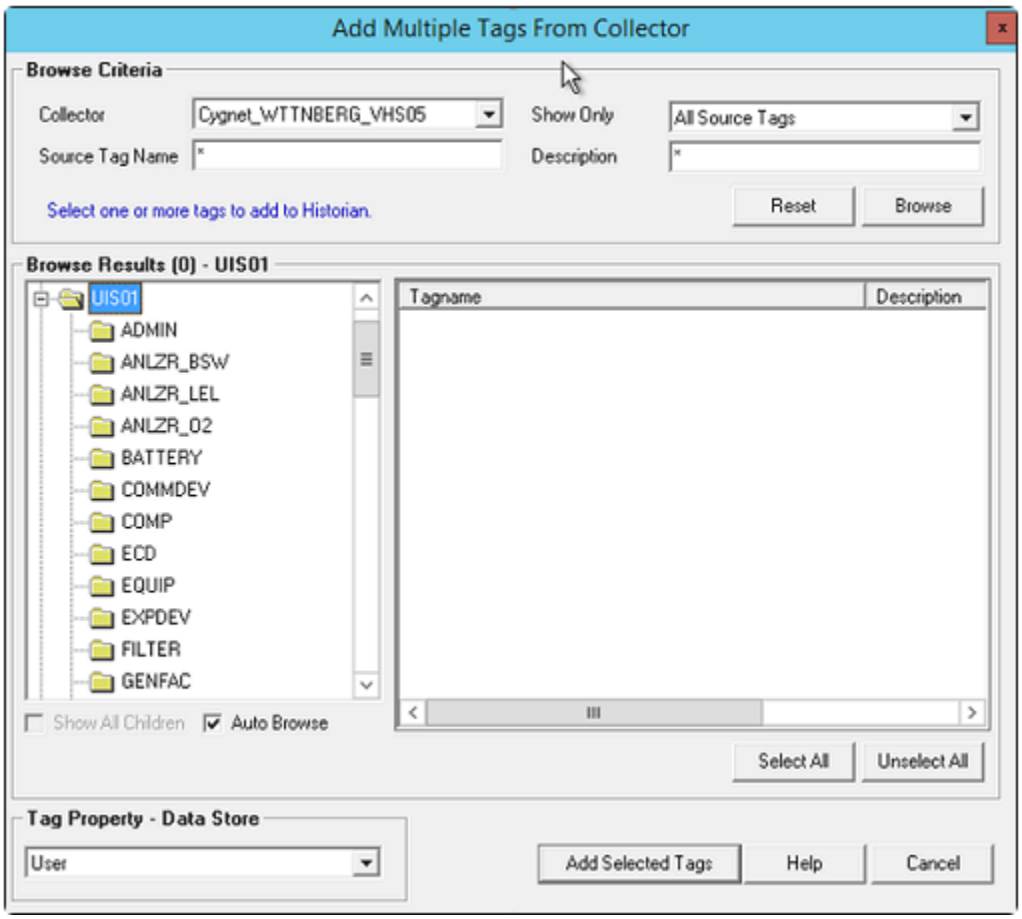
If your CygNet server supports hierarchical organization of tags in a tree structure, you can use the hierarchy to browse for tags and add them to the collector for data collection.

1. Access Historian Administrator *(on page 823)*.
2. Select **Collectors**, and then select the CygNet collector instance to which you want to add tags.
3. Select **Configuration**.
   The **Configuration** section appears.




4. Select **Add Tags**.

   The **Add Multiple Tags from Collector** window appears.




5. In the **Collector** field, select the CygNet collector to which you want to add tags.
   A hierarchical tree of tags appears in the **Browse Results** section.
6. If you want to view only the tags for which data is not collected, in the **Show Only** field, select **Source Tags Not Collected**. You can search for a tag by entering search criteria in the **Source Tag Name** or **Description** field.
7. Navigate to the node in the tree that you want to browse, and then select **Browse**.

> **Tip:**
> - To browse automatically, select the **Auto Browse** check box. The available tags appear in the **Browse Results** window whenever a node is selected in the tree.
> - To show all child elements within a hierarchy, select the **Show All Children** check box. All tags at or below the hierarchical level of the selected node in the tree appear in the **Browse Results** window.

The tags within the selected portion of the CygNet server tag hierarchy appear.

8. Select the tags for which you want to collect data, and then select **Add Selected Tags**.
   The tags are added to the collector. They appear in black text in the list of tags.

# Disable Bad Offline Values

By default, the CygNet collector adds a bad offline value when the collector is started or when a tag is modified. This topic describes how to disable this feature.

1. Access the following registry key: `HKEY_LOCAL_MACHINE\SOFTWARE\Intellution, Inc.\iHistorian\Services\CygNetCollector\<entry>`
2. Add the `WriteBadOfflineAtStartup (DWORD)` DWORD, and set it to 0 to disable adding bad offline values.
   Bad offline values will no longer be added.

# Disable Deleting Values Before Recalculation

By default, before recalculating tag values, the CygNet collectordeletes any previously stored values. This topic describes how to disable this feature.

1. Access the following registry key: `HKEY_LOCAL_MACHINE\SOFTWARE\Intellution, Inc.\iHistorian\Services\CygNetCollector\<entry>`
2. Add the `DeleteBeforeRecalc (DWORD)` DWORD, and set it to 0.
   Previously stored tag values will no longer be deleted before recalculation.

# Troubleshooting the CygNet Collector

The CygNet collector generates logs during initialization, configuration, and general operation. You can find them at *C:\Proficy Historian Data\LogFiles*.

### Troubleshooting Tips

Troubleshoot the collector by:

- Examining the log files for information.
- Examining the Windows Event Viewer for error/warnings.
- Verifying that the CygNet server is running before you start the collector.
- Verifying that CygNet ODBC Client Tools are installed.

Additional log information can be gathered by using the **CygNet Debug Mode** described above for debugging the CygNet to Collector interface.

### Turn on the Debug Mode

1. Access the following folder in the registry: *HKEY_LOCAL_MACHINE\SOFTWARE\GE Digital \iHistorian\Services\CygNetCollector\<entry>*
2. Double-click **DebugMode DWORD** .
3. Set **Base** as Decimal.
4. In the **Value data** field, enter 255.
5. Select **OK**.
6. Restart the collector.
7. Close the registry and open the *Proficy OPC Classic HDA server trace* log file.

> **Note:**
> - Do not turn on both **CygNet Debug Mode** and **DebugMode** at the same time.
> - Do not turn on **CygNet Debug Mode** or the **DebugMode key** for an extended period, 10 minutes should be more than sufficient. Leaving it on will create very large log files and that could use up a great deal of disk space.

### CygNet APIs Not Installed

**Error message:** Error Setting CygNet GlobalFunctions object: 'Class not registered'

**Root cause:** The CygNet 64 bit APIs are not installed.

**Solution:** Run the following command to install `CxScript64.dll`: `regsvr32 "C:\...\8.1.2 Install \CygNet 812\Support64\CxScript64.dll"`

# Chapter 17. The File Collector

## Overview of the File Collector

The File collector imports CSV and XML files into Historian. Since the files can contain data, tags, tag properties, and messages, the File collector is a very useful tool for importing third-party data into Historian.

**Features:**

- You can import CSV and XML files.
- The supported timestamp resolution is 1ms.
- Floating point, integer, and string data are supported. Binary data is not supported.
- The collector accepts device timestamps.
- The collector reads data, tags, and messages.
- You can set file specifications and the import interval. The timestamps for data or messages may be at intervals less than the import interval.
- You can create Historian tags.
- You can create Python Expression Tags for those collectors that support them.

Since a File collector is really an import function rather than a data collection operation, standard collector features such as compression, buffering, browsing, start/stop collection are not applicable.

**Note:**
You can import files using ANSI encoding only.

**How it works:**

The File collector uses the `ImportFiles` folder for its operations, which is available in the Historian program folder. It is created only after you run the collector. It contains the following subfolders:

| Directory | Function |
|---|---|
| `Error` | Contains the CSV and XML files that contains errors. These files will not be processed by the collector. |
| `Incoming` | Contains files that are to be processed by the collector. All the files that you want to import using the collector must be placed here. |
| `Processed` | Contains files that have been imported. |

| Directory | Function |
|---|---|
| `Working` | Contains files that the collector is importing. |

1. Place the files that you want to import in the `Incoming` folder.
2. At the beginning of each cycle, the collector processes the files, stores the result in an archive file and moves the files to the `Processed` folder.
3. While processing the files, the collector moves the files to the `Working` folder, and renames the files using the following format: YMDHMS-Data (for example, 010810103246-data.csv).
4. After processing the files, the collector restores the original file names. If an error occurs while processing the files, they are moved to the `Error` folder, and error messages are logged in the `Filecollector_YMDHMS.log` file.
5. After a specified duration has passed, the imported files are deleted from the `Processed` folder. However, files are not deleted from the `Error` folder.

**Supported date formats:**

| System Date Format | Supported Date Format in the XML Files | Supported Date Format in the CSV Files |
|---|---|---|
| dd-MMM-yy | • dd-MMM-yy<br>• yyyy-MM-dd<br>• yy/MM/dd<br>• MM/dd/yyyy<br>• MM/dd/yy<br>• M/d/yy<br>• M/d/yyyy | • yyyy-MM-dd<br>• yy/MM/dd<br>• MM/dd/yyyy<br>• MM/dd/yy<br>• M/d/yyyy |
| yyyy-MM-dd | • dd-MMM-yy<br>• yyyy-MM-dd<br>• yy/MM/dd<br>• MM/dd/yyyy<br>• MM/dd/yy<br>• M/d/yy<br>• M/d/yyyy | • dd-MMM-yy<br>• yyyy-MM-dd<br>• yy/MM/dd<br>• MM/dd/yyyy<br>• MM/dd/yy<br>• M/d/yy<br>• M/d/yyyy |
| yy/MM/dd | • dd-MMM-yy<br>• yyyy-MM-dd<br>• yy/MM/dd<br>• MM/dd/yyyy | • yyyy-MM-dd<br>• yy/MM/dd<br>• MM/dd/yyyy<br>• MM/dd/yy |

| System Date Format | Supported Date Format in the XML Files | Supported Date Format in the CSV Files |
|---|---|---|
| | • MM/dd/yy<br>• M/d/yy<br>• M/d/yyyy | • M/d/yy<br>• M/d/yyyy |
| MM/dd/yyyy | • dd-MMM-yy<br>• yyyy-MM-dd<br>• yy/MM/dd<br>• MM/dd/yyyy<br>• MM/dd/yy<br>• M/d/yy<br>• M/d/yyyy | • yyyy-MM-dd<br>• yy/MM/dd<br>• MM/dd/yyyy<br>• MM/dd/yy<br>• M/d/yy<br>• M/d/yyyy |
| MM/dd/yy | • dd-MMM-yy<br>• yyyy-MM-dd<br>• yy/MM/dd<br>• MM/dd/yyyy<br>• MM/dd/yy<br>• M/d/yy<br>• M/d/yyyy | • yyyy-MM-dd<br>• yy/MM/dd<br>• MM/dd/yyyy<br>• MM/dd/yy<br>• M/d/yy<br>• M/d/yyyy |
| M/d/yy | • dd-MMM-yy<br>• yyyy-MM-dd<br>• yy/MM/dd<br>• MM/dd/yyyy<br>• MM/dd/yy<br>• M/d/yy<br>• M/d/yyyy | • dd-MMM-yy<br>• yyyy-MM-dd<br>• yy/MM/dd<br>• MM/dd/yyyy<br>• MM/dd/yy<br>• M/d/yy<br>• M/d/yyyy |
| M/d/yyyy | • dd-MMM-yy<br>• yyyy-MM-dd<br>• yy/MM/dd<br>• MM/dd/yyyy<br>• MM/dd/yy<br>• M/d/yy<br>• M/d/yyyy | • yyyy-MM-dd<br>• yy/MM/dd<br>• MM/dd/yyyy<br>• MM/dd/yy<br>• M/d/yy<br>• M/d/yyyy |

# Configuration

## Add and Configure a File Collector

A File collector is used to send data from one Historian server to another one.

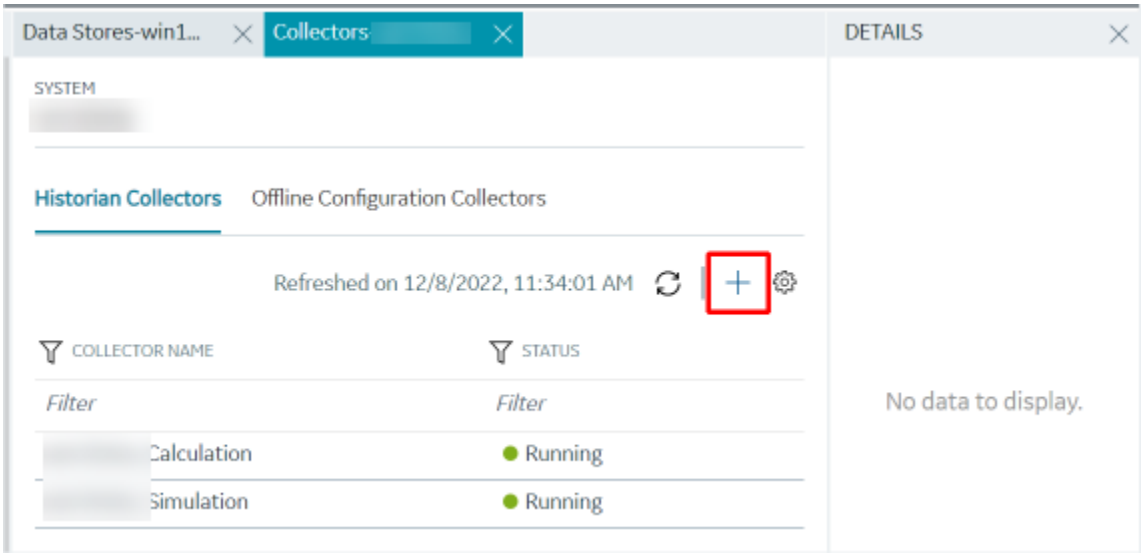
**Note:**

- You cannot send data to a cloud destination using the File collector.
- You can create only one instance of the File collector.

For more information, refer to Overview of the File Collector *(on page 2181)*.

This topic describes how to add a collector instance using Configuration Hub. You can also add a collector instance using the RemoteCollectorConfigurator utility *(on page 797)*, which does not require you to install Web-based Clients.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**. A list of collectors in the default system appears.
3. In the upper-right corner of the main section, select +.




The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.

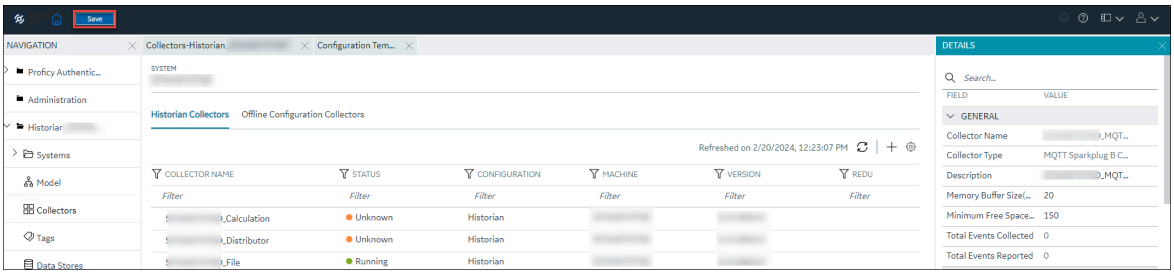
4. In the **MACHINE NAME** field, select the machine in which you want to add a collector instance. The collector will send data to this machine.
5. In the **COLLECTOR TYPE** field, select **File Collector**, and then select **Get Details**.
   The **INSTALLATION DRIVE** and **DATA DIRECTORY** fields are disabled and populated.
6. Select **Next**.
   The **Destination Configuration** section appears. Under **CHOOSE DESTINATION**, the **Historian Server** option is selected by default; the other options are disabled.
7. If you created security groups or enabled a strict client/collector authentication, enter the **USERNAME** and **PASSWORD** of the on-premises Historian server that you created during the installation of the collector.
   If you entered the **USERNAME**and **PASSWORD**, select **Test Connection**. This will help you to test if the Historian server that you are trying to connect is valid or if the credentials that you entered are valid.
   If the entered credentials are valid, a successful connection message appears.
8. Select **Next**.
   The **Collector Initiation** section appears. The **COLLECTOR NAME** field is populated with a value in the following format: `<Historian server name>_File`
9. If needed, modify the value in the **COLLECTOR NAME** field.
   The value that you enter:
   - Must be unique.
   - Must not exceed 15 characters.
   - Must not contain a space.
   - Must not contain special characters except a hyphen, period, and an underscore.
10. In the **RUNNING MODE** field, select one of the following options.
    - **Service - Local System Account**: Select this option if you want to run the collector as a Windows service using the credentials of the local user (that is, the currently logged-in user). By default, this option is selected, and the **USERNAME** and **PASSWORD** fields are disabled.
    - **Service Under Specific User Account**: Select this option if you want to run the collector as a Windows service using a specific user account. If you select this option, you must enter values in the **USERNAME** and **PASSWORD** fields. If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:
      - iH Security Admins
      - iH Collector Admins
      - iH Tag Admins

You can also configure the collector to start automatically when you start the computer.

11. Select **Add**.
    The collector instance is added. The fields specific to the collector section appear in the **DETAILS** section.
12. In the **COLLECTOR SPECIFIC CONFIGURATION** section, configure values as described in the following table.

| Field | Description |
|---|---|
| **Scan Interval** | The interval, in seconds, after which the collector initiates an import operation. The maximum value that you can enter is 65. |
| **CSV File Spec** | The file extension for a CSV file to be imported. You can specify more than one extension type, such as csv, txt, dat. |
| **XML File Spec** | The file extension for an XML file to be imported. |
| **Purge Processed (days)** | The number of days after which you want the contents of the `Processed Files` folder to be automatically purged. |
| **Purge Error (days)** | The number of days after which you want the contents of the `Error Files` folder to be automatically purged. |

13. Ad needed, enter values in the other sections common to all collectors *(on page 579)*.
14. In the upper-left corner of the page, select **Save**.




    The changes to the collector instance are saved.
15. If needed, restart the collector.

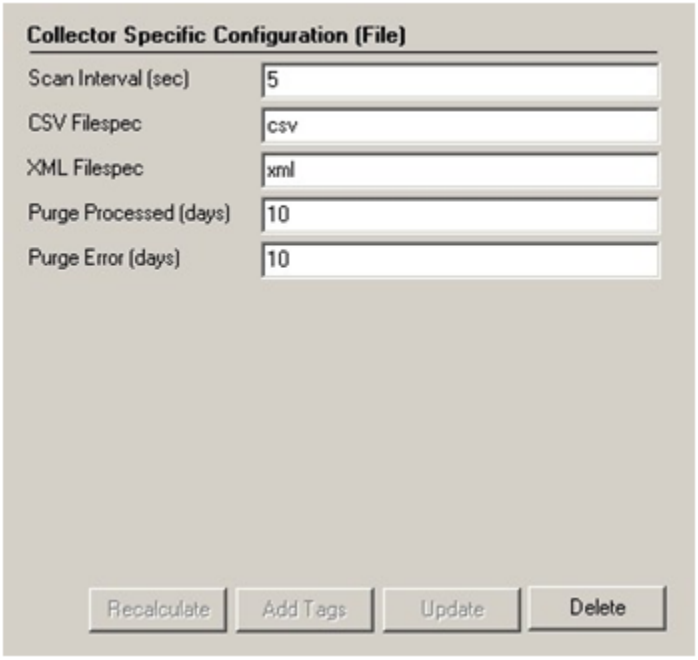
Import files using the collector *(on page 2188)*.

## Configure the File Collector Using Historian Administrator

Install collectors *(on page 143)*, and create an instance of the collector *(on page 486)*.

1. Access Historian Administrator.
2. Select **Collectors**, and then select the File collector instance that you want to configure.
3. Select **Configuration**.

   The **Configuration** section appears.




4. Enter values as specified in the following table.

| Field | Description |
|---|---|
| Scan Interval | The interval, in seconds, after which the collector initiates an import operation. The maximum value that you can enter is 65. |
| CSV File Specification | The file extension for a CSV file to be imported. You can specify more than one extension type, such as csv, txt, dat. |
| XML File Specification | The file extension for an XML file to be imported. |
| Purge Processed Files After(days) | The number of days after which you want the contents of the `Processed` folder to be automatically purged. |
| Purge Error Files After (days) | The number of days after which you want the contents of the `Error` folder to be automatically purged. |

5. Select **Update**.

6. Restart the collector.
   The collector is configured.

# Import Files

1. Configure the collector settings *(on page 486)*.
2. Ensure that the files you want to import are of the CSV format *(on page 2188)* or the XML format *(on page 2195)*.

**Note:**
You can import files using ANSI encoding only.

Place the files that you want to import in the `Incoming` folder. By default, this folder is located in the following folder: `C:/Proficy Historian Data/ImportFiles`.
The collector checks the `Incoming` folder every few seconds (based on the duration specified in the **Scan Interval** field while configuring the collector *(on page 486)*). The files are imported to Historian and are moved to the `Processed` folder. If an error occurs, however, they are moved to the `Error` folder. If that happens, refer to Troubleshooting the File Collector *(on page 2205)*.

## CSV File Format

Specify the collector source address and the tag source address fields in the CSV file. Otherwise, the collector and source names are not added to the tags created by the File collector.

The format for a CSV file is as follows:

```
[<command>]
<Header keywords separated by commas>
<values separated by commas>
*Comments
```

**Note:**

- The order of the values must match the order in the header. For example, if you are importing a tagname, timestamp, value, and quality, use the following syntax:

  ```
  [Data]
  Tagname,TimeStamp,Value,DataQuality TIGER.IMPORT_TAG1.F_CV,7/20/01 11:07,1,Good
  ```

- Data values for a single entry must be placed on a single line.
- You cannot import Last Modified User, Last Modified Date, and Calculation Execution Time fields on a tag.

Refer to the following sections for valid values and examples.

### Valid Values for Command

- [Tags]
- [Data]
- [Messages]
- [Alarms]

### Valid Values for Header Keywords

| DataQuality | Value |
|---|---|
| Tagname | Description |
| EngineeringUnits | Comment |
| DataType | StringLength<br>**Note:**<br>StringLength is twice the number of characters (ASCII, Single Byte). For example, "ABC" = 6 StringLength. |
| StoreMilliseconds | CollectorName |
| CollectorType | SourceAddress<br>**Note:**<br>A value is required for the source name to be added to tags created by the collector. |
| CollectorType | CollectionInterval |
| CollectionOffset | CollectionDisabled |
| LoadBalancing | TimeStamp |

| DataQuality | Value |
|---|---|
| | **Note:**<br>The timestamp resolution is milliseconds. |
| Type | TimeZoneBias |
| HiEngineeringUnits | LoEngineeringUnits |
| InputScaling | HiScale |
| LoScale | CollectorCompression |
| CollectorDeadband-PercentRange | CollectorCompressionTimeout |
| ArchiveCompression | ArchiveDeadbandPercentRange |
| ArchiveCompression-Timeout | Timeout |
| CollectorGeneral1 | CollectorGeneral2 |
| CollectorGeneral3 | CollectorGeneral4 |
| CollectorGeneral5 | ReadSecurityGroup |
| WriteSecurityGroup | AdministratorSecurityGroup |
| Calculation | CalculationDependencies |
| Acked | Condition |
| SubCondition | EventCategory |
| Message | Source |
| Severity | StartTime |
| EndTime | TimestampType |
| SpikeLogic | SpikeLogicOverride |
| InterfaceAbsolute-Deadband | InterfaceAbsoluteDeadbanding |
| LastModified | LastModifiedUser |

| DataQuality | Value |
|---|---|
| ArchiveAbsoluteDead-band | ArchiveAbsoluteDeadbanding |
| StepValue | Value |
| SpikeLogic | SpikeLogicOverride |
| NumberOfElements | CalcType |

For alarms and events, only the following header keywords are considered:

- Acked
- Acktime
- Actor
- AlarmID
- Condition
- DataSource
- Enabled
- EndTime
- EventCategory
- ItemID
- Message
- Quality
- Severity
- Source
- StartTime
- SubCondition
- Tagname
- Timestamp

## A CSV File that Imports Tags

```
[Tags]

Tagname,Description,DataType,HiEngineeringUnits,LoEngineeringUnits

TIGER.IMPORT_TAG1.F_CV,Import Tag 1,SingleFloat,35000,0

TIGER.IMPORT_TAG2.F_CV,Import Tag 2,SingleFloat,35000,0

TIGER.IMPORT_TAG3.F_CV,Import Tag 3,SingleFloat,35000,0

TIGER.IMPORT_TAG4.F_CV,Import Tag 4,SingleFloat,35000,0

TIGER.IMPORT_TAG5.F_CV,Import Tag 5,SingleFloat,35000,0
```

```
TIGER.IMPORT_TAG6.F_CV,Import Tag 6,SingleFloat,35000,0

TIGER.IMPORT_TAG7.F_CV,Import Tag 7,SingleFloat,35000,0

TIGER.IMPORT_TAG8.F_CV,Import Tag 8,SingleFloat,35000,0

TIGER.IMPORT_TAG9.F_CV,Import Tag 9,SingleFloat,35000,0

TIGER.IMPORT_TAG10.F_CV,Import Tag 10,SingleFloat,35000,0
```

## A CSV File that Imports Data and Data Quality

```
[Data]

Tagname,TimeStamp,Value,DataQuality

TIGER.IMPORT_TAG1.F_CV,7/20/01 11:07,1,Good

TIGER.IMPORT_TAG1.F_CV,7/20/01 11:08,2,Good

TIGER.IMPORT_TAG1.F_CV,7/20/01 11:09,3,Good

TIGER.IMPORT_TAG1.F_CV,7/20/01 11:10,4,Good

TIGER.IMPORT_TAG1.F_CV,7/20/01 11:11,5,Bad

TIGER.IMPORT_TAG1.F_CV,7/20/01 11:12,6,Bad

TIGER.IMPORT_TAG1.F_CV,7/20/01 11:13,7,Good

TIGER.IMPORT_TAG1.F_CV,7/20/01 11:14,8,Bad

TIGER.IMPORT_TAG1.F_CV,7/20/01 11:15,9,Good

TIGER.IMPORT_TAG1.F_CV,7/20/01 11:16,10,Good

TIGER.IMPORT_TAG1.F_CV,7/20/01 11:17,11,Good

TIGER.IMPORT_TAG1.F_CV,7/20/01 11:18,12,Good

TIGER.IMPORT_TAG1.F_CV,7/20/01 11:19,13,Bad

TIGER.IMPORT_TAG1.F_CV,7/20/01 11:20,14,Good

TIGER.IMPORT_TAG1.F_CV,7/20/01 11:21,15,Good

TIGER.IMPORT_TAG1.F_CV,7/20/01 11:22,16,Bad

TIGER.IMPORT_TAG1.F_CV,7/20/01 11:23,17,Good
```

## A CSV File that Imports Messages

```
[Messages]

TimeStamp,Topic,Username,MessageNumber,MessageString,Substitutions

28-Aug-2002 19:39:30.567,General,User1,0,A Test Message value 2 with milliseconds,,

28-Aug-2002 19:40:00.000,General,User1,0,A Test Message value 0,,
```

## A CSV File that Imports Tags with Step Values

```
[Tags]

Tagname,Description,DataType,HiEngineeringUnits,LoEngineeringUnits

TIGER.IMPORT_TAG1.F_CV,Import Tag 1,SingleFloat,35000,0
```

```
TIGER.IMPORT_TAG2.F_CV,Import Tag 2,SingleFloat,35000,0

TIGER.IMPORT_TAG3.F_CV,Import Tag 3,SingleFloat,35000,0

TIGER.IMPORT_TAG4.F_CV,Import Tag 4,SingleFloat,35000,0

TIGER.IMPORT_TAG5.F_CV,Import Tag 5,SingleFloat,35000,0

TIGER.IMPORT_TAG6.F_CV,Import Tag 6,SingleFloat,35000,0

TIGER.IMPORT_TAG7.F_CV,Import Tag 7,SingleFloat,35000,0

TIGER.IMPORT_TAG8.F_CV,Import Tag 8,SingleFloat,35000,0

TIGER.IMPORT_TAG9.F_CV,Import Tag 9,SingleFloat,35000,0

TIGER.IMPORT_TAG10.F_CV,Import Tag 10,SingleFloat,35000,0
```

## A CSV File that Imports Alarms

```
[Alarms]

DataSource,Condition,Source,StartTime,TimeStamp

FileCollector,HI,Mixer,3/10/2005 12:52:02,3/10/2005 12:52:02

FileCollector,HIHI,Mixer,3/10/2005 12:52:02,3/10/2005 12:59:12
```

## A CSV File that Imports Enumerated Set

```
[EnumeratedSet]

SetName,SetDescription,StateLowValue,StateHighValue,StateDescription,StateName,StateRawValueDataType,

LastModified,LastModifiedUser,AdministerSecurityGroup,NumberOfStatesInThisSet

TestSet5,TestDesc,1,10,State1Desc,State1,DoubleFloat,,,,2

TestSet5,TestDesc,11,20,State2Desc,State2,DoubleFloat,,,,2
```

## A CSV File that Imports Array Tags

```
[Tags]

Tagname,Description,DataType,HiEngineeringUnits,LoEngineeringUnits,NumberOfElements

TIGER.IMPORT_TAG1.F_CV,Import Tag 1,SingleFloat,35000,0,-1

TIGER.IMPORT_TAG2.F_CV,Import Tag 2,SingleFloat,35000,0,0

TIGER.IMPORT_TAG3.F_CV,Import Tag 3,SingleFloat,35000,0,-1

TIGER.IMPORT_TAG4.F_CV,Import Tag 4,SingleFloat,35000,0,-1
```

## A CSV File that Imports Array Tag Data

```
[Data]

Tagname,TimeStamp,Value,DataQuality

ArrayTag[0],6/11/2013 09:00:00,1,Good

ArrayTag[1],6/11/2013 09:00:00,2,Good

ArrayTag[2],6/11/2013 09:00:00,3,Good
```

```
ArrayTag[3],6/11/2013 09:00:00,4,Good

ArrayTag[0],6/11/2013 09:10:00,5,Good

ArrayTag[1],6/11/2013 09:10:00,6,Good

ArrayTag[2],6/11/2013 09:10:00,7,Good
```

## A CSV File that Imports MultiField Tag Data

```
[Data]

Tagname,TimeStamp,Value,DataQuality

MultiField.F1,05-22-2013 14:15:00,4,Good

MultiField.F1,05-22-2013 14:15:01,7,Good

MultiField.F1,05-22-2013 14:15:02,9,Good

MultiField.F2,05-22-2013 14:15:00,241,Good

MultiField.F2,05-22-2013 14:15:01,171,Good

MultiField.F2,05-22-2013 14:15:02,191,Good
```

**Note:**
Before importing MultiField tag data in this format, you must add the User Defined Type using Historian Administrator and associate that type to the MultiField tag.

## Importing s User-Defined Type

The following example allows you to import a user defined type **UDT1** with two fields, **Field1**, **Field2**. After creating, UDT1 associates with a tag called **Mfield**.

```
[UserDefinedType]

UserDefinedTypeName,UserDefinedTypeDescription,FieldName,FieldDescription,FieldDataType,IsMasterField,NumberOfFields

UDT1,UDTdesc,Field1,F1desc,SingleInteger,FALSE,2

UDT1,UDTdesc,Field2,F2desc,DoubleInteger,FALSE,2
```

```
[Tags]

Tagname,Description,DataType,UserDefinedTypeName

Mfield,mfdesc,MultiField,UDT1
```

## A CSV File that Imports Python Expression Tags

```
[Tag]

Tagname,CollectorName,CalcType,SourceAddress,DataType,Description

TagDerivedFromRawValue,SimulationCollector,PythonExpr,"{""imports"":[""math""],

""script"":""tag.value + math.pow10,tag.value/70)"",""parameters"":[{""name"":
```

```
""tag"",""source"":{""address"":""Simulation00001"",""dataType"":""SingleFloat""}}]}",

SingleFloat,Python Expression Tag example
```

**Note:**

- Python Expression tags do not support array or multi-field tags.
- It is important to include the CalcType header and set it to PythonExpr for each Python Expression tag. If the file contains a mix of tags that are Python Expression tags with those that are not, then the tags that are not Python Expression tags must have the CalcType field set to Raw.
- For Python Expression tags, the SourceAddress must contain the tag's minified JSON configuration constructed as described in the topic on Constructing the JSON Configuration for a Python Expression Tag. (Mini- fied JSON has no newline characters or comments. There are tools which can help you minify JSON.)
- Note that the example in the CSV file uses repeated quotation marks ("") in order to escape quotation marks (").
- It is important that your JSON is valid, since no validation will be performed on the JSON at tag creation.

## XML File Format

### Format to import a List of Tags

```
<Import>
    <TagList>
        <Tag>
            <Tagname> .....</Tagname>
            <Description> Test </Description>
        </Tag>
    </TagList>
</Import>
```

### Format to Import Data

```
<Import>
    <Datalist>
        <Tag>
            <Data>
```

```
                <Timestamp>..... </Timestamp>
                <Value>.... </Value>
            </Data>
        </Tag>
    </Datalist>
</Import>
```

## Format to Import Messages

```
<Import>
    <MessageList>
        <Data>
            <Timestamp>..... </Timestamp>
            <Topic>.... </Topic>
            <Username>.... </Username>
            <MessageNumber>.... </MessageNumber>
            <MessageString>.... </MessageString>
            <Substitutions>.... </Substitutions>
        </Data>
    </MessageList>
</Import>
```

The following table provides a list of tag properties that you can import:

| DataQuality | Value |
|---|---|
| Tagname | Description |
| EngineeringUnits | Comment |
| DataType | StringLength<br><br>**Note:**<br>StringLength is twice the number of characters (ASCII, Single Byte). For example, "ABC" = 6 StringLength. |
| StoreMilliseconds | CollectorName |
| CollectorType | SourceAddress |

| DataQuality | Value |
|---|---|
| | **Note:**<br>A value is required for the source name to be added to tags created by the collector. |
| CollectorType | CollectionInterval |
| CollectionOffset | CollectionDisabled |
| LoadBalancing | TimeStamp<br>**Note:**<br>The timestamp resolution is milliseconds. |
| Type | TimeZoneBias |
| HiEngineeringUnits | LoEngineeringUnits |
| InputScaling | HiScale |
| LoScale | CollectorCompression |
| CollectorDeadband-PercentRange | CollectorCompressionTimeout |
| ArchiveCompression | ArchiveDeadbandPercentRange |
| ArchiveCompression-Timeout | Timeout |
| CollectorGeneral1 | CollectorGeneral2 |
| CollectorGeneral3 | CollectorGeneral4 |
| CollectorGeneral5 | ReadSecurityGroup |
| WriteSecurityGroup | AdministratorSecurityGroup |
| Calculation | CalculationDependencies |
| Acked | Condition |
| SubCondition | EventCategory |
| Message | Source |
| Severity | StartTime |

| DataQuality | Value |
|---|---|
| EndTime | TimestampType |
| SpikeLogic | SpikeLogicOverride |
| InterfaceAbsolute-Deadband | InterfaceAbsoluteDeadbanding |
| LastModified | LastModifiedUser |
| ArchiveAbsoluteDead-band | ArchiveAbsoluteDeadbanding |
| StepValue | Value |
| SpikeLogic | SpikeLogicOverride |
| NumberOfElements | CalcType |

## An XML File that Imports Tags

```
<Import>

    <TagList Version="1.0.71">

        <Tag Name="TIGER.IMPORT_TAG1.F_CV">

            <Tagname>TIGER.IMPORT_TAG1.F_CV</Tagname>

            <Description>Import Tag 1</Description>

            <EngineeringUnits> PSI </EngineeringUnits>

        </Tag>

        <Tag Name="TIGER.IMPORT_TAG11.F_CV">

            <Tagname>TIGER.IMPORT_TAG11.F_CV</Tagname>

            <Description>Import Tag 1</Description>

            <EngineeringUnits> PSI </EngineeringUnits>

        </Tag>

        <Tag Name="TIGER.IMPORT_TAG12.F_CV">

            <Tagname>TIGER.IMPORT_TAG12.F_CV</Tagname>

            <Description>Import Tag 2</Description>

            <EngineeringUnits> PSI </EngineeringUnits>

        </Tag>

        <Tag Name="TIGER.IMPORT_TAG13.F_CV">

            <Tagname>TIGER.IMPORT_TAG13.F_CV</Tagname>

            <Description>Import Tag 3</Description>

            <EngineeringUnits> PSI </EngineeringUnits>
```

```
        </Tag>
        <Tag Name="TIGER.IMPORT_TAG14.F_CV">
            <Tagname>TIGER.IMPORT_TAG14.F_CV</Tagname>
            <Description>Import Tag 4</Description>
            <EngineeringUnits> PSI </EngineeringUnits>
        </Tag>
        <Tag Name="TIGER.IMPORT_TAG15.F_CV">
            <Tagname>TIGER.IMPORT_TAG15.F_CV</Tagname>
            <Description>Import Tag 5</Description>
            <EngineeringUnits> PSI </EngineeringUnits>
        </Tag>
        <Tag Name="TIGER.IMPORT_TAG2.F_CV">
            <Tagname>TIGER.IMPORT_TAG2.F_CV</Tagname>
            <Description>Import Tag 2</Description>
            <EngineeringUnits> PSI </EngineeringUnits>
        </Tag>
        <Tag Name="TIGER.IMPORT_TAG21.F_CV">
            <Tagname>TIGER.IMPORT_TAG21.F_CV</Tagname>
            <Description>Import Tag 1</Description>
            <EngineeringUnits> PSI </EngineeringUnits>
        </Tag>
    </TagList>
</Import>
```

## Example of an XML file that Imports Data and Data Quality

```
<Import>
    <DataList Version="1.0.71">
        <Tag Name="TIGER.IMPORT_TAG1.F_CV">
            <Data>
                <TimeStamp>20-Jul-2001 11:00:18.000</TimeStamp>
                <Value>0</Value>
                <DataQuality>Good</DataQuality>
            </Data>
            <Data>
                <TimeStamp>20-Jul-2001 11:00:36.000</TimeStamp>
                <Value>0</Value>
                <DataQuality>Good</DataQuality>
```

```
            </Data>
            <Data>
                <TimeStamp>20-Jul-2001 11:00:54.000</TimeStamp>
                <Value>0</Value>
                <DataQuality>Bad</DataQuality>
            </Data>
            <Data>
                <TimeStamp>20-Jul-2001 11:01:12.000</TimeStamp>
                <Value>0</Value>
                <DataQuality>Good</DataQuality>
            </Data>
        </Tag>
    </DataList>
</Import>
```

## An XML file that Imports Messages

```
<Import>
    <MessageList Version="1.0.71">
        <Data>
            <TimeStamp>28-Aug-2002 19:42:00.000</TimeStamp>
            <Topic>General</Topic>
            <Username>XMLUser</Username>
            <MessageNumber>0</MessageNumber>
            <MessageString>Another test message</MessageString>
            <Substitutions></Substitutions>
        </Data>
        <Data>
            <TimeStamp>28-Aug-2002 19:48:00.000</TimeStamp>
            <Topic>General</Topic>
            <Username>XMLUser</Username>
            <MessageNumber>1</MessageNumber>
            <MessageString>Message One</MessageString>
            <Substitutions></Substitutions>
        </Data>
     </MessageList>
</Import>
```

## An XML file that Imports a Tag with a Step Value

```
<Import>
    <TagList Version="1.0.71">
        <Tag Name="TAG6">
            <Tagname>TAG6</Tagname>
            <StepValue> TRUE </StepValue>
        </Tag>
    </TagList>
</Import>
```

## An XML file that Imports Alarms

```
<Import>
    <AlarmList Version="1.0.71">
        <Alarm>
            <Attribute name="Acked" value="false"/>
            <Attribute name="Actor" value="TheActor"/>
            <Attribute name="Condition" value="Condition"/>
            <Attribute name="DataSource" value="File collector"/>
            <Attribute name="Enabled" value="true"/>
            <Attribute name="EndTime" value="12/25/2005 12:47:59.003"/>
            <Attribute name="EventCategory" value="Process"/>
            <Attribute name="Message" value="My message."/>
            <Attribute name="Quality" value="Good"/>
            <Attribute name="Severity" value="250"/>
            <Attribute name="Source" value="SourceXML000003"/>
            <Attribute name="StartTime" value="12/25/2005 12:47:59.003"/>
            <Attribute name="SubCondition" value="Hi"/>
            <Attribute name="TagName" value="TheTagName"/>
            <Attribute name="Timestamp" value="12/09/2005 12:47:59.003"/>
        </Alarm>
        <Alarm>
            <Attribute name="Acked" value="false"/>
            <Attribute name="Actor" value="TheActor"/>
            <Attribute name="Condition" value="Condition"/>
            <Attribute name="DataSource" value="File collector"/>
            <Attribute name="Enabled" value="true"/>
            <Attribute name="EndTime" value="12/25/2005 12:47:59.004"/>
```

```
            <Attribute name="EventCategory" value="Process"/>
            <Attribute name="Message" value="My message."/>
            <Attribute name="Quality" value="Good"/>
            <Attribute name="Severity" value="250"/>
            <Attribute name="Source" value="SourceXML000004"/>
            <Attribute name="StartTime" value="12/25/2005 12:47:59.004"/>
            <Attribute name="SubCondition" value="Hi"/>
            <Attribute name="TagName" value="TheTagName"/>
            <Attribute name="Timestamp" value="12/25/2005 12:47:59.004"/>
        </Alarm>
    </AlarmList>
</Import>
```

## An XML file that Imports Enumerated Set

```
<Import>
    <EnumeratedSetList>
        <EnumeratedSet SetName="TestSet7">
            <EnumeratedState StateName="State1">
                <SetName>TestSet7</SetName>
                <SetDescription>TestDesc</SetDescription>
                <StateLowRawValue>1</StateLowRawValue>
                <StateHighRawValue>10</StateHighRawValue>
                <StateDescription>State1Desc</StateDescription>
                <StateName>State1</StateName>
                <StateRawValueDataType>DoubleFloat</StateRawValueDataType>
                <LastModified>9/17/2012 16:37:48.260000</LastModified>
                <LastModifiedUser>GECORPORATE\312006949</LastModifiedUser>
                <AdministratorSecurityGroup></AdministratorSecurityGroup>
                <NumberOfStatesInThisSet>2</NumberOfStatesInThisSet>
            </EnumeratedState>
            <EnumeratedState StateName="State2">
                <SetName>TestSet7</SetName>
                <SetDescription>TestDesc</SetDescription>
                <StateLowRawValue>11</StateLowRawValue>
                <StateHighRawValue>20</StateHighRawValue>
                <StateDescription>State2Desc</StateDescription>
                <StateName>State2</StateName>
```

```
                <StateRawValueDataType>DoubleFloat</StateRawValueDataType>
                <LastModified>9/17/2012 16:37:48.260000</LastModified>
                <LastModifiedUser>GECORPORATE\312006949</LastModifiedUser>
                <AdministratorSecurityGroup></AdministratorSecurityGroup>
                <NumberOfStatesInThisSet>2</NumberOfStatesInThisSet>
            </EnumeratedState>
        </EnumeratedSet>
    </EnumeratedSetList>
</Import>
```

## An XML File that Imports Array Tags

```
<Import>
    <TagList Version="1.0.71">
        <Tag Name="ArrayTag1">
            <Tagname>ArrayTag</Tagname>
            <Description>Import array Tag 1</Description>
            <EngineeringUnits> PSI </EngineeringUnits>
            <NumberOfElements>-1</NumberOfElements>
        </Tag>
        <Tag Name="ArrayTag2">
            <Tagname>ArrayTag2</Tagname>
            <Description>Import array Tag 2</Description>
            <EngineeringUnits> PSI </EngineeringUnits>
            <NumberOfElements>-1</NumberOfElements>
        </Tag>
    </TagList>
</Import>
```

## An XML File that Imports Array Tag data

```
<Import>
    <DataList Version="1.0.71">
        <Tag Name="ArrayTag[0]">
            <Data>
                <TimeStamp>11-June-2013 11:00:18.000</TimeStamp>
                <Value>1</Value>
                <DataQuality>Good</DataQuality>
            </Data>
```

```
            <Data>
                <TimeStamp>11-June-2013 11:01:18.000</TimeStamp>
                <Value>2</Value>
                <DataQuality>Good</DataQuality>
            </Data>
        </Tag>
        <Tag Name="ArrayTag[1]">
            <Data>
                <TimeStamp>11-June-2013 11:00:18.000</TimeStamp>
                <Value>3</Value>
                <DataQuality>Good</DataQuality>
            </Data>
            <Data>
                <TimeStamp>11-June-2013 11:01:18.000</TimeStamp>
                <Value>4</Value>
                <DataQuality>Good</DataQuality>
            </Data>
        </Tag>
    </DataList>
</Import>
```

## An XML File that Imports Python Expression Tags

```
<Import>
    <TagList Version="1.0.71">
        <Tag Name="TagDerivedFromRawValue">
            <Tagname>TagDerivedFromRawValue</Tagname>
            <CollectorName>OpcServerCollector</CollectorName>
            <SourceAddress>
                "{"imports":["math"],"script":"tag.value + math.pow(10,tag.value/70)","parameters":
                 [{"name":"tag","source":{"address":"IO/READONLY/Temperature","dataType":"DoubleFloat"}}]}"
            </SourceAddress>
            <CalcType>PythonExpr</CalcType>
            <DataType>DoubleFloat</DataType>
            <Description>Python Expression Tag example</Description>
        </Tag>
    </TagList>
</Import>
```

**Note:**

- Python Expression tags do not support array or multi-field tags.
- It is important to include the CalcType header and set it to PythonExpr for each Python Expression tag. If the file contains a mix of tags that are Python Expression tags with those that are not, then the tags that are not Python Expression tags must have the CalcType field set to Raw.
- For Python Expression tags, the SourceAddress must contain the tag's minified JSON configuration constructed as described in the topic on Constructing the JSON Configuration for a Python Expression Tag. (Mini- fied JSON has no newline characters or comments. There are tools which can help you minify JSON.)
- Note that the example in the CSV file uses repeated quotation marks ("") in order to escape quotation marks (").
- It is important that your JSON is valid, since no validation will be performed on the JSON at tag creation.

# Troubleshooting the File Collector

### Accessing the Log File

By default, the log file of the collector is saved in the following folder: `C:\Proficy Historian Data \LogFiles`.

If you are experiencing any problems with the collector, use the log file to troubleshoot. The log file sometimes logs errors that are not processed to Historian Administrator. For example, if you have no archives in your system and you attempt to import a .CSV file with formatting errors, the file is not processed and no alerts are sent to Historian Administrator (if there are no archives created, the message database has not been created). But this error does appear in the log file.

### Typical Error Messages and Suggested Solutions

| Error Message | Troubleshooting Tips |
|---|---|
| 12-Aug-0113:38:00 - Import Line Error: Input past end of file<br><br>12-Aug-0113:38:00 - Error Occurred On Line12: General Format Error | There is an extra line in the file. Remove it. |

| Error Message | Troubleshooting Tips |
|---|---|
| 10-Aug-01 15:47:02 - Import Line Error: Type mismatch<br><br>10-Aug-01 15:47:02 - Error Occurred On Line 2: General Format Error | Two fields were combined into one due to a missing comma. Add a comma. |
| 12-Aug-01 14:18:12 - Invalid Import Field: TimeResolution<br><br>12-Aug-01 14:18:12 - zProcessFragment>> Aborted Import Due To Formatting Errors | If an import field is invalid (in this case Time resolution should be StoreMilliseconds), the import will abort. |

## Identifying Lines in which an Error Occurs

Suppose the log file contains an error description as follows:

```
20-Jul-01 10:1717 - Import Line Error: Input past end of file

20-Jul-01 10:1717 - Error occurred on Line 7:General Format Error
```

The line number in the log is produced by the SDK, not the File collector. This means that the line number is counted relative to the header or field list for each section of the file, ignoring comments and blank lines.

## CSV File Import Issues

A CSV file may not be imported if:

- It contains an extra line.
- It contains extra commas on the data line.
- The file size is greater than 10 MB. Ideally, the file size must be between 1 and 2 MB.
- The data in the file goes beyond the Archive Active Hours value (by default, 1 month).
- The first archive is not yet created.

If you view the CSV file using Microsoft Excel, some characters are not visible. We recommend that you use a text editor to examine files that are causing format errors when you attempt to import them.

### Troubleshooting Large File Import

To prevent a locked file scenario when building large files for import into the Incoming directory, first build the file under a temporary file name or directory that will be ignored by the File collector, then rename or move the file to the real file name or Import directory when the file is fully built.

## Unable to Access Historian Administrator After Installing the File Collector

If you encounter problems in accessing Historian Administrator after installing the File collector, upgrade the Historian Server. On upgrading the File collector from 7.0 SP2\7.0 SP3\7.0 SP4 to 7.0 SP5 or later, you will notice the following:

- The previous version of the collector is in the stopped state and the new version of the collector is in the running state.
- The data collection is stopped for the tags added in the older version.

To avoid this issue:

- Change the collector to the newer version in **Tag properties** for all the tags which were added using the older version of the collector.
- Delete the older version of the collector from the **Collectors** section in Historian Administrator.

## File Collector is not available in Admin clients

If the File collector is running, but still not available in Admin clients, then you should register the following DLLs:

**32-bit machine**

- c:\windows\system32\regsvr32 ihAPICOM.dll
- c:\windows\system32\regsvr32 ihaSDK.dll

**64-bit machine**

- c:\windows\syswow64\regsvr32 ihAPICOM.dll
- c:\windows\syswow64\regsvr32 ihaSDK.dll

# Chapter 18. The HAB Collector

## Overview of the HAB Collector

The HAB collector collects data from Habitat, which is a SCADA application that contains real-time data. The collector interacts with the Habitat Sampler application to fetch data from the Habitat database records and stores the data in a Historian server.

**Features:**

- **Tags as well as alarms data collection**: The collector can collect data for tags as well as alarms. Although a single collector instance can be used for both, we recommend that you use separate collector instances for tags and alarms.
- **High availability:** You can achieve high availability by configuring redundant collector instances connected to redundant Habitat servers. For more information, refer to High Availability *(on page 2210)*.
- **Automatic tag sync:** The HAB collector creates tags automatically in Historian. When additional tags are added in Habitat, or when tags are renamed or deleted, the changes are reflected automatically in Historian. You can, however, choose to do it manually if you want to review the changes first. When you do so, only after you approve the changes, they are reflected in Historian. For more information, refer to Approve Tag Changes *(on page 2243)*.
- **Tag deletion versus disablement:** When a tag is deleted in Habitat, you can choose to delete it in Historian or disable data collection for the tag. For more information, refer to Configure the HAB Collector for Tags *(on page 2219)* and Configure the HAB Collector for Alarms *(on page 2230)*.
- **Multiple instances:** You can create multiple instances of the HAB collector either on the same machine or on different ones. For example, one instance can collect data for tags with a faster streaming rate, while the other instance can collect data for tags with a slower streaming rate. For each instance, you must create a configuration file. For more information, refer to Configure the HAB Collector for Tags *(on page 2219)* and Configure the HAB Collector for Alarms *(on page 2230)*.
- **No dependency on an external database:** Since data is stored in archive files, you do not require an external database (especially for alarms data, which is huge).
- **Supported timestamp resolution:** The supported timestamp resolution is 1 second.
- **Unsolicited as well as polled data collection:** For tags, both the polled as well as unsolicited data collection are supported. For alarms, only the unsolicited data collection is supported.
- **Supported data types:** Floating point, integer, string, and binary data are supported.
- The collector accepts device timestamps.

**How it works when automatic tag sync is enabled:**

1. You specify the site details of Habitat and collection definitions in the configuration file, and start the collector.
2. The collector connects to the Habitat Sampler application.
3. Depending on the key value that you have provided in the configuration file, tags are created in Historian with the following naming convention: *<host name of the Habitat server>.<composite key>.<field name in Habitat>*
4. Data collection begins.
5. When tags are added, renamed, or deleted later in Habitat, the changes are reflected automatically in Historian. No manual steps are required.

**How it works when automatic tag sync is disabled:**

1. You specify the site details of Habitat and collection definitions in the configuration file, and start the collector.
2. The collector connects to the Habitat Sampler application.
3. Depending on the key value that you have provided in the configuration file, tags in Habitat are listed in the `<collector name>_Tag_Unconfirmed.xml` file under the ProposedTags section. These tags use the following naming convention: *<host name of the Habitat server>.<composite key>.<field name in Habitat>*
4. You approve the tags *(on page 2243)* that must be created in Historian.
5. The tags that you approve are created in Historian.
6. Data collection begins.
7. When tags are added, renamed, or deleted later in Habitat, you must again approve these changes *(on page 2243)*. Only then they are reflected in Historian.

The tags created in Historian contain the MRID and composite key values of the corresponding Habitat record. Although MRID is constant for a tag, the composite key value can change if there is a change in the transmission/distribution lines. Therefore, the collector uses the MRID value to identify whether a tag is a newly added one or an old one that has been renamed. And the composite key is used in naming the tags created in Historian.

**Supported data types:**

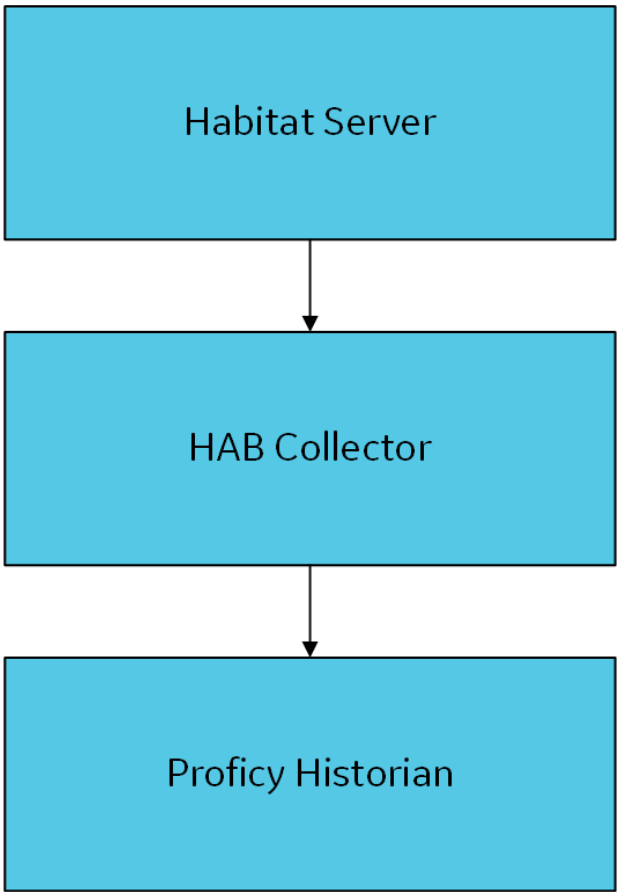
- Float
- Integer
- String
- Boolean

## High Availability

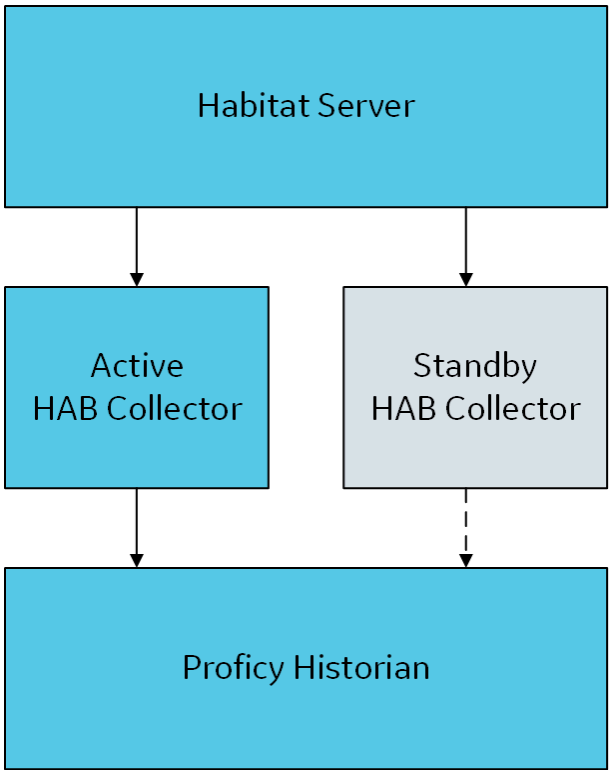
High availability of the Habitat server and the HAB collector provides uninterrupted data collection. You can configure a stand-by Habitat server and/or a second instance of the HAB collector to achieve high availability.

You can choose one of the following types of configuration:

- **Single collector instance and single Habitat server:** In this configuration, a single HAB collector instance collects data from a single Habitat server.

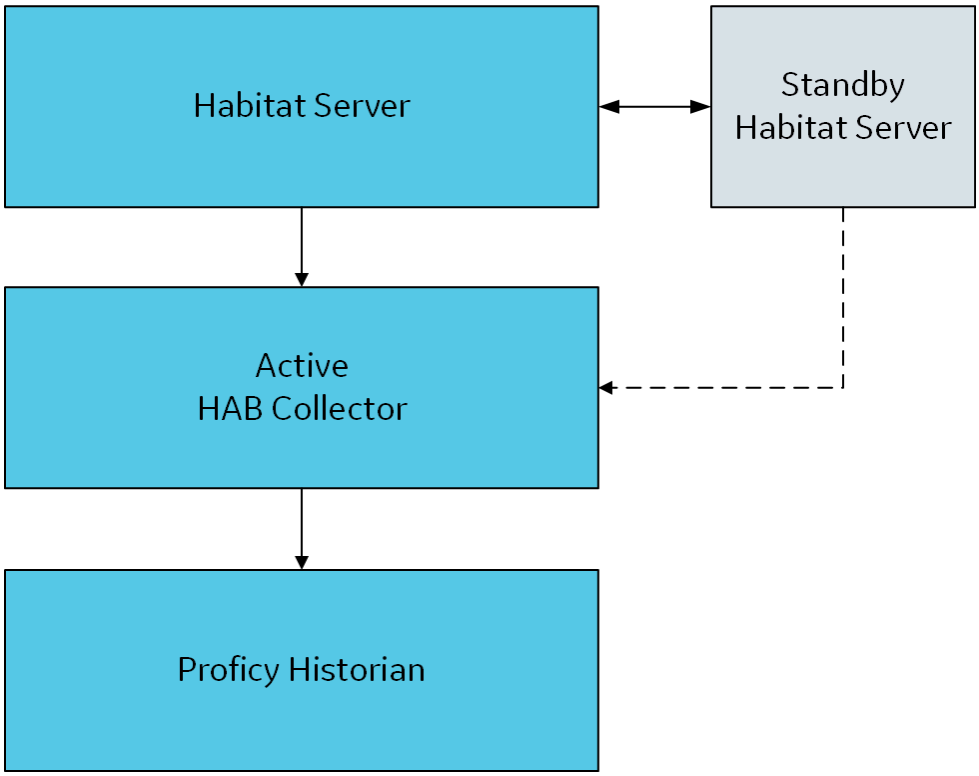



  To set up this configuration, create a collector instance *(on page 489)*, and provide the details of the Habitat site *(on page 2219)* in the configuration file of the collector instance.
- **Redundant collector instances and single Habitat server:** In this configuration, a single Habitat server is connected to two collector instances - one active and the other one standby. If the active collector instance is not available, data is collected by the standby collector instance.



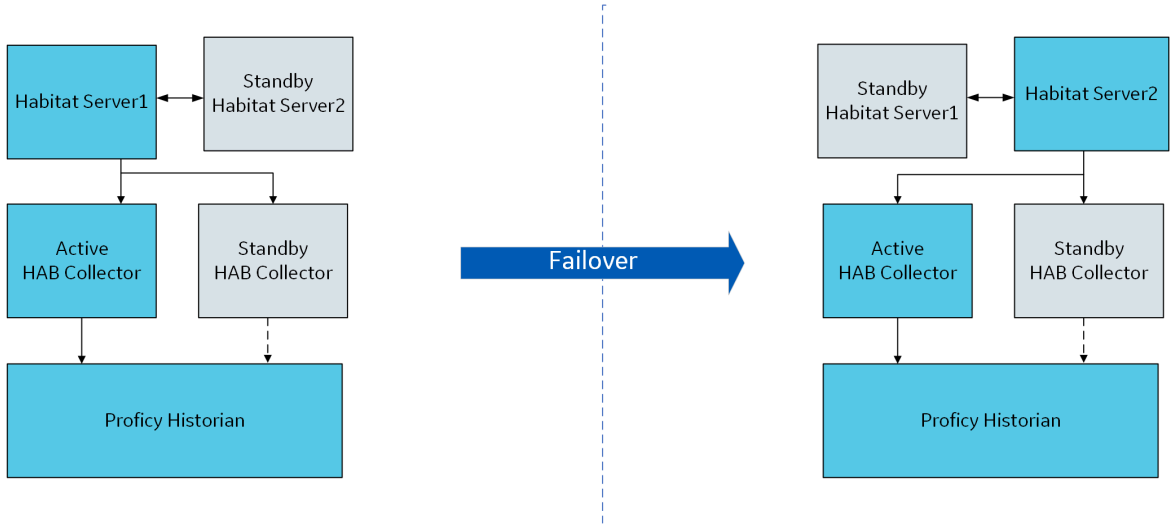

To set up this configuration, create two instances of the collector *(on page 489)*, and provide and same site details *(on page 2219)* for both of them.

- **Single collector instance and redundant Habitat servers:** In this configuration, a single collector instance is connected to two Habitat servers - one active and the other one is standby. If the active server is down, the standby server accepts requests from the collector and sends data.




To set up this configuration, create a collector instance *(on page 489)*, and provide the details *(on page 2219)* of the active and standby Habitat servers in the Node1 and Node2 parameters respectively in the configuration file.

- **Redundant collector instances and redundant Habitat servers:** In this configuration, two collector instances are connected to two Habitat servers. The active collector instance collects data from the active Habitat server. If the active collect instance is not available, data is collected by the standby collector instance. If the active Habitat server is down, the standby server accepts requests from the active collector instance.



To set up this configuration:

1. Create two instances of the collector *(on page 489)*.
2. For both the instances, provide the details *(on page 2219)* of the active and standby Habitat servers in the Node1 and Node2 parameters respectively in the configuration file.

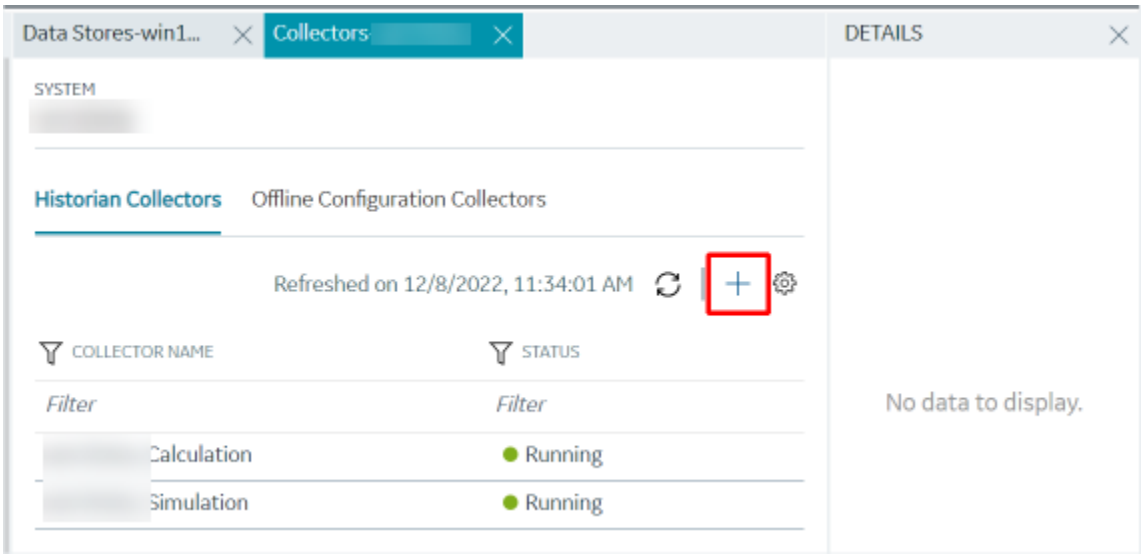
In addition to the high availability options provided by the collector and the Habitat server, you can set up data mirroring in the Historian server. For information, refer to About Data Mirroring *(on page 367)*. Or you can install Historian in a cluster environment *(on page 119)*.

# Configuration

## Add and Configure a HAB Collector

The HAB collector collects data from Habitat, which is a SCADA application that contains real-time data. The collector interacts with the Habitat Sampler application to fetch data from the Habitat database records and stores the data in a Historian server. For more information, refer to Overview of the HAB Collector *(on page 2208)*.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**. A list of collectors in the default system appears.
3. On the right, next to Settings in the main section, select +.



The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.

4. In the **MACHINE NAME** field, select the machine in which you want to add a collector instance.
5. In the **COLLECTOR TYPE** field, select **Hab Collector**, and then select **Get Details**.
   The **INSTALLATION DRIVE** and **DATA DIRECTORY** fields are disabled and populated.
6. Select **Next**.
   The **Source Configuration** section appears.
7. Enter values as described in the following table.

| Field | Description |
|---|---|
| **SERVER SITE** | Enter name that you want to assign to the site. A value is required and must be unique. It is used by Habitat to identify the collector instance. By default, this field is populated with a value in the following format: `<Historian server name>Hab` |
| **SERVER 1** (under NODE 1) | Enter the host name or IP address of the Habitat server in the primary site from which you want to collect data. This server acts as the primary/active server from which the collector receives data. A value is required. |
| **SERVER 1** (under NODE 2) | Enter the host name or IP address of the Habitat server in the second/backup site from which you want to collect data. This server acts as a standby server in case server 1 under node 1 fails. A value is required. If you do not have a secondary/backup site, enter the same value as **SERVER 1** under node 1. |
| **SERVER 2** (under NODE 1) | Enter the host name or IP address of the Habitat server that you want to use as a standby server in the same site as server 1. This server acts as a standby server in case server 1 under node 2 fails. |
| **SERVER 2** (under NODE 2) | Enter the host name or IP address of the Habitat server in the secondary/backup site from which you want to collect data. This server acts as a standby server in case server 2 under node 1 fails.<br><br>For example, suppose Machine A and Machine B are in node AB, and Machine X and Machine Y are in node XY. Suppose you |

| Field | Description |
|---|---|
| | want to use Machine A as the primary server and the remaining machines as standby servers. In that case, enter values as follows:<br>◦ **SERVER 1** (under Node 1): `Machine A`<br>◦ **SERVER 2** (under Node 1): `Machine B`<br>◦ **SERVER 1** (under Node 2): `Machine X`<br>◦ **SERVER 2** (under Node 2): `Machine Y`<br>If Machine A fails, the Machine B becomes active. If Machine B fails, Machine X becomes active. If Machine X fails, Machine Y becomes active. |
| **SOCKET** | The socket number (port number) used by the Habitat Sampler application to connect. Each collector instance can connect to only one socket. The default value is 8040. |
| **RETRY** | The duration, in seconds, after which the collector tries to communicate with the site. The default value is 5 seconds. |

8. Select **Next**.
   The **Destination Configuration** section appears. Under **CHOOSE DESTINATION**, the **Historian Server** option is selected by default. The other options are disabled because you cannot send data to a cloud destination using the HAB collector.
9. If you created security groups or enabled a strict client/collector authentication, enter the **USERNAME** and **PASSWORD** of the on-premises Historian server that you created during the installation of the collector.
   If you entered the **USERNAME**and **PASSWORD**, select **Test Connection**. This will help you to test if the Historian server that you are trying to connect is valid or if the credentials that you entered are valid.
   If the entered credentials are valid, a successful connection message appears.
10. Select **Next**.
    The **Collector Initiation** section appears. By default, the **COLLECTOR NAME** field is populated with a value in the following format: `<Historian server name>_Hab`
11. If needed, modify the value in the **COLLECTOR NAME** field.
    The value that you enter:
    ◦ Must be unique.
    ◦ Must not exceed 15 characters.

- Must not contain a space.
- Must not contain special characters except a hyphen, period, and an underscore.

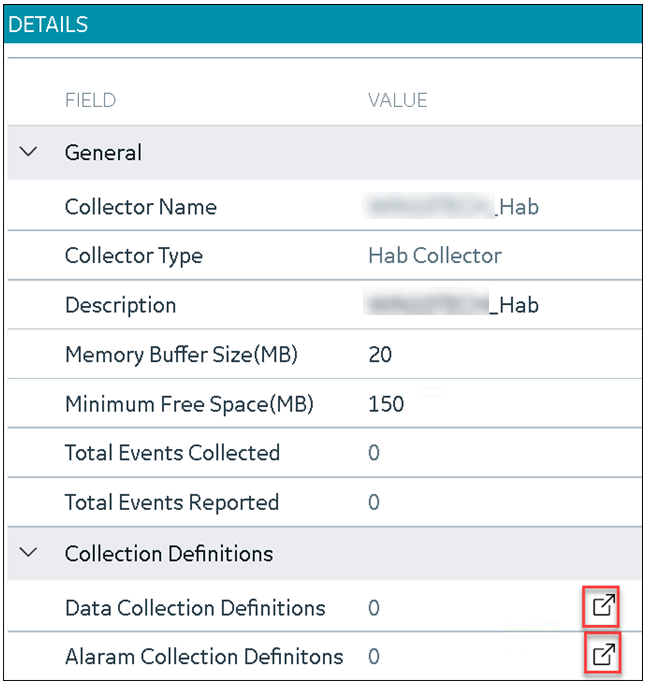
12. In the **SamplerID** field, enter the user ID to connect to Habitat Sampler.

    By default, this field contains the Collector Name. You can first provide the Collector Name and update the Sample ID field as this filed will automatically takes the Collector Name.

13. In the **RUNNING MODE** field, select one of the following options.
    - **Service - Local System Account**: Select this option if you want to run the collector as a Windows service using the credentials of the local user (that is, the currently logged-in user). If you select this option, the **USERNAME** and **PASSWORD** fields are disabled. By default, this option is selected.
    - **Service Under Specific User Account**: Select this option if you want to run the collector as a Windows service using a specific user account. If you select this option, you must enter values in the **USERNAME** and **PASSWORD** fields.

      If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:
      - iH Security Admins
      - iH Collector Admins
      - iH Tag Admins

14. Select **Add**.
    The collector instance is added. The fields specific to the collector appear in the **DETAILS** section.
15. Depending on whether you want to configure tags or alarms, select next to the corresponding field under **Collection Definitions**.



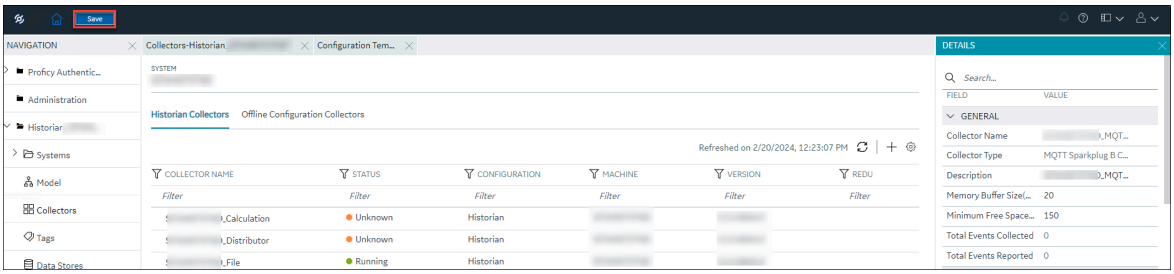

The **Data Collections** or **Alarm Collections** section appears.

16. Select +, and then enter values in the available fields for data collection and/or alarm collection *(on page 495)*. You can also copy a collection definition by right-clicking it and selecting **Duplicate**.
17. Under **COLLECTOR SPECIFIC CONFIGURATION**, configure values as described in the following table.

| Field | Description |
|---|---|
| **Auto Tag Sync** | If you enable this option, the collector creates tags automatically in Historian based on the key value. In addition, any tag changes in Habitat (such as adding, renaming, and deleting tags) will reflect automatically in Historian. No manual steps are required.<br><br>If you disable this option, any tag changes in Habitat will be captured in the `<collector name>_Tag_Unconfirmed.xml` file. Only after you approve these changes *(on page 2243)*, they are reflected in Historian. |
| **Tag Deletion Type** | Specify whether deleted tags in Habitat that you have approved must be deleted or disabled for data collection in Historian. Enter one of the following values: |

| Field | Description |
|---|---|
| | ◦ DISABLE_TAG (this is the default value)<br>◦ DELETE_TAG |
| **MTLS Security** | Indicates whether you want to use Mutual TLS (MTLS) protocol to enforce a secure and strong authentication mechanism. |
| **MTLS Data Encryption** | Indicates whether you want to encrypt the data that the collector shares to the data archiver (DA). |
| For more information on how to enable MTLS Security, refer to Enable MTLS Security *(on page 632)*. | |

18. As needed, provide values in the the other sections common to all collectors *(on page 579)*.
19. In the upper-left corner of the page, select **Save**.




    The changes to the collector instance are saved.
20. Start the collector *(on page 637)*.

If you have disabled the automatic tag sync option in the configuration file, tag changes in Habitat (such as adding, renaming, and deleting tags) are captured in the `<collector_name>_tag.xml` file. You must approve the changes so that they are reflected in Historian. To do so:

1. Right-click the HAB collector instance that you have created, and then select **Confirm Queued Tags**.

    A list of tags that have been changed appears.
2. Select the check boxes corresponding to the tags whose changes you want to approve, and then select **Confirm**. You can filter the list using the **TAG TYPE** field.

    The tag changes are approved. The status of the tags is updated in the `<collector_name>_tag.xml` file (that is, the Confirmed parameter is set to true).

## Configure the HAB Collector for Tags

Create an instance of the HAB collector *(on page 489)*.

A configuration file is created when you install collectors. This file contains the following types of information, which are required for the Sampler application to connect to Habitat.

- **Site parameters:** Contains the location details of Habitat from which you want to collect data. A single site can contain one or two nodes (the two nodes are used for a primary/standby configuration).

  A configuration file can contain only one Site section.

- **Collector parameters:** Contains the following parameters:
  - **Automatic tag sync:** If the automatic tag sync option is enabled in the configuration file, when tags are added, renamed, or deleted in Habitat, the changes are reflected automatically in Historian. No manual steps are required; data collection begins automatically depending on the collection definition that you have provided in the configuration file. If, however, the automatic tag sync in disabled, you must approve tag changes, and only then they are reflected in Historian. For more information, refer to Approve Tag Changes *(on page 2243)*. Depending on the collection definition that you have provided in the configuration file, tags are created in Historian, and data collection is initiated.
  - **Tag deletion versus disablement:** For tags deleted in Habitat, you must specify whether they must be deleted in Historian or just disabled for data collection.

  A configuration file can contain only one CollectorParameter section.
- **Collection definition parameters:** Specifies which data to collect and how. For example, the Habitat application, database, and the family from which you want to collect data, whether the data is polled or unsolicited (that is, periodic or exception), the mapping between tag properties and the corresponding fields in Habitat, etc. There are two types of collection definition parameters: tag data collection and alarm data collection.

  A configuration file can contain multiple DataCollectionDefinition sections. If you do not want to use a collection definition for now, you can set the State parameter to DISABLED.

In addition to tags, you can specify values for these parameters for alarms. We recommend that you create a separate collector instance to collect alarm data. You must then configure alarms *(on page 2230)* in a separate file.

1. Access the `<collector name>_Config.xml` file. By default, it is located at `C:\Program Files\GE Digital\Historian HAB Collector\Server`.
2. Enter values as specified in the following tables.

**Table 375. Site Parameters**

| Parameter | Description |
|---|---|
| Name | The name that you want to assign to the site. |
| Node1 | The host name or IP address of the Habitat server (node) in the site from which you want to collect data. This server acts as the primary/active server from which the collector receives data. |
| Node2 | If you want to achieve high availability by connecting to redundant Habitat servers, provide the host names or IP addresses of both the servers - the active server in Node1 and the standby server in Node2.<br><br>**Note:**<br>You can provide the IP addresses of up to four Habitat servers - two of them in Node1 and other two in Node2, separated by commas.<br><br>For example, if you want to connect to MachineA, MachineB, MachineC, and MachineD, enter:<br>`<Node1>MachineA,MachineB</Node1>`<br>`<Noade2>MachineC,MachineD</Node2>`<br>If you do so, MachineA becomes the active Habitat server, and in case of a failover, MachineB, MachineC, and MachineD each become active servers - one-by-one, in that order.<br><br>For information on the various ways of achieving high availability, refer to High Availability *(on page 2210)*. |
| Socket | The socket number (port number) used by the Habitat Sampler application to connect. Each collector instance can connect to only one socket. The default value is 8040. |

| Parameter | Description |
|---|---|
| Retry | The duration, in seconds, after which the collector tries to communicate with the site. The default value is 5. |

**Example:**

```
<Site>
    <Name>Site1</Name>
    <Node1>Host1</Node1>
    <Node2>Host2</Node2>
    <Socket>8040</Socket>
    <Retry>5</Retry>
</Site>
```

**Table 376. Collector Parameters**

| Parameter | Description | Valid Values |
|---|---|---|
| AutoTagSync | Indicates whether tags must be created initially when the collector is connected to the Habitat Sampler application. It also indicates whether tags must be created, renamed, or deleted later automatically when the corresponding changes are done in Habitat.<br><br>If you disable this option, you must manually confirm these changes. For example, if 10 tags are added in Habitat, you must confirm which of them must be created in Historian. For instructions, refer to Approve Tag Changes *(on page 2243)*. | ◦ TRUE<br>◦ FALSE |
| TagDeletionType | Specifies whether tags deleted in Habitat must be deleted or just disabled for data collection in Historian. | ◦ DISABLE_TAG<br>◦ DELETE_TAG |

**Example:**

```
<CollectorParameter>
    <AutoTagSync>TRUE</AutoTagSync>
    <TagDeletionType>DISABLE_TAG</TagDeletionType>
</CollectorParameter>
```

**Table 377. Collection Definition Parameters**

| Parameter | Description | Valid Values |
|---|---|---|
| Name | The name of the collection definition. A value is required and must be unique.<br><br>**Important:**<br>You must not change the name after the collector starts collecting data. If, however, you want to change the name after the collector has started collecting data, refer to FAQs on HAB Collector *(on page 2246)*. | |
| id | Identifies the collection definition. A value is required and must be unique.<br><br>**Important:**<br>You must not change the ID after the collector starts collecting data. If, however, you want to change the ID after the collector has started collecting data, refer to FAQs on HAB Collector *(on page 2246)*. | |
| Site | The name of the site in Habitat from which you want to collect data. Provide the same value as the one you provided for the Name parameter in the Site section. | |

| Parameter | Description | Valid Values |
|---|---|---|
| Valid | Indicates whether the collection definition is valid. | ◦ **TRUE**: Indicates that the collection definition is valid.<br>◦ **FALSE**: Indicates that an error has occurred while processing the collection definition. In this case, the collector also sets the State parameter to DISABLED so that this collection definition is not considered. |
| State | Indicates whether the collection definition is enabled. | ◦ **ENABLED**: Indicates that the collector is processing the collection definition.<br>◦ **DISABLED**: Indicates that this data collection is not being used. You can manually disable a collection definition. Or if an error has occurred while processing the collection definition, the collector automatically sets this parameter to DISABLED. |
| Application | The name of the Habitat application from which you want to collect data (for example, SCADA). | |
| Database | The name of the Habitat database from which you want to collect data (for example, SCADAMOM). | |
| Family | The name of the Habitat family from which you want to collect data (for example, EMS, DTS). A value is required and must be unique. | |

| Parameter | Description | Valid Values |
|---|---|---|
| RecordType | The name of the HDB record from which you want to collect data (for example, ANALOG,POINT,COUNT). | Any record type that contains the composite key and MRID fields (because the collector uses these two fields to create tags). |
| CollectionType | Indicates whether you want to perform polled or unsolicited data collection. | ◦ **Polled**: Indicates a periodic data collection, where data is collected at a regular time interval (indicated as PERIODIC in Habitat).<br>◦ **Unsolicited**: Indicates that data is collected only when values have changed since the last time data was collected (indicated as EXCEPTION in Habitat). |
| Key | The value that will be used to filter tags for data collection. You can use the wildcard character * to get a range of values. For example, if you want to collect data from all tags that begin with SUBSTN.LAKEVIEW, enter: SUBSTN.LAKEVIE*.*.*.*.* | |
| TagPrefix | The prefix that you want to use for tags. You can provide a different value for each collection definition, which helps you identify tags based on the collection definition. | |
| TagNameField | The composite key of the record. A value is required. If, however, the composite key is not available, enter any other unique identifier of the record. | |
| MRIDField | The MRID of the record. A value is required. If, however, MRID is not avail- | |

| Parameter | Description | Valid Values |
|---|---|---|
| | able, enter the composite key or any other unique identifier of the record. | |
| TagDescField | The description of the tag. A value is required. | |
| ValueFields | The list of fields in Habitat that contain tag values that you want to collect. A value is required.<br><br>For example, if tag values are stored in the DIS_ANALOG field in Habitat, respectively, enter:<br><br>`<TagTemplate>`<br>`<TagPrefix>GENTAG</TagPrefix>`<br>`<TagNameField>$KEY_PLC</TagNameField>`<br>`<MRID>MRID_PLC</MRID>`<br>`<ValueFields>DIS_ANALOG</value>`<br>`<Timestamp>SCTIME_ANALOG</Timestamp>`<br>`<Quality>DFLAGS_ANALOG</Quality>`<br>`</TagTemplate>`<br><br>For the Value parameter, you can enter multiple values, separated by commas. The values will then be concatenated in the corresponding Historian tag (for example, `<Value>DIS_ANALOG,ESTIMATE_ANALOG SCTIME_ANALOG,DFLAGS_ANALOG</Value>`).<br><br>For the Timestamp parameter, you can use a HDB timestamp or custom/alias timestamps.<br><br>For analog, you can use: | |

| Parameter | Description | Valid Values |
|---|---|---|
| | ◦ **FIELDTIME**: A combination of FLDTIME_ANALOG and FLD-MSEC_ANALOG, in the hour: min:sec: millisec format.<br>◦ **SCADATIME**: Used to capture SCTIME_ANALOG in the hour: min:sec: millisec format.<br>◦ **SAMPLETIME**: The time at which the data sample was collected in Habitat in the hour: min:sec format.<br><br>For point, you can use:<br>◦ **FIELDTIME**: A combination of FLDTIME_POINT and FLD-MSEC_POINT, in the hour: min:sec: millisec format.<br>◦ **SCADATIME**: Used to capture SCTIME_POINT in the hour: min:sec: millisec format.<br>◦ **SAMPLETIME**: The time at which the data sample was collected in Habitat in the hour: min:sec format.<br><br>For count, you can use:<br>◦ **FIELDTIME**: A combination of FLDTIME_COUNT and FLD-MSEC_COUNT, in the hour: min:sec: millisec format.<br>◦ **SAMPLETIME**: The time at which the data sample was collected in Habitat in the hour: min:sec format.<br><br>You can edit the names of these custom/alias timestamps using the reg- | |

| Parameter | Description | Valid Values |
|---|---|---|
| | istry entries FieldTimeCustomField-Name, ScadaTimeCustomFieldName, and SampleTimeCustomFieldName respectively. | |
| TimestampFields | The name of the property that stores the timestamp of a tag in Habitat. You can use a HDB timestamp or custom/alias timestamps.<br><br>**Examples:**<br><br>For analog, you can use:<br>◦ **FIELDTIME**: A combination of FLDTIME_ANALOG and FLD-MSEC_ANALOG, in the hour: min:sec: millisec format.<br>◦ **SCADATIME**: Used to capture SCTIME_ANALOG in the hour: min:sec: millisec format.<br>◦ **SAMPLETIME**: The time at which the data sample was collected in Habitat in the hour: min:sec format.<br>For point, you can use:<br>◦ **FIELDTIME**: A combination of FLDTIME_POINT and FLD-MSEC_POINT, in the hour: min:sec: millisec format.<br>◦ **SCADATIME**: Used to capture SCTIME_POINT in the hour: min:sec: millisec format.<br>◦ **SAMPLETIME**: The time at which the data sample was collected in Habitat in the hour: min:sec format.<br>For count, you can use: | |

| Parameter | Description | Valid Values |
|---|---|---|
| | ◦ **FIELDTIME**: A combination of FLDTIME_COUNT and FLD-MSEC_COUNT, in the hour: min:sec: millisec format.<br>◦ **SAMPLETIME**: The time at which the data sample was collected in Habitat in the hour: min:sec format.<br>You can edit the names of these custom/alias timestamps using the registry entries FieldTimeCustomFieldName, ScadaTimeCustomFieldName, and SampleTimeCustomFieldName respectively. | |
| QualityFields | The name of the property that stores the tag quality in Habitat. | |
| SampleOptions | Contains the following elements:<br>◦ **SampleRate:** The rate at which you want to collect data.<br>◦ **SampleUnit:** The unit of measurement for the sample rate.<br>◦ **Permanent:** Indicates whether you want to store the data in buffer files in Habitat in the event of a connection loss. We strongly recommend that you set this parameter to true to prevent loss of data.<br>For example, if you want to collect data every 5 seconds, enter:<br>`<SampleOptions>`<br>`<SampleRate>5</SampleRate>`<br>`<SampleUnit>sec</SampleUnit>`<br>`<Permanent>TRUE</Permanent>`<br>`</SampleOptions>` | Valid values for SampleUnit:<br>◦ sec<br>◦ min<br>◦ hour<br>◦ day<br>◦ week<br>◦ month<br>Valid values for Permanent:<br>◦ TRUE<br>◦ FALSE |

**Example:**

```
<DataCollectionDefinitions>
    <DataCollectionDefinition>
        <Name>ANALOG_HIST</Name>
        <id>1</id>
        <Site>Site1</Site>
        <Valid>TRUE</Valid>
        <Status>DISABLED</Status>
        <Family>DTS1</Family>
        <Application>SCADA1</Application>
        <Database>SCADAMOM1</Database>
        <RecordType>ANALOG</RecordType>
        <CollectionType>Unsolicited</CollectionType>
        <Key>*</Key>
        <TagTemplate>
            <TagPrefix>Collection_1</TagPrefix>
            <TagNameField>$KEY_PLC</TagNameField>
            <MRID>MRID_PLC</MRID>
            <ValueFields>DIS_ANALOG</value>
            <Timestamp>SCTIME_ANALOG</Timestamp>
            <Quality>DFLAGS_ANALOG</Quality>
        </TagTemplate>
        <SampleOptions>
            <SampleRate>5</SampleRate>
            <SampleUnit>sec</SampleUnit>
            <Permanent>FALSE</Permanent>
        </SampleOptions>
    </DataCollectionDefinition>
    <DataCollectionDefinition>
        <Name>ANALOG_HIST</Name>
        <id>2</id>
        <Site>Site1</Site>
        <Valid>TRUE</Valid>
        <Status>ENABLED</Status>
        <Family>DTS2</Family>
        <Application>SCADA2</Application>
```

```
        <Database>SCADAMOM2</Database>
        <RecordType>ANALOG</RecordType>
        <CollectionType>Polled</CollectionType>
        <Key>*</Key>
        <TagTemplate>
            <TagPrefix>Collection_2</TagPrefix>
            <TagNameField>$KEY_PLC</TagNameField>
            <MRID>MRID_PLC</MRID>
            <ValueFields>DIS_ANALOG</value>
            <Timestamp>SCTIME_ANALOG</Timestamp>
            <Quality>DFLAGS_ANALOG</Quality>
        </TagTemplate>
        <SampleOptions>
            <SampleRate>5</SampleRate>
            <SampleUnit>sec</SampleUnit>
            <Permanent>TRUE</Permanent>
        </SampleOptions>
    </DataCollectionDefinition>
</DataCollectionDefinitions>
```

3. Save and close the file.
   The HAB collector is configured.

Start the collector *(on page 2242)*.

## Configure the HAB Collector for Alarms

Create an instance of the HAB collector *(on page 489)*.

A configuration file is created when you install collectors. This file contains the following types of information, which are required for the Sampler application to connect to Habitat.

- **Site parameters:** Contains the location details of Habitat from which you want to collect data. A single site can contain one or two nodes (the two nodes are used for a primary/standby configuration).

  A configuration file can contain only one Site section.
- **Collector parameters:** Contains the following parameters:

  - **Automatic tag sync:** If the automatic tag sync option is enabled in the configuration file, when tags are added, renamed, or deleted in Habitat, the changes are reflected automatically in Historian. No manual steps are required; data collection begins automatically depending on the collection definition that you have provided in the configuration file. If, however, the automatic tag sync in disabled, you must approve tag changes, and only then they are reflected in Historian. For more information, refer to Approve Tag Changes *(on page 2243)*. Depending on the collection definition that you have provided in the configuration file, tags are created in Historian, and data collection is initiated.
  - **Tag deletion versus disablement:** For tags deleted in Habitat, you must specify whether they must be deleted in Historian or just disabled for data collection.

  A configuration file can contain only one CollectorParameter section.
- **Collection definition parameters:** Specifies which data to collect and how. For example, the Habitat application, database, and the family from which you want to collect data, whether the data is polled or unsolicited (that is, periodic or exception), the mapping between tag properties and the corresponding fields in Habitat, etc. There are two types of collection definition parameters: tag data collection and alarm data collection.

  A configuration file can contain multiple DataCollectionDefinition sections. If you do not want to use a collection definition for now, you can set the State parameter to DISABLED.

If you have created multiple instances of the collector, you must create one configuration file for each instance.

In addition to alarms, you can specify values for these parameters for tags. We recommend that you create a separate collector instance to collect tag data. You must then configure tags *(on page 2219)* in a separate file.

1. Access the `<collector name>_Config.xml` file. By default, it is located at `C:\Program Files\GE Digital\Historian HAB Collector\Server`.
2. Enter values as specified in the following tables.

**Table 378. Site Parameters**

| Parameter | Description |
|---|---|
| Name | The name that you want to assign to the site. |
| Node1 | The host name or IP address of the Habitat server (node) in the site from which you want to collect data. This server acts as the primary/active server from which the collector receives data. |

| Parameter | Description |
|---|---|
| Node2 | If you want to achieve high availability by connecting to redundant Habitat servers, provide the host names or IP addresses of both the servers - the active server in Node1 and the standby server in Node2.<br><br>**Note:**<br>You can provide the IP addresses of up to four Habitat servers - two of them in Node1 and other two in Node2, separated by commas.<br><br>For example, if you want to connect to MachineA, MachineB, MachineC, and MachineD, enter:<br>`<Node1>MachineA,MachineB</Node1>`<br>`<Noade2>MachineC,MachineD</Node2>`<br><br>If you do so, MachineA becomes the active Habitat server, and in case of a failover, MachineB, MachineC, and MachineD each become active servers - one-by-one, in that order.<br><br>For information on the various ways of achieving high availability, refer to High Availability *(on page 2210)*. |
| Socket | The socket number (port number) used by the Habitat Sampler application to connect. Each collector instance can connect to only one socket. |
| Retry | The duration, in seconds, after which the collector tries to communicate with the site. The default value is 5. |

**Example:**

```
<Site>
    <Name>Site1</Name>
    <Node1>Host1</Node1>
    <Node2>Host2</Node2>
    <Socket>8040</Socket>
```

```
    <Retry>5</Retry>
</Site>
```

**Table 379. Collector Parameters**

| Parameter | Description | Valid Values |
|---|---|---|
| AutoTagSync | Indicates whether tags must be created initially when the collector is connected to the Habitat Sampler application. It also indicates whether tags must be created, renamed, or deleted later automatically when the corresponding changes are done in Habitat.<br><br>If you disable this option, you must manually confirm these changes. For example, if 10 tags are added in Habitat, you must confirm which of them must be created in Historian. For instructions, refer to Approve Tag Changes *(on page 2243)*. | ◦ TRUE<br>◦ FALSE |
| TagDeletionType | Specifies whether tags deleted in Habitat must be deleted or just disabled for data collection in Historian. | ◦ DISABLE_TAG<br>◦ DELETE_TAG |

**Example:**

```
<CollectorParameter>
    <AutoTagSync>TRUE</AutoTagSync>
    <TagDeletionType>DISABLE_TAG</TagDeletionType>
</CollectorParameter>
```

**Table 380. Collection Definition Parameters**

| Parameter | Description | Valid Values |
|---|---|---|
| Name | The name of the collection definition. A value is required and must be unique.<br><br>**Important:**<br>You must not change the name after the col- | |

| Parameter | Description | Valid Values |
|---|---|---|
| | lector starts collecting data. If, however, you want to change the name after the collector has started collecting data, refer to FAQs on HAB Collector *(on page 2246)*. | |
| id | Identifies the collection definition. A value is required and must be unique.<br><br>**Important:**<br>You must not change the ID after the collector starts collecting data. If, however, you want to change the ID after the collector has started collecting data, refer to FAQs on HAB Collector *(on page 2246)*. | |
| Site | The name of the site in Habitat from which you want to collect data. Provide the same value as the one you provided for the Name parameter in the Site section. | |
| Valid | Indicates whether the collection definition is valid. | ◦ **TRUE**: Indicates that the collection definition is valid.<br>◦ **FALSE**: Indicates that an error has occurred while |

| Parameter | Description | Valid Values |
|---|---|---|
| | | processing the collection definition. In this case, the collector also sets the State parameter to DISABLED so that this collection definition is not considered. |
| Status | Indicates whether the collection definition is enabled. | ◦ **ENABLED**: Indicates that the collector is processing the collection definition.<br>◦ **DISABLED**: Indicates that this data collection is not being used. You can manually disable a collection definition. Or if an error has occurred while processing the collection definition, the collector automatically sets this parameter to DISABLED. |
| Application | The name of the Habitat application from which you want to collect data (for example, ALARM). | |
| Database | The name of the Habitat database from which you want to collect data (for example, ALARMLST). | |
| Family | The name of the Habitat family from which you want to collect data (for example, EMS, DTS). | |

| Parameter | Description | Valid Values |
|---|---|---|
| | A value is required and must be unique. | |
| RecordType | The name of the HDB record from which you want to collect data (for example, CIRCLG). | Any record type that contains the composite key and MRID fields (because the collector uses these two fields to create tags). |
| CollectionType | Indicates the collection type. For alarms, only unsolicited data collection is supported, which indicates that data is collected only when values have changed since the last time data was collected. | Unsolicited (indicated as EXCEPTION in Habitat). |
| Key | This parameter is not applicable to alarms. Enter *. | |
| AlarmFilter | The list of options to filter alarm data. It contains the following parameters:<br>◦ **Enabled**: Enter TRUE or FALSE to specify whether you want to filter alarm data. The default value is TRUE.<br>◦ **Location, Area, Category, Priority, Exception**: For each of these parameters, enter the criteria using which you want to filter alarm data. You can enter multiple values separated by commas (for example, LAKEVIEW,RICHVIEW). | |

| Parameter | Description | Valid Values |
|---|---|---|
| | The default value is *, which indicates that data for that parameter is not filtered.<br>For example, if you want to retrieve only the alarm data related to LAKEVIEW and RICHVIEW, in the area Australia, with priority 3, enter:<br>`<AlarmFilter>`<br>`<Enabled>True</Enabled>`<br>`<Location>LAKEVIEW,RICHVIEW</Location>`<br>`<Area>Australia</Area>`<br>`<Category>SCADATOP</Category>`<br>`<Priority>3</Priority>`<br>`<Exception>*</Exception>`<br>`</AlarmFilter>`<br>Note that data is filtered only if the criteria for *all* the parameters is satisfied.<br>**Important:**<br>The property names of the parameters for which you want to filter data must be included in the <Value> parameter under <TagTemplate>. For example, if the location, area, and priority are stored in the LOC_CIRCLG, AREA_- | |

| Parameter | Description | Valid Values |
|---|---|---|
| | CIRCLG, and PRIOR_-CIRCLG fields respectively, enter `<Value>LOC_CIRCLG,AREA_-CIRCLG,PRIOR_CIR-CLG</Value>` | |
| TagNameFields | Determines how the tags created in Historian must be named.<br><br>For example, if you want the tags to be named after the value in the LOC_CIRCLG field, enter `<TagNameFields>LOC_CIR-CLG</TagNameFields>`. When you do so, if the value in the LOC_CIRCLG field in Habitat is DOUGLAS, the tag created in Historian will be named <TagPrefix value>.<AlarmPrefix value>.DOUGLAS.<br><br>You can enter multiple values separated by commas.<br><br>For example, if you want the tags to be named after the values in the LOC_CIRCLG and PRIOR_CIRCLG fields, enter:<br>`<TagNameFields>LOC_CIRCLG,PRIOR_CIRCLG</TagNameFields>`<br>When you do so, if the values in the LOC_CIRCLG and PRIOR_CIRCLG fields are Douglas and 1 respectively, the tag created in Historian will be named <TagPre- | |

| Parameter | Description | Valid Values |
|---|---|---|
| | fix value>.<AlarmPrefix value>.DOUGLAS.1. | |
| TagPrefix | The prefix that you want to use for tags. You can provide a different value for each collection definition, which helps you identify tags based on the collection definition. | |
| TagTemplate | The list of tag properties for which you want to collect data. In Historian, we use the Value tag property.<br><br>You must map these properties with the corresponding ones in Habitat. For example, if the value of a tag is stored in the TEXT_CIRCLG field in Habitat, enter:<br><br>`<TagTemplate>`<br>`<Value>TEXT_CIRCLG</value>`<br>`<Timestamp></Timestamp>`<br>`<Quality></Quality>`<br>`</TagTemplate>`<br><br>For the Value parameter, if you want to collect data from multiple fields, enter them separated by commas. The values will then be concatenated in the corresponding Historian tag (for example, `<Value>TEXT_CIRCLG, PRIOR_CIRCLG, TIME_CIRCLG</Value>`). | |

| Parameter | Description | Valid Values |
|---|---|---|
| | **Important:**<br>◦ Quality is not applicable to alarms. In addition, to provide the timestamp column name, include it in the <Value> element itself separated by a comma. Therefore, do NOT enter values for the <Timestamp> and <Quality> elements, but retain the elements.<br>◦ If you have provided filter criteria under <AlarmFilter>, enter the names of those fields in the <Value> parameter. | |
| SampleOptions | Contains the following elements: | Valid values for SampleUnit:<br>◦ sec<br>◦ min<br>◦ hour |

| Parameter | Description | Valid Values |
|---|---|---|
| | ◦ **SampleRate:** The rate at which you want to collect data.<br>◦ **SampleUnit:** The unit of measurement for the sample rate.<br>◦ **Permanent:** Indicates whether you want to store the data in buffer files in Habitat in the event of a connection loss. We strongly recommend that you set this parameter to true to prevent loss of data.<br>For example, if you want to collect data every 5 seconds, enter:<br>`<SampleOptions>`<br>`<SampleRate>5</SampleRate>`<br>`<SampleUnit>sec</SampleUnit>`<br>`<Permanent>TRUE</Permanent>`<br>`</SampleOptions>` | ◦ day<br>◦ week<br>◦ month<br>Valid values for Permanent:<br>◦ TRUE<br>◦ FALSE |

**Example:**

```
<AlarmCollectionDefinitions>
    <AlarmCollectionDefinition>
        <Name>ALARM</Name>
        <id>1</id>
        <Site> Site1</Site>
        <Valid>TRUE</Valid>
        <Status>ENABLED</Status>
        <Family>DTS</Family>
        <Application>ALARM</Application>
        <Database>ALARMLST</Database>
```

```
        <RecordType>CIRCLG</RecordType>
        <CollectionType>Unsolicited</CollectionType>
        <Key>*</Key>
        <AlarmFilter>
            <Enabled>True</Enabled>
            <Location>LAKEVIEW,RICHVIEW</Location>
            <Area>Australia</Area>
            <Category>SCADATOP</Category>
            <Priority>3</Priority>
            <Exception>*</Exception>
        </AlarmFilter>
        <TagNameFields>LOC_CIRCLG</TagNameFields>
        <TagPrefix>ALARM.</TagPrefix>
        <TagTemplate>
      <Value>TEXT_CIRCLG,PRIOR_CIRCLG,LOC_CIRCLG,AREA_CIRCLG</Value>
      <Timestamp>SCADATIME</Timestamp>
      <Quality></Quality>
        </TagTemplate>
        <SampleOptions>
            <SampleRate>5</SampleRate>
            <SampleUnit>sec</SampleUnit>
            <Permanent>FALSE</Permanent>
        </SampleOptions>
    </AlarmCollectionDefinition>
</AlarmCollectionDefinitions>
```

3. Save and close the file.
   The HAB collector is configured.

Start the collector *(on page 2242)*.

## Start the HAB Collector

Configure the collector for tags *(on page 2219)* and alarms *(on page 2230)*.

1. Access Windows services.
2. Right-click the HAB collector instance that you want to start, and then select **Start**.
   The collector is started, and fetches data from the Habitat site details that you provided in the configuration file.

If you have disabled the automatic tag sync option in the configuration file, tag changes in Habitat (such as adding, renaming, and deleting tags) are captured in the `<collector_name>_tag_Unconfirmed.xml` file. Approve the changes *(on page 2243)* so that they are reflected in Historian.

> **Note:**
> If you have enabled the automatic tag sync option, tag changes are automatically reflected in Historian; no manual steps are needed.

## Approve Tag Changes

Start the collector *(on page 2242)*.

After you create a collector instance and configure it, you can choose between the following options:

1. **Automatic tag sync:** If you enable this option in the configuration file, the collector creates tags automatically in Historian based on the key value. In addition, any tag changes in Habitat (such as adding, renaming, and deleting tags) will reflect automatically in Historian. No manual steps are required.
2. **Manual tag changes:** If you disable the automatic tag sync option, any tag changes in Habitat will be captured in the `<collector name>_Tag_Unconfirmed.xml` file. Only after you approve these changes, they are reflected in Historian.

This topic describes how to approve tag changes manually if you have disabled the automatic tag sync option. You can do so using Configuration Hub or by updating the `<collector name>_Tag_Unconfirmed.xml` directly.

1. To approve changes using Configuration Hub:
    a. Access Configuration Hub *(on page 336)*.
    b. In the **NAVIGATION** section, select **Collectors**.
    c. Right-click the HAB collector whose changes you want to approve, and then select **Confirm Queued Tags**.
    A list of tags that have been changed appears.

d. Select the check boxes corresponding to the tags whose changes you want to approve, and then select **Confirm**.
The tag changes are approved. The status of the tags is updated in the `<collector_name>_tag.xml` file (that is, the Confirmed parameter is set to true).

2. To approve changes using the `<collector name>_Tag_Unconfirmed.xml` file:

a. Access the `<collector name>_Tag_Unconfirmed.xml` file. By default, it is located at `C:\Program Files\GE Digital\Historian HAB Collector\Server`.

> **Note:**
> This file is named `<collector name>_Tag_Unconfirmed.xml` only if there are changes for you to approve. Otherwise, it is named `<collector name>_Tag.xml`, which indicates that there have been no changes since your last approval.

The file contains a list of tags, each tag listed under one of the following sections:

- **ProposedTags:** Contains a list of tags in Habitat that are supposed to be created in Historian. They will be created after your approval.
- **DeletedTags:** Contains a list of tags deleted in Habitat, but are not yet deleted/ disabled in Historian.
- **RenamedTags:** Contains a list of tags that have been renamed in Habitat, but still contain the old names in Historian.

b. For each tag for which you want the changes to reflect in Historian, set the Confirmed parameter to true.
For example, the following code sample indicates that a tag named SUBSTN.LAKEVIEW.VOLTAGE is created in Habitat, but is not yet created in Historian:

```
<ProposedTags>
    <TAG>
        <TagName value="SUBSTN.LAKEVIEW.VOLTAGE"/>
        <CompositeKey value="SUBSTN.LAKEVIEW.T1"/>
        <MRID value="D3FR67H-F453-4859-SHDKRIBNSJ345"/>
        <SourceAddress value="D3FR67H-F453-4859-SHFURNSOV4853J">
        <CollectionType value="Unsolicited"/>
        <Confirmed value="false">
```

```
    </TAG>
</ProposedTags>
```

To create this tag in Historian, set the Confirmed value to true as follows:

```
<ProposedTags>
    <TAG>
        <TagName value="SUBSTN.LAKEVIEW.VOLTAGE"/>
        <CompositeKey value="SUBSTN.LAKEVIEW.T1"/>
        <MRID value="D3FR67H-F453-4859-SHDKRIBNSJ345"/>
        <SourceAddress value="D3FR67H-F453-4859-SHFURNSOV4853J">
        <CollectionType value="Unsolicited"/>
        <Confirmed value="true">
    </TAG>
</ProposedTags>
```

c. Save and close the file.

d. Rename the <*collector name*>_Tag_Unconfirmed.xml file to <*collector name*>_Tag.xml.

The changes are reflected in Historian:

- Newly added tags that you have approved are created in Historian and data collection in initiated for them. They are removed from the ProposedTags section in the file.
- Renamed tags that you have approved are renamed in Historian. They are removed from the RenamedTags section in the file.
- Deleted tags that you have approved are either deleted or disabled for data collection depending on the value for the TagDeletionType parameter in the configuration file. For more information, refer to Configure the HAB Collector for Tags *(on page 2219)* and Configure the HAB Collector for Alarms *(on page 2230)*. These tags are removed from the DeletedTags section of the <*collector name*>_Tag.xml file.

## Delete an Instance of the HAB Collector

Stop the collector from Windows services.

1. Open command prompt.
2. Change to the directory in which the CollectorInstanceUtility file is located. By default, it is located in the following folder: C:\Program Files\GE Digital\Historian HAB Collector\Server

3. Run the following command:

```
CollectorInstanceUtility.exe -Delete <collector instance name>
```

where *<collector instance name>* is the name assigned to the collector instance.

For example:

```
CollectorInstanceUtility.exe -Delete Hab1 MyHistServer "C:\Proficy Historian Data"
```

The collector instance is deleted.

## FAQs on HAB Collector

### How to change the name or ID of a collection definition after data collection begins?

Suppose you have provided the following information in the configuration file:

```
<Name>ANALOG_HIST</Name>

<id>1</id>

<Key>SUBSTN.KIRKLAND.*.*.*.*</Key>
```

Suppose you want to change the name and ID as follows:

```
<Name>ANALOG_HIST_NEW</Name>

<id>3</id>

<Key>SUBSTN.KIRKLAND.*.*.*.*</Key>
```

To make these changes:

1. Stop the collector instance:
    a. Access Windows services.
    b. Right-click the HAB collector instance that you want to stop, and then select **Stop**.
2. Update the configuration file *(on page 2219)*.
3. Search and select all the tags related to the collection definition, and remove them *(on page 2063)*.

> **Important:**
> Do not delete them permanently; just remove them so that the tags and their data will still be available.

4. Start the collector instance.
    a. Access Windows services.
    b. Right-click the HAB collector instance that you want to start, and then select **Start**.

Tags will be created based on the new collection definition details. You can also retrieve data for the older tags.

### How to split a collection definition into two?

Suppose you have provided the following information in the configuration file:

```
<Name>ANALOG_HIST</Name>

<id>1</id>

<Key>SUBSTN.KIRKLAND.*.*.*.*</Key>
```

Suppose you want to split the collection definition into two as follows:

```
<Name>ANALOG_HIST</Name>

<id>1</id>

<Key>SUBSTN.KIRKLAND.BUS.*.*.*</Key>
```

```
<Name>ANALOG_HIST_HIST2</Name>

<id>2</id>

<Key>SUBSTN.KIRKLAND.LN.*.*.*</Key>
```

To make these changes:

1. Stop the collector instance:
    a. Access Windows services.
    b. Right-click the HAB collector instance that you want to stop, and then select **Stop**.
2. Update the configuration file *(on page 2219)*.
3. Search and select all the tags related to the collection definition, and remove them *(on page 2063)*.

    **Important:**
    Do not delete them permanently; just remove them so that the tags and their data will still be available.

4. Start the collector instance.
    a. Access Windows services.
    b. Right-click the HAB collector instance that you want to start, and then select **Start**.

Tags will be created based on the new collection definition details. You can also retrieve data for the older tags.

### Can I add an alarm filter in a collection definition after data collection begins?

Yes.

Suppose you have the following collection definition:

```
<Name>ALARM</Name>
<id>1</id>
<Site>Site1</Site>
<Valid>TRUE</Valid>
<Status>ENABLED</Status>
<Family>EMS</Family>
<Application>ALARM</Application>
<Database>ALARMLST</Database>
<RecordType>CIRCLG</RecordType>
<CollectionType>Unsolicited</CollectionType>
<Key>*</Key>
<TagNameFields>LOC_CIRCLG</TagNameFields>
<TagPrefix>ALARM.</TagPrefix>
<AlarmFilter>
    <Enabled>TRUE</Enabled>
    <Location>*</Location>
    <Area>*</Area>
    <Category>*</Category>
    <Priority>*</Priority>
    <Exception>*</Exception>
</AlarmFilter>
...
```

Since you have not specified any location, area, etc. in the alarmfilter parameter, tags will be created for all the locations, areas, etc. (that is, no filter is applied).

Suppose you want to filter out the tags for the location Kirkland.

To do so, include the location as follows:

```
<Name>ALARM</Name>
<id>1</id>
<Site>Site1</Site>
<Valid>TRUE</Valid>
<Status>ENABLED</Status>
```

```
<Family>EMS</Family>

<Application>ALARM</Application>

<Database>ALARMLST</Database>

<RecordType>CIRCLG</RecordType>

<CollectionType>Unsolicited</CollectionType>

<Key>*</Key>

<TagNameFields>LOC_CIRCLG</TagNameFields>

<TagPrefix>ALARM.</TagPrefix>

<AlarmFilter>

    <Enabled>TRUE</Enabled>

    <Location>KIRKLAND</Location>

    <Area>*</Area>

    <Category>*</Category>

    <Priority>*</Priority>

    <Exception>*</Exception>

</AlarmFilter>

<TagTemplate>

    <Value>TEXT_CIRCLG,TIME_CIRCLG,PRIOR_CIRCLG,LOC_CIRCLG</Value>

...
```

After you restart the collector, data collection will occur only for one tag: ALARM.KIRKLAND even though other tags were created. You can choose to delete these additional tags.

Note that you must also include LOC_CIRCLG in the <Value> parameter as highlighted.

### If a tag is renamed and deleted in Habitat, and then if the tag is recreated with the original name in Habitat, will it be created in Historian?

Yes, but only after you perform a few steps.

Suppose a tag named has been renamed in Habitat as follows:

- Original name: A
- New name: B

If automatic tag sync is enabled, the tag in Historian will be renamed automatically. Otherwise, the tag will be renamed only after you approve the changes *(on page 2243)*.

Now, suppose tag B is deleted in Habitat. Depending on the value you have set for the TagDeletionType parameter, tag B will be disabled or deleted in Historian.

Now, suppose a new tag named A is created in Habitat. Accordingly, tag A will be created in Historian only if you perform the following steps:

1. Stop the collector instance:
    a. Access Windows services.
    b. Right-click the HAB collector instance that you want to stop, and then select **Stop**.
2. Remove tag B *(on page 2063)*.

> **Important:**
> Do not delete the tag permanently; just remove it so that the tag and its data will still be available.

3. Start the collector instance.
    a. Access Windows services.
    b. Right-click the HAB collector instance that you want to start, and then select **Start**.

# Chapter 19. iFIX Collector

## Overview of the iFIX Data Collectors

The iFIX collectors collect data from iFIX and store it in the Historian server. They include:

- The iFIX collector
- The iFIX Alarms and Events collector

They use the Easy Data Access (EDA) protocol to retrieve data from a running iFIX system.

When you install collectors, if iFIX is installed on the same machine as the collectors, instances of the iFIX collectors are created automatically. You can begin using these collectors, or create more instances *(on page 357)* as needed using Configuration Hub.

**Features:**

- You can browse the source for tags and their attributes.
- Only the polled data collection is supported; unsolicited collection is not supported. The minimum poll interval is 100ms.
- The supported timestamp resolution is milliseconds or seconds.
- The collector accepts device timestamps.
- Floating point, integer, string, and binary data are supported.
- You can create Python Expression Tags for those collectors that support them.

**Supported tag attributes:**

- Tagname
- Source Address
- Engineering Unit Description
- Data Type
- Hi Engineering Units
- Lo Engineering Units
- Is Array Tag

## About Adding an iFIX Collector Instance

This topic provides guidelines on how to configure the iFIX collector using Configuration Hub based on the running mode of iFIX. It also describes the collector behaviour and recommended configuration in each case.

| iFIX Running Mode | Recommended Configuration for the iFIX Collector | Collector Behaviour After You Add the Collector Instance |
|---|---|---|
| iFIX is running in service mode and is secured.<br><br>The iFIX Alarms and Events and the OPC Alarms and Events Servers are running as service. | Configure the iFIX collector services under a user account under which iFIX is running as a service. While adding an instance of the iFIX collector or the iFIX Alarms and Events collector using Configuration Hub, select **Service Under Specific User Account**. | • The iFIX collector starts running as a service. It appears in the collectors list in Configuration Hub.<br>• You can run the collector at a command prompt using the Collector Start action. A shortcut is created in the Windows Start menu so that you can run the collector in the command-line mode.<br>• By default, when not started as an SCU task, the iFIX collector points to the iFIX nodename. You must configure the iFIX node in the **Collector Configuration** section in Historian Administrator. |
| iFIX is running as a service and is not secured.<br><br>The iFIX Alarms and Events and the OPC Alarms and Events servers are running as service. | You can configure the iFIX collector service using a local system account or a specific user account. | • The iFIX collector starts running as a service.<br>• You can run the collector at a command prompt using the Collector Start action. A shortcut is created in the Windows Start menu so that you can run the collector in the command-line mode.<br>• By default, when not started as an SCU task, the iFIX collector points to the iFIX nodename. You must con- |

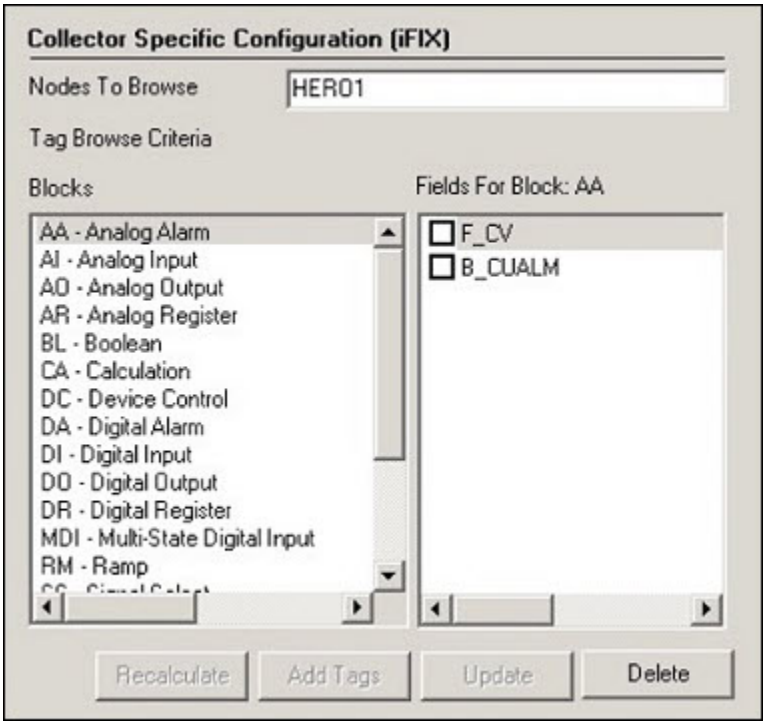
| iFIX Running Mode | Recommended Configuration for the iFIX Collector | Collector Behaviour After You Add the Collector Instance |
|---|---|---|
| | | figure the iFIX node in the **Collector Configuration** section in Historian Administrator. |
| iFIX is not running as a service mode and is secured. | Configure the iFIX collector services under a user account that is added in the IFIXUSERS group. Do not configure as a local system service. While adding an instance of the iFIX collector or the iFIX Alarms and Events collector using Configuration Hub, select **Service Under Specific User Account**. | • Since Remote Collector Manager tries to start the collector as a service, and iFIX is not running as a service, an error message appears while adding a collector instance. However, the instance is configured successfully although it does not appear in the collectors list in Configuration Hub.<br>• A shortcut is created in the Windows Start menu so that you can run the collector in the command-line mode, and the related registry folder is created.<br>• You must start the collector manually for the first time using the shortcut. It will then connect to the Historian server, and it will then appear in the collectors list in Configuration Hub.<br>• Once connected to server, you can start/stop it at a command prompt. |

| iFIX Running Mode | Recommended Configuration for the iFIX Collector | Collector Behaviour After You Add the Collector Instance |
|---|---|---|
| iFIX is not running as a service mode, and is not secured. | You can configure the iFIX collector service using a local system account or a specific user account. | • Since Remote Collector Manager tries to start the collector as a service, and iFIX is not running as a service, an error message appears while adding a collector instance. However, the instance is configured successfully.<br>• A shortcut is created in the Windows Start menu so that you can run the collector in the command-line mode, and the related registry folder is created.<br>• You must start the collector manually for the first time using the shortcut. It will then connect to the Historian server.<br>• Once connected to server, you can start/stop it at a command prompt. |
| iFIX is not running. | You can configure the iFIX collector service using a local system account or a specific user account, as per the security configuration of iFIX. | • Since Remote Collector Manager tries to start the collector as a service, and iFIX is not running as a service, an error message appears while adding a collector instance. However, the instance is configured successfully although it does not appear |

| iFIX Running Mode | Recommended Configuration for the iFIX Collector | Collector Behaviour After You Add the Collector Instance |
|---|---|---|
| | | in the collectors list in Configuration Hub.<br>• A shortcut is created in the Windows Start menu so that you can run the collector in the command-line mode, and the related registry folder is created.<br>• After you start iFIX, you must start the collector manually for the first time using the shortcut. It will then connect to the Historian server.<br>• Once connected to server, you can start/stop it at a command prompt. |

## Specify the Tags for Data Collection

1. Access Historian Administrator.
2. Select **Collectors**, and then select the iFIX collector instance to which you want to add tags.
3. Select **Configuration**.
   The **Configuration** section appears.




4. Select **Add Tags**.
   The **Add Multiple Tags from Collector** window appears.
5. In the **Collector** field, select the collector to which you want to add tags.
   A hierarchical tree of tags appears in the **Browse Results** section.
6. If you want to view only the tags for which data is not collected, in the **Show Only** field, select **Source Tags Not Collected**. You can search for a tag by entering search criteria in the **Source Tag Name** or **Description** field.
7. Navigate to the node in the tree that you want to browse, and then select **Browse**.

   **Tip:**
   - To browse automatically, select the **Auto Browse** check box. The available tags appear in the **Browse Results** window whenever a node is selected in the tree.
   - To show all child elements within a hierarchy, select the **Show All Children** check box. All tags at or below the hierarchical level of the selected node in the tree appear in the **Browse Results** window.

   The tags within the selected portion of the iFIX server tag hierarchy appear.
8. Select the tags for which you want to collect data, and then select **Add Selected Tags**.
   The tags are added to the collector. They appear in black text in the list of tags.
9.

10. Select fields for each of the blocks that you want included in your browse of the node for tags.
    a. In the Historian Non-Web Administrator, select the **Block** type from the **Block** window.
    b. Select the desired check boxes beside the fields.

    The **Block** name changes from black to blue, if any of its fields are selected.
11. Select **Update** to apply your selection.
12. Select **Add Tags** at the bottom of the page after you have selected the field.

    The **Browse Tags** window appears.
13. Select **Browse** in the **Browse Tags** window to execute the browse operation.

# Editing FixTag.dat File

### Overview

The most common block and field types are included in the default `FixTags.dat` file. You can edit this file to add or remove block and field types. The `FixTags.dat` file is an XML (eXtensible Markup Language) file. You can edit it with either a text editor like Notepad or with a third-party XML editing tool.

For Historian Non-Web Administrators, the `FixTag.dat` file is typically located in the `Program Files \Proficy\Historian\NonWebAdmin` directory of each individual Historian Non-Web Administrator client. Changes you make to the file apply only to that specific client and are not global to all Historian Non-Web Administrator clients

Because changes made to one `FixTag.dat` file are not automatically duplicated in other `FixTags.dat` files, it is recommended that you set up a standard procedure to ensure that all such files track changes made in any single file. One suggested method is to keep a master copy. To make a change, edit the master copy, deploy the changed file, and test it. If the change provides the desired effect, copy the new file to the non-web Historian Administrator directories, and save the new file as the master backup copy. All Administrators will then be using the same set of block and field types.

The `FixTags.dat` file has two main sections. The first section lists the configured FIX database block types and their descriptions. The second section lists field extensions for each of the block types defined in the first section. In addition, the second section also has an entry called `Selected`, which can be either `True` or `False`. This is used by Historian Administrator to indicate whether or not that block and field extension combination are included in a browse.

#### Example of First Section

```
<XML ID="dsoTags"><MYLIST>..<ONEITEM> <VALUE>DI</VALUE> <DATA>DI Digital Input</DATA></ONEITEM>.

.</MYLIST></XML>
```

**Example of Second Section**

```
<XML ID="DI"><MYLIST> <ONEITEM> <VALUE>F_CV</VALUE> <DATA>F_CV</DATA> <SELECTED>False</SELECTED> </ONEITEM>
 <ONEITEM> <VALUE>B_CUALM</VALUE> <DATA>B_CUALM</DATA> <SELECTED>False</SELECTED> </ONEITEM></MYLIST></XML>
```

**To Add a Field Extension to an Existing Block Type Using Notepad**

- In the FixTag.dat file, locate the entry that starts with <XML ID = "**">, where ** is the block name (such as AA, AI, DI, etc).
- Copy one of the existing <ONEITEM>...</ONEITEM> blocks of text and edit it for your new field. Each field extension is contained in a block of text contained within the text: <ONEITEM>...</ONEITEM>.

There are two entries `<VALUE>` and `<DATA>` which comprise the new field extension to be added, and a `<SELECTED>` section, which is either `True` or `False`, depending on whether or not you want the item to be included in the browse.

**Note:**
The `True` and `False` states change, depending on your selection in Historian Administrator.

An example for the Digital Input (DI) tag is shown below, where we have added the `A_ALMLASTTIME` field, which identifies the last time the block went into alarm.

```
<XML ID="DI"><MYLIST> <ONEITEM> <VALUE>F_CV</VALUE> <DATA>F_CV</DATA> <SELECTED>False</SELECTED> </ONEITEM>
<ONEITEM> <VALUE>A_ALMLASTTIME </VALUE> <DATA>A_ALMLASTTIME </DATA> <SELECTED>False</SELECTED> </ONEITEM>
<ONEITEM> <VALUE>B_CUALM</VALUE> <DATA>B_CUALM</DATA> <SELECTED>False</SELECTED> </ONEITEM></MYLIST>
```

**Add a New Database Block Using Notepad**

**Note:**
You must add at least one field.

1. Within the `<XML ID="dsoTags">` and `<MYLIST>` sections, add a new `<ONEITEM>` entry with the tag name and description. There are two entries, `<VALUE>` and `<DATA>`, which will be the new tag name (AA, AI, DI, etc.) to be added and the tag description (e.g., AA Analog Alarm) that will be displayed in Historian Administrator.
2. Create the fields that will be available for the tag. This is similar to modifying the field extensions discussed above, but in this case you must create the `<XML ID = "**">` structure (where `**` is your

tag name.) Again, this is most easily performed by copying and pasting an existing `XML ID` structure and modifying it for the new tag.

**Note:**

The `XML ID` section MUST be placed before the last three lines of the file:

```
</MYLIST>

</XML>

</BlockList>
```

# Example: Restarting the iFIX Collector Using a Heartbeat

Many applications commonly use a heartbeat to indicate when a program stops. If you want to use a heartbeat with the iFIX collector, you need to configure it through Historian Administrator and the iFIX Database Manager.

In the iFIX Database Manager, configure this address to an Analog Output (AO) block and use a separate Program (PG) block to check the status of the collector heartbeat output and restart the collector if it stopped. Then, in Historian Administrator, define a Heartbeat Output Address on the **Advanced** section of the Collector Maintenance page for the specified iFIX collector

Once configured and started, the Historian iFIX collector sends a value of 1 to the specified Heartbeat Output Address every 60 seconds. If that heartbeat value is not sent, then iFIX detects this status and restarts the iFIX collector.

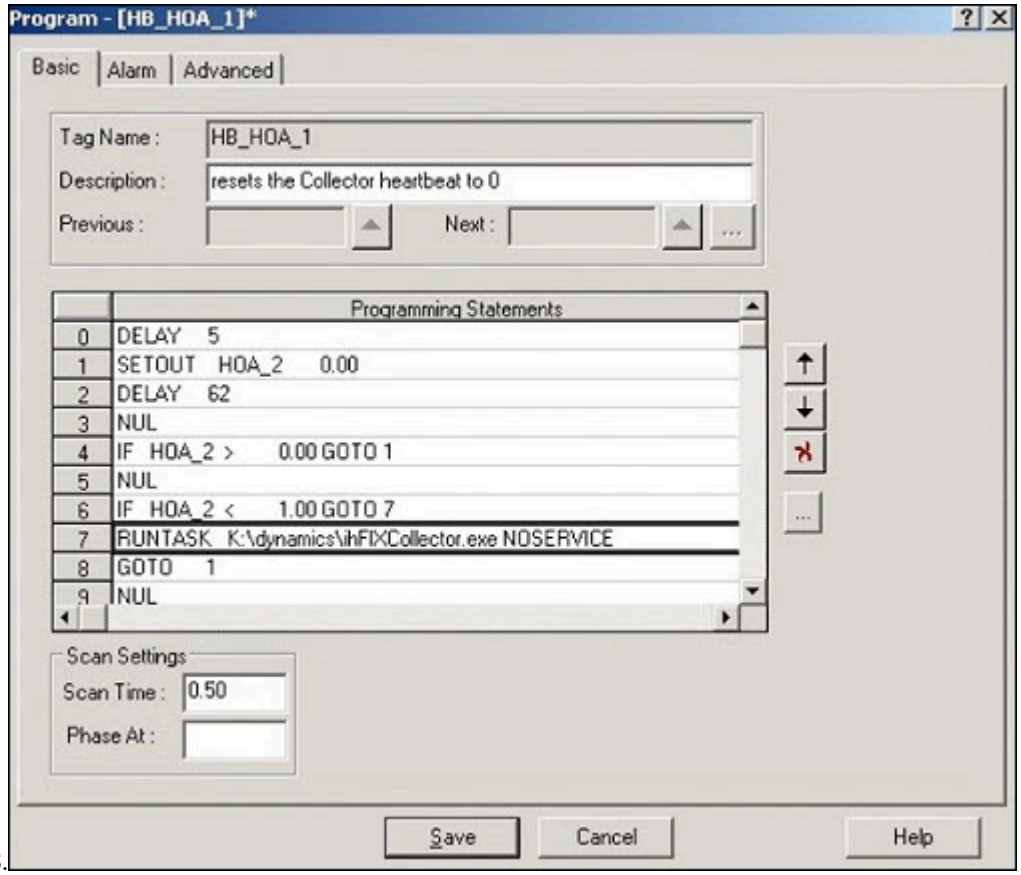
If your iFIX database PDB is quite large, you may need to increase the Delay Collection at Startup field on the **Advanced** section of the Collector Maintenance page in Historian Administrator. Doing so prevents excessive collector log entries if Historian cannot obtain a value before it initializes the source address.

In this example, an iFIX collector runs on NODE1 and the heartbeat output address is `NODE1.HOA_2.F_CV`. We create an Analog Output Block is named HOA_2 and the Program block is named `HB_HOA_1` in a new iFIX database. There are two parts to this example: configuration in iFIX and then Historian Administrator.

**Part A: Configuration in iFIX**

Follow these steps to configure the heartbeat for an iFIX Data Collector:

1. Start iFIX.
2. From the iFIX WorkSpace, open the Database Manager.

3. Select **New** from the **Database** menu.
4. Select **Add** from the **Block** menu. The **Select Block Type** window appears.
5. Select **AO** and select **OK**. The **Analog Output** window appears.
6. Enter `HOA_2` as the **Tag Name** and select **Save**.
7. Select **Add** from the **Block** menu. The **Select Block Type** window appears.
8. Select **PG** and select **OK**. The **Program** window appears.
9. Enter `HB_HOA_1` as the **Tag Name** for the block.
10. (Optional) Enter a description for the tag name.
11. Enter the programming statements shown in the following figure for lines 0 through
8. 

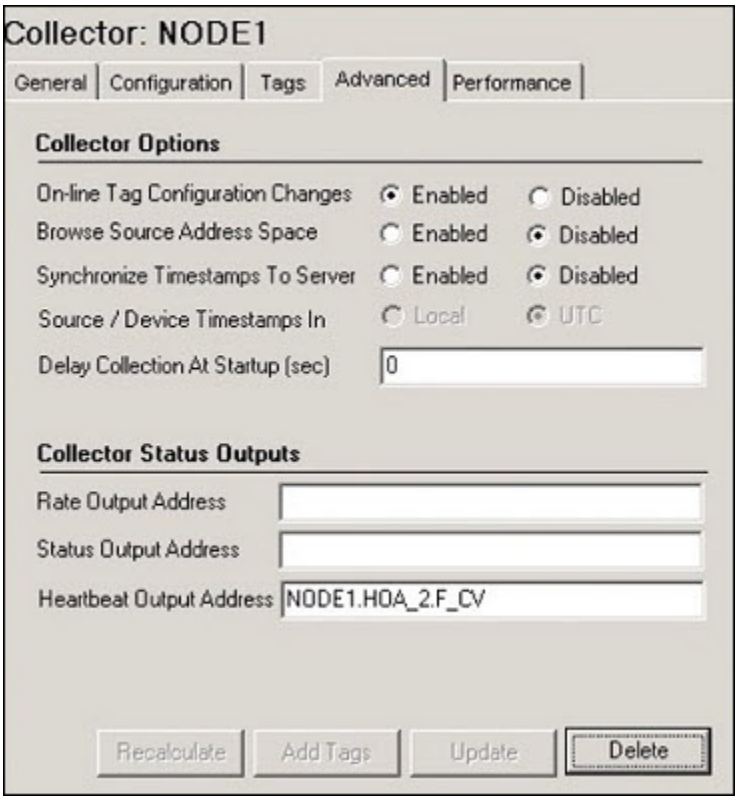
12. Enter the programming statements shown in the following figure for lines 0 through 8.
13. Select **Save**. A message box appears asking you if you want to put the block on scan.
14. Select **No**.
15. Select **Reload** from the **Database** menu in the iFIX Database Manager. The **Reload** window appears
16. Select the database you currently have loaded and select **Reload.** A message box appears to confirm the reload.
17. Select **Yes** to continue.

In about a minute, you should notice the iFIX collector start. The status of the Collector should change on the Collector Maintenance page of Historian Administrator.

**Part B: Configuration in Historian Administrator**

Follow these steps to configure the heartbeat in Historian Administrator:

1. From Historian Administrator, select on the **Collector Maintenance** page.
2. Select the iFIX collector.
3. Select **Advanced**.
4. Enter `NODE1.HOA_2.F_CV` in the **Heartbeat Output Address** field as shown in the following figure.




5. Select **Update**.

# Using an STK with the iFIX collector

To collect or browse tags with the iFIX collector when you use a system toolkit (STK) or Direct Driver Access (DDA) driver, which is also an STK, you need to edit the Fixtag.dat file to include the block types and field types for the STK. If you use a STK, there is no way to determine which block types the STK loads. Contact your STK vendor for more information.

**Example: Using the BR3 STK**

The Bristol Babcock BR3 driver uses the 160, 161, 162, and 163 block types. The following example shows how to add a `TYPE_160` block type with three field types (`F_CV`, `F_160`, and `F_XMITCNT`) for the BR3. For each new block type that you add, you need to follow the steps outlined in this example. So, if you want to add new block types for the 161, 162, and 163, you must repeat the steps outlined in this example.

**Adding a New Block Type**

Within the `<XML ID="dsoTags">` and `<MYLIST>` sections of the `Fixtag.dat` file, add a new `<ONEITEM>` entry for the new block type that you want to display in Historian Administrator. Enter the item name in the `<VALUE>` field and a description in the `<DATA>` field.

The following XML code shows what a TYPE_160 entry for the BR3 might look like in the first portion of the Fixtag.dat file.

```
<ONEITEM>
<VALUE>TYPE_160</VALUE>
<DATA>The 160</DATA>
</ONEITEM>
```

**Adding Field Types for a New Block Type**

At the end of the `Fixtag.dat` file, before the `</BlockList>` tag, add field types for each new block type that you want to add for the BR3 driver. Each field type is contained in a block of text that starts and ends with these tags: `<ONEITEM> . . . </ONEITEM>`. You must enter three values, `<VALUE>`, `<DATA>`, and `<SELECTED>` for each field type that you want to display in Historian Administrator for the specified block type.

The `<VALUE>` and `<DATA>` tags include the new field type you want to add. The `<SELECTED>` tag contains either the word `True` or `False`, depending on whether or not you want the item to be included in the browse. The `True` and `False` states will change, however, depending on your selection in Historian Administrator.

The following XML code shows an example of three field types, `F_CV`, `F_160`, and `F_XMITCNT`, added for the `TYPE_160` block type created for the BR3.

```
<XML ID="TYPE_160">
<MYLIST>
<ONEITEM>
<VALUE>F_CV</VALUE>
<DATA>F_CV</DATA>
```

```
<SELECTED>False</SELECTED>
</ONEITEM>
<ONEITEM>
<VALUE>F_160</VALUE>
<DATA>F_160</DATA>
<SELECTED>False</SELECTED>
</ONEITEM>
<ONEITEM>
<VALUE>F_XMITCNT</VALUE>
<DATA></DATA>
<SELECTED>False</SELECTED>
</ONEITEM>
</MYLIST>
</XML>
```

After you add the block types and item types to the `Fixtag.dat` file and save the file, you can view them in Historian Administrator program.

# Setting Up

## Upgrading the iFIX Collectors

Before you upgrade the iFIX collectors, back up the collector registry folders. This is because the custom, user-added Registry folders are not retained after you upgrade.

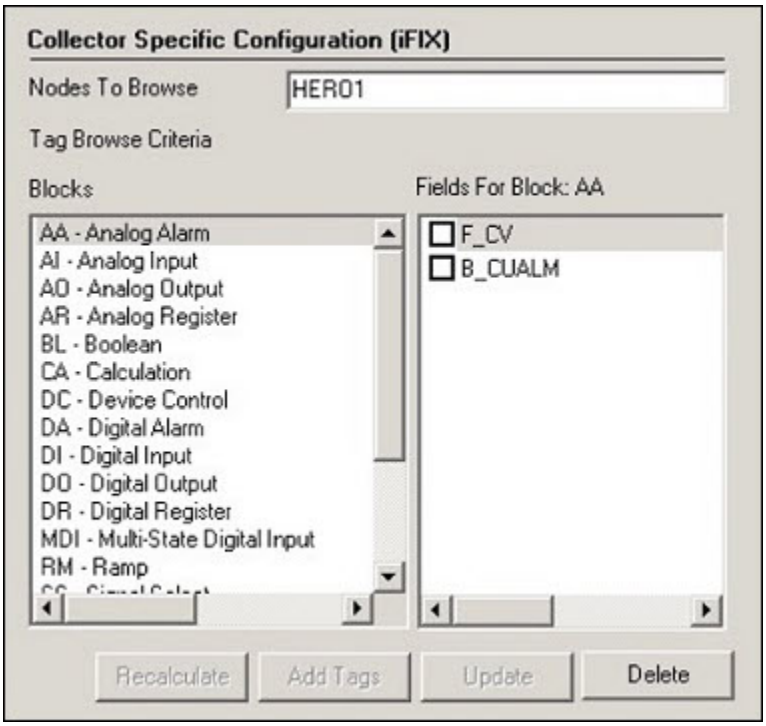
When you upgrade the iFIX collector and the iFIX Alarms and Events collectors, services are created for instances created by the installer in the previous version.

If an iFIX collector instance created in version 9.0 exists, after you upgrade collectors, another instance of the iFIX collector is created. Because of this, the Remote Collector Manager (RCM) will not work correctly. Therefore, if you want to use RCM, you must delete one of the instances. If needed, you can manually create another instance of the iFIX collector using Configuration Hub or the RemoteCollectorConfigurator utility. This is applicable to the iFIX Alarms and Events collector as well.

After upgrading an iFIX collector, if you cannot manage iFIX services using Configuration Hub, refer to Troubleshooting Remote Collector Management Issues *(on page 820)*.

## The Configuration Section for iFIX collectors

To access the **Configuration** section for an iFIX collector, select an iFIX collector from the list on the left and select **Configuration**. The following page appears. The Collector-Specific Configuration (iFIX) *(on page 2264)* topic describes the information that appears in this section.




## Collector-Specific Configuration (iFIX)

The **Configuration** section displays the following information.

| Field | Description |
|---|---|
| Node Browse Criteria | Displays the mask used to select tags when performing a browse of collector node. The default is the iFIX SCADA or Client node name on which you installed the collector.<br><br>If you want to browse for tags on other iFIX nodes via FIX networking, you can enter the other node name(s) here, separated by commas with no spaces. You must have the iFIX system configured for networking. For more information, refer to the iFIX product documentation on iFIX networking. |

| Field | Description |
|---|---|
| | **Note:**<br>If you have modified iFIX node name, then you must also update the Nodes to Browse list before browsing tags from the iFIX collector.<br><br>When you browse multiple nodes for tags to add to an iFIX collector, do not use space characters between node names or between the required comma and next node name. All characters after the space are ignored. |
| Tag Browse Criteria | See Specify the Tags for Data Collection *(on page 2255)* for more information.<br><br>**Note:**<br>If you need to add block or field types to the list, edit the `FixTag.dat` file for Historian Administrator you are using. See Editing FixTag.dat File *(on page 2257)* for more information. |

## Configuration of iFIX Data Collector-Specific Fields

Adding a Historian tag with the selected database block source address allows you to collect the history of the collector's events per minute. This is very useful for troubleshooting and optimization analysis. The following table outlines the iFIX Data Collector-specific fields.

| Field | Description |
|---|---|
| Rate Output Address | NTF in the iFIX database into which the collector writes the current value of the events/minute output, letting an operator or the HMI/SCADA application know the performance of the collector.<br><br>Use an iFIX tag for the output address. Enter the address as `NODE.TAG.FIELD` (for example, `MyNode.MySIM_AO.F_CV`).<br><br>This value displays the same statistic as the **Report Rate** field of the iFIX Data Collector in the **System Statistics** page of Historian Administrator. |
| Status Output Address | NTF in the iFIX database into which the collector writes the current value of the collector status (for example, running) output, letting an operator or |

| Field | Description |
|---|---|
| | the HMI/SCADA application know the current status of the collector. This address should be connected to a writable text field of at least 8 characters. This value is only updated upon a change in status of the collector<br><br>Use an iFIX tag for the output address. Enter the address as `NODE.TAG.FIELD` (for example, `MyNode.MyCollector_TX.A_CV`).<br><br>The text string usually displays either `Running` or `Stopped` matching the **Status** column value displayed for the iFIX Data Collector in the **System Statistics** page of Historian Administrator. |
| Heartbeat Output Address | Address in the source database into which the collector writes the heartbeat signal output. This address should be connected to a writable analog field.<br><br>Use an iFIX tag for the output address. Enter the address as `NODE.TAG.FIELD` (for example, `MyNode.MyCollector_AO.F_CV`)<br><br>The iFIX Data Collector writes a value of 1 to this location every 60 seconds while it is running.<br><br>Many applications use a heartbeat to indicate when something has stopped. For example, you could program the iFIX database to generate an alarm if the heartbeat output address is not written to once every 60 seconds notifying you that the iFIX Data Collector has stopped. |

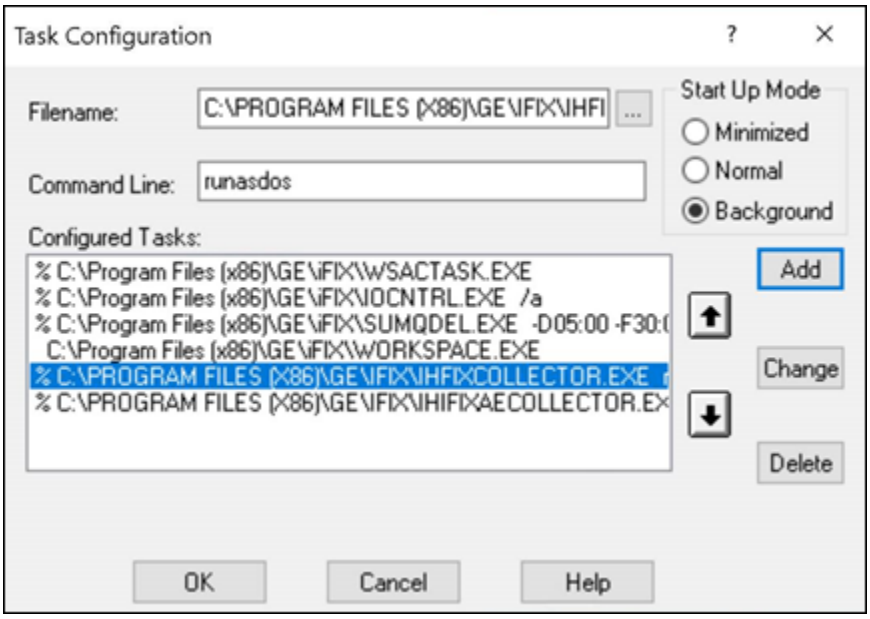
**Note:**

Adding a Historian tag with the selected database block source address allows you to collect the history of the collector's events per minute. This is very useful for troubleshooting and optimization analysis.

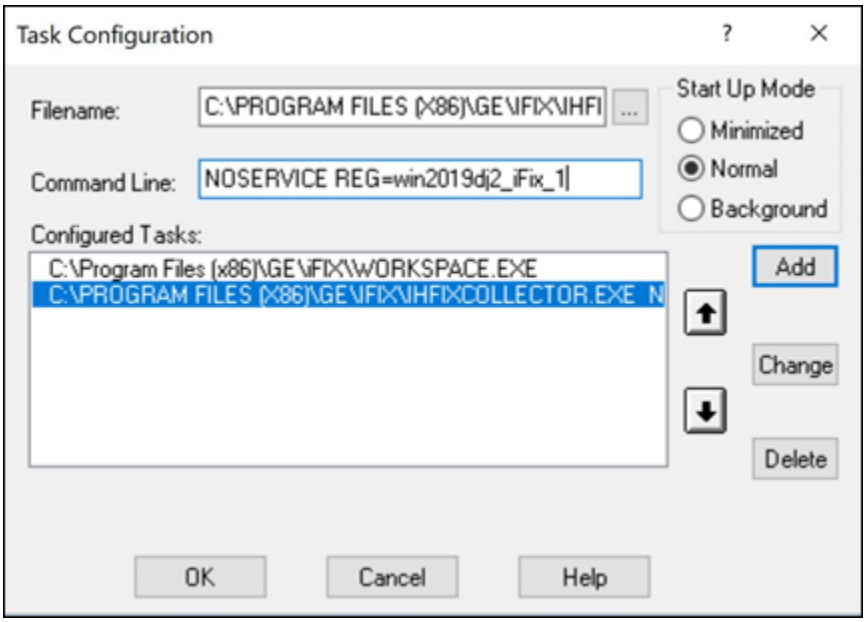
## Starting an iFIX Collector Instance

To start an instance of the iFIX collector, use one of the following options:

- **Using iFIX SCU:** Add the collector to the iFIX System Configuration (SCU) startup list. The collector then starts automatically whenever you start iFIX. This is the preferred way of starting the collector.
    - To run the default iFIX collector instances (the ones created during the installation of collectors), set the task parameters to `runasdos` as shown in the following image.




    - To run additional instances of the iFIX collectors, set the task parameters to `NOSERVICE REG=<<CollectorInterfaceName>>`, as shown in the following image for a collector with the interface name win2019dj2_iFix_1.



> **Note:**
> To find out the collector interface name:
> - If you have added the iFIX collector using Configuration Hub, the interface name is displayed in the **COLLECTOR NAME** column in the **Collectors** section.
> - If you have added the iFIX collector using the RemoteCollectorConfigurator utility, the interface name is the value that you have provided for the InterfaceName parameter while adding the collector instance.

If you set the start-up mode to normal, you can manage the instances using Configuration Hub.

- **As a console application:** From the Windows Start menu, select **Historian iFix Collector > Start iFIX Collector**. Similarly, to start an iFIX Alarms and Events collector, select **Historian iFix Alarms and Events Collector > Start iFIX AE Collector**.
- **Using Configuration Hub:** You can start collector instances *(on page 637)* and manage them *(on page 627)* using Configuration Hub. For information on the expected behavior and recommended configuration of the iFIX collectors based on the running mode of iFIX, refer to About Adding an iFIX Collector Instance *(on page 501)*.

# Troubleshooting Issues with iFIX and Historian

**Running iFIX as a Service with iFIX Workspace Listed in the SCU Task List**: Prior to iFIX 5.1, if you have configured iFIX to run as a service, you should not have WORKSPACE.EXE listed as a configured task in the Task Configuration window of the SCU. If WORKSPACE.EXE is listed as a configured task, it may lead to unpredictable results. For example, if you are also running Historian, no servers will appear in the Server Name field of the Configure the Historian Server window and you will not be able to browse Historian tags in the iFIX Expression Editor.

To rectify this, remove WORKSPACE.EXE from the list of configured tasks in the SCU.

**iFIX WorkSpace delay when remote session is lost:** If the connection between iFIX and a remote Historian session is lost, you may experience a 90 second delay in the iFIX Workspace Configuration environment, chart, or Expression Builder when accessing a pen associated with that Historian session.

In the Run Time Environment, all pens in a chart disappear for 90 seconds when the session to a remote Historian session is lost, even if they are associated with a local Historian server.

**Starting iFIX when a remote Historian session is unavailable**: If you are using Historian with iFIX, the iFIX Workspace attempts to connect to the Historian Server when it starts up. If a remote Historian server is unavailable, it may take one minute or longer for iFIX Workspace to display for each unavailable server.

**Accessing Mission Control when a remote Historian session is lost**: If a remote Historian session is lost while you are accessing the **HTC** section of Mission Control in the iFIX Workspace, the **HTC** section may become unresponsive for a minute or longer.

**Accessing tags in the iFIX chart after setting OPC "Collector to Made After Restart"**: If you add tags in Historian Administrator to a Server from an OPC Collector that has Configuration Changes set to Made After Collector Restart, you will be able to see those tags in the iFIX Expression Builder. You can add them to a chart, for example, but they have no collected data until you manually stop and restart the OPC Collector.

**Collecting data in an iFIX chart with Time Assigned By Source**: If you are retrieving data in an iFIX Chart from a Historian Server, have set the Time Assigned by field to Source, and have collectors running behind the Server time, the chart will display a flatline up to the current time of the local machine.

**Note:**
You must set Time Assigned by field to Source if you have unsolicited tags getting data from an OPC Collector.

**Synchronizing the time on iFIX SCADA Servers and View Clients**: To ensure that acknowledgements are not lost or attributed to the wrong alarm, synchronize the clocks on SCADA servers and iFIX View Client machines. If the clocks are not synchronized, alarms generated on the SCADA nodes and acknowledged on the iFIX View Client nodes could have significantly different timestamps. You can synchronize the clocks using the NET TIME command. Refer to the Windows Help system for more information.

The *Historian REST API Reference* manual specifies port 443 in examples and sample code. If you copy and paste the sample code from this Help manual, you must change this port number to your installed port.

If you have a previous install of Historian, and you have installed PHA/PKC 6.0/6.1, you will need to uninstall and then reinstall Historian.

# Chapter 20. Migrating iFix Data

## Migrating iFix Data to Historian

### About Migrating iFix to Historian

The iFIX migration tools are intended for users who are responsible for migrating their Classic or Advanced Historian systems and iFIX Alarms and Events collector data to Historian. This manual assumes that you are familiar with the iFIX and the Advanced Historian or Classic Historian environments.

This manual describes the steps for planning your Historian migration and performing the migration of your data from both the Advanced Historian and Classic Historian environments.

For more information on running iFIX, refer to the *iFIX Help*. If you are planning to implement Electronic Signatures and Electronic Records or Historian Security, it is recommended that you perform migration prior to setting up or initializing these features. If the migration will be a gradual process over time, please refer to *Using Historian in a Regulated Environment*, *Using Historian* and Implementing Security During Migration *(on page 2274)*.

### Before You Begin

Before you start migrating your Classic or Advanced Historian data, be aware of the following:

- Confirm that your Historian environment is set up. For more information, refer to the Getting Started with Historian *(on page 65)* manual.
- Confirm that you have ample disk space on the archive machine. Assume that you need to have at least as much free space as the amount of data that you intend to migrate. For example, if you have 300MB of Advanced Historian data, you need at least 300MB free disk space on the Historian Server to accept that data.
- Do not uninstall Advanced Historian until you have migrated all the data to your Historian. Advanced Historian must be running and must have all its components in place for migration. The Historian Server that you are migrating to does not need to be on the same computer as the Advanced Historian or Classic data that you are migrating.
- Migrating a large amount of data from Advanced or Classic Historian into Historian may take a substantial amount of time, depending on the size of the database and the processing power of your machines. For more information, refer to Estimating Migration time *(on page 2275)*.
- When migrating Historical data from a SCADA node to an Historian Server, it is recommended that you have an iFIX database loaded for the corresponding iFIX historical tags (on the source machine) prior to migration so that descriptions and EGUs can be retrieved.

- When migrating Historical data to Historian, it is recommended that you migrate the data in chronological order from the oldest to the newest. This prevents adding data out of order, which may impact migration performance and disk space usage.
- Migrate collection groups before historical data to maintain collection rates and deadbands.
- For migrating alarms and events data, determine the location and time range of your iFIX alarms logged via Alarm ODBC.

## Historian Migration Utilities

Historian provides three utilities that allow you to migrate your existing Classic and Advanced Historian configuration and data into the Historian environment and migrate alarms and events data from iFIX. The Historian migration tools are designed to help you migrate your data quickly and easily. The tools allow you to:

- Select which Advanced Historian tags or Classic Historian archive files to migrate.
- Set up configuration options.
- Migrate data into your Historian system.
- Migrate alarms and events data logged via iFIX Alarm ODBC into your Historian system.

## Plan Your Migration Strategy

In order to successfully migrate your data and alarms from existing iFIX or Advanced Historian applications into the Historian system, you should first plan out a sound migration strategy. Consider the following questions when planning your migration:

| Questions | Refer to section |
|---|---|
| How should I be adding tags to the archive? | Adding Tags to the Archive *(on page 2272)* |
| Should I compress my data? | Planning Compression *(on page 2272)* |
| In what order should I migrate my data? | Recommended Migration Order *(on page 2272)* |
| Do I need any security rights to migrate my data? | Implementing Security During Migration *(on page 2274)* |
| What if my data is being migrated into an already active Historian system? | Planning Migrations with Online Systems *(on page 2273)* |
| Should I account for Daylight Savings Time or Time Zone differences? | Planning Daylight Savings Time *(on page 2274)* |
| How long will my migration take? | Estimating Migration Time *(on page 2275)* |

| Questions | Refer to section |
|---|---|
| What if my ADH archives have been moved around or I'm attempting to migrate backups of older ADH archives? | Registering Advanced Historian Archives *(on page 2275)* |
| Is there anything I should do after migrating? | After Migrating Your Data *(on page 2276)* |

## Adding Tags to the Archive

You must have tags in Historian to hold the migrated data. You can add tags automatically during group migration, data migration, or manually prior to migration using Historian Administrator. When you add tags by migration, you can set the maximum number of tag properties when the migration program has access to the real time database. Having access to the iFIX real time database allows the program to retrieve the Description, HI and LO Engineering Units, and Engineering Unit descriptions. These are not stored in the Classic Historian data or group files.

Migrating group files will add tags for immediate collection. Immediate collection occurs as a result of migrating the collection interval during the process of migrating the group files. Qualifier and phase values are not preserved, deadband values are preserved.

## Planning Compression

Collector compression is automatically configured for collection groups that have a configured deadband in HTA when those collection groups are migrated to Historian.

By default, tags are migrated with archive compression turned off. Only archive compression has any effect on migration.

**To enable archive compression during migration:**

1. Add the tags through Historian Administrator, or, Migrate the historical groups.
2. After the tags are added to Historian, enable archive compression for each tag through Historian Administrator before migrating the Classic Historian data.
3. Subsequent migrated data can then pass through archive compression during migration.

## Recommended Migration Order

When migrating historical data to Historian, it is recommended that you migrate the data in a chronological order from oldest to newest. This prevents adding data out of order, which may impact migration performance. It is also recommended that you migrate collection groups before data.

## Planning Migrations with Online Systems

Typically, migration is performed after configuring and running new collection. Depending on the amount of data migrated, the process may take hours or days.

Alarm and event migration can take a significant amount of time. You can mitigate the risk of data loss by configuring the alarm collectors before starting the migration. You may also choose to migrate your alarms and events data in blocks of time ranges.

If you are migrating Classic Historian data and decide to select the **Overwrite Existing Tags** option in the **Migration Options** window, the existing Historian tag properties are replaced by the migrated Classic Historian tag properties and some configuration properties are overwritten. If you decide to clear the **Overwrite Existing Tags** option, tag information will not migrate into the Historian tag database. If you are migrating your Historian data into an existing Historian tag database, you may discover that tags with the same name exist in both the Historian database and the iFIX or Advanced Historian database.

**Note:**
Do not attempt to migrate the currently collecting Classic Historian file. If you do, you may receive an **Error -9**.

If you are migrating Advanced Historian data and decide to select the **Allow Updates** to **Existing Data** option in the Advanced Historian to Historian Migration Utility, the existing Historian tag data will be replaced by migrated Advanced Historian tag data if the data points have the same timestamps. If you decide to clear the Allow Updates to Existing Data option, the data will not be migrated (replaced) where there is existing data with duplicate timestamps in Historian.

## Limit Processing Load on Server During Migration

Both data migration utilities provide you with the ability to limit the amount of processing load on the server during migration. This allows your online Historian Server to continue processing data efficiently while migration occurs. By modifying the **Events rate/second** field in the **Classic Historian Migration Options** window or the **Max values/sec** field in the Advanced Historian to Historian Migration Utility, you can specify how much data is sent and therefore how much processing power the Migration Utility receives on your system. The minimum you can set this rate to is 10,000 (or 0, meaning no throttle); the maximum you can set this field to is 100,000.

If you are performing a migration on a computer that is also processing other data or applications, you may want to set the event processing speed lower to allow your other applications to process well. The Historian Classic Migration and Advanced Historian Migration defaults to 10,000 events per second. The

default value allows you to throttle back the Migration Utility to allow for other applications processing, as well as continue running your Migration Utility at a reduced speed. If you are not running other data or applications on your machine and you want to run the Migration Utility at maximum speed, set the Events rate/second or Max Values/sec fields to 0.

**Tip:**
It is recommended that you do not set the events per second past 25,000 or below 10,000 (unless to 0). The higher the events per second, the faster your Migration Utility processes the migration. For example, the migration may take twice as long at 10,000 as it will at 20,000 but it will take less CPU time.

**Note:**
You cannot change this entry while migrating. It must be set prior to or changed after migration.

## Implementing Security During Migration

In order to migrate data and alarms into the Historian System, you must be a member of the appropriate predefined Historian Security Groups if you have implemented Historian security. Refer to the Historian Group Rights section for information on the individual groups required for each task.

If any existing Historian Security groups are defined, you must supply a username and password before migrating your data in either the **Migration Options** window (for Classic Historian) or the **Advanced Historian to Historian Migration Utility** window (for Advanced Historian) or in the **Alarm Destination** window (for alarm migration).

## Applying Daylight Savings Time

When migrating alarms or Classic Historian data, you can select whether or not you want to apply Daylight Savings Time (DST) bias to timestamps. The Migration Utility converts the timestamps of migrated samples to UTC time (universal time format for storing timestamps) before writing the data to Historian. If you select this option, the Migration Utility will apply the DST offset before converting to UTC time. The timestamps are converted to UTC time by adjusting the time based on the local computer time zone offset and DST setting.

If you enabled DST in your operating system during data or alarm logging, you must select the **Treat as DST Timestamp** option. With Classic Historian, a switch from Daylight Time to Standard Time results in a loss of one hour of data between 1:00:00 and 1:59:59 AM. With Historian, no loss of data occurs when you switch from Daylight Time to Standard Time.

**Note:**
The migration utilities assume that the computer you are migrating your data from and the computer that you are migrating your data to are in the same Time Zone.

## Estimating Migration Time

Migrating a large historical database from either Advanced or Classic Historian into Historian may take hours or days, depending on the size of the database and the processing power of your machines.

To estimate migration time, refer to the total file size of the migration files. For Classic Historian migration, it takes approximately one minute to migrate 3 to 8.3 MBs of data. For the quickest migration, it is recommended that you clear the **Migrate Current Alarm** and the **Readback Values** option, if they are not required, in the **Classic Historian Migration Options** window.

For Advanced Historian data, it takes approximately 30 minutes to migrate 100 MB of data. This estimation assumes a default throttling of 10,000 events per second. If you have modified your **Max Values/sec** field in Advanced Historian, your migration time may vary. For more information, refer to Limit Processing Load on Server During Migration *(on page 2273)*.

**Note:**
In addition to migration file size, migration time depends upon the processing power of the computer running the migration utility. It is highly recommended that you select a computer with high processing power to run the migration program.

Alarm migration performance is bounded to the CPU power of both the migrating machine and the alarm archiver machine.

## Register Your Advanced Historian Archives

If you have changed the original location of your archives or you have backups of older archives that you wish to migrate, you must enable those archives by registering them.

Use the following guidelines when registering your archives:

- No other archive can hold data in the same time frame.
- Archives must be registered using PIARTOOL -AR {Full Path of file}.
- PIARTOOL - AL must show that the archive is registered prior to migration.

Note that any unregistered archive will not be migrated.

For more information on registering your archives, refer to step 2 of Migrating Remote Advanced Historian Data *(on page 2288)*.

## After Migrating Your Data

After migrating your data, ensure that you are no longer running Historical Collect (HTC). If you have HTC configured to start automatically from the iFIX WorkSpace, remove HTC from the SCU task startup list and replace it with the iFIX collector.

You may choose to do a Historian backup of your newly created Historian archives containing the migrated data.

After migrating your alarms, you may choose to change the data source name of the alarms so they match the data collector and appear as if they were collected along with the real time collector. For more information, see the Alarms and Events collector *(on page 2390)*.

## Adding the Historian Toolbar

To configure the toolbar to appear in the WorkSpace:

1. Start iFIX v4.0 or greater. The iFIX WorkSpace appears, if configured so.
2. Select **Toolbars** from the **WorkSpace** menu. The **Toolbars** window appears.
3. Select the **Customize** button. The **Customize Toolbars** window appears.
4. Select the **Import** button. The **Import Toolbars** window appears.
5. Select **Historian**.
6. Select the **Import** button. The **Historian Toolbar** appears.
7. Select the **Close** button to close the **Import Toolbars** window.
8. Select the **Close** button to return to the **WorkSpace**.

## Configure Historian Server Buttons

The **Configure Historian Server** button in the **Historian Toolbar** specifies the location of historical data retrieval for the WorkSpace, not the location of the historical data storage. You can view/retrieve data stored on these listed servers while you select a pen for a chart display. The **Configure Historian Servers** window also determines where HDA programs and historical ODBC retrieve data from, which is always the default server.

## Migration Checklist

The following is a list of general tasks for migrating your data to an online or new Historian system.

1. Verify that you are familiar with the setup recommendations. How *(on page 2270)*?
2. Estimate the number of archives that Historian will create during migration and verify that you have enough disk space to accommodate the new archives and backups of those archives. How *(on page 2270)*?
3. Migrate all collection groups. How *(on page 2287)*?

   If you are migrating to an online Historian system and tags exist in the iFIX database, the collector begins collecting on those tags.
4. Export tag configuration information.
5. Migrate Classic Historical data or, How *(on page 2278)*?
6. Migrate Advanced Historical data or, How *(on page 2284)*?
7. Backup the newly created archives. How?
8. Start your collectors if they are not already running.
9. Verify migrated data through one of the following options:
   - iFIX chart
   - Raw data dump into OLE DB
   - Classic Historian log file
   - Data Readback Verification option in the Classic (optional) and Advanced (automatic) Migration utilities.

# Migrating Classic Historical Data

## About Migrating Classic Historical Data

Historian supplies a utility that allows you to migrate your existing Classic Historian data (used in iFIX) into your Historian database. Before migrating your Classic Historian data, plan your migration thoroughly. Refer to Plan Your Migration *(on page 2271)* for more information.

Once you have planned your Classic Historian migration, perform the following:

1. Add the Historian Toolbar to the iFIX WorkSpace. How *(on page 2276)*?
2. Configure Classic Historian Migration Options. How *(on page 2280)*?
3. Migrate existing groups. How *(on page 2278)*?
4. Migrate your Classic Historian Data. How *(on page 2278)*?

## Migrating Classic Historian Data to Your Historian Database

The following procedure describes how to migrate Classic Historian data into your Historian database. For more information and tips on migrating data into an online system (one that is currently collecting new data as well), refer to the Plan Migrations with Online Systems *(on page 2273)*.

> **Note:**
> When migrating data, Classic Historian nodes must be online with a loaded database so that descriptions and EGU values are retrieved. If the Classic Historian nodes are not online with a loaded database, the migration utility will create tags that may have incorrect descriptions and EGU values.

**To migrate Classic Historian data into the Historian**:

1. In Classic Historian, select the **Historian Migration Button**, shown in the following figure, to open the **Historian Migration Utility**.

2. Select **Configure Options** from the **Options** menu.
3. Enter or modify any specific configuration information. For more information, refer to the Configuring Classic Migration Options section.
4. Select **Migrate Collection Groups** from the **File** menu. The Utility prompts you to specify if you would like to migrate all groups.
   The Historian Classic Migration Utility connects to the specified server and migrates the groups.

   > **Note:**
   > When migrating groups, Qualifier and Phase parameters are not migrated. If a group is not active, it is still migrated to the Historian. Groups are not preserved in Historian. All tags are added to Historian with the collection rate for the group.

5. Select **Migrate Historical Data** from the **File** menu. The **Select Historical Data File(s)** window appears.
6. Select one or more historical files and select **Open**.

The **Migration Utility** attempts to migrate all selected historical data files. The title bar displays the current file status (1 of 5, for example). Refer to the **Migration Utility** main page for information on the progress of the migration and any encountered errors.

**Note:**
The **Migration Utility** page only displays the most recent lines of the log file. For the full set of logged messages, refer to the log file, typically located in `C:\Program Files (x86)\GE\iFIX\Local\iFIX2IhMigration.Log`.

## Migrating Classic Historian 10 Character L24 Files

Classic Historian supported both 10 character and 30 character lab data files (L24). The Historian Migration Utility successfully migrates 30 character L24 files, while 10 character L24 files cannot be successfully migrated.

**Attention:**
Do not attempt to migrate 10 character L24 files.

## Comparing Classic Historian Data Plots and Historian Plots

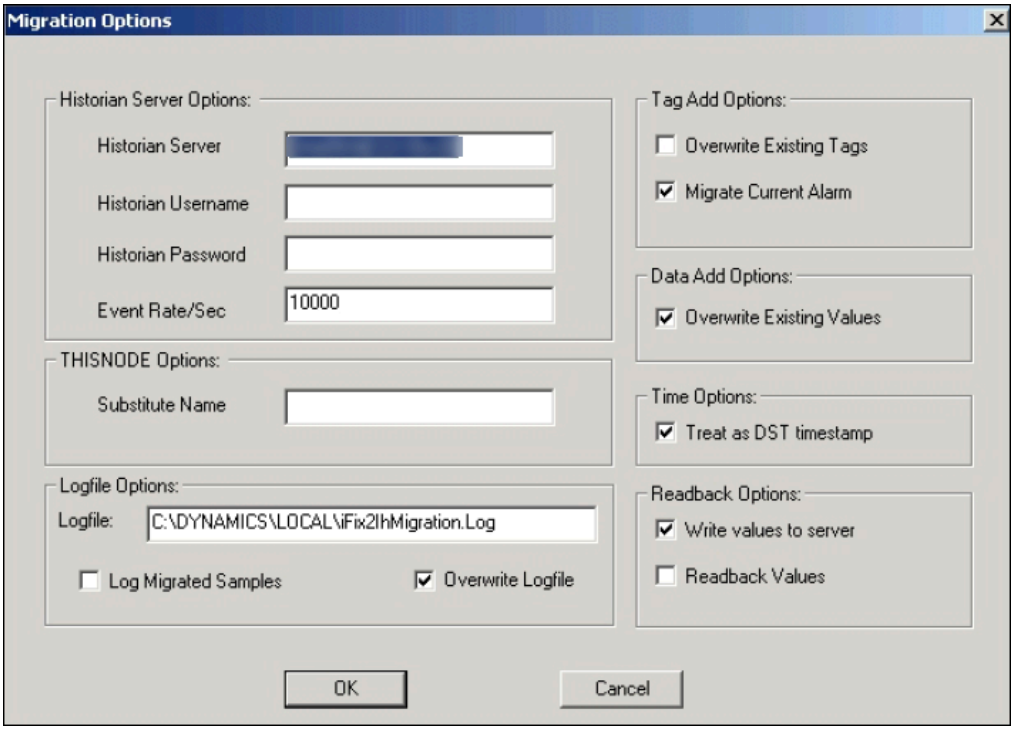
If you compare plots of average, minimum, and maximum data in Classic Historian with similar plots in Historian, you may notice the following differences:

- The sample pens match, but the plot of the average value differs. Classic Historian computes the average during the *"next"* period and Historian computes the average over the *"previous"* period.
- Classic Historian also records the minimum/maximum value from raw data values, whereas Historian uses interpolated values to compute the minimum/maximum value.
- Historian uses interpolated value because Historian works with compressed data. When you work with compressed data, interpolated values let you project what the likely minimum/maximum values were during a given time interval that includes few raw data values due to compression.

## Configuring Classic Migration Options

1. Open the **Migration Options** window, by selecting **Configure Options** from the **Options** menu in the Historian Classic Migration Utility.

   Figure 9. Classic Historian Migration Options window




   The Classic Historian Migration Tool automatically detects the default server and displays the default migration options.
2. Some of the options that you can configure include:
   - Historian Server Options *(on page 2280)*
   - THISNODE Options *(on page 2281)*
   - Tag Add Options *(on page 2281)*
   - Logfile Options *(on page 2282)*
   - Readback Options *(on page 2283)*

## Historian Server Options

Allows you to set up your server information, limit or expand the application processing time, and input any needed security information. The Historian Server Options include:

| Field | Description |
|---|---|
| Historian Server | The default server (set during installation). If you do not want to write data to the default server, enter the desired server in this field. |
| Historian Username and Password | If you have created and established Security Groups in your Historian Security Environment, you may need to enter the user name and password here. By default, if you do not supply any information, the current logged in user will be used in security checking. For more information about Historian Security, refer to Implementing Historian Security *(on page 251)* chapter of the *Getting Started* manual. |
| Event rate/sec | Allows you to limit the amount of server load caused by the Migration Utility. By reducing the **Events/rate per second** field in the **Migration Options** window, you can allocate more time to real time collection. The default rate is 10,000. For more information, refer to Limit Processing Load on Server During Migration *(on page 2273)*. |

## THISNODE Option

Specifies the Node name to use if you are migrating data collected using THISNODE. The utility automatically defaults to the name of the local iFIX node. If you want to change the Node name, enter a different Node name in this field.

## Tag Add Options

Allows you to set the following options:

| Field | Description |
|---|---|
| Overwrite Existing Tags | Selecting this option allows Historian to replace certain tag properties if the tags already exist in the Historian tag database. |
| Migrate Current Alarm | Whether or not the Migration Utility creates a tag and migrates the Current Alarm of each data sample (B_CUALM). Classic Historian paired the `B_CUALM` tag inside another tag and counted it as one. If this is enabled, the Historian Classic Migration Utility will separate these tags and add them both to the tag database. |

| Field | Description |
|---|---|
| | **Important:**<br>If you wish to migrate alarms and events data from iFIX to Historian, use the iFIX Alarms and Events collector Migration Tool.<br><br>For example:<br><br>The Classic Historian Tag:<br><br>`FIX.TAG1.F_CV`<br><br>will convert to the Historian Tags:<br><br>`FIX.TAG1.F_CV,FIX.TAG1.B_CUALM`<br><br>**Note:**<br>If you clear the **Migrate Current Alarm** check box, the Current Alarm tags are not created and the alarm information is not transferred into Historian. |
| Data Add Options<br><br>**Overwrite Existing Values** | Whether or not the Classic Migration Utility overwrites existing values that have the same timestamp as the values being migrated in an active Historian database during migration. |
| Time Options<br><br>**Treat as DST Timestamp** | Whether the migration should consider Daylight Savings Time (DST) when determining a UTC timestamp for migrated data. The timestamps are converted to UTC time by adjusting the time based on the local computer time zone offset and DST setting. For more information, refer to Managing Daylight Savings Time *(on page 2274)*. |

## Logfile Options

Allows you to configure the following options:

| Field | Description |
|---|---|
| Logfile Location | Modify the default location of the `iFIX2IhMigration.LOG` file. |
| Log Migrated Samples | Create a log file of all raw samples migrated. This will cause all the values, timestamps, and qualities to appear in the log file. |

| Field | Description |
|---|---|
| | **Important:**<br>Selecting this option significantly slows the performance of historical data migration. |
| Overwrite Logfile | Set whether or not the log file gets overwritten with each migration. By default, there is one log file (`iFIX2IhMigration.LOG`) that is overwritten each time you migrate data. Clearing this option causes the Historian Classic Migration Utility to append to the log file each time you migrate data. |

## Readback Options

Allows you to set the following options:

| Field | Description |
|---|---|
| Readback Values | Immediately read back samples after they are written. This verifies that the migration was successful. All samples that were written, including bad data quality and shutdown markers, are read back.<br><br>**Important:**<br>Selecting the **Readback Values** option puts additional load on the archiver and can slow the migration process.<br><br>If samples are not found during the readback, the Migration Utility sends a message to the migration log file (`\dynamics\local\ifix2ihmigration.log`). It does not stop the migration. |
| Write Values to Server | The data writes and tag adds are written to the Archiver.<br><br>If you want to perform a readback-only migration, clear this option and select the **Readback Values** option. This allows you to select a specific Classic historical file and perform a readback to confirm that all samples in the Classic file exist in Historian. If samples are not found during the readback, the Migration Utility sends a message to the migration log file (`\dynamics\local\ifix2ihmigration.log`). It does not stop the migration. |

# Migrating Advanced Historian Data

## Migrating Advance Historian Data

The Advanced Historian to Historian Migration Utility allows you to migrate Advanced Historian data into Historian.

Plan your Advance Historian data migration thoroughly. Refer to Plan Your Migration *(on page 2271)* for more information and establish your migration options.

1. Add the Historian Toolbar to the iFIX v2.5 or later WorkSpace. How *(on page 2276)*?
2. Explore the Advanced Migration Utility Options. How *(on page 2284)*?
3. Use the Classic Historian Migration Utility to migrate historical tag groups. How *(on page 2287)*?

> **Important:**
> If you plan to run the Advanced Historian to Historian Migration Utility on a non-SCADA node, migrate the groups after running the utility. If you try to migrate the groups first, you will not have correct EGU information for the iFIX/Advanced Historian historical tags.

4. Migrate Advanced Historian Local Data. How *(on page 2284)*?
5. Migrate Advanced Historian Remote Data. How *(on page 2288)*?

## Advanced Historian to Historian Migration Utility

You can open the Advanced Historian to Historian Migration Utility by selecting **Migration Tool for Advanced Historian** from the **Historian** directory in the **Start** menu. Some of the options you can configure include:

- Advanced Historian Connection Information *(on page 2285)* Displays information for the Advanced Historian machine you are migrating your historical data from.
- Historian Information *(on page 2285)* Displays the information for the Historian Server that you want to migrate your historical data to.
- Migration Options *(on page 2285)* Allows you to select all or specific tag masks and set the time range for migration.
- Migration Status *(on page 2286)* Displays the current status of the migration.
- Migration History *(on page 2286)* Displays the history of all migrations that have occurred on that machine.

## Advance Historian Connection Information

The **Advanced Historian Connection Information** section displays information for the Advanced Historian machine you are migrating your historical files from.

| Field | Description |
|---|---|
| Computer Name | The name of the machine you want to migrate the Advanced Historian data from. |
| Username and Password | The username and password, if you had Advanced Historian security configured for archives. |

## Historian Information

The **Historian Information** section displays the information for the Historian Server that you want to migrate your historical data to.

| Field | Description |
|---|---|
| Computer Name | The Historian Server to migrate Advanced Historian tags and data to. |
| Username and Password | If you have created and established Security Groups in your Historian Security Environment, you may need to enter the username and password here. By default, if you do not supply any information, the current logged in user will be checked. For more information on setting up Historian Security, refer to Implementing Historian Security *(on page 251)*. |

## Migration Option

The **Migration Options** section allows you to select all or specific tagmasks and set the time range for migration

| Field | Description |
|---|---|
| Tagname Mask | Specify certain tags for migration only (NODE1:* for instance) or select all tags (*). |
| Starting Date | Select a starting date for migrating data. |
| Ending Date | Set an ending date for migrating data. |

| Field | Description |
|---|---|
| | **Note:** The Migration Utility migrates data up to but not including the specified Ending Date. Carefully select your Ending Date to ensure that you include all required data in the migration. |
| Configure New Tags As Required | Specify whether the Migration Utility automatically creates tags upon migration, or migrates data only for tags that already exist in the Historian tag database. This allows you to manually create the tags you want to migrate in Historian first, then run the migration only for those tags. |
| Use Tag Configuration from iFIX | Maintains tag specific configuration for iFIX tags. If you do not select this option, Historian uses its tag configuration default settings when migrating iFIX tags.<br><br>For example, if you have an EGU set to 25-55 in an iFIX tag and you clear this option, the tag migrates with an EGU of 0-100. |
| Allow Updates to Existing Data | Allows Historian to replace data if data already exists for the tag at that timestamp. |
| End of Collection Marker | Write a bad data quality data point for each tag when the migration concludes. Select this option if you do not want your trend applications to display a continuous plot of the last good sample for that Historian tag. |

## Migration Status

Displays the current status of the migration progress. The **Migration Status** window provides real time migration status including the currently migrating tag and the tag range (2 of 1000) status. **Migration Status** window also allows you to start the migration process and abort it once it has started.

### Migration History

Displays a log of what times and tag masks were migrated and the status (success/failure) of that migration. The information displayed in the **Migration History** window is also available in the log file, typically located in the `C:\Historian Data\LogFiles` folder.

## Migrating Your Groups

Migrate your groups using the Classic Historian Migration Tool. This allows you to maintain any configured collection rates and deadbands.

**Important:**
If you plan to run the Advanced Historian to Historian Migration Utility on a non-SCADA node, migrate the groups after running the utility. If you try to migrate the groups first, you will not have correct EGU information for the iFIX/Advanced Historian historical tags.

1. Open the **Classic Migration Utility** from the Historian Toolbar.
2. Select **Configure Options** from the **Options** menu.
3. Select or clear the **Overwrite Existing Tags** option. This is the only option that applies to group migration. For more information, refer to the Configuring Classic Migration Options *(on page 2280)* section.
4. Select **Migrate Collection Groups** from the **File** menu. The Utility prompts you to specify if you would like to migrate all groups.
   The **Historian Classic Migration Utility** connects to the specified server and migrates the groups.

   **Note:**
   When migrating groups, Qualifier and Phase parameters are not migrated. If a group is not active, it is still migrated to the Historian with a collection rate.

## Migrating Existing Advance Historian Data

When migrating Advanced Historian data, you can only run the migration from a node that either has the Advanced Historian Server or the Advanced Historian client installed. Also, ensure that Automatically Create Archives is enabled in Historian Administrator.

**To migrate Advanced Historian data from an existing Advanced Historian node to your Historian Server:**

**Note:**
Before you begin, ensure the client's time is synchronized with the server's time and that there is enough free space on the Historian server to store the migrated data.

1. From the **Start** menu, select the **Migration Tool for Advanced Historian** from the Historian directory.
   The Advanced Historian to Historian Migration Utility *(on page 2284)* window appears.

2. Enter the name of the Computer from which you are migrating the Advanced Historian data.
3. If you configured Advanced Historian security, enter the username and password for security.
4. The **Advanced Historian Migration Tool** automatically detects and displays the default server in the **Historian Connection Information Computer Name** field.
   If you want to change the server you migrate data to, enter that server name in the **Computer Name** field of the **Historian Connection Information** section.
5. If necessary, enter a username and password in the **Historian Connection Information Username and Password** fields.
   If you do not provide a username and password, security defaults to check the currently logged in user.

   > **Note:**
   > You must have **Write**, **Tag Administrator**, and **Read** privileges in order to run the migration.
   > It is recommended that Security Administrators run the migration, for best performance.

6. If you do not wish to migrate all tags, enter a tagname mask for selected tags.
7. Select a start and end date from the drop-down calendars.
   For more information on estimating migration time, please refer to the Estimate Migration Time *(on page 2275)* section of this manual.
8. If you want the **Migration Tool** to create tags automatically if they do not exist, leave the **Configure New Tags as Required** option selected.
   For more information on this setting, refer to the Plan Migrations with Online Systems *(on page 2273)* section.
9. If you do not wish to overwrite existing values with migrated values containing the same timestamp, clear the **Allow Updates to Existing Data** option.
10. If you do not want your trend applications to display a continuous plot of the last good sample for that Historian tag, select the **End of Collection** marker.
    For more information on this setting, refer to Migration Options *(on page 2285)*.
11. Select the **Start Migration** button.
12. Refer to the **Migration History** window and the log file for information on the success of the tag migration and any errors or problems encountered during migration.
    You can re-run the migration utility for any time period.

## Migrating Remote Advanced Historian Data

This section describes how to migrate your Advanced Historian Data from a remote machine.

Ensure that you have installed the required software correctly. Refer to the Recommended Software Installation Order *(on page 2290)* section for more information.

If this procedure is not run as listed, you may experience problems inserting a chart control into the iFIX WorkSpace, running any programs using `fixtools.dll` or `fixhdadll.dll`, or running certain Historian features. Several `*.dll(s)` are replaced by older versions by the Advanced Historian . When Advanced Historian is removed, it replaces the older versions with the originals and Historian works correctly.

1. Ensure that PI services are running on the remote machine.

    a. Double-click the **Services** icon in the Control Panel.
    The **Services** window appears.

    b. Locate the Services beginning with 'PI'. Start all PI services except the **PI Shutdown** service.

2. Ensure that all needed archives are listed as registered. a.
    a. Open the **MSDOS** prompt.
    b. Navigate to the `Dynamics\AdvancedHistorian\adm` directory.
    c. Enter `piartool al' to list the registered archive files. –
    d. Register any unregistered archive files that you need to migrate by typing `piartool ar ` + {fully qualified archive name.} –

    Example - `piartool ar C:\Program Files (x86)\GE\iFIX\archives \DynamicArchive.001–`
3. On the local machine (machine with Historian loaded), run the Migration Utility.

    a. From the **Start** menu, select the **Migration Tool for Historian** in the Historian directory.
    The **Advanced Historian to Historian Migration Utility** *(on page 2284)* appears.

    b. Enter the name of the remote computer you are migrating the Advanced Historian data from.

    c. If you configured Advanced Historian security, enter the username and password for security.

    d. The **Advanced Historian Migration Tool** automatically detects and displays the default server in the **Historian Connection Information Computer Name** field.
    If you want to change the server to which you will migrate data, enter that server name in the **Computer Name** field of the **Historian Connection Information** section.

e. If you have set up security on your Historian Server, enter a username and password in the **Historian Connection Information Username and Password** fields. If you do not provide a username and password, security defaults to check the currently logged in user.
If you are already logged into Windows Server 2003 or Windows Serve 2008 with an account with rights to the Historian Server, you do not need to supply a new username and password.

   You must have Write, Tag Administrator, and Read privileges in order to run the migration. It is recommended that Security Administrators run the migration, for best performance.

f. If you do not wish to migrate all tags, enter a tagname mask for selected tags.

g. Select a start and end date from the drop-down calendars.
It is recommended migrating data in two month time blocks, from the oldest data to the newest. For more information on estimating migration time, please refer to the Estimate Migration Time *(on page 2275)* section of this manual.

h. If you want the Migration Tool to create tags automatically when they don't exist, leave the **Con-figure New Tags as Required** option selected.
For more information on this setting, refer to the Plan Migrations with Online Systems *(on page 2273)* section.

i. If you do not wish to overwrite any information in existing tags, clear the **Allow Updates to Existing Data** option.

j. If you do not want your trend applications to display a continuous plot of the last good sample for that Historian tag, select the **End of Collection** marker.
For more information on this setting, refer to Migration Options *(on page 2285)*

k. Select the **Start Migration** button.

l. Refer to the **Migration History** window and the log file for information on the success of the tag migration and any errors or problems encountered during migration.

## Recommended Software Installation Order

Before migrating your remote Advanced Historian Data, ensure that you have installed the required software correctly on your local machine. The following installation order is recommended:

1. Install iFIX 4.5 or higher.
2. Reboot your computer.

3. Install Advanced Historian
    a. Select Custom Installation with ONLY Client install.
    b. Reboot your computer.
4. Install Historian completely.

## Removing Advanced Historian (From Local Machine)

After you have completely migrated your remote Advanced Historian Data, it is recommended that you remove Advanced Historian from your local machine.

1. Stop all PI services from the Control Panel Services window.

    a. Double-click the **Services Icon** in the Control Panel.
    The **Services** window appears.

    b. Locate the **PI Network Manager Service** and select the **Stop** button.

2. Another window appears asking if you would like all the other services stopped.
    Select **Yes**.
3. Locate the **bufserv** service and stop it.
4. Completely remove Advanced Historian using the uninstall feature in the **Add/Remove Programs** window.
5. Reboot your computer.
6. Delete the `\AdvancedHistorian` directory from the Dynamics base path.

    **Note:**
    You may want to stop **PI Services** again, as detailed in step 1.

7. Use the **AHClean** utility located on the GE Automation web site in the Developers Corner.
8. Run `AHClean.exe` and select the **Clean Registry** button.
9. Reboot your computer.

# Migrating iFIX Alarms and Events Collector

## Migrating iFIX Alarms and Events collector

The following procedure describes how to migrate iFIX Alarms and Events collector data into your Historian database.

1. Locate and double-click the `ifixalmmig.exe` file in the `C:\Program Files (x86)\GE\iFIX\Local` directory.
2. Set up the alarm source *(on page 2293)* options:
    a. From the **Options** menu, choose **Alarm Source**. The **Alarm Source Options** window appears.
    b. Configure the ODBC Login *(on page 2293)* information.
    c. Enter the table name for the database, and select on **Fetch Columns**.
    The table's associated columns appear in the **SQL Column Name and iFIX Field Name** table.
    For more information, refer to the Database Configuration *(on page 2293)* .
    d. Configure Severity Mapping *(on page 2293)*.
3. Select **OK**.
4. Set up the Alarm Destination *(on page 2295)* options:
    a. From the **Options** menu, choose **Alarm Destination**. The **Alarm Destination Options** window appears.
    b. Configure the Historian Server Options *(on page 2295)*.
    c. Configure the Logfile *(on page 2295)*.
    d. Configure the Time Options *(on page 2295)*.
5. Select **OK**.
6. From the **File** menu, choose **Migrate Alarms**.
7. Select a start and end time for the alarm migration.

> **Note:**
> If the ODBC driver does not support the CAST function, It is recommended that you create the DATETIME data type filed in your Alarm ODBC table and update the field by combining The ALM_DATELAST and the ALM_TIMELAST using a concatenation and cast or convert function. If the data source does not include a native time, the start and end times will be disabled, and the entire table will be queried. This may result in an "out of resources" state.
>
> See your *Alarm ODBC backend Data Base* documentation for more information.

8. Select **OK**. The window closes, and migration commences.
Activity is logged to both the page and the log file configured in the **Alarm Destination Options** window.
9. Update the datasource name of your migrated alarms to match collected alarms.

## iFIX Alarms and Events collector Migration Options Configuration

Configuration of the iFIX Alarms and Events collector Migration tool is contained in two windows. The **Alarm Source Options** window contains configuration for the iFIX system you wish to migrate alarms and events data from. The **Alarm Destination Options** window contains configuration for the Historian archive you wish to migrate alarms and events data to.

In general, configuration of the alarm migration options should match the configuration of your iFIX Alarms and Events collector server. By doing this, your migrated and collected alarms are stored in the same format, easing the development of alarm analysis and reporting applications.

## Alarm Source Options

The **Alarm Source Options** window is split into four sections:

- ODBC Login Information
- Attribute Names
- Severity Mapping
- Database Configuration

### ODBC Login Information

The following configuration fields are shown in the ODBC Login Information section of the **Alarm Source Options** window. In general, the settings in this section should match the settings used in your iFIX Alarm ODBC Configuration in the iFIX SCU.

| Option | Description |
|---|---|
| Database Type | The type of database your iFIX alarms are stored in. The following options are available:<br>• Access<br>• Oracle<br>• SQL Server<br>• Sybase<br>**Note:** If you are unsure of which database your iFIX alarms are stored in, contact your systems administrator |

| Option | Description |
|---|---|
| User Name | The user name required to authenticate with your database. |
| Password | The password required to authenticate with your database. |
| Database Identifier | The name of the database your iFIX alarms are stored in. Select the **Browse** button to bring up the **Database IDs Available** window.<br><br>**Note:**<br>If you are migrating on a machine other than the machine running Alarm ODBC, you may have to configure a DSN in the control panel before entering it in the Database Identifier. |

### Attribute Names

Use this section only if you have logged alarm user and/or extension fields AND you want to control the name of the fields created in Historian. If you have changed the default names in the iFIX Real Time Alarms and Events Server, edit the configuration in this section to match.

| Option | Description |
|---|---|
| iFIX Field Name | The iFIX Field name of the alarm. |
| Attribute Name | The mapped attribute name of the alarm. |

### Severity Mapping

If you have changed the default severity settings in the real-time AE Server, edit the configuration in this section to match.

| Option | Description |
|---|---|
| Low | The severity to assign to low priority alarms. |
| Medium | The severity to assign to medium priority alarms. |
| High | The severity to assign to high priority alarms. |

### Database Configuration

If you have set up custom column names in the iFIX Alarm ODBC configuration, you may need to map the **SQL Column Names** to **iFIX Field Names**. If you have kept the default column names in the iFIX Alarm ODBC configuration, you should not have to make any changes in this section.

| Option | Description |
|---|---|
| Table Name | Enter the name of the database table in which iFIX Alarms and Events collector data is stored. |
| Fetch Columns | Selecting this button will fetch all columns from the table. |
| SQL Column Name | Identifies the name of the column within the database table. |
| iFIX Field Name | Identifies which iFIX Field name the SQL Column name maps to. |
| Clear Column Settings | Select to clear all column settings. |
| Save Column Settings | Select to save all column settings in the window. |

SQL columns and their iFIX meanings are automatically matched up when the columns are fetched. You only need to configure iFIX meanings if you have used a non-default column name. If you want to exclude columns from migration to save space in Historian, set the iFIX meaning to blank. For example, if you logged both **Native Time Last** and **Time Last** in Alarm ODBC, you only need to migrate the **Native Time Last** field. Set the iFIX meaning of **Time Last** to blank.

If you use the **Save Column Settings** option, iFIX meanings are saved to `C:\Program Files (x86)\GE\iFIX\local\ifixalmmig.ini`. When working with a different database table, you can erase the iFIX meanings by deleting the `ifix-almmig.ini` file or selecting the **Clear Column Settings** button.

**Note:**
The Alarms and Events database version must match the SQL Server version that it is running on.

## Alarm Destination Options

The **Alarm Destination Options** window has three sections:

- Historian Server Options
- Logfile Options
- Time Options

### Historian Server Options

| Option | Description |
|---|---|
| Historian Server | The server name of the Historian server you wish to migrate the iFIX Alarms and Events collector data to. |
| Historian Username | The username required to authenticate with the Historian server. |
| Historian Password | The password required to authenticate with the Historian server. |
| | |

### Logfile Options

| Option | Description |
|---|---|
| Logfile Location | Modify the location of the iFIXAlmMigration.Log file. |
| Overwrite Logfile | Set whether or not the log file gets overwritten with each migration. By default, there is one log file (`iFIXAlmMigration.Log`) that is appended each time you migrate alarms. Clearing this option causes the iFIX Alarm Migration Utility to overwrite the log file each time you migrate alarms. |

### Time Options

| Option | Description |
|---|---|
| Treat as DST timestamp | If enabled, the migration utility will treat all timestamps as if they are Daylight Savings Time.<br><br>Refer to Manage Daylight Savings Time *(on page 2274)* for more information. |

# Troubleshoot iFIX Alarms and Events collector

## Wrong Data Source Name on Migrated alarms

The data source of an alarm starts as the interface name of the alarms and events collector that sent it. If the alarms and events collector is not associated with a data collector, then the data source is converted to match the data collector's interface name. When migrating alarms, however, the original data source name will be retained. This can cause a disconnect between migrated data and newly collected data, and cause problems when analyzing alarms and events data.

To resolve this issue:

1. Select the **File** menu, then select **Update Alarms**.
2. In the **Old Datasource** field, enter the original data source name.
3. In the **New Datasource** field, enter the new data source name.
4. Select **OK**.

## Datetime Column Not Found

iFIX prior to v2.6 did not have a datetime column.

## No Columns Returned

If no columns are returned after entering a table name and selecting **Fetch Columns**, you may have incorrectly entered the Alarm ODBC table name. Re-enter the Alarm ODBC table name and select **Fetch Columns**.

# Chapter 21. The MQTT Collector

## Overview of the MQTT Collector

The MQTT collector collects data published to a topic using an MQTT broker.

> **Note:**
> We have tested with the MQTT brokers Mosquitto 2.0.15 and HiveMQ-4.2.1. You can, however, use other MQTT brokers as well.

**Supported MQTT versions:**

- MQTT V5
- MQTT V3.1.1

**Supported data formats:**

- Sparkplug B V1.0

  > **Note:**
  > The MQTT collector supports Sparkplug B payload and processes only NDATA or DDATA messages, discarding the other messages.

- KairosDB (that is, the Predix Timeseries format)

**Topology:** The MQTT collector supports a distributed model, that is, the MQTT broker, the collector, and the Historian server can be installed on the same machine or on different machines.

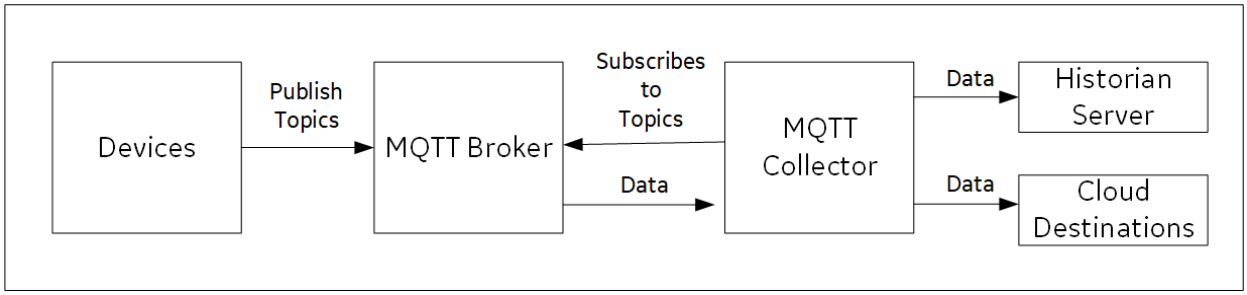
**Features:**

- You can subscribe to multiple-level topics using a wildcard.
- Only the unsolicited data collection is supported; polled collection is not supported.
- The timestamp resolution is seconds, milliseconds, and microseconds.
- Array, boolean, floating point, integer, and string data types are supported.

**Note:**
Although the **Recalculate** button in Historian Administrator is enabled, the functionality is not available because the MQTT collector is a non-historic collector (the source broker does not store the historical data).

**How it works:**

1. The MQTT collector connects to an MQTT broker and subscribes to a topic. You can use username/password-based authentication or certificate-based authentication. Transport Layer Security (TLS) authentication is used for subscribing the data from message broker to avoid middleware attacks so that the data is securely transferred from message broker to the MQTT collector.
2. The collector converts the data from the Sparkplug B v1.0 or the KairosDB format to a Historian-understandable format.
3. It verifies whether the tag is available in Historian; if not, it will add the tag and then add the data samples, and streams the data to the Historian server or a cloud destination.




**KairosDB Message Format:**

```
{
"body":
[
    {
        "attributes":{"machine_type":"<value>"},
        "datapoints":[[<value>,<value>,<value>]],
        "name":"<value>"}],
        "messageId":"<value>"}
```

The following table describes these parameters.

| JSON Parameter | Description | Required/Optional |
|---|---|---|
| machine_type | The name of the machine from which you want to collect data. | Optional |
| datapoints | Time (in epoch format), value, and quality. | Required |
| name | The tag name | Required |
| messageId | The type of the message | Optional |

**Note:**
For the parameters marked optional, you need not enter values. However, you must enter the parameter names. For example:

```
{"body":[{"attributes":{"machine_type":" "},
"datapoints":[[1558110998983,9547909,3]],"name":"QuadInteger"}],"messageId":" "}
```

**Supported Data Types**

| Source Data Type | Historian Data Type |
|---|---|
| Array | ihArray |
| DoubleFloat, DoubleInteger, FixedByte, QuadInteger, SingleFloat | ihDoubleFloat |
| ByteString, String | ihVariableString |
| Boolean | ihBool |

# Configuration

## Add and Configure an MQTT Collector

1. Ensure that you have an MQTT broker.

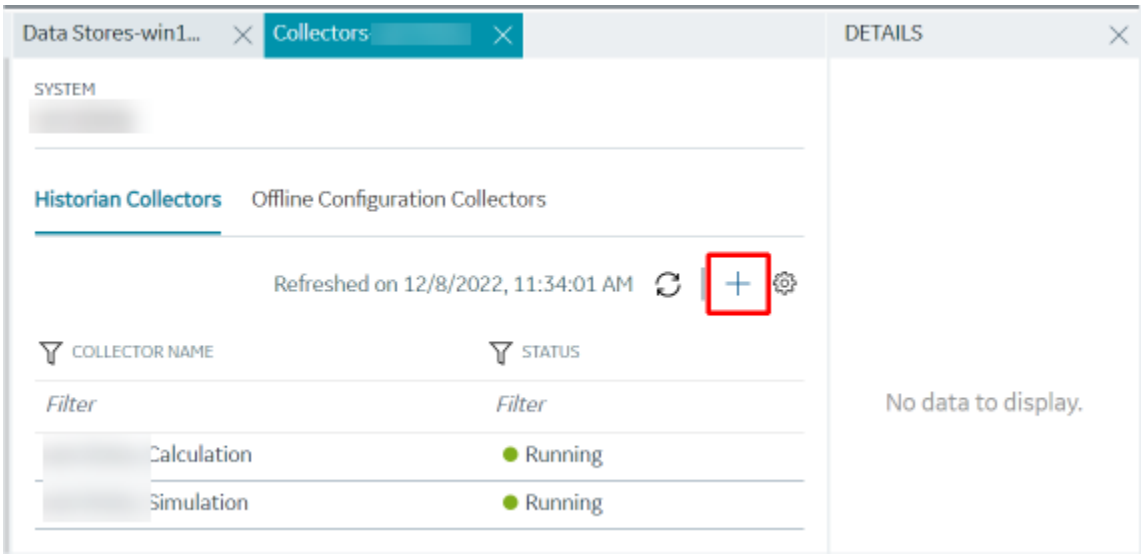
**Note:**
We have tested with the MQTT brokers Mosquitto 2.0.15 and HiveMQ-4.2.1. You can, however, use other MQTT brokers as well.

2. If you want to use username/password-based authentication or certificate-based authentication to connect the MQTT broker and the MQTT collector, configure the authentication in the MQTT broker.
3. If you want to use certificate-based authentication, ensure that the following files are available on your collector machine:
    - CA server root file
    - Private key file
    - Client certificate file

The MQTT collector collects data published to a topic using an MQTT broker. For more information, refer to Overview of the MQTT Collector *(on page 2298)*.

This topic describes how to add a collector instance using Configuration Hub. You can also add a collector instance using the RemoteCollectorConfigurator utility *(on page 797)*, which does not require you to install Web-based Clients.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.
   A list of collectors in the default system appears.
3. In the upper-right corner of the main section, select +.




   The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.
4. In the **MACHINE NAME** field, select the machine in which you want to add a collector instance.
5. In the **COLLECTOR TYPE** field, select **MQTT Collector**, and then select **Get Details**.
   The **INSTALLATION DRIVE** and **DATA DIRECTORY** fields are disabled and populated.

6. Select **Next**.
   The **Source Configuration** section appears.
7. Enter values as described in the following table.

| Field | Description |
|---|---|
| **MQTT BROKER ADDRESS** | Enter the host name of the MQTT broker using which you want to collect data. A value is required. |
| **MQTT BROKER PORT** | Enter the port number of the MQTT broker. A value is required. |
| **TOPIC** | Enter the MQTT topic from which you want to collect data. A value is required. You can enter multiple topics separated by commas.<br><br>If you want to use the Sparkplug B format, enter a value in the following format: `namespace/group_id/message_type/edge_node_id/device_id`<br><br>where:<br>◦ `namespace` is the Sparkplug version. Enter `spBv1.0`.<br>◦ `group_id` is the ID of the group of nodes from which you want to collect data.<br>◦ `message_type` is the message type from which you want to collect data. The collector processes data only from NDATA and DDATA message types.<br>◦ `edge_node_id` is used to identify the MQTT EoN node within the infrastructure.<br>◦ `device_id` a device attached to the MQTT EoN node either physically or logically.<br>You can use the wildcard character # for any of these parameters (except for namespace). |
| **USERNAME** | Enter the username to connect to the MQTT broker. A value is required if you have configured username/password-based authentication in the MQTT broker. |
| **PASSWORD** | Enter the password to connect to the MQTT broker. A value is required if you have configured username/password-based authentication in the MQTT broker. |

| Field | Description |
|---|---|
| **CA SERVER ROOT FILE** | Enter the path to the CA server root file to connect to the MQTT broker. A value is required if you have configured certificate-based authentication in the MQTT broker. |
| **PRIVATE KEY FILE** | Enter the path to the private key file to connect to the MQTT broker. A value is required if you have configured certificate-based authentication in the MQTT broker. |
| **CLIENT CERTIFICATE FILE** | Enter the path to the client certificate file to connect to the MQTT broker. A value is required if you have configured certificate-based authentication in the MQTT broker. |
| **REQUESTED QUALITY OF SERVICE (QOS) LEVEL** | Select the quality of service that you want to use while collecting data from an MQTT broker.<br>◦ **QoS 0**: Indicates that the message is delivered at most once or it is not delivered at all.<br>◦ **QoS 1**: Indicates that the message is always delivered at least once.<br>◦ **QoS 2**: Indicates that the message is delivered once.<br>For more information, refer to https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/. |
| **MQTT VERSION** | Select the version of the MQTT that you want to use. |
| **CLEAN SESSION** | Select one of the following options:<br>◦ **True**: Select this option if you do not want to create a new session when the MQTT broker and the collector are disconnected from each other.<br>◦ **False**: Select this option if you want to retain the session when the MQTT broker and the collector are disconnected from each other. This ensures that there is no loss of data. If you want to choose this option, ensure that you have selected **QoS 1** or **QoS 2** in the **REQUESTED QUALITY OF SERVICE (QOS) LEVEL** field. |
| **SESSION EXPIRY INTERVAL** | Enter the duration, in seconds, after which the data will be discarded when connection between the MQTT broker and collector is re-established. |

| Field | Description |
|---|---|
| | For example, if you enter 100 in this field, and if the MQTT broker and collector are disconnected for 90 seconds, the data is collected. If, however, the MQTT broker and the collector are disconnected for more than 100 seconds, the data will be discarded.<br><br>This field is applicable only for MQTT V5 and only if you set the **CLEAN SESSION** field to **False**. |
| **CONTENT TYPE** | Select the format that you want to use for the payload:<br>◦ **JSON**: Select this option if you want to use the KairosDB format.<br>◦ **SparkPlug B v1.0**: Select this option if you want to use the Sparkplug format. |

8. Select **Next**.
   The **Destination Configuration** section appears. The collector machine name provided by you is selected as the **Source Configuration** by default.

   Under **CHOOSE DESTINATION**, the **Historian Server** option is selected by default. In addition, the **DESTINATION HISTORIAN SERVER** field is disabled and populated with the collector machine name.

9. Select the destination to which you want to send data, and then enter the values in the corresponding fields. You can send data to an on-premises Historian server or to a cloud destination.

   a. If you need to send data to a cloud destination, select the cloud destinations as needed.
      - **Predix Timeseries**- Select this if you need to send data to Predix cloud. For more information, refer to Predix Cloud *(on page 621)*.
      - **Azure IoT Hub**- Select this if you need to send data to Azure Cloud in KairosDB format. For more information, refer to Azure IoT Hub (KairosDB format) *(on page 608)*.
      - **MQTT**- Select this if you need to send data to any of the following cloud destination.
        - Alibaba cloud. For more information, refer to Alibaba Cloud *(on page 587)*.
        - AWS cloud. For more information, refer to AWS Cloud *(on page 594)*.
        - Google cloud. For more information, refer to Google Cloud *(on page 614)*.

   b. If you need to send data to an on-premises Historian server, select **Historian Server**.

If you created security groups or enabled a strict client/collector authentication, enter the **USERNAME** and **PASSWORD** of the on-premises Historian server that you created during the installation of the collector.
If you entered the **USERNAME**and **PASSWORD**, select **Test Connection**. This will help you to test if the Historian server that you are trying to connect is valid or if the credentials that you entered are valid.
If the entered credentials are valid, a successful connection message appears.

10. Select **Next**.
    The **Collector Initiation** section appears.
11. If needed, modify the value in the **COLLECTOR NAME** field. The value must be unique, must contain the string `MQTT`, and must not contain a space.
    The value that you enter:
    - Must be unique.
    - Must contain the string `MQTT`.
    - Must not exceed 15 characters.
    - Must not contain a space.
    - Must not contain special characters except a hyphen, period, and an underscore.
12. In the **RUNNING MODE** field, select one of the following options.
    - **Service - Local System Account**: Select this option if you want to run the collector as a Windows service using the credentials of the local user (that is, the currently logged-in user). If you select this option, the **USERNAME** and **PASSWORD** fields are disabled.
    - **Service Under Specific User Account**: Select this option if you want to run the collector as a Windows service using a specific user account. If you select this option, you must enter values in the **USERNAME** and **PASSWORD** fields.

      If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:
      - iH Security Admins
      - iH Collector Admins
      - iH Tag Admins

    You can also configure the collector to start automatically when you start the computer.
13. Select **Add**.
    The collector instance is added. The fields specific to the collector section appear in the **DETAILS** section.

14. In the **COLLECTOR SPECIFIC CONFIGURATION** and **INSTANCE CONFIGURATION** sections, configure the values as described in the following table.

**COLLECTOR SPECIFIC CONFIGURATION**

| Field | Description |
|---|---|
| **MTLS Security** | Indicates whether you want to use Mutual TLS (MTLS) protocol to enforce a secure and strong authentication mechanism. |
| **MTLS Data Encryption** | Indicates whether you want to encrypt the data that the collector shares to the data archiver (DA). |
| For more information on how to enable MTLS Security, refer to Enable MTLS Security *(on page 632)*. | |

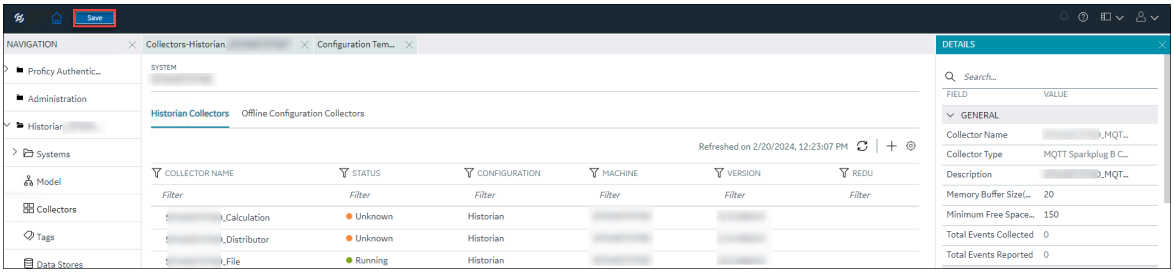
**INSTANCE CONFIGURATION**

| Field | Description |
|---|---|
| **MQTT Broker Address** | Enter the host name of the MQTT broker using which you want to collect data. A value is required. |
| **MQTT Broker Topic** | Enter the MQTT topic from which you want to collect data. A value is required. You can enter multiple topics separated by commas.<br><br>If you want to use the Sparkplug B format, enter a value in the following format: `namespace/group_id/message_type/edge_node_id/device_id`<br><br>where:<br>◦ `namespace` is the Sparkplug version. Enter `spBv1.0`.<br>◦ `group_id` is the ID of the group of nodes from which you want to collect data.<br>◦ `message_type` is the message type from which you want to collect data. The collector processes data only from NDATA and DDATA message types. |

| Field | Description |
|---|---|
| | ◦ `edge_node_id` is used to identify the MQTT EoN node within the infrastructure.<br>◦ `device_id` a device attached to the MQTT EoN node either physically or logically.<br>You can use the wildcard character # for any of these parameters (except for namespace). |
| **MQTT Brker Port** | Enter the port number of the MQTT broker. A value is required. |
| **Username** | Enter the username to connect to the MQTT broker. A value is required if you have configured username/password-based authentication in the MQTT broker. |
| **Password** | Enter the password to connect to the MQTT broker. A value is required if you have configured username/password-based authentication in the MQTT broker. |
| **CA Server Root File** | Enter the path to the CA server root file to connect to the MQTT broker. A value is required if you have configured certificate-based authentication in the MQTT broker. |
| **Private Key File** | Enter the path to the private key file to connect to the MQTT broker. A value is required if you have configured certificate-based authentication in the MQTT broker. |
| **CLIENT Certificate File** | Enter the path to the client certificate file to connect to the MQTT broker. A value is required if you have configured certificate-based authentication in the MQTT broker. |
| **Requested Quality Of Service (QoS) Level** | Select the quality of service that you want to use while collecting data from an MQTT broker.<br>◦ **QoS 0**: Indicates that the message is delivered at most once or it is not delivered at all.<br>◦ **QoS 1**: Indicates that the message is always delivered at least once.<br>◦ **QoS 2**: Indicates that the message is delivered once.<br>For more information, refer to https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/. |
| **MQTT Version** | Select the version of the MQTT that you want to use. |

| Field | Description |
|---|---|
| **CLEAN Session** | Select one of the following options:<br>◦ **True**: Select this option if you do not want to create a new session when the MQTT broker and the collector are disconnected from each other.<br>◦ **False**: Select this option if you want to retain the session when the MQTT broker and the collector are disconnected from each other. This ensures that there is no loss of data. If you want to choose this option, ensure that you have selected **QoS 1** or **QoS 2** in the **REQUESTED QUALITY OF SERVICE (QOS) LEVEL** field. |
| **SESSION Expiry Interval** | Enter the duration, in seconds, after which the data will be discarded when connection between the MQTT broker and collector is re-established.<br><br>For example, if you enter 100 in this field, and if the MQTT broker and collector are disconnected for 90 seconds, the data is collected. If, however, the MQTT broker and the collector are disconnected for more than 100 seconds, the data will be discarded.<br><br>This field is applicable only for MQTT V5 and only if you set the **CLEAN SESSION** field to **False**. |
| **Content Type** | Select the format that you want to use for the payload:<br>◦ **JSON**: Select this option if you want to use the KairosDB format.<br>◦ **SparkPlug B v1.0**: Select this option if you want to use the Sparkplug format. |

15. As needed, enter values in the other sections common to all collectors *(on page 579)*.
16. In the upper-left corner of the page, select **Save**.



The changes to the collector instance are saved.

17. If needed, restart the collector.

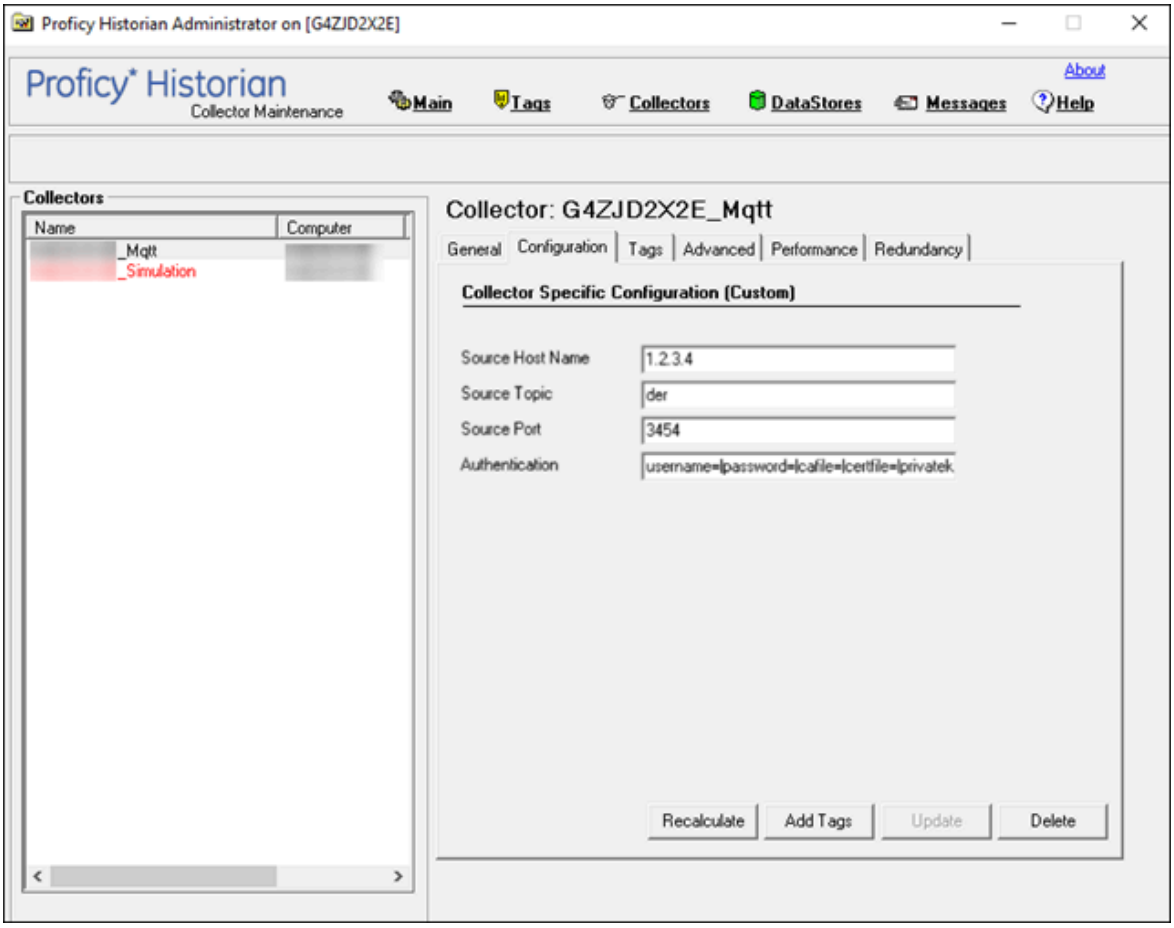
Specify the tags whose data you want to collect using the collector. In the **CHOOSE CONFIGURATION** field,

- If you have selected **Historian Configuration**, specify the tags using Configuration Hub *(on page 357)*.
- If you have selected **Offline Configuration**, modify the offline configuration file of the collector. By default, the file is available in the following location: `<installation folder of Historian>\GE Digital\<collector name>`. For information, refer to Offline Configuration for Collectors *(on page 2039)*. This option is applicable only if you have selected a cloud destination.

## Configure the MQTT Collector Using Historian Administrator

If you want to connect the MQTT collector with an MQTT broker using a username/password-based or a certificate-based authentication, configure the same in the MQTT broker.

1. Access Historian Administrator.
2. Select **Collectors**, and then select the MQTT collector instance that you want to configure.
3. Select **Configuration**.
   The **Configuration** section appears.




4. Enter values as specified in the following table.

| Field | Description |
|---|---|
| Source Host Name | The hostname or IP address of the machine on which the MQTT message broker is running. |
| Source Topic | The topic for which you want to get the data from the message broker. |
| Source Port | The port number of the machine on which the MQTT message broker is running. |
| Authentication | The authentication details to connect to the MQTT broker.<br>If you want to use MQTT V3.1.1, enter a value in the following format: `username=<value>\|password=<value>\|cafile=<path to the CA server root file\|certfile=<path` |

| Field | Description |
|---|---|
| | `to the client certificate file>\|privatekeyfile=<path to the private key file>\|qos=<quality of service value>\|version=MQTT_V311`<br>If you want to use MQTT V5, enter a value in the following format: `username=<value>\|password=<value>\|cafile=<path to the CA server root file\|certfile=<path to the client certificate file>\|privatekeyfile=<path to the private key file>\|qos=<quality of service value>\|version=MQTT_V5\|clean-session=<true or false>\|session-expiry-interval=<interval in seconds>\|content-type=<json or SparkPlug B v1.0>`<br>**Note:**<br>For information on setting the quality of service value, refer to https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/. |

5. Select **Update**.
6. Restart the collector.

   The collector is configured.

# Chapter 22. The MQTT Sparkplug B Collector

## Overview of the MQTT Sparkplug B Collector

The MQTT Sparkplug B collector enables you to connect as a Primary Host application and collect data based on the MQTT Sparkplug B specification (Sparkplug 3.0.0). Specifically, you can subscribe to a topic that was published using the Sparkplug B payload and collect time-series data in a structured and understandable format.

In general, a topic will be published in the following format:

```
Namespace/Groupname/<Message Type>/NodeID/<DeviceID>
```

For example,

```
spBv1.0/MyAdminGroup/NBIRTH/AdminNode1/ABC-12345
```

The table below lists the message types that are sent as a topic:

| **Message type** | **Description** |
|---|---|
| NBIRTH | Message to notify the client (MQTT Sparkplug B Collector) that data is sent from a node. This message includes information such as tag information, data types, names, and alias if any. |
| NDEATH | Message to notify the client (MQTT Sparkplug B Collector) that there is no data to be sent from the node. |
| DBIRTH | Message to notify the client (MQTT Sparkplug B Collector) that data is sent from a device. This message includes information such as tag information, data types, names, and alias if any. |
| DDEATH | Message to notify the client (MQTT Sparkplug B Collector) that there are no data to be sent from the device. |
| NDATA | Message to notify the client (MQTT Sparkplug B Collector) that the actual data (metric) from the node is available to be collected. |

| Message type | Description |
|---|---|
| DDATA | Message to notify the client (MQTT Sparkplug B Collector) that the actual data (metric) from the device is available to be collected. |
| NCMD | Message to notify the Publisher that the client (MQTT Sparkplug B Collector) is receiving NDATA without receiving the NBIRTH payload. |
| DCMD | Message to notify the Publisher that the client (MQTT Sparkplug B Collector) is receiving DDATA without receiving the DBIRTH payload. |
| STATE | The collector's state message. This message is sent by the client (MQTT Sparkplug B Collector) to the Publisher through a Broker stating the status of the Collector. If the cient is active, the Publisher will send the needed data through the MQTT Broker.<br><br>Example message format:<br><br>Topic- `spBv1.0/STATE/<hostname>`<br><br>Payload-<br>`{ online : true,`<br>`timestamp : 1702879923}`<br>`#The payload is in JSON format.` |

**Supported MQTT Versions**:

- MQTT V5
- MQTT V3.1.1

**Supported data format**:

- Sparkplug B V1.0

**Topology**: The MQTT Sparkplug B collector supports a distributed model, that is, the MQTT broker, the collector, and the Historian server can be installed on the same machine or on different machines.

**Features**:

- You can subscribe to multiple-level topics based on Sparkplug B paylod using a wildcard.
- You can collect time-series data.
- You can specify a device or node from which you need to collect data from.
- You can collect the data more specifically by providing device ID, node ID, and group ID.
- You can configure tag mask as needed by selecting the interfacename, groupid, edge nodeid, deviceid along with delimiter.
- You can select what tags you want to be added in Historian by using Tags to Add mask.
- You can select what tags you do not want to be added in Historian by using Tags to Exclude mask.
- Only the unsolicited data collection is supported; polled collection is not supported.
- The timestamp resolution is seconds, milliseconds, and microseconds.
- Boolean, floating point, integer, and string data types are supported.

> **Note:**
> Although the **Recalculate** button in Historian Administrator is enabled, the functionality is not available because the MQTT collector is a non-historic collector (the source broker does not store the historical data).

**How it works**:

1. The Publisher connects to the MQTT Broker and subscribes to topics. The topics include STATE, NCMD, and DCMD.
2. The MQTT Sparkplug B collector connects to the MQTT broker and subscribes to a topic based on the Device ID and Node Id that you provide while creating an instance. The topics include STATE, NBIRTH, DBIRTH, NDATA, DDATA, NDEATH, and DDEATH.

   You can use username/password-based authentication or certificate-based authentication. Transport Layer Security (TLS) authentication is used for subscribing the data from message broker to avoid middleware attacks so that the data is securely transferred from message broker to the MQTT Sparkplug B collector.

   > **Note:**
   > It is recommended to use a certificate based authentication for a secured data transfer.

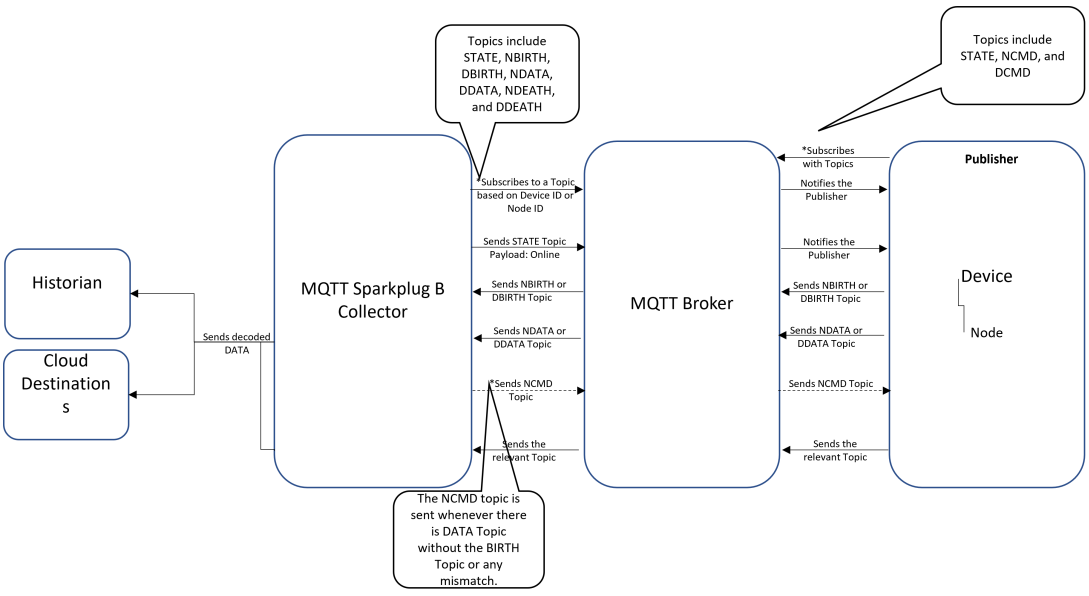
3. The MQTT Sparkplug B collector publishes the STATE message topic to the MQTT Broker.
4. The MQTT Broker then publishes this topic to the Publisher.

5. When there is data in the Device or Node, the publisher publishes a DBIRTH or NBIRTH message topic to the MQTT Broker.
6. The MQTT Broker then publishes this topic to the MQTT Sparkplug B collector.
7. The Publisher then publishes the DDATA or NDATA to the MQTT Broker.
8. The MQTT Broker publishes the data (metrics) to the MQTT Sparkplug B collector.
9. The collector converts the data from the Sparkplug B v1.0 format to a Historian-understandable format. It verifies tag availability in the Historian and, if not present, adds the tag, then adds the data samples, streaming the data to the Historian server or a cloud destination.

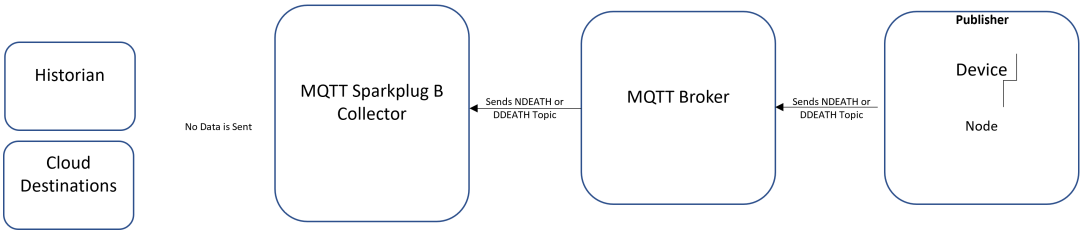
The process explains how the topics and data are published among the Publisher, MQTT Sparkplug B collector, and MQTT Broker. Similarly, when there is no data or a Device or Node is inactive, the Publisher publishes DDEATH or NDEATH message topics to the MQTT Spakplug B collector through the MQTT Broker.

**Note:**
Sometimes, the publisher might send DDATA or NDATA without sending the corresponding BIRTH message, in such cases, the MQTT Sparkplug B collect sends the DCMD or NCMD message to the MQTT Broker. The the Broker sends the message to the publisher.

Example Diagram of How Data is Sent among Publisher, Broker, and MQTT Sparkplug B Collector



Example Diagram of How DEATH Topic is Sent among Publisher, Broker, and MQTT Sparkplug B Collector




**Sparkplug B Message format**:

```
{
        "timestamp": <timestamp>,
        "metrics": [{
            "name": "<metric>",
            "alias": <alias>,
            "timestamp": <timestamp>,
            "dataType": "<data Type>",
            "value": "<value>"
        }],
        "seq": <sequence_number>
}
```

| JSON Parameter | Description |
|---|---|
| timestamp | The time at which the message was published. For example, 1642496400000 (time in milliseconds). Generally, the timestamp format is a 64-bit integer in milliseconds or microseconds since the Unix epoch. |
| metrics | The metrics or the data that was published. For example,<br>• name: BoilerTag1<br>• alias: BT1Temp<br>• timestamp: 1642496400000<br>• dataType: Float<br>• value: 80.5 |

| JSON Parameter | Description |
|---|---|
| seq | The unique sequence number of the message. The first message will have the seq as 0. This will increment by one for every other messages. |

**Supported Data Types**:

| Source Data Type | Historian Data Type |
|---|---|
| DoubleFloat, DoubleInteger, FixedByte, QuadInteger, SingleFloat | ihDoubleFloat |
| ByteString, String | ihVariableString |
| Boolean | ihBool |

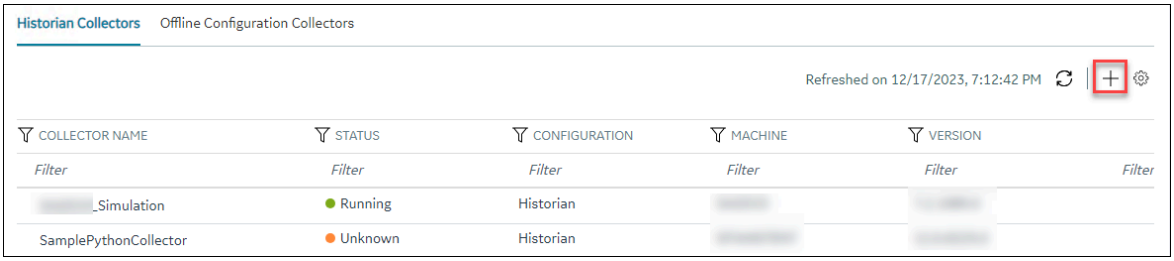
# Adding an MQTT Sparkplug B Collector Instance

## Add an MQTT Sparkplug B Collector Instance using Configuration Hub

1. Install the Historian server *(on page 104)* and collectors *(on page 142)*.
2. Ensure that you have an MQTT broker.
3. If you want to use username/password-based authentication or certificate-based authentication to connect the MQTT broker and the MQTT Sparkplug B collector, configure the authentication in the MQTT broker.
4. If you want to use certificate-based authentication, ensure that the following files are available on your collector machine:
    - CA server root file
    - Private key file
    - Client certificate file

This topic describes how to add and configure an MQTT Sparkplug B collector instance using Configuration hub. You can also add and configure an MQTT Sparkplug B collector instance using RemoteCollectorConfigurator.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**. A list of collectors in the default system appears.
3. In the upper-right corner of the main section, select .




The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.

4. In the **MACHINE NAME** field, select the machine in which you want to add a collector instance.
5. In the **COLLECTOR TYPE** field, select **MQTT Sparkplug B Collector**, and then select **Get Details**.

> **Note:**
> The **INSTALLATION DRIVE** and **BASE DATA DIRECTORY** fields cannot be changed. This is the drive location and the data directory folder that you provided during Collectors installation.

The **INSTALLATION DRIVE** and **DATA DIRECTORY** fields are populated with the drive location and the data directory folder.

6. Select **Next**.
   The **Source Configuration** section appears.
7. Enter the values as described in the following table:

| Field | Description |
|---|---|
| **BROKER CONFIGURATION** | |
| **BROKER ADDRESS** | Enter the host name of the MQTT broker using which you want to collect data. A value is required. |
| **BROKER PORT** | Enter the port number of the MQTT broker. A value is required. |
| **CLIENT ID** | Enter the client ID of the MQTT Sparkplug B collector is running. This is required if you want to send the data to a cloud destination. If you do not have a client ID set up, by default, the interface name is taken. |

| Field | Description |
|---|---|
| **PRIMARY HOST ID** | Enter the unique host ID of the Collector. The Collector will publish the STATE message topic using this host ID and then the Publisher will subscribe and start publishing the topics to this host ID. |
| **REORDER TIMEOUT** | Enter the duration for waiting before sending a CMD message if a sequence is skipped. You can enter the duration in milliseconds. |
| **MQTT VERSION** | Select the version of the MQTT that you want to use. The following versions are supported:<br>◦ MQTT_V311<br>◦ MQTT_V5 |

**TOPIC**: The parameters that need to be included in the topic:

```
Namespace/Groupname/<Message Type>/NodeID/<DeviceID>
```

You can also use wildcards in the **GROUP ID**, **EDGE NODE ID**, and **DEVICE ID** fields. The following wildcards are supported:

- **+** (single-level wildcard): Supported for all the three fields.
- **#** (Multi-level wildcard): Supported for the **EDGE NODE ID** and **DEVICE ID**

**+** (Single-level wildcard): Can be used to subscribe to only one topic level. For example, if you subscribe to a topic `<Admin>/+/<ABC-123>`, you will receive messages from all the nodes corresponding to the group and device. That is,

```
<Admin>/Node1/<ABC-123>
<Admin>/Node2/<ABC-123>
<Admin>/Node3/<ABC-123>
...
<Admin>/Noden/<ABC-123>
```

# (Multi-level wildcard): Can be used to subscribe to any number of levels within a topic. For example, if you subscribe to a topic `<Admin>/#`, you will receive messages from all the nodes and devices corresponding to the group,

| Field | Description |
|---|---|
| <Admin>/Node1/<ABC-123><br><Admin>/Node2/<ABC-123><br><Admin>/Node3/<ABC-123><br><Admin>/Node1/<ABC-124<br><Admin>/Node2/<ABC-124><br>...<br><Admin>/Noden/<Devicen> | |
| **GROUP ID** | Enter the Sparkplug B group name to which you want your collector to subscribe. If this is empty along with the other fields below **TOPIC**, the collector will subscribe to all the available groups, nodes, and devices. |
| **EDGE NODE ID** | Enter the Sparkplug B edge node ID to which you want your collector to subscribe. If this is empty, then the Collector will subscribe to all the edge nodes corresponding to the entered **GROUP ID**. If the **GROUP ID** and **DEVICE ID** are also empty, then the collector will subscribe to all the available groups, nodes and devices. |
| **DEVICE ID** | Enter the Sparkplug B device name. If this is empty, the collector subscribes to node messages if a **NODE ID** is entered, otherwise, if a **DEVICE ID** is entered, it subscribes to device messages. |
| **TAG CONFIGURATION** | |
| **TAG NAME PREFIX FORMAT** | |
| **ELEMENT** | Enter a prefix to be included in the tag. By using this field, you can clearly identify a tag. For example, you can clearly differentiate the tags that are collected.<br><br>The following options are available: |

| Field | Description |
|---|---|
| | ◦ **<interfacename>**<br>◦ **<groupid>**<br>◦ **<edgenodeid>**<br>◦ **<deviceid>**<br><br>For example, if all four fields are provided and the interface/collector name is "sparkplug1" and the Topic contains group id = g1 edge node id = n1, device id = d1 then device tags will be created in Historian as “sparkplug1.g1.n1.d1.tag1”. |
| **DELIMITER** | Enter a delimiter you need to be included in the tag. You can use any special characters as delimiter. However, it is recommended that you use a delimiter that is ideal and clear to be identified. For example, "/", ".", "_".<br><br>**Note:**<br>"?" and "*" are not allowed. |
| **PREVIEW** | The preview of how the tags will be created and stored based on the **TAG PREFIX** and the **DELIMTER** that you selected. |
| **TAG MASK** | |
| **TAGS TO ADD** | Provide a mask along with wildcard to collect those tags that include the mask you provided and store in Historian. For example, *Pressure*. This will collect all the tags that begin with "Pressure". If you enter Pressure*, all the tags that end with "Pressure" will be collected. Similarly, if you enter *Pres?, all the tags that contain "pres" at the beginning will be collected. It can be "Pressure", "Press", or "Pres1". |

| Field | Description |
|---|---|
| | **Note:** Whenever a new tag is collected, the collector verifies the tag availability in the Historian and, if not present, adds the tag, then adds the data samples, streaming the data to the Historian server or a cloud destination. |
| **TAGS TO EXCLUDE** | Provide a mask along with wildcard to exclude those tags that include the mask you provided. For example, *Pressure*. This will exclude all the tags that begin with "Pressure". If you enter Pressure*, all the tags that end with "Pressure" will be excluded. Similarly, if you enter *Pres?, all the tags that contain "pres" at the beginning will be excluded. It can be "Pressure", "Press", or "Pres1". |
| **AUTHENTICATION** | |
| **USER CREDENTIALS** | |
| **USERNAME** | Enter the username to connect to the MQTT broker. A value is required if you have configured username/password-based authentication in the MQTT broker. |
| **PASSWORD** | Enter the password to connect to the MQTT broker. A value is required if you have configured username/password-based authentication in the MQTT broker. |
| **SSL/TLS** | |
| **CA SERVER ROOT FILE** | Enter the path to the CA server root file to connect to the MQTT broker. A value is required if you have configured certificate-based authentication in the MQTT broker. |

| Field | Description |
|---|---|
| **PRIVATE KEY FILE** | Enter the path to the private key file to connect to the MQTT broker. A value is required if you have configured certificate-based authentication in the MQTT broker. |
| **CLIENT CERTIFICATE FILE** | Enter the path to the client certificate file to connect to the MQTT broker. A value is required if you have configured certificate-based authentication in the MQTT broker. |

8. Select **Next**.
   The **Destination Configuration** section appears.

   Under **CHOOSE DESTINATION**, the **Historian Server** option is selected by default. In addition, the **DESTINATION HISTORIAN SERVER** field is disabled and populated with the collector machine name.

9. Select the destination to which you want to send data, and then enter the values in the corresponding fields. You can send data to an on-premises Historian server or to a cloud destination.

   a. If you need to send data to a cloud destination, select the cloud destinations as needed.
      - **Predix Timeseries**- Select this if you need to send data to Predix cloud. For more information, refer to Predix Cloud *(on page 621)*.
      - **Azure IoT Hub**- Select this if you need to send data to Azure Cloud in KairosDB format. For more information, refer to Azure IoT Hub (KairosDB format) *(on page 608)*.
      - **MQTT**- Select this if you need to send data to any of the following cloud destination.
        - Alibaba cloud. For more information, refer to Alibaba Cloud *(on page 587)*.
        - AWS cloud. For more information, refer to AWS Cloud *(on page 594)*.
        - Google cloud. For more information, refer to Google Cloud *(on page 614)*.

   b. If you need to send data to an on-premises Historian server, select **Historian Server**.
      If you created security groups or enabled a strict client/collector authentication, enter the **USERNAME** and **PASSWORD** of the on-premises Historian server that you created during the installation of the collector.

If you entered the **USERNAME**and **PASSWORD**, select **Test Connection**. This will help you to test if the Historian server that you are trying to connect is valid or if the credentials that you entered are valid.
If the entered credentials are valid, a successful connection message appears.

10. After you selected the destination, select **Next**.
    The **Collector Initiation** section appears.
11. If needed, modify the value in the **COLLECTOR NAME** field.

    > **Note:**
    > If you will be using the collector in Historian Administrator, the **COLLECTOR NAME** must include Sparkplug B in it.

    The value that you enter:
    - Must be unique.
    - Must not exceed 15 characters.
    - Must not contain a space.
    - Must not contain special characters except a hyphen, period, and an underscore.
12. In the **RUNNING MODE** field, select one of the following options.
    - **Service - Local System Account**: Select this option if you want to run the collector as a Windows service using the credentials of the local user (that is, the currently logged-in user). If you select this option, the **USERNAME** and **PASSWORD** fields are disabled.
    - **Service Under Specific User Account**: Select this option if you want to run the collector as a Windows service using a specific user account. If you select this option, you must enter values in the **USERNAME** and **PASSWORD** fields.

      If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:
      - iH Security Admins
      - iH Collector Admins
      - iH Tag Admins

    You can also configure the collector to start automatically when you start the computer.
13. Select **Add**.
    The collector instance is added. The fields specific to the collector section appear in the **DETAILS** section.
14. If needed, enter values in available fields *(on page 579)*.

15. In the upper-left corner of the page, select **Save**.
16. If needed, restart the collector.

- Specify the tags whose data you want to collect using the collector. In the **CHOOSE CONFIGURATION** field,
  - If you have selected **Historian Configuration**, specify the tags using Configuration Hub *(on page 357)*.
  - If you have selected **Offline Configuration**, modify the offline configuration file of the collector. By default, the file is available in the following location: `<installation folder of Historian>\GE Digital\<collector name>`. For information, refer to Offline Configuration for Collectors *(on page 2039)*. This option is applicable only if you have selected a cloud destination.
- If needed, you can configure the collector instance *(on page 2331)*.

## Add an MQTT Sparkplug B Collector Instance using RemoteCollector Configurator

1. Install the Historian server *(on page 104)* and collectors *(on page 142)*.
2. Ensure that you have an MQTT broker.
3. If you want to use username/password-based authentication or certificate-based authentication to connect the MQTT broker and the MQTT Sparkplug B collector, configure the authentication in the MQTT broker.
4. If you want to use certificate-based authentication, ensure that the following files are available on your collector machine:
   - CA server root file
   - Private key file
   - Client certificate file

This topic describes how to add and configure an MQTT Sparkplug B collector instance using RemoteCollectorConfigurator utility. If you want to add an offline collector instance, refer to Add an Offline Collector Instance *(on page 819)*.

1. Run the `RemoteCollectorConfigurator.exe` file. By default, it is located in the following folder: `C:\Program Files\GE Digital\NonWebCollectorInstantiationTool`.
   A list of options to manage collector instances appears.
2. Connect to the collector machine by entering `1` or `2`, depending on whether collectors are installed locally or on a remote machine.

3. To create a collector instance using the RemoteCollectorConfigurator utility, you need to provide parameters in a JSON format. For this you can create a sample JSON using this tool. To create sample JSON, enter `7`.
4. To create a sample JSON file for MQTT Sparkplug B collector, enter **21**.
5. To set Historian as the destination type, enter **1**.
6. Enter the folder path where you want the sample JSON file to be created. By default, the file will be created in the same folder in which the RemoteCollectorConfigurator utility is located.
   A sample JSON file with name "**SampleSparkplugB.json"** is created in the given location.

   **Sample JSON File**

```
{
"CollectorSystemName":"TESTSYSTEM",
"DestinationHistorian":"TESTSYSTEM",
"General1":"HostName=localhost|Port=|ClientId=|PrimaryHostId=|GroupId=test|EdgeNodeId=#|DeviceId=|Username=|Pas
sword=|Cafile=|Certfile=|PrivateKeyfile=|Version=MQTT_V311|ReorderTimeout=",
"General2":"<interfacename>.<groupid>.<edgenodeid>.<deviceid>.",
"General3":"*",
"General4":"",
"General5":"",
"InterfaceDescription":"Sample SparkplugB Collector",
"InterfaceName":"SampleSparkplugBCollector",
"InterfaceSubType":"SparkplugB",
"SourceHistorianUserName":"",
"SourceHistorianPassword":"",
"Type":"16",
"SamplerId":"",
"MTLSEnable":"false",
"DataPathDirectory":"C:\\Proficy Historian Data",
"CollectorDestination":"Historian",
"DestinationHistorianUserName":"",
"DestinationHistorianPassword":"",
"mode":"1",
"winUserName":"",
"winPassword":""
}
```

7. Configure the values as specified the following table:

**Table 381.**

| Field | Descriprion |
|---|---|
| CollectorSystemName (Mandatory) | Name of the machine on which the Collector is installed. |
| DestinationHistorian (Mandatory) | Name of the destination Historian machine. |
| General1 | |
| HostName (mandatory) | MQTT broker Ip or hostname. |
| Port (mandatory) | MQTT broker port. |
| ClientID | Unique name to identify the client. |
| PrimaryHostID (mandatory) | The unique host ID of the Collector. The Collector will publish the STATE message topic using this host ID and then the Publisher will subscribe and start publishing the topics to this host ID. |
| GroupId (mandatory) | The Sparkplug B group name to which you want your collector to subscribe. If this is empty along with the other fields below **TOPIC**, the collector will subscribe to all the available groups, nodes, and devices. Use of the wildcard "+" is supported. |
| EdgeNodeId (mandatory) | The Sparkplug B edge node ID to which you want your collector to subscribe. If this is empty, then the Collector will subscribe to all the edge nodes corresponding to the entered **GROUP ID**. If the **GROUP ID** and **DEVICE ID** are also empty, then the collector will subscribe to all the available groups, nodes and devices. Use of the wildcards "+" and "#" are supported. |
| DeviceId (mandatory) | The Sparkplug B device name. If this is empty, the collector subscribes to node messages if a **NODE ID** is entered, otherwise, if a **DEVICE ID** is |

| Field | Descriprion |
|---|---|
| | entered, it subscribes to device messages. Use of the wildcards "+" and "#" are supported. |
| Username | The username to connect to the MQTT broker. A value is required if you have configured username/password-based authentication in the MQTT broker. |
| Password | The password to connect to the MQTT broker. A value is required if you have configured username/password-based authentication in the MQTT broker. |
| Cafile | The path to the CA server root file to connect to the MQTT broker. A value is required if you have configured certificate-based authentication in the MQTT broker. |
| Certfile | The path to the client certificate file to connect to the MQTT broker. A value is required if you have configured certificate-based authentication in the MQTT broker. |
| PrivateKeyfile | The path to the private key file to connect to the MQTT broker. A value is required if you have configured certificate-based authentication in the MQTT broker. |
| ReorderTimeout | Enter the duration for waiting before sending a CMD message if a sequence is skipped. You can enter the duration in milliseconds. |
| Version | The version of the MQTT that you want to use. That is, MQTT_V311 or MQTT_V5. |
| General2 | |
| <interfacename>.<groupid>.<edgenodeid>.<deviceid>. | The prefix to be included in the tag. By using this field, you can clearly identify a tag. For example, you can clearly differentiate the tags that are collected. |

| Field | Descriprion |
|---|---|
| | The following options are available:<br>◦ **<interfacename>**<br>◦ **<groupid>**<br>◦ **<edgenodeid>**<br>◦ **<deviceid>**<br>For example, if all four fields provided and the interface / collector name is "sparkplug1" and the Topic contains group id = g1 edge node id = n1, device id = d1 then device tags will be created in Historian as “sparkplug1.g1. n1.d1.tag1”. |
| General3 | |
| "*" Tag include mask field | The mask along with wildcard to collect those tags that include the mask you provided and store in Historian. For example, *Pressure*. This will collect all the tags that begin with "Pressure". If you enter Pressure*, all the tags that end with "Pressure" will be collected. Similarly, if you enter *Pres?, all the tags that contain "pres" at the beginning will be collected. It can be "Pressure", "Press", or "Pres1".<br>**Note:**<br>Whenever a new tag is collected, the collector verifies the tag availability in the Historian and, if not present, adds the tag, then adds the data samples, streaming the data to the Historian server or a cloud destination. |
| General4 | |
| "" Tag exclude mask field | The mask along with wildcard to exclude those tags that include the mask you provided. For example, *Pressure*. This will exclude all the |

| Field | Descriprion |
|---|---|
| | tags that begin with "Pressure". If you enter Pressure*, all the tags that end with "Pressure" will be excluded. Similarly, if you enter *Pres?, all the tags that contain "pres" at the beginning will be excluded. It can be "Pressure", "Press", or "Pres1". |
| General5 | |
| InterfaceDescription | A description of the collector instance. |
| InterfaceName | A name of the collector instance.<br> **Note:**<br>If you will use this collector on Historian Administrator, the name must include Sparkplug B in it. |
| InterfaceSubType | Sparkplug B. **Do not change this field**. |
| SourceHistorianUserName (optional) | username of the source Historian. |
| SourceHistorianPassword (optional) | password of the source Historian. |
| Type | 16. **Do not change this field**. |
| SamplerId (optional) | The ID of the sampler. |
| MTLSEnable | False. |
| DataPathDirectory | The directory path where you want the data to be stroed. It is recommended to use C:\\Proficy Historian Data. |
| CollectorDestination | Historian. **Do not change this field**. |
| DestinationHistorianUserName (optional) | username of the destination Historian. |
| DestinationHistorianPassword (optional) | password of the destination Historian. |
| Mode | 1. **Do not change this field**. |
| winUserName (optional) | username of the window user. |
| winPassword (optional) | password of the window user. |

> **Note:**
> Do not change the case of the keys in the JSON file.

8. Run the `RemoteCollectorConfigurator.exe` file. By default, it is located in the following folder: `C:\Program Files\GE Digital\NonWebCollectorInstantiationTool`.
   A list of options to manage collector instances appears.
9. Connect to the collector machine by entering `1` or `2`, depending on whether collectors are installed locally or on a remote machine.
10. To create the collector instance, enter `4`.
11. To create the instance using the updated JSON file, enter `2`.
12. Enter the location where the updated JSON file is saved.
13. On the keyboard, press **Enter**.
    The MQTT Sparkplug B collector instance is created.

- Specify the tags *(on page 357)* whose data you want to collect using the collector.
- If you did not enter a value, modify the offline configuration file of the collector. By default, this file is available in the following location: `<installation folder of Historian>\GE Digital \<collector name>`. For information, refer to Creating Offline Configuration XML file *(on page 2040)*.

# Configuring an MQTT Sparkplug B Collector Instance

## Configure an MQTT Sparkplug B Collector Instance using Configuration Hub

This topic describes how to configure an MQTT Sparkplug B collector instance using Configuration Hub.

- Ensure that you installed the Historian server *(on page 104)* and collectors *(on page 142)*.
- Ensure that you created an MQTT Sparkplug B collector instance, using RemoteCollectorConfigurator *(on page 2325)* or Configuration Hub *(on page 519)*.

1. Access Configuration Hub *(on page 336)*.
2. Select **Collectors**, and then select the MQTT Sparkplug B collector instance that you want to configure.
   The configurations specific to the collector instance appear in the **DETAILS** section.
3. In the **COLLECTOR SPECIFIC CONFIGURATION** and **INSTANCE CONFIGURATION** sections, configure the values as described in the following table.

**COLLECTOR SPECIFIC CONFIGURATION**

| Field | Description |
|---|---|
| **MTLS Security** | Indicates whether you want to use Mutual TLS (MTLS) protocol to enforce a secure and strong authentication mechanism. |
| **MTLS Data Encryption** | Indicates whether you want to encrypt the data that the collector shares to the data archiver (DA). |
| For more information on how to enable MTLS Security, refer to Enable MTLS Security *(on page 632)*. | |

**INSTANCE CONFIGURATION**

| Field | Description |
|---|---|
| **Broker Address** | Enter the host name of the MQTT broker using which you want to collect data. A value is required. |
| **Broker Port** | Enter the port number of the MQTT broker. A value is required. |
| **Client ID** | Enter the client ID of the MQTT Sparkplug B collector is running. This is required if you want to send the data to a cloud destination. If you do not have a client ID set up, by default, the interface name is taken. |
| **Primary Host ID** | Enter the unique host ID of the Collector. The Collector will publish the STATE message topic using this host ID and then the Publisher will subscribe and start publishing the topics to this host ID. |
| **REORDER TIMEOUT** | Enter the duration for waiting before sending a CMD message if a sequence is skipped. You can enter the duration in milliseconds. |

| Field | Description |
|---|---|
| **Group ID** | Enter the Sparkplug B group name to which you want your collector to subscribe. If this is empty along with the other fields below **TOPIC**, the collector will subscribe to all the available groups, nodes, and devices. |
| **Edge Node ID** | Enter the Sparkplug B edge node ID to which you want your collector to subscribe. If this is empty, then the Collector will subscribe to all the edge nodes corresponding to the entered **GROUP ID**. If the **GROUP ID** and **DEVICE ID** are also empty, then the collector will subscribe to all the available groups, nodes and devices. |
| **Device ID** | Enter the Sparkplug B device name. If this is empty, the collector subscribes to node messages if a **NODE ID** is entered, otherwise, if a **DEVICE ID** is entered, it subscribes to device messages. |

You can also use wildcards in the **GROUP ID**, **EDGE NODE ID**, and **DEVICE ID** fields. The following wildcards are supported:

- **+** (single-level wildcard): Supported for all the three fields.
- **#** (Multi-level wildcard): Supported for the **EDGE NODE ID** and **DEVICE ID**

**+** (Single-level wildcard): Can be used to subscribe to only one topic level. For example, if you subscribe to a topic `<Admin>/+/<ABC-123>`, you will receive messages from all the nodes corresponding to the group and device. That is,

```
<Admin>/Node1/<ABC-123>
<Admin>/Node2/<ABC-123>
<Admin>/Node3/<ABC-123>
...
<Admin>/Noden/<ABC-123>
```

# (Multi-level wildcard): Can be used to subscribe to any number of levels within a topic. For example, if you subscribe to a topic `<Admin>/#`, you will receive messages from all the nodes and devices corresponding to the group,

| Field | Description |
|---|---|
| <Admin>/Node1/<ABC-123><br><Admin>/Node2/<ABC-123><br><Admin>/Node3/<ABC-123><br><Admin>/Node1/<ABC-124<br><Admin>/Node2/<ABC-124><br>...<br><Admin>/Noden/<Devicen> | |
| **Tags to Add** | Provide a mask along with wildcard to collect those tags that include the mask you provided and store in Historian. For example, *Pressure*. This will collect all the tags that begin with "Pressure". If you enter Pressure*, all the tags that end with "Pressure" will be collected. Similarly, if you enter *Pres?, all the tags that contain "pres" at the beginning will be collected. It can be "Pressure", "Press", or "Pres1".<br><br>**Note:**<br>Whenever a new tag is collected, the collector verifies the tag availability in the Historian and, if not present, adds the tag, then adds the data samples, streaming the data to the Historian server or a cloud destination. |
| **Tags to Exclude** | Provide a mask along with wildcard to exclude those tags that include the mask you provided. For example, *Pressure*. This will exclude all the tags that begin with "Pressure". If you enter Pressure*, all the tags that end with "Pressure" will be excluded. Similarly, if you enter *Pres?, all the tags that contain "pres" at the beginning will be excluded. It can be "Pressure", "Press", or "Pres1". |

| Field | Description |
|---|---|
| **Tag Prefix** | Enter a prefix to be included in the tag. By using this field, you can clearly identify a tag. For example, you can clearly differentiate the tags that are collected.<br><br>The following options are available:<br><br>◦ **<interfacename>**<br>◦ **<groupid>**<br>◦ **<edgenodeid>**<br>◦ **<deviceid>**<br><br>For example, if all four fields provided and the interface / collector name is "sparkplug1" and the Topic contains group id = g1 edge node id = n1, device id = d1 then device tags will be created in Historian as “sparkplug1.g1. n1.d1.tag1”. |
| **Username** | Enter the username to connect to the MQTT broker. A value is required if you have configured username/password-based authentication in the MQTT broker. |
| **Password** | Enter the password to connect to the MQTT broker. A value is required if you have configured username/password-based authentication in the MQTT broker. |
| **CA Server Root File** | Enter the path to the CA server root file to connect to the MQTT broker. A value is required if you have configured certificate-based authentication in the MQTT broker. |
| **Private Key File** | Enter the path to the private key file to connect to the MQTT broker. A value is required if you have configured certificate-based authentication in the MQTT broker. |
| **Client Certificate File** | Enter the path to the client certificate file to connect to the MQTT broker. A value is required |

| Field | Description |
|---|---|
| | if you have configured certificate-based authentication in the MQTT broker. |
| **MQTT Version** | Select the version of the MQTT that you want to use. The following versions are supported:<br>◦ MQTT_V311<br>◦ MQTT_V5 |

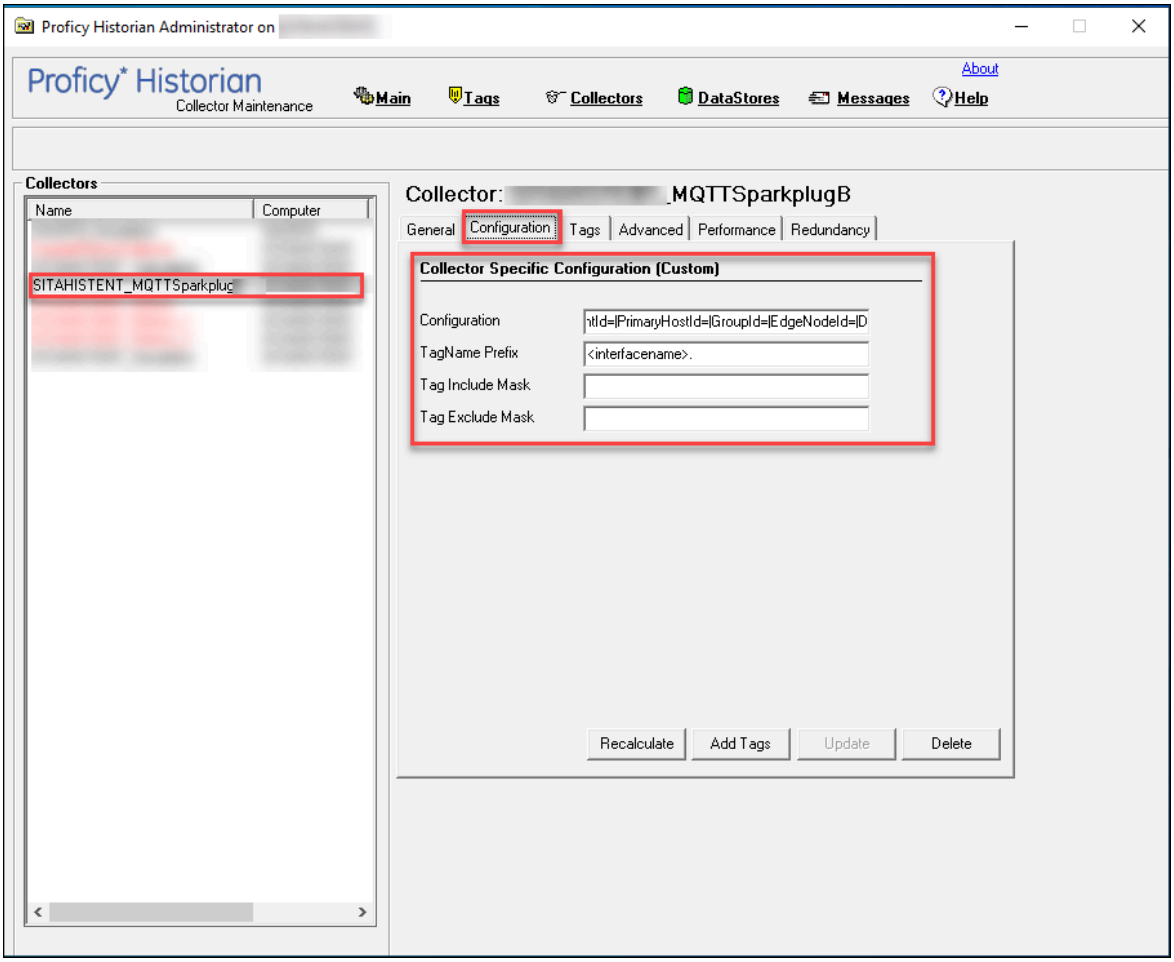
4. As needed, enter values in the other sections common to all collectors *(on page 579)*.
5. Restart the collector.
   The collector instance is configured.

## Configure an MQTT Sparkplug B Collector Instance using Historian Administrator

- Ensure that you installed the Historian server *(on page 104)* and collectors *(on page 142)*.
- Ensure that you created an MQTT Sparkplug B collector instance, using RemoteCollectorConfigurator *(on page 2325)* or Configuration Hub *(on page 519)*.

This topic describes how to configure an added MQTT Sparkplug B collector instance in Historian Administrator.

1. Access Historian Administrator *(on page 823)*.
2. Select **Collectors**, and then select the MQTT Sparkplug B collector instance that you want to configure.




3. In the right-side, below the **Collector: <Name of the Collector Instance>**, select **Configuration**. The fields specific to the collector instance appear.
4. Enter values as specified in the following table.

**Table 382.**

| | |
|---|---|
| **Configuration**: HostName=localhost\|Port=\|ClientId=\|PrimaryHostId=\|GroupId=\|EdgeNodeId=\|DeviceId=\|username=\|password=\|cafile=\|certfile=\|privatekeyfile=\|version=MQTT_V311\|reordertimeout= | |
| HostName (mandatory) | MQTT broker Ip or hostname. |
| Port (mandatory) | MQTT broker port. |
| ClientID | Unique name to identify the client. |
| PrimaryHostID (mandatory) | The unique host ID of the Collector. The Collector will publish the STATE message topic using this host ID and then the Publisher will sub- |

| | |
|---|---|
| | scribe and start publishing the topics to this host ID. |
| GroupId (mandatory) | The Sparkplug B group name to which you want your collector to subscribe. If this is empty along with the other fields below **TOPIC**, the collector will subscribe to all the available groups, nodes, and devices. Use of the wildcard "+" is supported. |
| EdgeNodeId (mandatory) | The Sparkplug B edge node ID to which you want your collector to subscribe. If this is empty, then the Collector will subscribe to all the edge nodes corresponding to the entered **GROUP ID**. If the **GROUP ID** and **DEVICE ID** are also empty, then the collector will subscribe to all the available groups, nodes and devices. Use of the wildcards "+" and "#" are supported. |
| DeviceId (mandatory) | The Sparkplug B device name. If this is empty, the collector subscribes to node messages if a **NODE ID** is entered, otherwise, if a **DEVICE ID** is entered, it subscribes to device messages. Use of the wildcards "+" and "#" are supported. |
| username | The username to connect to the MQTT broker. A value is required if you have configured username/password-based authentication in the MQTT broker. |
| password | The password to connect to the MQTT broker. A value is required if you have configured username/password-based authentication in the MQTT broker. |
| cafile | The path to the CA server root file to connect to the MQTT broker. A value is required if you have configured certificate-based authentication in the MQTT broker. |

| | |
|---|---|
| certfile | The path to the client certificate file to connect to the MQTT broker. A value is required if you have configured certificate-based authentication in the MQTT broker. |
| privatekeyfile | The path to the private key file to connect to the MQTT broker. A value is required if you have configured certificate-based authentication in the MQTT broker. |
| version | The version of the MQTT that you want to use. That is, MQTT_V311 or MQTT_V5. |
| reordertimeout | |
| TagName Prefix | The prefix to be included in the tag. By using this field, you can clearly identify a tag. For example, you can clearly differentiate the tags that are collected.<br><br>The following options are available:<br><br>◦ **<interfacename>**<br>◦ **<groupid>**<br>◦ **<edgenodeid>**<br>◦ **<deviceid>**<br><br>For example, if all four fields provided and the interface / collector name is "sparkplug1" and the Topic contains group id = g1 edge node id = n1, device id = d1 then device tags will be created in Historian as “sparkplug1.g1. n1.d1.tag1”. |
| Tag include Mask | The mask along with wildcard to collect those tags that include the mask you provided and store in Historian. For example, *Pressure*. This will collect all the tags that begin with "Pressure". If you enter Pressure*, all the tags that end with "Pressure" will be collected. Similarly, if you enter *Pres?, all the tags that con- |

| | |
|---|---|
| | tain "pres" at the beginning will be collected. It can be "Pressure", "Press", or "Pres1".<br><br>**Note:**<br>Whenever a new tag is collected, the collector verifies the tag availability in the Historian and, if not present, adds the tag, then adds the data samples, streaming the data to the Historian server or a cloud destination. |
| Tag Exclude Mask | The mask along with wildcard to exclude those tags that include the mask you provided. For example, *Pressure*. This will exclude all the tags that begin with "Pressure". If you enter Pressure*, all the tags that end with "Pressure" will be excluded. Similarly, if you enter *Pres?, all the tags that contain "pres" at the beginning will be excluded. It can be "Pressure", "Press", or "Pres1". |

5. After you configure the collector instance, select **Update**.
   The instance is updated with the new configuration.

# Chapter 23. The ODBC Collector

## Overview of the ODBC Collector

The ODBC collector collects data from an application based on an ODBC driver and stores the data in an on-premises Historian Server or a cloud destination. It supports collecting of all the Historian supported data types of data from the ODBC server.

**Topology:** The ODBC collector supports a distributed model, where the ODBC server, the collector, and the Historian server are installed on different machines. Typically, however, the collector is installed on the same machine as the ODBC server and sends data to a remote Historian server.

**Features:**

- You can browse the source for tags and their attributes on an ODBC server that supports browsing.
- Only the unsolicited data collection is supported; when changes to the ODBC source tags are detected, they are sent to the Historian server. The minimum poll interval is 100ms. The collector duplicates raw samples from the ODBC server into the Historian data archive.
- The supported timestamp resolution is 1ms.
- Floating point, integer, and string data are supported.

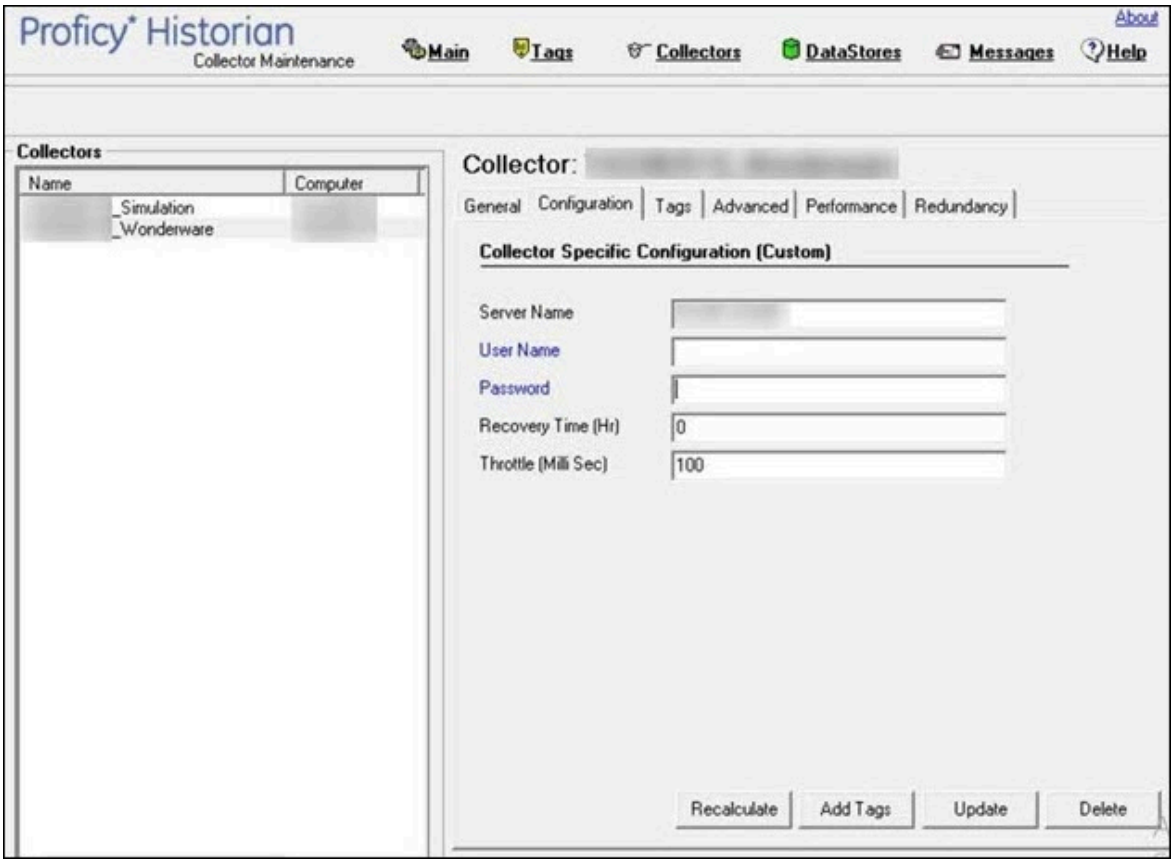
**Supported Data Attributes:**

| Historian Data Type | ODBC Server Data Type |
|---|---|
| ihByte | Byte |
| ihFloat | SingleFloat |
| ihDoubleFloat | DoubleFloat |
| ihInteger | SingleInteger |
| ihDoubleInteger | DoubleInteger |
| ihScaled | Not applicable |
| ihFixedString | Not applicable |
| ihVariableString | Not applicable |
| ihBlob | Not applicable |
| ihTime | Not applicable |

| Historian Data Type | ODBC Server Data Type |
|---|---|
| ihInt64 | Not applicable |
| ihUInt64 | Not applicable |
| ihUInt32 | Not applicable |
| ihUInt16 | Not applicable |
| ihBool | Not applicable |

**Limitations:**

- A single collector instance can collect data from a single ODBC server. To collect data from multiple ODBC servers, you must add multiple instances.
- Only good and bad quality types are supported. OPC Quality and OPC Subquality are not supported.
- If you want a domain user to use the ODBC collector, after you add an instance of a collector, when you later configure it, do not provide values in the **User Name** and **Password** fields. This is because the ODBC driver uses Windows authentication.



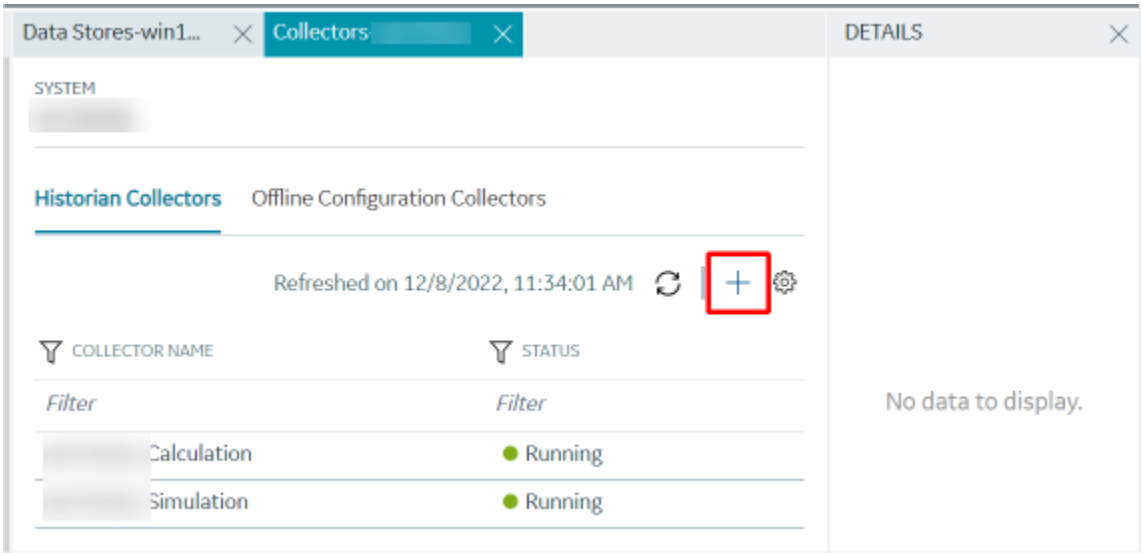
# Configuration

## Add and Configure an ODBC Collector Using Configuration Hub

The ODBC collector collects data from an application based on an ODBC driver. For more information, refer to Overview of the ODBC Collector *(on page 2341)*.

This topic describes how to add a collector instance using Configuration Hub. You can also add a collector instance using the RemoteCollectorConfigurator utility *(on page 797)*, which does not require you to install Web-based Clients.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.
   A list of collectors in the default system appears.
3. In the upper-right corner of the main section, select +.




   The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.
4. In the **MACHINE NAME** field, select the machine in which you want to add a collector instance.
5. In the **COLLECTOR TYPE** field, select **ODBC Collector**, and then select **Get Details**.
   The **INSTALLATION DRIVE** and **DATA DIRECTORY** fields are disabled and populated.
6. Select **Next**.
   The **Source Configuration** section appears.
7. Enter values as described in the following table.

| Field | Description |
|---|---|
| **ODBC SERVER** | Enter the host name or IP address of the ODBC server from which you want to collect data. A value is required. |
| **USERNAME** | Enter the username to connect to the ODBC server. A value is required. |
| **PASSWORD** | Enter the password to connect to the ODBC server. A value is required. |

8. Select **Next**.
   The **Destination Configuration** section appears. The collector machine name provided by you is selected as the **Source Configuration** by default.

   Under **CHOOSE DESTINATION**, the **Historian Server** option is selected by default. In addition, the **DESTINATION HISTORIAN SERVER** field is disabled and populated with the collector machine name.

9. Select the destination to which you want to send data, and then enter the values in the corresponding fields. You can send data to an on-premises Historian server or to a cloud destination.

   a. If you need to send data to a cloud destination, select the cloud destinations as needed.
      - **Predix Timeseries**- Select this if you need to send data to Predix cloud. For more information, refer to Predix Cloud *(on page 621)*.
      - **Azure IoT Hub**- Select this if you need to send data to Azure Cloud in KairosDB format. For more information, refer to Azure IoT Hub (KairosDB format) *(on page 608)*.
      - **MQTT**- Select this if you need to send data to any of the following cloud destination.
        - Alibaba cloud. For more information, refer to Alibaba Cloud *(on page 587)*.
        - AWS cloud. For more information, refer to AWS Cloud *(on page 594)*.
        - Google cloud. For more information, refer to Google Cloud *(on page 614)*.

   b. If you need to send data to an on-premises Historian server, select **Historian Server**.
      If you created security groups or enabled a strict client/collector authentication, enter the **USERNAME** and **PASSWORD** of the on-premises Historian server that you created during the installation of the collector.

If you entered the **USERNAME**and **PASSWORD**, select **Test Connection**. This will help you to test if the Historian server that you are trying to connect is valid or if the credentials that you entered are valid.
If the entered credentials are valid, a successful connection message appears.

10. Select **Next**.
    The **Collector Initiation** section appears.
11. If needed, modify the value in the **COLLECTOR NAME** field.
    The value that you enter:
    - Must be unique.
    - Must contain the string `ODBC`.
    - Must not exceed 15 characters.
    - Must not contain a space.
    - Must not contain special characters except a hyphen, period, and an underscore.
12. In the **RUNNING MODE** field, select one of the following options.
    - **Service - Local System Account**: Select this option if you want to run the collector as a Windows service using the credentials of the local user (that is, the currently logged-in user). If you select this option, the **USERNAME** and **PASSWORD** fields are disabled.
    - **Service Under Specific User Account**: Select this option if you want to run the collector as a Windows service using a specific user account. If you select this option, you must enter values in the **USERNAME** and **PASSWORD** fields.

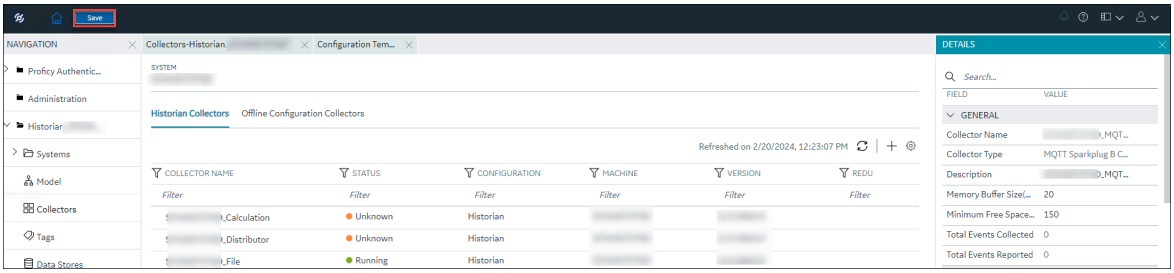
      If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:
      - iH Security Admins
      - iH Collector Admins
      - iH Tag Admins

    You can also configure the collector to start automatically when you start the computer.
13. Select **Add**.
    The collector instance is added. The fields specific to the collector appear in the **DETAILS** section.
14. In the **COLLECTOR SPECIFIC CONFIGURATION** section, configure values as described in the following table.

| Field | Description |
|---|---|
| **Recovery Time (hours)** | Enter the maximum time, in hours, for which the collector will attempt to recover data after |

| Field | Description |
|---|---|
| | the collector is started or when connection between the collector and the ODBC server is re-established. This time is calculated as the duration between the current time and the last known write time.<br><br>Continuous data collection is resumed only after the previous data has been recovered.<br><br>By default, this value is set to 0, which means data recovery is not attempted. The maximum value you can provide is 168 hours (that is, 7 days). |
| **Throttle (Milliseconds)** | Enter the frequency, in milliseconds, at which you want the ODBC collector to query the ODBC server for tag data. This will minimize the load on the ODBC server. You can enter a value up to 16 hours.<br><br>**Note:**<br>If this field is blank, enter the required minimum value of 100 milliseconds. |
| **MTLS Security** | Indicates whether you want to use Mutual TLS (MTLS) protocol to enforce a secure and strong authentication mechanism. |
| **MTLS Data Encryption** | Indicates whether you want to encrypt the data that the collector shares to the data archiver (DA). |
| For more information on how to enable MTLS Security, refer to Enable MTLS Security *(on page 632)*. | |

15. As needed, enter values in the other sections common to all collectors *(on page 579)*.
16. In the upper-left corner of the page, select **Save**.




The changes to the collector instance are saved.

17. If needed, restart the collector.

Specify the tags whose data you want to collect using the collector. In the **CHOOSE CONFIGURATION** field,

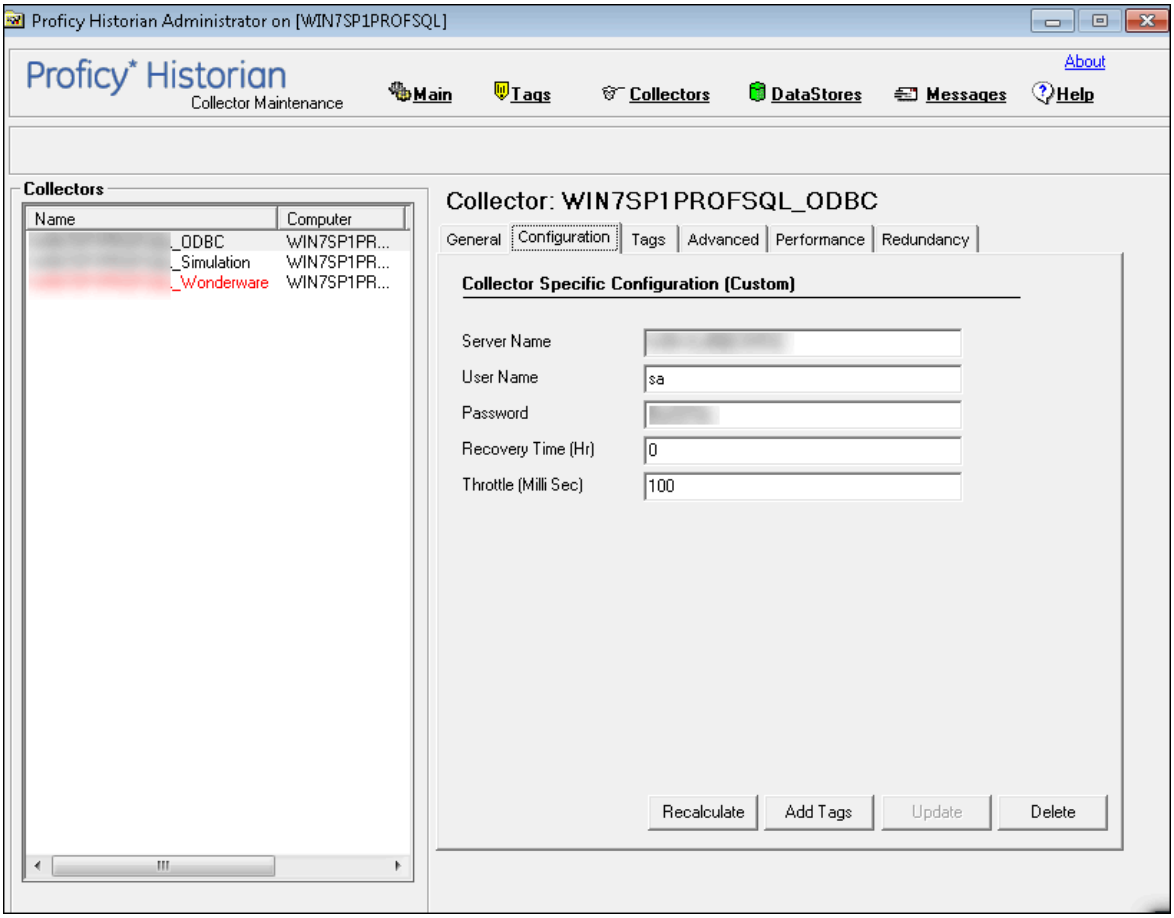
- If you have selected **Historian Configuration**, specify the tags using Configuration Hub *(on page 357)*.
- If you have selected **Offline Configuration**, modify the offline configuration file of the collector. By default, the file is available in the following location: `<installation folder of Historian>\GE Digital\<collector name>`. For information, refer to Offline Configuration for Collectors *(on page 2039)*. This option is applicable only if you have selected a cloud destination.

## Configure the ODBC Collector Using Historian Administrator

You can establish a connection between the ODBC collector and an ODBC server, and set the recovery time and throttle value by configuring the ODBC collector.

**Note:**
ODBC Driver for SQL is required to install the ODBC collector and connect it with the ODBC server. If you install the ODBC collector and the ODBC server on the same machine, install the ODBC Driver for SQL on the same machine as well.

1. Access Historian Administrator.
2. Select **Collectors**, and then select the ODBC collector instance that you want to connect to an ODBC server.
3. Select **Configuration**.
   The **Configuration** section appears.




4. Enter values as specified in the following table.

| Field | Description |
|---|---|
| Server Name | The name of the ODBC server database. |
| User Name | The username of the ODBC server database. |
| Password | The password of the ODBC server database. |
| Recovery Time (hours) | The maximum time, in hours, for which the collector will attempt to recover data after the collector is started or when connection between the collector and the ODBC server is re-established. This time is calculated as the duration between the current time and the last known write time.<br><br>Continuous data collection is resumed only after the previous data has been recovered. |

| Field | Description |
|---|---|
| | By default, this value is set to 0, which means data recovery is not attempted. The maximum value you can provide is 168 hours (that is, 7 days). |
| Throttle (Milliseconds) | The frequency, in milliseconds, at which you want the ODBC collector to query the ODBC server for tag data. This will minimize the load on the ODBC server. You can enter a value up to 16 hours.<br><br>**Note:**<br>If this field is blank, enter the required minimum value of 100 milliseconds. |

If you have provided incorrect ODBC server details (that is the server name, username, and password) either in Historian Administrator or in the `ODBC_Mapping.xml` file, the ODBC collector cannot connect with the OBDC server. To fix this issue:

a. Delete the ODBC collector instance.
b. Provide values for the following properties in the registry key `HKLM\Softwares\GE Digital\iHistorian\Services\ODBCCollector`:
   - **General1**: ODBC server name
   - **General2**: ODBC server username
   - **General3**: ODBC server password
c. Ensure that the mapping file is configured correctly *(on page 2349)*.
d. Restart the collector.

If this workaround is not successful, restart the collector instance.

# Map Data Format

For the ODBC collector to interpret the received data accurately, you must map the format and structure of the data between the ODBC server and Historian.

1. Access the `ODBC_Mapping.xml` file. By default, this file is located at `C:\Program Files\GE Digital\Historian ODBC Collector\Server`. In the ODBC collector registry path, this file is stored in the Mapping File variable.
2. For each data type in the ODBC server, add an entry in the equivalent Historian data type as described in the following table. If a Historian data type does not have an equivalent ODBC data type, enter `*NA*`.

| Historian Data Type | ODBC Server Data Type |
|---|---|
| ihByte | Byte |
| ihFloat | SingleFloat |
| ihDoubleFloat | DoubleFloat |
| ihInteger | SingleInteger |
| ihDoubleInteger | DoubleInteger |
| ihScaled | *NA* |
| ihFixedString | *NA* |
| ihVariableString | *NA* |
| ihBlob | *NA* |
| ihTime | *NA* |
| ihInt64 | *NA* |
| ihUInt64 | *NA* |
| ihUInt32 | *NA* |
| ihUInt16 | *NA* |
| ihBool | *NA* |

For example, if the ODBC server contains a Float data type named ID, enter `<ihFloat>ID</ihFloat>`

```
<DataTypeMapping>
  <ihDataTypeUndefined>*NA*</ihDataTypeUndefined>
  <ihScaled>*NA*</ihScaled>
  <ihFloat>ID</ihFloat>
  <ihDoubleFloat>*NA*</ihDoubleFloat>
  <ihInteger>2</ihInteger>
```

```
 <ihDoubleInteger>*NA*</ihDoubleInteger>
 <ihFixedString>*NA*</ihFixedString>
 <ihVariableString>3</ihVariableString>
 <ihBlob>*NA*</ihBlob>
 <ihTime>*NA*</ihTime>
 <ihInt64>*NA*</ihInt64>
 <ihUInt64>*NA*</ihUInt64>
 <ihUInt32>*NA*</ihUInt32>
 <ihUInt16>*NA*</ihUInt16>
 <ihByte>*NA*</ihByte>
 <ihBool>*NA*</ihBool>
 <ihMultiField>*NA*</ihMultiField>
 <ihArray>*NA*</ihArray>
</DataTypeMapping>
```

3. In the Quality and SubQuality elements, provide the range of values retrieved from the quality column. For quality elements that are not applicable, enter `*NA*`.
   For example, if the values from 0 to 97 are considered as bad quality, and if the numbers from 98 to 100 are considered as good quality, provide the values as follows:

```
<Quality>
 <ihOPCBad>[0,98)</ihOPCBad>
  <ihOPCUncertain>*NA*</ihOPCUncertain>
  <ihOPCNA>*NA*</ihOPCNA>
  <ihOPCGood>[99,101)</ihOPCGood>
 </Quality>


 <SubQuality>
 <ihOPCNonspecific>*NA*</ihOPCNonspecific>
  <ihOPCConfigurationError>*NA*</ihOPCConfigurationError>
  <ihOPCNotConnected>*NA*</ihOPCNotConnected>
  <ihOPCDeviceFailure>*NA*</ihOPCDeviceFailure>
   <ihOPCSensorFailure>*NA*</ihOPCSensorFailure>
  <ihOPCCommFailure>*NA*</ihOPCCommFailure>
  <ihOPCOutOfService>float</ihOPCOutOfService>
  <ihScaledOutOfRange>*NA*</ihScaledOutOfRange>
   <ihOffLine>*NA*</ihOffLine>
  <ihNoValue>*NA*</ihNoValue>
  <ihCalculationError>*NA*</ihCalculationError>
```

```
 <ihConditionCollectionHalted>*NA*</ihConditionCollectionHalted>
 <ihCalculationTimeout>*NA*</ihCalculationTimeout>
</SubQuality>
```

4. In the TagInfo element, provide the tag details, which are used to browse for tags. Provide the column names available in the ODBC server in the corresponding tag element.

```
<TagInfo>
 <DBName>DB1</DBName> <!--Cannot be *NA*-->
 <TableName>Temperature</TableName> <!--Cannot be *NA*-->
 <TagName>Boiler_Temp</TagName> <!--Cannot be *NA*-->
 <Description>*NA*</Description>
 <EngineeringUnits>*NA*</EngineeringUnits>
 <DataType>*NA*</DataType>
 <MinimumEngineeringUnit>*NA*</MinimumEngineeringUnit>
 <MaximumEngineeringUnit>*NA*</MaximumEngineeringUnit>
</TagInfo>
```

> **Note:**
> If you enter *NA* for the DataType element, you can provide only one data type mapping for the DataTypeMapping element and all the remaining elements must be marked *NA*

You can choose to automatically run queries from the info you provide in the TagInfo element. To do so, enter `<Mode>1<Mode>` in the TagInfo element. If you want to provide queries manually, enter `<Mode>0<Mode>` in the TagInfo element.

5. In the DataInfo element, provide the tag data details, which are used to create a query to collect the data.

```
<DataInfo>
 <DBName>DB1</DBName> <!--Cannot be *NA*-->
 <TableName>Temperature</TableName> <!--Cannot be *NA*-->
 <TagName>Boiler_Temperature</TagName> <!--Cannot be *NA*-->
 <Timestamp>10-06-26 02:31:29,573</Timestamp> <!--Cannot be *NA*-->
 <Value>97</Value> <!--Cannot be *NA*-->
 <Quality>good</Quality> <!--Cannot be *NA*-->
 <SubQuality>*NA*</SubQuality>
</DataInfo>
```

You can choose to automatically run queries from the info you provide in the DataInfo element. To do so, enter `<Mode>1<Mode>` in the DataInfo element. If you want to provide queries manually, enter `<Mode>0<Mode>` in the DataInfo element.

6. If you want to provide your own queries, provide them in the following format:

```
<Query>
    <Browse></Browse>
    <ReadData></ReadData>
    <TagCount></TagCount>
</Query>
</Mapping>
```

```
<Query>
<Browse>SELECT [TagName],[Description],[TagType],[Unit],[MinEU],[MaxEU] FROM
 [Runtime].[dbo].[TagHistory]</Browse>
<ReadData>SELECT TagName, [DateTime], Value, Quality, QualityDetail FROM History where History.TagName =
 '?Tagname?' AND wwRetrievalMode = 'FULL' AND wwVersion = 'Latest' AND DateTime &gt; '?Start?' ORDER BY
 DateTime ASC</ReadData>
<TagCount>SELECT count(*) from [Runtime].[dbo].[TagHistory]</TagCount>
</Query>
```

# Data Recovery

**Note:**
We recommend that the collector for which the data recovery is intended is in the same time zone as the server. If there is a mismatch, there is a possibility that data recovery will be incomplete.

## Automatic Data Recovery

In this mode, data is automatically recovered since the last time data has been collected.

**How it works:**

1. The collector determines the duration between the current time and the last time data has been written to the Historian data archive, which is stored in the LastSampleWriteTime registry key.
2. It compares this duration with the value in the Recovery Time field specified in the collector settings *(on page 2347)*.
3. It uses the shorter duration to perform a raw data query on all the tags.
4. It then processes the returned samples in chronological order.

For example, if the collector was stopped for 8 hours, but Max Recovery Time was 4 hours, only 4 hours of data would be recovered.

As per the recovery logic, an end-of-collection marker is placed at the point in time where the collector was stopped. This end-of-collection marker may or may not be there after the recovery is complete. As part of the recovery logic, if recovery data point time matches the timestamp of the end-of-collection marker, it is overwritten with the recovered good data.

### Manual Data Recovery

In this mode, you can fill gaps in the data, but you cannot fill old data.

**To perform a manual recovery:**

1. Access Historian Administrator.
2. Select **Collectors**, and then select the ODBC collector instance for which you want to manually recover data.
3. Select **Recalculate**.

   The **Recalculate** window appears.
4. Enter start time, end time, and other required information. We recommend that you choose small time intervals to reduce the load on the server and the collector.
5. Select **Recalculate**.

The tag data is recalculated. After the manual recalculation begins, the collector recovers data of the selected tags data from the collector, and sends it to Historian between the start time and end time.

At the time of recovery, if the connection to server is lost, and if the reconnect mechanism is enabled, the collector will try to connect to the server and fetch the data once connection re-establishes.

### Manual Data Recovery

Assume that the collector is connected to Historian for the first time today and the archive was created at 10 am. The user initiates manual recalculation from 1am to 2 am. For that time interval, the archives were not even created. With respect to Historian, it is old unknown data and the data write fails. If there is a data gap between 1am and 2 am, manual recalculation successfully fills the data gap.

## Reconnect to the ODBC Server Automatically

You can reconnect to the ODBC server automatically as soon as the server is up and running. By default, the collector polls for the server connection every 5 seconds. You can change this interval as well. The collector is stopped until reconnected to the server.

1. Access the following registry key: `HKEY_LOCAL_MACHINE\SOFTWARE\GE Digital \iHistorian\Services\ODBCCollector`.
2. Create a DWORD named `EnableReconnect`.
3. Enter the decimal value 1.
4. If you want to change the reconnection interval (from the default value of 5 seconds):
    a. Create a DWORD named `ReconnectInterval`.
    b. Enter a decimal value between 5 and 60. This value represents the number of seconds for the collector to wait before trying to reconnect to the ODBC server.
5. Select **OK**, then close the registry.

# Troubleshooting the ODBC Collector

The ODBC collector generates logs during initialization, configuration, and general operation. By default, you can find them in the general logging folder, `C:\Proficy Historian Data\LogFiles`.

## Troubleshooting Tips

- Ensure that the ODBC server is running before the starting the ODBC collector.
- If the ODBC collector does not start automatically, refer to the Historian log file to view log entries to determine the problem.

## The ODBC Collector Cannot Connect to the ODBC Server

If you have provided incorrect ODBC server details (that is the server name, username, and password) either in Historian Administrator or in the `ODBC_Mapping.xml` file, the ODBC collector cannot connect with the OBDC server.

**Workaround**:

1. Delete the ODBC collector instance.
2. Provide values for the following properties in the registry key `HKLM\Softwares\GE Digital \iHistorian\Services\ODBCCollector`:
    - **General1**: ODBC server name
    - **General2**: ODBC server username
    - **General3**: ODBC server password
3. Ensure that the mapping file is configured correctly *(on page 2349)*.
4. Restart the collector.

If this workaround is not successful, restart the collector instance.

# Chapter 24. The OPC Classic DA Collector

## Overview of the OPC Classic DA Collector

The OPC Classic Data Access (DA) collector collects data from any OPC 1.0 or OPC 2.0 compliant OPC server (such as CIMPLICITY). The collector automatically determines the capability of the OPC server to which it is connected and supports appropriate features based on this information.

**Features:**

- You can browse the source for tags and their attributes on an OPC server that supports browsing.
- Both the polled and unsolicited data collection are supported; when changes to the OPC source tags are detected, they are sent to the Historian server. Unsolicited data collection is supported for OPC 2.0 only. The minimum poll interval is 100ms. The collector duplicates raw samples from the OPC server into the Historian data archive.

  For unsolicited data collection, if collector compression is disabled, all new values produce an exception. And, the deadband percentage is determined by the collector deadband percent. You can only configure the collector deadband percent by enabling compression.
- The supported timestamp resolution is 1ms.
- Floating point, integer, binary, and string data are supported.
- Python expression tags are supported.
- Device timestamps are accepted.

**Supported data types:**

| The OPC Data Type | Recommended Data Type in Historian |
|---|---|
| I1 - 16 bit signed integer | Single Integer |
| I4 - 32 bit signed integer | Double Integer |
| R4 - 32 bit float | Single Float |
| R8 - 64 bit double float | Double Float |
| UI2 - 16 bit unsigned single integer | Unsigned Single Integer |
| UI4 - 32 bit unsigned double integer | Unsigned Double Integer |
| UI8 - 64 bit unsigned quad integer | Unsigned Quad Integer |
| I8 - 64 bit quad integer | Quad Integer |

| The OPC Data Type | Recommended Data Type in Historian |
|---|---|
| BSTR | Variable String |
| BOOL | Boolean |
| I1 - 8 bit single integer | Byte |

**Note:**
The collector requests data from the OPC server in the native data type. Then the collector converts the received value to a Historian Data Type before sending it to the data archiver.

**Supported tag attributes:**

- Tagname
- Source Address
- Engineering Unit Description
- Data Type
- Hi Engineering Units
- Lo Engineering Units
- Is Array Tag

The Engineering Unit Description, Hi Engineering Units and Lo Engineering Units vary based on the OPC server vendor.

**Note:**
While some of these attributes are queried on a browse, they are not shown in the browse interface. These attributes are used when adding a tag, but it is not visible to you if all attributes come from the server or not.

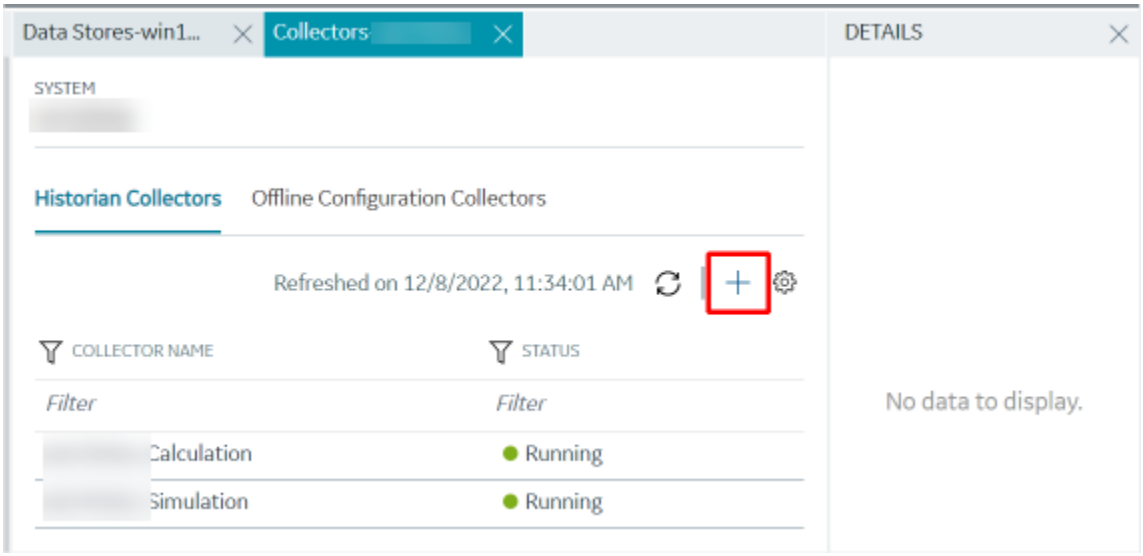
# Configuration

## Add and Configure an OPC Classic Data Access Collector

The OPC Classic Data Access (DA) collector collects data from any OPC 1.0 or OPC 2.0 compliant OPC Classic server. For more information, refer to Overview of the OPC Classic DA Collector *(on page 2356)*.

This topic describes how to add a collector instance using Configuration Hub. You can also add a collector instance using the RemoteCollectorConfigurator utility *(on page 797)*, which does not require you to install Web-based Clients.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.
   A list of collectors in the default system appears.
3. In the upper-right corner of the main section, select +.




   The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.
4. In the **MACHINE NAME** field, select the machine in which you want to add a collector instance.
5. In the **COLLECTOR TYPE** field, select **OPC Collector**, and then select **Get Details**.
   The **INSTALLATION DRIVE** and **DATA DIRECTORY** fields are disabled and populated.
6. Select **Next**.
   The **Source Configuration** section appears.
7. Enter values as described in the following table.

| Field | Description |
|---|---|
| **OPC SERVER** | Select the machine on which you have installed the OPC Classic DA server from which you want to collect data. |

| Field | Description |
|---|---|
| **MACHINE NAME** | Enter the host name or IP address of the OPC server. This field appears only if you have selected a remote OPC server. A value is required. |
| **OPC DA SERVER PROG ID** | Enter the prog ID of the OPC server. This field appears only if you have selected a remote OPC server. A value is required. |

8. Select **Next**.
   The **Destination Configuration** section appears. The collector machine name provided by you is selected as the **Source Configuration** by default.

   Under **CHOOSE DESTINATION**, the **Historian Server** option is selected by default. In addition, the **DESTINATION HISTORIAN SERVER** field is disabled and populated with the collector machine name.

9. Select the destination to which you want to send data, and then enter the values in the corresponding fields. You can send data to an on-premises Historian server or to a cloud destination.

   a. If you need to send data to a cloud destination, select the cloud destinations as needed.
      - **Predix Timeseries**- Select this if you need to send data to Predix cloud. For more information, refer to Predix Cloud *(on page 621)*.
      - **Azure IoT Hub**- Select this if you need to send data to Azure Cloud in KairosDB format. For more information, refer to Azure IoT Hub (KairosDB format) *(on page 608)*.
      - **MQTT**- Select this if you need to send data to any of the following cloud destination.
        - Alibaba cloud. For more information, refer to Alibaba Cloud *(on page 587)*.
        - AWS cloud. For more information, refer to AWS Cloud *(on page 594)*.
        - Google cloud. For more information, refer to Google Cloud *(on page 614)*.

   b. If you need to send data to an on-premises Historian server, select **Historian Server**.
      If you created security groups or enabled a strict client/collector authentication, enter the **USERNAME** and **PASSWORD** of the on-premises Historian server that you created during the installation of the collector.

If you entered the **USERNAME**and **PASSWORD**, select **Test Connection**. This will help you to test if the Historian server that you are trying to connect is valid or if the credentials that you entered are valid.
If the entered credentials are valid, a successful connection message appears.

10. Select **Next**.
    The **Collector Initiation** section appears. The **COLLECTOR NAME** field is populated with a value in the following format: `<system name>_OPC_<OPC server name>`
11. If needed, modify the value in the **COLLECTOR NAME** field.
    The value that you enter:
    - Must be unique.
    - Must not exceed 15 characters.
    - Must not contain a space.
    - Must not contain special characters except a hyphen, period, and an underscore.
12. In the **RUNNING MODE** field, select one of the following options.
    - **Service - Local System Account**: Select this option if you want to run the collector as a Windows service using the credentials of the local user (that is, the currently logged-in user). By default this option is selected, and the **USERNAME** and **PASSWORD** fields are disabled.
    - **Service Under Specific User Account**: Select this option if you want to run the collector as a Windows service using a specific user account. If you select this option, you must enter values in the **USERNAME** and **PASSWORD** fields.

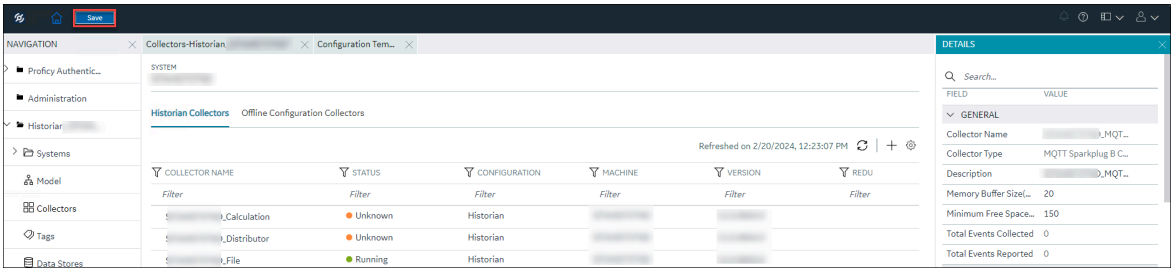
      If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:
      - iH Security Admins
      - iH Collector Admins
      - iH Tag Admins

    You can also configure the collector to start automatically when you start the computer.
13. Select **Add**.
    The collector instance is added. The fields specific to the collector section appear in the **DETAILS** section.
14. Under **COLLECTOR SPECIFIC CONFIGURATION**, configure values as described in the following table.

| Field | Description |
|---|---|
| **OPC Server Prog ID** | The program ID of the OPC server from which you want to collect data. |
| **Read Mode** | The read mode that you want the collector to use. For information, refer to the documentation of the OPC server that you are using or the OPC specification on the OPC Foundation website. |
| **First Browse Criteria** | A comma-separated first-level search criterion for browsing tags from the data source. The top-level and second-level criteria are used together by the AND operation to browse tags.<br><br>For example, if you enter `USGB014` in the **First Browse Criteria** field and `F_CV, B_CUALM` in the **Second Browse Criteria** field, it returns all the tags that contain:<br>◦ USGB014<br><br>-and-<br><br>◦ F_CV or B_CUALM |
| **Second Browse Criteria** | A comma-separated second-level search criterion for browsing tags from the data source. The top-level and second-level criteria are used together by the AND operation to browse tags. |
| **Threading Model** | The type of the threading model selected for the collector. The model selected must match the threading model of the OPC server.<br>◦ **Multithreaded**: Select this option for better performance. We recommend that you configure your collector to use the default multi-threading model.<br>◦ **Apparent**: Select this option for best compatibility. Some OPC servers do not work well with multi-threading. If you experience problems running your collector with multi-threading, use the apartment model. |

| Field | Description |
|---|---|
| | The default setting is multi-threaded. For information, refer to the documentation of the OPC server you are using. |
| **Configuration Changes** | Indicates whether the collector configuration changes are processed in real time or after restarting the collector.<br>◦ **Made On-Line**: Select this option to process any configuration changes immediately (after 30 seconds) after you select the **Update** button.<br>**Note:**<br>▪ Some OPC servers cannot handle processing configuration changes online. If you experience any instability with changes made online, use the next option.<br>◦ **Made After Collector Restart**: Select this option to hold all configuration changes until you manually restart the collector. |
| **MTLS Security** | Indicates whether you want to use Mutual TLS (MTLS) protocol to enforce a secure and strong authentication mechanism. |
| **MTLS Data Encryption** | Indicates whether you want to encrypt the data that the collector shares to the data archiver (DA). |
| For more information on how to enable MTLS Security, refer to Enable MTLS Security *(on page 632)*. | |

15. As needed, enter values in the other sections common to all collectors *(on page 579)*.
16. In the upper-left corner of the page, select **Save**.



The changes to the collector instance are saved.

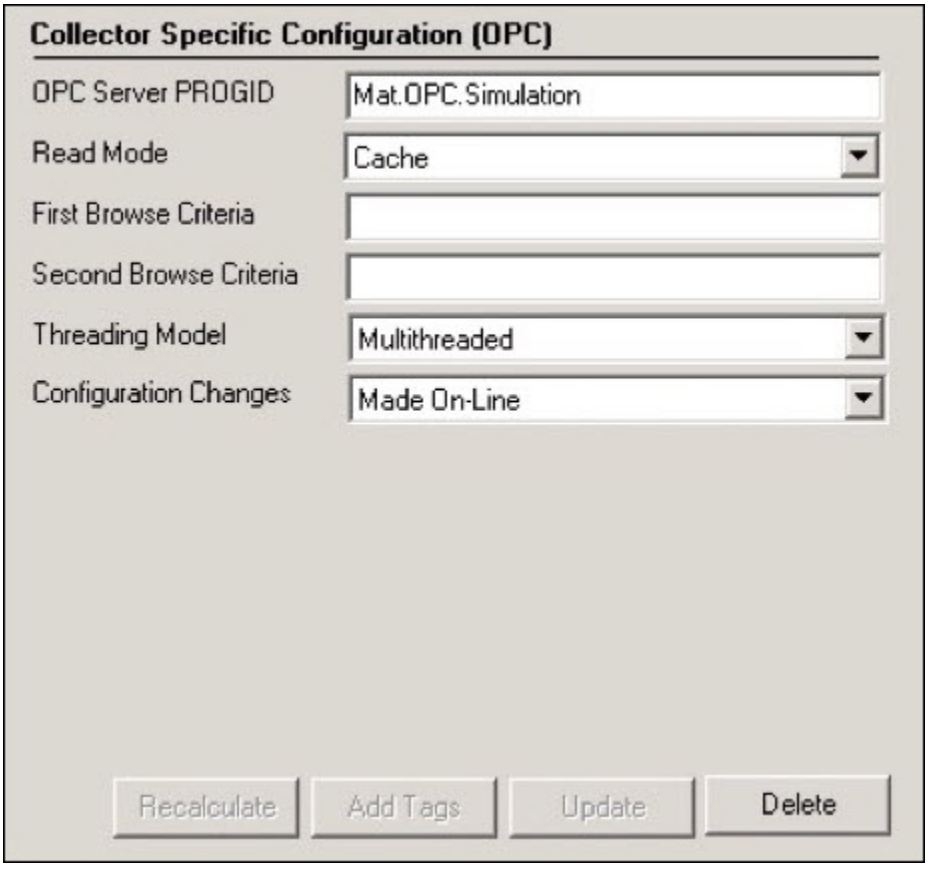
17. If needed, restart the collector.

Specify the tags whose data you want to collect using the collector. In the **CHOOSE CONFIGURATION** field,

- If you have selected **Historian Configuration**, specify the tags for data collection *(on page 357)*.
- If you have selected **Offline Configuration**, modify the offline configuration file of the collector. By default, the file is available in the following location: `<installation folder of Historian>\GE Digital\<collector name>`. For information, refer to Offline Configuration for Collectors *(on page 2039)*. This option is applicable only if you have selected a cloud destination.

## Configure the OPC Classic DA Collector Using Historian Administrator

1. Access Historian Administrator *(on page 823)*.
2. Select **Collectors**, and then select the OPC Classic DA collector instance that you want to configure.
3. Select **Configuration**.
   The **Configuration** section appears.




4. Enter values as specified in the following table.

| Field | Description |
|---|---|
| OPC Server Prog ID | The program ID of the OPC server from which you want to collect data. |
| Read Mode | The read mode that you want the collector to use. For information, refer to the documentation of the OPC server that you are using or the OPC specification on the OPC Foundation website. |
| First Browse Criteria | A comma-separated first-level search criterion for browsing tags from the data source. The top-level and second-level criteria are used together by the AND operation to browse tags.<br><br>For example, if you enter `USGB014` in the First Browse Criteria field and `F_CV, B_CUALM` in the Second Browse Criteria field, it returns all the tags that contain: |

| Field | Description |
|---|---|
| | ◦ USGB014<br><br>-and-<br><br>◦ F_CV or B_CUALM |
| Second Browse Criteria | A comma-separated second-level search criterion for browsing tags from the data source. The top-level and second-level criteria are used together by the AND operation to browse tags. |
| Threading Model | The type of the threading model selected for the collector. The model selected must match the threading model of the OPC server.<br>◦ **Multithreaded**: Select this option for better performance. We recommend that you configure your collector to use the default multi-threading model.<br>◦ **Apparent**: Select this option for best compatibility. Some OPC servers do not work well with multi-threading. If you experience problems running your collector with multi-threading, use the apartment model.<br>The default setting is multi-threaded. For information, refer to the documentation of the OPC server you are using. |
| Configuration Changes | Indicates whether the collector configuration changes are processed in real time or after restarting the collector.<br>◦ **Made On-Line**: Select this option to process any configuration changes immediately (after 30 seconds) after you select the **Update** button.<br><br>**Note:**<br>▪ Some OPC servers cannot handle processing configuration changes online. If you experience any instability with changes made online, use the next option.<br><br>◦ **Made After Collector Restart**: Select this option to hold all configuration changes until you manually restart the collector. |

| Field | Description |
|---|---|
| | **Note:** Starting and stopping the collector in the **General** section of the **Collector Maintenance** page of Historian Administrator does not constitute a manual restart. You must either start and stop the collector from the Services window or the console application.<br><br>To allow configuration changes, you also must enable the **On-Line Tag Configuration Changes** option on the **Advanced** section of the **Collector Maintenance** page of Historian Administrator. |

5. Select **Update**.
6. Restart the collector.
   The collector is configured.

## Configure GE Intelligent Platform Drivers and Deadbands

If you want to add items of different data types to Historian using the OPC Classic DA collector to a GE Intelligent Platform v7.x driver or an OPC server, and you are using deadbands, you must manually modify the source address for each item you add. This is to specify the engineering unit (EGU) range for that item.

1. Access Historian Administrator.
2. Select **Tag Maintenance** , and then select the item that you want to modify.
3. Select **Collection**.
4. In the **Source Address** field, add the following fields: `|SIGNALCONDITIONING,LOWEGU,HIGHEGU,HARDWAREOPTIONS`
5. Repeat the steps for each item that you want to modify. If, however, you want all the items to use the same data type, change the settings of the following registry key: `\\HKEY_LOCAL_MACHINE\SOFTWARE\In-tellution\Drivers\SI7\OPC\ItemDefaults`. These values apply to all items not specified by the source address after you restart the driver.

### Using Deadbands with the SI7 Driver

When you use the SI7 driver, it sets the global default values for EGU limits used for deadband calculations to the following values:

- 1 Lo EGU = 0
- 1 Hi EGU = 65535
- 1 EGU Span = Hi EGU - Lo EGU = 65535

You may want change the default values for this driver if the items that you add use data types other than Integer (such as Float).

If, however, you want all the items to use the same data type, change the settings for `HiEGU` and `LoEGU` for the following registry key: `\\HKEY_LOCAL_MACHINE\SOFTWARE\In-tellution\Drivers\SI7\OPC\ItemDefaults`. These values apply to all items not specified by the source address after you restart the driver.

# Specifying Tags for Data Collection

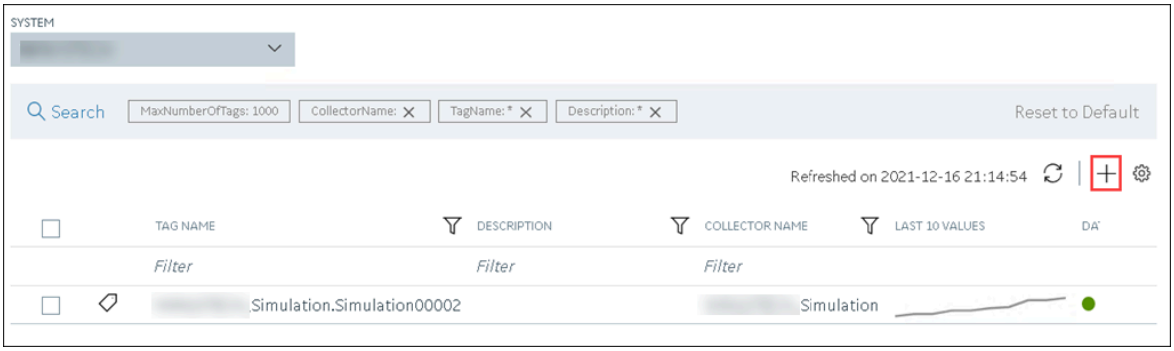
## Add Tags for the Data Store Using Configuration Hub

- Add the collector instance *(on page 357)* using which you want to collect data. Ensure that the collector is running.
- By default, the tag data is stored in the user data store, which is created automatically when you set up Configuration Hub. If, however, you want to store the data in a different data store, create it *(on page 370)*.

This topic describes how to specify the tags for which you want to collect data by browsing through the tags in the data source. For example, for an iFIX collector, if there are 1,00,000 tags in the iFIX server, you must specify the ones for which you want to collect data. Only then data is collected for those tags.

In addition to adding tags from the data source, you can create tags manually *(on page 473)*.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
3. Select +.

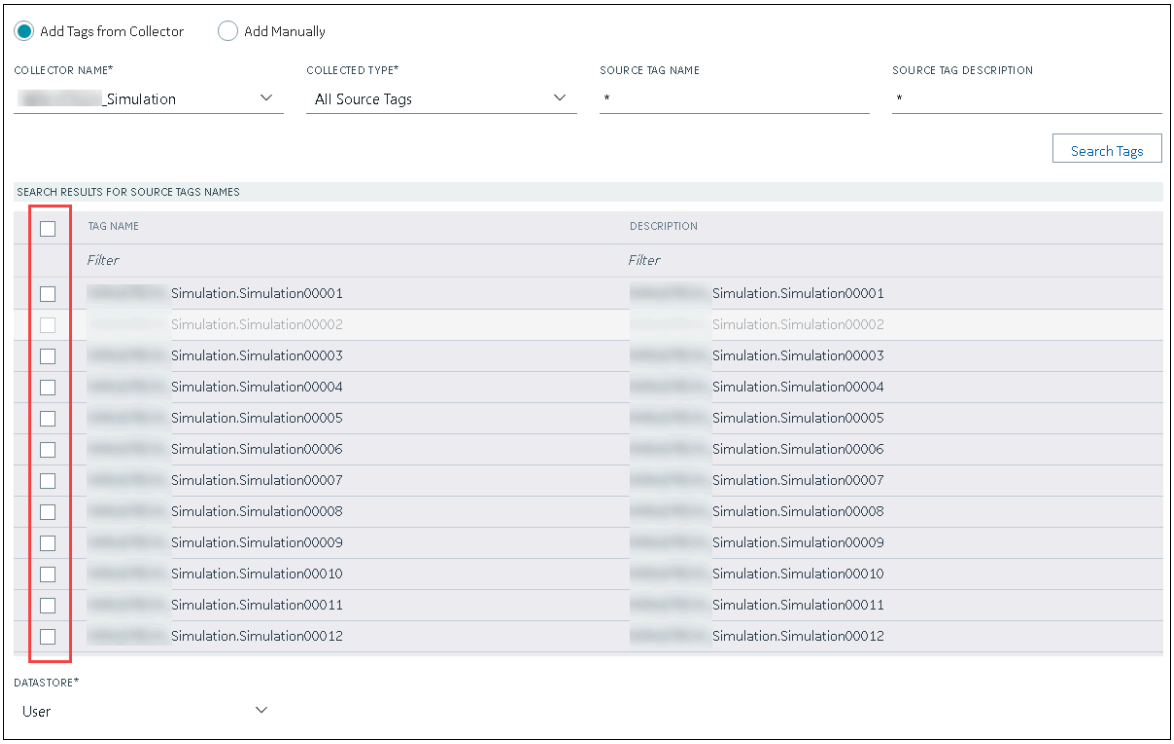



The **Add Tag-<system name>** page appears. The **Add Tags from Collector** option is selected by default.

4. Enter values as described in the following table.

| Field | Description |
|---|---|
| **COLLECTOR NAME** | Select the collector instance that you want to use to collect data. A value is required. |
| **COLLECTED TYPE** | Specify whether you want to browse through all the tags in the data source or only from the tags that you have not added yet. A value is required. |
| **SOURCE TAG NAME** | Enter the name of the tag (either completely or partially) to narrow down the search results. |
| **SOURCE TAG DESCRIPTION** | Enter the description of the tag (either completely or partially) to narrow down the search results. |

5. Select **Search Tags**.
   A list of tags that match *all* the criteria that you have specified appears. If a tag is already added, it is disabled.
6. Select the check box corresponding to each tag for which you want to collect data.




7. In the **DATA STORE** field, if you want to store the data in a different data store than the user data store, select the same.
8. Select **Add Tag**.
   Data collection begins for the selected tags.

As needed, configure each tag by providing values for the tag properties. For information on the delta query modes, refer to Counter Delta Queries.

## Specify Tags for Data Collection Using Historian Administrator

If your OPC server supports hierarchical organization of tags in a tree structure, you can use the hierarchy to browse for tags and add them to the collector for data collection.

1. Access Historian Administrator.
2. Select **Collectors**, and then select the OPC Classic DA collector instance to which you want to add tags.
   A hierarchical view of tags appears in the **Browse Results** section.
3. If you want to view only the tags for which data is not collected, in the **Show Only** field, select **Source Tags Not Collected**. You can search for a tag by entering search criteria in the **Source Tag Name** or **Description** field.

4. If you want to search by a tag name or description, enter the value in the **Source Tag Name** or **Description** field.
5. Navigate to the node in the tree you want to browse, and then select **Browse**.

> **Tip:**
> - To browse automatically, select the **Auto Browse** check box. The available tags appear in the **Browse Results** window whenever a node is selected in the tree.
> - To show all child elements within a hierarchy, select the **Show All Children** check box. All tags at or below the hierarchical level of the selected node in the tree appear in the **Browse Results** window.

The tags within the selected portion of the OPC server tag hierarchy appear.

- Some OPC servers do not support data blocks with a length greater than 1. These servers display only the first item in an array instead of showing all of them. For example, an OPC server may contain 3000 analog values from datablock:1 to datablock:3000, but only datablock:1 is displayed.
- If you want to archive data from poll records of a length greater than 1, we recommend that you use the Excel Add-In for Historian to configure a large block of tags (including the missing items), and then add the tags.
- If you are unable to browse items containing a forward slash ( / ) in your OPC server, you may have to change the default separator in the collector configuration. To do so, modify the registry key `HKEY_LOCAL_MACHINE\SOFTWARE\Intellution, Inc\iHistorian\Services\OPCCollector\<collector interface name>\OPCBrowseTreeSep key`, and change the string value to a character not available in your OPC server item IDs. Typical values include |, !, or &. Create this key if it does not exist.
- If you are cannot browse readable items in your OPC server, you may need to change the browse access mask used by the collector. To do so, modify the registry key `[HKEY_LOCAL_MACHINE\SOFTWARE\Intellution, Inc.\iHistorian\Services\OPCCollector\<collector interface name>`, and add the DWORD key `"OPCBrowseAccessRightsMask"=dword:00000003`. Valid values are 0, 1, 2, 3 with 1 being the default. Use 0 or 3 if you are unable to browse readable items. Creating or changing the value takes effect on the next browse attempt and does not require a collector restart.
- Some items such as unsupported data types and user-defined items in simulation servers may not be returned when you browse for tags. However, sometimes, even the items that do not appear in the search results can be added using the **Add Tag Manually** button.

6. Select the tags for which you want to collect data, and then select **Add Selected Tags**. Collected tags will appear in black in the tag list.
   The tags are added to the collector. They appear in black text in the list of tags.

# OPC Group Creation

It is recommended that you limit the number of OPC groups created by the Historian system to increase performance. To limit the number of OPC groups created on the OPC server, consider grouping Historian tags (collected by the OPC collector) using the least amount of collection intervals possible.

# Troubleshooting the OPC Classic DA Collector

## Troubleshooting Tips

When reviewing the OPC Classic DA collector log file, or the log messages from Event Viewer:

- If you notice a message that states that Historian could not create the buffer files, then the issue is most likely that you do not have enough free space available for the buffer files.
- If you notice the OPC Classic DA collector connection attempt, a COM initialization attempt, and then a shutdown, the cause of the error is most likely that the collector is trying to start before the OPC server fully starts.
- If you look at the log files and you do not see anything special, aside from a startup attempt and a shutdown message, then begin by assuming that the OPC server is not fully starting. If the workaround for that issue does not appear to work, try adjusting the buffer size.

## Issue: The Collector Fails to Start

**Possible Causes:**

- There is not enough free space for the collector to create its buffer files on startup.
- Historian is trying to run the OPC Collector before the OPC Server fully starts.

Try the workarounds in the following sections.

## Issue: Not Enough Space for Buffer Files

**Workaround:** Try one of the following options:

- Free up the disk space.
- Move the buffer files to a different location.

- Change the buffer size by adding a DWORD `MinimumDiskFreeBufferSize` in the registry under `HKEY_LOCAL_MACHINE\SOFTWARE\Intellution, Inc.\iHistorian\Services\OPCCollector\<collector interface name>`. We recommend setting it to 10 or 20 MB. After you save your changes, restart your machine.

### Issue: The Collector does not Connect to the Historian Server

**Workaround:** Check the `Logfiles` folder on the collector machine. If the log file specifies "could not create buffer files", repeat the workaround in the previous issue.

### Issue: The Collector Tries to Start Before the OPC Server Starts

**Workaround:** Specify a time delay for the collector to start. To do so, add a DWORD named `MachineUpTimeDelay` in the registry under `HKEY_LOCAL_MACHINE\SOFTWARE\Intellution, Inc.\iHistorian\Services\OPCCollector\<collector interface name>`. We recommend that you set its value to 120 seconds first, and restart the collector. If the problem still exists, try increasing the value slightly until the issue is resolved.

### Issue: The Collector Becomes Unresponsive After the First Polled Read

**Workaround:** If the polled reads take more than a few seconds to start:

1. Create a DWORD named `OPCFirstReadMode` in the registry under `HKEY_LOCAL_MACHINE\SOFTWARE\Intellution, Inc.\iHistorian\Services\OPCCollector\<collector interface name>`.
2. Provide one of the following values:
    - 1: Use this value if you want the collector to perform the faster `OPC_DS_CACHE` read on the first poll.
    - 2: Use this value if you want the collector to perform the slower `OPC_ DS_DEVICE` read.
3. Restart the collector.

#### Validating Items Before Adding to Polled Collection Group

To validate items before adding them to the polled collection groups from the collector, create a DWORD named `OPCValidateBeforeAdd` in the registry under `HKEY_LOCAL_MACHINE\SOFTWARE\Intellution, Inc.\iHistorian\Services\OPCCollector\<collector interface name>`, and set the value to 1.

### Issues with Browsing for Tags

**Workaround:**

- Some OPC servers do not support data blocks with a length greater than 1. These servers display only the first item in an array instead of showing all of them. For example, an OPC server may contain 3000 analog values from datablock:1 to datablock:3000, but only datablock:1 is displayed.
- If you want to archive data from poll records of a length greater than 1, we recommend that you use the Excel Add-In for Historian to configure a large block of tags (including the missing items), and then add the tags.
- If you are unable to browse items containing a forward slash ( / ) in your OPC server, you may have to change the default separator in the collector configuration. To do so, modify the registry key `HKEY_LOCAL_MACHINE\SOFTWARE\Intellution, Inc\iHistorian\Services\OPCCollector\<collector interface name>\OPCBrowseTreeSep key`, and change the string value to a character not available in your OPC server item IDs. Typical values include |, !, or &. Create this key if it does not exist.
- If you are cannot browse readable items in your OPC server, you may need to change the browse access mask used by the collector. To do so, modify the registry key `[HKEY_LOCAL_MACHINE\SOFTWARE\Intellution, Inc.\iHistorian\Services\OPCCollector\<collector interface name>`, and add the DWORD key `"OPCBrowseAccessRightsMask"=dword:00000003`. Valid values are 0, 1, 2, 3 with 1 being the default. Use 0 or 3 if you are unable to browse readable items. Creating or changing the value takes effect on the next browse attempt and does not require a collector restart.
- Some items such as unsupported data types and user-defined items in simulation servers may not be returned when you browse for tags. However, sometimes, even the items that do not appear in the search results can be added using the **Add Tag Manually** button.
- If you connect the collector to a Kepware Redundancy Master with both primary and secondary servers using Industrial Gateway Server (IGS) that share the same channel, device, and tags:
    - After creating a tag in Historian, if you modify the tag's source address, the tag will be considered invalid until the collector is restarted; data is not updated for the tag and the quality is not set to bad.
    - Always restart the Redundancy Master service before restarting the collector.
    - If you add IGS tags to Historian, you cannot delete them in IGS. If you try to do so, the following error message appears in IGS: Rejecting attempt to delete reference object.

      To avoid this error, first delete the tags in Historian, and then delete the device in IGS.
    - If you cannot browse IGS for tags, restart the collector.

# Chapter 25. The OPC Classic HDA Collector

## Overview of the OPC Classic HDA Collector

The OPC Classic Historical Data Access (HDA) collector collects data from any OPC HDA 1.2 - compliant OPC server (such as CIMPLICITY). The collector automatically determines the capability of the OPC server to which it is connected and supports the appropriate features based on this information.

**Topology:**

The OPC Classic HDA collector and the OPC Classic HDA server support remote connectivity. If the OPC Classic HDA server and the OPC Classic HDA collector are on different machines, ensure that:

- The DCOM setting is provided for both the server and collector machines.
- Before starting the collector, ensure that NT AUTHORITY/SYSTEM has SysAdmin privileges.

**Features:**

- You can browse the source for tags and their attributes on an OPC server that supports browsing.
- Only unsolicited data collection is supported; when changes to the OPC source tags are detected, they are sent to the Historian server. The minimum poll interval is 100ms. The collector duplicates raw samples from the OPC server into the Historian data archive.

  For unsolicited data collection, if collector compression is disabled, all new values produce an exception. And, the deadband percentage is determined by the collector deadband percent. You can only configure the collector deadband percent by enabling compression.

  > **Note:**
  > You must set the Time Assigned by field to Source if you have unsolicited tags getting data from an OPC Classic HDA collector.

- The supported timestamp resolution is 1ms.
- Floating point, integer, binary, and string data are supported.
- Device timestamps are accepted.

**Supported data types:**

| The OPC Data Type | Recommended Data Type in Historian |
|---|---|
| I1- 16 bit signed integer | Single Integer |
| I4- 32 bit signed integer | Double Integer |
| R8- 64 bit double float | Single Float |
| UI2- 16 bit unsigned single integer | Double Float |
| UI4- 32 bit unsigned double integer | Unsigned Integer |
| UI8- 64 bit unsigned quad integer | Unsigned Double Integer |
| I8- 64 bit quad integer | Quad Integer |
| BSTR | Variable Sting |
| BOOL | Boolean |
| I1- 8 bit single integer | Byte |

**Note:**
The OPC Classic HDA collector requests data from the OPC Classic HDA server in the native data type. The OPC Classic HDA collector then converts the received value to a Historian Data Type before sending it to the data archiver.

**Supported tag attributes:**

- Tagname
- Source Address
- Engineering Unit Description
- Data Type
- Hi Engineering Units
- Lo Engineering Units
- Is Array Tag

The Engineering Unit Description, Hi Engineering Units and Lo Engineering Units vary based on the OPC server vendor.

**Note:**
While some of these attributes are queried on a browse, they are not shown in the browse interface. These attributes are used when adding a tag, but it is not visible to you if all attributes come from the server or not.

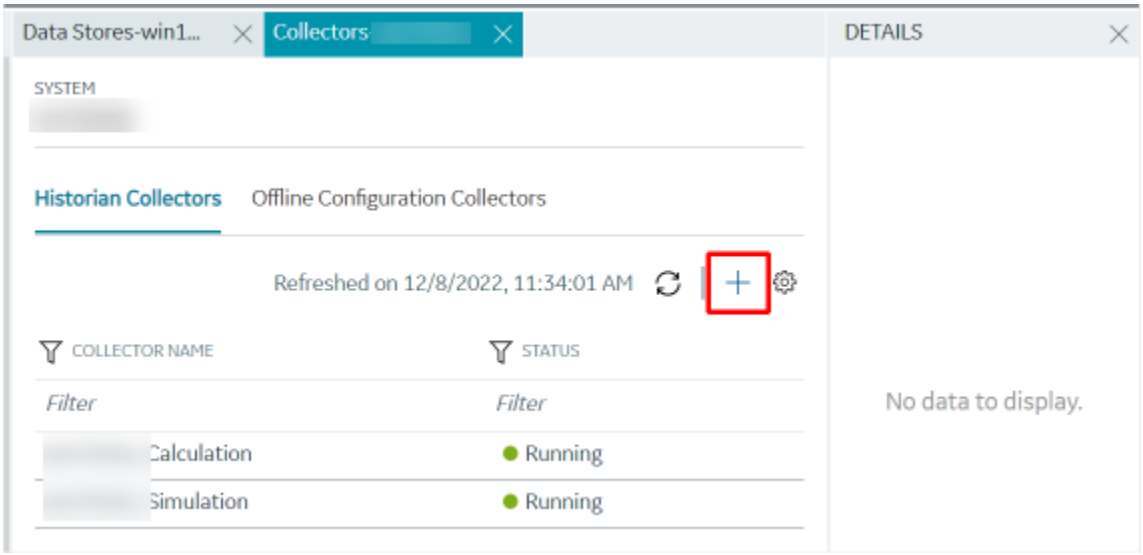
# Configuration

## Add and Configure an OPC Classic HDA Collector

The OPC Classic Historical Data Access (HDA) collector collects data from any OPC HDA 1.2 - compliant OPC Classic HDA server. For more information, refer to Configure the OPC Classic HDA Collector Using Historian Administrator *(on page 2380)*.

This topic describes how to add a collector instance using Configuration Hub. You can also add a collector instance using the RemoteCollectorConfigurator utility *(on page 797)*, which does not require you to install Web-based Clients.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.
3. In the upper-right corner of the main section, select +.




The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.

4. In the **MACHINE NAME** field, select the machine in which you want to add a collector instance.

5. In the **COLLECTOR TYPE** field, select **OPC HDA Collector**, and then select **Get Details**.
   The **INSTALLATION DRIVE** and **DATA DIRECTORY** fields are disabled and populated.
6. Select **Next**.
   The **Source Configuration** section appears.
7. Enter values as described in the following table.

| Field | Description |
|---|---|
| **OPC HDA SERVER** | Select the machine on which you have installed the OPC Classic HDA server from which you want to collect data. |
| **MACHINE NAME** | Enter the host name or IP address of the OPC server. This field appears only if you have selected a remote OPC server. A value is required. |
| **OPC DA SERVER PROG ID** | Enter the prog ID of the OPC server. This field appears only if you have selected a remote OPC server. A value is required. |

8. Select **Next**.
   The **Destination Configuration** section appears. The collector machine name provided by you is selected as the **Source Configuration** by default.

   Under **CHOOSE DESTINATION**, the **Historian Server** option is selected by default. In addition, the **DESTINATION HISTORIAN SERVER** field is disabled and populated with the collector machine name.

9. Select the destination to which you want to send data, and then enter the values in the corresponding fields. You can send data to an on-premises Historian server or to a cloud destination.

   a. If you need to send data to a cloud destination, select the cloud destinations as needed.
      - **Predix Timeseries**- Select this if you need to send data to Predix cloud. For more information, refer to Predix Cloud *(on page 621)*.
      - **Azure IoT Hub**- Select this if you need to send data to Azure Cloud in KairosDB format. For more information, refer to Azure IoT Hub (KairosDB format) *(on page 608)*.
      - **MQTT**- Select this if you need to send data to any of the following cloud destination.

- Alibaba cloud. For more information, refer to Alibaba Cloud *(on page 587)*.
- AWS cloud. For more information, refer to AWS Cloud *(on page 594)*.
- Google cloud. For more information, refer to Google Cloud *(on page 614)*.

b. If you need to send data to an on-premises Historian server, select **Historian Server**.
If you created security groups or enabled a strict client/collector authentication, enter the **USERNAME** and **PASSWORD** of the on-premises Historian server that you created during the installation of the collector.
If you entered the **USERNAME**and **PASSWORD**, select **Test Connection**. This will help you to test if the Historian server that you are trying to connect is valid or if the credentials that you entered are valid.
If the entered credentials are valid, a successful connection message appears.

10. Select **Next**.
The **Collector Initiation** section appears.
11. If needed, modify the value in the **COLLECTOR NAME** field.
The value that you enter:
    - Must be unique.
    - Must contain the string `OPCHDA`.
    - Must not exceed 15 characters.
    - Must not contain a space.
    - Must not contain special characters except a hyphen, period, and an underscore.
12. In the **RUNNING MODE** field, select one of the following options.
    - **Service - Local System Account**: Select this option if you want to run the collector as a Windows service using the credentials of the local user (that is, the currently logged-in user). By default, this option is selected, and the **USERNAME** and **PASSWORD** fields are disabled.
    - **Service Under Specific User Account**: Select this option if you want to run the collector as a Windows service using a specific user account. If you select this option, you must enter values in the **USERNAME** and **PASSWORD** fields.

      If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:
        - iH Security Admins
        - iH Collector Admins
        - iH Tag Admins

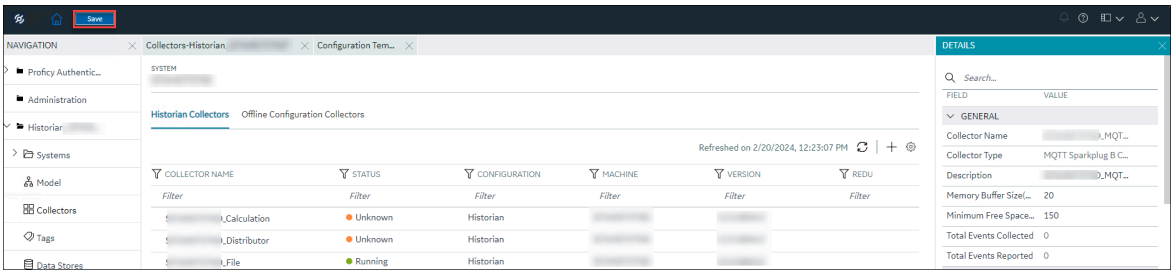
    You can also configure the collector to start automatically when you start the computer.
13. Select **Add**.

The collector instance is added. The fields specific to the collector section appear in the **DETAILS** section.

14. As needed, under **COLLECTOR SPECIFIC CONFIGURATION**, configure values as described in the following table.

| Field | Description |
|---|---|
| **Recovery Time** | It indicates the maximum time, in hours, for which the collector will attempt to recover data after the collector is started or when connection between the collector and the OPC server is re-established. This time is calculated as the duration between the current time and the last known write time.<br><br>Continuous data collection is resumed only after the previous data has been recovered.<br><br>You can enter a value between 1 and 150. |
| **MTLS Security** | Indicates whether you want to use Mutual TLS (MTLS) protocol to enforce a secure and strong authentication mechanism. |
| **MTLS Data Encryption** | Indicates whether you want to encrypt the data that the collector shares to the data archiver (DA). |
| For more information on how to enable MTLS Security, refer to Enable MTLS Security *(on page 632)*. | |

15. As needed, enter values in the other sections common to all collectors *(on page 579)*.
16. In the upper-left corner of the page, select **Save**.



The changes to the collector instance are saved.

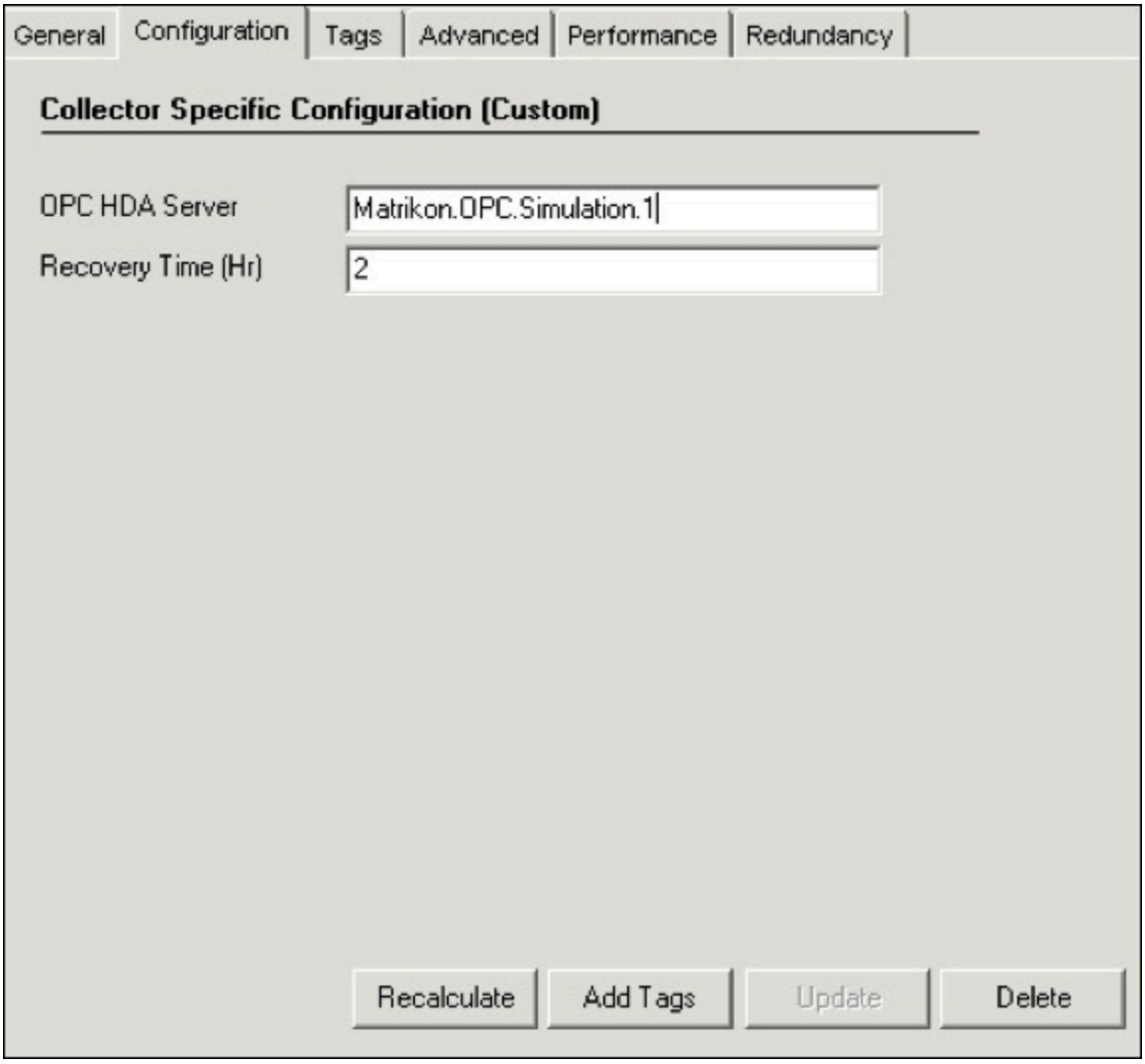
17. If needed, restart the collector.

Specify the tags whose data you want to collect using the collector. In the **CHOOSE CONFIGURATION** field,

- If you have selected **Historian Configuration**, specify the tags using Configuration Hub *(on page 357)*.
- If you have selected **Offline Configuration**, modify the offline configuration file of the collector. By default, the file is available in the following location: `<installation folder of Historian>\GE Digital\<collector name>`. For information, refer to Offline Configuration for Collectors *(on page 2039)*. This option is applicable only if you have selected a cloud destination.

## Configure the OPC Classic HDA Collector Using Historian Administrator

1. Access Historian Administrator *(on page 823)*.
2. Select **Collectors**, and then select the OPC Classic HDA collector instance that you want to configure.
3. Select **Configuration**.
   The **Configuration** section appears.




4. Enter values as specified in the following table.

| Field | Description |
|---|---|
| OPC HDA Server | The program ID of the OPC server from which you want to collect data. |
| Recovery Time | The maximum time, in hours, for which the collector will attempt to recover data after the collector is started or when connection between the collector and the OPC server is re-established. This time is calculated as the duration between the current time and the last known write time. |

| Field | Description |
|---|---|
| | Continuous data collection is resumed only after the previous data has been recovered.<br><br>You can enter a value between 1 and 150. |

5. Select **Update**.
6. Restart the collector.
   The collector is configured.

# Specifying Tags for Data Collection

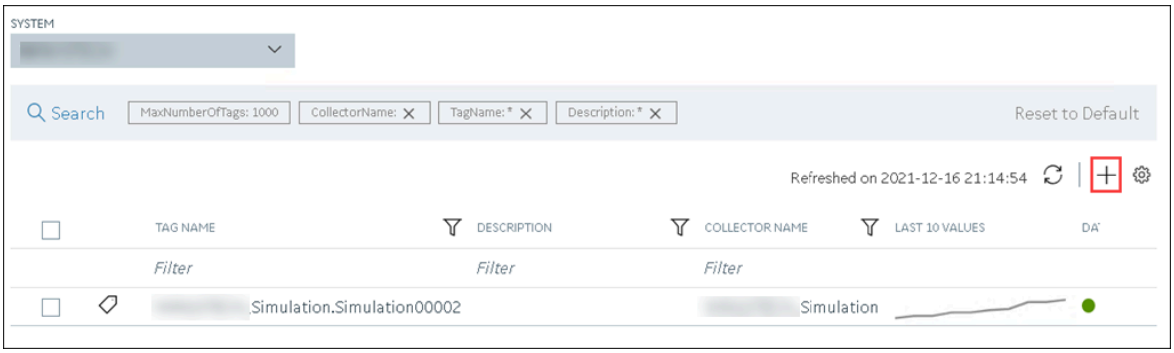
## Add Tags for the Data Store Using Configuration Hub

- Add the collector instance *(on page 357)* using which you want to collect data. Ensure that the collector is running.
- By default, the tag data is stored in the user data store, which is created automatically when you set up Configuration Hub. If, however, you want to store the data in a different data store, create it *(on page 370)*.

This topic describes how to specify the tags for which you want to collect data by browsing through the tags in the data source. For example, for an iFIX collector, if there are 1,00,000 tags in the iFIX server, you must specify the ones for which you want to collect data. Only then data is collected for those tags.

In addition to adding tags from the data source, you can create tags manually *(on page 473)*.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
3. Select +.

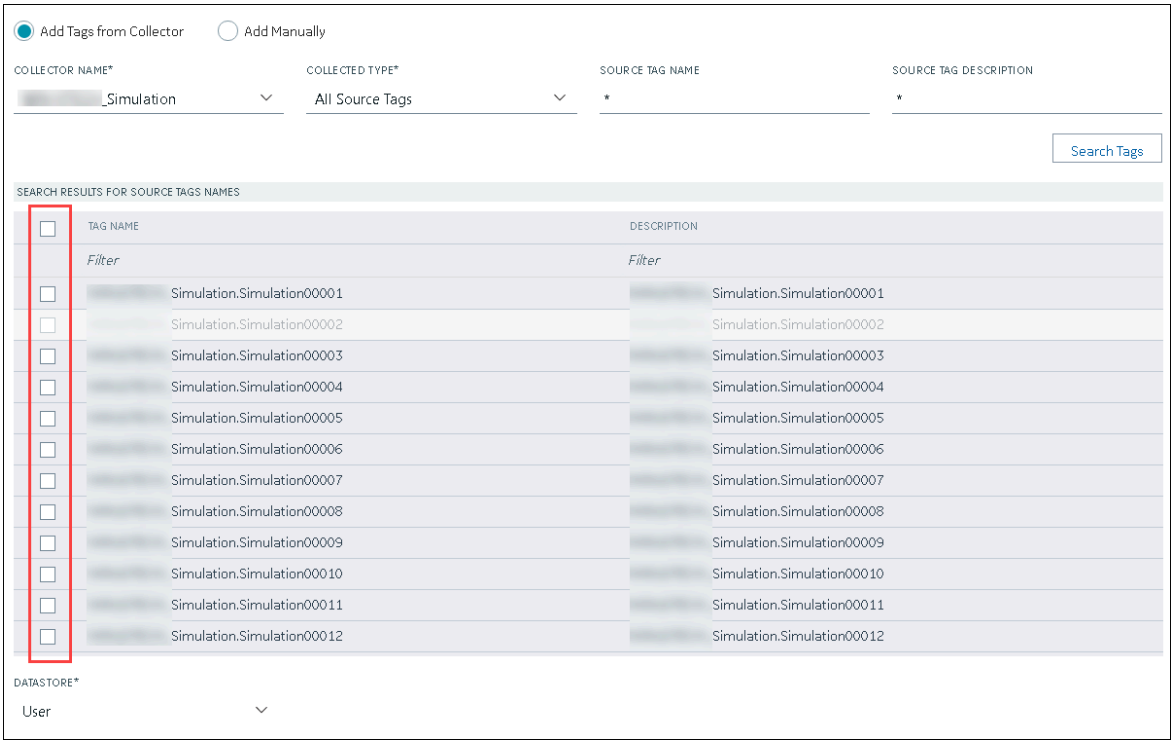


The **Add Tag-<system name>** page appears. The **Add Tags from Collector** option is selected by default.

4. Enter values as described in the following table.

| Field | Description |
|---|---|
| **COLLECTOR NAME** | Select the collector instance that you want to use to collect data. A value is required. |
| **COLLECTED TYPE** | Specify whether you want to browse through all the tags in the data source or only from the tags that you have not added yet. A value is required. |
| **SOURCE TAG NAME** | Enter the name of the tag (either completely or partially) to narrow down the search results. |
| **SOURCE TAG DESCRIPTION** | Enter the description of the tag (either completely or partially) to narrow down the search results. |

5. Select **Search Tags**.
   A list of tags that match *all* the criteria that you have specified appears. If a tag is already added, it is disabled.
6. Select the check box corresponding to each tag for which you want to collect data.



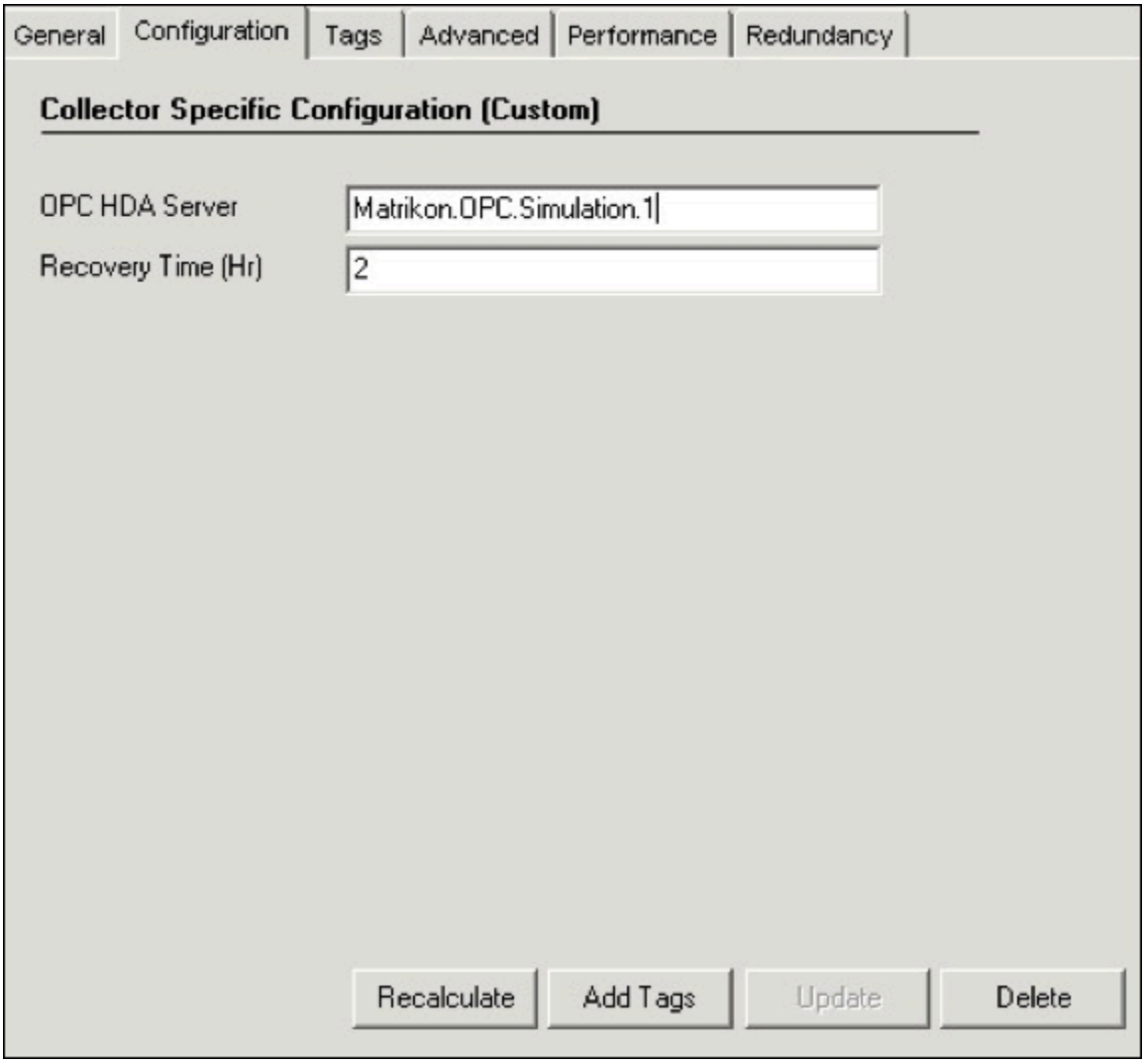
7. In the **DATA STORE** field, if you want to store the data in a different data store than the user data store, select the same.
8. Select **Add Tag**.
   Data collection begins for the selected tags.

As needed, configure each tag by providing values for the tag properties. For information on the delta query modes, refer to Counter Delta Queries.

## Specify the Tags for Data Collection Using Historian Administrator

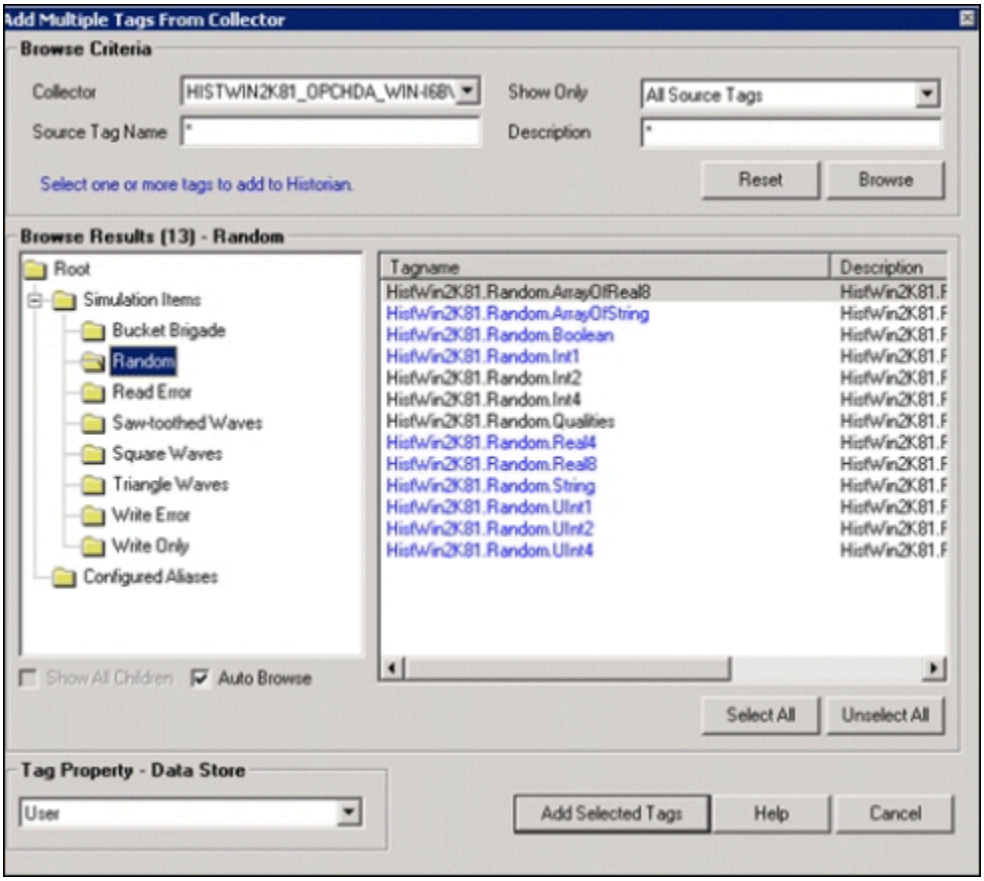
If your OPC server supports hierarchical organization of tags in a tree structure, you can use the hierarchy to browse for tags and add them to the collector for data collection.

1. Access Historian Administrator.
2. Select **Collectors**, and then select the OPC Classic HDA collector instance to which you want to add tags.
3. Select **Configuration**.
   The **Configuration** section appears.




4. Select **Add Tags**.

   The **Add Multiple Tags from Collector** window appears.




5. In the **Collector** field, select the OPC Classic HDA collector to which you want to add tags.
   A hierarchical tree of tags appears in the **Browse Results** section.
6. If you want to view only the tags for which data is not collected, in the **Show Only** field, select **Source Tags Not Collected**. You can search for a tag by entering search criteria in the **Source Tag Name** or **Description** field.
7. Navigate to the node in the tree that you want to browse, and then select **Browse**.

   **Tip:**
   - To browse automatically, select the **Auto Browse** check box. The available tags appear in the **Browse Results** window whenever a node is selected in the tree.
   - To show all child elements within a hierarchy, select the **Show All Children** check box. All tags at or below the hierarchical level of the selected node in the tree appear in the **Browse Results** window.

   The tags within the selected portion of the OPC Classic HDA server tag hierarchy appear.
8. Select the tags for which you want to collect data, and then select **Add Selected Tags**.
   The tags are added to the collector. They appear in black text in the list of tags.

# Data Recovery

**Note:**
We recommend that the collector for which the data recovery is intended is in the same time zone as the server. If there is a mismatch, there is a possibility that data recovery will be incomplete.

## Automatic Data Recovery

In this mode, data is automatically recovered since the last time data has been collected.

**How it works:**

1. The collector determines the duration between the current time and the last time data has been written to the Historian data archive, which is stored in the LastSampleWriteTime registry key.
2. It compares this duration with the value in the Recovery Time field specified in the collector settings *(on page 2347)*.
3. It uses the shorter duration to perform a raw data query on all the tags.
4. It then processes the returned samples in chronological order.

For example, if the collector was stopped for 8 hours, but Max Recovery Time was 4 hours, only 4 hours of data would be recovered.

As per the recovery logic, an end-of-collection marker is placed at the point in time where the collector was stopped. This end-of-collection marker may or may not be there after the recovery is complete. As part of the recovery logic, if recovery data point time matches the timestamp of the end-of-collection marker, it is overwritten with the recovered good data.

## Manual Data Recovery

In this mode, you can fill gaps in the data, but you cannot fill old data.

**To perform a manual recovery:**

1. Access Historian Administrator.
2. Select **Collectors**, and then select the OPC Classic HDA collector instance for which you want to manually recover data.
3. Select **Recalculate**.

   The **Recalculate** window appears.

4. Enter start time, end time, and other required information. We recommend that you choose small time intervals to reduce the load on the server and the collector.
5. Select **Recalculate**.

The tag data is recalculated. After the manual recalculation begins, the collector recovers data of the selected tags data from the collector, and sends it to Historian between the start time and end time.

At the time of recovery, if the connection to server is lost, and if the reconnect mechanism is enabled, the collector will try to connect to the server and fetch the data once connection re-establishes.

### Manual Data Recovery

Assume that the collector is connected to Historian for the first time today and the archive was created at 10 am. The user initiates manual recalculation from 1am to 2 am. For that time interval, the archives were not even created. With respect to Historian, it is old unknown data and the data write fails. If there is a data gap between 1am and 2 am, manual recalculation successfully fills the data gap.

## Reconnect to the OPC HDA Server Automatically

You can reconnect to the OPC Classic HDA server automatically as soon as the server is up and running. By default, the collector polls for the server connection every 5 seconds. You can change this interval as well. The collector is stopped until reconnected to the server.

1. Access the following registry key: `HKEY_LOCAL_MACHINE\SOFTWARE\GE Digital \iHistorian\Services\OPCHDACollector`.
2. Locate the Key created with the ProgID of the OPC Classic HDA server.
3. Create a DWORD named `EnableOPCHDAReconnect`.
4. Enter the decimal value 1.
5. If you want to change the reconnection interval (from the default value of 5 seconds):
    a. Create a DWORD named `ReconnectInterval`.
    b. Enter a decimal value between 5 and 60. This value represents the number of seconds for the collector to wait before trying to reconnect to the OPC server.
6. Select **OK**, then close the registry.

## Troubleshooting the OPC Classic HDA Collector

The OPC Classic HDA collector generates log files during initialization, configuration, and general operation. By default, they are available in the following folder: `C:\Proficy Historian Data \LogFiles`.

If you encounter issues with the collector:

- Verify that the OPC Classic HDA server is running before the OPC Classic HDA collector starts up.
- If the OPC Classic HDA collector does not start automatically, refer to the log file to identify the issue.
- Enable the `CreateOfflineArchives` option in the destination Historian as the OPC Classic HDA collector sends the old data.

# Chapter 26. The OPC Classic Alarms and Events Collector

## About the OPC Classic Alarms and Events Collector

The OPC Classic Alarms and Events collector collects alarms and events data from an OPC Classic Alarms and Events server (such as CIMPLICITY), and stores it alongside Historian process data.

The OPC Classic Alarms and Events collector does not support pre-processing raw data with Python Expression Tags during collection.

## About Event Types, Categories, and Conditions

The OPC Classic Alarms and Events collector captures all event types, categories, sub-categories, and conditions sent to it by the OPC Alarms and Events server.

### Event Types

Three basic types of events are sent by an OPC Alarms and Events server: Condition, Simple, and Tracking. Each of these types has its own categories, sub-categories, and conditions. For example, a Condition event may have a Level category, which itself may have several conditions, such as LO LO, LO, HI, and HI HI.

**Condition**

Condition events record the transition of states in an alarm. For example, a condition event could be recorded for an alarm when the level changes from LO to HI HI.

**Tracking**

Tracking events are not associated with conditions, but rather track activity between the OPC Alarms and Events server and an OPC client. For example, if an operator acknowledges an alarm, a tracking event is recorded.

**Simple**

Simple events record everything not covered by Condition or Tracking events. For example, if a device were to fail, a simple event would be recorded.

### Event Categories

Event categories are used to group similar event types and are configured on the OPC Alarms and Events server. For example, you might set up categories for System Events, Process Events, and Batch Events.

You might likewise set up categories for different areas of your process, such as Premix, Dry Mix, or Bake. Categories can hold multiple event types, and a given source can generate events for multiple categories.

**Note:**
Category names must be unique within the OPC Alarms and Events server.

### Event Conditions

Conditions are named states of alarms and events within the OPC Alarms and Events server. Conditions can include LO LO, LO, HI, and HI HI, as well as SYSTEM_FAILURE, LIMIT EXCEEDED, NORMAL STATE, and others. Conditions may also contain sub-conditions, which help to narrow down the event conditions further. Refer to your OPC server documentation for a complete listing of its conditions.

For more information, refer to your OPC server documentation.

## About Event Attributes

Events usually also include attributes, which give greater detail to the status of the event. Attributes vary from server to server; user-defined attributes as well as vendor-defined attributes may be configured on your OPC Alarms and Events server.

Some common attributes are:

- Start time
- End time
- Acknowledgement status
- Acknowledgement time
- Operator name
- Data Source
- Quality
- Severity

Historian archives all event attributes sent to it by the OPC Alarms and Events server. Consult your OPC Alarms and Events server documentation for more information.

## Workflow for Using the OPC Alarms and Events Collector

To use the OPC Classic Alarms and Events collector, you must perform the following tasks.

| Number | Task | Notes |
|---|---|---|
| 1 | Install the collector *(on page 142)*. | This step is required. This will place the collector binaries on the machines. |
| 2 | Add an instance *(on page 357)* of the OPC Classic Alarms and Events collector. | This step is required. |
| 3 | Configure the general options of the OPC Classic Alarms and Events collector by accessing the collector using Historian Administrator. | This step is required. |
| 4 | Configure the collector-specific options of the OPC Classic Alarms and Events collector. | This step is required. |
| 5 | Specify the filtering options for the alarms and events data *(on page 2393)*. | This step is required. |

# Configure the OPC Alarms and Events Collector

The following table provides the OPC Classic Alarms and Events collector-specific configuration fields.

1. Access Historian Administrator.
2. Access the OPC Classic Alarms and Events collector, and then select **Configuration**.
3. Provide values as specified in the following table.

| Field | Description |
|---|---|
| **OPC Server PROGID** | The PROGID of the OPC Alarms and Events server. |
| **Link to Data Collector** | The data collector to link to the alarms and events. This allows you to join alarms and events data with tag data when querying the Historian database for data. This is usually located on the same server as the OPC Alarms and Events server. |

| Field | Description |
|---|---|
| **Filtering** | Enables or disables filtering of the alarms and events data. For information, refer to Filter Alarms and Events Data *(on page 2393)*. |
| **Show Last Alarms** | Displays the last 10 collected alarms and events data points. |
| **Close Alarms** | Opens the **Close Alarms** window. |

**Important:**
Although the collector will function properly with no associated data collector, alarms and events data will not be associated with tag data from the data collector if it is not specified in this field. As a result, queries through the Excel Add-in or the OLE DB provider will not be able to join tag and alarm data.

# Filter Alarms and Events Data

By default, the OPC Classic Alarms and Events collector collects all the alarms and events data sent to it, and archive it. This ensures that all your alarms and events data will be archived, without any special configuration. If you archive all of your alarms and events data, it can impact the amount of storage required for Historian to operate. Therefore, you may want to specify which alarms and events data you want the OPC Classic Alarms and Events collector to collect. Alarms and Events filtering works on an inclusive model. If filtering is not enabled, all the alarms and events data is collected. If filtering is enabled, then data for only the selected alarms and events is collected.

This topic describes how to apply the various types of filters.

1. Access Historian Administrator.
2. Select **Collectors**.
3. Select the OPC Classic Alarms and Events collector instance that you want to configure.
4. Select **Configuration**.
5. In the **Filtering** section, select **Enabled**.
6. Select **Filters**.
7. Select the filter criterion as described in the following table.

| Filtering Option | Description | Procedure |
|---|---|---|
| Severity range | Includes alarms between a low and high filter range. For example, filter alarms whose severity range is between 100 and 200. | a. Select the **Filter by Severity Range** check box.<br>b. Enter a range of values in the **Collect From** and **To** boxes. |
| Event Type | Include events based on a selected type *(on page 2390)*. | a. Select the **Filter by Event Type** check box.<br>b. Select the type of events you want to filter. |
| Area | Includes alarms and events based a user-defined process area. This is useful if you only want to collect alarms from specific process areas. This option works only if you have defined areas in the alarms and events server. | a. Select the **Filter by Area** check box.<br>b. Select **Edit**.<br>c. Select the areas by which you want to filter, and then select **Copy**. To add an area manually, enter the area you want to filter by in the **Area** box and select **Copy**. |
| Source | Includes alarms and events data based on the alarm source. This is useful if you only want to collect alarms from specific parts of your process. | a. Select the **Filter by Source** check box.<br>b. Select **Edit**.<br>c. Select the sources using which you want to filter, and then select **Copy**. To add a source manually, enter the source you want to filter by in the **Source** box and select **Copy**. |

| Filtering Option | Description | Procedure |
|---|---|---|
| Event Category | Includes events based on a selected category *(on page 2390)*. | a. Select the **Filter by Event Category** check box.<br>b. Select **Edit**.<br>c. In the **Choose Event Category** box, select an event category.<br>d. In the **Categories Available** box, select the categories using which you want to filter data. To add a category manually, enter the category you want to filter by in the **Category** box, and then select **Copy**. |

**Note:**
You can filter alarms and events by the event category only if the **Filter by Event Type** check box is selected for respective event category. For example, if you want to receive only alarms, enable the **Collect Condition Events** option in **Event Type**, and add the Event Category tag in the **Filter by Event Category** section. This is because alarms belong to the Event Category tag, and the Event Category tag belongs to the Event Type condition. For other mappings, refer to iFIX Message Mappings.

8. Select **Update**.
   The filtering options you have specified are saved.

# Chapter 27. OPC Classic HDA Server

## About OPC Classic HDA

**About OPC Classic Historical Data Access (HDA)**

OPC Classic HDA is widespread standard, which provides specifications to retrieve and analyze historical process data. This data is typically stored in a process data archive, database, or a remote terminal unit (RTU). You can analyze this data for trending, fault prediction, performance assessment, and so on.

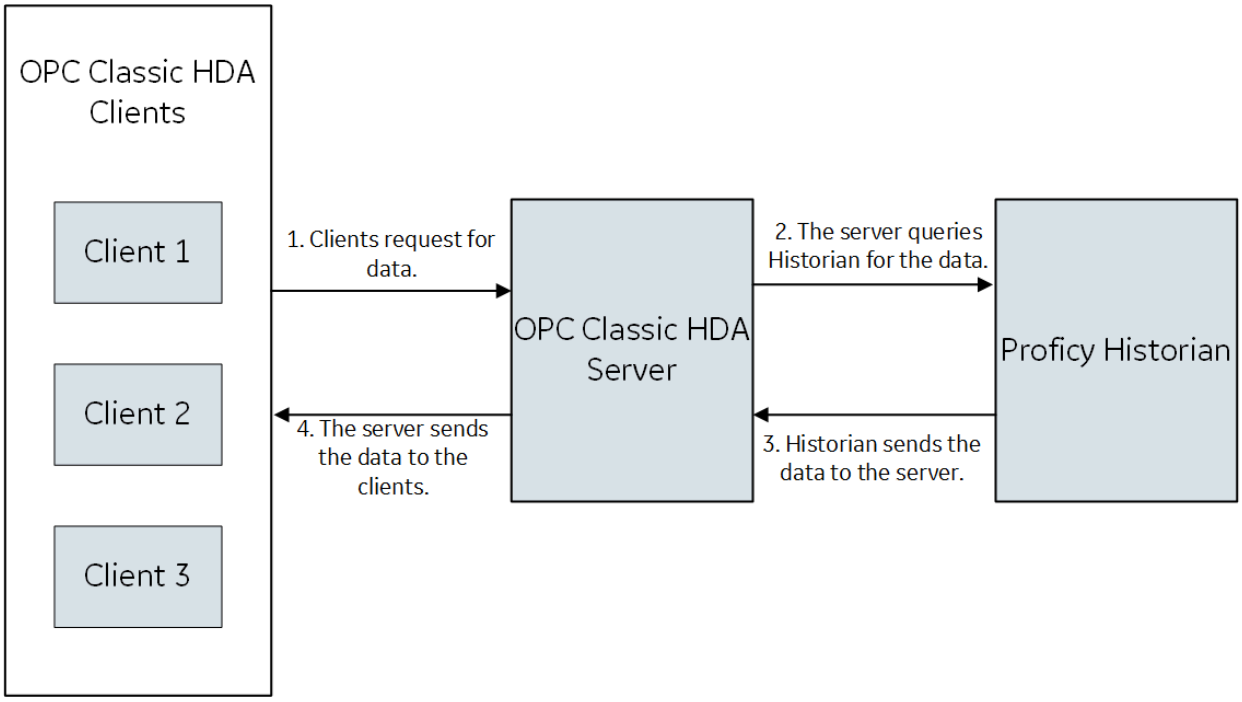
**Advantages of Using OPC Classic HDA**

- With OPC Classic HDA, the exchange of historical data between an application and any data archive is consistent. Therefore, OPC HDA client applications that implement trends, reports, or spreadsheets can retrieve historical process data from Historian and other OPC Classic HDA servers.
- The OPC Classic HDA specifications are based on Microsoft's Object Linking and Embedding (OLE) and Distributed Component Object Model (DCOM) technologies. Therefore, OPC is endorsed by Microsoft.
- You can configure DCOM client and server software programs to run on the same computer node or distributed across a network of computers.
- The OPC Classic HDA is created to allow various automation applications to communicate with one another based on the historical data, regardless of the manufacturer. This allows greater flexibility and reliability when setting up automation systems.
- The OPC Classic HDA server specification provides a common view of automation information managed by the system for which the server was written.
- Developing OPC Classic-compliant applications is simplified because only one I/O interface is required.
- Using OPC-compliant applications increases the flexibility of your automation processes because they can also communicate with devices other than those specified by the applications' developers.
- Multiple OPC Classic HDA compliant client applications can communicate with an OPC Classic HDA server simultaneously.

For more information on OPC Classic HDA, visit the OPC Foundation's website.

**Data Flow**

The following diagram shows the data flow between the OPC Classic HDA clients, the OPC Classic HDA server, and Proficy Historian.




## About the Historian OPC Classic HDA Server

The Historian OPC Classic HDA server retrieves historical process data from Proficy Historian, and sends it to OPC Classic HDA clients. It dynamically updates the clients when tags are added and/or deleted in Historian. Clients that comply with this specification can connect to the OPC Classic HDA server to retrieve data from Historian.

**Features of the OPC Classic HDA Server**

- The server complies with the OPC HDA Server specification 1.2.0.
- You can browse for all the tags available in Historian.
- You can convert Historian timestamps, data types, and qualities to OPC HDA timestamps, data types, and qualities, respectively.
- You can automatically reconnect to the Historian server when connection is lost.
- You can connect multiple instances of the OPC Classic HDA clients to the same server without any additional configuration. The OPC Classic HDA server has been tested with the following OPC Classic HDA clients:
    - OPC Foundation HDA Sample client. You can download this client from www.opcfoundation.org
    - Advosol HDA test client. You can download this client from http://www.advosol.com/t-free-tools.aspx

**Limitations**

- The OPC Classic HDA server supports only synchronous read raw interface.

# Setting Up

## Set Up the Historian OPC Classic HDA Server

- Install the HDA server and Historian Administrator. For instructions, refer to Install Client Tools *(on page 150)*.
- Disconnect and reconnect all the OPC HDA clients.

1. Access Historian Administrator.
2. Select **Browse for Server**, and select the Historian server that you want to connect with the OPC Classic HDA server.
3. Select **Set Selected Server as Target of HDA Server**, and then select **OK**.
   The selected server is set as the target Historian server.
4. Configure any external OPC HDA clients to connect to the OPC Classic HDA server. You can connect multiple instances of the clients to a single server.
   When an OPC HDA client tries to connect to the Historian OPC Classic HDA server, and if an instance of the server is not available, the instance is started, and the client connects to the running instance of the server. The Historian OPC Classic HDA server continues to run as long as there are clients connected to it.

   **Note:**
   The OPC Classic HDA server uses Proficy.Historian.HDA as the program ID (ProgID).

If you have installed the OPC Classic HDA server on a remote machine, enable the firewall.

## Enable Tag-Level Security

1. Access Registry, and go to the following path: `HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Intellution, Inc.\iHistorian\HDAServer`
2. In the **TargetHistorianServer** registry key, enter details in following format:
   `MachineName,UserName,Password,Timeout`

## Turn On Debug Mode for Trace Log Files

The Trace Logging feature logs communications with the OPC Classic HDA server into an html file. Advanced users can use this log file to trace a history of communication events with the OPC Classic HDA collector server. It is designed for use by the support personnel only to assist in diagnosing issues with the Historian OPC Classic HDA server. By default, the log file is located in the following folder: `C:\Proficy Historian Data\LogFiles`.

The log file contains the following information:

- The date and time when an event has occurred.
- Errors and exceptions in parameters passed by an OPC HDA client.
- The status of an OPC HDA client.

For an example trace log file, refer to Example Trace Log File *(on page 2403)*.

1. Access Registry, and then access the following key folder:
    - For Windows 32-bit: `HKEY_LOCAL_MACHINECLASSES_ROOT\SOFTWARE\Intellution,inc\iHistorian\HDAServer\`
    - For Windows 64-bit: `HKEY_LOCAL_ MACHINE\SOFTWARE\Wow6432Node\\Intellution, Inc.\iHistorian\HDAServer\`
2. Modify values for **DebugMode DWORD** as follows, and then select **OK**:
    a. In the **Base** box, select **Decimal**.
    b. In the **Value** box, enter `255`.

The debug mode is turned on for the trace logs.

## Browse Large Number of Collectors and Tags

In the OPC UA HDA server, the flat address space is used by default. This type of model is used to browse a small number of tags. If you want to browse a large number of collectors and tags, enable the hierarchical address space.

1. Access Registry, and then go to the following key folder: .
    - For Windows (32-bit): `HKEY_LOCAL_MACHINE\SOFTWARE\Intellution, Inc.\iHistorian\HDAServer\`
    - For Windows (62-bit): `HKEY_LOCAL_ MACHINE\SOFTWARE\Wow6432Node\\Intellution, Inc.\iHistorian\HDAServer\`
2. Create a DWORD named `HierarchicalAddressSpace`.
3. In the **Value** box, enter `255`.

> **Note:**
> For a flat address space, this value is set to 0.

4. Select **OK**, and then close the Registry.
5. Restart the OPC Classic HDA server by disconnecting and reconnecting all the OPC HDA clients.

# Reference

## Supported Attributes

The following table provides a list of the attributes supported by the OPC Classic HDA collector.

| Historian Property Type | OPC Classic HDA collector Attributes | Data Type | Description |
|---|---|---|---|
| Data Type | OPCHDA_ DATA_TYPE | Int16 | The data type for the item. For information, refer to Supported Data Types *(on page 2400)*. |
| TagDescription | OPCHDA_ DESCRIPTION | String | The description of the item. |
| Engineering Units | OPCHDA_ ENG_UNITS | String | The units for the item (for example, kg/sec). |
| Archiving | OPCHDA_ ARCHIVING | Boolean | Indicates whether Historian stores data for this item. The following values are valid:<br>• 0: Indicates that data for this item is not stored.<br>• 1: Indicates that data for this item is stored. |
| ItemID | OPCHDA_ ITEMID | String | The Historian tag name. This is used to allow filtering in `Create-Browsemethod`. |

## Supported Data Types

The following table provides a list of data types in Historian and the corresponding ones in the OPC Classic HDA server and the OPC UA HDA server.

| Data Type in Proficy Historian | Data Type in the OPC Classic HDA server or the OPC UA HDA server. |
|---|---|
| Single Integer | VT_I2&endash; 16 bit signed integer |
| Double Integer | VT_I4- 32 bit signed integer |
| Quad Integer | VT_I8- 64 bit quad integer |
| Unsigned Single Integer | VT_UI2- 16 bit unsigned single integer |
| Unsigned Double Integer | VT_UI4- 32 bit unsigned single integer |
| Unsigned Quad Integer | VT_UI8- 62 bit quad integer |
| Byte | VT_I1 |
| Boolean | VT_BOOL |
| SingleFloat | VT_R4- 32 bit float |
| Double Float | VT_R8- 64 bit double float |
| Variable String | VT_BSTR |
| Fixed String | VT_BSTR |
| Date | VT_DATE |
| Blob | VT_BSTR |

## Supported Quality Values

The following table provides a list of the quality values in Historian and the corresponding ones in the OPC Classic HDA server.

| Proficy Historian Quality | OPC Classic HDA Quality | Description |
|---|---|---|
| ihOPCGood | OPC_ QUALITY_ GOOD | Indicates that there is no need for inspection. This quality is returned for the tags that have all the values archived properly. |
| ihOPCBad | OPC_ QUALITY_ BAD | Indicates a need for attention. This quality is returned for the tags that had problems during the collection. |

| Proficy Historian Quality | OPC Classic HDA Quality | Description |
|---|---|---|
| ihOPCUncertain | OPC_ QUALITY_ UNCERTAIN | Indicates a need for inspection. This quality is returned when the data collection time is low and when there is no specific quality value. |
| ihOPCNA | OPC_ QUALITY_ BAD | Indicates a need for attention. |

## Supported Filter Attributes

The following table provides a list of attributes for which filtering is supported by the OPC Classic HDA server, and the corresponding ones in Historian, along with the supported operators for each.

| Historian Property Type | OPC Classic HDA server Attributes | Supported Operators |
|---|---|---|
| Data Type | OPCHDA_DATA_TYPE | OPC_HDA_EQUAL_TO |
| Tag Description | OPCHDA_DESCRIPTION | OPC_HDA_EQUAL_TO |
| Engineering Units | OPCHDA_ENG_UNITS | OPC_HDA_EQUAL_TO |
| Item ID | OPCHDA_ITEMID | OPC_HDA_EQUAL_TO |

**Important:**
The OPC Classic HDA server filter strings are case-sensitive.

Although filtering is supported for the OPC_HDA_ARCHIVING attribute, do not use the filtering in the OPC Classic HDA clients because the value of this attribute is always true. Instead, you can refer to the log file to trace a history of communication events dealt with by the OPC Classic HDA server. You can also turn on the debug mode to more know more details in the log file. If an invalid filter is passed by an OPC HDA client, the OPC Classic HDA server returns all the available items in the address space.

**Tip:**

- The wildcard character (*) will return all the tags.
- The wildcard (key*) will return tags which begins with “key”.
- The wildcard (key) returns an exact tag name.

## Example Trace Log File

The following message illustrates the type of information that the trace log file provides. In this message:

- The black color indicates the status of events.
- The blue color indicates the status of clients.
- The red color indicates errors and exceptions passed by clients.

```
---------
3/5/2009 10:12:19 AM  HDA Client is connecting ...
3/5/2009 10:12:20 AM  Connected to Historian server : [ES-4FCNR1S]
3/5/2009 10:12:20 AM  HDA Client Connected
3/5/2009 10:12:21 AM  HDA Address Space Created
3/5/2009 10:12:36 AM  HDA Client Disconnected
3/5/2009 11:50:04 AM  HDA Client is connecting ...
3/5/2009 11:50:05 AM  Connected to Historian server : [ES-4FCNR1S]
3/5/2009 11:50:05 AM  HDA Client Connected
3/5/2009 11:50:25 AM  Historian Archiver : Tag Added [ES-4FCNR1S.Simulation00261]
3/5/2009 11:50:25 AM  Historian Archiver : Tag Added [ES-4FCNR1S.Simulation00262]
3/5/2009 11:50:25 AM  Historian Archiver : Tag Added [ES-4FCNR1S.Simulation00264]
3/5/2009 11:50:32 AM  HDA Address Space Created
3/5/2009 11:50:59 AM  Historian Archiver : Tag Added [ES-4FCNR1S.SimulationString00001]
3/5/2009 11:51:25 AM  Historian Archiver : Tag Added [ES-4FCNR1S.SimulationString00002]
3/5/2009 11:51:25 AM  Historian Archiver : Tag Added [ES-4FCNR1S.Ramp_20%Noise]
3/5/2009 11:51:25 AM  HDA Client Disconnected
3/5/2009 2:34:27 PM  HDA Client is connecting ...
3/5/2009 2:34:28 PM  Connected to Historian server : [ES-4FCNR1S]
3/5/2009 2:34:28 PM  HDA Client Connected
3/5/2009 2:34:28 PM  HDA Address Space Created
3/5/2009 2:34:45 PM  HDA Client Disconnected
3/5/2009 2:35:06 PM  HDA Client is connecting ...
3/5/2009 2:35:07 PM  Connected to Historian server : [ES-4FCNR1S]
3/5/2009 2:35:07 PM  HDA Client Connected
3/5/2009 2:35:08 PM  HDA Address Space Created
3/5/2009 2:35:08 PM  HDA Address Space Created
3/5/2009 2:35:08 PM  Invalid attribute at index [0]
3/5/2009 2:35:08 PM  HDA Address Space Created
3/5/2009 2:35:08 PM  Invalid type at index [0]
3/5/2009 2:35:08 PM  HDA Address Space Created
3/5/2009 2:36:00 PM  HDA Client is connecting ...
3/5/2009 2:36:01 PM  Connected to Historian server : [ES-4FCNR1S]
3/5/2009 2:36:01 PM  HDA Client Connected
3/5/2009 2:36:11 PM  HDA Client is connecting ...
3/5/2009 2:36:11 PM  HDA Client Connected
3/5/2009 2:36:12 PM  HDA Address Space Created
3/5/2009 2:36:12 PM  Invalid attribute at index [0]
3/5/2009 2:36:12 PM  HDA Address Space Created
3/5/2009 2:36:12 PM  Invalid type at index [0

---------
```

## OPC Classic HDA Aggregates

In the OPC Classic HDA server, an aggregate is a function that is used to process raw data from an OPC Classic HDA server over a given range of time divided into discrete intervals. These aggregates can be further used to for various purposes, such as visualizing the trends in the data.

The standard aggregates supported are:

- Minimum
- Maximum
- Average

The custom aggregates supported are:

- Nearest
- Before
- After
- Proficy Historian Interpolative

Consider the following as a Historian Source for the Minimum, Maximum and Average Aggregates:

| Timestamp | Value | Quality | Notes |
|---|---|---|---|
| Jan-01-2002 12:00:00 | 0 | No Data | First archive entry Point Created |
| Jan-01-2002 12:00:10 | 10 | Raw, Good | |
| Jan-01-2002 12:00:20 | 20 | Raw, Good | |
| Jan-01-2002 12:00:30 | 30 | Raw, Good | |
| Jan-01-2002 12:00:40 | 40 | Raw, Bad | Scan failed, Bad data entered |
| Jan-01-2002 12:00:50 | 50 | Raw, Good | |
| Jan-01-2002 12:01:00 | 60 | Raw, Good | |
| Jan-01-2002 12:01:10 | 70 | Raw, Bad | Value is flagged as questionable |
| Jan-01-2002 12:01:20 | 80 | Raw, Good | |
| Jan-01-2002 12:01:30 | 90 | Raw, Good | |
| NULL | No Data | No more entries, awaiting next scan. | |

Consider the following Historian Source for Nearest, Before and After Aggregates:

| Timestamp | Value | Quality | Notes |
|---|---|---|---|
| Feb-28-2002 12:01:30 | 987 | Raw, Bad | |
| Mar-01-2002 12:01:30 | 98765 | Raw, Good | |
| Mar-02-2002 12:01:30 | 9876 | Raw, Bad | |
| Jan-01-2002 12:00:00 | 0 | No Data | First archive entry Point Created |
| Jan-01-2002 12:00:10 | 10 | Raw, Good | |
| Jan-01-2002 12:00:20 | 20 | Raw, Good | |
| Jan-01-2002 12:00:30 | 30 | Raw, Good | |
| Jan-01-2002 12:00:40 | 40 | Raw, Bad | Scan failed, Bad data entered |
| Jan-01-2002 12:00:50 | 50 | Raw, Good | |
| Jan-01-2002 12:01:00 | 60 | Raw, Good | |
| Jan-01-2002 12:01:10 | 70 | Raw, Bad | Value is flagged as questionable |
| Jan-01-2002 12:01:20 | 80 | Raw, Good | |
| Jan-01-2002 12:01:30 | 90 | Raw, Good | |

## Average Aggregate

The purpose of Average aggregation is to find the average value for a given time interval. It adds up the values of all good raw data in a specified time interval. The sum is then divided by the number of good values. Bad values are ignored in the computation.

If the user specifies a time range where no good data exists for an interval, the quality of the aggregate for that interval will be bad, OPCHDA_NODATA.

This aggregate returns the timestamp of the start of the interval.

Example 1

**Start**: Jan-01-2002 12:00:10 **End**: Jan-01-2002 12:00:20 **Interval**: 00:00:05

| Timestamp | Value | Quality |
|---|---|---|
| Jan-01-2002 12:00:10 | 10 | Calculated, Good |
| Jan-01-2002 12:00:15 | 0 | No Data, Bad |

Example 2

**Start**: Jan-01-2002 12:00:35 **End**: Jan-01-2002 12:01:00 **Interval**: 00:00:05

| Timestamp | Value | Quality |
|---|---|---|
| Jan-01-2002 12:00:35 | 0 | No Data, Bad |
| Jan-01-2002 12:00:40 | 0 | Bad data in the interval |
| Jan-01-2002 12:00:45 | 0 | No Data, Bad |
| Jan-01-2002 12:00:50 | 50 | Calculated, Good |
| Jan-01-2002 12:00:55 | 0 | No Data, Bad |

## Maximum Aggregate

The maximum actual time aggregate retrieves the maximum good raw value within the interval [s,e), and returns that value the timestamp of the aggregate will always be the start of the interval for every interval. If the same maximum exists at more than one timestamp, the oldest one is retrieved.

Example 1

**Start**: Jan-01-2002 12:00:10 **End**: Jan-01-2002 12:00:20 **Interval**: 00:00:05

| Timestamp | Value | Quality |
|---|---|---|
| Jan-01-2002 12:00:10 | 10 | Raw, Good |
| Jan-01-2002 12:00:15 | 0 | No Data, Bad |

Example 2

**Start**: Jan-01-2002 12:00:35 **End**: Jan-01-2002 12:01:00 **Interval**: 00:00:05

| Timestamp | Value | Quality |
|---|---|---|
| Jan-01-02 12:00:35 | 0 | No Data, Bad |
| Jan-01-02 12:00:40 | 0 | No Data, Bad |
| Jan-01-02 12:00:45 | 0 | No Data, Bad |
| Jan-01-02 12:00:50 | 50 | Raw, Good |
| Jan-01-02 12:00:55 | 0 | No Data, Bad |

## Minimum Aggregate

The minimum actual time aggregate retrieves the minimum good raw value within the interval [s,e), and returns that value. The timestamp of the aggregate will always be the start of the interval for every interval.

Example 1

**Start**: Jan-01-2002 12:00:10 **End**: Jan-01-2002 12:00:20 **Interval**: 00:00:05

| Timestamp | Value | Quality |
|---|---|---|
| Jan-01-02 12:00:35 | 0 | No Data, Bad |
| Jan-01-02 12:00:40 | 0 | Bad data in the interval |
| Jan-01-02 12:00:45 | 0 | No Data, Bad |
| Jan-01-02 12:00:50 | 50 | Raw, Good |
| Jan-01-02 12:00:55 | 0 | No Data, Bad |

## Before Aggregate

When the Before aggregate function is called with a timestamp (Start time), the previous valid value before Start time will be fetched. This search is performed between **Start time** and (**Start time – MaxDurationSec**). The call for the aggregation is initiated by the source client. The custom aggregations do not require any interval value. The default value for MaxDurationSec is 100 days.

**Note:**
To change the value of MaxDurationSec navigate to `HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Intellution, Inc.\iHistorian\HDAServer`. Create a string value with name **MaxDurationSec** and provide the value in seconds.

**Start**: Apr-01-2002 12:00:10 **End**: NA **Interval**: NA

| Timestamp | Value | Quality |
|---|---|---|
| Mar-01-2002 12:01:30 | 98765 | Raw, Good |

## After Aggregate

When the After aggregate function is called with a timestamp (Start time), next valid value after 'Start time' will be fetched. This search will happen between **Start time** and (**Start time + MaxDurationSec**).

The call for the aggregation is initiated by the source client. The custom aggregations do not require any interval value. The default value for MaxDurationSec is 100 days.

**Note:**

To change the value of MaxDurationSec navigate to `HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Intellution, Inc.\iHistorian\HDAServer`. Create a string value with name **MaxDurationSec** and provide the value in seconds.

**Start**: Feb-01-2002 12:00:10 **End**: NA **Interval**: NA

| Timestamp | Value | Quality |
|---|---|---|
| Mar-01-2002 12:01:30 | 98765 | Raw, Good |

## Nearest Aggregate

When the Nearest aggregate function is called with a timestamp (Start time), the nearest valid value of Start time will be fetched. This search is performed between **Start time** and (**Start time ±**

**MaxDurationSec**). The call for the aggregation is initiated by the source client. The custom aggregations do not require any interval value. The default value for MaxDurationSec is 100 days.

**Note:**
To change the value of MaxDurationSec navigate to `HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Intellution, Inc.\iHistorian\HDAServer`. Create a string value with name **MaxDurationSec** and provide the value in seconds.

**Start**: Feb-01-2002 12:00:10 **End**: NA **Interval**: NA

| Timestamp | Value | Quality |
|---|---|---|
| Mar-01-2002 12:01:30 | 98765 | Raw, Good |

## Proficy Historian Interpolative Aggregate

The purpose of the interpolated aggregation is to return meaningful data for a time interval where raw data is not present. The value at a datapoint is estimated based on the raw data present at other datapoints.

For interpolative aggregate to return meaningful data, good values should be present at the boundary conditions.

# Chapter 28. OPC UA HDA Server

## Overview of the OPC UA HDA Server

**About OPC Unified Access (UA) Historical Data Access (HDA)**

OPC UA HDA is widespread standard, which provides specifications to retrieve and analyze historical process data. This data is typically stored in a process data archive, database, or a remote terminal unit (RTU). You can analyze this data for trending, fault prediction, performance assessment, and so on.

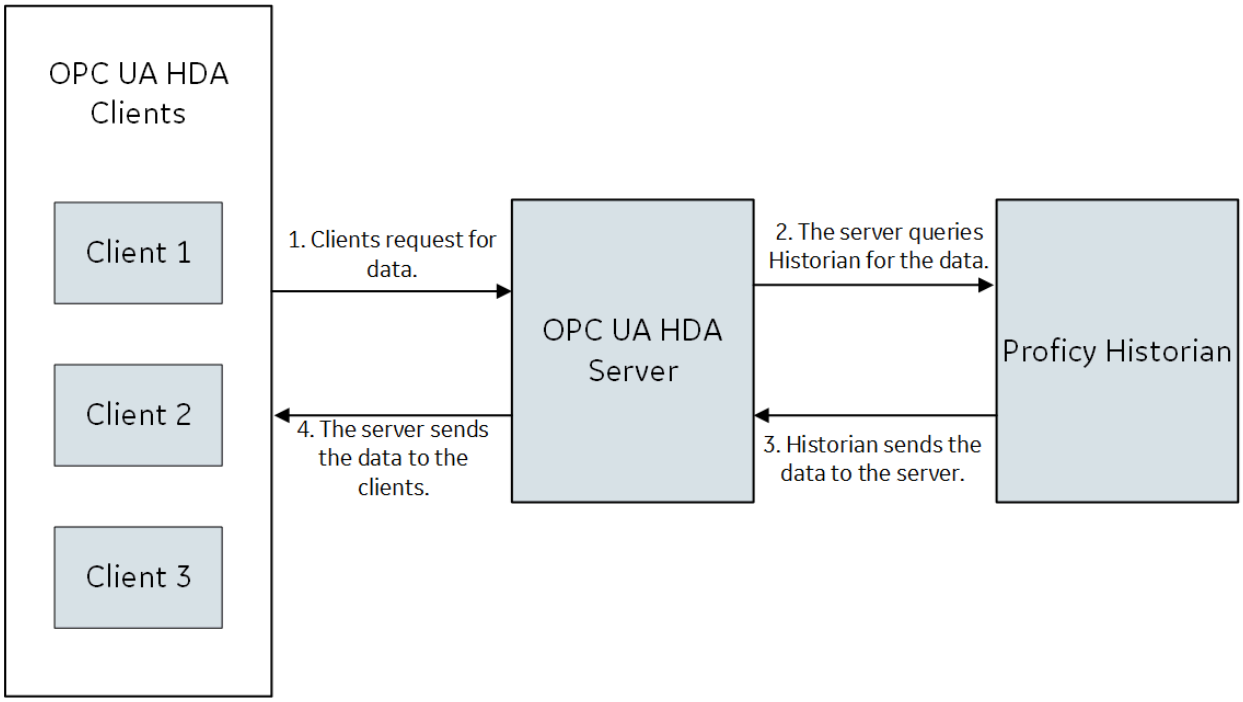
**Advantages of Using OPC UA HDA**

- With OPC UA HDA, the exchange of historical data between an application and any data archive is consistent. Therefore, OPC UA HDA client applications that implement trends, reports, or spreadsheets can retrieve historical process data from Historian and other OPC UA HDA servers.
- OPC UA HDA is created to allow various advanced automation applications to communicate with one another based on the historical data, regardless of the manufacturer or the platform/ programming language on which they have been created. This allows greater flexibility and reliability when setting up automation systems.
- The OPC UA HDA server specification provides a common view of automation information managed by the system for which the server was written.
- Developing OPC UA-compliant applications is simplified because only one I/O interface is required.
- Using OPC-compliant applications increases the flexibility of your automation processes because they can also communicate with devices other than those specified by the applications' developers.
- Multiple OPC UA HDA compliant client applications can communicate with an OPC UA HDA server simultaneously.

For more information on OPC UA HDA, visit the OPC Foundation's website.

**Data Flow**

The following diagram shows the data flow between the OPC UA HDA clients, the OPC UA HDA server, and Proficy Historian.




## About the Historian OPC UA HDA Server

The Historian OPC UA HDA server retrieves historical process data from Proficy Historian, and sends it to OPC UA HDA clients. It dynamically updates the clients when tags are added and/or deleted in Historian. Clients that comply with this specification can connect to the OPC UA HDA server to retrieve data from Historian.

**Features of the OPC UA HDA Server**

- The server complies with the OPC HDA Server specification 1.7.2.
- The server can retrieve current data as well as historical data.
- You can browse for all the tags available in Historian.
- You can convert Historian timestamps, data types, and qualities to OPC HDA timestamps, data types, and qualities, respectively.
- You can automatically reconnect to the Historian server when connection is lost.
- You can connect multiple instances of the OPC UA HDA clients to the same server without any additional configuration. The OPC UA HDA server has been tested with the following OPC UA HDA clients; you can, however, use any client that supports the encryption types supported by the OPC UA HDA server:

- OPC UA Client. You can download this client from https://opcfoundation.org/products/view/opc-ua-client-free-product.
- Prosys OPC Client. You can download this client from https://www.prosysopc.com/products/opc-client/.

**Limitations**

- The OPC UA HDA server supports only synchronous read raw interface.
- The OPC UA HDA server currently does not support aggregation and alarms and events data. It only supports tag data.
- If you make changes to the OPC UA HDA server settings *(on page 2416)*, restart the OPC UA HDA Server service.

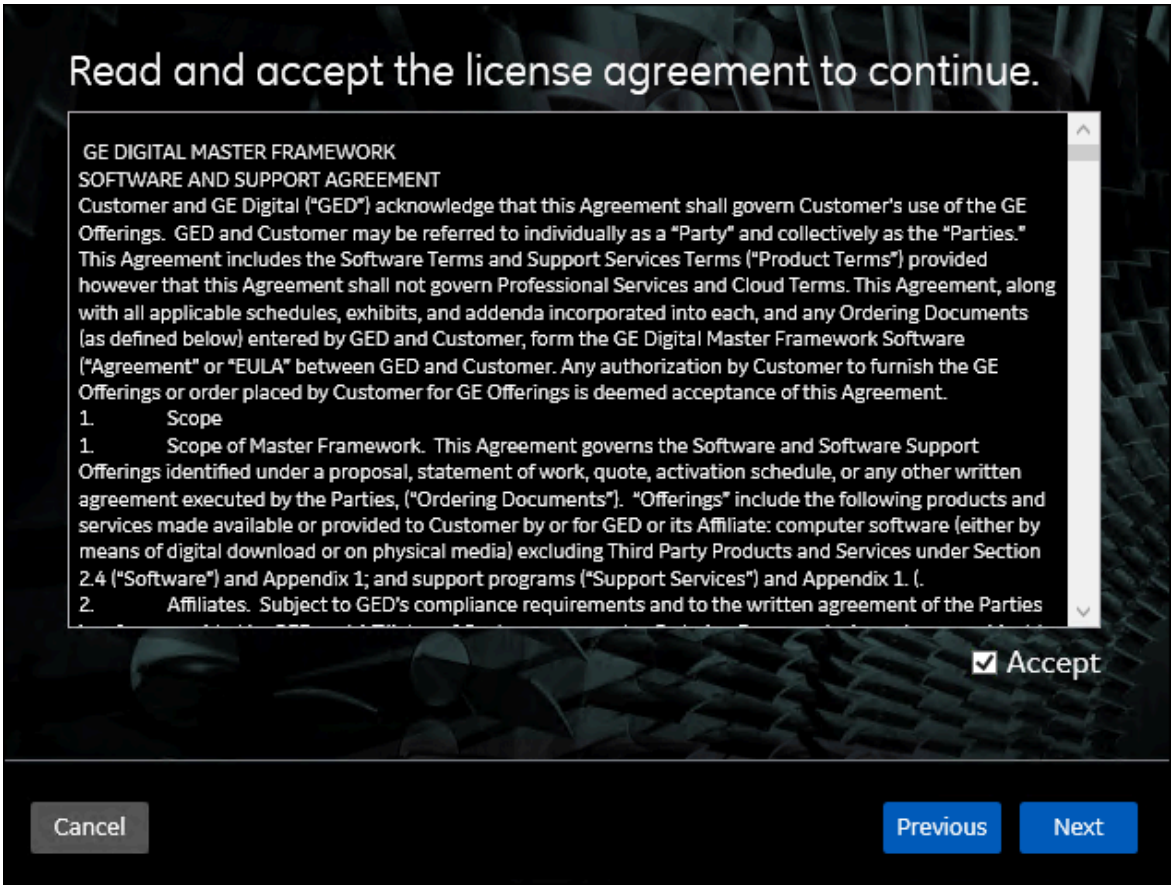
# Configuration

## Install the OPC UA HDA Server

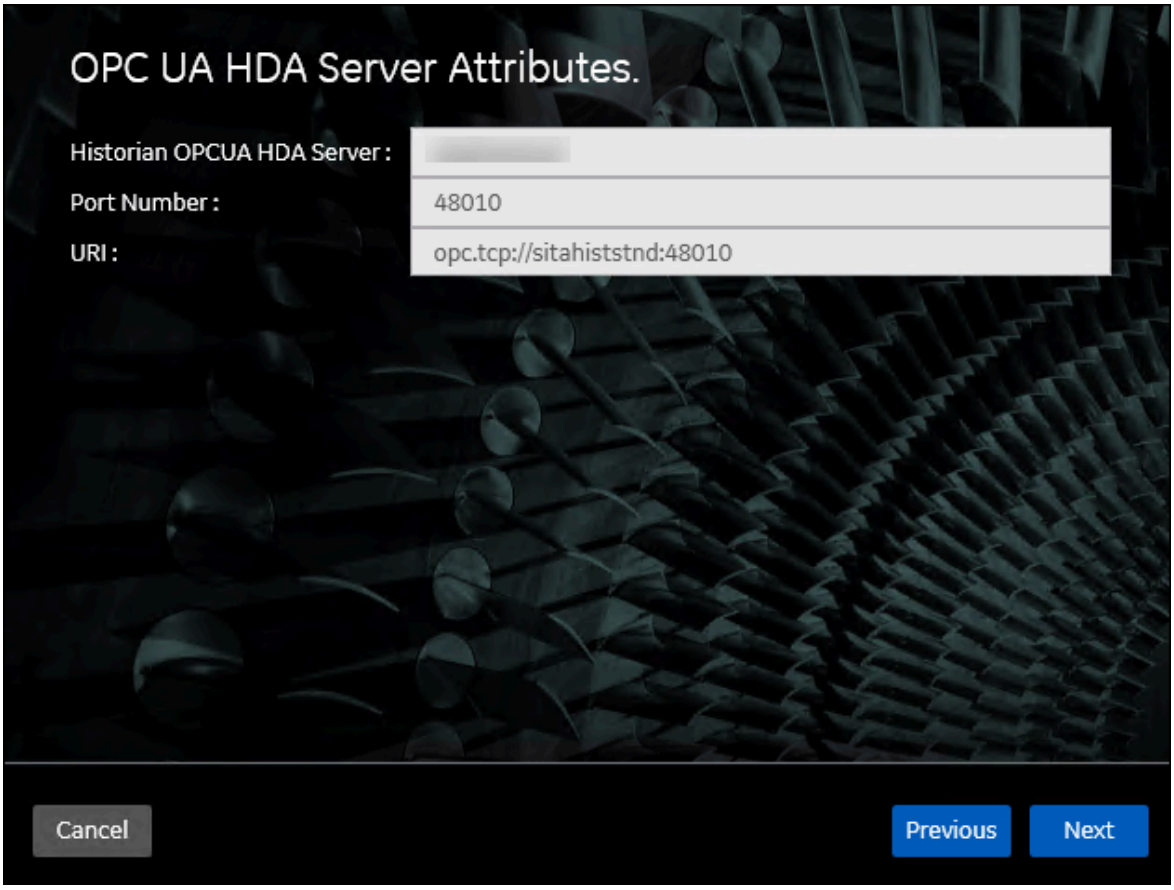
Install Historian *(on page 104)*.

**Note:**
You can install Historian and the OPC UA HDA server on the same machine or on different machines.

1. Run the Historian installer.
2. Select **Install Historian OPC UA HDA Server**.
   The welcome page appears.
3. Select **Next**.
   The license agreement appears.



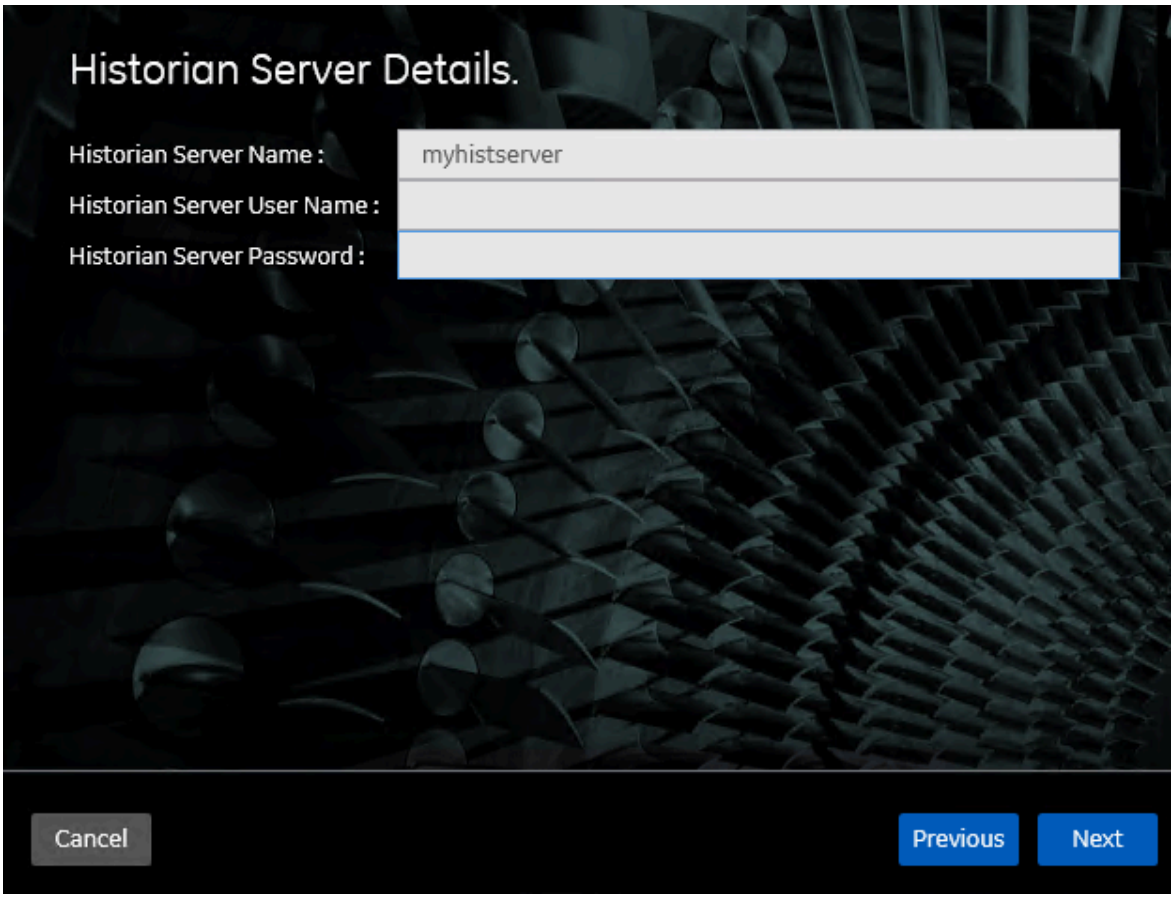

4. Select the **Accept** check box, and then select **Next**.
   The **Where to install the Historian OPC UA HDA Server** page appears, asking you to select the installation drive.
5. Select the installation drive, and then select **Next**. You can retain the default one, or choose a different one.
   The **OPC UA HDA Server Attributes** page appears.




6. Provide values as described in the following table, and then select **Next**.

| Field | Description |
|---|---|
| **Historian OPCUA HDA Server** | Enter the host name or the IP address of the machine on which you want to install the OPC UA HDA server. By default, the local host name appears. |
| **Port Number** | Enter the port number that you want the OPC UA HDA server to use. |
| **URI** | The URI to access the OPC UA HDA server. This field is disabled and populated with a value in the following format: opc.tcp://*<host name>*:*<port number>*, where *<host name>* and *<port number>* are the values that you have entered in the preceding fields. |

The **Historian Server Details** page appears.




7. Provide values as described in the following table, and then select **Next**.

| Field | Description |
|---|---|
| **Historian Server Name** | Enter the name of the Historian server that you want to connect to the OPC UA HDA server. |
| **Historian Server User Name** | Enter the username of the Historian server. |
| **Historian Server Password** | Enter the password of the Historian server. |

The **You are ready to install** page appears.

8. Select **Install**.

The Historian OPC UA HDA server is installed. Reboot the machine when prompted to do so.

- If you have installed the OPC UA HDA server on a remote machine, enable the firewall.
- Install an OPC UA client.
- Configure the OPC UA HDA server.

## The OPC UA HDA Server Workflow

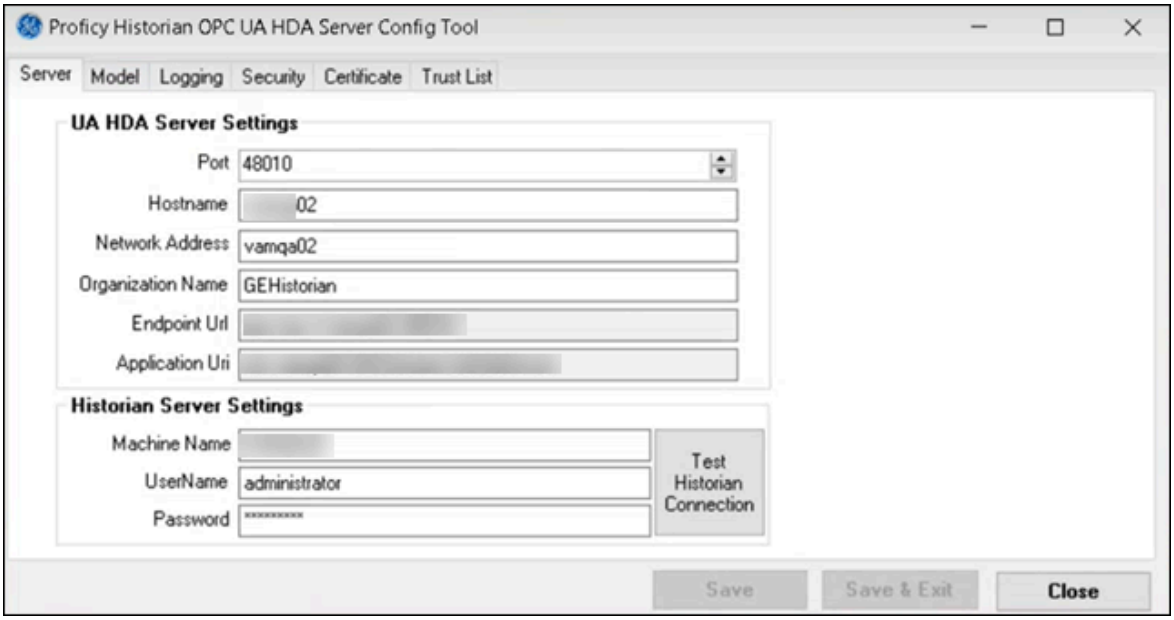
This topic provides the high-level steps to get started with the Historian OPC UA HDA server. These steps are required only for the initial setup. After you perform these steps, you can use an OPC UA HDA client to retrieve and analyze the historical data.

| Step Number | Step | Notes |
|---|---|---|
| 1 | Install Historian *(on page 104)*. | This step is required. |
| 2 | Install the OPC UA HDA Server *(on page 191)*. | This step is required. You can install Historian and the OPC UA HDA server either on the same machine or on different machines. |
| 3 | Configure the OPC UA HDA server settings *(on page 2416)*. | This step is required only if you want to change the default settings or the values you provided while installing the OPC UA HDA server. |
| 4 | Install an OPC UA HDA client. | This step is required only if you want to use a third-party OPC UA HDA client. |
| 5 | Connect the OPC UA HDA server and the OPC UA HDA client *(on page 2420)*. | This step is required only if you want to use a third-party OPC UA HDA client. In this step, the server and the client exchange certificates to establish a secure connection. |
| 6 | Authenticate the user to connect to the OPC UA HDA server *(on page 2421)*. | This step is required only if you want to use a third-party OPC UA HDA client. You can use authentication based on user credentials or a certificate. Anonymous authentication is not supported. |

## Configure the OPC UA HDA Server Settings

- Install the OPC UA HDA server *(on page 191)*.
- Install an OPC UA client that you want to use with the OPC UA HDA server.

1. Run the `ihistopcuahdaserverconfigtool.exe` file as an administrator. This file is located in the following folder: `<installation drive>:\Program Files\GE Digital\Historian OPCUA HDA Server`
   The **Proficy Historian OPC UA Server Configuration** window appears.




2. Provide values in the available fields as described in the following table.

| Field | Description |
|---|---|
| **Port** | Enter the TCP port number on which you want the OPC UA HDA server to run. By default, the value in this field is 48010. |
| **Hostname** | Enter the host name of the machine on which you have installed the OPC UA HDA server. |
| **Network Address** | Enter the DNS name or IP address of the machine on which you have installed the OPC UA HDA server. The network address must be the computer name or an IP address, as this represents how OPC clients will locate the OPC UA HDA server. |
| **Organization Name** | Enter the name of the organization that is deploying the OPC UA HDA server. |
| **Endpoint Url** | Identifies the network endpoint that the OPC clients use to communicate with the OPC UA HDA server. This field is disabled and populated with a value in the following format: opc.tcp://*<network address>*:*<port>*/. |

| Field | Description |
|---|---|
| **Application Url** | Identifies a unique identifier for the OPC UA HDA server. This field is disabled and populated with a value in the following format: urn:*<host name>*:*<organization name>*:HDAServer). |
| **Machine Name** | Enter the name of the machine on which Historian is installed. |
| **UserName** | Enter the username to connect to the Historian server. |
| **Password** | Enter the password to connect to the Historian server. |

3. Select **Model**.
   The **Model** section appears.
4. If you want to access the Historian model using an OPC UA HDA client:

   a. Select the **Yes, I want to use a model** check box.

   b. Under **Web-based Clients Configuration**, in the **Server Name** and **Port** fields, enter the host name (or IP address) and public HTTPS port number respectively of the machine on which you have installed Web-based Clients.

   c. Select **Test connection**.
   Connection to the machine on which you have installed Web-based Clients is verified, and the **Server Name** and **Port** fields under **Proficy Authentication Configuration** are populated with the corresponding values you provided while installing Web-based Clients.

   d. Verify that the server name and port number of the machine on which you have installed Proficy Authentication are correct, and select **Test connection**.

5. Select **Logging**.
   The **Logging** section appears.
6. If you want to enable logging, select the **Logging Enabled** check box, and then provide values as described in the following table.

| Field | Description | Default Value | Valid Values |
|---|---|---|---|
| **No of Log Files** | The maximum number of log file backups that you want to retain. | 5 | 1 to 100 |
| **Max Entries per Log file** | The maximum number of lines that you want in the log file before it is backed up and a new log is created. | 100000 | 1000 to 1000,000,000 |
| **Optimize Log Output** | Indicates whether the log output is buffered before it is saved to disk. | False | ◦ True<br>◦ False |
| **Application Trace Level** | The level of trace information that you want the OPC UA HDA server to log. | Error | ◦ None<br>◦ Error<br>◦ Warning<br>◦ Info<br>◦ Debug |
| **Stack Log Level** | The level of trace information that you want the OPC UA HDA stack to log. | Error | ◦ None<br>◦ Error<br>◦ Warning<br>◦ Info<br>◦ Debug |
| **Log File Path** | The path of the log file. You can provide a folder other than the default one; the log file will be created in that folder. | *<installation drive>*:\ProgramData\GEDigitalUA\logs\UaSdkCppBundleSource/HDAServer.log | |

7. Select **Security**, and provide values as described in the following table.

| Field | Description |
|---|---|
| **Allow secure communication with data privacy (SignAndEncrypt)** | If you select this option, all communication is kept private, and the OPC clients are authenticated. |
| **Allow secure communication without data privacy (SingOnly)** | If you select this option, all communication is visible, but the OPC clients are authenticated. |
| **Allow communication with no security (None)** | If you select this option, a certificate is not used for communication between OPC clients and the OPC UA HDA server. This option is not recommended; you can use it only in a non-production environment. |
| **Basic256Sha256** | This policy is acceptable and more likely to be supported by older versions of the OPC UA HDA clients. |
| **Aes128Sha256RsaOaep** | This policy is secure and is faster than the most secure policies. However, older versions of the OPC UA HDA clients do not support it. |
| **Aes256Sha256RsaPss** | This policy is the more secure one. However, older versions of the OPC UA HDA clients do not support it. |
| **Basic128Rsa15** | This policy has theoretical problems and is not recommended. |
| **Basic256** | This policy has known vulnerabilities and must not be used unless absolutely necessary. |

8. Restart the OPC UA HDA Server service.

Establish a connection between the OPC UA HDA server and the OPC UA client that you want to use *(on page 2420)*.

## Connect the OPC UA HDA Server and the OPC UA HDA Client

Configure the OPC UA HDA server settings *(on page 2416)*.

To establish a connection between the OPC UA HDA server and an OPC UA client:

1. The OPC UA HDA server sends a certificate to the OPC UA client.
2. The OPC UA client sends a certificate to the OPC UA HDA server.

This topic describes how to perform these steps.

1. Start the OPC UA HDA server. To do so, run the Historian OPC UA HDA Server service.
2. Access the OPC UA client application, and create a connection with the endpoint URL that you provided while configuring the OPC UA HDA server settings *(on page 2416)*.
3. Select the security settings that you want to use for the OPC UA client.
   The OPC UA HDA server sends the server certificate to the OPC UA client. A message appears, asking you to accept the server certificate.
4. Accept the server certificate.
   The OPC UA client sends a certificate to the OPC UA HDA server. However, an error message appears, stating that verifying the certificate has failed. This happens because, by default, the client certificate is rejected.
5. Accept the client certificate:

   a. In the **Proficy Historian OPC UA Server Configuration** widow, select **Trust List**.

   The client certificate appears. A icon appears next to the certificate, indicating that it is rejected.

   b. Select the row containing the client certificate, and then select **Trust Certificate**.
   A message appears, asking you to confirm that you want to trust the client certificate.

   c. Select **Yes**.

   A icon appears next to the certificate, indicating that it is trusted.

6. Access the OPC UA HDA client, and process the connection that you have created.
   A message appears, asking you to accept the server certificate.
7. Accept the server certificate.
   Connection between the OPC UA HDA server and the OPC UA HDA client is established.

Authenticate the user who will use the OPC UA HDA client *(on page 2421)*.

## Authenticate a User to Connect to the OPC UA HDA Server

Connect the OPC UA HDA server and the OPC UA HDA client *(on page 2420)*.

You can authenticate a user based on one of the following methods:

- **User credentials:** In this method, you will use the credentials of a Historian user. When you do so, the OPC UA HDA server validates the credentials by connecting to the Historian server, and then grants access.

  > **Note:**
  > The user is authenticated regardless of whether the Historian server is part of a stand-alone or a distributed Historian system.

- **Certificate:** In this method, you will provide a trusted certificate and a private key of the OPC UA HDA server. The server validates that the certificate is the same as the user certificate stored in the server, and then grants access.

  > **Tip:**
  > You can generate a self-signed certificate and its keys using the **Proficy Historian OPC UA Server Configuration** tool, which is provided with the OPC UA HDA server.

Anonymous authentication is not supported.

1. Access the OPC UA HDA client.
2. If you want to authenticate a user using the user credentials, select the appropriate option in the user authentication window, and then enter the username and password of the Historian user.
3. If you want to authenticate a user using a certificate:
   a. Select the appropriate option in the user authentication window.
   b. Provide the certificate and the private key (which are stored in the .der and .pem formats respectively). If needed, enter a password for the private key.

The user is authenticated. A list of Historian tags appears in the OPC UA HDA client. These tags are represented as models in the client, categorized based on the data type. The OPC UA HDA server collects tag data from Historian, and sends it to the client. You can now use the client to access a trend chart of the tag data and analyze it.

## Supported Attributes

The following table provides a list of the attributes supported by the OPC UA HDA server:

| Historian Property Type | OPC UA HDA Attributes | Data Type | Description |
|---|---|---|---|
| Engineering Units | OpcUa_Range | String | The units for the item (for example, kg/sec). |
| Engineering Units Range | OpcUa_EUInformation | Int16 | The upper and lower range of the engineering units. |
| Data Type | OPCHDA_ DATA_TYPE | Int16 | The data type for the item. For information, refer to Supported Data Types *(on page 2400)*. |
| TagDescription | OPCHDA_ DESCRIPTION | String | The description of the item. |

## Supported Data Types

The following table provides a list of data types in Historian and the corresponding ones in the OPC Classic HDA server and the OPC UA HDA server.

| Data Type in Proficy Historian | Data Type in the OPC Classic HDA server or the OPC UA HDA server. |
|---|---|
| Single Integer | VT_I2&endash; 16 bit signed integer |
| Double Integer | VT_I4- 32 bit signed integer |
| Quad Integer | VT_I8- 64 bit quad integer |
| Unsigned Single Integer | VT_UI2- 16 bit unsigned single integer |
| Unsigned Double Integer | VT_UI4- 32 bit unsigned single integer |
| Unsigned Quad Integer | VT_UI8- 62 bit quad integer |
| Byte | VT_I1 |
| Boolean | VT_BOOL |
| SingleFloat | VT_R4- 32 bit float |
| Double Float | VT_R8- 64 bit double float |
| Variable String | VT_BSTR |

| Data Type in Proficy Historian | Data Type in the OPC Classic HDA server or the OPC UA HDA server. |
|---|---|
| Fixed String | VT_BSTR |
| Date | VT_DATE |
| Blob | VT_BSTR |

## Supported Quality Values

The following table provides a list of the quality values in Historian and the corresponding ones in the OPC UA HDA server.

| Proficy Historian Quality | OPC Classic HDA Quality | Description |
|---|---|---|
| ihOPCGood | OPC_ QUALITY_ GOOD | Indicates that there is no need for inspection. This quality is returned for the tags that have all the values archived properly. |
| ihOPCBad | OPC_ QUALITY_ BAD | Indicates a need for attention. This quality is returned for the tags that had problems during the collection. |

# Troubleshooting OPC UA HDA Server Issues

### Unable to Connect to the Server

**Issue:** When you attempt to connect to the OPC UA HDA server from an OPC UA HDA client, a message appears, stating that you cannot connect to the server.

**Diagnostics:**

- Ensure that user authentication for the OPC UA HDA server is based on either user credentials or certificates. Anonymous authentication is not supported.
- If you have selected authentication based on user credentials, ensure that you have entered the correct credentials of the Historian server.

### Unable to See the Latest Historian Model Changes in the OPC UA HDA Client

**Issue:** Changes to the Historian model are not reflected in the OPC UA HDA client.

**Diagnostics:**

- Ensure that you have selected the **Yes, I want to use a model** check box and tested that the OPC UA HDA server machine is connected with the machines on which you have installed Web-based Clients and Proficy Authentication. These settings are in the **Model** section in the **Proficy Historian OPC UA Server Configuration** window. For instructions, refer to Configure the OPC UA HDA Server Settings *(on page 2416)*.
- Restart the OPC UA HDA Server service.

# Chapter 29. The OPC UA DA Collector

## Overview of the OPC UA DA Collector

The OPC UA Data Access (DA) collector gathers and collects data from a OPC UA 1.0-compliant OPC UA DA server (such as CIMPLICITY). The collector automatically determines the capability of the OPC UA DA server to which it is connected, and supports the appropriate features based on this information.

**Features:**

- You can browse the source for tags and their attributes on an OPC server that supports browsing.
- Both the polled and unsolicited data collection are supported; when changes to the OPC source tags are detected, they are sent to the Historian server. Unsolicited data collection is supported for OPC 2.0 only. The minimum poll interval is 100ms. The collector duplicates raw samples from the OPC server into the Historian data archive.

  For unsolicited data collection, if collector compression is disabled, all new values produce an exception. And, the deadband percentage is determined by the collector deadband percent. You can only configure the collector deadband percent by enabling compression.
- The supported timestamp resolution is 1ms.
- Floating point, integer, binary, and string data are supported.
- Python expression tags are supported.
- Device timestamps are accepted.

**Supported data types:**

| OPC UA DA Collector Data Type | Recommended Data Type in Historian |
|---|---|
| `OpcUaType_Null` | `ihTKVariableString` |
| `OpcUaType_Boolean` | `ihTKBool` |
| `OpcUaType_SByte` | `ihTKByte` |
| `OpcUaType_Byte` | `ihTKByte` |
| `OpcUaType_Int16` | `ihTKInteger` |
| `OpcUaType_UInt16` | `ihTKUInt16` |
| `OpcUaType_Int32` | `ihTKDoubleInteger` |
| `OpcUaType_UInt32` | `ihTKUInt32` |

| OPC UA DA Collector Data Type | Recommended Data Type in Historian |
|---|---|
| `OpcUaType_Int64` | `ihTKInt64` |
| `OpcUaType_UInt64` | `ihTKUInt64` |
| `OpcUaType_Float` | `ihTKFloat` |
| `OpcUaType_Double` | `ihTKDoubleFloat` |
| `OpcUaType_DateTime` | `ihTKVariableString` |
| `OpcUaType_Guid` | `ihTKDataTypeUndefined` |
| `OpcUaType_StatusCode` | `ihTKDataTypeUndefined` |
| `OpcUaType_String` | `ihTKVariableString` |
| `OpcUaType_ByteString` | `ihTKDataTypeUndefined` |
| `OpcUaType_XmlElement` | `ihTKDataTypeUndefined` |
| `OpcUaType_NodeId` | `ihTKDataTypeUndefined` |
| `OpcUaType_ExpandedNodeID` | `ihTKDataTypeUndefined` |
| `OpcUaType_DiagnosticInfo` | `ihTKDataTypeUndefined` |
| `OpcUaType_QualifiedName` | `ihTKDataTypeUndefined` |
| `OpcUaType_LocalizedText` | `ihTKDataTypeUndefined` |
| `OpcUaType_ExtensionObject` | `ihTKDataTypeUndefined` |
| `OpcUaType_DataValue` | `ihTKDataTypeUndefined` |

**Supported tag attributes:**

- Tagname
- Source Address
- Engineering Unit Description
- Data Type
- Hi Engineering Units
- Lo Engineering Units
- Is Array Tag

The Engineering Unit Description, Hi Engineering Units and Lo Engineering Units vary based on the OPC server vendor.

**Note:**
While some of these attributes are queried on a browse, they are not shown in the browse interface. These attributes are used when adding a tag, but it is not visible to you if all attributes come from the server or not.

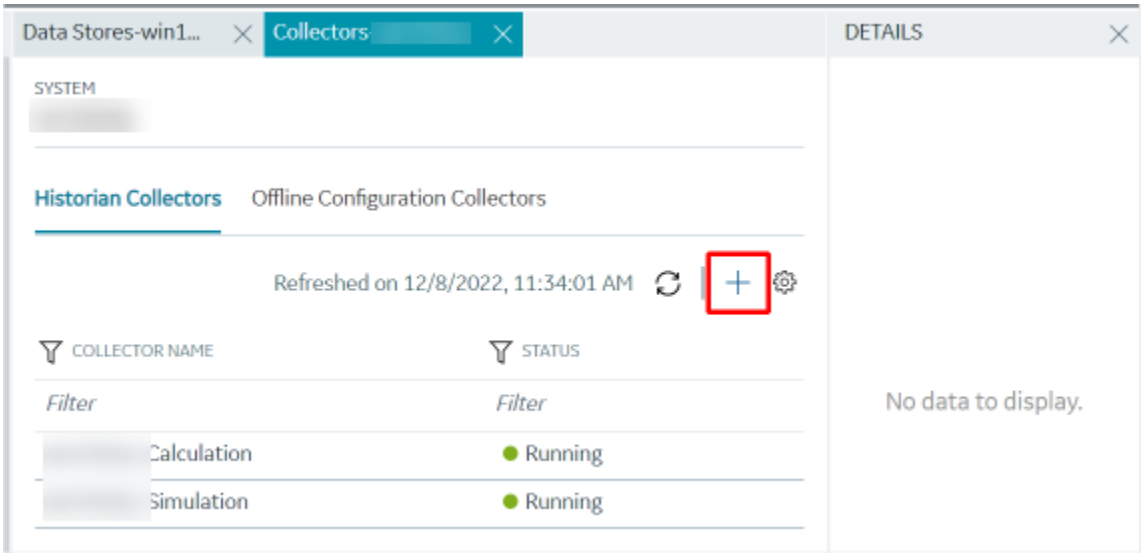
# Configuration

## Add and Configure an OPC UA Data Access Collector

The OPC UA Data Access (DA) collector gathers and collects data from a OPC UA 1.0-compliant OPC UA DA server. For more information, refer to Configure an OPC UA DA Collector Using Historian Administrator *(on page 2432)*.

This topic describes how to add a collector instance using Configuration Hub. You can also add a collector instance using the RemoteCollectorConfigurator utility *(on page 797)*, which does not require you to install Web-based Clients.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**. A list of collectors in the default system appears.
3. In the upper-right corner of the main section, select +.




The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.

4. In the **MACHINE NAME** field, select the machine in which you want to add a collector instance.
5. In the **COLLECTOR TYPE** field, select **OPC UA DA Collector**, and then select **Get Details**.
   The **INSTALLATION DRIVE** and **DATA DIRECTORY** fields are disabled and populated.
6. Select **Next**.
   The **Source Configuration** section appears.
7. In the **OPC UA SERVER URI** field, enter the URI to connect to the OPC server in the following format:
   `opc.tcp://<host name or IP address of the OPC UA server>:<port number>`
8. Select **Next**.
   The **Destination Configuration** section appears. The collector machine name provided by you is selected as the **Source Configuration** by default.

   Under **CHOOSE DESTINATION**, the **Historian Server** option is selected by default. In addition, the **DESTINATION HISTORIAN SERVER** field is disabled and populated with the collector machine name.
9. Select the destination to which you want to send data, and then enter the values in the corresponding fields. You can send data to an on-premises Historian server or to a cloud destination.

   a. If you need to send data to a cloud destination, select the cloud destinations as needed.
      - **Predix Timeseries**- Select this if you need to send data to Predix cloud. For more information, refer to Predix Cloud *(on page 621)*.
      - **Azure IoT Hub**- Select this if you need to send data to Azure Cloud in KairosDB format. For more information, refer to Azure IoT Hub (KairosDB format) *(on page 608)*.
      - **MQTT**- Select this if you need to send data to any of the following cloud destination.
        - Alibaba cloud. For more information, refer to Alibaba Cloud *(on page 587)*.
        - AWS cloud. For more information, refer to AWS Cloud *(on page 594)*.
        - Google cloud. For more information, refer to Google Cloud *(on page 614)*.

   b. If you need to send data to an on-premises Historian server, select **Historian Server**.
      If you created security groups or enabled a strict client/collector authentication, enter the **USERNAME** and **PASSWORD** of the on-premises Historian server that you created during the installation of the collector.
      If you entered the **USERNAME**and **PASSWORD**, select **Test Connection**. This will help you to test if the Historian server that you are trying to connect is valid or if the credentials that you entered are valid.
      If the entered credentials are valid, a successful connection message appears.
10. Select **Next**.

The **Collector Initiation** section appears. The **COLLECTOR NAME** field is populated with a value in the following format: `<Historian server name>_OPCUACollector_<number>`

11. If needed, modify the value in the **COLLECTOR NAME** field.
    The value that you enter:
    - Must be unique.
    - Must contain the string `OPCUACollector`.
    - Must not exceed 15 characters.
    - Must not contain a space.
    - Must not contain special characters except a hyphen, period, and an underscore.
12. In the **RUNNING MODE** field, select one of the following options.
    - **Service - Local System Account**: Select this option if you want to run the collector as a Windows service using the credentials of the local user (that is, the currently logged-in user). If you select this option, the **USERNAME** and **PASSWORD** fields are disabled.
    - **Service Under Specific User Account**: Select this option if you want to run the collector as a Windows service using a specific user account. If you select this option, you must enter values in the **USERNAME** and **PASSWORD** fields.

      If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:
      - iH Security Admins
      - iH Collector Admins
      - iH Tag Admins

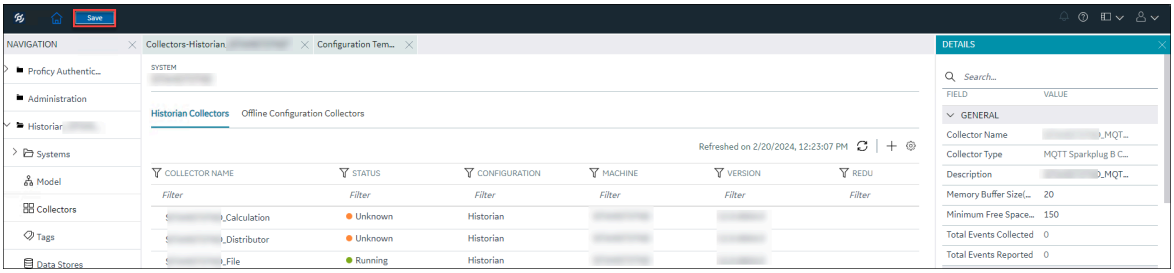
    You can also configure the collector to start automatically when you start the computer.
13. Select **Add**.
    The collector instance is added. The fields specific to the collector section appear in the **DETAILS** section.
14. In the **COLLECTOR SPECIFIC CONFIGURATION** section, configure values as described in the following table.

| Field | Description |
|---|---|
| **OPC UA Server URL** | The URI to connect to the OPC UA server. Enter a value in the following format: `opc.tcp://<host name or IP address of the OPC UA server>:<port number>` |

| Field | Description |
|---|---|
| **Secured Connectivity** | Indicates whether you want a secured connection between the OPC UA server and the collector. By default, this field is set to false.<br><br>You can establish a secured connectivity in one of the following ways:<br>◦ **Using certificates:** To use certificates, switch off the **User Security** toggle.<br>◦ **Using user authentication:** To use user authentication, switch on the **User Security** toggle. |
| **User Security** | This field is enabled only if you have enabled secured connectivity. Switch on this toggle if you want to use user authentication to connect to the OPC server. When you do so, the **User Name** and **Password** fields are enabled. You can either enter the user credentials in these fields, or you can use the values in the `ClientConfig.ini` file. For instructions, refer to Connect with the OPC UA DA Server Securely *(on page 2434)*. |
| **Username** | This field is enabled only if you have set the secured connectivity to true and switched on the **User Security** toggle. Enter the username that you want to use to connect to the OPC server. If you do not provide a value, the username from the `ClientConfig.ini` file is considered. |
| **Password** | This field id enabled only if you have set the secured connectivity to true and selected the **Enable User Security** check box. Enter the password that you want to use to connect to the OPC server. If you do not provide a value, the password from the `ClientConfig.ini` file is considered. |
| **MTLS Security** | Indicates whether you want to use Mutual TLS (MTLS) protocol to enforce a secure and strong authentication mechanism. |
| **MTLS Data Encryption** | Indicates whether you want to encrypt the data that the collector shares to the data archiver (DA). |

| Field | Description |
| --- | --- |
| For more information on how to enable MTLS Security, refer to Enable MTLS Security *(on page 632)*. | |

15. As needed, enter values in the other sections common to all collectors *(on page 579)*.
16. In the upper-left corner of the page, select **Save**.




The changes to the collector instance are saved.

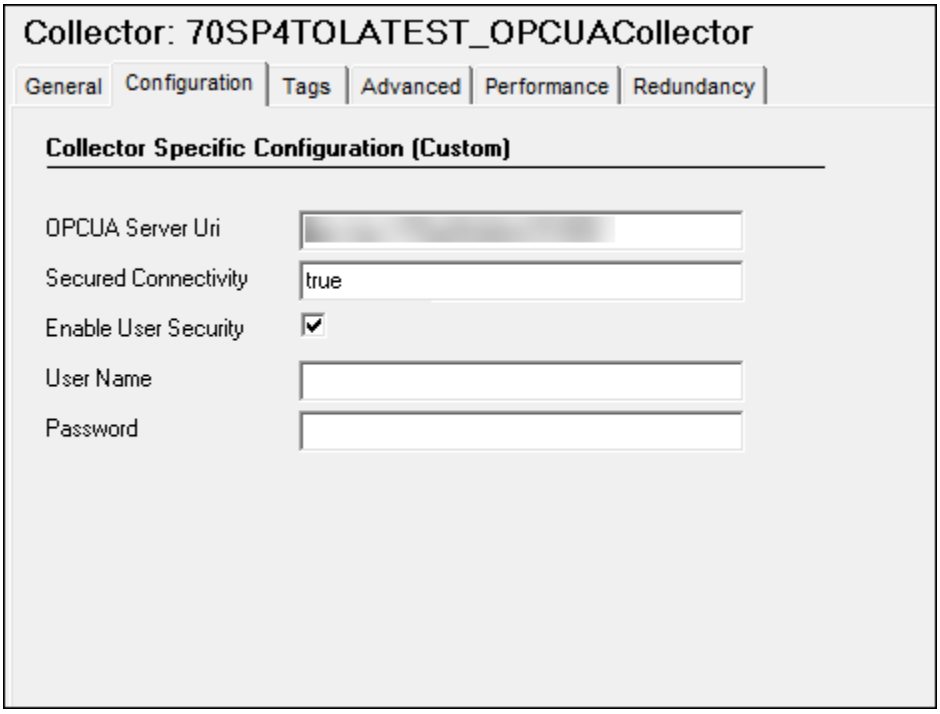
17. If needed, restart the collector.

If you have enabled secured connection, establish a secured connection between the OPC server and the collector *(on page 2434)*.

## Configure an OPC UA DA Collector Using Historian Administrator

1. Access Historian Administrator.
2. Select **Collectors**, and then select the OPC UA DA collector instance that you want to configure.
3. Select **Configuration**.

   The **Configuration** section appears.




4. Enter values as specified in the following table.

| Field | Description |
|---|---|
| OPCUA Server URI | The Unified Resource Identifier (URI) of the OPC UA DA server from which you want to collect data. Enter a value in the following format: `opc:tcp://<host name of the OPC server>:<port number>`. By default, the local host is considered.<br><br>**Note:**<br>The URI is a superset of the Uniform Resource Locator (URL). |
| Secured Connectivity | Indicates whether you want a secured connection between the OPC server and the collector. By default, this field is set to false.<br><br>You can establish a secured connectivity in one of the following ways:<br>◦ **Using certificates:** To use certificates, clear the **Enable User Security** check box.<br>◦ **Using user authentication:** To use user authentication, select the **Enable User Security** check box. |
| Enable User | This field is enabled only if you have enabled secured connectivity. Select this check box if you want to use user authentication to connect to the OPC server. When you do so, the **User Name** and **Password** fields are enabled. You can either enter the user credentials in these fields, or you can use the values in the `ClientConfig.ini` file. |

| Field | Description |
|---|---|
| Security | |
| User Name | This field appears only if you have enabled secured connectivity and selected the **Enable User Security** check box. Enter the username that you want to use to connect to the OPC server. If you do not provide a value, the username from the `ClientConfig.ini` file is considered. |
| Password | This field appears only if you have enabled secured connectivity and selected the **Enable User Security** check box. Enter the password that you want to use to connect to the OPC server. If you do not provide a value, the password from the `ClientConfig.ini` file is considered. |

5. Select **Update**.
6. Restart the collector.
   The collector is configured.

If you have enabled secured connection, establish a secured connection between the OPC server and the collector *(on page 2434)*.

## Add a Client Certificate to the Trusted List

If you have enabled secured connectivity between the OPC UA DA server and the collector, you must add a client certificate to the OPC UA DA server's trusted certificates list.

While configuring the collector settings *(on page 2432)*, ensure that the **Secured Connectivity** field is set to true.

1. Start the OPC UA DA server in a secured mode.
2. Access the installation folder of the server. Normally, it is inside the `ProgramFiles` folder.
3. In the `Rejected` folder, copy the client certificate, and paste it into the trusted certificates folder.
   To locate the trusted certificates folder, refer to your OPC UA DA server documentation.
4. Restart the collector.

## Connect with the OPC UA DA Server Securely

This topic describes how to establish a secured connection between your OPC UA DA server and the collector.

All the security related configuration for OPC UA collector to establish secured connectivity to OPC UA server will be done by using ClientConfig.ini file. This file is located in C:\Program Files\GE Digital \Historian. The OPC UA DA Collector\Server64 ClientConfig.ini file has options to select Trust Certificate type, Security Policy, Security Mode, Username and Password. There are default values provided, however these can be configured accordingly.

1. Access the `ClientConfig.ini` file. By default, it is located at `<Installation Drive>:\Program Files\GE Digital\Historian OPC UA DA Collector\Server64`.
2. Enter values as specified in the following table.

| Parameter | Description |
|---|---|
| ApplicationName | Enter `OPCUACollector`. |
| TrustCertificate | Enter one of the following values:<br>◦ `0`: Enter this value if want no trust.<br>◦ `1`: Enter this value if you want to trust temporarily.<br>◦ 2: Enter this value if you want to trust permanently. If you enter this value, you must copy the server certificate in the trusted certificate list of the collector. |
| SecurityPolicy | The security policy that you want to use. A value is required only if the value for theTrustCertificate parameter is 2. Enter one of the following values:<br>◦ `0`: Does not use a security policy.<br>◦ `1`: Uses the Basic128Rsa15 policy. This policy has theoretical problems and is not recommended.<br>◦ `2`: Uses the Basic 256 policy. This policy has known vulnerabilities and must not be used unless absolutely necessary.<br>◦ `3`: Uses the Aes256Sha256RsaPss policy. This policy is the more secure one. However, older versions of the OPC UA HDA clients do not support it.<br>◦ `4`: Uses the Aes128Sha256RsaOaep policy. This policy is secure and is faster |

| Parameter | Description |
|---|---|
| | than the most secure policies. However, older versions of the OPC UA HDA clients do not support it.<br>◦ 5: Uses the Basic256Sha256 policy. This policy is acceptable and more likely to be supported by older versions of the OPC UA HDA clients. |
| SecurityMode | The security mode that you want to use. Enter one of the following values:<br>◦ 0: Enter this value if you want to allow communication without security. If you select this option, a certificate is not used for communication between the server and the collector. This option is not recommended; you can use it only in a non-production environment.<br>◦ 1: Enter this value if you want to allow secure communication without data privacy. If you select this option, all communication is visible, but the collector is authenticated.<br>◦ 2: Enter this value if you want to allow secure communication with data privacy. If you select this option, all communication is kept private, and the collector is authenticated. |
| CertificateTrustListLocation | Enter the path to the trusted certificates folder in the OPC server. |
| CertificateRevocationListLocation | Enter the path to the revoked certificates folder in the OPC server. |
| IssuersCertificatesLocation | Enter the path to the issuer certificates folder in the OPC server. |
| IssuersRevocationListLocation | Enter the path to the |

| Parameter | Description |
|---|---|
| ClientCertificate | |
| ClientPrivateKey | |
| RetryInitialConnect | Enter `true` or `false` to specify whether to reconnect to the OPC server automatically if the collector fails to connect to the server initially. |
| AutomaticReconnect | Enter `true` or `false` to specify whether to reconnect to the OPC server automatically if the collector fails to connect to the server subsequently (after the initial connection). |
| Username | Enter the username that you want to use to connect to the server. |
| Password | Enter the password that you want to use to connect to the server. |

### Sample `ClientConfig.ini` File

```
ApplicationName = OPCUACollector

TrustCertificate = 2

SecurityPolicy = 4

SecurityMode = 1

CertificateTrustListLocation =/[ApplicationPath]/pkiclient/trusted/certs/

CertificateRevocationListLocation =/[ApplicationPath]/pkiclient/trusted/crl/

IssuersCertificatesLocation =/[ApplicationPath]/pkiclient/issuers/certs/

IssuersRevocationListLocation =/[ApplicationPath]/pkiclient/issuers/crl/

ClientCertificate =/[ApplicationPath]/pkiclient/own/certs/uaclientcpp.der

ClientPrivateKey =/[ApplicationPath]/pkiclient/own/private/uaclientcpp.pem

RetryInitialConnect               =true

AutomaticReconnect                =true

Username =admin

Password =admin
```

# Working with the Collector

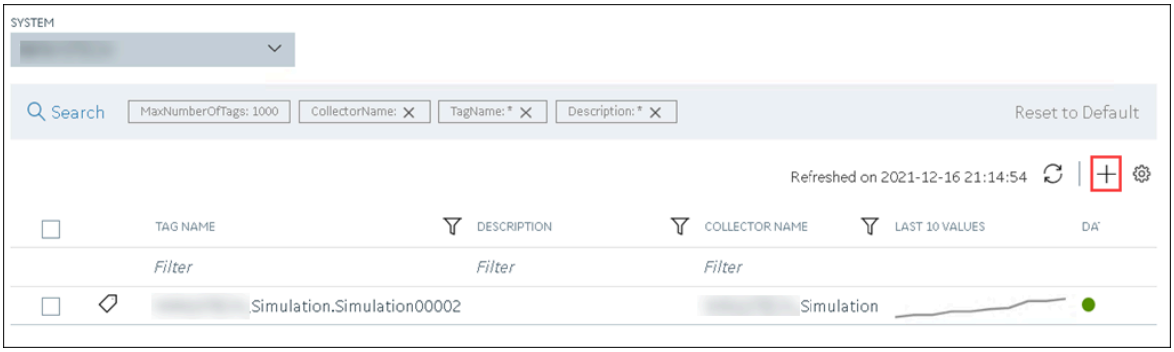
## Add Tags for the Data Store Using Configuration Hub

- Add the collector instance *(on page 357)* using which you want to collect data. Ensure that the collector is running.
- By default, the tag data is stored in the user data store, which is created automatically when you set up Configuration Hub. If, however, you want to store the data in a different data store, create it *(on page 370)*.

This topic describes how to specify the tags for which you want to collect data by browsing through the tags in the data source. For example, for an iFIX collector, if there are 1,00,000 tags in the iFIX server, you must specify the ones for which you want to collect data. Only then data is collected for those tags.

In addition to adding tags from the data source, you can create tags manually *(on page 473)*.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
3. Select +.

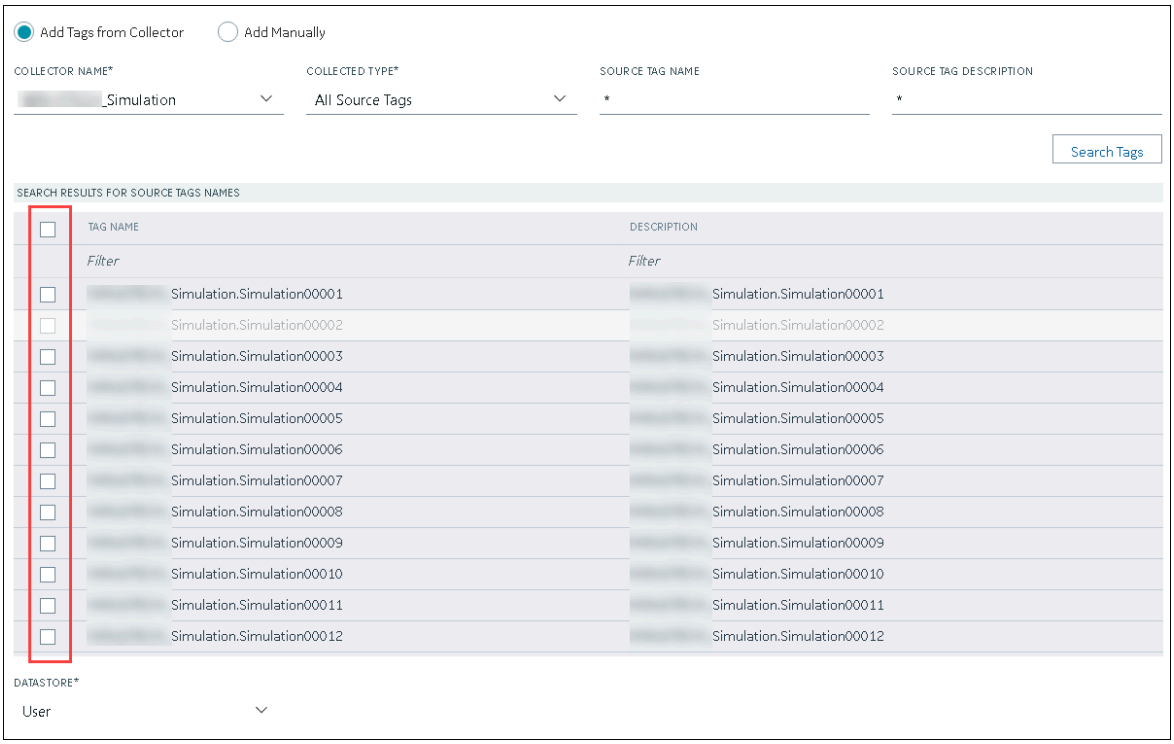



   The **Add Tag-<system name>** page appears. The **Add Tags from Collector** option is selected by default.
4. Enter values as described in the following table.

| Field | Description |
|---|---|
| **COLLECTOR NAME** | Select the collector instance that you want to use to collect data. A value is required. |
| **COLLECTED TYPE** | Specify whether you want to browse through all the tags in the data source or only from the tags that you have not added yet. A value is required. |

| Field | Description |
|---|---|
| **SOURCE TAG NAME** | Enter the name of the tag (either completely or partially) to narrow down the search results. |
| **SOURCE TAG DESCRIPTION** | Enter the description of the tag (either completely or partially) to narrow down the search results. |

5. Select **Search Tags**.
   A list of tags that match *all* the criteria that you have specified appears. If a tag is already added, it is disabled.
6. Select the check box corresponding to each tag for which you want to collect data.

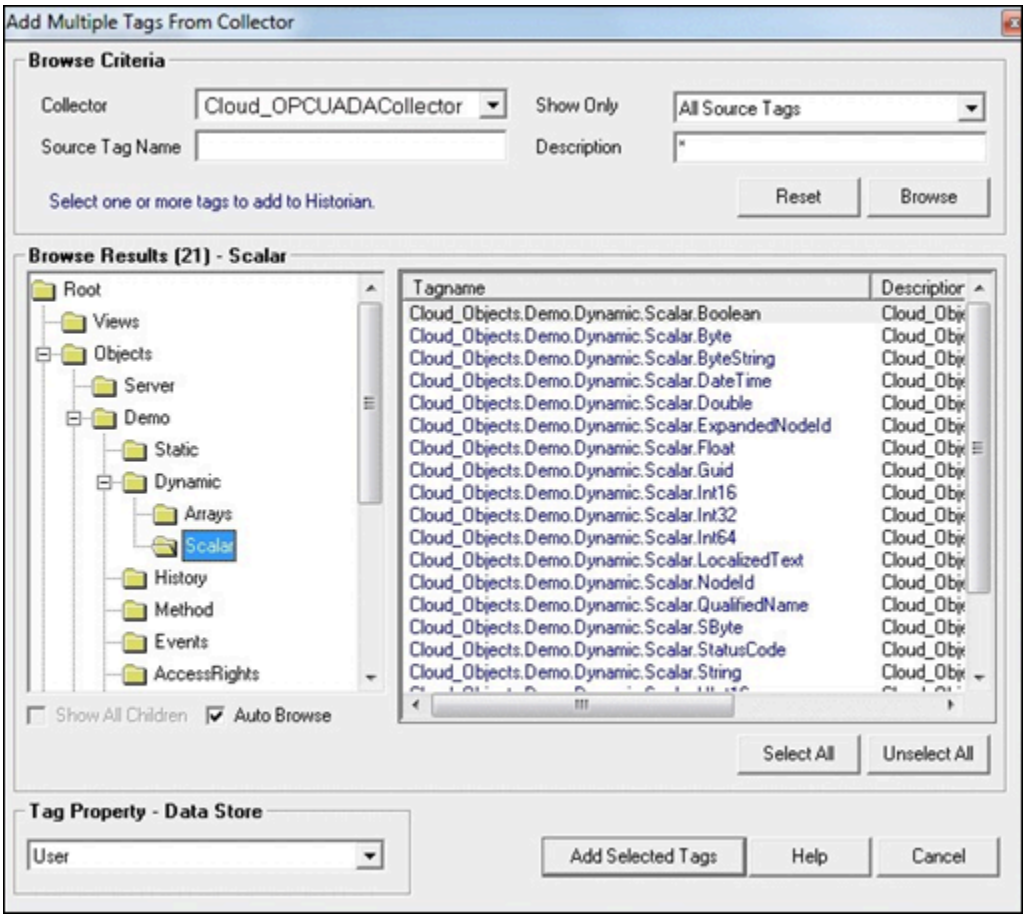



7. In the **DATA STORE** field, if you want to store the data in a different data store than the user data store, select the same.
8. Select **Add Tag**.
   Data collection begins for the selected tags.

As needed, configure each tag by providing values for the tag properties. For information on the delta query modes, refer to Counter Delta Queries.

## Specify the Tags for Data Collection

If your OPC server supports hierarchical organization of tags in a tree structure, you can use the hierarchy to browse for tags and add them to the collector for data collection.




1. Access Historian Administrator.
2. Select **Collectors**, and then select the OPC UA DA collector instance to which you want to add tags. A hierarchical view of tags appears in the **Browse Results** section.
3. If you want to view only the tags for which data is not collected, in the **Show Only** field, select **Source Tags Not Collected**. You can search for a tag by entering search criteria in the **Source Tag Name** or **Description** field.
4. If you want to search by a tag name or description, enter the value in the **Source Tag Name** or **Description** field.
5. Navigate to the node in the tree you want to browse, and then select **Browse**.

**Tip:**

- To browse automatically, select the **Auto Browse** check box. The available tags appear in the **Browse Results** window whenever a node is selected in the tree.
- To show all child elements within a hierarchy, select the **Show All Children** check box. All tags at or below the hierarchical level of the selected node in the tree appear in the **Browse Results** window.

The tags within the selected portion of the OPC server tag hierarchy appear.

- Some OPC servers do not support data blocks with a length greater than 1. These servers display only the first item in an array instead of showing all of them. For example, an OPC server may contain 3000 analog values from datablock:1 to datablock:3000, but only datablock:1 is displayed.
- If you want to archive data from poll records of a length greater than 1, we recommend that you use the Excel Add-In for Historian to configure a large block of tags (including the missing items), and then add the tags.
- If you are unable to browse items containing a forward slash ( / ) in your OPC server, you may have to change the default separator in the collector configuration. To do so, modify the registry key `HKEY_LOCAL_MACHINE\SOFTWARE\Intellution, Inc\iHistorian\Services\OPCCollector\<collector interface name>\OPCBrowseTreeSep key`, and change the string value to a character not available in your OPC server item IDs. Typical values include |, !, or &. Create this key if it does not exist.
- If you are cannot browse readable items in your OPC server, you may need to change the browse access mask used by the collector. To do so, modify the registry key `[HKEY_LOCAL_MACHINE\SOFTWARE\Intellution, Inc.\iHistorian\Services\OPCCollector\<collector interface name>`, and add the DWORD key `"OPCBrowseAccessRightsMask"=dword:00000003`. Valid values are 0, 1, 2, 3 with 1 being the default. Use 0 or 3 if you are unable to browse readable items. Creating or changing the value takes effect on the next browse attempt and does not require a collector restart.
- Some items such as unsupported data types and user-defined items in simulation servers may not be returned when you browse for tags. However, sometimes, even the items that do not appear in the search results can be added using the **Add Tag Manually** button.

6. Select the tags for which you want to collect data, and then select **Add Selected Tags**. Collected tags will appear in black in the tag list.
   The tags are added to the collector. They appear in black text in the list of tags.

# About OPC UA DA Collector Groups

It is a best practice to limit the number of OPC UA DA collector groups created by the Historian system to increase performance. To limit the number of OPC UA DA collector groups created on the OPC UA DA server, group Historian tags collected by the OPC UA DA collector using the fewest number of collection intervals possible.

# Troubleshooting the Collector

The OPC UA DA collector generates log files during initialization, configuration, and general operation. By default, you can find them at `C:\Proficy Historian Data\LogFiles`.

### Troubleshooting Tips

If the collector does not connect to the OPC server, or if tags are not displayed:

- Ensure that the certificate is added to Trusted list *(on page 2434)*.
- Ensure that you have provided a valid username and password *(on page 2432)*.
- Restart the collector whenever there is any change made to the configuration using Historian Administrator or in the `ClientConfig.ini` file.
- Check that secured Connectivity is true and Enable User security is checked to have connection with User Authentication.
- Ensure that the OPC server supports the security policy and the security mode if you see the following error message in the log file: Matching of secure endpoint not available between server and collector.
- Ensure that the `RetryInitialConnect` and `AutomaticReconnect` parameters are set to true in the `ClientConfig.ini` file.

# Chapter 30. OSI PI Collector

## Overview of the OSI PI Collector

The OSI PI collector collects data samples from an OSI PI data server and stores it in the Historian Server or a cloud destination. You can collect data directly from the OSI PI Data Archive v3.2 or later via OSI PI AOSI PI v1.3.4 or later.

**Topology:** This collector supports a distributed model, where the PI Data Server, the collector, and Proficy Historian are installed on different machines. The OSI PI collector must be installed on the same machine as the OSI PI Data Archive.

The OSI PI collector uses unsolicited collection, whereby changes to the OSI PI archives are detected, and are forwarded to the Historian server. The collector is intended to duplicate raw samples from the OSI PI Data Archive in an Historian data archive. You can specifically request the collector to transfer values from the OSI PI snapshot cache (as seen in the previous version of OSI PI Collectors), however, it is recommended to transfer the values directly from the PI archives to the Historian archives.

One OSI PI collector instance can collect data from a single OSI PI data archiver. To collect from multiple OSI PI data archives to an Historian archive, you must configure multiple OSI PI collector instances.

**Features**

- You can browse the source for tags and their attributes. Tag browsing performance with OSI PI has been confirmed as satisfactory up to 130,000 tags. Beyond that threshold, OSI PI may take a long time to return the large number of tags. In such a case, it is recommended that the tags be exported from PI to an Excel work sheet and then uploaded to Historian.
- Only the unsolicited data collection is supported; polled collection is not supported.
- The supported timestamp resolution is milliseconds or seconds.
- The collector accepts device timestamps.
- You can create Python Expression Tags for those collectors that support them.
- Floating point, integer, string, and enumerated data are supported; Binary and array data is not supported. You can configure the OSI PI collector to automatically handle updates of digital states in Historian as enumerated sets without restarting the OSI PI collector. For instructions, refer to Configuring Auto-synchronization of Digital States *(on page 2453)*.

  If digital states are renamed or deleted in the OSI PI collector, the corresponding enumerated sets in Historian are not automatically renamed or deleted without restarting the PI collector. Unused enumerated sets cannot be automatically deleted, because the PI server does not notify

the collector about the rename or delete activity. Therefore, you must manually rename *(on page 2453)*/delete *(on page 2454)* digital states.

> **Note:**
> In some instances, OSI PI digital tags, which are created as enumerated tags in Historian, can contain values from the PI System digital set, rather than their assigned digital set. In these instances, the System digital values will be reflected as 0 with BAD quality in Historian.

# Before You Begin

**Software Requirements**

If you are using Historian 7.0 SP4, the following configuration is required:

- OSI PI Data Archive (version 3.2\3.3\3.4)
- OSI PI v1.3.4 or greater
- OSI PI SDK (About PI-SDK) v1.4.2 or greater

> **Note:**
> The OSI PI SDK is required for running OSI PI Collector, however, the OSI PI SDK does not ship with Historian. If the OSI PI SDK is not installed, the OSI PI Collector will not start. If you install the OSI PI Collector on a machine that does not contain your PI Server, be sure to install the OSI PI SDK on the machine with the OSI PI Collector.

- Historian 3.0 or greater

If you are using 7.0 SP5, the following configuration is required:

- OSI PI AF Server version 2015 R2 SP1 or greater
- OSI PI Data Archiver v 3.4.380 or greater
- OSI PI AF SDK 2.7.0 or greater

> **Note:**
> The OSI PI AF SDK is required for running OSI PI Collector or PI Distributor, however, the OSI PI AF SDK does not ship with Historian. If the OSI PI AF SDK is not installed, the OSI PI Collector or PI Distributor will not start. If you install the OSI PI Collector or PI Distributor on a machine that does not contain your PI Server, be sure to install the OSI PI AF SDK on

the machine with the OSI PI Collector and the PI Distributor. PI SDK is no longer supported with the 7.0 SP5 version of the PI Collector. If you are using the OSI PI Collector or PI Distributor from the 7.0 SP5 (or greater) installer, the PI AF SDK or greater is now required.

The PI AF Software Development Kit (PI AF SDK or AFSDK) is installed with the PI AF Client and provides programmatic access to PI Server data (PI Data Archive and AF).

- About-PI SDK utility

**Hardware Requirements**

There are no additional hardware requirements for the OSI PI Collector or PI Distributor.

**Configuring the OSI PI Data Archive**

By default, no specific configuration is required for the OSI PI Data Archive to allow the OSI PI Collector to collect data and archive it to the Historian Server or Predix Cloud. However, a user with read permissions must be configured in the OSI PI data server for the OSI PI Collector.

**Note:**
The OSI PI Collector reads data directly from the OSI PI archives and attempts to maintain a near real-time operation. It has been tested up to 25,000 events per second. However, performance is dependent on hardware and network capabilities, so if the collector begins to fall significantly behind real-time, it may be more suitable to partition the data retrieval into two or more collectors.

**Upgrading and Using the PI Snapshot Collection**

In some scenarios you may wish to run the collector in the same way as in the previous versions (Pre-6.0) of PI Collector. For example, you may upgrade from an older version of the collector and still want to retain the existing, original behavior.

If this is not the case then you do not need to configure anything and may skip this section.

Use the following steps where you want to retain the existing original behavior.

To configure the newly added OSI PI Collector to retrieve data from the OSI PI snapshot cache:

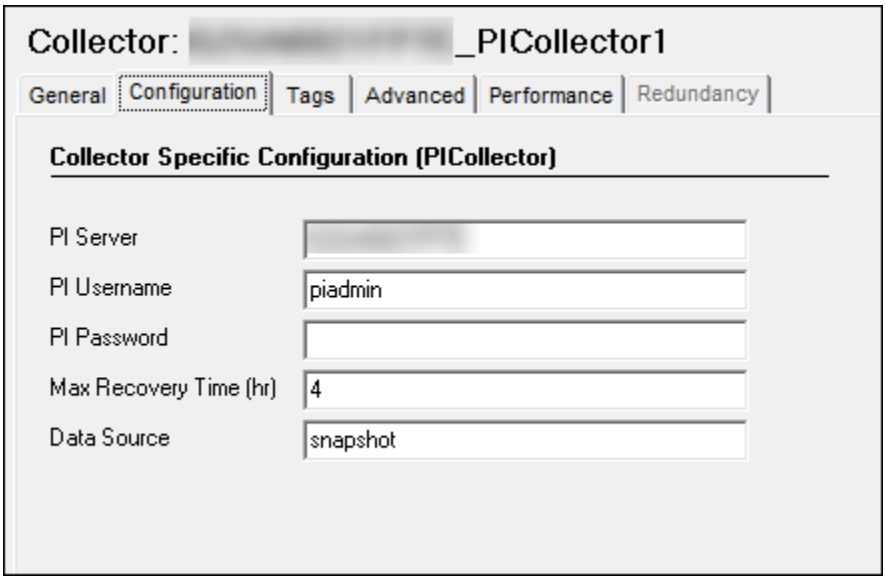
1. Run the Registry Editor (`regedit.exe`).
2. Navigate to `HKEY_LOCAL_MACHINE\SOFTWARE\Intellution, Inc.\iHistorian\Services\PICollector`.
3. In the case of a 64-bit system this may be found at `HKEY_LOCAL_ MACHINE\SOFTWARE\Wow6432Node \Intellution, Inc.\iHistorian\Services\PICollector`

4. If you have multiple collectors, each would have to be configured to allow for snapshot operation, and the key would be the name of the other collector.
5. Locate the registry key name `General5`.
6. Change it from `Archive` to `Snapshot`.
7. Restart the Historian PI Collector service from the **Windows services** control panel.

   The latest log file will indicate that the collector is using the snapshot database for collection. Refer to OSI PI Collector Troubleshooting *(on page 2455)*, for information about tracing operational issues with the OSI PI Collector log files.

**To configure existing OSI PI Collector to retrieve data from the OSI PI snapshot cache**

1. Access Historian Administrator.
2. Select the **Collectors** page.
3. Select the **OSI PI Collector**.
4. Select **Configuration**.
5. In the **Data Source** field, enter the data source as snapshot.




# OSI PI Collector Configuration

This section describes details about how to configure an OSI PI Collector.

## Configuring the OSI PI Collector

1. Start Historian Administrator.
2. Select the **Collectors** page.

3. Select the OSI PI Collector instance you wish to configure.
4. If the collector is not listed in the available collectors list, you must start the collectors manually from the Windows Service Control Panel to register it.

    a. Open the Windows services control panel on the machine that contains the OSI PI collector

    b. Find the service name Historian OSI PI Collector.

    c. Select it and select **Start** on the top of the **Services** panel.
    As it is not configured it may start and then promptly stop- this is not an issue. The current step is solely to register it in Historian.

    d. Continue from Step 1 if is not listed in Historian Administrator once restarted, refer to OSI PI Collector Troubleshooting *(on page 2455)*.

5. Configure the OSI PI Collector's **General** options.
    a. (Optional) Enter a description for your OSI PI Collector.
    b. (Optional) In the **Computer Name** field, enter the host name of the computer your collector is running on.
    c. (Optional) If you want to change the default settings for memory usage and free disk space, change the values in the **Memory Buffer Size** and **Minimum Free Space** fields.
6. Configure the OSI PI Collector-specific options.

    a. Select **Configuration**.

    b. In the **PI Server** field, enter the OSI PI server name. For example, `localhost`.

    c. In the `PI User name` field, enter the OSI PI user name. For example, `PiAdmin`.

    d. If required, enter the OSI PI password into the **PI Password** field.
    This field can be left blank if no password is assigned for the OSI PI user.

    > **Note:**
    > If you change the PI Server name, user name or password fields, you must restart the collector service before the new configuration will take effect.

    > **Note:**
    > If the username and password are provided with the PI Collector, it uses explicit login to connect to the PI Server. If the username and password are not provided,

> then we use the implicit login functionality of PI SDK. In this scenario, either it would look for PI Trust or PI Mapping to connect to the PI Server.

e. In the **Max Recovery Time (hr)** field, enter a new Maximum Recovery Time. By default, the maximum recovery time is 4 hours.

7. Configure the default collection options.

   a. Select **Tags**.

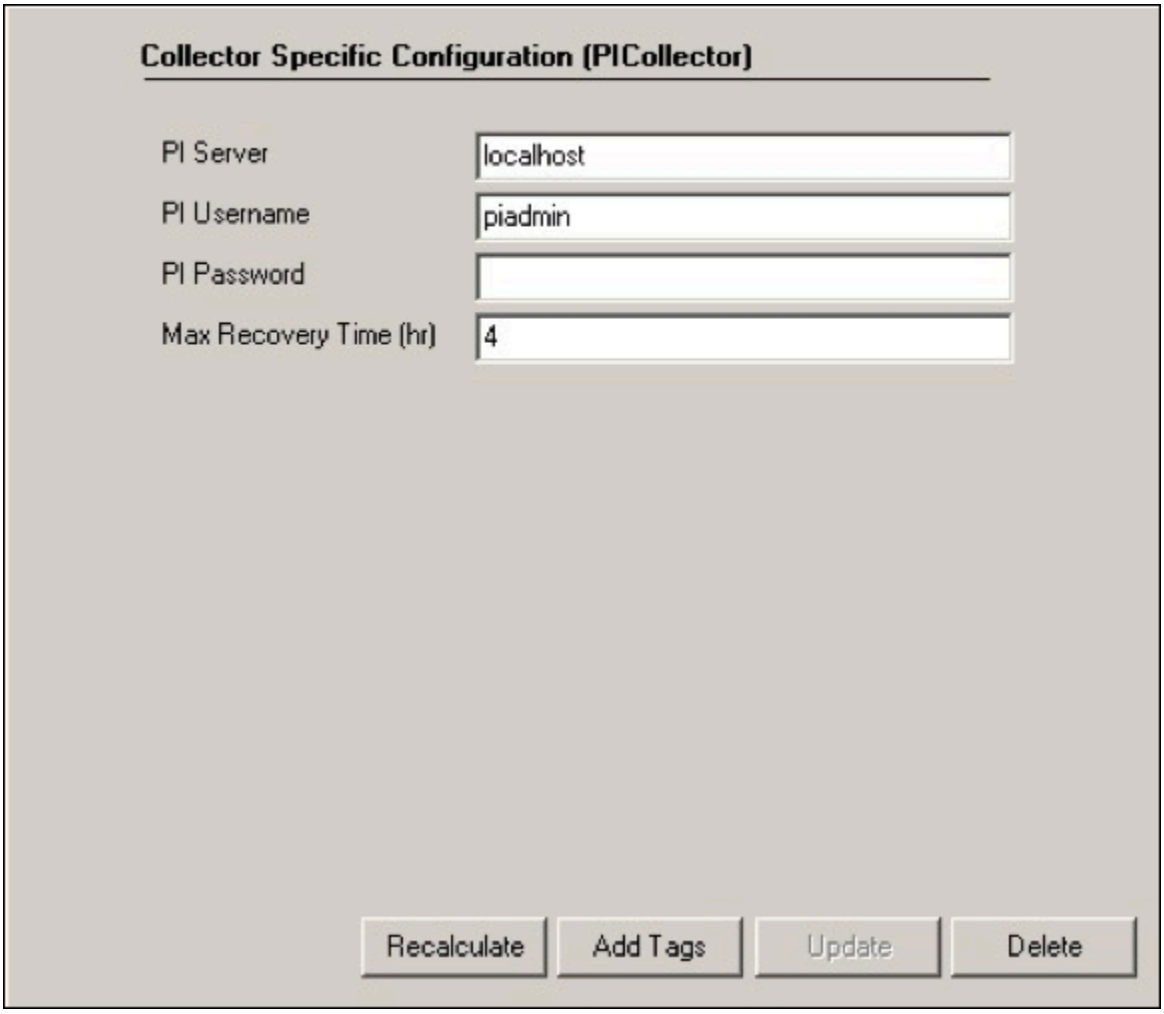
   b. Enter a prefix to add to OSI PI Tags in Historian, for example: `PI_`.

   c. Enter a value for the `Collection Interval`.
   The default collection interval is 1 second. Note that polled collection is not supported.

8. Configure the collector's advanced settings.
   a. To delay collection when the collector starts up, enter a value into the **Delay Collection at Startup (sec)** field.

## OSI PI Collector-specific Field Descriptions

The following figure shows the OSI PI Collector configured to collect data from an OSI PI archive on `localhost`, logging in as the `piadmin` user. It is also configured for a maximum recovery time of 4 hours.




The following table describes the OSI PI Collector-specific configuration fields.

| This field... | Indicates... |
|---|---|
| PI Server | The name of the computer that OSI PI is running on. This should match the server entry in the OSIsoft About PI-SDK utility. |
| PI Username | The user name required to connect to the OSI PI Data Archive. PI trusts are not supported and an explicit PI user must be set. |
| PI Password | The password required to authenticate with the OSI PI Data Archive. If no password is configured for the OSI PI user, this field can be left blank. |
| MaxRecovery Time (hr) | The Maximum Recovery time in hours. |

## Tag Attributes Available in Browse

You can specify tags for collection in Historian Administrator for the OSI PI Collector or OSI PI Distributor by browsing or by adding tags manually. The following table outlines the tag attributes available when browsing:

| Historian Tag Browse Attribute | OSI PI Tag Attribute |
|---|---|
| `Browse Source Address` | `Tag` |
| `Description Property` | `Descriptor` |
| `Engineering Unit Description` | `Engunits` |
| `Hi Engineering Units` | `Zero+span` |
| `Lo Engineering Units` | `Zero` |

When entering tags manually, it is important to match the Historian tag data type with the data type of the OSI PI tag. See OSI PI Collector and Distributor Supported Data Types *(on page 2451)*.

Regardless of how you add tags to Historian, you should match the timestamp resolution. If your OSI PI tag is using timestamps with milliseconds, then configure your Historian tag to store timestamps with millisecond resolution as well.

The OSI PI distributor reads data from the Historian tag displayed in the **Tag Source Address** field, and sends it to the OSI PI tag name displayed in the **Spare 1** field. To control the source and destination tags, change the **Tag Source Address** and **Spare 1** fields.

## Configuring Recovery Mode

Recovery logic is activated when the OSI PI Collector and OSI PI Data Archive reestablish a connection after a connection loss, or when the OSI PI Collector is started. The OSI PI Collector will attempt to recover all data samples between the current time and the last known write time, up to a maximum number of hours configured for the collector. Continuous collection resumes only after the previous data has been recovered.

The default recovery time available is 4 hours. You can disable recovery mode by setting the **Maximum Recovery Time** to 0 hours.

**To configure a maximum recovery time:**

1. Start Historian Administrator.
2. Access the **Collectors** page.
3. Select **OSI PI Collector**.
4. Select **Configuration**.
5. In the **Maximum Recovery Time (hr)** field, enter a Maximum Recovery Time, in hours. If this field is left blank, the Maximum Recovery Time will be set to 4 hours.

## OSI PI Collector and Distributor Supported Data Types

The following table maps OSI PI data types to their Historian data type equivalents:

| OSI PI Tag Types | Recommended Historian Data Type |
|---|---|
| `INT32` | `Single Integer, Double Integer` |
| `FLOAT16, FLOAT32` | `Float` |
| `FLOAT64` | `Double Float` |
| `PISTRING` | `Variable String` |
| `DIGITAL` | `Single Integer`<br>The accompanying digital states are transferred from OSI PI into Historian upon initialization of the collector. |

**Data Quality Mapping**

Data quality mapping from Historian to OSI PI Data Archive is restricted to `good`/`bad`. OSI PI's subtypes are not currently mapped by Historian.

## OSI PI Collector - Notes

**Time Stamps Not Modified by Historian**

The Historian OSI PI collector does not modify the time stamps placed on data samples by the OSI PI Data Archive. As a result, if your system clocks are not synchronized on both your OSI PI Data Archive server and Historian server, it is possible that Historian may receive data samples in the future or in the past.

**PITimestamp Data Type not Supported**

The `PITimestamp` data type is not supported by the OSI PI Collector. If you attempt to collect a tag with the `PITimestamp` data type, an error will be logged and the value will not be collected.

**PI Digital States/Enumerations**

Upon manual startup of the OSI PI collector (that is, a restart via the services control panel), the PI Digital States are imported into Historian. When enumeration tags are subsequently configured (digital tags in OSI PI), the matching enumeration is set as well and data will then be matched against the enumeration.

For a PI digital tag, we create a Historian enumerated tag and for a PI digital set. If the PI tag contains values from its enumerated set, then it values are written correctly in the Historian tag. But, in some special cases, the PI tag may contain values from a special set (System set). Historian does not support these values as we cannot assign two enumerated sets to one tag. In these cases, the values are written as `0- Bad quality`.

**Note:**
To update the enumerations, see Configuring Auto-synchronization of Digital States *(on page 2453)*.

**Connecting to an OSI PI Collective**

The Historian PI Collector has the ability to connect to an OSI PI Collective (PI redundancy). This allows data to be moved to Historian to prevent data loss.

## Starting and Stopping the OSI PI Collector

The OSI PI Collector runs as a Windows service and can be controlled through the **Services** control panel. You must have Administrator rights to access the **Services** control panel.

**To Modify the OSI PI Collector Service:**

1. Select **Start > Settings > Control Panel**.
2. Double-click the **Administrative Tools** control panel to open it.
3. Double-click the **Services** control panel to open it.
4. Double-click the **Historian PI Collector** service.
5. To configure the Historian PI Collector service to start when Windows starts, set the **Startup Type** to **Automatic**.
   To configure the Historian PI Collector service to start manually, set the **Startup Type** to **Manual**.

6. To start the service, select the **Start** button. To stop the service, select the **Stop** button.
   Other options are available in the **Services** control panel. For more information, refer to *Windows* documentation.

## Configuring Auto-synchronization of Digital States

**Auto-synchronization with the SynchInterval Registry Key**

The Historian OSI PI Collector allows transferring digital sets from the PI Server to Historian as enumerated sets. The digital sets from the PI Server are transferred to Historian not only during collector startup but also with the frequency of the Synch Interval.

Auto-synchronization of digital states can be configured with the `SynchInterval` Registry Key. The frequency at which the OSI PI Collector fetches the digital set from the PI server to Historian is determined by the value configured for this key, which is specified in minutes.

**Values for the SynchInterval Registry Key**

The default value of this key is `0`, which indicates there is no synchronization of the digital set.

Set this registry key to a value greater than `0` if you want the OSI PI Collector to automatically fetch the digital set from the PI Server to Historian.

For example, the OSI PI Collector fetches the digital set from the PI server to Historian every 10 minutes if the value configured in this key is `10`.

**Note:**

- The collector must be restarted to apply the change in the value of this registry key.
- This key is available for all Historian Collectors, but it is functional only in the OSI PI Collector.
- Digital sets from the PI Server are transferred to Historian only during collector startup.

## Renaming Digital States

If digital states are renamed in the PI Collector, the corresponding enumerated sets in Historian are not automatically renamed without a restart of the PI collector. Unused enumerated sets cannot be automatically deleted, because the PI Server does not notify the PI collector about rename activity.

When a digital state is renamed, a new enumerated set is created. There will therefore be two enumerated sets for the same digital set, one of which has the old name and one of which has the new name. The old enumerated set is redundant as it will not get any data if no tag is assigned to it. The tag association is also lost.

To address this:

- Manually assign the new enumerated set to the applicable tag in Historian. It will then be synchronized to represent the data for the renamed digital state.
- Go to the Historian VB admin and delete the unused enumerated set at the Historian side.

For example, consider the following situation, where `PHsite1.PumpStatus` in Historian is assigned to `Tag1`:

| PI Server | Historian |
|---|---|
| `Site1.PumpStatus` | `PHSite1.PumpStatus` |
| `ON= 0` | `OFF= 0` |
| `OFF= 1` | `ON= 1` |

If the digital state `Site1.PumpStatus` is renamed to `SiteLodha.PumpStatus`, a new enumerated set `PHSiteLodha.PumpStatus` is created at the Historian side.

The old enumerated set on the Historian side `PHSite1.PumpStatus` will not show any data, but it will exist redundantly inside **Historian**.

To address this:

- Manually assign `Tag1` (applicable tag in Historian) to `PHSiteLodha.PumpStatus`.
- Go to the **Historian** VB admin and manually delete the old enumerated `PHSite1.Pump Status`.

## Deleting Digital States

If digital states are deleted in the PI Collector, the corresponding enumerated sets in Historian are not automatically deleted without a restart of the PI collector. Unused enumerated sets cannot be automatically deleted, because the PI server does not notify the PI collector about delete activity.

To address this:

- Go to Historian Administrator and delete the unused enumerated set at the Historian side.

For example, consider the following situation, where `PHsite1.PumpStatus` in Historian is assigned to `Tag1`:

| PI Server | Historian |
|---|---|
| `Site1.PumpStatus` | `PHSite1.PumpStatus` |
| `ON= 0` | `OFF= 0` |
| `OFF= 1` | `ON= 1` |

If the digital state `Site1.PumpStatus` is deleted, then the enumerated set `PHSite1.PumpStatus` remains at the Historian side.

To address this:

- Go to Historian Administrator and delete the `PHSite1.PumpStatus` enumerated set, on the Historian side.

# OSI PI Collector Troubleshooting

**Tracing Operational Issues with OSI PI Collector Log Files**

The OSI PI Collector generates logs during initialization, configuration, and general operation. These can be found in the general logging folder ([Historian Data folder]\LogFiles). Log files for the OSI PI Collector begin with PICollector and have a sequence number for each collector restart.

The first response to an issue should be an examination of the latest log file, as it will contain details on:

1. Configuration of the collector.
2. Initialization and connection to the OSI PI server.
3. Addition and removal of tags.
4. Reconnection and recovery of data.

**OSI PI tags cannot be browsed, collector logs indicate SDK connection issues**

The OSI PI server used must be an entry in the About-PI SDK utility. If there is an issue with collecting tag information and/or enumeration information, please confirm that the OSI PI server that is set in the collector configuration has an entry in the About PI-SDK, and that it is accessible from the About PI-SDK utility.

**OSI PI Collector overrun problems - missing blocks of values, or is missing values**

If the OSI PI Collector is unable to maintain updates from the OSI PI archiver and therefore is missing values, you may experience overrun problems. An overrun occurs when the data source is changing tag values faster than the collector collecting values, which causes it to consistently remain behind the

archiver updates. If this is the case then the collector is running against the hardware and/or network limits and you may consider partitioning the tags into two or more sets, each with independent collectors.

**Startup issue when data source is Snapshot**

Historian OSI PI Collector fails to start as a service when you enter Snapshot as the **Data Source** in the **Configuration** section of the collector.

To start the collector:

1. Open command prompt as Administrator.
2. Navigate to the location where Historian installed.
3. Navigate to the path `Program Files (x86)\GE Digital\Historian OSI Pi Collector\Server`
4. Enter the following command:

   ```
   ihPiCollector.exe noservice
   ```

The Collector starts from the CLI interface.

# Chapter 31. OSI PI Distributor

## OSI PI Distributor

### Overview of the OSI PI Distributor

The Historian OSI PI Distributor gathers data from Historian, and writes it to an OSI PI data server. Data can be written directly to the OSI PI Data Archive v3.2 or greater, via the OSI PI v 1.3.4 or greater. Typically, the distributor is installed on the Historian server, distributing data to a remote OSI PI Data archive.

The OSI PI Distributor uses unsolicited distribution, whereby changes in Historian tags values are detected, and are forwarded to a remote OSI PI data server. The distributor is intended to duplicate data from an Historian archive to an OSI PI data archive.

One OSI PI Distributor can distribute data to a single OSI PI data archive. To distribute to multiple OSI PI archives from an Historian archive, you need to configure multiple OSI PI distributors. You can also configure multiple OSI PI distributors to a single OSI PI data archive.

**Note:**
The OSI PI Distributor can only write data to PI Archive. It cannot write data to PI Snapshot.

#### OSI PI Distributor Features

| Feature | Capability |
|---|---|
| Browse Source For Tags | Yes |
| Browse Source For Tag Attributes | Yes |
| Polled Collection | No |
| Minimum Poll Interval | N/A |
| Unsolicited Collection | Yes |
| Time stamp Resolution | milliseconds or seconds |
| Accept Device Time stamps | Yes |
| Floating Point Data | Yes |
| Integer Data | Yes |

| Feature | Capability |
|---|---|
| String Data | Yes |
| Binary Data | No |
| Array Data | No |
| Collector Status Outputs | Yes |

## Getting Started

Before you begin using the OSI PI Distributor, you should read the System Requirements *(on page 2458)* and About Configuring OSI PI Data Archiver for OSI PI Distributor *(on page 2458)* topics.

## System Requirements

The following software is required to use the PI Distributor:

- OSI PI Data Archive v 3.2 or greater
- OSI PI v 1.3.4 or greater
- Historian 3.0 or greater

## About Configuring OSI PI Data Archiver for OSI PI Distributor

In order for the OSI PI Distributor to write data to the OSI PI data archiver, make the following configuration changes to the OSI PI data server:

- Before using the OSI PI Distributor, you must create writable tags in the OSI PI data archive.
- In order for the OSI PI Distributor to write data to a OSI PI data archive, the destination tags must first be given write access in OSI PI. Consult your OSI PI documentation for more details.
- The OSI PI user must have write privileges.

  In addition to making the destination OSI PI data tags writable, you must also ensure that the OSI PI user the OSI PI Distributor is logging into the OSI PI data server has write access to the OSI PI data archive. If PI Trust or PI Mapping security is used, the corresponding PI users/Identities/Groups must also have write access to the OSI PI data archive. Consult your OSI PI documentation for more details.

## Configuring Multiple OSI PI Distributors to use Registry Keys

**Note:**
This procedure is for advanced Windows users only. If you are not familiar with Windows Registry editing, contact your Network Administrator for assistance.

To configure Multiple OSI PI Distributors to use Registry Keys

1. From a command prompt, run the Registry Editor (`regedit.exe`).
2. Navigate to `HKEY_LOCAL_MACHINE\SOFTWARE\Intellution, Inc.\iHistorian\Services\PIDistributor 3`.
3. Create a new key and give it a unique name. For example, `AlbPI01`.
4. Edit the new registry key and add a string value named `InterfaceName`.
5. In the `InterfaceName` value, enter a name for the OSI PI Server's interface.
   The string *"OSI PI"* is reserved, and should not be used.
6. Add a second string value to the new key named `Historian NodeName`.
7. In the Historian `NodeName` value, enter the name of the Historian archive server to which the new distributor will be sending data.
8. Close the Registry Editor.
9. Open a command window.
10. From the command prompt, run the OSI PI Distributor and command it to use the new registry key, with the `-multiple` and `REG=` parameters.
    For example, if you named the registry key `AlbPI01`, you would use the following command:

    ```
    iHPIdistributor.exe -multiple REG=AlbPI01
    ```

    **Note:**
    On a 64-bit Windows Operating System, all 32-bit components (such as collectors, Client Tools, and APIs) related registry keys will be located here:

    ```
    HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\\Intellution, Inc.\iHistorian\
    ```

# OSI PI Distributor Configuration

## Configuring an OSI PI Distributor

To configure an OSI PI Distributor:

1. Start Historian Administrator.
2. Select the **Collectors** page.
3. Select the OSI PI Distributor you want to configure.
4. Configure the OSI PI Distributor's **General** options.
    a. Enter a description for your OSI PI Distributor.
    b. In the **Computer Name** field, enter the hostname of the computer your distributor is running on, for example, `localhost`.
    c. If you wish to set limits on memory usage and free disk space, change the values in the **Memory Buffer Size** and **Minimum Free Space** fields.
5. Configure the OSI PI Distributor-specific options:

    a. Select **Configuration**.

    b. In the **PI Server** field, enter the name of the machine where the OSI PI server is running. For example, `localhost`.

    c. In the **PI Username** field, enter the OSI PI user name. For example, `PIAdmin`.

    d. Enter the OSI PI password into the **OSI PI Password** field. The password is required to authenticate the OSI PI Data Archive.
    If no password is configured for the OSI PI user, this field can be left blank. Passwords are case-sensitive.

    e. In the **Max Recovery Time (hr)** field, enter a new maximum recovery time, in hours.
    Recovery logic is activated when the OSI PI Distributor and Historian re-establish a connection after a connection loss, or when the distributor is restarted. The OSI PI Distributor will attempt to recover all data samples between the current time and the last known write time, up to a maximum number of hours configured for the distributor. New distribution resumes only after the previous data has been recovered.

    The default recovery time available is 4 hours. You can disable recovery mode by setting the **Maximum Recovery Time** to 0 hours.

6. Configure the default distribution options.
    a. Select **Tags**.
    b. Enter a prefix to add to OSI PI tags in Historian. For example, PI_
    c. Enter a value for the Collection Interval. The default collection interval is 1 second.
7. Configure the distributor's advanced settings.
    To delay distribution when the distributor starts up, enter a value into the **Delay Collection at Startup (sec)** field.

## Tag Attributes Available in Browse

You can specify tags for collection in Historian Administrator for the OSI PI Collector or OSI PI Distributor by browsing or by adding tags manually. The following table outlines the tag attributes available when browsing:

| Historian Tag Browse Attribute | OSI PI Tag Attribute |
|---|---|
| `Browse Source Address` | `Tag` |
| `Description Property` | `Descriptor` |
| `Engineering Unit Description` | `Engunits` |
| `Hi Engineering Units` | `Zero+span` |
| `Lo Engineering Units` | `Zero` |

When entering tags manually, it is important to match the Historian tag data type with the data type of the OSI PI tag. See OSI PI Collector and Distributor Supported Data Types *(on page 2451)*.

Regardless of how you add tags to Historian, you should match the timestamp resolution. If your OSI PI tag is using timestamps with milliseconds, then configure your Historian tag to store timestamps with millisecond resolution as well.

The OSI PI distributor reads data from the Historian tag displayed in the **Tag Source Address** field, and sends it to the OSI PI tag name displayed in the **Spare 1** field. To control the source and destination tags, change the **Tag Source Address** and **Spare 1** fields.

## OSI PI Collector and Distributor Supported Data Types

The following table maps OSI PI data types to their Historian data type equivalents:

| OSI PI Tag Types | Recommended Historian Data Type |
|---|---|
| `INT32` | `Single Integer, Double Integer` |
| `FLOAT16, FLOAT32` | `Float` |
| `FLOAT64` | `Double Float` |
| `PISTRING` | `Variable String` |

| OSI PI Tag Types | Recommended Historian Data Type |
|---|---|
| `DIGITAL` | `Single Integer`<br><br>The accompanying digital states are transferred from OSI PI into Historian upon initialization of the collector. |

**Data Quality Mapping**

Data quality mapping from Historian to OSI PI Data Archive is restricted to `good`/`bad`. OSI PI's subtypes are not currently mapped by Historian.

## Starting and Stopping the OSI PI Distributor Service

The OSI PI Distributor runs as a Windows service and can be controlled through the **Services** control panel. You must have Administrator rights to access the **Services** control panel.

**To start/stop the OSI PI Distributor service**

1. Select **Start > Settings > Control Panel**.
2. Double-click the **Administrative Tools** control panel to open it.
3. Double-click the **Services** control panel to open it
4. Double-click the **Historian PI Distributor** service.
5. To configure the Historian PI Collector service to start when Windows starts, set the **Startup Type** to **Automatic**.
   To configure the Historian PI Distributor service to start manually, set the **Startup Type** to **Manual**.
6. To start the service, select the **Start** button. To stop the service, select the **Stop** button.
   The other options are available in the **Services** control panel. For more information, refer to *Windows* documentation.

# Chapter 32. The Python Collector

## Overview of the Python Collector

Using the Python collector, you can execute Python scripts and store the resulting values in Historian tags. You can retrieve this data from the Historian archive, perform the calculations written in Python script, and store the resulting values in new Historian tags. Also, you can run multiple Python scripts simultaneously.

> **Note:**
> You can use the Python Collector either Configuration Hub or Historian Administrator. However, for a seamless usage and experience, it is recommended to use the Python collector in Configuration Hub.

**Features:**

- You can perform data calculations on values that are already in the archiver.
- You can run the Python-based scripts to compute values.
- You can retrieve the resulting values stored in Historian using any of the Historian clients.
- You can run multiple Python scripts at the same time.
- You can verify a Python script and check for errors before executing it.
- The supported timestamp resolution is 1ms.
- Each tag can have its own Python script stored under the tag source address. The computed value for the `Result` variable within the Python script is stored in the associated Python tag.
- Both the polled and unsolicited data collection are supported.
- Integer, string and other data types are supported. For more information, refer to Supported Data Types *(on page 2464)*.
- The collector accepts device timestamps.
- The collector reads data and tags.

**Limitations:**

- Historian tags used in the CurrentValue function cannot contain % in the tag name. For example, CurrentValue('Simulation.Sin_1%Noise') will result in an error. Therefore, before using a tag in a Python script, rename it so that it does not contain %.

## Supported Data Types

The Python collector supports integer, string, and other data types. This topic lists all the data types that are supported by the Python collector.

| Data Type | Data Type in Python Collector |
|---|---|
| int, long | Single Integer, Double Integer |
| float | Single Float, Double Float |
| bool | boolean |
| Byte | Byte |
| str | Variable String |
| Sequence types: list, tuple | Array of numeric type or strings |
| Mapping data type: dictionary | Multifield (user defined types) |

# Install the Python Collector

1. Install the Historian server *(on page 104)* and collectors *(on page 142)*.
2. Install Python 3.8 on the same machine on which you have installed collectors.

1. Add the following entries to update the Python collector's registry:
    - In the following location, include the path to the Python install lib folder:

      `Computer\HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\GE Digital\iHistorian\CollectorServiceExtensions\PythonExpressions\PythonPath`

    - In addition, add the path to the Python 3.8 lib folder or any custom modules to Python path. This path can also include location of any custom modules or functions (global functions or variables to be used from within the python tag calculation/script).

      **Examples**:

      `C:\Program Files (x86)\GE Digital\Historian Python Expressions\Python38\lib`

      `C:\Program Files (x86)\GE Digital\Historian Python Expressions\Python38\user`

      `C:\Users\Administrator\AppData\Local\Programs\Python\Python38-32\Lib`

```
C:\Users\Administrator\AppData\Local\Programs\Python\Python38-32\Lib\site-packages

C:\Users\Administrator\AppData\Local\Programs\Python\Python38-32\DLLs
```

You can also configure Python library path using Configuration Hub. For more information, refer to Configure Python Library path using Configuration Hub.

2. Update any default modules to be used for python tags to key:
   `DefaultModuleImports`

Add a Python Collector Instance, either using Configuration Hub *(on page 556)* or RemoteCollectorConfigurator *(on page 2467)*.

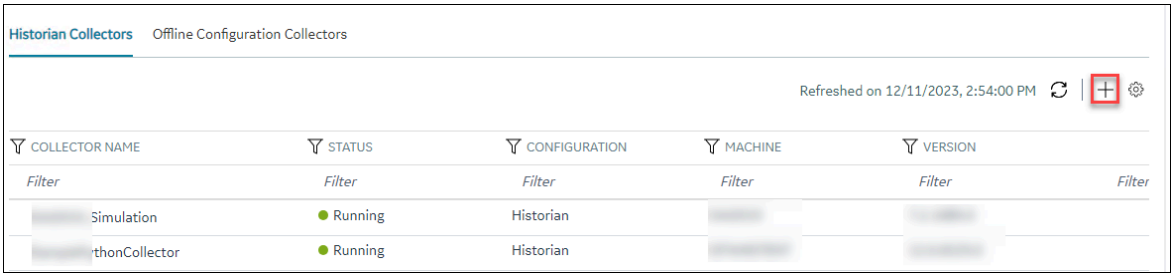
# Adding a Python Collector Instance

## Add a Python Collector Instance using Configuration Hub

- Install Python Collector *(on page 2464)*.

This topic describes how to add a Python collector instance using Configuration hub.

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**. A list of collectors in the default system appears.
3. In the upper-right corner of the main section, select +.




The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.

4. In the **MACHINE NAME** field, select the machine in which you want to add a collector instance.
5. In the **COLLECTOR TYPE** field, select **Python Collector**, and then select **Get Details**.

> **Note:**
> The **INSTALLATION DRIVE** and **BASE DATA DIRECTORY** fields cannot be changed. This is the drive location and the data directory folder that you provided during Collectors installation.

The **INSTALLATION DRIVE** and **DATA DIRECTORY** fields are populated with the drive location and the data directory folder.

6. Select **Next**.
   The **Destination Configuration** section appears. The collector machine name provided by you is selected as the **Source Configuration** by default.

   Under **CHOOSE DESTINATION**, the **Historian Server** option is selected by default. In addition, the **DESTINATION HISTORIAN SERVER** field is disabled and populated with the collector machine name.

7. Select the destination to which you want to send data, and then enter the values in the corresponding fields. You can send data to an on-premises Historian server or to a cloud destination.

   a. If you need to send data to a cloud destination, select the cloud destinations as needed.
      - **Predix Timeseries**- Select this if you need to send data to Predix cloud. For more information, refer to Predix Cloud *(on page 621)*.
      - **Azure IoT Hub**- Select this if you need to send data to Azure Cloud in KairosDB format. For more information, refer to Azure IoT Hub (KairosDB format) *(on page 608)*.
      - **MQTT**- Select this if you need to send data to any of the following cloud destination.
        - Alibaba cloud. For more information, refer to Alibaba Cloud *(on page 587)*.
        - AWS cloud. For more information, refer to AWS Cloud *(on page 594)*.
        - Google cloud. For more information, refer to Google Cloud *(on page 614)*.

   b. If you need to send data to an on-premises Historian server, select **Historian Server**.
      If you created security groups or enabled a strict client/collector authentication, enter the **USERNAME** and **PASSWORD** of the on-premises Historian server that you created during the installation of the collector.
      If you entered the **USERNAME**and **PASSWORD**, select **Test Connection**. This will help you to test if the Historian server that you are trying to connect is valid or if the credentials that you entered are valid.
      If the entered credentials are valid, a successful connection message appears.

8. After you selected the destination, select **Next**.
   The **Collector Initiation** section appears.
9. If needed, modify the value in the **COLLECTOR NAME** field.
   The value that you enter:
   - Must be unique.
   - Must not exceed 15 characters.
   - Must not contain a space.
   - Must not contain special characters except a hyphen, period, and an underscore.
10. In the **RUNNING MODE** field, select one of the following options.
    - **Service - Local System Account**: Select this option if you want to run the collector as a Windows service using the credentials of the local user (that is, the currently logged-in user). If you select this option, the **USERNAME** and **PASSWORD** fields are disabled.
    - **Service Under Specific User Account**: Select this option if you want to run the collector as a Windows service using a specific user account. If you select this option, you must enter values in the **USERNAME** and **PASSWORD** fields.

      If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:
      - iH Security Admins
      - iH Collector Admins
      - iH Tag Admins

    You can also configure the collector to start automatically when you start the computer.
11. Select **Add**.
    The collector instance is added. The fields specific to the collector section appear in the **DETAILS** section.

Specify the tags *(on page 357)* whose data you want to collect using the collector.

## Add a Python Collector Instance using RemoteCollectorConfigurator

- Install Python Collector *(on page 2464)*.
- If the destination of a collector is an Azure IoT Hub device, ensure that the device is running.

Before you begin using a collector, you must add an instance of the collector. You can add multiple instances of the same collector or instances of multiple collectors where you have installed the collectors.

This topic describes how to add a collector instance using the RemoteCollectorConfigurator utility. If you want to add an offline collector instance, refer to Add an Offline Collector Instance *(on page 819)*.

1. If you want to use an interactive UI:

    a. Run the `RemoteCollectorConfigurator.exe` file. By default, it is located in the following folder: `C:\Program Files\GE Digital \NonWebCollectorInstantiationTool`.
    A list of options to manage collector instances appears.

    b. Connect to the collector machine by entering `1` or `2`, depending on whether collectors are installed locally or on a remote machine.

    c. Enter `4`.
    You are prompted to choose between entering the installation parameters manually and providing a JSON file.

    d. If you want to manually enter the parameters and values, enter `1`, and then run the following command:

    ```
    {"<parameter>":"<value>","<parameter>":"<value>"}
    ```

    If you want to use a JSON file containing the installation parameters and values, enter `2`, and then enter the path to the JSON file that you have created. Instead of manually creating the JSON file, you can use the RemoteCollectorConfigurator utility to generate it automatically *(on page 796)*.

    You can leave the Historian username and password blank if there are no Historian security user groups.

2. If you want to use the Command Prompt window:

    a. Access the installation folder of the RemoteCollectorConfigurator utility. By default, it is `C:\Program Files\GE Digital\NonWebCollectorInstantiationTool`.

    b. Run Command Prompt in this location.

    c. If you want to manually enter the installation parameters and values, run the following command:

    ```
    RemoteCollectorConfigurator.exe "<Destination Historian>" "<Destination Historian username>"
    "<Destination Historian password>" InterfaceCreateViaCmd "{\"<parameter>\":\"<value>\",
    \"<parameter>\":\"<value>\"}"
    ```

If you want to use a JSON file containing the installation parameters and values, run the following command:

```
RemoteCollectorConfigurator.exe "<Destination Historian>" "<Destination Historian username>"
"<Destination Historian password>" InterfaceCreateViaFile "<path to the JSON file>"
```

Instead of manually creating the JSON file, you can use the RemoteCollectorConfigurator utility to generate it automatically *(on page 796)*.

If ih security groups are available, you must enter the Windows username and password of the destination Historian. If you have enabled the **Enforce Strict Collector Authentication** option, you must provide the credentials of a user who is added to at least one of the following security groups:

- iH Security Admins
- iH Collector Admins
- iH Tag Admins

For information on the parameters, refer to Collector Instance Parameters *(on page 801)*.

**Sample JSON file**

```
{
"CollectorSystemName":"TESTSYSTEM",
"DestinationHistorian":"TESTSYSTEM",
"General1":"10",
"General2":"4",
"General3":"",
"General4":"",
"General5":"",
"InterfaceDescription":"Sample Python Collector",
"InterfaceName":"SamplePythonCollector",
"InterfaceSubType":"Python",
"Type":"8",
"DataPathDirectory":"C:\\Proficy Historian Data",
"CollectorDestination":"Historian",
"DestinationHistorianUserName":"",
"DestinationHistorianPassword":"",
"Mode":"1",
"WinUserName":"",
"WinPassword":""
```

```
}
```

The collector instance is added.

- Specify the tags *(on page 357)* whose data you want to collect using the collector.
- If you did not enter a value, modify the offline configuration file of the collector. By default, this file is available in the following location: <*installation folder of Historian*>\GE Digital \<*collector name*>. For information, refer to Creating Offline Configuration XML file *(on page 2040)*.

# Configuring the Python Collector

## Configure the Python Collector using Configuration Hub

Before you begin, install the Python collector *(on page 2464)* and add an instance *(on page 556)* of the collector.

1. Access Configuration Hub *(on page 336)*.
2. Select **Collectors**, and then select the Python Collector instance that you want to configure.
   The fields specific to the collector instance appear in the **DETAILS** section.
3. In the **COLLECTOR SPECIFIC CONFIGURATION** section, enter values as specified in the following table.

| Field | Description |
|---|---|
| **Calculation Timeout (sec)** | This is not supported for the Python collector. |
| **Max Recovery Time (hr)** | The maximum time, in hours till now, that the collector will attempt to restore data. This is applicable only to event-based tags. The default value is 4 hours.<br><br>If you want to disable automatic calculation of the tag, set the value of this field to 0. |
| **MTLS Security** | Indicates whether you want to use Mutual TLS (MTLS) protocol to enforce a secure and strong authentication mechanism. |
| **MTLS Data Encryption** | Indicates whether you want to encrypt the data that the collector shares to the data archiver (DA). |

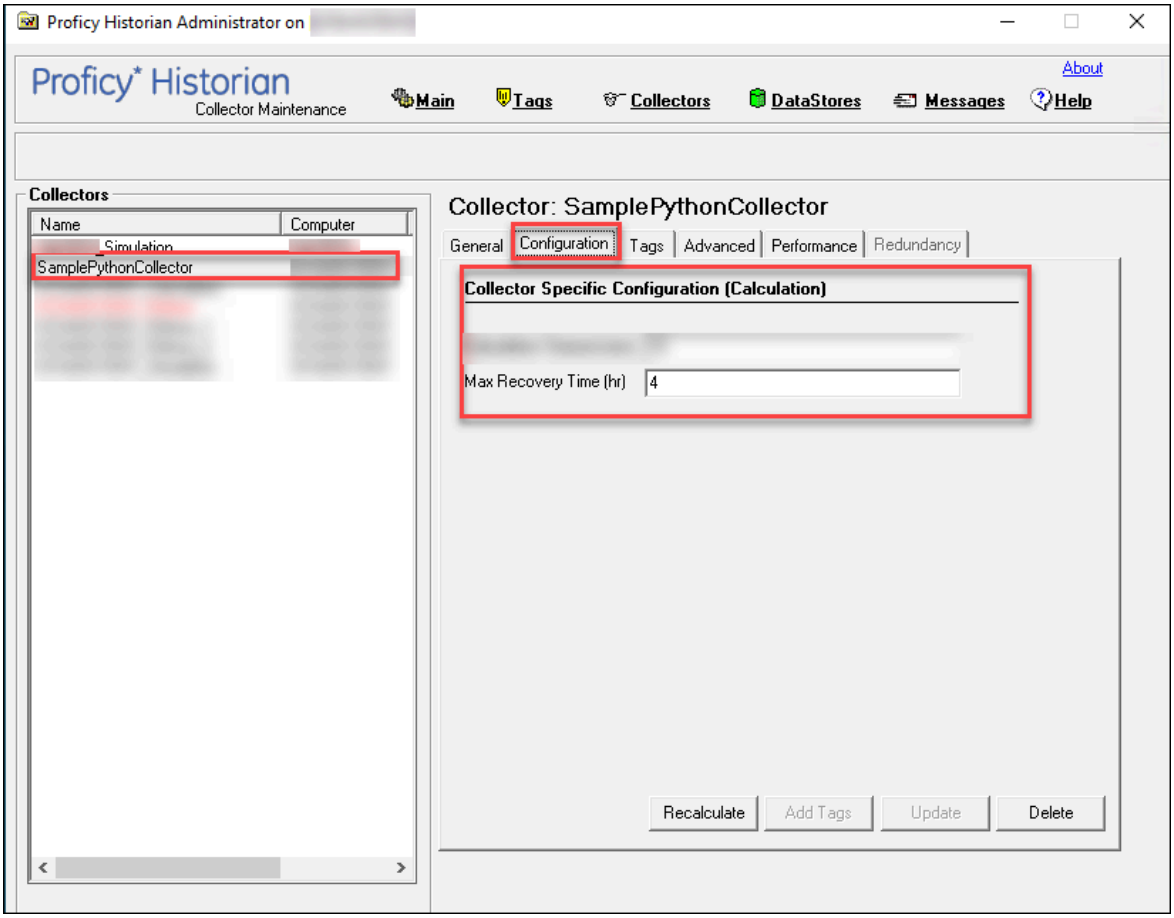
| Field | Description |
|---|---|
| For more information on how to enable MTLS Security, refer to Enable MTLS Security *(on page 632)*. | |

4. As needed, enter values in the other sections common to all collectors *(on page 579)*.
5. Restart the collector.

   The collector instance is configured.

## Configure the Python Collector using Historian Administrator

1. Access Historian Administrator *(on page 823)*.
2. Select **Collectors**, and then select the Python Collector instance that you want to configure.




3. In right-side, below the **Collector: <Name of the Collector Instance>**, select **Configuration**.

   The fields specific to the collector instance appear.
4. Enter values as specified in the following table.

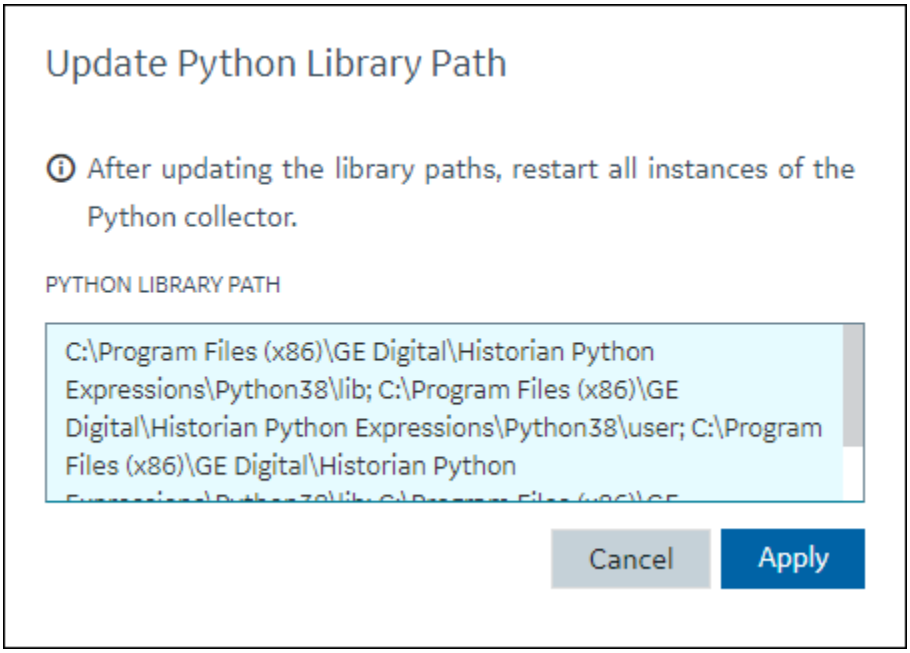
| Field | Description |
|---|---|
| Calculation Timeout (sec) | This is not supported for the Python collector. |
| Max Recovery Time (hr) | The maximum time, in hours till now, that the collector will attempt to restore data. This is applicable only to event-based tags. The default value is 4 hours.<br><br>If you want to disable automatic calculation of the tag, set the value of this field to 0. |

5. Select **Update**.

## Configure Python Library Path using Configuration Hub

If you want to configure additional paths for Python library or if you want to include paths that contain additional, custom or third-party modules, you can add those path to the registry using Configuration Hub. This topic describes how to configure additional Python Library path using Configuration Hub.

1. Access Configuration Hub *(on page 336)*.
2. Select **Collectors**, and then select the Python Collector instance that you want to configure.
   The fields specific to the collector instance appear in the **DETAILS** section.
3. In the **INSTANCE CONFIGURATION** section, select ⧉.
   The **Update Python Library Path** window appears.




4. In **PYTHON LIBRARY PATH** enter the library path(s) as needed.

> **Note:**
> While entering multiple paths, ensure to separate the library paths using semicolons.
>
> For example,
>
> ```
> C:\Program Files (x86)\GE Digital\Historian Python Expressions38\Python38\lib;
> C:\Program Files (x86)\GE Digital\Historian Python Expressions38\Python38\user;
> C:\Users\Administrator\AppData\Local\Programs\Python\Python38-32\Lib;
> C:\Users\Administrator\AppData\Local\Programs\Python\Python38-32\Lib\site-packages;
> C:\Users\Administrator\AppData\Local\Programs\Python\Python38-32\DLLs;
> ```

5. Select **Apply**
   The library paths are added to the registry.
6. For the changes to take effect, restart the collector instance.

# Using the Python Collector

## Write Data to an Arbitrary Tag

You can write data to an arbitrary tag in the Historian archive through the `Historian.AddData` function. This function is used in a python script to write values, time stamps and qualities of one or more tags to the Historian archive.

Use the following syntax ri write data to an arbitrary tag:

```
Historian.AddData(TagNames, Values, Timestamps, Qualities)
```

The following table provides information on the parameters.

| **Parameter** | **Description** |
|---|---|
| TagNames | Identifies the names of the tags. A value is required, and must exist in the archive to which you want to send the tag data. You can provide a single tag name or an array of tag names, enclosed in double quotation marks. |
| Values | Identifies the values of the tags. A value is required, and must be a single value or an array of values, depending on whether the tag name is a single name or an array. Values must be enclosed in double quotation marks. You can enter only a single value for each tag name. |
| TimeStamps | Identifies the timestamp of the tag data. Enter an absolute time value, enclosed in double quotation marks. All times are provided as string in standard ISO 8601 format. |
| Qualities | Identifies the quality of the tag data. Enter an integer from 0 to 100, with 0 indicating bad quality and 100 indicating good quality. |

**Example of using AddData Function**

```
from datetime import datetime

tags = ["TestTag1", "TestTag2", "TestTag3"]

values = ["111", "222", "333"]

datetime_str = Historian.CurrentTime()

datetime_object = datetime.strptime(datetime_str, '%m/%d/%y %H:%M:%S.%f')
```

```
times = [datetime_object.isoformat(),datetime_object.isoformat(), datetime_object.isoformat()]

Historian.AddData(tags, values,times, "")

Result=1
```

# Create Triggers

## About Calculation Triggers

You can create the following types of triggers for a calculation:

- **Polled or scheduled:** Used to trigger a calculation based on a scheduled time interval. For example, you can calculate the average value of tag data collected every hour.

  The polled type trigger functions the same as the other collectors. Although Historian internally optimizes calculation execution times, the data for polled tags is timestamped on the data collection interval. For example, if the calculation engine is unable to process the polled triggers as scheduled, the calculations will be executed later, but with data interpolated back to the scheduled time. If there are too many triggers to be processed, some triggers will be dropped and no samples are logged for that calculation time.

  For information on creating a polled trigger, refer to create a polled trigger *(on page 2476)*.

- **Unsolicited or event-based:** Used to trigger a calculation based on an event. For example, you can calculate the average value of tag data when the data exceeds a certain value.

  When you set an event-based trigger, you must also set up a dependency list of one or more tags. Event-based triggers will keep calculations as up to date as possible. They are also useful when you want to do on-demand calculations. You can use a trigger tag that is written to by an external program or operation.

  If you want to perform raw sample replication you would use an event-based trigger. To retrieve data from a tag, use the formula:

  ```
  Result=Historian.CurrentValue('Tag1')+Historian.CurrentValue('Tag2')
  ```

  If you are using recovery mode, all referenced tags in an unsolicited calculation must be listed as trigger tags because recovery will be performed only for the configured trigger tags.

  Event-based triggers have a dependency list of trigger tags. The trigger fires whenever there is a data change for the trigger tag (for example, changes in the quality and value of a trigger tag). The

value of a trigger tag can change when the tag exceeds the collector compression (if you enabled collector compression).

The calculation is processed each time any tag in the dependency list changes. If you have multiple tags in the list and they change even one millisecond apart, then you will have multiple events, and the calculation formula will be processed for each.

However, the following actions do not trigger a calculation:

- Deletion of a tag that is in the dependency list.
- Re-addition of a tag in the dependency list.

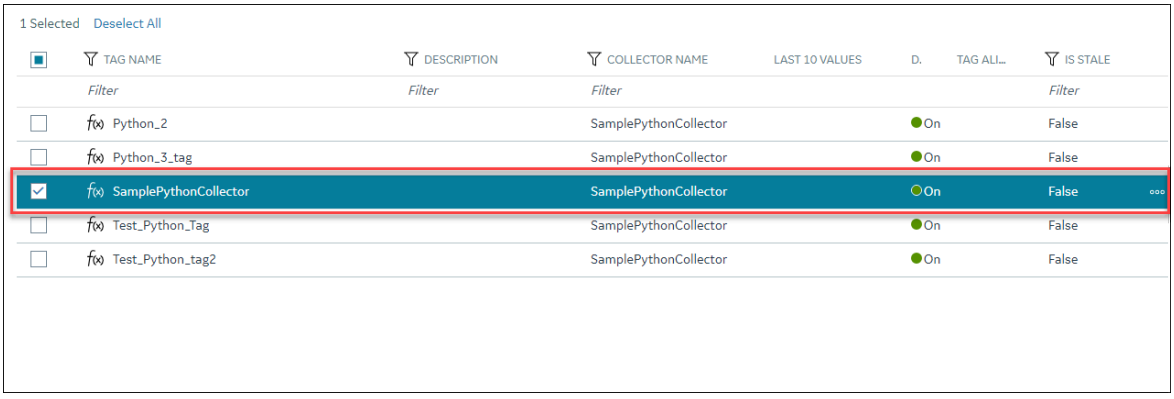
The calculation is triggered at the same time as the timestamp of the sample in the trigger tag. The values of all other tags in the formula are interpolated forward to this time so that the timestamps of all input tags are the same. Even if these are sequential events, they have the same timestamp. The calculation time becomes the timestamp for the sample stored in the destination tag.

Event-based triggers have a collection interval. The Python Collector notifies the archiver not to send notification of changes to trigger tags any faster than the collection interval setting.

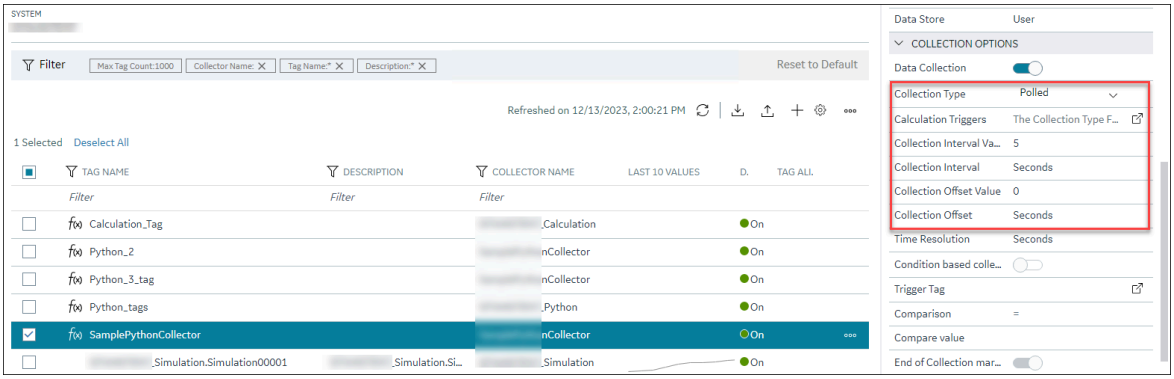
For information on creating an unsolicited trigger, refer to create an unsolicited trigger *(on page 2477)*.

## Create a Polled Trigger

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
   A list of tags appears.
3. From the list of tags, select the Python collector tag (say, SamplePythonCollector).



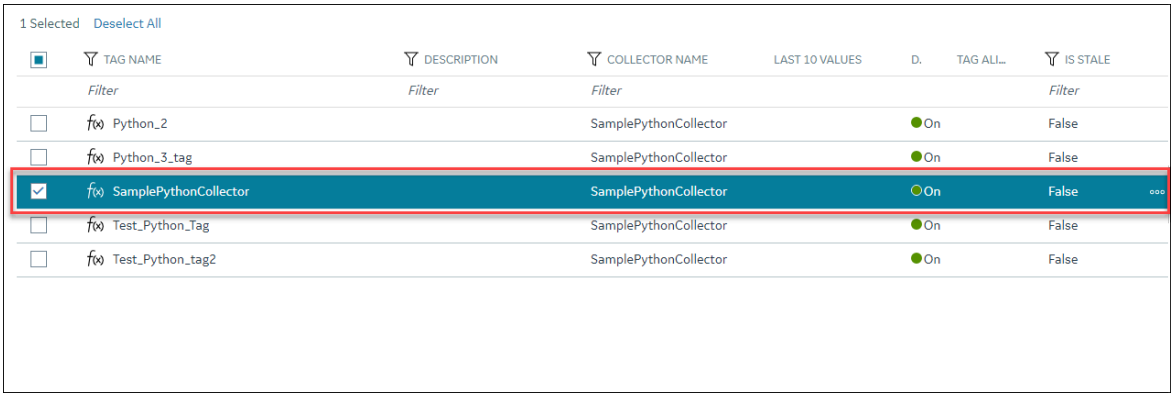

4. In **DETAILS**, in the **Collection Options** section, select **Polled** from **Collection Type**.



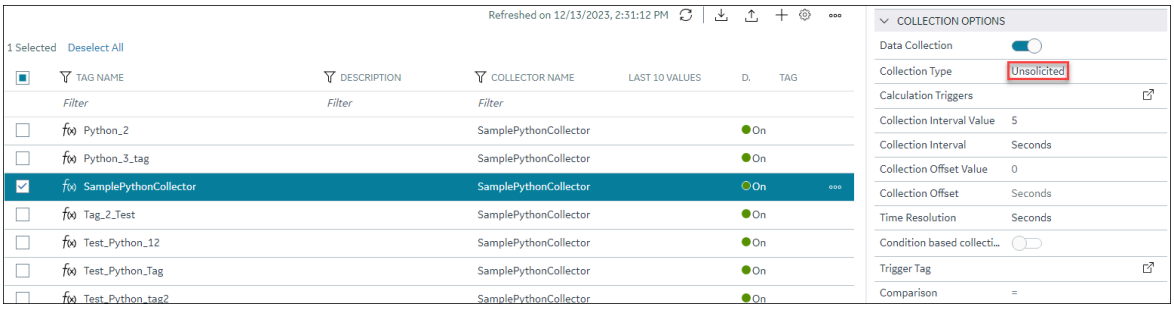

5. Set the **Collection Interval Value** and **Collection Offset** values. For example, if you want to set a trigger every day, set these values to 24 hours and 8 hours, respectively.
6. In the upper left corner of the page select **Save**.
   The triggers are created.

## Create an Unsolicited Trigger

1. Access Configuration Hub *(on page 336)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
   A list of tags appears.
3. From the list of tags, select the Python collector tag (say, SamplePythonCollector).

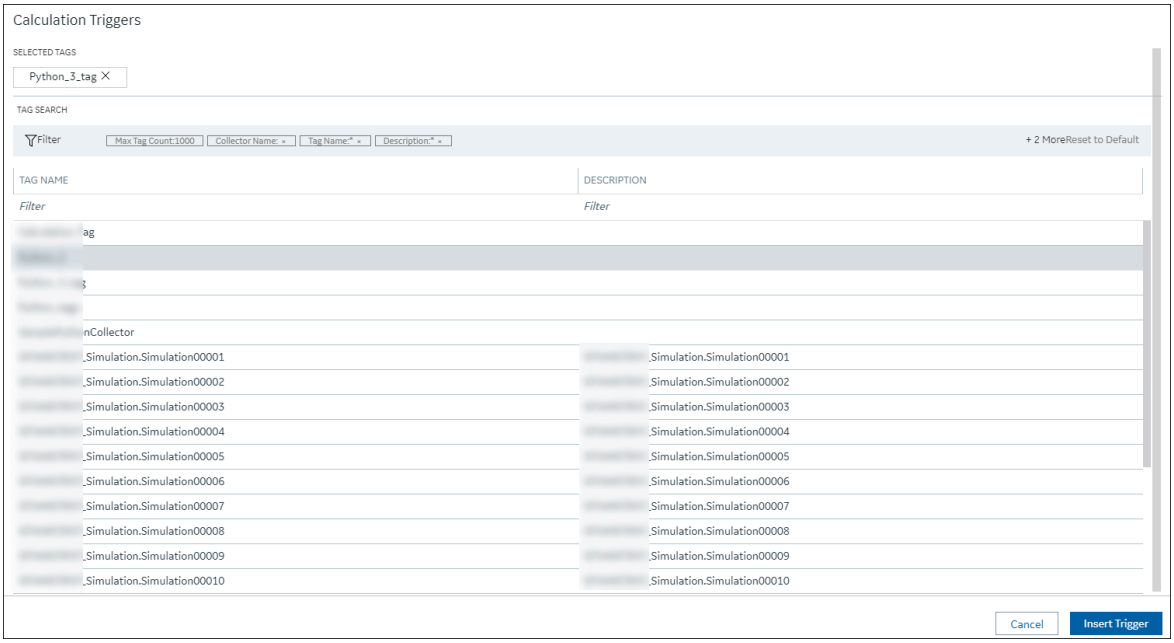



4. In **DETAILS**, in the **Collection Options** section, select **Unsolicited** from **Collection Type**.



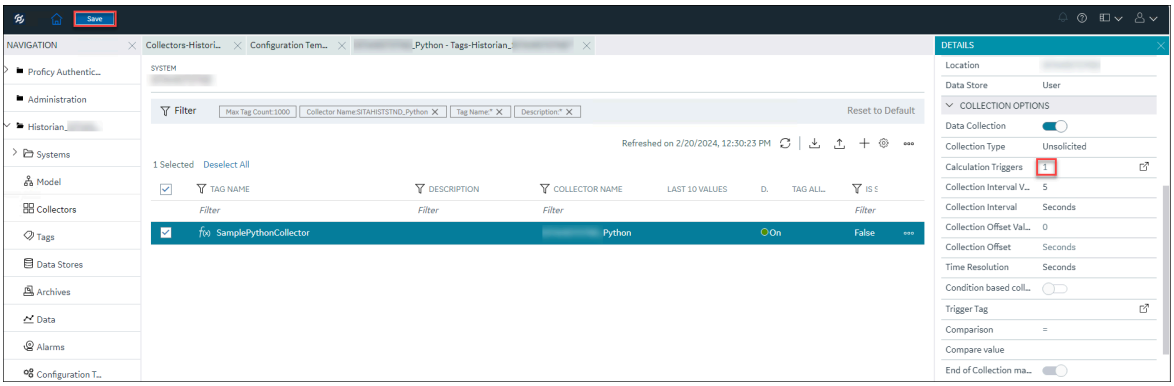
5. Select Calculation Triggers and select .

   The Calculation Triggers window appears.
6. Search for tags or select tags from the list as trigger tags.
7. You can remove the selected tags by selecting X from the Selected Tags list.




8. Select **Insert Trigger**.

   The trigger tag count will be displayed.




9. In the upper left corner of the page select **Save**.

   The triggers are created.

# About Calculations

## About Calculations using Python Collector

To perform a calculation using the Python Collector, you must define the Python script. You can define the Python script in one of the following ways:

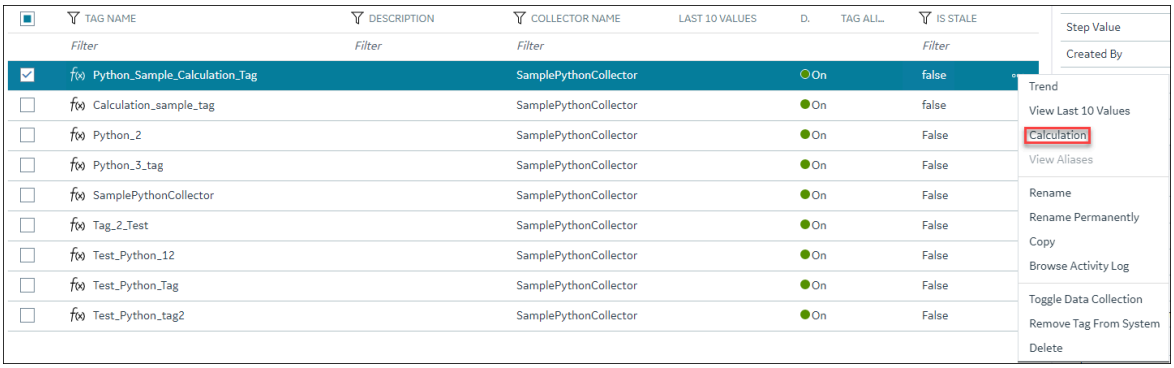
> **Note:**
> To seamlessly create Python script, it is recommended to use Configuration Hub instead of Historian Administrator.

- Using pre-built functions *(on page 2478)*.
- Using third-party or custom Python modules *(on page 2481)*.

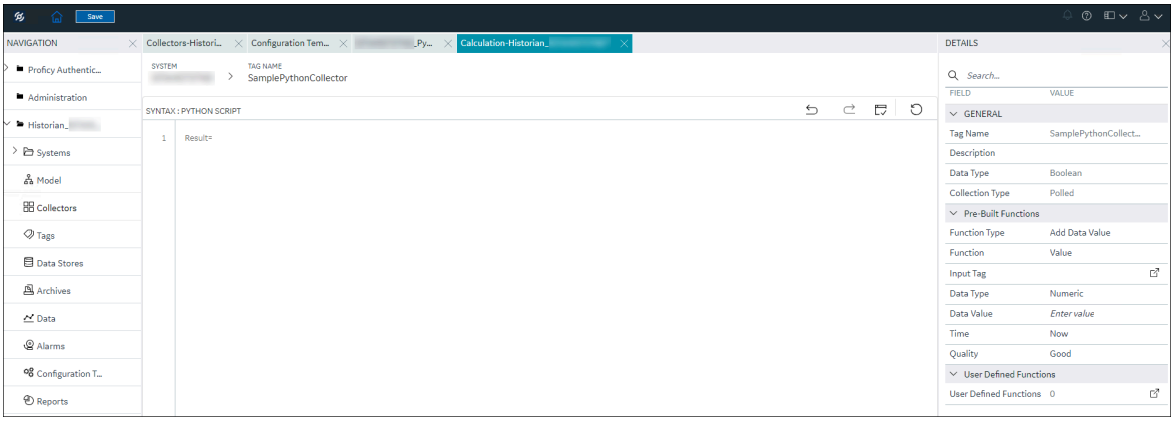
## Create Python Script using Built-in Functions

This topic describes how to create a Python script using the pre-built functions. For more information on the built-in functions, refer to available functions *(on page 2483)*.

1. Access Configuration Hub *(on page 336)*
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
   A list of tags appears.
3. Select a Python collector tag, right-click or select more options, and then select **Calculation**.




The **PYTHON SCRIPT** editor appears.

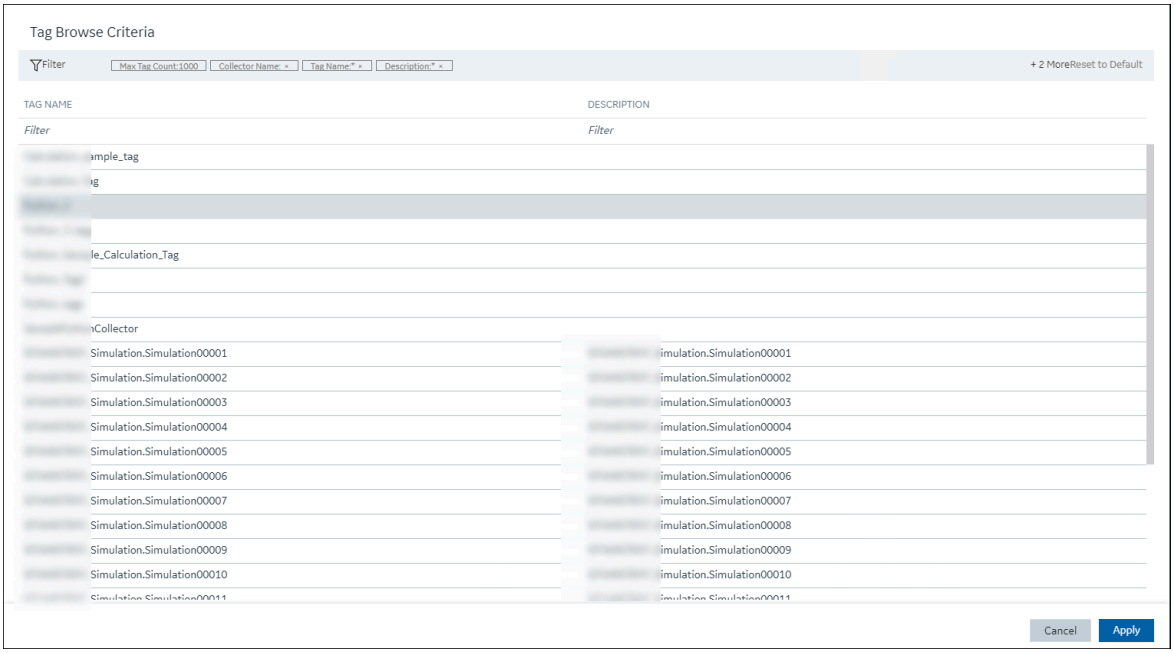



4. In the scripting section, remove `Null` (and retain the `Result =`).
5. In the **DETAILS** section, under **Pre-Built Functions**, in the **Function Type** field, select a function type.

   Depending on the function type you have selected, a list of functions appears in the **Function** field.
6. In the **Function** field, select a function.

   Based on the selected function, the related fields will be displayed.
7. Enter values in the other fields that appear after selecting a function.
8. In the **Input Tag** field, select , and then select the tag where applicable.

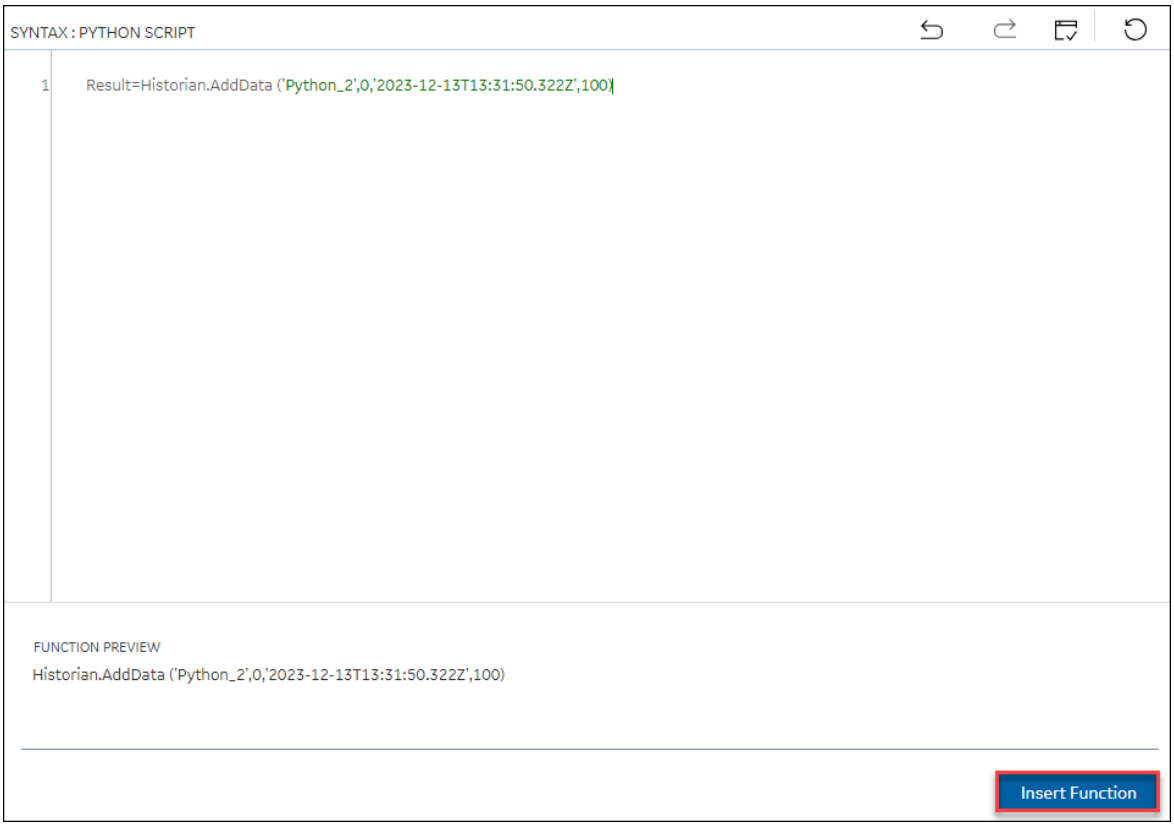
   The **Tag Browser Criteria** window appears.




9. Select **Apply**.

   The selected tag is added as the input tag.

10. Select **Insert Function**.




    The function is inserted at the cursor position and the function preview appears below the calculation editor.

11. To test the function, select .
    A message appears, stating whether the syntax is correct.
12. In the upper-left corner of the page, select **Save**.
    The Python script is created.

## Create Python Script by Importing Third-party or Custom Python Modules or Functions

This topic describes how to create a Python script by importing custom or third-party Python modules or functions.

1. In the same machine where you installed Python 3.8, install the needed modules or create your own functions.
2. After you install the modules, for the collector to recognize the modules, update the following path to include the location where you installed the modules or created your own functions:

```
Computer\HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\GE

 Digital\iHistorian\CollectorServiceExtensions\PythonExpressions\PythonPath
```

After you update the path, the changes are reflected as follows:

```
C:\Program Files (x86)\GE Digital\Historian Python Expressions\Python38\lib
```

```
C:\Program Files (x86)\GE Digital\Historian Python Expressions\Python38\user
```

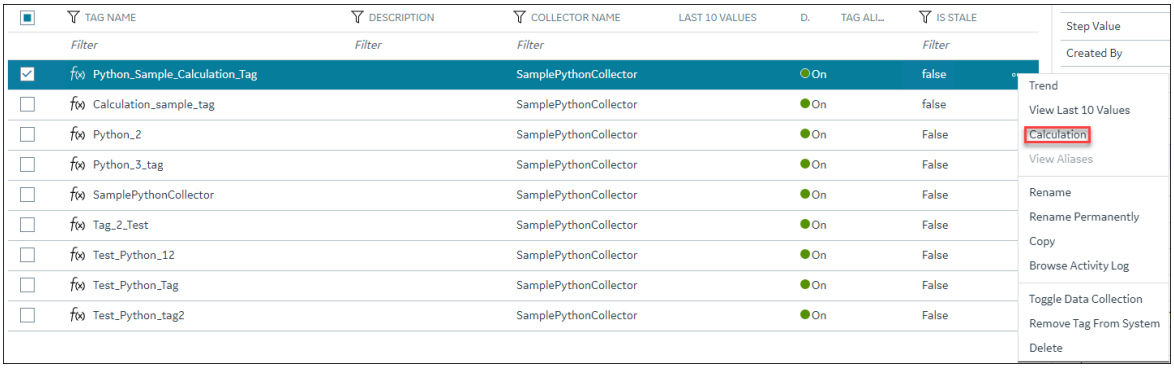
```
C:\Users\Administrator\AppData\Local\Programs\Python\Python38-32\Lib
```

```
C:\Users\Administrator\AppData\Local\Programs\Python\Python38-32\Lib\site-packages
```

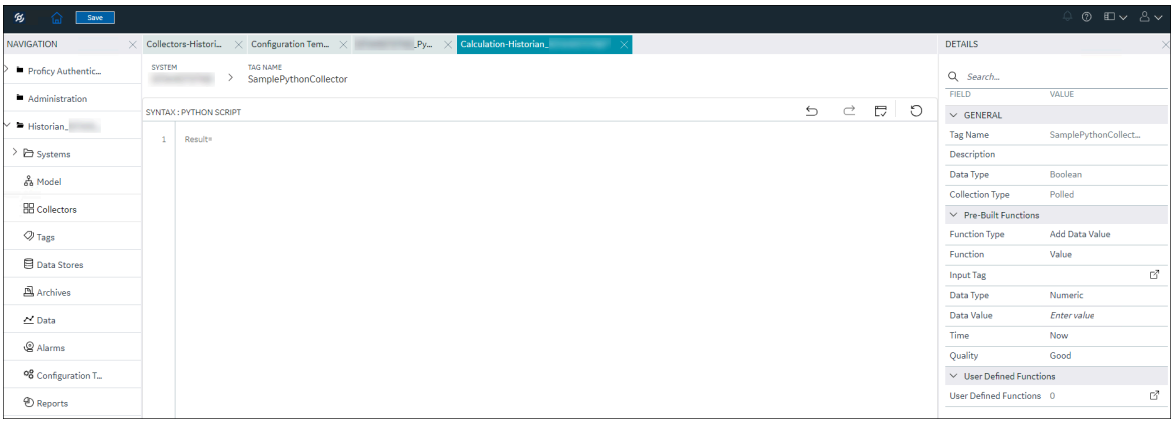
You can also configure Python library path using Configuration Hub. For more information, refer to Configure Python Library path using Configuration Hub.

3. Access Configuration Hub *(on page 336)*
4. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.

   A list of tags appears.
5. Select a Python collector tag, right-click or select more options, and then select **Calculation**.




The **PYTHON SCRIPT** editor appears.




6. In the scripting section, remove `Null` (and retain the `Result =`).

7. In the scripting section, to import the custom or third-party functions/modules, enter the script in the following format:

```
import mymodule

Result=mymodule.myfunction()

#Example

import SimulationCalculation

Result=SimulationCalculation.GetInteger()
```

8. To test the function, select .
   A message appears, stating whether the syntax is correct.
9. In the upper-left corner of the page, select **Save**.
   The Python script is created.

   After few seconds, the added script takes effect.

## Available Functions

This topic describes the built-in functions that you can use to create a calculation formula. You can also use third-party or custom Python modules in the script *(on page 2481)*.

| Function Type | Function | Description | Example |
|---|---|---|---|
| Add Data | `Historian.AddData(tags, values,times)` | Write data to a tag in the Historian archive. | `Result=Historian.AddData ('Python_Sample_Calculation',0,'2023-12-13T15:13:44.536Z',100)` |
| Check Data Quality | `Historian.CurrentQuality('tag name')` | The current quality of the tag (0 for bad quality and 100 for good quality). | `Result=Historian.CurrentQuality('Tag')` |
| Check Data Quality | `Historian.InterpolatedQuality('tag name','time')` | The current quality of the interpolated tag. | `Result=Historian.InterpolatedQuality('Tag1','2023-12-13T16:32:03.626Z')` |
| Check Data Quality | `Historian.NextGoodQuality('tag name', 'time')` | The good quality of the raw sample after the time. Time is provided | `Result=Historian.NextGoodQuality('Tag1','2023-12-13T16:34:19.154Z')` |

| Function Type | Function | Description | Example |
|---|---|---|---|
| | | as string in standard ISO 8601 format. | |
| Check Data Quality | `Historian.NextQuality('tag name','time')` | The quality of the tag (0 for bad quality and 100 for good quality) after the time. Time is provided as string in standard ISO 8601 format. | `Result=Historian.NextQuality('Tag2','2023-12-13T16:35:41.709Z')` |
| Check Data Quality | `Historian.PreviousGoodQuality('tag name','time')` | The good quality of the raw sample prior to the time. Time is provided as string in standard ISO 8601 format. | `Result=Historian.PreviousGoodQuality('Tag3','2023-12-13T16:36:47.736Z')` |
| Check Data Quality | `Historian.PreviousQuality('tag name','time')` | The quality of the tag (0 for bad quality and 100 for good quality) prior to the time specified. Time is provided as string in standard ISO 8601 format. | `datetime_str = Historian.CurrentTime()`<br>`datetime_object = datetime.strptime(datetime_str, '%m/%d/%y %H:%M:%S.%f')`<br>`Result=Historian.PreviousQuality('Test_Sim.Simulation00001',datetime_object.isoformat())` |
| Insert A Calculation-Non-filtered Calculation | `Historian.Calculation('tag name', 'CalculationMode', 'StartTime', 'EndTime')` | Unfiltered calculated data query that returns a single value, similar to the Excel Add-In feature. For a list of the calculation mode, refer to calculation modes *(on page 1073)*. All times are provided as string in standard ISO 8601 format. | `from datetime import datetime, timedelta`<br>`datetime_str = Historian.CurrentTime()`<br>`endTime = datetime.strptime(datetime_str, '%m/%d/%y %H:%M:%S.%f')`<br>`startTime = endTime - timedelta(hours=0,minutes=0,seconds=10)` |

| Function Type | Function | Description | Example |
|---|---|---|---|
| | | | `Result=Historian.Calculation('Test_Sim.Simulation00002','Average', startTime.isoformat(), endTime.isoformat())` |
| Insert A Calculation- Filtered | `Historian. CalculationFilter('tag name', 'CalculationMode', 'StartTime', 'EndTime', 'FilterTagname', 'FilterMode', 'FilterComparison', 'FilterValue')` | Filtered calculated data query that returns a single value, similar to the Excel Add-In feature. All times are provided as string in standard ISO 8601 format. | `from datetime import datetime, timedelta datetime_str = Historian.CurrentTime() endTime = datetime.strptime(datetime_str, '%m/%d/%y %H:%M:%S.%f') startTime = endTime - timedelta(hours=0, minutes=0,seconds=1) Result=Historian.CalculationFilter('Test_Sim.Simulation00004','Average', startTime.isoformat(), endTime.isoformat(),'Test_Sim.Simulation00006','AfterTime','GreaterThan','100')` |
| Insert A Calculation Quality- Non-filtered Calculation | `Historian.CalculationQuality('tag name', 'CalculationMode', 'StartTime', 'EndTime')` | Unfiltered calculated data query that returns the quality of resulting value. For a list of the calculation mode, refer to calculation modes *(on page 1073)*. All times are provided as string in standard ISO 8601 format. | `Result=Historian.CalculationQuality('Python_2','Average','2023-12-13T16:54:27.296Z','2023-12-13T16:54:28.296Z')` |

| Function Type | Function | Description | Example |
|---|---|---|---|
| Insert A Calculation Quality- Filtered Calculation | `Historian.CalculationFilterQuality('tag name', 'CalculationMode', 'StartTime', 'EndTime', 'FilterTagname', 'FilterMode', 'FilterComparison', 'FilterValue')` | Filtered calculated data query that returns the quality of value. All times are provided as string in standard ISO 8601 format. | `Result=Historian.CalculationFilterQuality('Tag','Average','2023-12-13T16:56:08.123Z','2023-12-13T16:56:09.123Z','Tag','AfterTime','Equal','')` |
| Insert A Tagname | `'Tag'` | The selected tag name. | `Result='Python_3_tag'` |
| Insert A Timestamp | `Historian.CurrentTime()` | The calculation execution time, which becomes the timestamp of the stored value. The result is returned as a string.<br><br>For real-time processing of polled tags, the calculation execution time is the time when the calculation is triggered. For unsolicited tags, the calculation execution time is the timestamp delivered with the subscription.<br><br>**Note:** When a calculation is performed, the timestamp of the result is the time that the calculation has | `The result is returned as a string.`<br>`Format of timestamp is: "%m/%d/%y %H:%M:%S.%f"`<br>`Example: “04/11/23 22:09:40.000”`<br><br>`#Example to convert string timestamp to Python datetime object:`<br><br>`from datetime import datetime`<br>`datetime_str = Historian.CurrentTime()`<br>`datetime_object = datetime.strptime(datetime_str, '%m/%d/%y %H:%M:%S.%f')` |

| Function Type | Function | Description | Example |
|---|---|---|---|
| | | begun, not the time that it completed.<br><br>For recovery of polled or unsolicited tags, the calculation execution time is the time when the calculation would have been performed if the collector were running. | |
| Insert A Timestamp | `Historian.NextGoodTime('tag name', 'time')` | The timestamp of the good raw sample after the time. Time is provided as string in standard ISO 8601 format. | `Result=Historian.NextGoodTime('Tag',2023-12-13T17:54:44.223Z)` |
| Insert A Timestamp | `Historian.NextTime('tag name', 'time')` | The timestamp of the raw sample after the timestamp. Time is provided as string in standard ISO 8601 format. | `Result=Historian.NextTime('Tag','2023-12-13T17:56:21.726Z')` |
| Insert A Timestamp | `Historian.PreviousGoodTime('tag name', 'time')` | The timestamp of the latest good quality of the raw sample prior to the time. Time is provided as string in standard ISO 8601 format. | `Result=Historian.PreviousGoodTime('Tag','2023-12-13T17:57:45.308Z')` |
| Insert A Timestamp | `Historian.PreviousTime('tag name', 'time')` | The timestamp of the raw sample prior to the time. Time is provided as string in standard ISO 8601 format. | `Result=Historian.PreviousTime('Test_Sim.Simulation00001','2023-03-27T16:02:08.070557')` |

| Function Type | Function | Description | Example |
|---|---|---|---|
| Insert A Timestamp | `'time shortcut'` | The timestamp.value as a string format, in ISO 8601 format. | `Result='2023-12-13T18:00:38.186Z'` |
| Insert A Value | `Historian.CurrentValue('tag name')` | The value of the tag, interpolated to the calculation execution time. The `CurrentValue` function returns 0 if the quality is 0 (bad quality). This occurs if you initialized it to 0, or if a previous call failed. | `Result=Historian.CurrentValue('Test_Sim.Simulation00001')` |
| Insert A Value | `Historian.InterpolatedValue('tag name', 'time')` | The tag value, interpolated to the time that you enter. Time is provided as string in standard ISO 8601 format. | `Result=Historian.InterpolatedValue('Tag', '2023-12-13T18:05:29.367Z')'` |
| Insert A Value | `Historian.NextGoodValue('tag name', 'time')` | The value of the good raw sample after the time. Time is provided as string in standard ISO 8601 format. | `Result=Historian.NextGoodValue('Tag', '2023-12-13T18:06:41.622Z')` |
| Insert A Value | `Historian.NextValue('tag name', 'time')` | The value of the raw sample after the timestamp. Time is provided as string in standard ISO 8601 format. | `Result=Historian.NextValue('Tag', '2023-12-13T18:07:44.369Z')` |
| Insert A Value | `Historian.PreviousGoodValue('tag name', 'time')` | The latest good value of the raw sample prior to the time. Time is provided as string in standard ISO 8601 format. | `from datetime import datetime, timedelta`<br>`from time import time`<br><br>`# Getting today's date and time` |

| Function Type | Function | Description | Example |
|---|---|---|---|
| | | | todays_Date = datetime.now()<br><br># subtract 1 minute from current time<br>timeToQuery = todays_Date - timedelta(hours=0,minutes=1,seconds=0)<br>Result=Historian.PreviousGoodValue('Test_Sim.Simulation00001',timeToQuery.isoformat()) |
| Insert A Value | `Historian.PreviousValue(‘tag name’, ‘time’)` | The tag value of the raw sample prior to the time specified.<br><br>Time is provided as string in standard ISO 8601 format. | datetime_str = Historian.CurrentTime()<br>datetime_object = datetime.strptime(datetime_str, '%m/%d/%y %H:%M:%S.%f')<br>Result=Historian.PreviousValue('Test_Sim.Simulation00001',datetime_object.isoformat()) |
| Insert An Advanced Calculation- Non-filtered Calculation | `Historian.AdvancedCalculation('Tagname', 'CalculationMode', ‘CriteriaString’ 'StartTime', 'EndTime')` | Unfiltered calculated data query with criteria string that returns a single value. All times are provided as string in standard ISO 8601 format. | `Result=Historian.AdvancedCalculation('Tag','Average','','2023-12-13T18:15:22.189Z','2023-12-13T17:15:23.189Z')` |
| Insert An Advanced Calculation- Filtered Calculation | `Historian.AdvancedCalculationFilter ('Tagname', 'CalculationMode', ‘Criteria-` | Filtered calculated data query with criteria string that returns a single value. All times are provid- | `Result=Historian.AdvancedCalculationFilter('Tag','Average','','2023-12-13T18:16:29-` |

| Function Type | Function | Description | Example |
|---|---|---|---|
| | `String', 'StartTime', 'EndTime', 'FilterTagname', 'FilterMode', 'FilterComparison', 'FilterValue')` | ed as string in standard ISO 8601 format. | `.333Z','2023-12-13T18:16:30-.333Z','Tag','AfterTime','Equal','')` |
| Insert An Advanced Calculation Quality- Non-filtered Calculation | `Historian.AdvancedCalculationQuality('Tagname', 'CalculationMode', 'CriteriaString' 'StartTime', 'EndTime')` | Unfiltered calculated data query with criteria string that returns the quality of value. All times are provided as string in standard ISO 8601 format. | `Result=Historian.AdvancedCalculationQuality('Tag','Average','','2023-12-13T18:18:32-.238Z','2023-12-13T18:18:33-.238Z')` |
| Insert An Advanced Calculation Quality- Filtered Calculation | `Historian.AdvancedCalculationFilterQuality ('Tagname', 'CalculationMode', 'CriteriaString', 'StartTime', 'EndTime', 'FilterTagname', 'FilterMode', 'FilterComparison', 'FilterValue')` | Filtered calculated data query with criteria string that returns the quality of value. All times are provided as string in standard ISO 8601 format. | `Result=Historian.AdvancedCalculationFilterQuality('Tag','Average','','2023-12-13T18:20:19-.954Z','2023-12-13T18:20:20-.954Z','Tag','AfterTime','Equal','')` |
| NA | `Historian.LogMessage(string_message)` | Allows you to write messages to the collector log file for debugging purposes. The collector log files are located in the `Proficy Historian Data\LogFiles\LogFiles` folder. | `Result=Historian.LogMessage(test_message)` |

# Examples of using Calculation Functions

## Examples: Using the Built-in Functions

### Retrieving the Current value

You can use the current value function to retrieve the current value of an existing Historian tag.

```
Result=Historian.CurrentValue('Tag1’)
```

Get the current value of two tags and return the greater value of the two value:

### Return the Greater Value of Two Tags

```
x=Historian.CurrentValue('Tag1')

y=Historian.CurrentValue('Tag2')

if x > y:

 Result= x

else:

 Result= y
```

### Add the Values of Two Tags

```
Result=Historian.CurrentValue(‘Tag1')+Historian.CurrentValue('Tag2')
```

## Examples: Custom or Third-party Python Modules

To use additional Python modules, install them over Python 3.8.

For example, to install NumPy:

1. Run the following command:

   ```
   Pip install numpy
   ```

2. Add the site packages to the Python path as follows:

   ```
   Computer\HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\GE

    Digital\iHistorian\CollectorServiceExtensions\PythonExpressions\PythonPath
   ```

   After you do so, the changes are reflected as follows:

   ```
   C:\Program Files (x86)\GE Digital\Historian Python Expressions\Python38\lib
   ```

   ```
   C:\Program Files (x86)\GE Digital\Historian Python Expressions\Python38\user
   ```

```
C:\Users\Administrator\AppData\Local\Programs\Python\Python38-32\Lib
```

```
C:\Users\Administrator\AppData\Local\Programs\Python\Python38-32\Lib\site-packages
```

## Retrieve the Age of the First Person from a List Using a REST API

```
import requests

import json

api_url = https://myRestAPIURL/

response = requests.get(api_url)

json_data = json.loads(response.text)


Results=json_data["results"][0]

dateOfBirth =Results["dateofbirth"]

age = dateOfBirth["age"]

Result=age
```

Returns the following result:

```
{
    "results": [
        {
            "gender": "female",
            "name": {
                "title": "Mrs",
                "first": "Lumi",
                "last": "Tikkanen"
            },
            "dateofbirth": {
                "date": "1982-01-08T21:23:26.095Z",
                "age": 39
            },
            "phone": "08-609-184",
            "cell": "049-127-63-22"
        }
    ]
}
```

### Calculate the Sum of all Values in a Column in an SQL Database

```
import pyodbc


conn = pyodbc.connect('Driver={ODBC Driver 17 for SQL Server};'
                      'Server=MySQLServer;'
                      'Database=MyDB;'
                      'Trusted_Connection=yes;')


cursor = conn.cursor()
cursor.execute('SELECT column1 FROM Table_1')


sum = 0;
for i in cursor:
 sum = sum + i.column1


Result = sum
conn.close()
```

### Calculate a Score Using Linear Regression in NumPy

```
import numpy as np
from sklearn.linear_model import LinearRegression


X = np.array([[2, 7], [10, 4], [5, 7], [2, 3]])
y = np.dot(X, np.array([1, 2])) + 3
reg = LinearRegression().fit(X, y)
Result = reg.score(X, y)
```

### Reading Data from a File Using Pandas

To read the data using Pandas, you must create:

- Training dataset using 80% of the data
- Linear regression model using the training data
- Return the coefficient of the model

```
import pandas as pd
import numpy as np
from sklearn import linear_model
```

```
data = pd.read_csv("C:\\myFile.csv")
data = data[["Column1","Column2"]]
train = data[:(int((len(data)*0.8)))]

regressionLine = linear_model.LinearRegression()

train_x = np.array(train[["Column1"]])
train_y = np.array(train[["Column2"]])

regressionLine.fit(train_x,train_y)

Result=regressionLine.coef_[0][0]
```

## Mathematical Optimization

The Rosenbrock function to perform mathematical optimization is defined in SciPy as follows:

`sum(100.0*(x[1:] - x[:-1]**2.0)**2.0 + (1 - x[:-1])**2.0)`

```
import numpy as np
from scipy.optimize import rosen

a = 0.2 * np.arange(9)
Result=rosen(a)
```

## Calculating the Determinant of a Matrix using SciPy

```
#calculate the determinant of a square matrix
import numpy as np
from scipy import linalg

A = np.array([[1,2,4], [4,3,7], [2,7,3]])
Result=linalg.det(A)
```

## Creating a Data Frame from an Array and Calculating the Sum of all Elements

```
import numpy as np
import pandas as pd

df = pd.DataFrame(np.array([[1, 2, 3], [4, 5, 6]]))
Result=float(df.sum().sum())
```

### Generating a Random Value

```
from random import seed

from random import random


seed(10)

Result=random()*random()
```

## Example: Storing Current Values of Arrays

You can store the current value of array of strings into another array tag. You can use the below calculation/script, in this example, the CurrentValue function returns the result as a python list that will be stored into the appropriate historian array type:

```
Result = Historian.CurrentValue('Test_Sim.SimulationArrayString')
```

**Note:**
When you convert a value to boolean type, you must store the results as `Result = bool(Historian.CurrentValue('TP.Simulation00001'))`

You can take an array value collected from a field device and adjust the values before storing it in another array tag Array2 using the below calculation/script:

```
x= Historian.CurrentValue("Array1")

x(1) = x(1)+10

Result = x
```

You can use Calculation() function to read the array tag as shown below:

```
from datetime import datetime, timedelta

datetime_str = Historian.CurrentTime()

endTime = datetime.strptime(datetime_str, '%m/%d/%y %H:%M:%S.%f')

startTime = endTime - timedelta(hours=0,minutes=0,seconds=1)

Result=Historian.Calculation('Test_Sim.SimulationArray00001','Average', startTime.isoformat(), endTime.isoformat())
```

**Note:**
When you import numpy values, it is recommended to include the string data type, failing to include will cause the code to incorrectly read the numpy values. The below code will fail with

error: Msg="Unable to set result for Result" if the tag using this script is set as an array of integers. Testing the function will pass but running the calculation will fail.

```
import numpy as np

randnums= np.random.randint(1,101,5)

Result=randnums
```

## Example: Storing Dictionary Data as Multifield Data

Dictionary data type in Python (key/value pairs) can be stored in Historian as a Multifield (User Defined Type).

**Note:**
If you want to store the data quality of multifield elements, store the result into a Historian tag that matches the data type of the field. Copying a multifield into a dictionary type only stores the field name and data.

Maximum array elements that can be copied to python tuple are 10. If more elements are available in the source in Historian, read each element in your python calculation.

For example, create a Multifield called “MySample” in Historian with three fields named: Field1, Field2, Field3. Once it is created, add a Python tag with the same multifield data type, and use below script to update its values:

```
thisdict = {

  "Field1": 11.55,

  "Field2": 22.8,

  "Field3": 33.55,

}

Result = thisdict
```

OR

```
thisdict = {

  "Field1": "1",

  "Field2": "2",

  "Field3": "3",

}

Result = thisdict
```

Using the value of a field within the python script, if you have a user-defined type MySample with fields Field1 and Field2, you can create Tag1 and use the value of one field in the Python Tag. The destination tag is not a multifield tag.

```
sampleDict = Historian.CurrentValue('TestMF')

Result = sampleDict['Field3']

OR

Result = Historian.CurrentValue('TestMF.Field1')+5

OR

thisdict = {

  "brand": "BrandNew",

  "model": "BrandModel",

  "year": 1964

}

Result = thisdict["year"]
```

**Storing the current value of a multifield tag into another tag whose data type is the same user defined multifield type**

Below example stores the current value of a multifield tag into another tag whose data type is same as the user defined multifield type. CurrentValue function returns the result as a python dictionary (key value pairs of string data type).

```
Result = Historian.CurrentValue('TestMFString')
```

## Example: Storing Python Integer List in Historian

Python lists and tuples can be stored in Historian as arrays of either numeric or string data types. Historian creates the arrays as python lists

Maximum array elements copies to python list is 10. If more elements are in the source in historian, read each element in your python calculation.

To store python integer list in historian, create a Single Integer array type and use the below calculation/ script:

```
import array as arr

a = arr.array('i', [2, 4, 6, 8])

Result = a
```

OR

```
tuple = (11,22,33)

Result = tuple
```

## Example: Storing Python String List in Historian

Python lists and tuples can be stored in Historian as arrays of either numeric or string data types. Historian creates the arrays as python lists

Maximum array elements copies to python list is 10. If more elements are in the source in historian, read each element in your python calculation.

to store python string list in historian, create a variable string tag of array type and use the below calculation/script:

```
mylist = ["gauge", "valve", "rotor"]

Result = mylist
```

OR

```
thistuple = ('Motor', 'Shaft', 'Rotor', 'Stator')

Result = thistuple
```

## Example: Use Historian Data as Input to a Python Script

You can use the Historian Array data as input to a python script, to use the Historian Array data as input, you can use the name of the array tag like "Array1" or the individual element of the array like "Array1[4]". For example, if you have an array tag "Array1" of floating point values and a calculation tag "FloatCalc1" of float data type, then you can use the array as input to calculate a float value.

```
Result=Historian.CurrentValue("Array1[4]")+5
```

# Chapter 33. Server-to-Server Collector

## Overview

### Overview of the Server-to-Server Collector

The Historian Server-to-Server collector allows you to collect data and messages from a source Historian server to a destination Historian server, a Predix Time Series instance, an Azure IoT HUb instance, or an MQTT endpoint such as AWS IoT Core. The Server-to-Server collector includes many of the features of the Calculation collector. The primary difference is that the Server-to-Server collector stores the result in a destination tag on the destination server, whereas the Calculation collector reads and writes to the same server.

The Server-to-Server collector can also run as a stand-alone component where both the source and destination Historian databases are on remote machines.

When a time-based or an event-based trigger of a destination tag occurs:

1. The calculation formula for the destination tag is executed.

   This typically involves fetching data from one or more tags on the source server.
2. A raw sample or calculation error is determined.

   You can use conditional logic in your calculation formula to determine if a sample should be sent to the destination.
3. The raw sample is delivered to the destination server, utilizing store and forward when necessary.

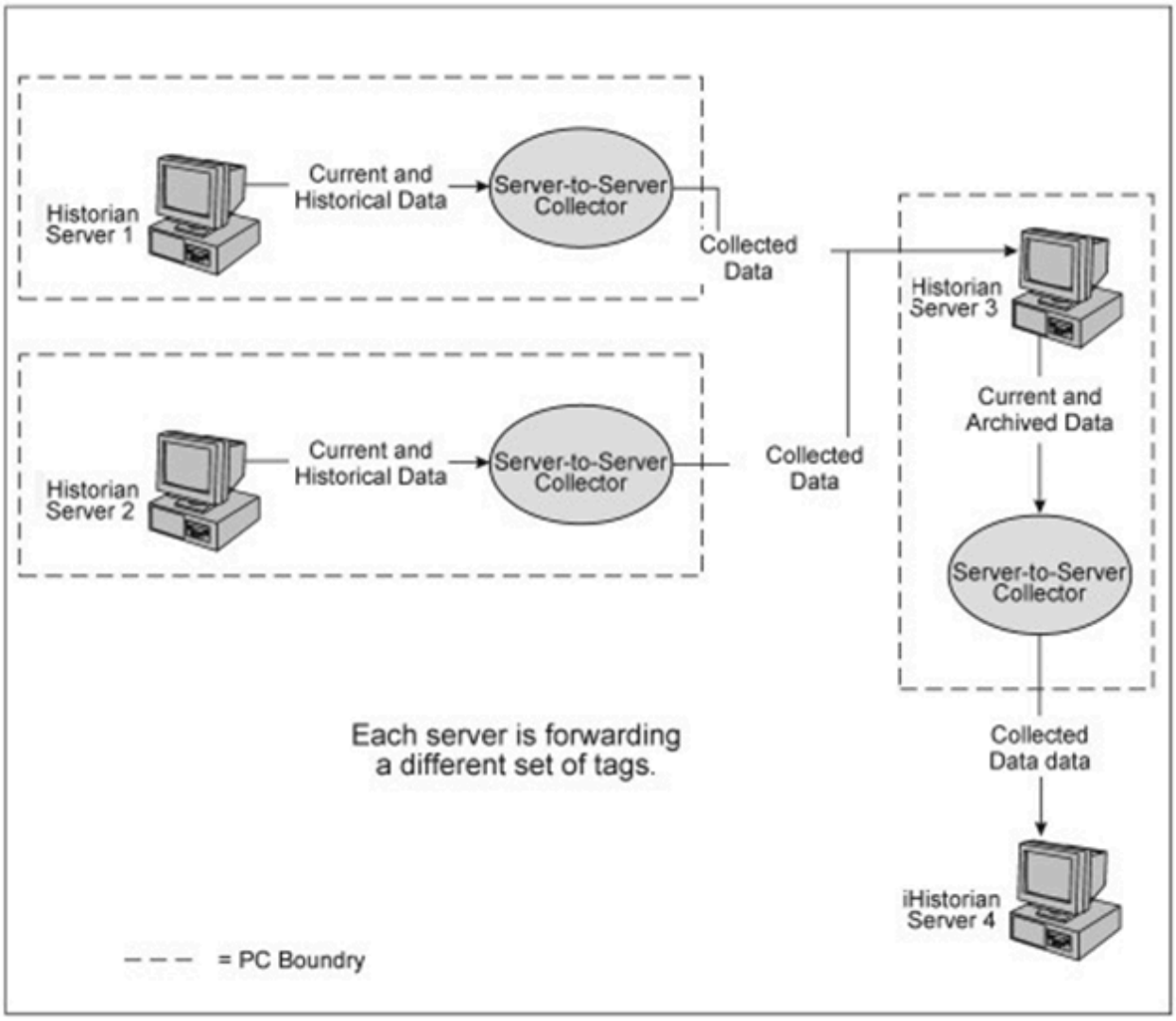
Message replication, if enabled, is event-based. Messages and alerts are sent to the destination server as they happen.

The destination tag is fundamentally a different tag than the source tag. Therefore:

- When a tag is added by browsing, only certain tag properties are copied from the source tag to the destination tag. Consider what properties are necessary for your application and configure them manually. For information on which properties are copied, refer to Tag Properties that are Copied *(on page 2508)*.
- If you change a tag property on the source tag (EGU Limits, descriptions, and so on), the property does not automatically change on the destination tag. You can manually change the properties of a destination tag.

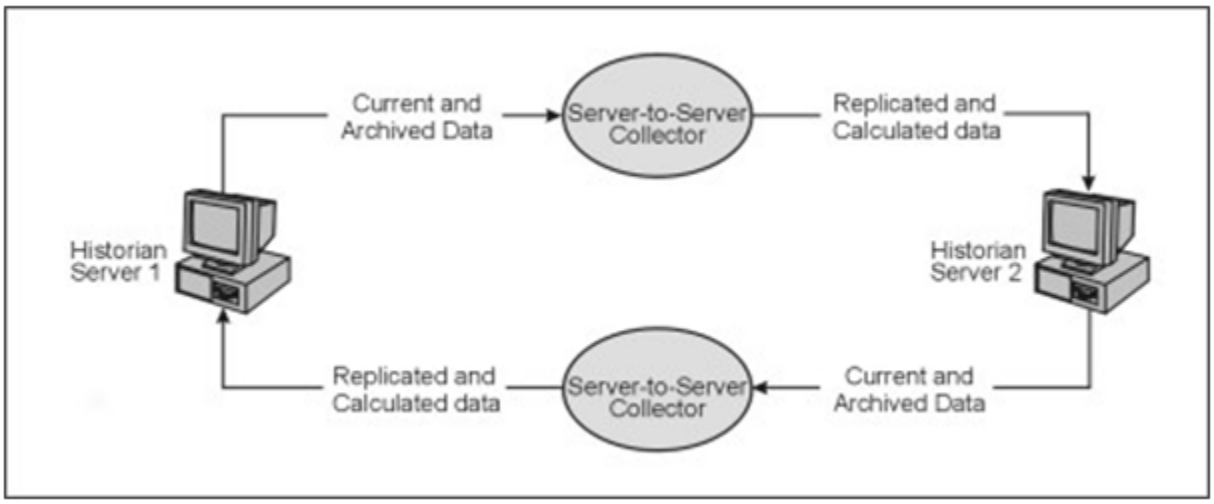
**Data Flow in Multiple Server-to-Server Collectors**

The following image shows that you can use multiple Server-to-Server collectors in an application to pass data from multiple nodes to one node and that a server can be a source and a destination at the same time. Each Historian server is forwarding a different set of tags.




**Data Flow in Bi-directional Server-to-Server Collectors**

You can configure bi-directional data collection, where each collector collects a different set of tags. The following figure shows bi-directional server-to-server data collection.




**Note:**
You cannot collect the same tag in both directions. This is not a way to perform bi-directional synchronization of a tag.

**Features**

| Feature | Capability |
|---|---|
| Browse Source for Tags | Yes |
| Browse Source for Tag Attributes | Yes |
| Polled Collection | No |
| Minimum Poll Interval | No |
| Unsolicited Collection | Yes |
| Timestamp Resolution | Yes - 100 milliseconds |
| Data Compression | Yes |
| Accept Device Timestamps | Yes |
| Floating Data Point | Yes |
| Integer Data | Yes |
| String Data | Yes |
| Binary Data | No |
| Allows VB scripting | No |

| Feature | Capability |
|---|---|
| Python Expression Tags | No |

**Licensing**

When the destination server is Proficy Historian, the Server-to-Server collector requires licensing on the destination machine. This means that the destination Historian server must be licensed for the Historian Enterprise edition, have the Enterprise Collectors option licensed, or be a Historian Edge server. It is similar to any other collector. The destination machine will have the Server-to-Server listed in its collector list.

**Interface Name**

Historian uses the following naming convention for the Server-to-Server collector interface name:

```
<source Historian server>_To_<destination Historian server>
```

**Best Practices**

- We recommend that you install the Server-to-Server collector on the source Historian machine. When you do so, the collector can preserve the collected data (store and forward) even if the collector and the destination server become disconnected.
- Collection on a tag-by-tag basis is preferred, according to scheduled poll times or upon data changes. One sample is collected for each trigger.
- The Server-to-Server collector can perform calculations on multiple input tags as long as the input tags are on the same source Historian.
- Use polled triggers to perform scheduled data transformations like daily or hourly averages. Use unsolicited triggers to replicate data in real time, as it changes.
- Use event-based triggers to replicate data throughout the day. The samples can be held ingoing an outgoing store and forward buffer when necessary. You cannot schedule batch replication of raw samples. For example, you cannot, at the end of the day, send all raw samples for tags to the destination.
- All input source tags for the calculations must originate from the source archiver. For instance, you cannot directly add a tag from `server1` plus a tag from `server2` and place the result on `server2`. You could, however, collect tags from `server1` to `server2`, and then use the Calculation collector or the Server-to-Server collector to accomplish this. This requires two Server-to-Server collectors, one running on each machine. You could also use the Historian OLE DB provider.

**Limitations**

- If you enable alarm replication, the alarm data is sent to the destination server. However, alarm filtering is not available in the Server-to-Server collector.
- You cannot configure bi-directional message replication.

## About Recovery Mode

Normally, the Server-to-Server collector operates in a real-time mode. A real-time mode is when the collector is polling data or has subscribed to events and triggers calculations based on these events occurring in real-time. Messages are also sent as they occur. Recovery mode allows you to recover tag and alarm data when the connection between the collector and the source server is re-established. After a connection loss, the configuration settings *(on page 2506)* for the Server-to-Server collector determine how much tag and alarm data is recovered and if messages are included in the recovery.

**When Does Recovery Occur?**

Recovery mode executes:

- When the collector is started.
- When the collector is resumed after a pause.
- When there is an on-the-fly change (similar to a pause and resume). Only tags in the new tag configuration are recovered.
- When there has not been a collector stop and start, but the connection to the source Historian is restored.

**What Happens When Recovery Occurs?**

In recovery mode, after connecting to the source Historian, the collector will:

- Set up subscriptions for all alarms and trigger tags.
- Perform recovery in chronological order (oldest to newest).
- Perform message recovery, if enabled.
- Begin polling and processing subscriptions in real-time mode.

The following items are recovered:

- **Event-based tags:** This includes the data from the last write time until now. The system retrieves all tags.
- **Messages:** The system checks for new messages and verifies errors. Once the system verifies a connection to the destination, it sends the messages one at a time.
- **Alarm data:** This includes all alarm data from the last write time until now.

**Note:**
Alarm recovery uses a different write time than tag recovery. Alarm recovery starts from the time of the last alarm is replicated to the destination.

If your formula contains tags not in the trigger list or dependencies exist among tags (for example, if a calculation tag is a trigger for another calculation tag), you might not recover all data.

## About Collection of Raw Samples

To minimize the effect of missing samples, we recommend that you view collected data on the destination with interpolated queries rather than raw data queries.

Here are some suggestions about how best to configure your system when you want raw samples of collected tags to match on the source and the destination. This is often not achievable, but here are some tips:

- Use the formula `Result=CurrentValue("TriggerTag")`.
- Do not use collector compression on the destination tag.
- Use archive compression on the destination if it is set on the source.

  The reason for this is that unsolicited triggers occur based on value changes, not based on what is stored in the archive. A value change may not be stored on the destination if archive compression is being used. It is up to the destination tag to apply the archive compression before the value is stored.
- Use event-based triggers with `0 ms` collection intervals.
- In the Server-to-Server collector, disable the **Synchronize Timestamps to Server** option in the **Advanced** section in the collector configuration in Historian Administrator.

# Using the Collector

## Workflow for Using the Server-to-Server Collector

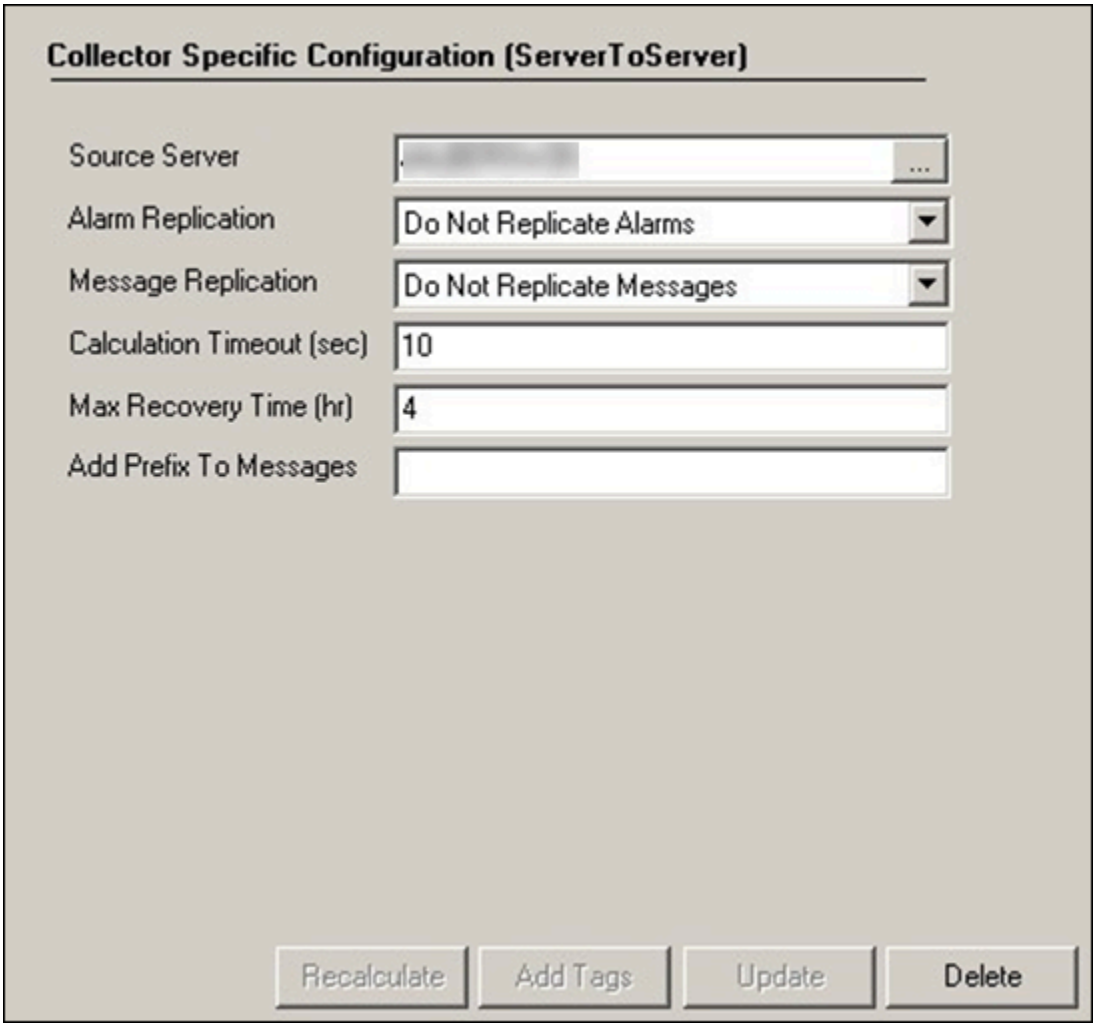
To use the Server-to-Server collector, you must perform the following tasks:

| Number | Task | Notes |
|---|---|---|
| 1 | Install the collectors *(on page 142)* on the machine on which you want to run the collector. | This step is required. This will place the collector binaries on the machine. |
| 2 | Install Remote Management Agents *(on page 187)*. | This step is required to manage collectors installed on a remote machine. |
| 3 | Add an instance *(on page 357)* of the Server-to-Server collector using Configuration Hub. | This step is required. |
| 4 | Configure the Server-to-Server collector *(on page 2506)* using Historian Administrator. | This step is required only if you want to change the default values. |
| 5 | Create a destination tag. You can do so by browsing for the tag *(on page 2064)*, adding it manually *(on page 2056)*, or copying a tag *(on page 2060)*. | This step is required. If configured, the tag will contain a prefix. |
| 6 | Assign a trigger to the tag that you have created, similar to assigning a trigger for the Calculation collector. | This step is optional. The trigger can be scheduled (polled) or unsolicited (event-based). When you use an event-based trigger, you must also set up a dependency list of one or more tags. For more information, refer to Create a Polled Trigger *(on page 2112)* and Create an Unsolicited Trigger *(on page 2114)*. |
| 7 | Create a formula similar to creating a formula for the Calculation collector. For instructions, refer to About Calculation Formulas *(on page 2118)*. | This step is optional.<br><br>You can expect thousands of tags per second to be processed, depending on your calculation formula. However, the destination server has a limited number |

| Number | Task | Notes |
|---|---|---|
| | | of incoming events per second, which is shared by all the collectors.<br><br>For instance, for a polled collection, you can expect to perform hundreds of calculations per second. Polled calculations, using calculated data functions, are slower than unsolicited calculations using the CurrentValue() function. |

## Configure the Server-to-Server Collector Instance

1. Access Historian Administrator.
2. Select the Server-to-Server collector from the list of collectors, and then select **Configuration**.
   The **Collector Specific Configuration (ServerToServer)** section appears.




3. Provide values as specified in the following table, and then select **Update**.

| Field | Description |
|---|---|
| **Alarm Replication** | Indicates whether you want to enable or disable alarm replication. If you enable alarm replication, all collected alarm data will be transferred from the source server to the destination server. If you enable alarm replication, you also enable alarm recovery. However, if you set the **Max Recovery Time** value to zero, alarm recovery does not happen. |
| **Message Replication** | Indicates whether you to want to enable or disable message replication. If you enable message replication, messages will be transferred from the source server to the destination server. You can use this data for audits. If you enable message replication, you also enable message recov- |

| Field | Description |
|---|---|
| | ery. However, if you set the **Max Recovery Time** value to zero, message recovery does not happen. |
| **Calculation Timeout (sec)** | The maximum time allowed for a tag's calculation formula to execute before being terminated. The default value is 10 seconds. |
| **Max Recovery Time (hr)** | The maximum duration, in hours, for which the collector will attempt to restore data during recovery logic. The default value is 4 hours. |
| **Add Prefix to Messages** | The prefix to identify replicated messages on the destination.<br>Alarms and events data will automatically have a prefix added to it with the following syntax:<br>`MachineName_Datasource`<br>For example, if your alarm is forwarded from the server `Almserver12` with a data source named `OPCAE`, the prefix will be `Almserver12_OPCAE`. |

The Server-to-Server collector is configured.

# Tag Properties that are Copied

## Tag Properties that are Copied

When you add a tag by choosing from the S2S Collector browse list, only certain tag properties are copied from the source tag to the destination tag. If you intend to copy raw samples from the source to the destination, after you add the tag, be sure to set these properties to their desired values. See *Tag Properties Copied to the Destination Tag* described below.

Important tag properties that do not automatically copy over when you add the tag include:

- Input scaling settings

  Since the output of the source tag is the input to the destination tag, you actually want to match the EGU limits on the source to input limits on the destination, if you are using Input Scaling.

- Timestamp resolution

Make sure that the timestamp resolution properties match. For example, do not use the second timestamp resolution on the destination tag, if your source tag uses millisecond timestamp resolution. If your source tag uses millisecond timestamp resolution, then you also want to set your destination tag to also use millisecond timestamp resolution.

The following table describes the tag properties in Historian Administrator Tags page that are copied when the destination tag is created via select from the browse. If a property is not listed in this table, it is not copied.

| Tab Name | Properties Copied |
|---|---|
| General | Description<br>EGUDescription |
| Collection | Data Type<br>DataLength |
| Scaling0 | HiEGU<br>LoEGU<br>InputScaling<br>HiScale<br>LoScale |
| Compres-sion | ArchiveCompres-sion<br>ArchiveDead-band(%) |

# Examples of Data Collection

## Raw Samples Collection Example

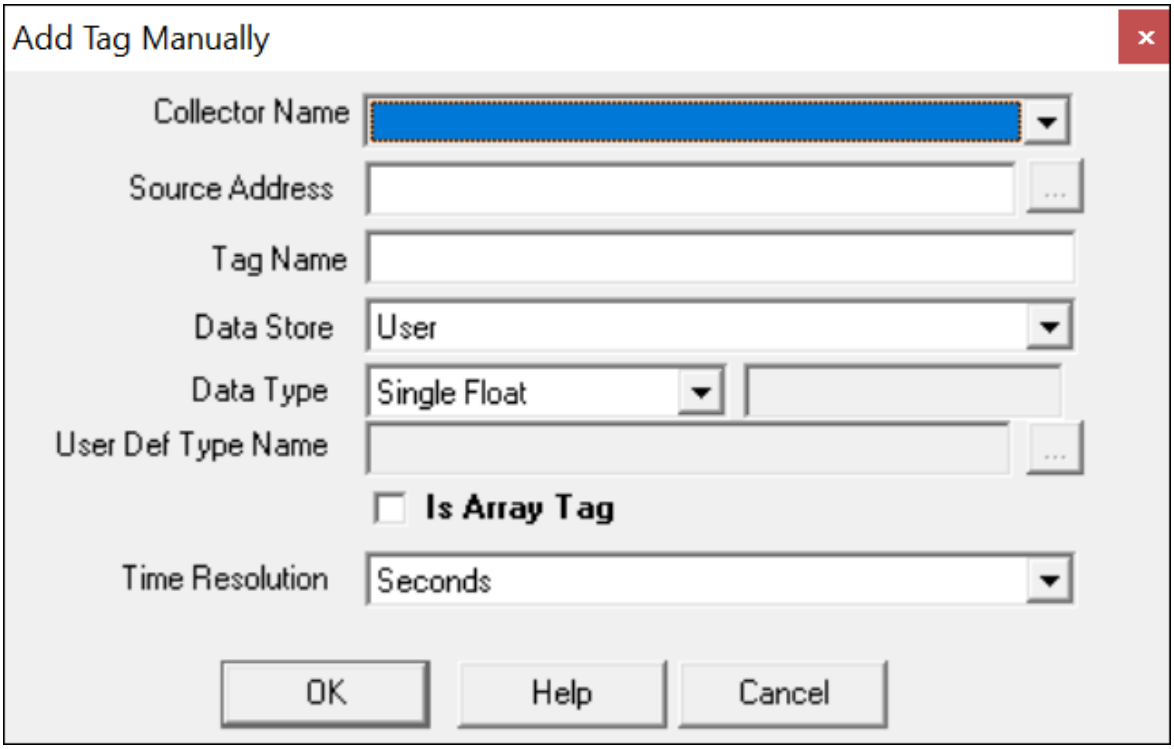
This topic describes how to collect raw samples using the Server-to-Server collector. The tagnames are the same on the source and destination. In this example, you add a tag manually, and give it a tagname on the destination server (representing the meaning of the calculated value).

1. Using Historian Administrator, browse the Server-to-Server collector for tags *(on page 2064)* on the remote server.
2. Select a tag.
   The collector creates a tag on the destination with the tagname to hold the collected data. The formula of the created tag is `Result=CurrentValue` and the source is the trigger.

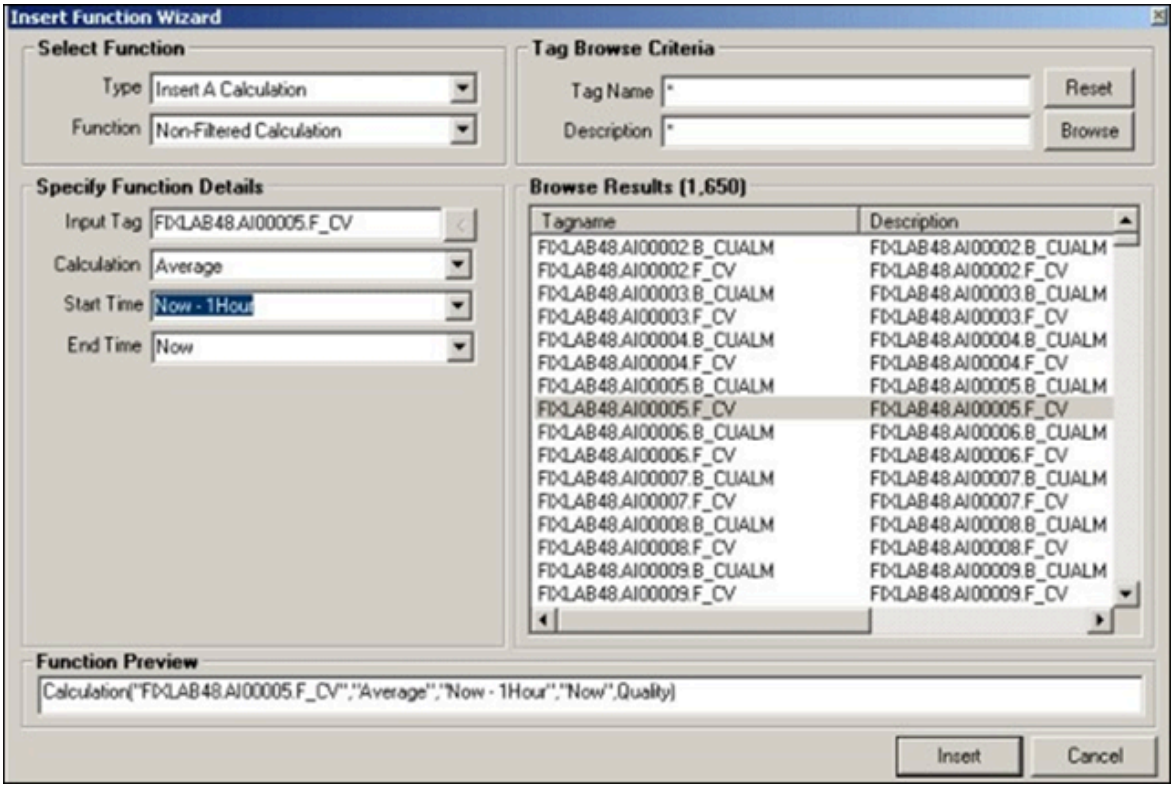
## Advanced Collection Example

In this example, you calculate a value (such as an hourly average of a source tag) or add two source tags together.

1. Using Historian Administrator, add a tag manually to the destination, as shown in the following image:

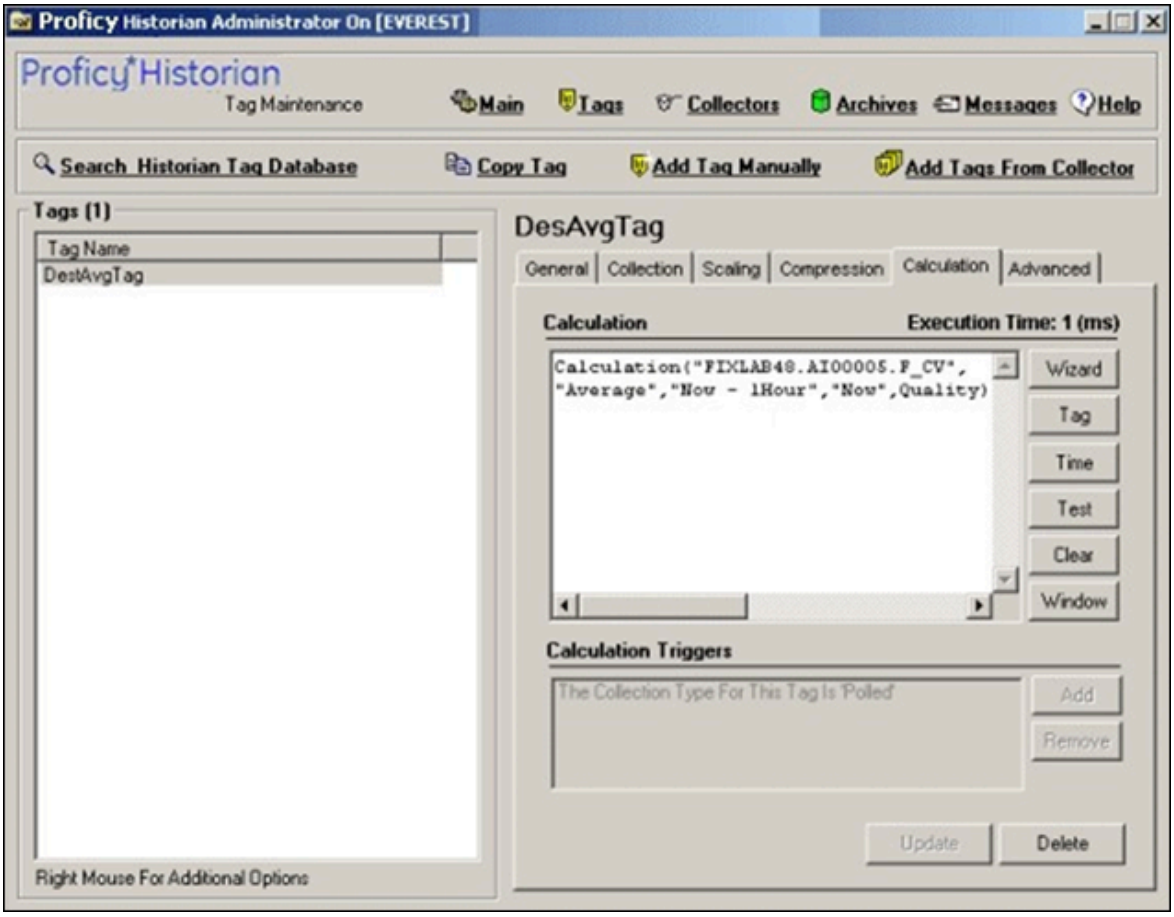



2. In the **Tags** section, select **Collection**.
3. In the **Collection Type** box, select **Polled**.
4. Set the **Collection Interval** to 1 hour.
5. Select **Calculation**.
6. In the **Calculation** section, enter the name of the tag, or use the **Insert Function Wizard** to browse and select the tagname. Then, build your calculation formula.
   The following figure shows an example of inserting a calculated value for a tag with the **Insert Function Wizard**.




7. Select **Insert**.

   The **Tag Maintenance** page appears, showing the formula in the **Calculation** section.




8. Select **Update** to save your changes.
   A message appears, asking you if you want to test the formula.
9. Select **Yes**.
   The Server-to-Server collector will begin processing the created tag upon the next collector reload.

# Creating Calculation Formulas

## About Calculation Formulas

To perform a calculation using the Calculation collector, you must define the calculation formula. You can do so in one of the following ways:

- Using the Insert Function wizard *(on page 2123)*, which helps you use any of the built-in functions *(on page 2127)* orcreate your own function *(on page 2125)*.
- Entering the syntax of the formula directly in the form of a VBScript code *(on page 2121)*.

Before you create calculation formulas, refer to the general guidelines *(on page 2119)*.

There are two predefined global values called Result and Quality. These global values control the value and quality of the output sample. If the Result is not set in the formula, then no sample is stored.

## General Guidelines for Defining a Calculation Formula

This section provides guidelines that you must follow when defining a calculation formula.

**Identify Time Intensive Calculations**

Use the `Calculation Execution Time` property of each tag to identify time-intensive queries. In Historian Administrator, look for the **Execution Time** on the **Calculation** section for an estimate of how long, on average, it takes for the calculation per tag (starting from the time the collector was started).

You can also include that column when you export tags to Excel using the Excel Add-In feature. For information, refer to Exporting Tags *(on page 2746)*.

You can also include that column (AverageCollectionTime) when you query the ihTags table using the Historian OLE DB Provider. Sorting by this column will let you find them fast.

**Troubleshoot Issues with Large Configurations**

If the timestamps of your raw samples appear slightly old, do not assume that the collector has stopped working. It is possible that the collector is just running behind.

For instance, if you have a report rate of 15,000, but the newest raw sample that you see is 20-30 minutes old, wait for 1-2 minutes, and review the newest raw sample again. If the collector stopped, the newest raw sample will be unchanged. If it did change, then the engine is still running, but is lagging behind. If that happens, check if the collector overrun count is increasing. If yes, the collector is dropping samples, and you must decrease the load.

**Error Handling in VBScript**

Start each script with the On Error Resume Next statement so that errors are trapped. If you use this statement, the script runs even if a run-time error occurs. You can then implement error handling in your VBScript.

It is a good practice to include statements in your VBScript that catch errors when you run the script. If there is an unhandled error, a value of 0 with a bad data quality is stored. When you catch an error in the VBScript, consider including a statement in your calculation that sets the Quality=0 when the error occurs. (The 0 value means that the quality is bad.) If you do not specifically include this setting in your script, Historian stores a good data quality point (Quality=100), even if an error has occurred in your formula. If Quality=100 is not appropriate for your application, consider setting the quality to 0.

You cannot use the On Error `GoTo` Label statement for error handling, as it is not supported in VBScript. As a workaround, you can write code in the full Visual Basic language and then place it in a `.DLL` so that you can call it from within your VBScript using the `CreateObject` function. For examples of calculations that use the `CreateObject` function, refer to Examples of Calculation Formulas *(on page 2138)*.

**Unsupported VBScript Functions**

You can use any VBScript syntax to build statements in a calculation formula with the exception of the following functions:

- `MsgBox`
- `InputBox`

**Milliseconds not Supported in VBScript**

The `CDate()` function does not support the conversion of a time string with milliseconds in it. Whenever you use the `CDate()` function, a literal time string, or a time string with a shortcut, do not specify milliseconds in the time criteria. Milliseconds are not supported in VBScript.

You cannot use milliseconds in times passed into built-in functions such as the `PreviousTime` and `NextValue` functions. For example, you cannot loop through raw samples with millisecond precision.

**Notes on VBScript Time Functions**

Using the VBScript time functions such as Now, Date, or Time can lead to unexpected results, especially in recalculation or recovery scenarios. To avoid these issues, use the `CurrentTime` built-in function provided by Historian, instead of Now, Date, or Time. For example, the VBScript Now is always the clock time of the computer and is likely not useful when recalculating or recovering data for times in the past. However, the "Now" time shortcut is equivalent to `CurrentTime` and can be used as input to the other built in functions.

**Using Quotation Marks in VBScript**

If you want to use quotation marks in a tag name, you must insert a double quotes for each quotation mark that you want to use, as required for proper VBScript syntax. For example, if you want to get the current value of a tag named TagCost"s, you must enter:

```
Result = CurrentValue("TagCost""s")
```

In this example, note the double quotation marks that appear before the letter s in the TagCost"s name in the formula.

**Avoiding Circular References in VBScript**

Do not use circular references in calculation formulas. For instance, if the tag name is `Calc1`, a formula with a circular reference would be `Result=CurrentValue("Calc1")`. Whether the tag is polled or unsolicited, you get a bad value back using the circular reference.

**Uninterrupted Object Method Calls**

Object method calls are not interrupted. It is possible to exceed the Calculation Timeout setting if you have a method call that takes a long time to execute. The Calculation Timeout error still occurs, but only after the method completes.

**Help for VBScript**

You can get detailed Help for VBScript by referencing the Microsoft documentation on the MSDN web site. A *VBScript User's Guide and Language Reference* is available here: http://msdn.microsoft.com/en-us/library/t0aew7h6.aspx

**Avoiding Deleted Tags**

You can reference a deleted tag in a calculation formula, without an error appearing. For instance, you could enter a formula such as `Result=CurrentValue("DeletedTag")`, where `DeletedTag` is the name of the deleted tag. You can do this because when you delete a tag, Historian removes deleted tags from the Tag Database (so you cannot browse for it), but it retains the data for that tag in the archive.

However, it is recommended that you do not reference deleted tag names in your calculation formulas, because if the archive files are removed with the data for the deleted tag, the calculation will not work properly.

## Create a Calculation Formula Using a VBScript Code

This topic describes how to create a calculation formula by entering a VBScript code. You can also create a calculation formula using the Insert Function wizard *(on page 2123)*.

**Important:**
If a tag contains bad data quality, you cannot store its value through a calculation formula. For example, if your VBScript includes: `Result = 7 Quality = 0`, Historian does not store the `7`, it stores `0`.

**Tip:**
For examples, refer to Examples of Scheduling Polled Triggers *(on page 2113)* and Examples of Scheduling Unsolicited Triggers *(on page 2117)*.

Create the tag that you want to use to store the calculation results. You can create the tag manually using Configuration Hub *(on page 473)*, Historian Administrator *(on page 2057)* or the Web Admin console *(on page 2059)*. Or, you can copy a tag *(on page 2060)*.

1. To create a calculation formula using Configuration Hub:
    a. Access Configuration Hub *(on page 336)*.
    b. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
    A list of tags appears.
    c. Select a Calculation tag, right-click or select more options, and then select **Calculation**.
    The calculation editor appears.
    d. Enter the VB script in the editor.
    e. To test the function, select .
    A message appears, stating whether the syntax is correct.
    f. In the upper-left corner of the page, select **Save**.
    The calculation formula is created.
2. To create a calculation formula using Historian Administrator:
    a. Access Historian Administrator *(on page 823)*.
    b. Select **Tags**, select the tag for which you want to create a calculation formula, and then select **Calculation**.
    c. In the **Calculation Triggers** section, select **Add**.
    The **Insert Function Wizard** window appears.
    d. Under **Select Function**, in the **Type** field, select **Add A Calculation Trigger**.
    e. Under **Tag Browse Criteria**, enter the search criteria to find the tag.
    The search results appear in the **Browse Results** section.
    f. Select the tag that you want to add, and then select **Insert**.

g. In the **Calculation** field, enter the calculation formula using the VBScript syntax.

h. To verify that the syntax is correct, select **Test**.
A message appears, stating whether the syntax is correct.

## Built-in Functions

This topic describes the built-in functions that you can use to create a calculation formula. You can also create your own calculation function *(on page 2136)*.

**Note:**

- In this table, `Time` refers to the actual time; this time can include absolute and relative time shortcuts. Refer to the Date/Time Shortcuts *(on page 2138)* section for more information.
- You cannot control the timestamp of the stored sample. It is determined by the triggering tag or polling schedule.
- You cannot use microseconds for any of the built-in calculation functions.

  For all the functions that retrieve previous values, it is similar to performing a `RawByNumber` query with a count of 1 and direction of backward. A less-than operation (not less-than-or-equal-to) is used on the timestamp to get the sample. Similarly, for all the functions that retrieve next values, it is similar to performing a `RawByNumber` query with a count of 1 and direction of forward. A greater-than operation (not greater-than-or-equal-to) is used on the timestamp to get the sample.

| Function Name | Description |
|---|---|
| `CurrentValue(<tag name>)` | The value of the tag, interpolated to the calculation execution time. The `CurrentValue` function returns 0 if the quality is 0 (bad quality). This occurs if you initialized it to 0, or if a previous call failed. |
| `CurrentQuality(<tag name>)` | The current quality of the tag (0 for bad quality and 100 for good quality). |
| `CurrentTime` | The calculation execution time, which becomes the timestamp of the stored value. |

| Function Name | Description |
|---|---|
| | For real time processing of polled tags, the calculation execution time is the time when the calculation is triggered. For unsolicited tags, the calculation execution time is the timestamp delivered with the subscription.<br><br>**Note:**<br>When a calculation is performed, the timestamp of the result is the time that the calculation has begun, not the time that it completed.<br><br>For recovery of polled or unsolicited tags, the calculation execution time is the time when the calculation would have been performed if the collector were running. |
| `Previous-Value(<tag name>, Time)` | The tag value of the raw sample prior to the current time. |
| `Previous-Quality(<tag name>, Time)` | The quality of the tag (0 for bad quality and 100 for good quality) prior to the current time. |
| `Previous-GoodVal-ue(<tag name>, Time)` | The latest good value of the raw sample prior to the current time. |
| `Previous-GoodQual-ity(<tag name>, Time)` | The good quality of the raw sample prior to the current time. |
| `Previous-Time(<tag name>, Time)` | The timestamp of the raw sample prior to the current time. |
| `Previ-ousGood-Time(<tag name>, Time)` | The timestamp of the latest good quality of the raw sample prior to the current time. |

| Function Name | Description |
|---|---|
| `NextValue(<tag name>, Time)` | The value of the raw sample after the current timestamp. |
| `NextQuality(<tag name>, Time)` | The quality of the tag (0 for bad quality and 100 for good quality) after the current time. |
| `NextTime(<tag name>, Time)` | The timestamp of the raw sample after the current timestamp. |
| `NextGoodValue(<tag name>, Time)` | The value of the good raw sample after the current time. |
| `NextGoodQuality(<tag name>, Time)` | The good quality of the raw sample after the current time. |
| `NextGoodTime(<tag name>, Time)` | The timestamp of the good raw sample after the current time. |
| `InterpolatedValue(<tag name>, Time)` | The tag value, interpolated to the time that you enter. |
| `Calculation` | Unfiltered calculated data query that returns a single value, similar to the Excel Add-In feature. For a list of the calculation mode, refer to Calculation Modes *(on page 1073)*. |
| `AdvancedCalculation` | Unfiltered calculated data query that returns a single value, similar to the Excel Add-In feature. For a list of the calculation mode, refer to Calculation Modes *(on page 1073)*. |
| `AdvancedFilteredCalculation` | Advanced Filtered calculated data query that returns a single value, similar to the Excel Add-In feature. |
| `FilteredCalculation` | Filtered calculated data query that returns a single value, similar to the Excel Add-In feature. |

| Function Name | Description |
|---|---|
| `LogMessage(string_message)` | Allows you to write messages to the Calculation collector or the Server-to-Server collector log file for debugging purposes. The collector log files are located in the `Historian\LogFiles` folder.<br><br>**Note:**<br>The `LogMessage` function is the only function that does not appear in the wizard. |
| `GetMultiFieldValue(Variable, <field name>)` | Returns the value of the field that you have specified. The variable contains the current value of all the fields of a multi-field tag. Before using this function, you must read the tag into a variable, using the `CurrentValue()` function. You can then use the `GetMultiFieldValue` function to access the value of the field.<br><br>The value of the field that you enter must be the same as the name of the field in the user defined type. If the field name is not found, a null value is returned. |
| `GetMultiFieldQuality(Variable, <field name>)` | Returns the quality (0 for bad quality and 100 for good quality) of the field that you have specified. The variable contains the current value of all the fields of a multi-field tag. Before using this function, you must read the tag into a variable, using the `CurrentValue()` function. You can then use the `GetMultiFieldValue` function to access the value of the field.<br><br>The value of the field that you enter must be the same as the name of the field in the user-defined type. If the field name is not found, a null value is returned.<br><br>If the user-defined type can store individual quality, you get the field quality. Otherwise, you get the sample quality. |
| `SetMultiFieldValue(Variable, <field name>, Value, Quality)` | Sets the value and the quality for the field that you have specified.<br><br>You can use this function to construct a multifield value containing values for each field, and then use the `result=` syntax to store the value in Historian. |

## Counting the Number of Bad Quality Samples

The following example shows how to loop through samples of a tag named C2 to count the number of bad quality samples.

```
Dim count, starttime, endtime, tagquality count=0

StartTime=CurrentTime EndTime=DateAdd("n",-1,StartTime) Do while StartTime>EndTime

TagQuality=PreviousQuality("C2",StartTime)

startTime=PreviousTime("C2",StartTime) IF TagQuality=0 THEN

count=count + 1

END IF loop Result=count
```

### Counting the Number of Collected Digital 1s For a Tag

The following example counts the number of collected digital 1s for a tag so that, for instance, you can determine how many times a pump is turned ON and OFF.

```
Dim count, starttime, endtime,tagquality,TagValue

count=0

StartTime=CurrentTime

EndTime=DateAdd("h",-1,StartTime)

On error resume next

Do while StartTime>=EndTime

TagValue=PreviousValue("FIX.DI.F_CV",StartTime)

TagQuality=PreviousQuality("FIX.DI.F_CV",StartTime)

startTime=PreviousTime("FIX.DI.F_CV",StartTime)

IF TagQuality=100 AND TagValue=1 then

count=count + 1

END IF

loop

Result=count
```

### Determining the Trigger When Using Multiple Trigger Tags

The following example shows how to determine which tag triggered the calculation, from a list of two possible trigger tags. The example compares the two trigger tags and determines which one has the newest raw sample. This method of getting the newest raw sample can also be used to determine if a remote collector is sending data or is disconnected from the server.

In this example, archive compression is disabled for both of these tags.

```
dim timetag1

dim timetag2

dim tag1

dim tag2
```

```
tag1 = "BRAHMS.AI1.F_CV"
tag2 = "BRAHMS.AI2.F_CV"

' Get the timestamp of the newest raw sample for tag1:
timetag1 = previousTime(tag1, CurrentTime)

' Get the timestamp of the newest raw sample for tag2:
timetag2 = previousTime(tag2, CurrentTime)

if timetag1 > timetag2 then
' If tag1 triggered me, then:
result = 1 else
' If tag2 triggered me, then:
result = 2
end if
```

## Using Array or Multifield Data in Calculation

You can create tags of arrays and multifield types and use the Calculation collector, Server-to-Server collector, Server-to-Server distributor with these tags.

### Arrays

To use the Array data as input to a calculation formula you can use the name of the array tag like "Array1" or the individual element of the array like "Array1[4]". For example, if you have an array tag "Array1" of floating point values and a calculation tag "FloatCalc1" of float data type, then you can use the array as input to calculate a float value.

```
result = currentvalue("Array1[4]")+5
```

You can use Calculation() function to read the array tag as shown in the following code.

```
Result = Calculation("Array1","Average","Now 1Minute","Now",Quality)
```

In this example, the calculation tag should be an array tag because the average of an array is an array, not a single value. Each element is averaged over the time range. Since an average of an integer or float array is a floating point value, the calculation tag must be a single or double float array.

If you want to find the minimum of array elements in a given time, then use vbscript code to compute and store the result in a Float tag as shown.

```
if CurrentValue("Array1[0]") < CurrentValue("Array1[1]") then
    Result = CurrentValue("Array1[0]")
else
    Result = CurrentValue("Array1[1]")
end if
```

### Multifield

If you have a user-defined type "MySample" with fields "r;FloatVal" and "r;IntVal" you can create `Tag1` and use the value of one field in an Integer Calc Tag. The destination tag is not a multifield tag.

```
result = currentvalue("Tag1.IntVal")+5
```

## Storing Array or Multifield data in Calculation tags

### Array

If your calculation tag is an array tag, then you can copy the entire array values into it. For example, you can copy the entire values from `Array1` into `Array2` using the given code.

```
result = CurrentValue("Array1")
```

You can take an array value collected from a field device and adjust the values before storing it in another array tag `Array2` using this code:

```
dim x
x=CurrentValue("Array1")
x(1) = x(1)+10
result = x
```

You can simply construct an array value inside your formula and store it in `Array2`, for example:

```
dim MyArray(2)' The 2 is the max index not the size
MyArray(0)=1
MyArray(1)=2
MyArray(2)=3 result = MyArray
```

### Multifield

You can have the collector combine collected data into a multifield tag. Create a calculation `Tag1` using the user-defined Type "MySample," then use this formula to fill in the fields:

```
Dim InputValue, myval,x,y
```

```
' get the current value of another multifield tag

InputValue = CurrentValue("tag1")


' get the values of each of the fields

x = GetMultiFieldValue(InputValue, "IntVal")

y = GetMultiFieldValue(InputValue, "floatval")


' store the field values in this tag

SetMultiFieldValue myval,"IntVal",x,100

SetMultiFieldValue myval,"floatval",y,100

Result = myval
```

## Using Array or Multifield data to trigger calculation

### Array

You can use the array tag as a trigger tag for your float or array calculation tags. For example, you can use `Array1` as a trigger so that when it changes, the `"CalcArray1"` tag will be updated. You cannot use an individual array element such as `"Array1[3]"` as a trigger, you must use the entire array tag as the trigger tag.

### Multifield

You can use a multifield tag as a trigger tag by either using the tagname `"Tag1"` or tagname with the field name `"Tag1.FloatVal"`.

## Sending Array or Multifield data to a Remote Historian

### Array

You can use the Server to Server Collector or Server to Server Distributor to send array data to a destination Historian. If the destination Historian is version 6.0 or later, you can simply browse the tags and add them.

You cannot send an array to the older versions of archiver (Pre 6.0 versions) as these archivers will store the array tags as a blob data type in the destination and you will not be able to read them. However, you can send individual elements of an array to these archivers, for example, `result = currentvalue("Array1 [4]")`.

### Multifield

The destination needs to be Historian 6.0 or above to store a multifield tag but you can send individual fields to a pre Historian 6.0 archiver.

For multifield tags, you must create the User Defined Type manually at the destination

You can write an entire multified tag data sample in one write or you can create multiple tags in the destination, one for each field you want to copy. For example, if you have one tag `"Tag1"` with two fields `"FloatVal"` and `"IntVal"` on a source archiver, then you can create two tags (`"Tag1.FloatVal"` and `"Tag1.IntVal"`) on the destination.

**Note:**
If you change a field name or add or remove fields you must update your collection and your destination tags.

### Reading and writing a Multifield tag using MultiField functions

The following example shows how to read an entire multifield tag, using the `GetMultiFieldValue` function and to write the value to a field in another tag using the `SetMultiFieldValue` function.

```
Dim CurrMultifieldValue


' Read the value of a multi field tag into a variable

CurrMultifieldValue = CurrentValue("MyMultifieldTag")


' Read the field value of multifield tag into the temporary variable

F1 = GetMultiFieldValue(CurrMultifieldValue, "Temperature Field")


' Perform a calculation on the value

Celcius = (F1 32)/ 9* 5


' Set the calculated value to another field of the multifield tag

SetMultiFieldValue(CurrMultifieldValue, "Temperature Field Celcius", Celcius, 100)

result = CurrMultifieldValue
```

## User-defined Functions

In addition to the built-in functions *(on page 2127)*, you can create custom calculation functions. After you create a custom calculation function, it is available for use with other calculations as well.

Functions are useful as shortcuts for large blocks of source code. By creating a function out of commonly used calculation formulas, you can save time and effort instead of typing a few lines of calculation formula every time you want to perform the same operation, it is compressed to a single line.

The syntax of a function is simple:

```
Function functionname (variable list)

   [calculation formulas]

End Function
```

The operations a function performs are contained within the Function / End Function statements. If you need to send data to the function a tag name, for example you simply create a variable in the function's parameters to receive the data. Multiple variables must be separated by commas. These variables exist only within the function.

The following is an example of a function. This function, named `checkValue()`, looks at a tag and assigns it an alarm if it is over a specified value.

### A Function to Assign an Alarm to a Tag Based on a Condition

The following function, named checkValue, assigns an alarm to a tag if the tag value reaches a specified value.

```
Function checkValue (tagname,sourcename,value)

  If CurrentValue(tagname) > value Then

  Set AlarmObj = new Alarm

  AlarmObj.SubConditionName = "HI"

  AlarmObj.Severity = 750

  AlarmObj.NewAlarm

  "alarmname", "Simulated", "tagname", "Now"

  checkValue = true

Else

  checkValue = false

  End If

End Function
```

If you want to use this function, enter the values for tag name, source name, and value, as shown in the following example:

```
alm_set = checkValue("DD098.FluidBalance","FluidBalance_ALM",5000)
```

In this example, if the value of the DD098.FluidBalance tag exceeds 5000, the function returns a true value, indicating that the alarm was set; the *alm_set* variable will be set to `true`. Otherwise, the *alm_set* variable will be set to `false`.

## Create a User-Defined Function

This topic describes how to create your own function to use in a calculation formula. For more information, refer to User-defined Functions *(on page 2136)*. You can also use any of the built-in functions *(on page 2127)*.

Create the tag that you want to use to store the calculation results. You can create the tag manually using Configuration Hub *(on page 473)*, Historian Administrator *(on page 2057)* or the Web Admin console *(on page 2059)*. Or, you can copy a tag *(on page 2060)*.

1. To create a user-defined function using Configuration Hub:
    a. Access Configuration Hub *(on page 336)*.
    b. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
    A list of tags appears.
    c. Select a calculation tag, right-click or select more options, and then select **Calculation**.
    The calculation editor appears.
    d. In the **CALCULATION** section, remove `Null` (retain `Result =`).
    e. In the calculation editor, place the cursor in the location where you want to insert the user-defined function.
    f. In the **DETAILS** section, under **User Defined Functions**, select .
    The **Add/Edit User Defined Functions (UDF)** window appears.
    g. Select **Add New**.
    A function is created with the following naming convention: UserFunction<*number*>. The same function is used in the function editor. You can change the function name by entering the new name in the function editor and selecting **Update UDF**.
    h. Enter the VB script for the function. Or, if you want to use a pre-built function, select the function type and function in the respective fields under **Pre-Built Functions**. And, enter values in the other fields that appear after selecting a function.
    The function appears in the function preview below the function editor.
    i. Select **Insert Preview**.
    The function is included in the function editor at the cursor position.
    j. To test the function syntax, select .

A message appears, stating whether the function syntax is correct.

k. Select **Update UDF**.
The user-defined function is created.

l. To use the function in the calculation formula, select **Insert UDF**.
The function is inserted in the calculation formula at the cursor position.

m. To test the calculation formula, select .
A message appears, stating whether the syntax is correct.

n. In the upper-left corner of the page, select **Save**.
The calculation formula is created.

2. To create a user-defined function using Historian Administrator:

a. In Historian Administrator, select **Tags**, select the tag for which you want to create a calculation formula, and then select **Calculation**.

b. In the **Calculation** section, remove `Null` (retain `Result =`).

> **Tip:**
> Avoid selecting other tags until you save your changes or you will lose your code changes.

c. Select **Functions**.
The **User Defined Functions** window appears.

d. Select **New**.
The **Edit Function** window appears.

e. Define the function.
You can build formulas using the wizard, or create it manually by entering functions in the **Edit Function** box. For information, refer to User-defined Functions *(on page 2136)*.

f. Select **Syntax** to check for errors.

g. Select **Update**.
Your function appears in the list, and is available for use in other calculations as well.

h. To use the function, select **Insert Function**.
The function is inserted in your calculation formula.

## Date/Time Shortcuts

The following table outlines the date/time shortcuts that you can use in calculation formulas.

**Table 383. Date/Time Shortcuts**

| Shortcut | Description |
|---|---|
| Now | Now (the time and date that you execute the query) |
| Today | Today at midnight |
| Yesterday | Yesterday at midnight |
| BOY | First day of year at midnight |
| EOY | Last day of year at midnight |
| BOM | First day of month at midnight |
| EOM | Last day of month at midnight |

**Relative Date/Time Shortcuts**

Optionally, you can add or subtract relative times to the following absolute times. You must use them in conjunction with the date/time shortcuts listed in the preceding table (for example, Today+5h+3min instead of 5h3min).

- Second
- Minute
- Hour
- Day
- Week

## Create a Calculation Formula Using the Pre-built Functions

This topic describes how to create a calculation formula using the pre-built functions. You can also create a calculation formula using a VBScript code *(on page 2121)*.

For information on a list of the available types and associated functions, refer to Types of Functions Supported *(on page 2135)*. For information on each pre-defined function, refer to Built-In Functions *(on page 2127)*. In addition to the built-in functions, you can create your own customized functions *(on page 2136)*.

1. Create the tag that you want to use to store the calculation results. You can create the tag manually using Configuration Hub *(on page 473)*, Historian Administrator *(on page 2057)* or the Web Admin console *(on page 2059)*. Or, you can copy a tag *(on page 2060)*.
2. Access the advanced options of the collector, and then disable the **On-line Tag Configuration Changes** option using Historian Administrator *(on page 2057)*. In Configuration Hub *(on page 473)* this property is available in Collector Options section. If you do so, each time you update a calculation formula, the collector does not reload tags.

1. To create a calculation formula using Configuration Hub:

    a. Access Configuration Hub *(on page 336)*.

    b. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
    A list of tags appears.

    c. Select a calculation tag, right-click or select more options, and then select **Calculation**.
    The calculation editor appears.

    d. In the **CALCULATION** section, remove `Null` (and retain the `Result =`).

    e. In the Calculation editor, place the cursor in the location where you want to insert the pre-built function.

    f. In the **DETAILS** section, under **Pre-Built Functions**, in the **Function Type** field, select a function type.
    Depending on the function type you have selected, a list of functions appears in the **Function** field.

    g. In the **Function** field, select a function.
    Based on the selected function, the related fields will be displayed.

    h. Enter values in the other fields that appear after selecting a function.

    i. In the **Input Tag** field, select , and then select the tag where applicable.
    The function preview appears below the calculation editor.

    j. Select **Insert Function**.
    The function is inserted at the cursor position.

    k. To test the function, select .
    A message appears, stating whether the syntax is correct.

l. In the upper-left corner of the page, select **Save**.
The calculation formula is created.

2. If you want to create a calculation formula using Historian Administrator:

   a. Access Historian Administrator *(on page 823)*.

   b. Select **Tags**, select the tag for which you want to create a calculation formula, and then select **Calculation**.

   c. In the **Calculation** section, remove `Null` (retain `Result =`).

   > **Tip:**
   > Avoid selecting other tags until you save your changes or you will lose your code changes.

   d. Select **Wizard**.
   The **Insert Function Wizard** window appears.

   e. Under **Select Function**, select values in the available fields, and then select **Insert**.

   f. If you want to perform an unsolicited (also called event-based) calculation, in the **Type** field, select **Add A Calculation Trigger**. Search and select the tag that you want to add, and then select **Insert**.
   The calculation formula is created.

   g. To verify that the syntax is correct, select **Test**.
   A message appears, stating whether the syntax is correct.

## Types of Functions Supported

The following table describes the types of actions supported. All the value functions return a single value.

| Type of Action | Available Functions for the Action |
|---|---|
| Insert a value | • Current value<br>• Previous value<br>• Next value<br>• Interpolated value |

| Type of Action | Available Functions for the Action |
|---|---|
| Insert a calculation | • Unfiltered calculation<br>• Filtered calculation |
| Insert a timestamp | • Time shortcut<br>• Previous value timestamp<br>• Next value timestamp<br>• Current time |
| Check data quality | • Current value quality<br>• Previous value quality<br>• Next value quality |
| Set data quality | • Set Quality Good<br>• Set Quality Bad |
| Add data value | Value |
| Insert a tag name | Tagname |
| Insert an alarm calculation | • Previous Alarm<br>• Next Alarm<br>• Get Alarm Property<br>• Set Alarm Property<br>• Add Event<br>• New Alarm<br>• Update Alarm<br>• Return to Normal |
| Insert a multifield operation | • GetMultiFieldValue<br>• GetMultiFieldQuality<br>• SetMultiFIeldValue |

# Data Input

## Calculation and Server-to-Server Collectors

The Calculation and Server-to-Server collectors have some unique behavior not found in other standard collectors. This section provides details about Recovery *(on page 2152)* and Manual Recalculation *(on page 2153)*.

## Recovery

This feature is unique to Calculation and Server-to-Server collectors. If the calculation engine is not running for a period of time, recovery makes it look like it was running. Recovery can also be used to fill in a hole of time where the collector was not able to communicate with the source archiver.

Recovery is applicable to both unsolicited and polled tags. Messages are also recovered. Comments are not recovered.

Normally, it is impossible to go back to the past and collect data. However, since these collectors are 'deriving' data instead of 'collecting' data, it is possible to recover past data, especially since the source of the derived data is archived in the Historian. It is important to understand that while recovery is possible in the calculation and Server-to-Server collectors, it only makes sense for certain types of calculation formulas.

Intended candidates for data recovery are formulas whose only inputs are Historian tags, since past data for these tags can be interpolated. Formulas that use data from external text files or from ADO via CreateObject will most likely not recover correct data because the inputs are not historized. If you are using these types of formulas, you should turn off recovery for the whole collector or insert VBScript code in the formula of individual tags to detect recovery. An example of this is given in the Historian documentation. A similar approach can be used to set a Max Recovery Time on a tag basis, overriding the collector wide setting.

Even calculation tags using only Historian tags as inputs have some caveats for recovery. If you are deriving calculated data from other calculated data, be sure to set up a trigger tag for each of the tags used in your formula. This way the tags will be processed in chain order. All tags are processed in time order.

The recovery logic is not intended to overcome polled collection overruns. If you configure too much collection, then you will get overruns.

You can control the amount of recovered data using Max Recovery Time configuration setting. You can turn off the recovery by setting it to zero.

## Manual Recalculation

The Manual re-calc/re-replicate option is often the best choice for generating past derived data.

**Note:**
If you perform a server-to-server recalculation on source and destination servers whose clocks are not synchronized, extra data points may appear and original data points may not be

recalculated. To ensure this does not occur, ensure the time is synchronized on both source and destination servers.

**S2S/S2C collector Backfill procedure**

With the Recalculate feature you can recalculate all tags for the time period during and after the connection loss. The recalculated tags will use the most accurate values in calculations.

During the period of connection loss, the collector buffers the data. When the connection is restored, the buffered data is forwarded to the Historian Server. When the buffered data arrives, the timestamps show earlier time than the most recent calculation timestamp.

Since the timestamp is earlier, the polled calculations will not execute again with the new data but the unsolicited calculations will re-trigger. Therefore, it is possible that calculations performed for tags during and after the connection loss might be not be entirely accurate.

**Run S2C Backfill via Command line**

```
ihServerToServerCollector.exe RELOADFILENAME=[file location]

RELOADUSERNAME=[Username] <start time> <end time>
```

**Example**: `C:\Program Files\Proficy\Proficy Historian\x86\Server>ihServerToServerCollector.exe RELOADFILENAME=c:\taglist.txt RELOADUSERNAME=\Administrator 1516875659 1516875785`

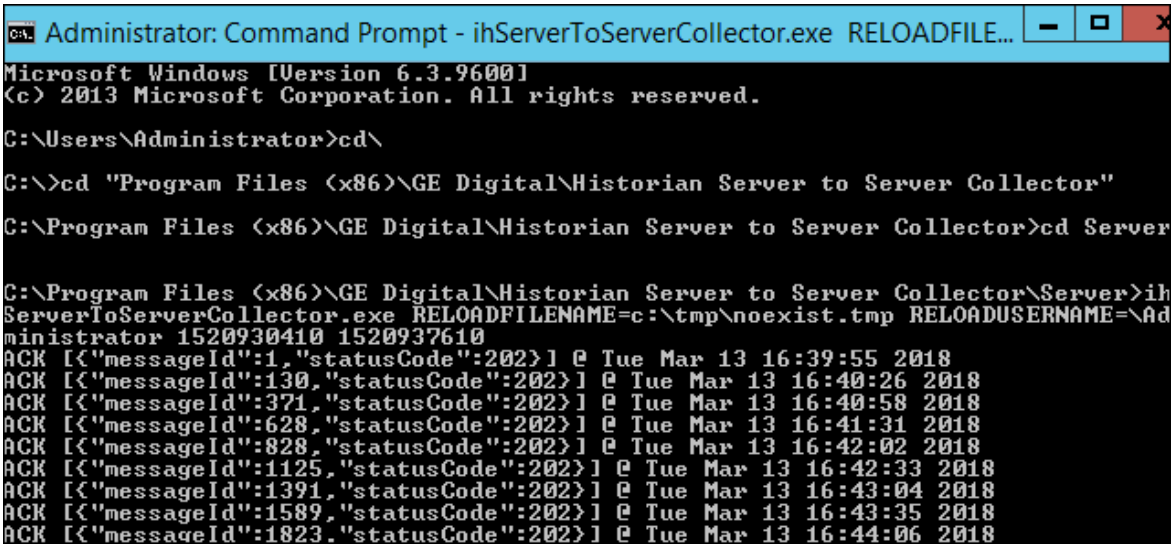
For multi-instance support, the command requires the interface name as shown in the following example:

**Example for Multi-instance Support**: `C:\Program Files (x86)\GE Digital\Historian Server to Server Collector\Server>ihServerToServerCollector.exe RELOADFILENAME=c:\taglist.txt RELOADUSERNAME=\Administrator 1669462600 1669463200 REG=sekhartest05_To_PredixOFFSS`

See the following information about the parameters:

- RELOADFILENAME: This is an optional parameter. File name should be absolute path, this file consists of the tag names, for which Backfill should be performed, each tag should be separated by new line. Any discrepancies in the file/no file exists/parameter not provided leads to Backfill all the tags related to the collector at the current time. After the Backfill, file gets deleted.
- RELOADUSERNAME: This is an optional parameter. This username is used only when destination server is Historian for auditing purpose, and gets ignored when the destination is cloud.

- TIMESTAMP: This parameter accepts Start and end time in seconds in epoch format for which Backfill should happen. https://www.epochconverter.com/




**How Data Recovery works:**

- When the recovery logic is executed, the collector will setup subscriptions for all the trigger tags.
- Next, it will recover data. The collector first determines how long it has been since the last write. It compares the current time to data in the registry key `LastCalcRepWriteTime`, which stores the last time data was written to the archive. The collector compares this to the Max Recovery Time that is specified in the user settings and performs a raw data query on the shorter of these two periods. Then it will take the shorter of these two and do a raw data query for all trigger tags. It will then process the returned samples in sequential order based on time. For example, if the collector was shut down for 8 hours, but Max Recovery Time was 4 hours, only 4 hours of data would be recovered.
- Recovery is performed before real time processing. Once recovery is complete, it will start polling and processing subscriptions in real time. The subscriptions in real time are queued up till the recovery is done.
- Recovery logic will place an end-of-collection marker at the point in time where the collector was shut down. This end-of-collection marker may or may not be there once the recovery is complete. As part of recovery logic, if it calculates a data point exactly at that timestamp where the end-of-collection marker is there, then it will be overwritten with the calculated good data.
- The recovery logic does not write samples to trigger tags or tags that are just in the formula. It is intended to write samples to the calculation tags.
- Messages are added to the log file that indicate when entering and exiting recovery mode.

**Examples**

The examples below assume the following tag configuration.

- Machine 1:

  Runs Data Archiver, iFIX collector (Collector 1), and Calculation collectors.

- Machine 2:

  Runs iFIX collector (Collector 2), which collects and sends data to the archiver in Machine 1 (as a Remote Collector).

  TagA and TagB are the iFix tags coming from Collector1 and Collector2, respectively. Both of these tags are scanned at a 1-minute poll rate.

The following example demonstrates the recovery function for an unsolicited 1-minute interval calculation tag that has a simple current value function.

Create an event based 1-minute interval Calculation Tag (CalcTag1) in Machine 1 consisting of the following calculation: `Result=CurrentValue (TagA)`

Stop the Calculation collector for 5 minutes and then restart it to trigger data recovery for the 5-minute shutdown period. For the following example, the Calculation collector was stopped at 2002-12-27 17:05:36 and started at 2002-12-27 17:10:48.

Since there is no interruption for the iFIX collector, the raw data query for TagA results the following output:

***Raw Data Query for TagA during shutdown period***

114) 81 [2002-12-27 17:02:00:00000] Good NonSpecific

115) 72 [2002-12-27 17:03:00:00000] Good NonSpecific

116) 64 [2002-12-27 17:04:00:00000] Good NonSpecific

117) 56 [2002-12-27 17:05:00:00000] Good NonSpecific

118) 39 [2002-12-27 17:06:00:00000] Good NonSpecific

119) 31 [2002-12-27 17:07:00:00000] Good NonSpecific

120) 22 [2002-12-27 17:08:00:00000] Good NonSpecific

121) 14 [2002-12-27 17:09:00:00000] Good NonSpecific

122) 6 [2002-12-27 17:10:00:00000] Good NonSpecific

A raw data query for CalcTag1 during the shutdown period generates the following:

***Raw Data Query for CalcTag1 (before recovery)***

96) 81 [2002-12-27 17:02:00:00000] Good NonSpecific

97) 72 [2002-12-27 17:03:00:00000] Good NonSpecific

98) 64 [2002-12-27 17:04:00:00000] Good NonSpecific

99) 56 [2002-12-27 17:05:00:00000] Good NonSpecific

100) 0 [2002-12-27 17:05:36:00000] Bad OffLine

Note that an end-of-collection marker is placed at the shutdown point (that is, at 17:05:36) with a bad data quality.

Once the recovery is complete, this is what we see for the recovered CalcTag1. Note that data during the shutdown period is recovered completely. Compare this result set with the one for TagA. Both are the same.

*Raw Data Query for CalcTag1 (after recovery)*

96) 81 [2002-12-27 17:02:00:00000] Good NonSpecific

97) 72 [2002-12-27 17:03:00:00000] Good NonSpecific

98) 64 [2002-12-27 17:04:00:00000] Good NonSpecific

99) 56 [2002-12-27 17:05:00:00000] Good NonSpecific

100) 0 [2002-12-27 17:05:36:00000] Bad OffLine

101) 39 [2002-12-27 17:06:00:00000] Good NonSpecific

102) 31 [2002-12-27 17:07:00:00000] Good NonSpecific

103) 22 [2002-12-27 17:08:00:00000] Good NonSpecific

104) 14 [2002-12-27 17:09:00:00000] Good NonSpecific

105) 6 [2002-12-27 17:10:00:00000] Good NonSpecific

Also note that the end-of-collection marker is not overwritten by the recovery logic here. If it calculated a data point exactly at the end-of-collection marker, then it would have been overwritten by the calculated good value.

The following example demonstrates the recovery function for an unsolicited calculation tag that has multiple triggers.

Create an event based Calculation Tag (CalcTag2) in Machine 1 consisting of the following calculation:

```
Result=CurrentValue (TagA) + CurrentValue (TagB)
```

where TagA and TagB are both trigger tags, coming from Collector1 and Collector2 respectively. Set the collection offset of 5 seconds for TagA and 10 seconds for TagB, forcing the calculation to be performed twice per minute.

Stop the Calculation collector for 5 minutes, and then restart it to trigger data recovery for this 5-minutes shutdown period. For the following example, the Calculation collector was stopped at 02/18/2003 12:15:33 and started at 02/18/2003 12:21:53.

Since the iFIX collector was not interrupted, a raw data query for TagA and TagB values generates the following output:

***Raw Data Query for TagA during the shutdown period***

10) 13 [2003-02-18 12:10:05:00000] Good NonSpecific

11) 12 [2003-02-18 12:11:05:00000] Good NonSpecific

12) 11 [2003-02-18 12:12:05:00000] Good NonSpecific

13) 11 [2003-02-18 12:13:05:00000] Good NonSpecific

14) 10 [2003-02-18 12:14:05:00000] Good NonSpecific

15) 18 [2003-02-18 12:15:05:00000] Good NonSpecific

16) 17 [2003-02-18 12:16:05:00000] Good NonSpecific

17) 16 [2003-02-18 12:17:05:00000] Good NonSpecific

18) 16 [2003-02-18 12:18:05:00000] Good NonSpecific

19) 15 [2003-02-18 12:19:05:00000] Good NonSpecific

20) 14 [2003-02-18 12:20:05:00000] Good NonSpecific

21) 13 [2003-02-18 12:21:05:00000] Good NonSpecific

***Raw Data Query for TagB during the shutdown period***

10) 35 [2003-02-18 12:10:10:00000] Good NonSpecific

11) 34 [2003-02-18 12:11:10:00000] Good NonSpecific

12) 33 [2003-02-18 12:12:10:00000] Good NonSpecific

13) 32 [2003-02-18 12:13:10:00000] Good NonSpecific

14) 31 [2003-02-18 12:14:10:00000] Good NonSpecific

15) 31 [2003-02-18 12:15:10:00000] Good NonSpecific

16) 39 [2003-02-18 12:16:10:00000] Good NonSpecific

17) 38 [2003-02-18 12:17:10:00000] Good NonSpecific

18) 37 [2003-02-18 12:18:10:00000] Good NonSpecific

19) 36 [2003-02-18 12:19:10:00000] Good NonSpecific

20) 36 [2003-02-18 12:20:10:00000] Good NonSpecific

21) 35 [2003-02-18 12:21:10:00000] Good NonSpecific

A raw data query for CalcTag2 during the shutdown period generates the following:

*Raw Data Query for CalcTag2 (before recovery)*

12) 50 [2003-02-18 12:09:05:00000] Good NonSpecific

13) 50 [2003-02-18 12:09:10:00000] Good NonSpecific

14) 49 [2003-02-18 12:10:05:00000] Good NonSpecific

15) 48 [2003-02-18 12:10:10:00000] Good NonSpecific

16) 47 [2003-02-18 12:11:05:00000] Good NonSpecific

17) 46 [2003-02-18 12:11:10:00000] Good NonSpecific

18) 45 [2003-02-18 12:12:05:00000] Good NonSpecific

19) 44 [2003-02-18 12:12:10:00000] Good NonSpecific

20) 44 [2003-02-18 12:13:05:00000] Good NonSpecific

21) 43 [2003-02-18 12:13:10:00000] Good NonSpecific

22) 42 [2003-02-18 12:14:05:00000] Good NonSpecific

23) 41 [2003-02-18 12:14:10:00000] Good NonSpecific

24) 49 [2003-02-18 12:15:05:00000] Good NonSpecific

25) 49 [2003-02-18 12:15:10:00000] Good NonSpecific

26) 0 [2003-02-18 12:15:11:00000] Bad OffLine

Once data recovery is complete, this is what we see for the recovered data for CalcTag2. Note that data during the shutdown period is completely recovered:

***Raw Data Query for CalcTag2 (after recovery)***

12) 50 [2003-02-18 12:09:05:00000] Good NonSpecific

13) 50 [2003-02-18 12:09:10:00000] Good NonSpecific

14) 49 [2003-02-18 12:10:05:00000] Good NonSpecific

15) 48 [2003-02-18 12:10:10:00000] Good NonSpecific

16) 47 [2003-02-18 12:11:05:00000] Good NonSpecific

17) 46 [2003-02-18 12:11:10:00000] Good NonSpecific

18) 45 [2003-02-18 12:12:05:00000] Good NonSpecific

19) 44 [2003-02-18 12:12:10:00000] Good NonSpecific

20) 44 [2003-02-18 12:13:05:00000] Good NonSpecific

21) 43 [2003-02-18 12:13:10:00000] Good NonSpecific

22) 42 [2003-02-18 12:14:05:00000] Good NonSpecific

23) 41 [2003-02-18 12:14:10:00000] Good NonSpecific

24) 49 [2003-02-18 12:15:05:00000] Good NonSpecific

25) 49 [2003-02-18 12:15:10:00000] Good NonSpecific

26) 0 [2003-02-18 12:15:11:00000] Bad OffLine

27) 48 [2003-02-18 12:16:05:00000] Good NonSpecific

28) 56 [2003-02-18 12:16:10:00000] Good NonSpecific

29) 55 [2003-02-18 12:17:05:00000] Good NonSpecific

30) 54 [2003-02-18 12:17:10:00000] Good NonSpecific

31) 54 [2003-02-18 12:18:05:00000] Good NonSpecific

32) 53 [2003-02-18 12:18:10:00000] Good NonSpecific

33) 52 [2003-02-18 12:19:05:00000] Good NonSpecific

34) 51 [2003-02-18 12:19:10:00000] Good NonSpecific

35) 50 [2003-02-18 12:20:05:00000] Good NonSpecific

36) 50 [2003-02-18 12:20:10:00000] Good NonSpecific

37) 49 [2003-02-18 12:21:05:00000] Good NonSpecific

38) 48 [2003-02-18 12:21:10:00000] Good NonSpecific

39) 47 [2003-02-18 12:22:05:00000] Good NonSpecific

40) 46 [2003-02-18 12:22:10:00000] Good NonSpecific

The following example demonstrates the recovery function for an unsolicited calculation tag that has multiple triggers, but for which none of the triggers is in the formula.

TagA and TagB are the iFix tags coming from Collector1 and Collector2, respectively. Both tags are scanned at a 1-minute poll rate. This example uses two more iFix tags, TagC and TagD, coming from Collector1.

Create an event-based Calculation Tag (CalcTag3) in Machine 1 consisting of the following calculation:

```
Result=CurrentValue (TagA) + CurrentValue (TagB)
```

Make sure that the trigger tags for this calculation tag are TagC and TagD, which are not in the formula. Set the collection offset of 5 seconds for TagC and 10 seconds for TagD, forcing the calculation to be performed twice per minute.

Stop the Calculation collector for 5 minutes, and then restart it to trigger data recovery for this 5-minutes shutdown period. For the following example, the Calculation collector was stopped at 02/18/2003 02:24:37 and started at 02/18/2003 02:31:44.

Since the iFIX collector was not interrupted, a raw data query for TagA and TagB values generates the following output:

***Raw Data Query for TagA during shutdown period***

56) 13 [2003-02-18 14:21:05:00000] Good NonSpecific

57) 12 [2003-02-18 14:22:05:00000] Good NonSpecific

58) 11 [2003-02-18 14:23:05:00000] Good NonSpecific

59) 11 [2003-02-18 14:24:05:00000] Good NonSpecific

60) 10 [2003-02-18 14:25:05:00000] Good NonSpecific

61) 19 [2003-02-18 14:26:05:00000] Good NonSpecific

62) 18 [2003-02-18 14:27:05:00000] Good NonSpecific

63) 17 [2003-02-18 14:28:05:00000] Good NonSpecific

64) 16 [2003-02-18 14:29:05:00000] Good NonSpecific

65) 16 [2003-02-18 14:30:05:00000] Good NonSpecific

66) 15 [2003-02-18 14:31:05:00000] Good NonSpecific

*Raw Data Query for TagB during shutdown period*

141) 36 [2003-02-18 14:20:10:00000] Good NonSpecific

142) 36 [2003-02-18 14:21:10:00000] Good NonSpecific

143) 35 [2003-02-18 14:22:10:00000] Good NonSpecific

144) 34 [2003-02-18 14:23:10:00000] Good NonSpecific

145) 33 [2003-02-18 14:24:10:00000] Good NonSpecific

146) 32 [2003-02-18 14:25:10:00000] Good NonSpecific

147) 31 [2003-02-18 14:26:10:00000] Good NonSpecific

148) 31 [2003-02-18 14:27:10:00000] Good NonSpecific

149) 39 [2003-02-18 14:28:10:00000] Good NonSpecific

150) 38 [2003-02-18 14:29:10:00000] Good NonSpecific

151) 37 [2003-02-18 14:30:10:00000] Good NonSpecific

152) 36 [2003-02-18 14:31:10:00000] Good NonSpecific

A raw data query for CalcTag3 during the shutdown period generates the following:

*Raw Data Query for CalcTag3 (before recovery)*

6) 49 [2003-02-18 14:21:05:00000] Good NonSpecific

7) 49 [2003-02-18 14:21:10:00000] Good NonSpecific

8) 48 [2003-02-18 14:22:05:00000] Good NonSpecific

9) 47 [2003-02-18 14:22:10:00000] Good NonSpecific

10) 46 [2003-02-18 14:23:05:00000] Good NonSpecific

11) 45 [2003-02-18 14:23:10:00000] Good NonSpecific

12) 45 [2003-02-18 14:24:05:00000] Good NonSpecific

13) 44 [2003-02-18 14:24:10:00000] Good NonSpecific

14) 0 [2003-02-18 14:24:11:00000] Bad OffLine

A data query for the recovered CalcTag3 values once data recovery is complete generates the following. Note that data during the shutdown period is completely recovered:

***Raw Data Query for CalcTag3 (after recovery)***

6) 49 [2003-02-18 14:21:05:00000] Good NonSpecific

7) 49 [2003-02-18 14:21:10:00000] Good NonSpecific

8) 48 [2003-02-18 14:22:05:00000] Good NonSpecific

9) 47 [2003-02-18 14:22:10:00000] Good NonSpecific

10) 46 [2003-02-18 14:23:05:00000] Good NonSpecific

11) 45 [2003-02-18 14:23:10:00000] Good NonSpecific

12) 45 [2003-02-18 14:24:05:00000] Good NonSpecific

13) 44 [2003-02-18 14:24:10:00000] Good NonSpecific

14) 0 [2003-02-18 14:24:11:00000] Bad OffLine

15) 43 [2003-02-18 14:25:05:00000] Good NonSpecific

16) 42 [2003-02-18 14:25:10:00000] Good NonSpecific

17) 51 [2003-02-18 14:26:05:00000] Good NonSpecific

18) 50 [2003-02-18 14:26:10:00000] Good NonSpecific

19) 49 [2003-02-18 14:27:05:00000] Good NonSpecific

20) 49 [2003-02-18 14:27:10:00000] Good NonSpecific

21) 48 [2003-02-18 14:28:05:00000] Good NonSpecific

22) 56 [2003-02-18 14:28:10:00000] Good NonSpecific

23) 55 [2003-02-18 14:29:05:00000] Good NonSpecific

24) 54 [2003-02-18 14:29:10:00000] Good NonSpecific

25) 54 [2003-02-18 14:30:05:00000] Good NonSpecific

26) 53 [2003-02-18 14:30:10:00000] Good NonSpecific

27) 52 [2003-02-18 14:31:05:00000] Good NonSpecific

28) 51 [2003-02-18 14:31:10:00000] Good NonSpecific

29) 49 [2003-02-18 14:32:05:00000] Good NonSpecific

30) 49 [2003-02-18 14:32:10:00000] Good NonSpecific

31) 48 [2003-02-18 14:33:05:00000] Good NonSpecific

32) 47 [2003-02-18 14:33:10:00000] Good NonSpecific

# Examples of Calculation Formulas

## Converting a Collected Value

The following code sample converts a temperature value from degrees Celsius to degrees Fahrenheit.

```
Result=CurrentValue("Temp F")*(9/5)+32
```

## Calculations Inside Formulas

The following code sample contains a calculation within a formula. In this case, we are taking the average of values of the tag `Simulation00001` over the previous hour. Typically, use a polled trigger to schedule the execution of the formula.

```
Result=Calculation("Simulation00001","Average","Now-1hour","Now",Quality)
```

## Conditional Calculation

The following code sample stores the value of a tag only if it is 100.

```
IF CurrentQuality("Simulation00001")=100 THEN

Result=CurrentValue("Simulation00001")

END IF
```

## Combining Tag Values and Assigning a Trigger

The following code sample adds current values of multiple tags using two calculation triggers.

```
Result=CurrentValue("SERVER1.Simulation00003")+CurrentValue("SERVER1.Simulation00006")
```

The calculation triggers used in the sample are SERVER1.Simulation0003 and SERVER1.Simulation0006. The calculation is triggered if the value of either Server1.Simulation0003 or Server1.Simulation0006 changes.

## Using CreateObject in a Formula

The following code sample reads data from another Historian Server using the Historian OLE DB provider, and stores it in a destination tag. When using this example, specify the username and password.

```
'connection and recordset variables

Dim Cnxn

Dim rsCurrentValueFromOtherServer

'open connection

Set Cnxn = CreateObject("ADODB.Connection")
```

```
'connect to default server using current username and password

'establish connection

Cnxn.Open "Provider=ihOLEDB.iHistorian.1;User Id=;Password="

'Create and open first Recordset using Connection execute

Set rsCurrentValueFromOtherServer = CreateObject("ADODB.Recordset")

'Get the value from the other server

Set rsCurrentValueFromOtherServer = Cnxn.Execute("select value from ihRawData

where SamplingMode=CurrentValue and tagname = Simulation00001")

'Set the result to the current value of other tag

Result=rsCurrentValueFromOtherServer("Value")

'Clean up

IF rsCurrentValueFromOtherServer.State = adStateOpen THEN

rsCurrentValueFromOtherServer.Close

END IF

IF Cnxn.State = adStateOpen THEN Cnxn.Close

END IF

Set rsCurrentValueFromOtherServer = Nothing

Set Cnxn = Nothing
```

## Using a File

The following code sample shows how to read and write text files during a calculation. You may have data in a file to use as input to a calculation, or you may want to write debug values to a text file instead of using the `LogMessage` function.

```
Dim filesys, writefile, count,readfile

'need to create a file system object since there is no

'file I/O built into VBScript

Set filesys = CreateObject("Scripting.FileSystemObject")

'open the text file, or create it if it does not exist

set readfile = filesys.OpenTextFile("C:\somefile.txt", 1, true)

'try to read from the file

IF readfile.AtEndOfLine <> true THEN

count= readfile.ReadAll

END IF

'add one to the number stored in the count count = count+1

'close the file for reading

readfile.Close

'open the same file but for writing
```

```
Set writefile= filesys.OpenTextFile("C:\somefile.txt", 2, true)

'write the updated count writefile.Write count

'close file for writing

writefile.Close

Result = count
```

## Converting a Number to a String

If your device and collector expose data as numeric codes, you can change to a string description. This examples also demonstrates that a calculation can output a string.

```
DIM X

x=CurrentValue ("tag1")

select case x

case 1

Result="one"

case 2

Result="two"

case else

Result="other"

End select
```

## Detecting Recovery Mode Inside a Formula

The following code sample detects the recovery mode or recalculation inside a formula. If there are individual tags, you do not want to perform a recovery.

```
Dim MAXDIFF, TimeDiff

'Maximum difference in timestamps allowed (Must be > 2,

'units = seconds) MAXDIFF = 10

'Calculate time difference

TimeDiff = DateDiff("s", CurrentTime(), Now)

'Compare times, if difference is < MAXDIFF seconds perform calc

If TimeDiff < MAXDIFF Then

'Place calculation to be performed here:

Result = CurrentValue("DENALI.Simulation00001") Else

'Place what is to be done when no calc is performed here

Result = Null

End If
```

## Looping Through Data Using the SDK

The following code sample uses the SDK to perform a query on a data set. It determines the minimum raw value over a one-hour time period.

```
on error resume next

Dim MyServer 'As Historian_SDK.Server

Dim I

Dim J

Dim K

Dim strComment

Dim lngInterval

Dim TagCount

Dim strDataQuality

Dim iDataRecordset

Dim iDataValue

Dim lEndTime, lStartTime, lNumSamples

Dim lNumSeconds, lNumSamplesPerSecond

Dim RawMin

'Instantiate The SDK

Set MyServer = CreateObject("iHistorian_SDK.Server")

'Attempt Connection

If Not MyServer.Connect("DENALI", "administrator","") Then

result = err.description

else

Set iDataRecordset = MyServer.Data.NewRecordset

'Find the number of samples.

'build query

With iDataRecordset

.Criteria.Tagmask = "EIGER.Simulation00001"

.Criteria.StartTime = DateAdd("h",-1,Now)

.Criteria.EndTime = Now

.Criteria.SamplingMode = 4 'RawByTime

.Criteria.Direction = 1 'forward

.Fields.AllFields

'do query

If Not .QueryRecordset Then

result = err.description
```

```
End If

'Some Large number so that real samples are less

RawMin = 1000000

For I = 1 To iDataRecordset.Tags.Count

For J = 1 To iDataRecordset.Item(I).Count

Set iDataValue = iDataRecordset.Item(I).Item(J)

' if the value is good data quality

if iDataValue.DataQuality = 1 then

if iDataValue.Value < RawMin then

rawMin = iDataValue.Value

end if

end if

lNumSamples = lNumSamples + 1

Next

Next

End With

End If

Result = RawMin

'Disconnect from server

MyServer.Disconnect
```

## Using an ADO Query

The following code sample uses a query combining Historian data with ADO data. In the example, you convert a collected value, number of barrels per day (`BarrelsUsedToday`), to a dollar amount. The code then obtains the price per barrel (`CostOfBarrel`) from the SQL server, and finally stores the total dollars in an integer tag (`TotalCostToday`).

You can also do this with a linked server and the Historian OLE DB provider, but this example maintains a history of the results.

```
Dim CostOfBarrel, BarrelsUsedToday, TotalCostToday

'Calculate the total number of barrels used over

'the previous 24hours.

BarrelsUsedToday = Calculation("BarrelsUsedTag","Total","Now 1Day","Now",Quality)

'Retrieve cost per barrel used

Dim SQLExpression

Dim Cnxn

Dim rsCurrentValue
```

```
SQLExpression = "SELECT Barrel_Cost AS Value1 FROM RawMaterial_Costs WHERE Barrel_Type = CrudeOil and

samplingmode = CurrentValue"

'open connection

Set Cnxn = CreateObject("ADODB.Connection")

'connect to default server using current username and password

'establish connection

Cnxn.Open "Provider=SQLOLEDB.1;User ID=sa; Password=;Initial Catalog=Northwind"

'Create and open first Recordset using Connection execute

Set rsCurrentValue = CreateObject("ADODB.Recordset")

'Get the value from the other server

Set rsCurrentValue= Cnxn.Execute(SQLExpression)

'Set the result to the current value of other tag

CostOfBarrel = rsCurrentValue("Value1")

'Clean up

If rsCurrentValue.State = adStateOpen then

rsCurrentValue.Close

End If

If Cnxn.State = adStateOpen then

Cnxn.Close

End If

Set rsCurrentValue = Nothing

Set Cnxn = Nothing

'Retrieve number of barrels used

BarrelsUsedToday = Calculation("BarrelsUsed","Count","Now 1Day","Now",Quality)

'Calculate total cost of barrels today

TotalCostToday = CostOfBarrel * BarrelsUsedToday
```

## Windows Performance Statistics Physical Memory Usage

The following code sample creates a formula that collects data reflecting private byte usage.

```
`Get a reference to the local data archiver process object

Set RawProc = GetObject("winmgmts:Win32_PerfRawdata_Perfproc_process.name='ihDataArchiver.'")

`Scale the virtual bytes number to a value within

`the tag's EGU range

result =RawProc.PrivateBytes *.001
```

## Windows Performance Statistics Virtual Memory Usage

The following code sample creates a formula that collects data reflecting virtual byte usage.

```
`Get a reference to the local data archiver process object

                Set RawProc = GetObject("winmgmts:Win32_PerfRawdata_Perfproc_process.name='ihDataArchiver.'")

`Scale the virtual bytes number to a value within the

`tag's EGU range

result =RawProc.VirtualBytes *.0001
```

## Determining Collector Downtime

The following code sample determines the amount of downtime, in seconds, that the Calculation collector has experienced over the last day. Downtime occurs when there are two consecutive bad quality data points for the pulse tag. If the last known data point for the pulse tag is bad quality, all the time between its timestamp and the current time is regarded as downtime. In the following sample, the pulse tag is configured to be polled, with a collection interval of one day.

```
Dim pulseTag, totalDownTime, startTime, endTime

Dim prevTime, prevQuality, lastPrevTime, lastPrevQuality

pulseTag = "calcPulseTag"

totalDownTime = 0

endTime = CurrentTime()

startTime = DateAdd("d", -1, endTime)

lastPrevTime = curTime lastPrevQuality = 0

Do

  'get the timestamp and quality of the tag value previous to the last one we checked

  On Error Resume Next

  prevTime = PreviousTime(pulseTag, lastPrevTime)

  If Err.Number <> 0 Then

    'no more values for this tag exit gracefully

    Exit Do

End If

prevQuality = PreviousQuality(pulseTag, lastPrevTime)

'if we have two consecutive bad data points, add to the downtime

If prevQuality = 0 And lastPrevQuality = 0 Then

  If prevTime > startTime Then

    totalDownTime = totalDownTime + DateDiff("s", prevTime, lastPrevTime)

Else

    totalDownTime = totalDownTime + DateDiff("s", startTime, lastPrevTime)
```

```
End If
End If
  'store the timestamp and quality for comparison with the next values
lastPrevQuality = prevQuality
  lastPrevTime = prevTime
Loop While lastPrevTime > startTime
Result = totalDownTime
```

## Analyzing the Collected Data

The following code sample analyzes the collected data to determine the amount of time that a condition was true and had good quality in the last day.

```
Dim tagName, startTime, endTime
tagName = "testTag"
startTime = "Now 1Day"
endTime = "Now"
Result = CalculationFilter(tagName, "TotalTimeGood", startTime, endTime, 100, tagName, "AfterTime", "Equal", 1)
```

## Simulating Demand Polling

To simulate demand polling, create the following tags.

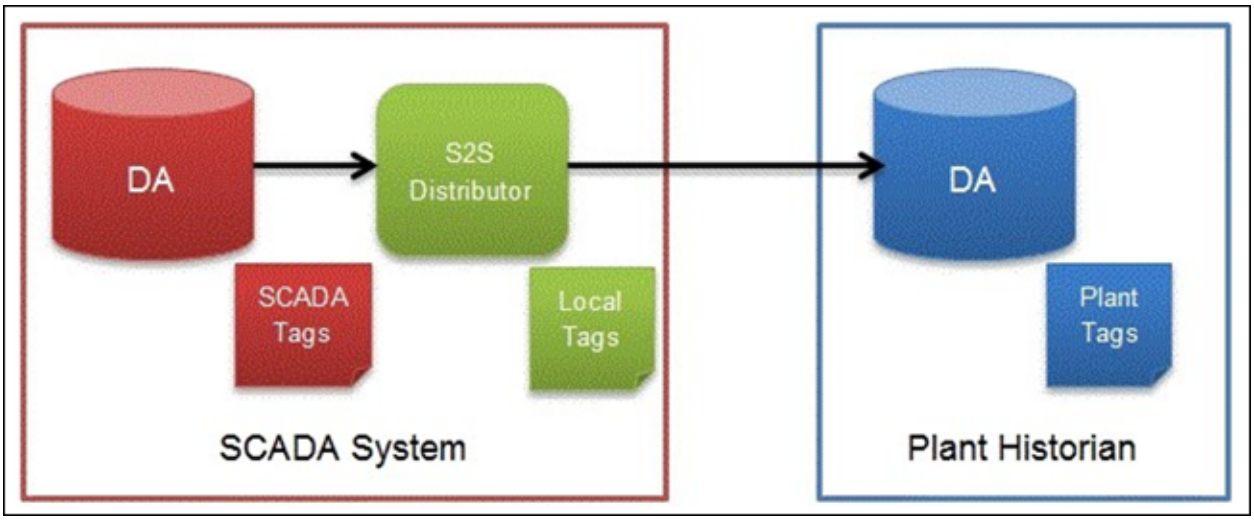
| Tag | Description |
|---|---|
| Polled Tag | A polled tag with a collection interval of the longest period you want between raw samples. Do not enable collector or archive compression. This tag should point to the same source address as the unsolicited tag. |
| Unsolicited Tag | An unsolicited tag with a 0 or 1 second collection interval. This tag ensures you will be notified whenever changes occur. This tag should point to the same source address as the polled tag. |
| Combined Tag | An unsolicited calculation tag that is triggered by either the polled tag or the unsolicited tag, and combines the raw samples of both into a single tag. Use a 0 or 1 second collection interval and use the following formula:<br>`dim timetag1`<br>`dim timetag2`<br>`dim tag1`<br>`dim tag2`<br>`Dim x` |

| Tag | Description |
| --- | --- |
| | tag1 = "T20.di-1.F_CV"<br>tag2 = "t20.T20.DI-1.F_CV"<br>x = DateAdd("s", 1,CurrentTime) ' add 1 second to calc time<br>' Get the timestamp of the newest raw sample for tag1:<br>timetag1 = previousTime(tag1, x)<br>' Get the timestamp of the newest raw sample for tag2:<br>timetag2 = previousTime(tag2, x)<br>if timetag1 > timetag2 then<br>' If tag1 triggered me, then:<br>result = PreviousValue(tag1,CurrentTime)<br>else<br>' If tag2 triggered me, then:<br>result = PreviousValue(Tag1, CurrentTime)<br>end if |

# Chapter 34. The Server-to-Server Distributor

## Overview of the Server-to-Server Distributor

The Historian Server-to-Server distributor is used to send data from a smaller Historian server to a larger, centralized Historian server. You can then use this data for reporting and analytics.




You can use either the Server-to-Server collector or the Server-to-Server distributor to send data to a central Historian. However, using the Server-to-Server distributor has the following advantages:

- It simplifies the process of configuring tags at the destination Historian.
- It provides more flexibility at the SCADA level for tag configuration compared to the Server-to-Server collector.
- It allows you to manage tags both from the source and destination Historian servers, whereas the Server-to-Server collector allows you to manage tags only from the destination Historian server.

You cannot, however, use the Server-to-Server distributor to send data to a cloud destination. Use the Server-to-Server collector for this purpose.

**Features**

- You can browse the source for tags and their attributes.
- Only the unsolicited data collection is supported; polled collection is not supported.
- The supported timestamp resolution is 100 milliseconds.
- The collector accepts device timestamps.
- The collector supports data compression.
- Floating point, integer, and string data are supported.

**Limitations**

- The Server-to-Server distributor forwards only raw data samples, messages, and alarms. It does not perform any calculations on the data.

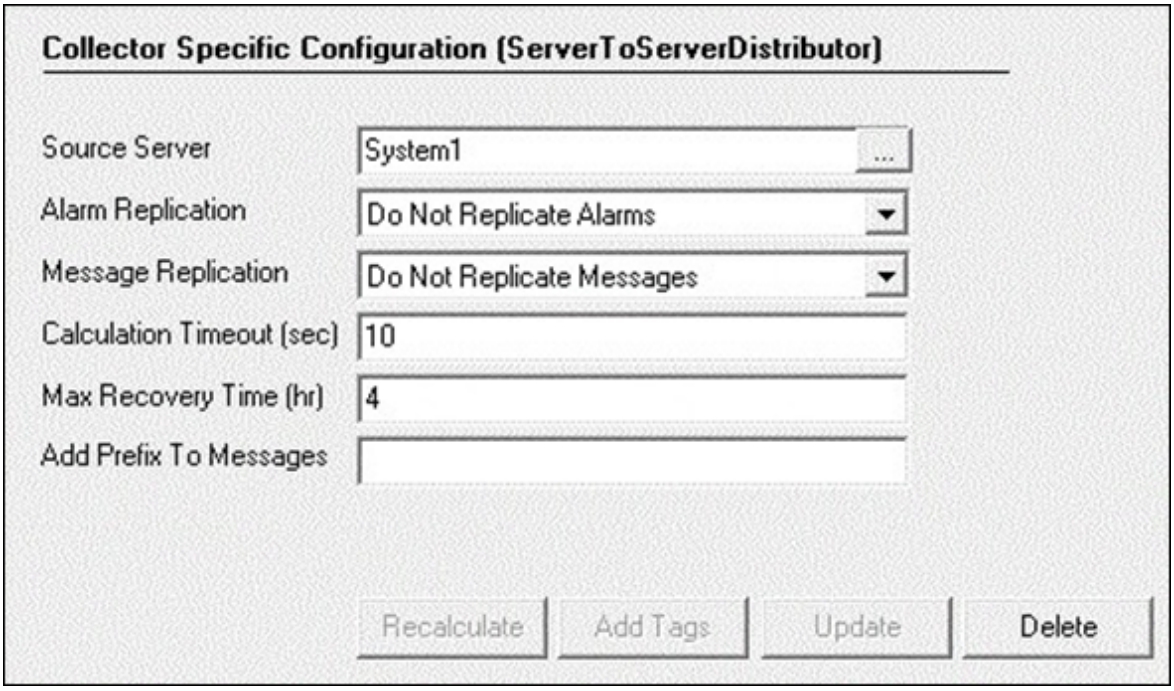
# Workflow for Using the Server-to-Server Distributor

To use the Server-to-Server distributor, you must perform the following tasks:

| Number | Task | Notes |
|---|---|---|
| 1 | Install the collectors *(on page 142)* on the source Historian server. | This step is required. This will place the collector binaries on the machines. |
| 2 | Add an instance of the Server-to-Server distributor *(on page 563)* on the source Historian server. | This step is required. |
| 3 | Start the Server-to-Server distributor *(on page 637)*. | This step is required. |
| 4 | Configure the Server-to-Server distributor *(on page 563)*. | This step is required only if you want to change the default values. |
| 5 | Create a destination tag. You can do so by browsing for the tag *(on page 672)*, adding it manually *(on page 473)*, or copying a tag *(on page 700)*. | This step is required. The naming convention of the tag is `nodename.<name of the tag>`. If configured, the tag will contain a prefix. |

## Configure the Server-to-Server Distributor

1. Access Historian Administrator *(on page 822)* on the source Historian server.
2. Select the Server-to-Server distributor from the list of collectors, and then select **Configuration**. The **Collector Specific Configuration (ServerToServerDistributor)** section appears.




3. Provide values as specified in the following table, and then select **Update**.

| Field | Description |
|---|---|
| **Source Server** | The source Historian server that you want to use. |
| **Alarm Replication** | Indicates whether you want all the alarm data to be transferred from the source server to the destination server. If you enable alarm replication, you must also enable alarm recovery. However, if you set the **Max Recovery Time** value to zero, alarm recovery does not happen. |
| **Message Replication** | Indicates whether you want message data to be transferred from the source server to the destination server. If you enable message replication, you must also enable message recovery. However, if you set the **Max Recovery Time** value to zero, message recovery does not happen. |
| **Calculation Timeout (sec)** | The maximum time allowed for a tag's calculation formula to execute before being terminated. The default value is 10 seconds. |
| **Max Recovery Time (hr)** | The maximum duration, in hours, for which the collector will attempt to restore data during recovery logic. The default value is 4 hours. |
| **Add Prefix to Messages** | The prefix to identify replicated messages on the destination.<br><br>Alarms and events data will automatically have a prefix added to it with the following syntax: `MachineName_Datasource` |

| Field | Description |
|---|---|
| | For example, if your alarm is forwarded from the server `Almserver12` with a data source named `OPCAE`, the prefix will be `Almserver12_OPCAE`. |

The Server-to-Server distributor is configured.

# Chapter 35. The Simulation Collector

## Overview of the Simulation Collector

The Simulation collector generates random numbers and string patterns for demonstration purposes. You can configure the number of tags that you want to generate.

**Features:**

- The collector generates random scaled values between 0 and 32,767. It uses the high and low engineering units fields of each tag to scale the 0 to 32,767 pre-set values into appropriate engineering units.
- The collector also provides five-string simulation tags that generate random alphanumeric data.
- In addition to generating random values, the collector can generate sequential values for some tags. For a list of such tags, refer to Tags with Sequential Values *(on page 2560)*.
- You can import browse for tags and their attributes.
- The supported timestamp resolution is 1ms.
- Floating point, integer, and string data are supported. Binary data is not supported.
- You can create Python Expression tags.
- Only polled data collection is supported with a minimum poll interval of 100ms.

**Note:**
You can create more simulation string tags by manually adding string tags with the following naming convention to the collector: `CollectorName.Simulation.StringXXXX`

**Supported Tag Attributes:**

- Tagname
- Data Type
- Hi Engineering Units
- Lo Engineering Units
- Hi Scale
- Lo Scale

# Configuration

## Configure the Simulation Collector Using Configuration Hub

Install collectors *(on page 143)*, and create an instance of the collector *(on page 567)*.

1. Access Configuration Hub *(on page 336)*.
2. Select **Collectors**, and then select the File collector instance that you want to configure.
   The fields specific to the collector instance appear in the **DETAILS** section.
3. Enter values as specified in the following table.

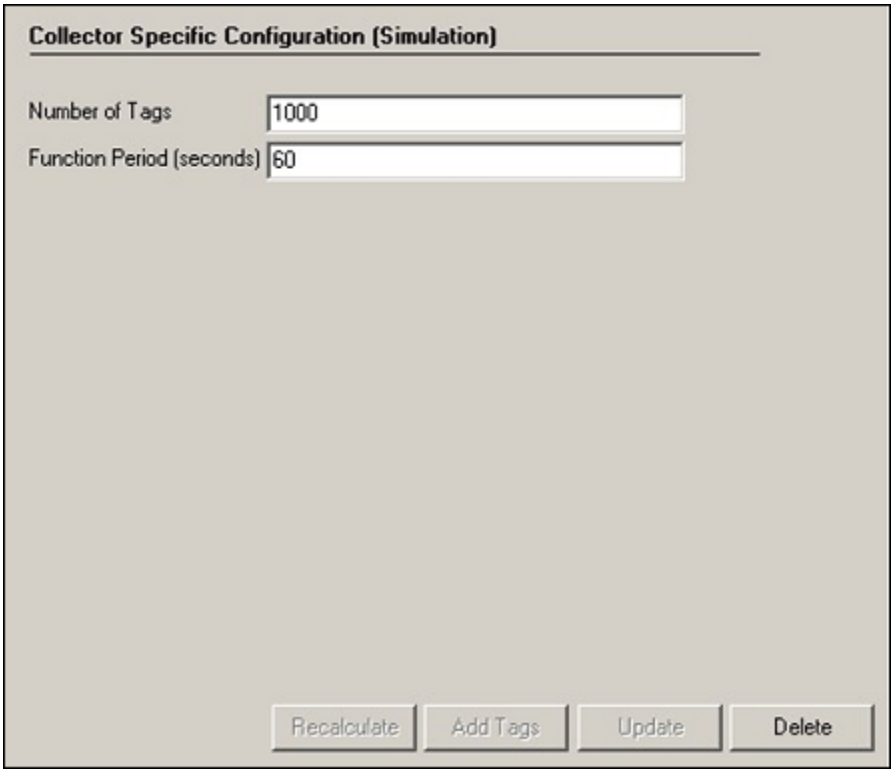
| Field | Description |
|---|---|
| Number of Tags | The number of Historian tags that you want the create for the collector. |
| Function Period (seconds) | The period, in seconds, of the `SIN`,`STEP`, and `RAMP` functions implemented in the collector. |
| **MTLS Security** | Indicates whether you want to use Mutual TLS (MTLS) protocol to enforce a secure and strong authentication mechanism. |
| **MTLS Data Encryption** | Indicates whether you want to encrypt the data that the collector shares to the data archiver (DA). |
| For more information on how to enable MTLS Security, refer to Enable MTLS Security *(on page 632)*. | |

4. As needed, enter values in the other sections common to all collectors *(on page 579)*.
5. Restart the collector.
   The collector instance is configured.

## Configure the Simulation Collector Using Historian Administrator

Install collectors *(on page 143)*, and create an instance of the collector *(on page 567)*.

1. Access Historian Administrator *(on page 823)*.
2. Select **Collectors**, and then select the Simulation collector instance that you want to configure.
   The fields specific to the collector instance appear.




3. Enter values as specified in the following table.

| Field | Description |
|---|---|
| Number of Tags | The number of Historian tags that you want the create for the collector. |
| Function Period (seconds) | The period, in seconds, of the `SIN`,`STEP`, and `RAMP` functions implemented in the collector. |

4. As needed, enter values in the other sections *(on page 579)*.
5. Restart the collector.
   The collector instance is configured.

# Tags with Sequential Values

In addition to generating random values, the Simulation collector generates sequential values. The following table provides a list of tags for which the collector generates sequential values. All the tags have a range of 0 to 1000.

| Tags | Description |
|---|---|
| Constant | Maintains a constant value. |

| Tags | Description |
|---|---|
| Constant_1%Noise | Same as Constant, but produces 1% random noise. |
| Constant_5%Noise | Same as Constant, but produces 5% random noise. |
| Constant_20%Noise | Same as Constant, but produces 20% random noise. |
| Ramp | Steadily increases value every polling period to create a smooth upward trend. |
| Ramp_1%Noise | Same as the Ramp tag, but produces 1% random noise. |
| Ramp_5%Noise | Same as Ramp tag, but produces 5% random noise. |
| Ramp_20%Noise | Same as Ramp tag, but produces 20% random noise. |
| Sin | Produces a Sine wave centered on a value of 500. |
| Sin_1%Noise | Same as Sine tag, but produces 1% random noise. |
| Sin_5%Noise | Same as Sine tag but produces 5% random noise. |
| Sin_20%Noise | Same as Sine tag but produces 20% random noise. |
| Step | Produces a step trend centered on a value of 500. |
| Step_1%Noise | Same as Step tag, but produces 1% noise. |
| Step_5%Noise | Same as Step tag, but produces 5% noise. |
| Step_20%Noise | Same as Step tag, but produces 20% noise. |

# Chapter 36. Windows Performance Collector

## Windows Performance Collector

### Overview of the Windows Performance Collector

The Windows Performance Collector collects Windows performance counter data and sends it to the Historian server for archival. The data collected can be used to monitor and assess the performance and efficiency of the computer running the Historian software and also assess the status of the Historian Archiver and other system statistics derived by the Historian counters. You can collect almost any Windows performance counter that is visible in Windows Performance Monitor and thereby determine if there are any issues with the operating system or the computer that affects the performance.

While both the Windows Performance Monitor and Windows Performance Collector can collect the performance counters data, the collector provides the advantage of storing it in Historian archives, making it easy to view in Historian clients and compare to other data stored in Historian. The archived values can be viewed as the Last 10 Values in Historian Administrator or included in an Excel report together with other Historian data.

You can also use this collector to collect performance data from other GE Intelligent Platforms products such as CIMPLICITY, iFIX, and also Historian's own performance counters.

The Windows Performance Collector collects polled data only and creates Historian tags with the data type that best matches the data type of the performance counter being collected.

**Windows Performance Collector - Requirements**

To use a Windows Performance Collector, you require Historian 6.0 or higher installed on the following machines:

- The computer running the collector.
- The computer running Data Archiver.
- The computer running the Administrator.

The Windows Performance Collector can only collect performance counters from the local computer it is running on. You can run only one Windows Performance collector on a computer.

### Windows Performance Collector Feature Summary

The following table outlines the features of the Windows Performance Collector.

| Feature | Capability |
|---|---|
| Minimum Poll Interval | Yes- 1 second |
| Unsolicited Collection | No |
| Timestamp Resolution | Yes- 1 millisecond |
| Accept Device Timestamps | No |
| Floating Point Data | Yes |
| Integer Data | Yes |
| String Data | Yes |
| Binary Data | No |
| Collector Compression | Yes |
| Reads data, tags | Yes |
| Start/stop collection from the system | Yes |
| Condition based Collection Support | No |
| Python Expression Tags | Yes |

**Note:**
Though the Minimum Poll Interval supported by the Windows Performance Collector is 1 second, it is recommended to set the interval to 5 seconds when adding large number of tags for proper collection of data.

# Windows Performance Collector Configuration

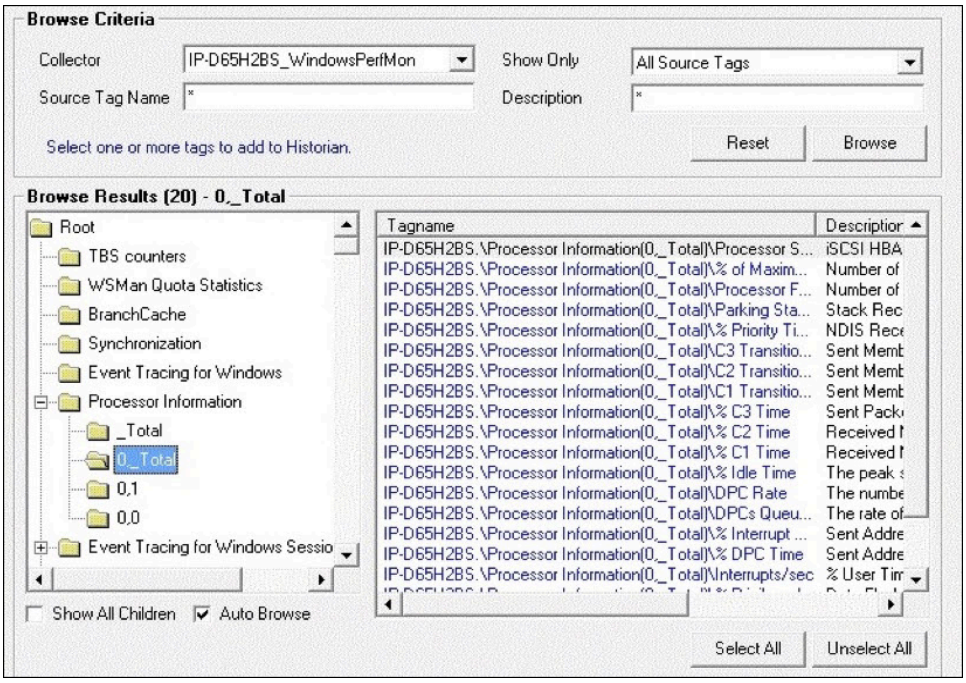
## Understanding Windows Performance Collector Tag Hierarchy

The Windows Performance Collector provides the ability to browse performance tags hierarchically. This hierarchy is different from the hierarchy of the Windows Performance Monitor.

The Windows Performance Collector displays Objects, Instances, and Counters. When you select an Object, its Instances appear in a folder structure under the specific Object. The Objects are displayed as they are added. They are not displayed in an alphabetical order as they are created dynamically. The Counters are displayed on the right. If no counters exist for a particular Instance, they are not displayed.

The following figure provides an example of the hierarchy:

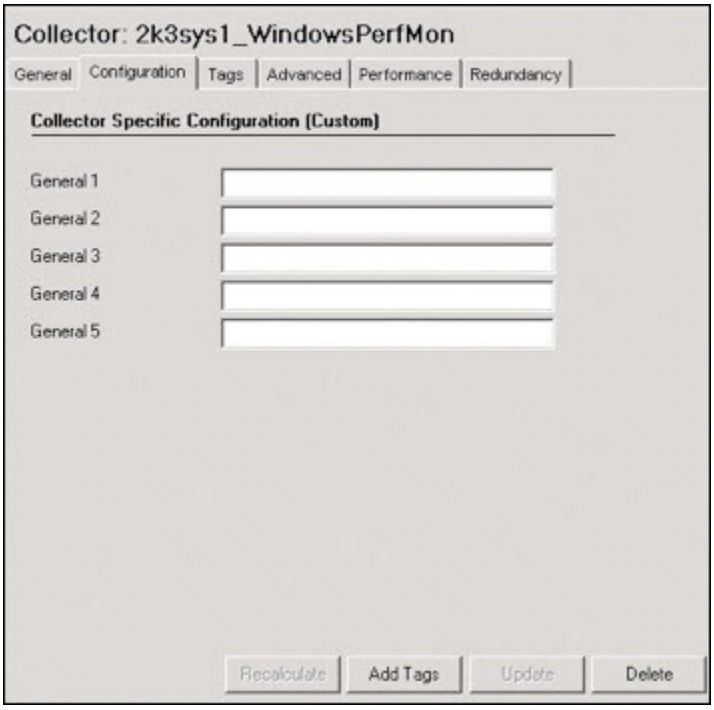
Figure 10. Windows Performance Collector Browse Criteria Screen




In the Windows Performance Collector, while you are collecting data, if any of the counters of any application or process stops running, the collector does not stop showing that particular instance but will continue running it with zero values for that particular Counter. It does not update these counters dynamically.

## The Configuration Section for Windows Performance Collector

To access the **Configuration** section for a Windows Performance collector, select the Windows Performance Collector from the list of collectors and select **Configuration**. The following figure appears.




**Note:**

The fields, General 1 through General 5 are reserved for future use.

# Chapter 37. The Wonderware Collector

## Overview of the Wonderware Collector

The Wonderware Collector gathers data samples from a Wonderware Historian 2014 R2 Server application and stores the corresponding data entries in the Historian Server.

**Note:**
Wonderware is a registered trademark of Schneider Electric Software.

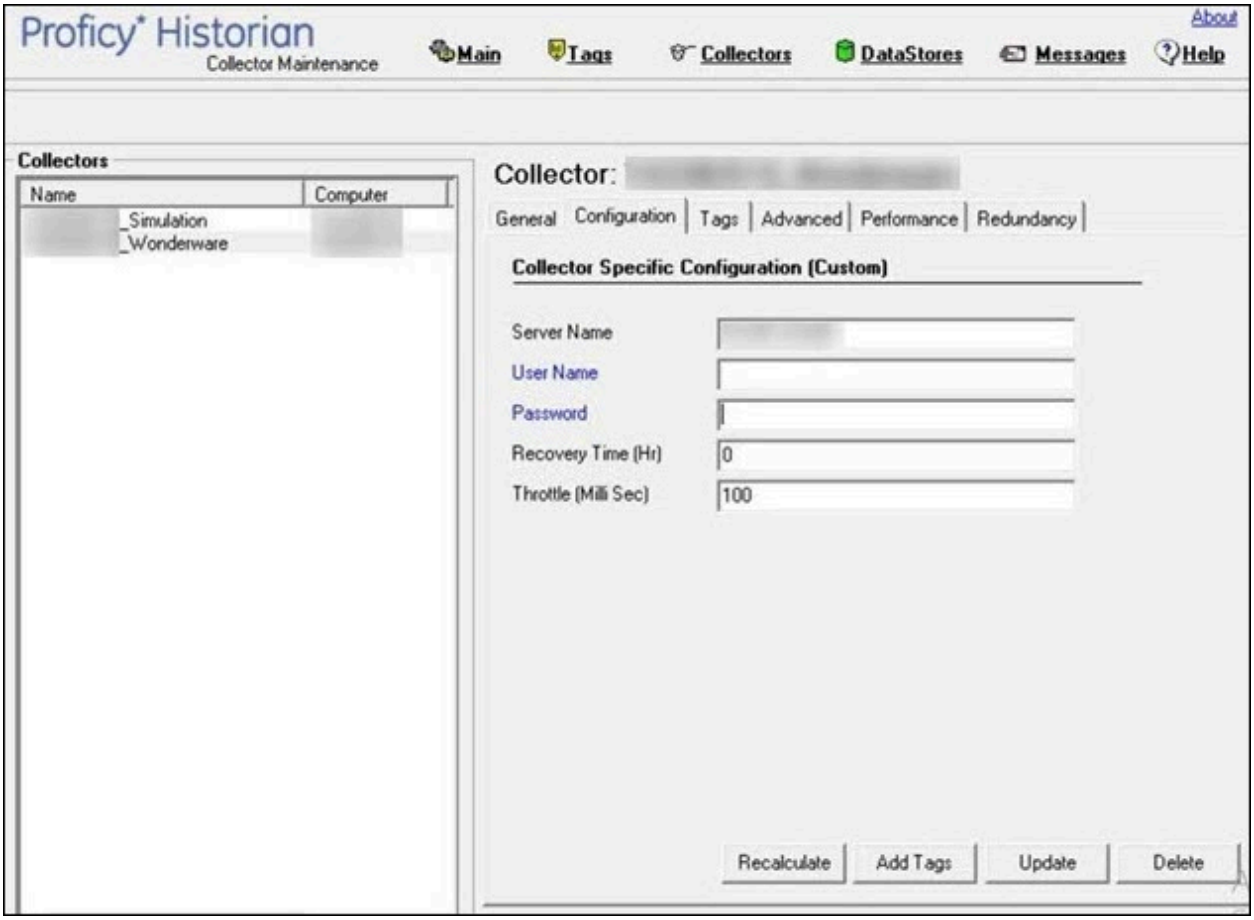
This collector supports collecting of analog, digital and string types of data from the Wonderware Historian Server. This collector supports a distributed model, where the Wonderware Historian Server, the Historian Data Collector, and GE Historian software are installed on different machines. Typically, however, the collector is installed on the same computer as the Wonderware Data Archiver and sends data to a remote GE Historian server.

The Wonderware Collector uses unsolicited collection, whereby changes to the Wonderware tags are detected, and are forwarded to the Historian server. Raw samples from the Wonderware Collector are duplicated into the GE Historian data archive.

One Wonderware Collector can collect data from a single Wonderware Historian server. To collect from multiple Wonderware Historian servers to an Historian archiver, you must install multiple collectors.

**Note:**
The ODBC Driver for the SQL Server is required for the Historian Data Collector for Wonderware installation; however, the ODBC Driver for SQL does not ship with Historian. If the ODBC Driver for SQL is not installed, the Historian Data Collector for Wonderware will not connect to the Wonderware server. If you install the Historian Data Collector for Wonderware on a machine that does not contain the Wonderware server, be sure to install the ODBC Driver for SQL on the machine with the Historian Data Collector for Wonderware.

**Limitations:** If you want a domain user to use the Wonderware Collector, after you add an instance of a collector, when you later configure it, do not provide values in the **User Name** and **Password** fields. This is because ODBC Driver uses Windows authentication.




## Installation Prerequisites

The Historian Data Collector for Wonderware requires the installation of the SQL server native client (`sqlncli.msi`), which can be downloaded from the following link:

https://support.microsoft.com/en-us/kb/2726013

## Wonderware Collector Features

The following table outlines the features of the Wonderware Collector.

| Feature | Capability |
|---|---|
| Browse Source for Tags | Yes*(on a Wonderware server that supports browsing) |
| Browse Source for Tag Attributes | Yes |
| Polled Collection | No |

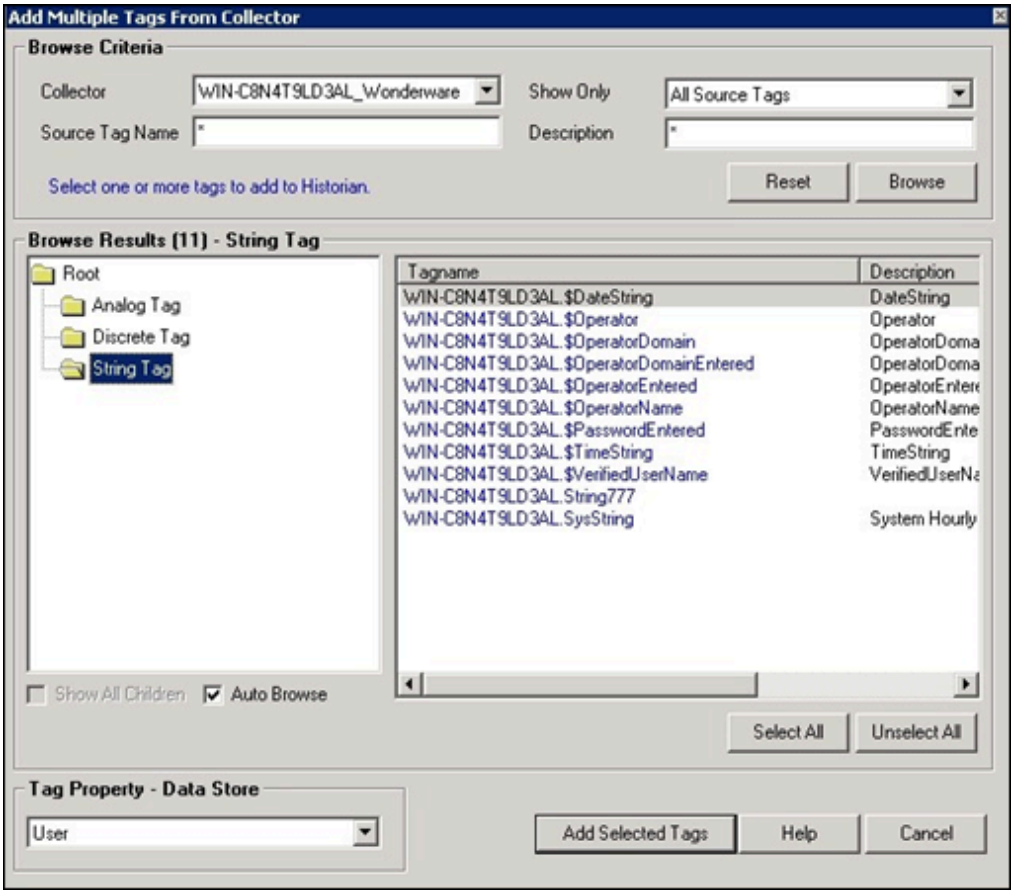
| Feature | Capability |
|---|---|
| Minimum Poll Interval | 100 ms (milliseconds) |
| Unsolicited Collection | Yes |
| Time stamp Resolution | 1 ms |
| Floating Point Data | Yes |
| Integer Data | Yes |
| String Data | Yes |
| Python Expression Tags | No |
| Time Assigned | **Note:** You must set this field to Source as the Wonderware Co only supports unsolicited tags. |

## Hierarchical Tags Available in Browse

The Schneider Electric Wonderware server supports the hierarchical organization of your tags in a tree structure. Historian uses the server's hierarchy allowing you to browse the Wonderware Collector in the Non-Web Administrator mode.

**To browse for data collector tags in a hierarchy:**

1. Browse your Wonderware data source for new Wonderware data tags.
2. From the **Collector** list, select the Wonderware Collector you wish to browse. A hierarchical tree appears in the **Browse Results** window.




3. To limit the displayed tags to only those that are not collected, from the **Show Only** list select **Source Tags Not Collected** from the drop-down menu.
4. To limit the displayed tags to match a tag name or tag description, enter the value to match in the **Source Tag Name** or **Description** text boxes.
5. Navigate to the node in the tree you want to browse, and then select **Browse**. The tags within the selected portion of the Wonderware tag hierarchy will be displayed
6. Select the tag(s) you want to add to Historian, and select **Add Selected Tags**. Collected tags appear in black text in the tag list.

> **Note:**
> If Wonderware Collector encounters null value at the time of collecting data, it will be ignored and that specific sample will not be sent to the Historian server.

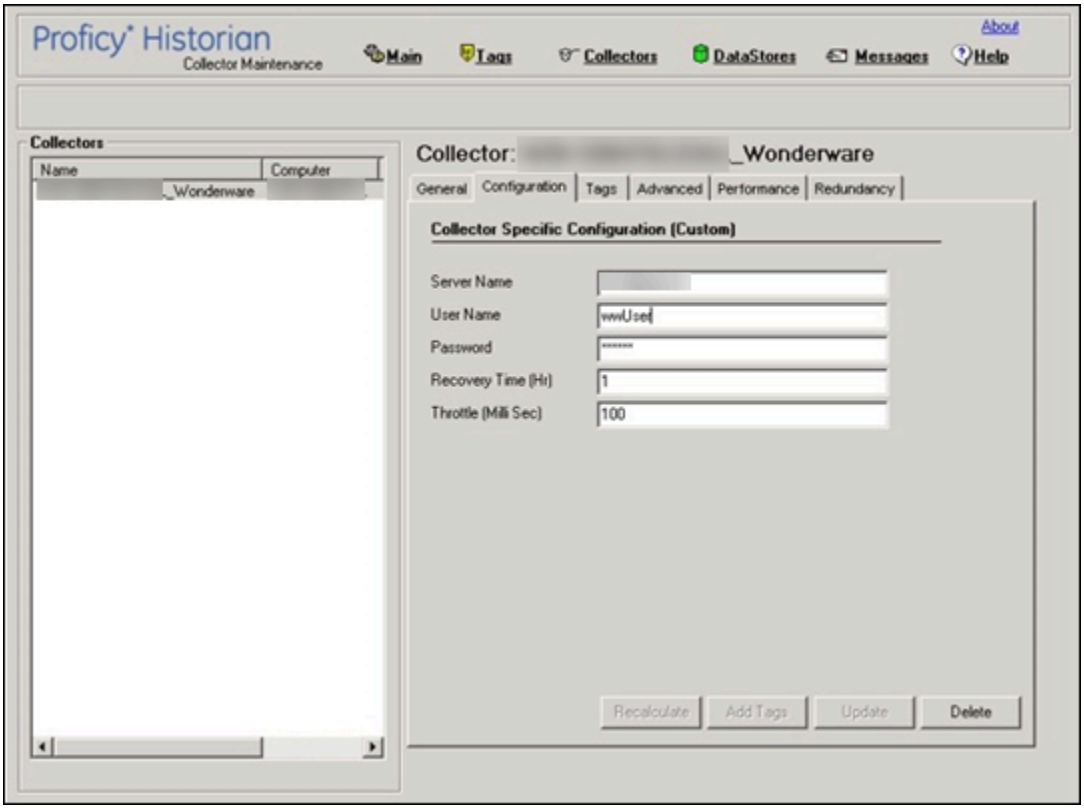
## Supported Data Types for Wonderware Collector

The following table lists the data types recommended for use with Historian.

| Data Types | Recommended Historian Data Types |
|---|---|
| Analog- EuroFloat | Double Float |
| Analog- MSFloat | Double Float |
| Analog- MSDouble | Double Float |
| Analog- Integer | Double Integer |
| Discrete | Single Integer |
| String | Variable String |

# Configuring Wonderware Collector

To access the **Configuration** section for the Wonderware Collector, select the Wonderware Collector from the list on the left of the Administrator Tool **Collectors** section and then select **Configuration**. A page similar to the following appears:




Collector-Specific Configuration for the Wonderware Collector

Enter the value for the Wonderware Collector-specific field parameters:

| Field | Description |
|---|---|
| Server Name | Wonderware Server InSQL Database Server name. |
| User Name | Wonderware Server InSQL Database User name, for example, wwUser. |
| Password | Wonderware Server InSQL Database password, for example, XXXXXX. |
| Recovery Time (hours) | Recovery logic is activated when the Wonderware Collector and Wonderware Historian re-establish a connection after a connection loss, or when the Wonderware Collector is started.<br><br>The Wonderware Collector attempts to recover all data samples between the current time and the last known write time, up to a maximum number of hours configured for the collector. Continuous collection resumes only after the previous data has been recovered.<br><br>**Note:**<br>The default recovery time is 0 hours. |
| Throttle (Milliseconds) | Frequency of Wonderware data polling.<br><br>To minimize the load on the Wonderware Server, the configurable throttling option is provided by the Wonderware Collector. By default, the Wonderware Collector tries to query the tag data every 100 milliseconds based on the collection interval time. You can change this value to any time between 100 milliseconds to 16 hours.<br><br>**Note:**<br>If **Throttle** field is blank, enter the required minimum value of 100 milliseconds. |

## Data Recovery

**Note:**
We recommend that the collector for which the data recovery is intended is in the same time zone as the server. If there is a mismatch, there is a possibility that data recovery will be incomplete.

### Automatic Data Recovery

In this mode, data is automatically recovered since the last time data has been collected.

**How it works:**

1. The collector determines the duration between the current time and the last time data has been written to the Historian data archive, which is stored in the LastSampleWriteTime registry key.
2. It compares this duration with the value in the Recovery Time field specified in the collector settings *(on page 2347)*.
3. It uses the shorter duration to perform a raw data query on all the tags.
4. It then processes the returned samples in chronological order.

For example, if the collector was stopped for 8 hours, but Max Recovery Time was 4 hours, only 4 hours of data would be recovered.

As per the recovery logic, an end-of-collection marker is placed at the point in time where the collector was stopped. This end-of-collection marker may or may not be there after the recovery is complete. As part of the recovery logic, if recovery data point time matches the timestamp of the end-of-collection marker, it is overwritten with the recovered good data.

### Manual Data Recovery

In this mode, you can fill gaps in the data, but you cannot fill old data.

**To perform a manual recovery:**

1. Access Historian Administrator.
2. Select **Collectors**, and then select the OPC Classic HDA collector instance for which you want to manually recover data.
3. Select **Recalculate**.

   The **Recalculate** window appears.
4. Enter start time, end time, and other required information. We recommend that you choose small time intervals to reduce the load on the server and the collector.
5. Select **Recalculate**.

The tag data is recalculated. After the manual recalculation begins, the collector recovers data of the selected tags data from the collector, and sends it to Historian between the start time and end time.

At the time of recovery, if the connection to server is lost, and if the reconnect mechanism is enabled, the collector will try to connect to the server and fetch the data once connection re-establishes.

### Manual Data Recovery

Assume that the collector is connected to Historian for the first time today and the archive was created at 10 am. The user initiates manual recalculation from 1am to 2 am. For that time interval, the archives were not even created. With respect to Historian, it is old unknown data and the data write fails. If there is a data gap between 1am and 2 am, manual recalculation successfully fills the data gap.

## Initiating Manual Recovery

Manual recovery can be performed from Historian Administrator. Manual recalculate is done for filling the data gaps but not for filling old data.

It is advised to keep the Wonderware Collector in the same time zone as the Wonderware server. If there is a mismatch, there is a possibility that auto recovery of data will be incomplete.

**To initiate manual recovery:**

1. In Historian Administrator, select the Wonderware Collector.
2. Select **Recalculate**. The **Recalculate** window appears.
3. Enter the start and end time and choose all or selected tags based on the criteria from Recalculate window.
4. Select **Recalculate**.
   Once manual recalculate starts, the collector recovers selected tags data from Wonderware server to GE Historian between start time and end time.

   It is advised to choose small time intervals, so that the load on Wonderware server/collector will be reduced.

   > **Note:**
   > At the time of recovery, if the connection to Wonderware Server is lost, and if the reconnect mechanism is enabled, the collector will try to connect to the server and pull the data once connection reestablishes.

   **Example**:

   Assume that the Collector connected to GE historian for the first time today and the archive was created at 10 am. The user initiates manual recalculation from 1am to 2am.

For that time interval, the archives were not even created. With respect to Historian it is old unknown data and the data write fails.

If there is a data gap between 1pm to 2 pm, manual recalculation successfully fills the data gap.

## Reconnecting to the Wonderware Server

The Wonderware Collector supports auto-reconnect to the Wonderware Server. If the connectivity between the Wonderware server and the collector is down due to network connectivity issues, the collector will auto-reconnect to the server when the server is back and running. The collector polls for the server connection for a set time of every 5 seconds. The collector shuts down when the reconnect functionality is disabled.

**To enable Auto-Reconnect to the Wonderware Server:**

1. From the Start menu, select **Run** and type `Regedit`, then, select **OK**.
   The Registry Editor appears.
2. Open the key folder: `HKEY_LOCAL_MACHINE\SOFTWARE\GE Digital\iHistorian \Services\ WonderwareCollector`.
3. Create a new DWORD labelled, **EnableReconnect**.
4. Enter the decimal value 1.
5. Select **OK**, then close the Registry Editor.
6. Restart the Historian Data Collector for Wonderware for the change to take effect.

> **Important:**
> If this registry is not created and set to a value of 1, then the auto-reconnect functionality will not be enabled.

**To configure the timer:**

a. From the **Start** menu, select **Run** and type **Regedit**. Then, select **OK**. The **Registry Editor** appears.
b. Open the key folder: HKEY_LOCAL_MACHINE\SOFTWARE\GE Digital\iHistorian\Services\ WonderwareCollector.
c. Create a new DWORD labelled, ReconnectInterval.
d. Enter a decimal value greater than 5. This value represents the number of seconds for the collector to wait before trying to re-connect to the Wonderware Server.
e. Select **OK**, and then edit the Registry Editor. Min Value = 5 seconds (default) Max Value = 60 seconds

f. Select **OK**, and then edit the Registry Editor.
g. Restart the Wonderware Collector for the change to take effect.

# Troubleshooting the Wonderware Collector

The Wonderware Collector generates logs during initialization, configuration, and general operation. These can be found in the general logging folder `C:\Proficy Historian Data\LogFiles`.

### Troubleshooting Tips

- Be sure to run the Wonderware server before the Historian Data Collector starts up.
- If the Wonderware Collector does not start automatically, refer to the Historian log file to view log entries to determine the problem.

# Chapter 38. OLE DB Provider

## Overview of the OLE DB Provider

OLE DB is a collection of standard COM-based interfaces defined by Microsoft that abstract standard SQL commands into native API access for any data source. OLE DB adds tremendous value to Historian by providing simple access to data from within the SQL environment, without the need for complex scripting.

The Historian OLE DB provider is a data access mechanism that allows you to query Historian data using SQL statements or other client reporting tools.

**Supported Applications:** Using the OLE DB provider, you can create reports and integrate Historian with the following applications:

- Microsoft Power BI
- Seagate Crystal Reports v8.0, and above (v11.0 or above required for use with Historian Alarms and Events)
- VisiconX with iFIX v4.0 and later
- Microsoft Excel 2003 and later
- Visual Basic v6.0, Service Pack 5
- Visual Basic for Applications (VBA) v6.0
- Microsoft SQL Server v7, Service Pack 3
- Microsoft SQL Server 2008, or SQL Server Express 2008
- Oracle 8.x and above

**Note:**
Other OLE DB clients are likely to work with the OLE DB provider, but have not been tested.

**Components:** When you install the OLE DB provider on the Historian server machine, the following items are available:

- The `ihSQL.exe` file: The Historian Interactive SQL tool.
- The `Samples` folder: Contains sample reports for Crystal Reports, Microsoft Excel, Microsoft Visual Basic, iFIX, and Oracle.
- The `ihOLEDB.dll` file: A dynamic-link library file to support the OLE DB provider on 32-bit and 64-bit operating systems. This file is located in the `WINDOWS\SysWOW64` and `WINDOWS\System32` folders.

If you install the OLE DB provider on a client computer, some of these items may not be available. These items may also appear differently, depending upon the number of archivers you installed.

**Limitations:** The OLE DB provider has read-only access. You cannot insert, update, or delete data in archives using the OLE DB provider.

# Setting Up

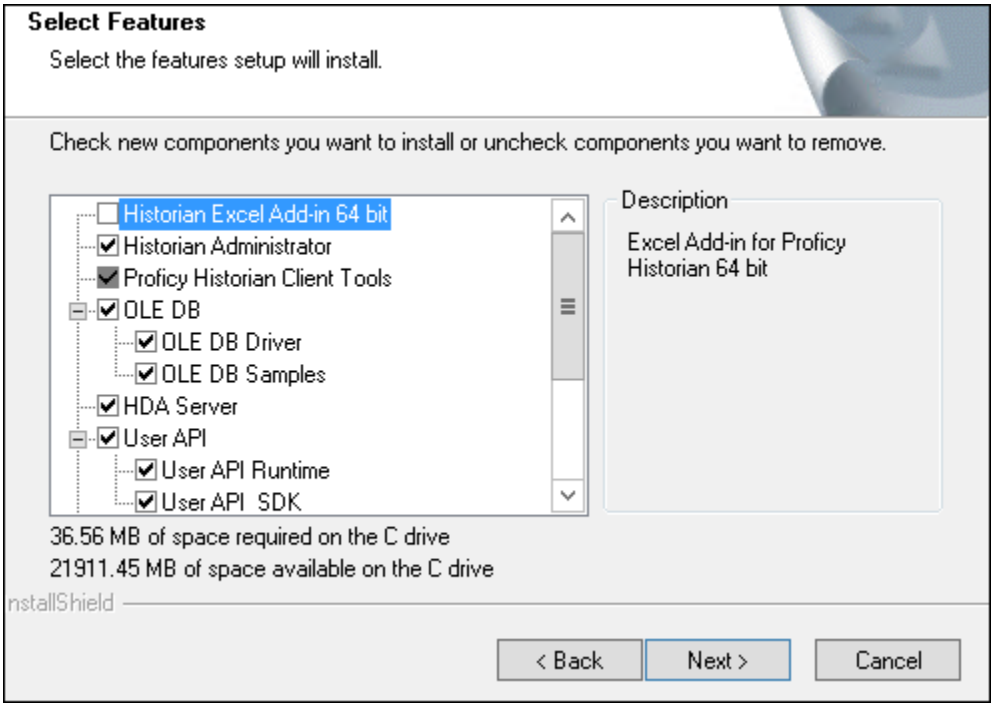
## Install Client Tools

When you install Client Tools, the following components are installed by default:

- Client Tools
- Historian Administrator
- OLE DB provider (driver and samples)
- The OPC Classic HDA server
- User API and SDK
- Historian Client Access API
- Collector Toolkit

This topic describes how to install Client Tools using the installer. You can also install it at a command prompt *(on page 153)*.

1. Run the `InstallLauncher.exe` file.
2. Select **Install Client Tools**.
   The **Select Features** page appears, displaying a list of components that you can install with Client Tools.



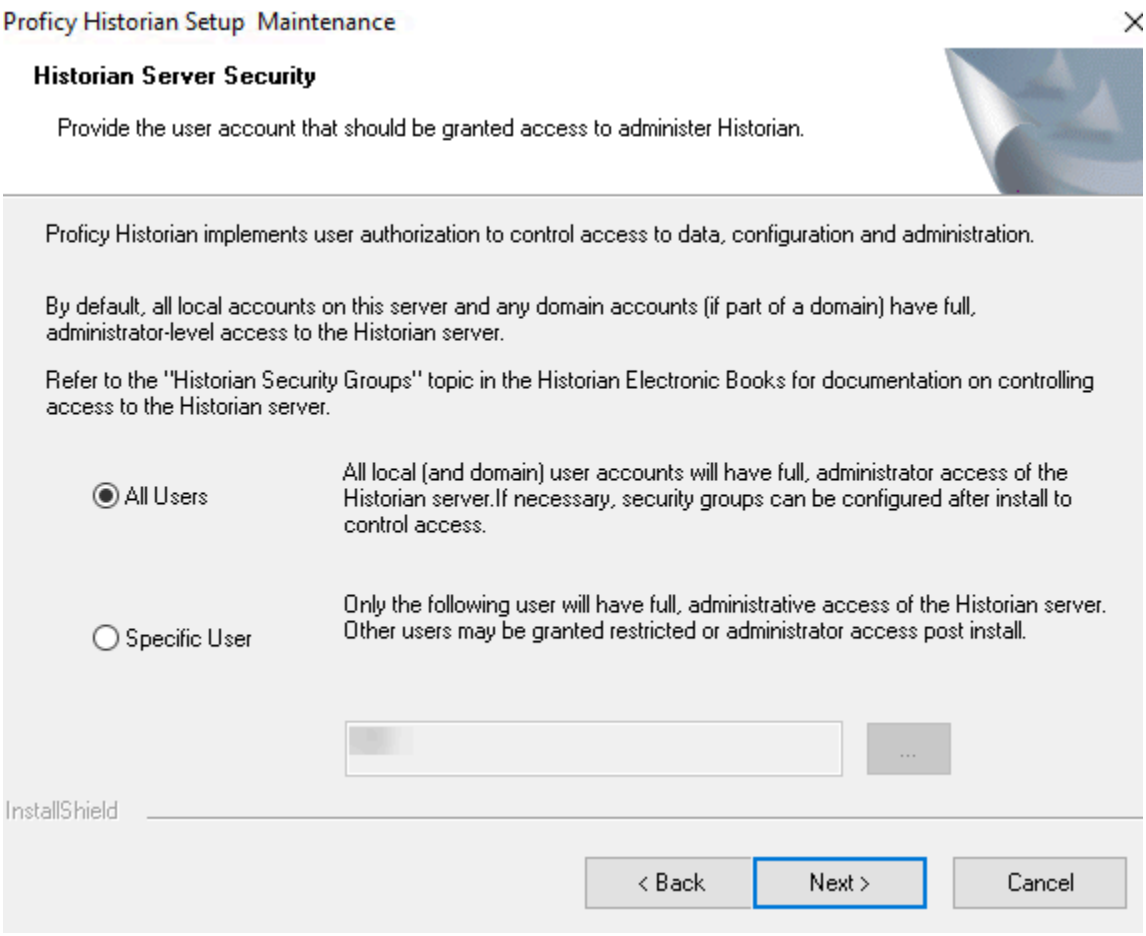

By default, the check boxes for components such as **Historian Administrator**, **HDA Server**, **OLE DB**, and **User API and SDK** are selected. If you do not want to install them at this time, clear the check boxes. You cannot, however, clear the **Proficy Historian Client Tools** check box.

**Important:**
If you are reinstalling, you must select all of the previously installed components. If you do not do so, the component will be uninstalled.

By default, the **Historian Excel Add-in 64-bit** check box is cleared. If you want to install Excel Add-In along with Client Tools installation, select the check box.

**Note:**
If using certain versions of Windows (like Windows 10 or Windows 2019), you may receive an error message while installing Excel Add-In, stating that some of the DLL files are not registered. You can ignore these messages.

The **Historian Server Security** page appears.




3. Select the user account that should be granted the access to administer Historian.
   You are provided with the following options:

| Option | Description |
|---|---|
| **All Users** | If you select this option, all local, including domain user accounts will have the administrator access of the Historian server. If needed, to control access, you can configure security groups after installation. |
| **Specific User** | If you select this option, the user that you select will have the administrator access of the Historian server. If needed, you can grant a restricted or administrator access to other users after installation. |

4. Select **Next**.
5. When you are asked to reboot your system, select **Yes**.

Client Tools, along with the selected components, are installed in the following folder: `<installation drive>:\Program Files\Proficy\Proficy Historian\x86\<tool name>`. If you have selected HDA Server, Microsoft .NET Framework 4.5 and the OPC Core Components 3.00 redistributable are installed as well.

## Connect to a Historian Server

1. Install Client Tools *(on page 150)*, which will automatically install the OLE DB provider.
2. Initialize the COM library on the machine on which you have installed the OLE DB provider.

This topic provides basic steps to connect the OLE DB provider to a Historian server so that you can import the data. For instructions specific to a client, refer to:

- Import Historian Data into Power BI Desktop *(on page 2580)*
- Import Historian Data into Crystal Reports *(on page 2588)*
- Import Historian data into Microsoft Excel *(on page 2592)*

1. To connect an OLE DB client to a local Historian server, run the following command:

```
Provider=iHOLEDB.iHistorian.1
```

2. To connect an OLE DB client to a remote Historian server, run the following command:

```
Provider=iHOLEDB.iHistorian.1;PersistSecurity Info=False;

USER ID=[<Historian server username>];

Password=[<Historian server password>];

Data Source=[<network name of your Historian server>]
```

# Working with Clients

## Power BI Desktop

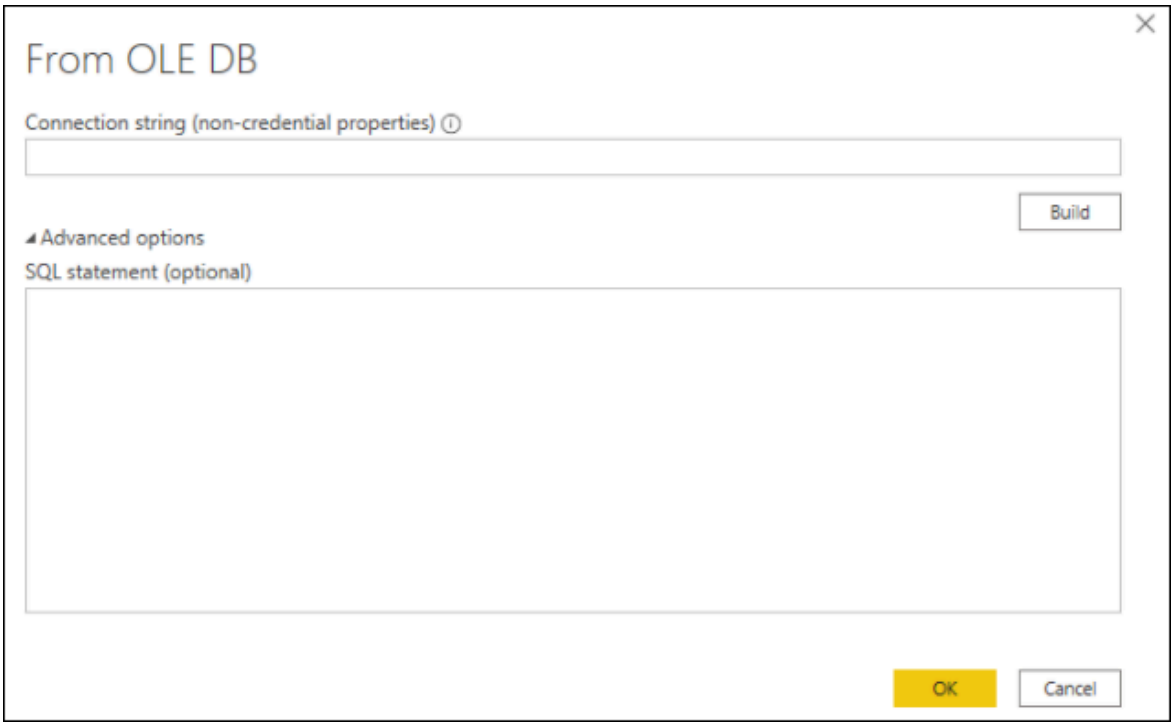
## Import Historian Data into Power BI Desktop

Microsoft Power BI Desktop is an application that transforms and visualizes data. Using this application, you can connect to multiple data sources and combine the data into a data model.

This topic describes how to import Historian data into Power BI Desktop.

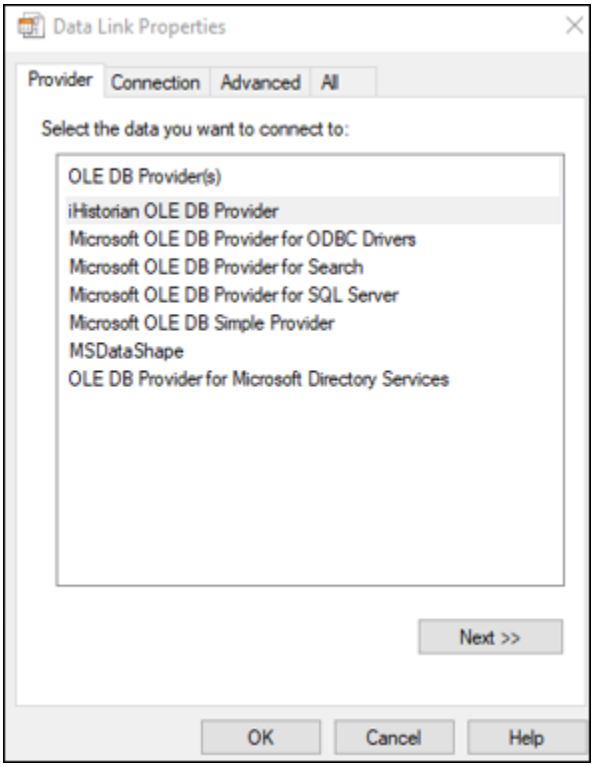
1. Access Power BI Desktop.
2. Select **Get Data > Other > OLE DB**, and then select **Connect**.

   The **From OLE DB** window appears.



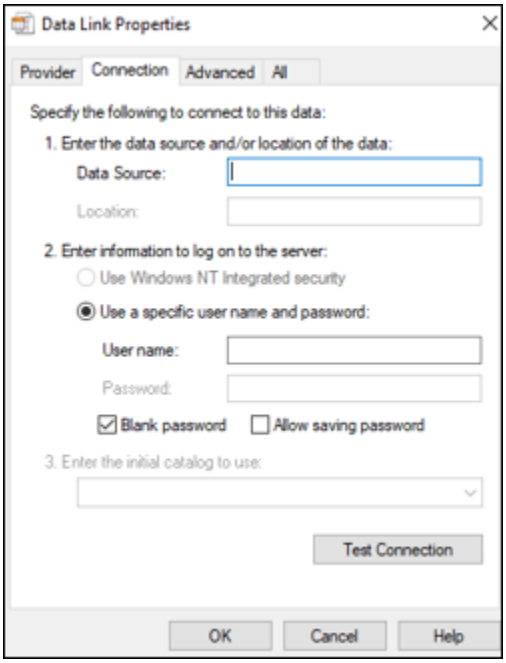

3. Select **Build**.

   The **Data Link Properties** window appears, displaying a list of the Historian OLE DB providers in the **Provider** section.




4. Select **Next**.

   The **Connection** section appears.




5. Leave the default values as is, and select **Test Connection.**

   After the connection succeeds, the connection string is populated in the **From OLE DB** window.

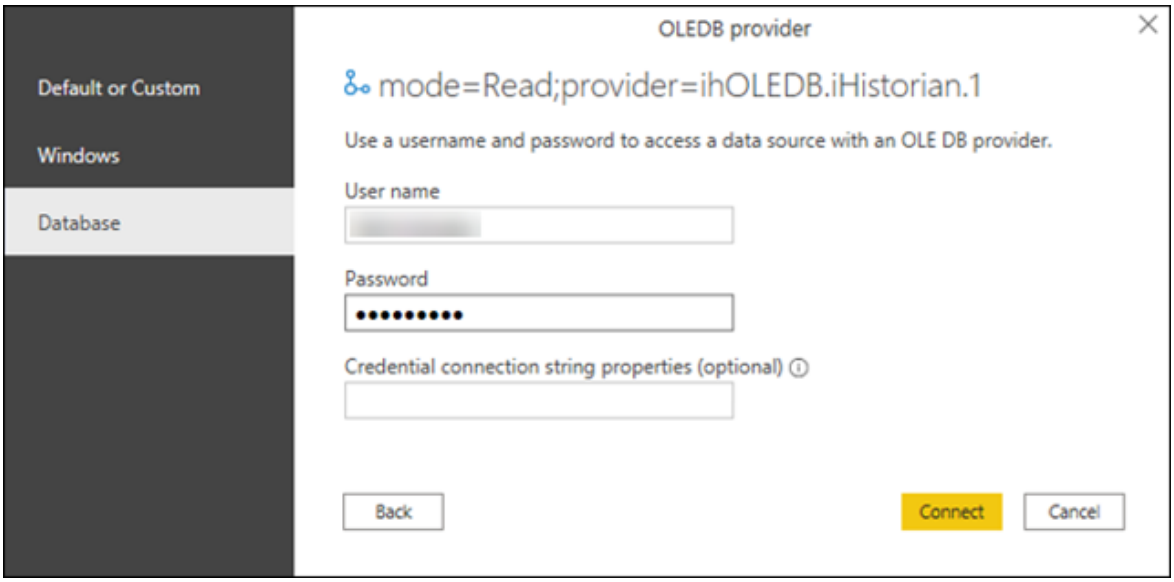
> **Important:**
> Do not use the connection string that is populated. If you do so, an error occurs.

6. Change the connection string to `provider="ihOLEDB.iHistorian.1";mode=Read`

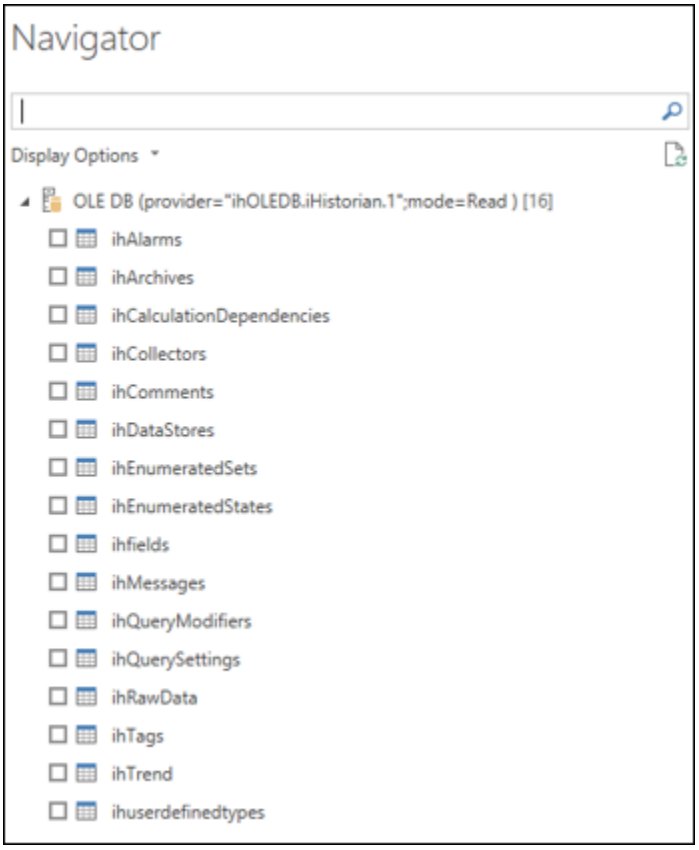
7. Select **OK**.

   The **OLE DB Provider** window appears.

8. In the **Database** section, enter the credentials to connect to the Historian server, and then select **Connect**.

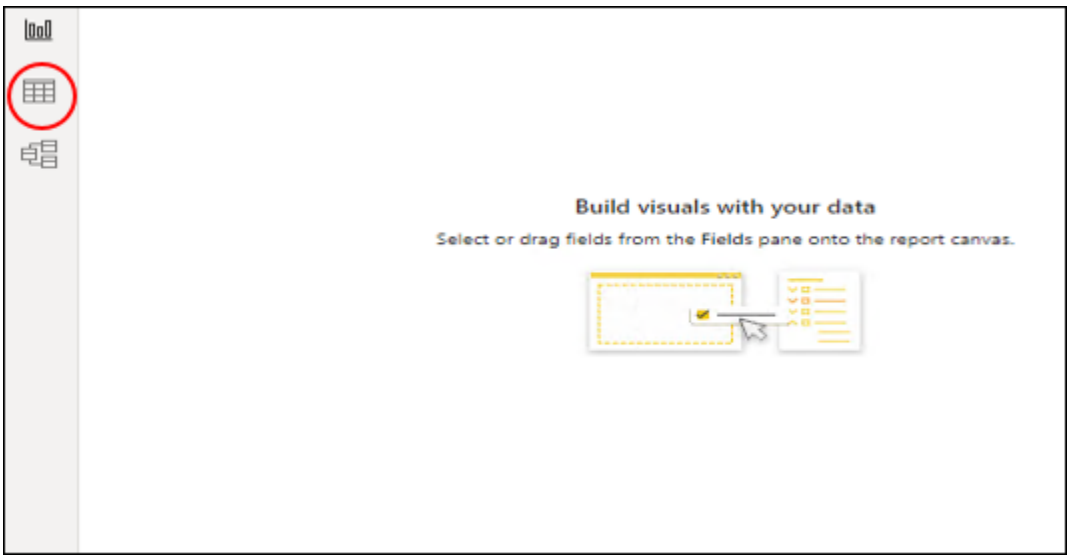



A list of Historian table appears in the **Navigator** section.




If an error message appears after entering the credentials, restart your machine.

9. Select the table whose data you want to import, and then select **Load**.
10. Select .




Data from the selected Historian table appears.

You can now create a Power BI report and then publish it.

## Working with VisiconX

Using the OLE DB provider with VisiconX, you can:

- Use tables or SQL queries.
- Insert multiple controls into a picture to the same or different servers.
- Provide a username and password or be prompted when opening a picture.

1. Access the Historian OLE DB provider from VisiconX. For instructions, refer to Using VisiconX.
2. To make all the VisiconX controls use synchronous (SYNC) executes:
    a. Access the `FixUserPreferences.ini` file in the `Dynamics/Local` folder.
    b. Add the following lines to the end of the file:

```
[VisiconX]

RUNASYNC=FALSE
```

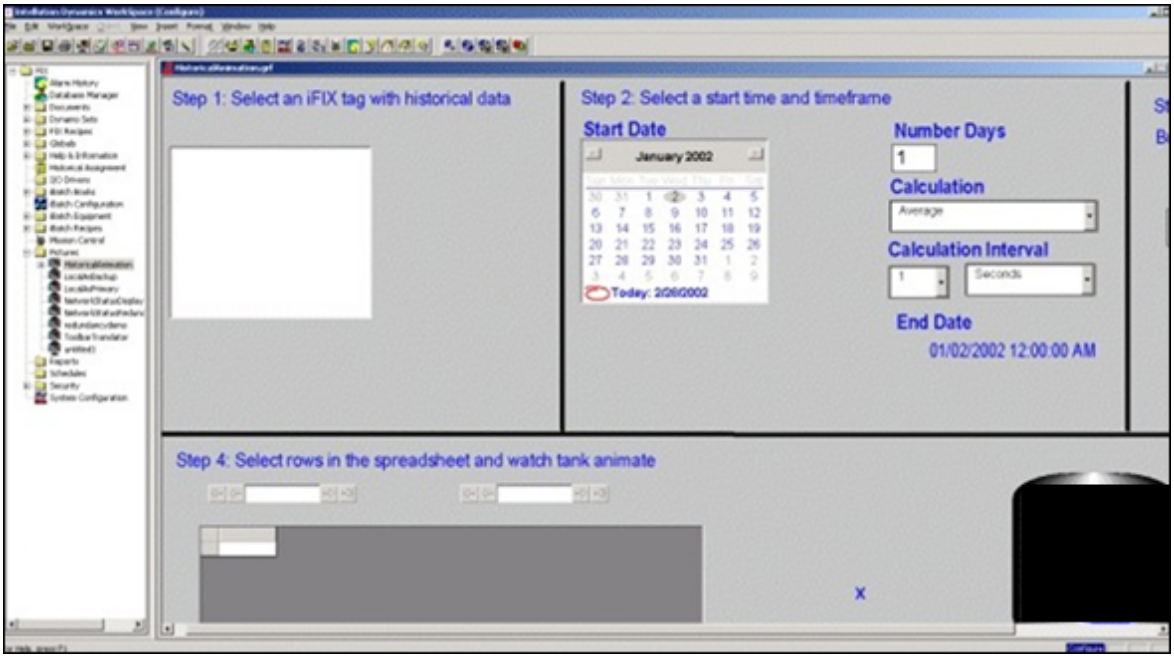
    c. Save the file, and restart the collector.

## Access the iFIX Sample Picture

To use iFIX with VisiconX, edit the `FixUserPreferences.ini` configuration file.

The `HistoricalAnimation.grf` file contains an iFIX sample picture with the VisiconX controls. It is located in the `Historian\Samples\iFIX` folder.

1. Copy the `HistoricalAnimation.grf` file to your `Dynamics/Pic` folder.
2. Start iFIX.
3. Open **iFIX WorkSpace**.
4. Double-click the **Pictures** folder.
5. Double-click the **HistoricalAnimation** picture.
   The picture appears in the workspace.




You can now perform the following tasks:

- Modify the picture.
- Switch between the configure and run modes.
- Follow the steps on the picture.
- View the properties of the VisiconX controls.
- Change the properties.

**Note:**
You can have multiple VisiconX controls that each link to different Historian servers.

## Create a Background Schedule to Run Crystal Reports

In iFIX, you can create a background schedule that runs Crystal reports. This topic contains a sample Visual Basic code to create a background schedule:

```
Private ReportFileName

Private CrystalReport


Private Sub KKTimer_OnTimeOut(ByVal lTimerId As Long)


Set CrystalApplication = CreateObject("Crystal.CRPE.Application")

Set CrystalReport = CrystalApplication.OpenReport("C:\Program Files (x86)\GE\iFIX\APP\RTtemplate.rpt")

 CrystalReport.Printout False
```

```
Set CrystalReport = Nothing

Set CrystalApplication = Nothing

End Sub
```

## Working with Oracle

You can import Historian data into Oracle by using an ADO program. A sample program is provided in the `Historian/Samples/Oracle` folder.

Use SQL WorkSheet to test that Oracle imported the data and created the tables properly.

## Crystal Reports

Crystal Reports allows you to create reports easily through its experts and wizards. When working with Crystal Reports,remember that:

- Crystal does not support the `SET` command. You must use a `WHERE` clause in a `SELECT` statement to specify query parameters.
- A single report can only retrieve data from one server, but you can create subreports from different servers within a report.
- The Crystal Reports application does not display milliseconds in timestamps.
- IIf you want to create a report on numeric data in the Value or Quality column in the `ihRawData` table, you may want to convert all Variant data types to Float data types so that Crystal displays them correctly in the report. Refer to Format Decimal Point Precision *(on page 2590)* for instructions.
- Analysis of the ihTrend and ihAlarm tables in Crystal Reports is not supported.

**Table 384. Crystal Reports Samples**

| File Name | Description |
|---|---|
| `SimpleCrystal80Report.rpt` | Contains values cast from **Variant** to **Float**. |
| `MultipleServersSubreport.rpt` | Contains data from two servers by using a subreport. |
| `iFIX1_CHART_OLEDB.rpt` | Contains data from iFIX Sample System converted from the iFIX Historical ODBC driver to OLE DB. |

**Table 384. Crystal Reports Samples (continued)**

| File Name | Description |
|---|---|
| `iFIX1_CROSSTAB_OLED-B.rpt` | Contains data from iFIX Sample System converted from the iFIX Historical ODBC driver to OLE DB. |
| `iFIX1_DAILY_OLED-B.rpt` | Contains data from iFIX Sample System converted from the iFIX Historical ODBC driver to OLE DB. |

## Connect to the Historian Server

To generate reports in Crystal Reports, you must ensure that the Historian Server name is correct. By default, this server name is `T20`.

1. Open the report file in Crystal Reports.
2. Select the **Database** menu.
3. If the **Database** menu does not appear automatically:

    a. Wait for approximately 90 seconds for the connection timeout to occur.
    After the 90-second timeout, the **Data Link Properties** window appears. Although it may appear as if Crystal Reports has stopped working or is frozen before the timeout occurs, this functionality is as expected.

    b. Change the **Data Source** field.
    You can leave this field empty to use the default Historian server or enter a specific Historian server name.

    c. Select **OK**.

    d. Skip the next step.

4. If the **Database** menu appears automatically:

    a. Select **Database > Set Location**.
    The **Set Location** window appears.

    b. Select **Set Location**.
    The **Data Explorer** window appears.

    c. Select a source, and then select **Set**.

    d. Select **Done**.

The Historian server is connected with Crystal Reports.

## Create a Crystal Report

Ensure that Crystal Reports is integrated with the Historian server whose data you want to analyze. For instructions, refer to Connect to the Historian Server *(on page 2588)*.

This topic describes how to import Historian table data into Crystal Reports and create a report.

1. In Crystal Reports, select **File > New**.
   The **Crystal Reports Gallery** window appears.
2. Select **Using the Report Expert > Standard Report Expert**, and then select **OK**.
   The **Standard Report Expert** appears.
3. Select **Database**.
   The **Data Explorer** appears.
4. Open the `More Data Sources` folder, and then open the `OLE DB` folder.
5. Select **Make New Connection > Add**.
   The **Data Link Properties** window appears.
6. Select **Historian OLE DB Provider > Next**.
   The **Connection** section appears.
7. Leave these fields empty to use the default server and currently logged-in user. Otherwise, do the following:
   a. Enter the name of the Historian server in the **Data Source** field.
   b. Clear the **Blank Password** check box.
   c. Enter a Windows username and password.
8. Select **OK**.
   The Historian OLE DB provider tables appear in the **Data Explorer**.
9. Select the table that you want to query, select **Add**, and then select **Close** to exit the **Data Explorer** window.
10. In the **Fields** section of the **Standard Report Explorer** window, select a field that you want to report on, and then select **Add** to move the field into the **Fields to display** list.

> **Note:**
> If you want to create a report on numeric data in the Value or Quality column in the `ihRawData` table, you may want to convert all Variant data types to Float data types so that Crystal displays them correctly in the report. Refer to Format Decimal Point Precision *(on page 2590)* for instructions.

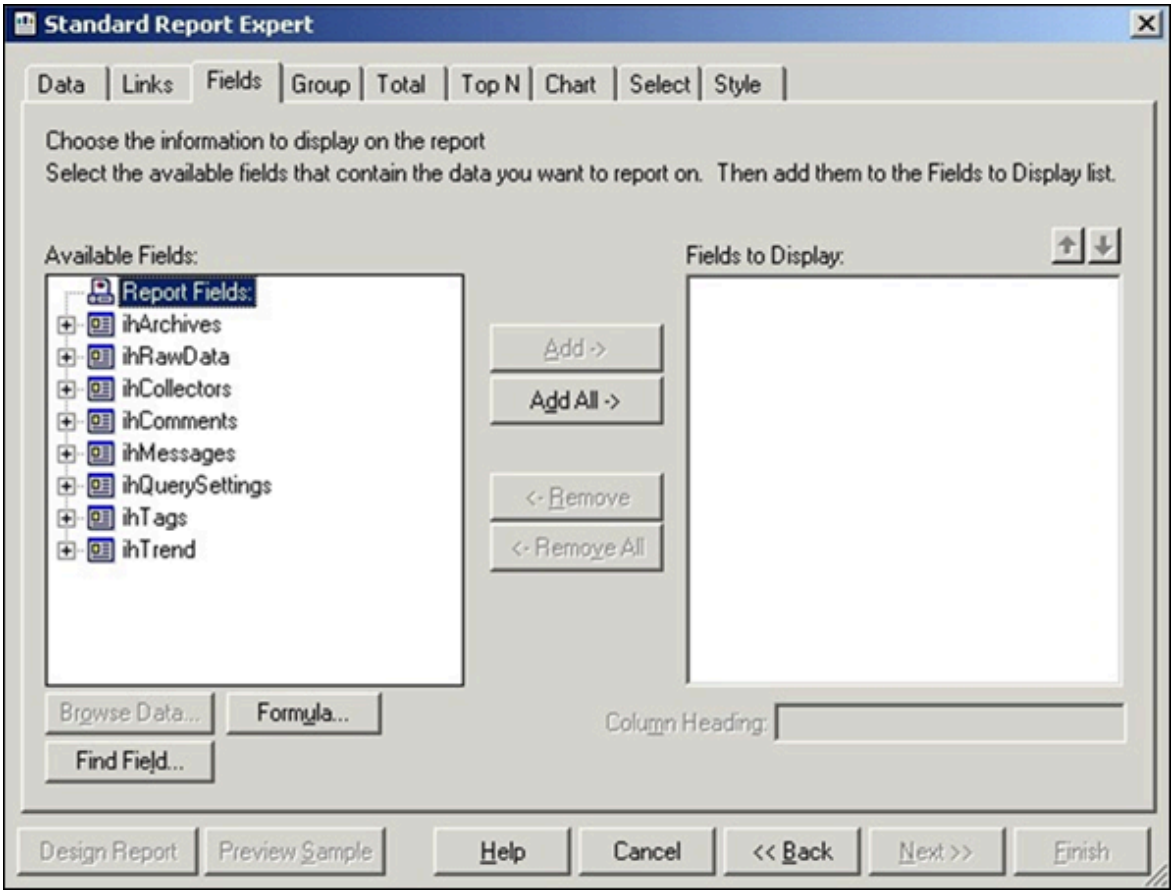
11. Repeat the previous step for each field that you want to add, and then select **Finish**.
    The Crystal Report is generated.

## Format Decimal Point Precision

Connect to the OLE DB provider, and add the Historian database tables.

To format decimal point precision in your reports, you must convert Variant data types to Float data types in Crystal Reports. For instance, if retrieving the Value column from the ihRawData table, you must convert the values to Float. You need not perform these steps if you are working with strings.

1. Access **Standard Report Expert**, and then select **Fields**.
   The **Fields** section appears.

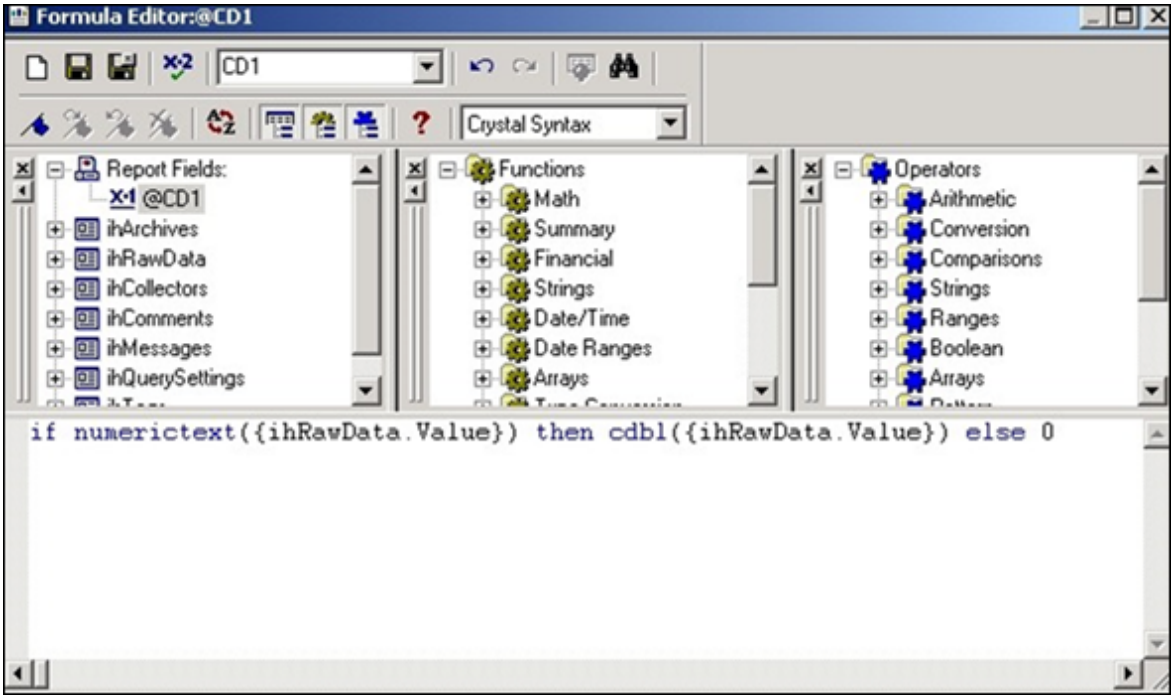



2. Select **Formula**.
   The **Formula Name** window appears.
3. Enter a name for the formula.
   The **Formula Editor** section appears.

> **Tip:**
> You can also access the **Formula Editor** section by selecting **Insert > Field Object**. Right-click the formula fields, and then select **New**.

4. In the **Formula** field, enter the following text :

```
if numerictext({ihRawData.Value}) then cdbl({ihRawData.Value}) else

                        0
```




5. Select **Save**.
   You can now use the formula as a normal numeric column instead of the **Value** column in the report.

## Change the Date and Time Format

This topic describes how to format the date/time column of Historian tables in Crystal Reports. When formatting timestamps, note that milliseconds do not appear in Crystal Reports.

1. Select a field in a column that contains timestamps.
2. Right-click the field, and then select **Format Field**.
   The **Format Editor** window appears.
3. Select **Date/Time**, and specify the date format that you want to use.

4. Select **OK**.
   The timestamps are updated to display the new format.

## Microsoft Excel

With Excel, you can import a snapshot of Historian data at a single point in time. You can choose Historian as a data source in Excel. You can specify the connection settings manually *(on page 2592)* or using a UDL file *(on page 2593)*.

After you import the data, you can create and edit SQL queries in Excel.

**When to Use Excel Instead of the Historian Excel Add-In**

Use the Excel Add-In when you want to get data into Microsoft Office 2003, 2007 or 2010 (32-bit/64-bit). Use the Historian OLE DB provider with Excel, instead of the Excel Add-In, when you want to do any of the following:

- Perform advanced filtering, sorting, and joining of data.
- Obtain detailed information from the ihTrend table.
- Run calculations using the SQL aggregate functions.
- Perform advanced summaries.

**Table 385. Microsoft Excel Samples**

| File Name | Description |
|---|---|
| `ihOLEDB_LASTHOUR.XLS` | One-sheet report that uses auto-refresh to display the last hour of data using relative shortcuts. |
| `ihTags.odc` | Data source file that retrieves the `ihTags` table from the default server. |
| `iHistorian.udl` | Sample universal data link (`.UDL`) file that connects to the default Historian server with the currently logged-in user. |

These sample are found in the following folder: `Historian\Samples\Excel`

## Import Historian Data Into Excel Manually

This topic describes how to import Historian data into Excel by providing the connection details manually. You can also import the data by creating a UDL file *(on page 2593)* or by using the sample UDL file *(on page 2595)*.

1. Open an Excel worksheet.
2. Select **Data > Import External Data > Import Data**.
   The **Select Data Source** window appears.
3. Select **My DataSources > +Connect to New Data Source.odc > Open**.
   The **Data Connection Wizard** appears.
4. Select **Other/Advanced** from the list of data sources to which you can connect, and then select **Next**.
   The **Data Link Properties** window appears.
5. Select **Historian OLE DB Provider** from the **OLE DB Provider** list, and then select **Next**.
   The **Connection** section appears in the **Data Link Properties** window.
6. Leave these fields empty to use the default server and the currently logged-in user. Otherwise, do the following:
   a. Enter the name of the Historian server in the **Data Source** field.
   b. Clear the **Blank Password** check box.
   c. Enter a Windows username and password.
   d. Select the **Allow Saving Password** check box if applicable.
7. Select **Test Connection** to confirm that the data source, username, and password provide a successful connection, and then select **OK**.
   The **Select Database and Table** page appears in the wizard.
8. Select the table that you want to query, and then select **Next**.
   The **Save Data Connection File and Finish** page appears in the wizard.
9. Accept the default settings, and select **Finish**.
   The **Import Data** window appears.

   **Note:**
   If you want to run a specific SQL command instead of the default table command setting, refer to Edit SQL Queries in Excel *(on page 2595)*.

10. Select **OK** to import the column data from the selected table.
    Historian data populates the current spreadsheet.

## Import Historian Data Into Excel by Creating a UDL File

This topic describes how to create a UDL file with connection information and then import Historian data into Excel using the UDL file. You can also provide the connection details manually *(on page 2592)* or using the sample UDL file *(on page 2595)*.

1. Create a UDL file with connection details:
    a. Create a text document.

    We recommend that you use the **My Data Sources** folder in the **My Documents** folder.

    b. Rename the file extension .UDL.

    c. Double-click the `.UDL` file.
    The **Data Link Properties** window appears.

    d. Select **Provider > Historian OLE DB Provider > Next**.
    The **Connection** section appears in the **Data Link Properties** window.

2. Leave these fields empty to use the default server and the currently logged-in user. Otherwise, do the following:
    a. Enter the name of the Historian server in the **Data Source** field.
    b. Clear the **Blank Password** check box.
    c. Enter a Windows username and password.
    d. Select the **Allow Saving Password** check box if applicable.
3. Select **Test Connection** to confirm that the data source, username, and password provide a successful connection, and then select **OK**.
    The **Select Database and Table** page appears in the wizard.
4. Select **Data > Import External Data > Import Data**.
    The **Select Data Source** window appears.
5. Select the `.UDL` file that you have created, and then select **Open**.
    The **Select Table** window appears.
6. Select the table that you want to query, and then select **OK**.
    The **Import Data** window appears.

    **Note:**
    If you want to run a SQL command instead of the default table command setting, refer to Edit SQL Queries in Excel *(on page 2595)*.

7. Select **OK** to import the column data from the selected table.
    Historian data is imported into the spreadsheet.

## Import Historian Data into Excel Using the Sample UDL File

With the sample universal data link (`.UDL`) file, you can specify the connection information so that Excel can connect to the tables in the OLE DB provider and import data using the default server and the currently logged-in user.

This topic describes how to import Historian data into an Excel spreadsheet using the sample .UDL file. You can also import Historian data by providing the connection details manually *(on page 2592)* or by creating a .UDL file *(on page 2593)*.

1. Open an Excel spreadsheet.
2. Select **Data > Import External Data > Import Data**.
   The **Select Data Source** window appears.
3. Select the `Historian.udl` file in the `Historian\Samples\Excel` folder, and then select **Open**.
   The **Select Table** window appears.
4. Select the table that you want to query, and then select **OK**.
   The **Import Data** window appears.

   > **Note:**
   > If you want to run a SQL command instead of the default table command setting, refer to Edit SQL Queries in Excel *(on page 2595)*.

5. Select **OK**.
   Historian data appears in the spreadsheet.

## Edit SQL Queries in Excel

By default, data import functionality in Excel selects all columns from the specified Historian table using the default query parameters. This command is the equivalent of running the SQL command `SELECT * FROM TABLE_NAME`, where *TABLE_NAME* is the name of the table that you want to query.

You can change the query by issuing a different SQL query if you are familiar with SQL syntax. Refer to the Microsoft Excel documentation for more information.

If you are unsure if the SQL syntax is correct, you can test your SQL query outside of Excel using the Historian Interactive SQL application. See Historian Interactive SQL Application *(on page 2606)* for more details.

## Format Date and Time

This topic describes how to format the date/time column for Historian tables in Excel if you need to display a specific date format. For more specific information on formatting spreadsheets, refer to the Microsoft Excel online Help.

1. Right-click the heading of the column that you want to format.
2. Select **Format Cells > Number**.
3. Select **Date**.
4. In the **Type** field, select the date format that you want to use.
   To display milliseconds, instead of selecting the **Date** category, select **Custom**, and then enter `dd-mmm-yyyy hh:mm:ss.000` in the **Type** field.
5. Select **OK**.
   The date and time format is set.

## Refresh Data

After you import Historian data into an Excel worksheet, you can refresh it to get the most updated data. This feature is most useful when using relative start times, such as `Now - 2h`. You can also set a refresh interval to refresh data automatically.

1. Open the Excel worksheet into which you have imported the Historian data.
2. Select **External Data > Refresh Data**

   > **Tip:**
   > If the **External Data** toolbar is not available, select **View > Toolbars**.

   The data is refreshed.
3. To automatically set refresh intervals, select **Data Range Properties**, and provide the interval at which you want to refresh data automatically.
   Data is refreshed automatically at the interval that you have specified.

## Visual Basic and ADO

You can access the OLE DB provider using Microsoft ActiveX Data Objects (ADO). This approach is more generic than using the Historian SDK.

Visual Basic supports asynchronous (ASYNC) connections. You can open multiple ADO connections to the same data source from within a Visual Basic program. You are limited to one server per connection,

and one username and password. A different user can make another connection to the same server, however, by using a different username and password.

We recommend that you use client-side cursors instead of server-side cursors in Visual Basic. If you use a server-side cursor, the `RowCount` property on the `recordset` object will always be `-1` instead of the actual row count.

**Table 386. Visual Basic and ADO Samples**

| File Name | Description |
|---|---|
| `SimpleADOExample.vbp` | Visual Basic project file that uses a simple ADO example with a connect string. |
| `modSimpleADOExample.bas` | File that is part of the `SimpleADOExample.vbp` project file. |
| `iholedb_databoundgrid.vbp` | Visual Basic project file that displays a data-bound grid example that fetches data from the `ihRawData` table. |
| `frmMain.frm` | File that is part of the `iholedb_databoundgrid.vbp` project file. |
| `frmMain.frx` | File that is part of the `iholedb_databoundgrid.vbp` project file. |

These samples are available in the following folder: `Historian\Samples\VB`

## Retrieve Milliseconds

Use the following code to retrieve timestamps to a resolution of milliseconds.

```
Public Function Time_To_String_With_Milliseconds(TheTime As Double) As String

Dim Temp As String

Dim TimeFraction As Double

Dim Msc As Long

Dim TempTime As Date


On Error GoTo errc


If TheTime = 0 Then

Time_To_String_With_Milliseconds = ""

Exit Function

End If
```

```
TimeFraction = TheTime * 86400#

TimeFraction = TimeFraction - Fix(TimeFraction)


Msc = CLng(TimeFraction * 1000)


TempTime = TheTime - (TimeFraction / 86400#)

If Msc = 1000 Then

Msc = 0

TempTime = DateAdd("s", 1, TempTime)

End If


Time_To_String_With_Milliseconds = LCase(Format$(TempTime, "dd-mmm-yyyy hh:nn:ss") + "." + Format$(Msc, "000"))


errc:

End Function
```

## Set a Maximum Limit to Records

Use the following example code to set a maximum limit to the number of rows returned in your query:

```
SET rstTitles = New ADODB.Recordset

rstTitles.MaxRecords = 10

strSQLTitles = "SELECT Tagname FROM ihTags"

rstTitles.Open strSQLTitles, strCnxn, adOpenStatic, adLockReadOnly, adCmdText
```

## Use Parameterized Queries

Use the following example code to use parameterized queries:

```
Private Sub SampleParameterizedQuery()

    Dim ihConnectString As String

    Dim ihRecordSet     As ADODB.Recordset

    Dim ihConnection    As ADODB.Connection

    Dim ihParameter     As ADODB.Parameter

    Dim ihCommand       As ADODB.Command


    'Set Up the Historian Connect String...

    Set ihConnectString = "Provider=ihOLEDB.iHistorian.1;User Id=;Password="


    'Create Our Other Objects...
```

```
    Set ihConnection = CreateObject("ADODB.Connection")
    Set ihRecordSet = CreateObject("ADODB.Recordset")
    Set ihCommand = CreateObject("ADODB.Command")

    'Open the Connection to the Historian Archiver...
    ihConnection.ConnectionString = ihConnectString
    ihConnection.Open
    'Set up the Command Object
    With ihCommand

        'Set the Active Connection to the Historian Connection Opened Above..
        .ActiveConnection = ihConnection

        'Set the Command Text to a Parameterized Sql Statement....
        .CommandText = "select * from ihTags where datatype = ?"

        'Set the Type of the Command...
        .CommandType = adCmdText

        'Refresh Our Parameter List...
        .Parameters.Refresh
    End With

    'Create a Single Parameter Object...
    Set ihParameter = ihCommand.CreateParameter("Temp", adChar, adParamInput, 100)
    'Set the Parameters Value...
    ihParameter.Value = "SingleFloat"
    'Add the Parameter to the Command Object...
    ihCommand.Parameters.Append ihParameter
    'Run the Command!
    Set ihRecordSet = ihCommand.Execute
End Sub
```

For more information, refer to Parameterized SQL Queries *(on page 2636)*.

## Proficy Real-Time Information Portal

Proficy Real-Time Information Portal is a web-based tool for accessing, analyzing, and visualizing production information. It has sophisticated trending and reporting capabilities that take advantage of the vast archival and retrieval capabilities of Historian.

In Proficy Real Time Information Portal, parameters are used to build SQL queries that you can reuse with different values. In the place of a constant value in a SQL query, you can use a parameter, which takes a dynamic value at execution time. Parameterized SQL queries are driven by Proficy Real Time Information Portal components such as list boxes, combo boxes, or grids.

The SQL Query Builder application in Proficy Real Time Information Portal is used to define a parameterized query.

**To define a parameterized query:** In the **Specify Selected Item Wizard** or **Specify Criterion Wizard**, in the **Parameter** field, enter the name of the parameter.

The following conditions apply when you define a parameterized query:

- Parameter names must be unique.
- A question mark (?) is appended to the parameter name, and the parameter is enclosed in parentheses. For example, the parameter `temperature` becomes `{temperature?}`.
- You can specify a default value for the parameter.
- You can also select a data type for the parameter. By default, the data type is set to `char`. However, you can select `int`, `date`, `num`, or `char` as the type of database column.

## Linked Servers in Microsoft SQL Server

If you want to relate Historian data with other data in SQL Server tables such as batch events, iFIX Alarms and Events collector, iDownTime data, and any other information that is available in a relational database, you can use the OLE DB provider as a linked server in Microsoft SQL Server. You can also use the OLE DB provider as a linked server if you do not want to duplicate data with an import.

With linked servers, when you query data from Historian, the SQL server fetches the requested data from Historian at the time the query is executed. Data is not duplicated because nothing is imported or stored in the SQL server. The data is simply returned as part of a query, just as any other query on a SQL Server database would return data.

Another advantage of using the OLE DB provider as a linked server is that you do not need to install Historian in the client machines. For example, a client tool such as Microsoft Query Analyzer can be used to retrieve Historian product data over the network on a computer with no Historian software installed.

## Configure the OLE DB Provider as a Linked Server Manually

The following steps are necessary in order to access a linked server via the `OPENQUERY` statement.

This topic describes how to configure the OLE DB provider as a linked server manually. You can also configure it automatically *(on page 2601)*.

1. From the **Start** menu, open the **SQL Server Enterprise Manager**.
2. Select an SQL server, and open the `Security` folder.
3. Right-click the `Linked Servers` folder, and select **New Linked Server**.
   The **Linked Server Properties** window appears.
4. Enter a name for the linked server, such as `iHist`.
5. In the **Provider Name** field, select **Historian OLE DB Provider**.
6. In the **Data Source** field, enter the name of the Historian server, and then select **Provider Options**.
   The **Provider Options** window appears.

   > **Note:**
   > - Select the **Level Zero Only** option only if using older versions of SQL server. For better performance while executing small queries, select the **Allow in Process** option. Clear the option if larger queries are to be executed.
   > - For configuring the Historian 64-bit OLE DB provider as a linked server, the **Allow in Process** option is mandatory.

7. Select **OK**.
8. If Historian security is enabled, enter a Historian username and password.
9. For SQL Server 2008 (32-bit/64-bit), follow these steps:
   a. Select **Security**.
   b. Select the **Be made using this security context** option.
   c. Enter a Historian username and password in the **Remote Login** and **With Password** fields.
10. Select **OK**.
    The linked server is created.

## Configure the OLE DB Provider as a Linked Server Automatically

Configure a linked server and options using **Enterprise Manager**, as described in Configuring the Historian OLE DB provider as a Linked Server *(on page 2601)*. Then, since the options **Allow In Process** and **Level Zero Only** apply to all linked servers that use the provider, you can create additional linked server definitions to other Historian servers using the `sp_addlinkedserver` stored procedure.

This topic describes how to configure the OLE DB provider as a linked server automatically using the `sp_addlinkedserver` system stored procedure from Microsoft SQL Server. You can also configure it manually *(on page 2601)*.

1. To configure a linked server definition, use the following example code:

```
EXEC sp_addlinkedserver @server='MYSERVER_LS', @srvproduct='',
@provider='iHOLEDB.iHistorian.1', @datasrc='MY_SERVER'
```

2. To search for linked server definitions, use the following example code:

```
EXEC sp_linkedservers
```

3. To delete linked server definitions, use the following example code:

```
EXEC sp_dropserver 'MYSERVER_LS', 'droplogins'
```

## Access a Linked Server

Configure a linked server and options using **Enterprise Manager**, as described in Configuring the Historian OLE DB provider as a Linked Server *(on page 2601)*.

This topic describes how to access the OLE DB provider as a linked server in an SQL server using the following methods:

- **`OPENQUERY`:** This is the recommended method of accessing data by means of a linked server. To use this method, you must first configure a linked server definition. You can then use that linked server name in the `OPENQUERY` command.
- **Four-Part Name Syntax:** To use this method, you must first configure a linked server definition. You can then use that linked server name in the four-part name syntax.
- **`OPENROWSET` and `OPENDATASOURCE`:** These methods are considered adhoc methods of accessing data. They are recommended only for infrequently accessed data. When using either method, you must specify the data source, username, and password in each query instead of configuring it once in a linked server definition. If you want to limit the number of users to a defined set of servers and usernames, you can disable all methods of adhoc access by selecting the **Disallow Adhoc Accesses** option in the **Provider Options** window.

> **Note:**
> You cannot use `OPENQUERY` to access the `ihTrend` table. Use four-part name syntax to access the `ihTrend` table.

1. To fetch a list of Historian tags, run the following query:

```
SELECT * FROM OPENQUERY(iHist,'SELECT * FROM ihTags')
```

2. To fetch tag values from Historian, use the following example code:

```
SELECT TagName, TimeStamp, Value, Quality FROM OPENQUERY (iHist,'

SET

StartTime=yesterday-12Day, EndTime=Today, IntervalMilliseconds=1Hour, SamplingMode=Calculated,

 CalculationMode=Maximum

SELECT * FROM ihRawData WHERE TagName LIKE *simulation00001')
```

3. To access the ihTrend table from a linked server, run the following query:

```
SELECT * FROM iHist...[SELECT timestamp, *.value FROM ihTrend]
```

   Although the four-part name syntax works with all tables, it is only necessary to use it with the ihTrend table, because the ihTrend table does not work with `OPENQUERY`.

4. To use OPENROWSET with an SQL query, use the following example code:

```
SELECT * FROM OPENROWSET('ihOLEDB.iHistorian.1',

'MY_SERVER';'';'','SET starttime="2002-01-30 10:00:00", endtime="2002
```

   **Note:**
   This example uses double quotes around date and time because single quotes do not work inside the overall single-quoted query. It is important for you to use double quotes in this scenario.

5. To access a table, use the following example code:

```
SELECT * FROM OPENDATASOURCE('iHOLEDB.iHistorian.1', 'Data Source=MY_SERVER')...ihTags
```

6. To use OPENDATASOURCE with an SQL query and security, use the following example code:

```
SELECT * FROM OPENDATASOURCE('iHOLEDB.iHistorian.1',

'Data Source=MY_SERVER;User ID=user1;Password=thepassword')...[SE
```

7. To join Historian data with iFIX data logged with AlarmODBC, use the following example code, which determines the last date and time a specific analog tag was raised as an alarm. The date and time are then used to collect the data from the previous hour leading up to the alarm. You can use this example to determin if the value spiked into the alarm or slowly approached the alarm limit.

```
declare @var1 as varchar(300)

declare @iHistServer as varchar(10)
```

```
declare @Tagname as varchar(40)
declare @HistTagname as varchar(50)
declare @AlarmStatus as varchar(10)
declare @Node as varchar(8)
declare @StartDt as varchar(30)
declare @EndDt as varchar(30)
declare @queryDt as varchar(30)
SET @iHistServer = 'iHistMY_SERVER'
SET @Node = 'MY_SCADA'
SET @Tagname = 'Simulation00001'
SET @HistTagname = 'MY_SERVER.' + @Tagname
SET @AlarmStatus = 'HIHI'
SET @queryDt= DATEADD(day, -1, CURRENT_TIMESTAMP)
SET @EndDt = (SELECT TOP 1 DateTimeLast FROM AlarmODBC WHERE AlarmStatus = @AlarmStatus AND Node = @Node and
 Tagname =
SET @StartDt = DATEADD(hour, -1, @EndDt)
set @var1 = 'SELECT * FROM OPENQUERY
('+ @iHistServer +','''SET StartTime="'+ @StartDt +'", EndTime="'+ @Enddt +'"
SELECT Tagname, TimeStamp, Value, Quality FROM ihRawData WHERE TagName = '+ @HistTagname +''')' exec (@var1)
```

8. To access linked server data using a stored procedure, use the following example code, which interfaces with the alarm's ODBC table to get the last alarm time for a specified tag in the past 24 hours. It then uses this time to retrieve data for the tag from one hour leading up to the time the alarm occurred.
   The input parameters are the linked Historian server name, tag name, alarm status, and SCADA node name on which the alarm was created. This example uses a sim tag in the Historian database rather than setting up a collector to an iFIX SCADA node. Preferably, an iFIX tag name must be concatenated with the node and field (`node.tagname.fieldname`).

   a. To execute a stored procedure, use the following example code:

   ```
   EXEC alarmhist 'iHistMY_SERVER', 'simulation00001', 'HIHI', 'MY_SCADA'
   ```

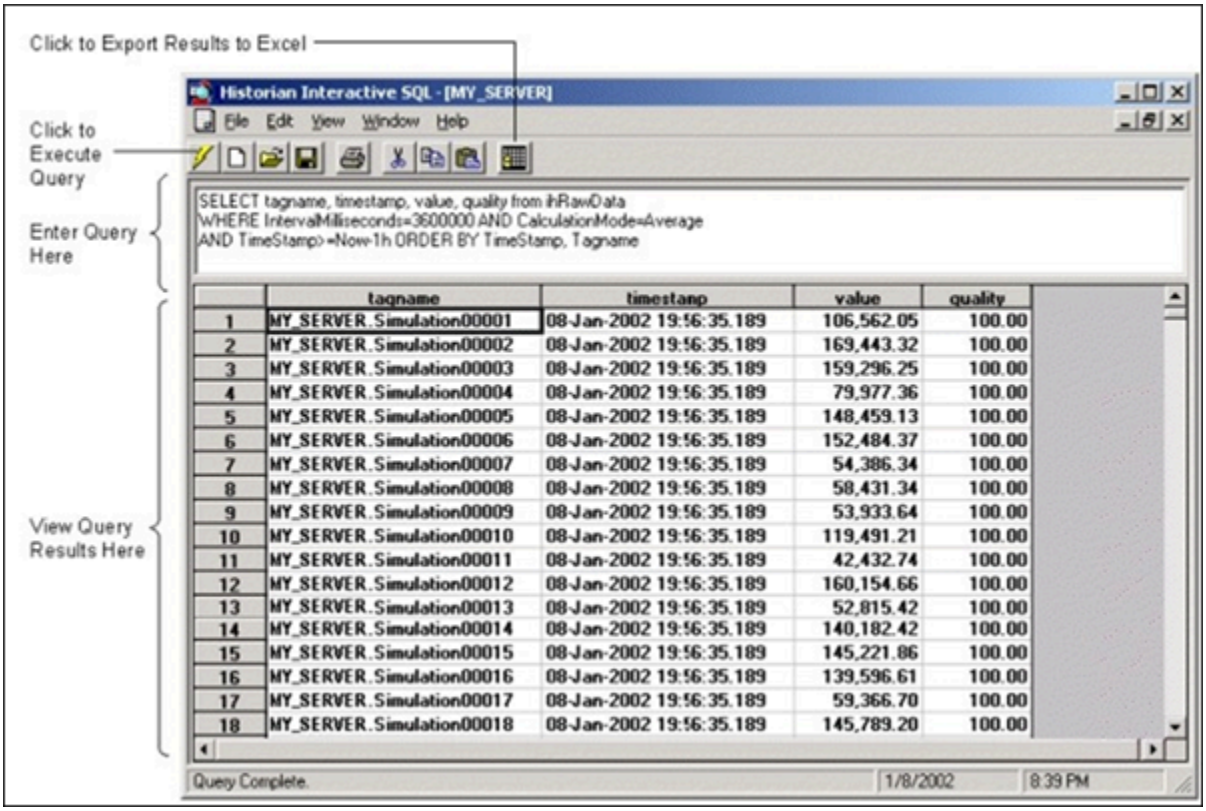
   b. When you create the stored procedure in **Enterprise Manager**, include the following lines before the create procedure command to avoid an error:

   ```
   SET ANSI_NULLS ON
   GO
   (@iHistServer varchar(10),
   @Tagname varchar(40),
   ```

```
@AlarmStatus varchar(10),

@Node varchar(8))

AS

declare @var1 as varchar(400)

declare @HistTagname as varchar(50)

declare @StartDt as varchar(30)

declare @EndDt as varchar(30)

declare @queryDt as varchar(30)

declare @count as int

declare @CalculationMode as varchar(20)

SET @HistTagname = 'MY_SERVER.' + @Tagname

SET @queryDt= DATEADD(day, -1, CURRENT_TIMESTAMP)

SET @count = (SELECT COUNT(*) FROM AlarmODBC WHERE AlarmStatus = @AlarmStatus AND Node = @Node AND

 Tagname = @Tagname

If @count > 0

BEGIN

If @AlarmStatus = 'HIHI' or @AlarmStatus = 'HI'

BEGIN

SET @CalculationMode = 'Maximum'

END

ELSE

BEGIN

SET @CalculationMode = 'Minimum'

END

SET @EndDt = (SELECT TOP 1 DateTimeLast FROM AlarmODBC WHERE AlarmStatus = @AlarmStatus AND Node =

 @Node AND Tagname =

SET @StartDt = DATEADD(hour, -1, @EndDt)

SET @var1 = 'SELECT * FROM OPENQUERY

('+ @iHistServer +','''SET StartTime="'+ @StartDt +'",

EndTime="'+ @EndDt +'", IntervalMilliseconds=60000,

SamplingMode=Calculated,CalculationMode='+ @CalculationMode +'

SELECT Tagname, TimeStamp, Value, Quality FROM ihRawData WHERE TagName = '+ @HistTagname +''')'

print (@var1)

exec (@var1)

END

GO
```

# About Working with Queries

Using the Historian Interactive SQL application (`ihSQL.exe`), you can run an SQL query and display the results of the query in the same window. It is useful if you want to test a query using the OLE DB provider.




It can open and save SQL queries and can show multiple windows, each containing a query request to the same server or different servers. For instance, you might want to open more than one window to compare two different time periods on the same server, or the same time period on different servers.

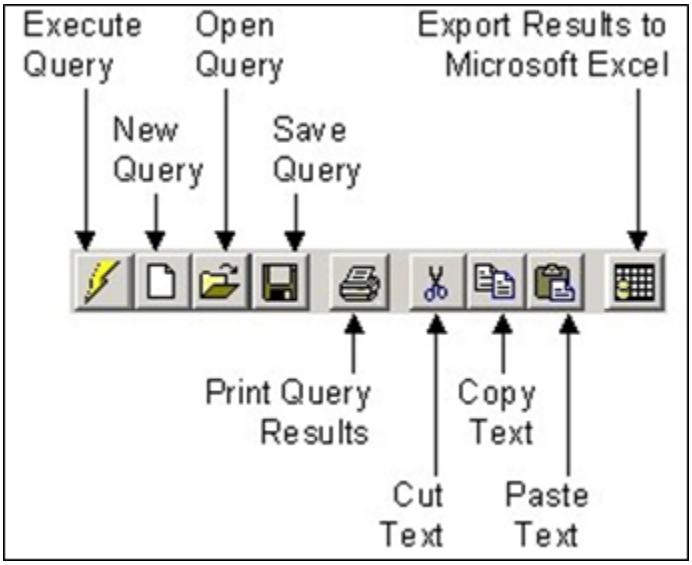
The Historian Interactive SQL application allows you to access data quickly and efficiently. Using this application, you can:

- Test SQL syntax before using it in an application.
- Troubleshoot OLE DB connections or Historian errors.
- Perform more complex searching or filtering of data than you can in the Historian SDK and administration applications.
- Retrieve data from any available Historian server.
- Save and access queries.
- Export query results to Microsoft Excel.

The Historian Interactive SQL application toolbar provides quick access to common functions such as:

- Executing queries
- Switching to a new Historian server
- Exporting query results to Microsoft Excel
- Saving a query
- Printing query results

The following figure shows the toolbar for the Historian Interactive SQL application, outlining what each button does.




## Access the Historian Interactive SQL Application

When you start the application, you can log in to the default server or another Historian server.

1. From the **Start** menu, select **Programs > Historian > Historian Interactive SQL**.

   > **Important:**
   > The first time you use `ihSQL.exe`, you may need to select **Run As Administrator**. Otherwise, you may not be able to log in.

   The **Historian Interactive SQL Login** window appears.

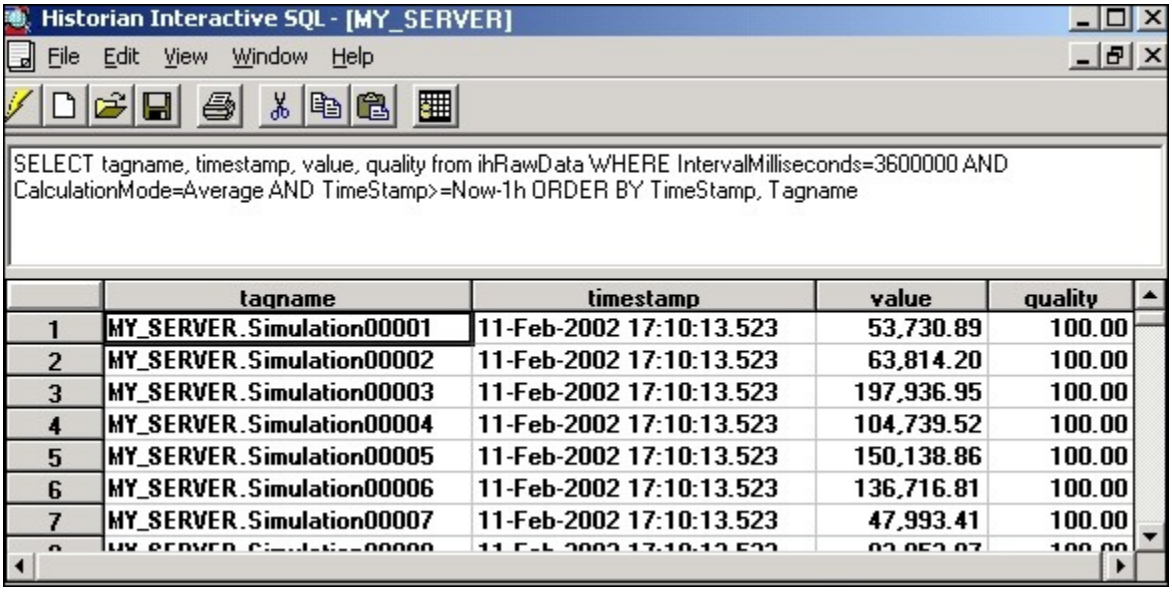
2. Select a Historian server, and then enter the username, password, and domain to connect to the server. If you do not enter user credentials, the currently logged-in user is considered.
3. Select **OK**.
   A new session of the Historian Interactive SQL application appears, and it is connected to the server that you have specified. The session begins with the default values for SET variables *(on page 2630)*.

> **Note:**
> If modifications or additions are made to the list of available Historian servers using any of the Historian clients (Excel, non-web Administrator, or iFIX WorkSpace: Expression Builder and iFIX Migration Tools), those settings are global for any Historian clients running on that computer.

## Run a Query

You can run a query against the data that is contained in the Historian database tables. A query is a `SET` or `SELECT` statement, or a combination of both of these SQL statements. When you execute a `SELECT` or `SET` statement in the Historian Interactive SQL application, you can execute only one `SET` and one `SELECT` statement per query.

1. Access the Historian Interactive SQL application *(on page 2607)*.
2. If you want to run a saved query, select **File > Open**, and then select the query that you want to run.
3. If you want to run a new query, enter your query in the **Query Entry** field.




4. Select or press Ctrl+E.
   The query results appear.

## Connect to a Server

The Historian Interactive SQL application allows you to make multiple connections to the same server or different servers. This allows you to look at data from different servers.

1. Access the Historian Interactive SQL application *(on page 2607)*.
2. Select **File > New**.
   The **Historian Interactive SQL Login** window appears.

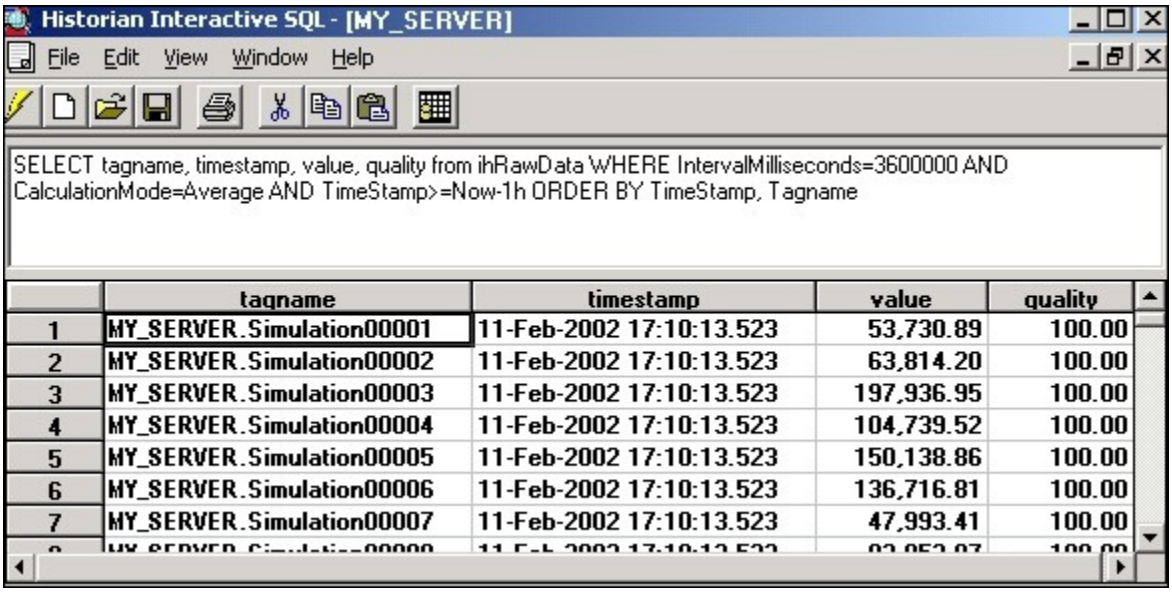
3. Select a Historian server, and then enter the username, password, and domain to connect to the server. If you do not enter user credentials, the currently logged-in user is considered.
4. Select **OK**.
   A new session of the Historian Interactive SQL application appears, and it is connected to the server that you have specified. The session begins with the default values for SET variables *(on page 2630)*.

> **Note:**
> If modifications or additions are made to the list of available Historian servers using any of the Historian clients (Excel, non-web Administrator, or iFIX WorkSpace: Expression Builder and iFIX Migration Tools), those settings are global for any Historian clients running on that computer.

## Save a Query

When you save a query, it is saved as an `.SQL` file in the current working directory. You can later open the query in the Historian Interactive SQL application or in other client applications.

1. Access the Historian Interactive SQL application *(on page 2607)*.
2. Enter your query into the **Query Entry** field.



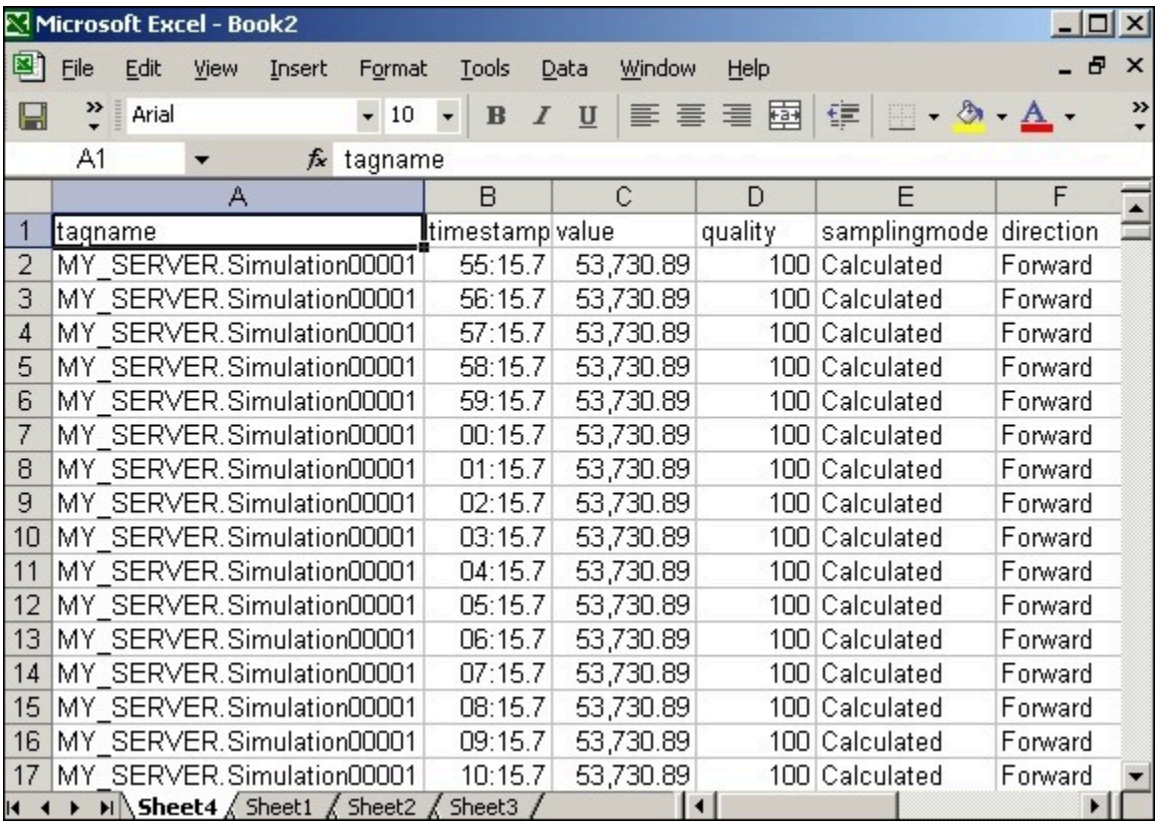

3. Select **File > Save**.
   The **Save Query to File** window appears.
4. Enter a name for the query.

> **Important:**
> Use the `.SQL` file extension.

5. Select .
   The query is saved in the working directory.

## Export Query Results to Excel

1. Run the query that you want to export *(on page 2608)*.
2. Select .
   The query results are exported to an Excel spreadsheet.




| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | tagname | timestamp | value | quality | samplingmode | direction |
| 2 | MY_SERVER.Simulation00001 | 55:15.7 | 53,730.89 | 100 | Calculated | Forward |
| 3 | MY_SERVER.Simulation00001 | 56:15.7 | 53,730.89 | 100 | Calculated | Forward |
| 4 | MY_SERVER.Simulation00001 | 57:15.7 | 53,730.89 | 100 | Calculated | Forward |
| 5 | MY_SERVER.Simulation00001 | 58:15.7 | 53,730.89 | 100 | Calculated | Forward |
| 6 | MY_SERVER.Simulation00001 | 59:15.7 | 53,730.89 | 100 | Calculated | Forward |
| 7 | MY_SERVER.Simulation00001 | 00:15.7 | 53,730.89 | 100 | Calculated | Forward |
| 8 | MY_SERVER.Simulation00001 | 01:15.7 | 53,730.89 | 100 | Calculated | Forward |
| 9 | MY_SERVER.Simulation00001 | 02:15.7 | 53,730.89 | 100 | Calculated | Forward |
| 10 | MY_SERVER.Simulation00001 | 03:15.7 | 53,730.89 | 100 | Calculated | Forward |
| 11 | MY_SERVER.Simulation00001 | 04:15.7 | 53,730.89 | 100 | Calculated | Forward |
| 12 | MY_SERVER.Simulation00001 | 05:15.7 | 53,730.89 | 100 | Calculated | Forward |
| 13 | MY_SERVER.Simulation00001 | 06:15.7 | 53,730.89 | 100 | Calculated | Forward |
| 14 | MY_SERVER.Simulation00001 | 07:15.7 | 53,730.89 | 100 | Calculated | Forward |
| 15 | MY_SERVER.Simulation00001 | 08:15.7 | 53,730.89 | 100 | Calculated | Forward |
| 16 | MY_SERVER.Simulation00001 | 09:15.7 | 53,730.89 | 100 | Calculated | Forward |
| 17 | MY_SERVER.Simulation00001 | 10:15.7 | 53,730.89 | 100 | Calculated | Forward |

Format the date and time *(on page 2596)* so that they appear correctly.

## Optimize the Query Performance

To optimize query performance, follow these guidelines:

- Perform `GROUP BY` on the server whenever available. For instance, Crystal Reports gives you the option to group on the server as opposed to the client.
- Use `DISTINCT` to eliminate duplicate rows.
- Be specific when specifying tag names. For instance, when using wildcards, be as specific as possible.
- Limit the duration between start and end times.
- Get as precise a data type as possible to improve storage efficiency and allow reporting tools such as Power BI or Crystal Reports to properly format the data in reports.
- Do not rely on `TOP` or `ROWCOUNT` to optimize performance because they do not change the load on the archive or network but instead they just limit what is returned to the caller.

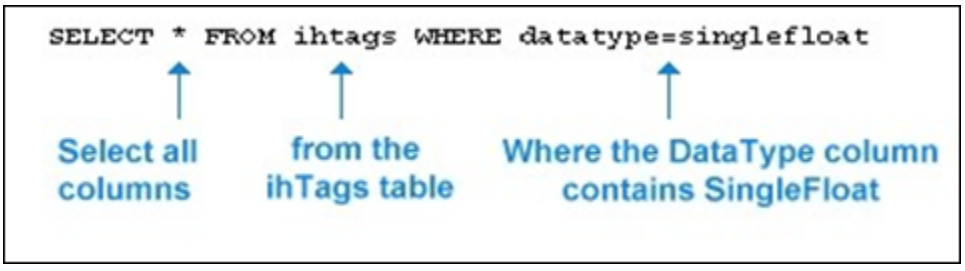
# Supported SQL Syntax

The OLE DB provider supports the `SET` and `SELECT` statements in SQL queries. The following conditions apply for the supported SQL syntax:

- The supported statements follow the standard SQL-92 conventions.
- Adhering to SQL standards, these statements are not case-sensitive.
- The OLE DB provider does not allow SQL inserts, updates, deletes, or commits; therefore, there is no event notification. You can only retrieve and analyze data.
- String data types are not supported.

Some reporting packages, such as Crystal Reports, hide the SQL syntax by allowing you to use experts and wizards. However, familiarity with SQL syntax may help you in troubleshooting and tuning your SQL commands.

The following figure shows a `SELECT` statement.




With a `SELECT` statement, you can specify the Historian table and columns from which you want to retrieve data. The OLE DB provider establishes the server name at connection time. You can filter the data returned from `SELECT` by specifying a filter option in the `WHERE` clause.

## Supported SELECT Statements Syntax

`SELECT` statements allow you to retrieve data from the Historian database for reporting and analysis. The `SELECT` statements that the OLE DB provider supports follow standard SQL-92 conventions. You can use `SELECT` statements to retrieve information from any of the columns in any of the Historian tables. The `SELECT` statement returns a snapshot of data at the given time of the query.

The order that you specify the columns in the `SELECT` statement controls how the data is returned. For more information on the tables and each of the columns in each table, refer to Historian Database Tables *(on page 2647)* .

**Note:**
To query tag names with spaces in them, you must enclose the full tag name in double quotes. For example, to query the `Copy of 5vkn391s.Simulation00001` tag from the `ihTrend` table, use the following query: `SELECT "Copy of 5vkn391s.Simulation00001" from ihTrend`.

### WHERE Clauses

You can use a `WHERE` clause to specify search conditions in a `SELECT` statement. You can specify a condition for any column in the table using the `WHERE` clause.

For example, you can search all rows of data in the `ihTags` table, where the `DataType` column equals `SingleFloat`. In another instance, you can find all tags that belong to a particular collector. Or, you can search for all tags with a certain poll rate, or range of poll rates, or ones with polling disabled.

You can provide maximum 200 conditions in a `SELECT` statement.

For more information on the columns for each individual Historian table, refer to Historian Database Tables *(on page 2647)*.

#### Example 1: Search for All Single Float Tags

```
SELECT* FROM ihtags WHERE datatype=singlefloat
```

#### Example 2: Specify Query Parameters to Obtain String Data

```
SELECT* FROM ihrawdata WHERE tagname=SimulationString00001

AND samplingmode=interpolated

AND IntervalMilliseconds=1H
```

In this example, you change the `SamplingMode` column from the default value of `Calculated` to `Interpolated` in order to retrieve string data.

**Example 3: Use a WHERE Clause to Specify a Time Range**

```
SELECT* FROM ihmessages WHERE timestamp>bom
```

**Example 4: Use a Complex WHERE Clause to Find All Tags With a Specific Name and Description Pattern**

```
SELECT* FROM ihtags
WHERE(tagname LIKE '*001*' AND description LIKE '*sim*')
OR (tagname LIKE '*02*'
AND (description LIKE '*sec*' OR description LIKE '*sim*'))
AND (timestamptype=source OR timestamptype=collector)
```

For more information on building complex `WHERE` clauses, see Logical Operators and Parenthetical Expressions.

**ORDER BY**

If you do not specify `ORDER BY`, the output of the row order cannot be assumed. For example, if you want to order the rows returned from the `ihCollectors` table by the `CollectorName` column, you must include that column name in `ORDER BY`.

As a more common example, when requesting timestamps with data, use the `Timestamp` column with `ORDER BY` to ensure that the samples are sorted in order by time.

`ORDER BY` sorts the returned records by one or more specified columns in either ascending or descending order. By default, the ascending order is considered. You can order results by one or more columns. If you sort by multiple columns, the sorting priority begins with the first column listed in the query, and then the next column, and so on.

| Abbre-viation | Description |
|---|---|
| `ASC` | Specifies that the values must be sorted in ascending order, from lowest value to highest value. |
| `DESC` | Specifies that the values must be sorted in descending order, from highest value to lowest value. |

The OLE DB provider treats `Null` values as the lowest possible values. It processes `ORDER BY` before it performs any `RowCount` truncation.

**Example 1: Retrieve Collectors in Descending Order Sorted by the Collectorname Column**

```
SELECT * FROM ihcollectors ORDER BY collectorname DESC
```

**Example 2: Retrieve Messages in Ascending Order Sorted by Timestamp and Other Columns**

```
SELECT * FROM ihmessages
WHERE timestamp>='5-oct-2001 00:00:00'
AND timestamp<='18-jan-2002 00:00:00'
ORDER BY timestamp, topic, username, messagenumber, messagestring
```

### TOP

With the `TOP` predicate, you can limit the number of rows returned to a specified number or percentage of rows. And then, enter the rest of the query. Typically, you include `ORDER BY` in the query to sort the rows in a specified order.

When you select the top number or top percentage of rows, the returned value is limited by the `RowCount`. For instance, suppose you want the top 30 percent of rows from a query that can return a possible 10,000 rows, but the `RowCount` is set to 1000. The percentage logic processes the 3000 rows first, then it reduces the number to 1000 rows, as specified by `RowCount`. The final result returns 1000 rows, even though the top 30 percent is processed first. Use a `SET` statement or `WHERE` clause to change or disable the `RowCount` behavior.

**Example 1: Return the Top 40 Tags in Alphabetical Order**

```
SELECT TOP 40 * FROM ihtags ORDER BY Tagname
```

**Example 2: Return the Top 10 Most Recent Messages**

```
SELECT TOP 10 timestamp, topic, username, messagestring FROM
ihmessages WHERE timestamp<Now ORDER BY timestamp DESC
```

**Example 3: Return the Top 10 Percent, RowCount Disabled**

```
SET rowcount=0
SELECT TOP 10 PERCENT timestamp, topic, username, messagestring
FROM ihmessages WHERE timestamp<Now
ORDER BY timestamp DESC
```

### LIKE

Use the `LIKE` expression when searching for column data similar to a specified text string. By using wildcards, you can specify the text strings that you want to search. You can use the wildcard before and/or after the text that you want to search for. Use an asterisk (*) for multiple unknown characters in a search string. Use a question mark (?) for a single unknown character.

> **Note:**
> You can also use a percentage (%) to select all tags that contain a specific string in the tag name and an underscore (_) to select all tags when you are unsure of only one character in the tag name. You must enclose these wildcard characters in single quotes (for example, `'%'` or `'_'`) when you use them in Historian tag names, but do not use single quotes if you want them to be treated as wildcards in SQL.

**Example 1: Use LIKE With Multiple Character Replacement**

```
SELECT * FROM ihtags WHERE tagname LIKE *.Simulation*

ORDER BY tagname

SELECT * FROM ihtags WHERE tagname LIKE %.Simulation%
```

**Example 2: Use LIKE With Single Character Replacement**

```
SELECT * FROM ihtags WHERE tagname LIKE MYSERVER.Simulation0000?

ORDER BY tagname

SELECT * FROM ihtags WHERE tagname LIKE MYSERVER.Simulation0000'_'

ORDER BY tagname
```

### AS

Use `AS` when you want to control the name of an output column. You can use `AS` in all columns and tables except the `ihTrend` table. In the `ihTrend` table, you can only use `AS` with the `TimeStamp` column.

**Example: Set the Output Column Name**

```
SELECT status, collectorname AS Name, collectortype,

status AS 'The Status', collectordescription FROM ihcollectors
```

### DISTINCT

`DISTINCT` eliminates duplicate rows when all columns are equal. Floating-point values, however, may not compare as expected, depending on the precision. For example, if the

numbers to the right of the decimal point are not equal for all values, similar columns are not eliminated. The columns must be exactly equal to be eliminated.

**Example 1: Retrieve the Set of Unique Data Types Used in an Archive**

```
SELECT DISTINCT datatype FROM ihtags
```

**Example 2: Retrieve the Set of Tags With Raw Data Samples on a Specific Date**

```
SELECT DISTINCT tagname FROM ihRawData WHERE samplingmode=rawbytime
AND timestamp>='11/28/2001' AND timestamp<='11/29/2001'
```

### GROUP BY

`GROUP BY` combines records with identical values in the specified field list into a single record. Then, you can compute an aggregate value for the grouped records. The aggregate column does not exist in the actual table. Another calculated column is created with the results.

**Example: Group Messages by User Name and Topic**

```
SELECT username, topic, COUNT(*) FROM ihmessages
WHERE timestamp >= '1-dec-2001 00:00:00'
AND timestamp <= '7-dec-2001 00:00:00'
GROUP BY username, topic ORDER BY username, topic
```

### SQL Aggregate Functions

SQL aggregate functions perform a calculation on a set of values in a column and return a single value. For instance, when comparing multiple tags, you can retrieve the minimum (`MIN`) of the returned minimum values. You usually use aggregate functions with the `GROUP BY` clause, but it is not required. For more information, see Group By.

**Table 387. Supported Aggregate Functions**

| Function | Description |
|---|---|
| `AVG` | Returns the average of the values in a group. Null values are ignored. |
| `COUNT` | Returns the number of items in a group. Null values are not ignored. |
| `MAX` | Returns the maximum value in a group. Null values are ignored. |
| `MIN` | Returns the minimum value in a group. Null values are ignored. |

**Table 387. Supported Aggregate Functions (continued)**

| Function | Description |
|---|---|
| `SUM` | Returns the sum of all the values in a group. `SUM` can be used with numeric columns only. Null values are ignored. |
| `STDEV` | Returns the statistical standard deviation of all values in a group. Null values are ignored. |
| `STDEVP` | Returns the statistical standard deviation for the population for all values in a group. Null values are ignored. |
| `VAR` | Returns the statistical variance of all values in a group. Null values are ignored. |
| `VARP` | Returns the statistical variance for the population for all values in a group. Null values are ignored. |

#### STDEV, STDEVP, VAR, and VARP

If a variance is defined as the deviation from an average data set value, and *N* is the number of values in the data set, then the following equations apply:

```
VAR = (Sum of Variances)^2 / (N - 1)

VARP = (Sum of Variances)^2 / (N)

STDEV = SquareRoot (VAR)

STDEVP = SquareRoot (VARP)
```

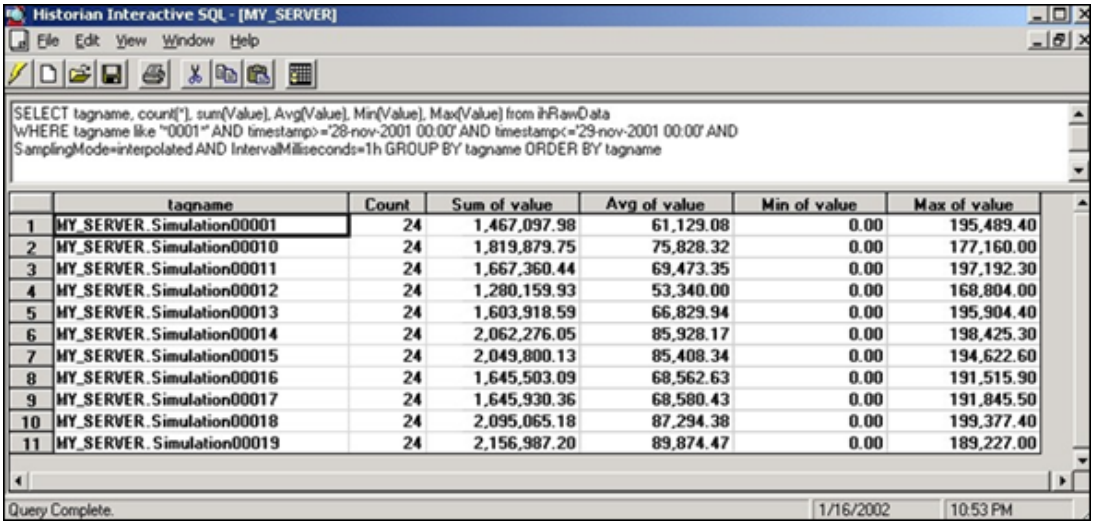
#### Example 1: Retrieve the Total Number of Tags

```
SELECT COUNT(*) FROM ihTags
```

#### Example 2: Calculate Values for Multiple Tags

```
FROM ihrawdata WHERE tagname LIKE '*0001*'

AND timestamp>='28-dec-2001 00:00'

AND timestamp<='29-dec-2001 00:00'

AND samplingmode=interpolated

AND intervalmilliseconds=1h GROUP BY tagname ORDER BY tagname
```

The following figure displays the results of this query. Note the column names (**Sum of value**, **Avg of value**, **Min of value**, and **Max of value**) returned for the calculated columns.




| | tagname | Count | Sum of value | Avg of value | Min of value | Max of value |
|---|---|---|---|---|---|---|
| 1 | MY_SERVER.Simulation00001 | 24 | 1,467,097.98 | 61,129.08 | 0.00 | 195,489.40 |
| 2 | MY_SERVER.Simulation00010 | 24 | 1,819,879.75 | 75,828.32 | 0.00 | 177,160.00 |
| 3 | MY_SERVER.Simulation00011 | 24 | 1,667,360.44 | 69,473.35 | 0.00 | 197,192.30 |
| 4 | MY_SERVER.Simulation00012 | 24 | 1,280,159.93 | 53,340.00 | 0.00 | 168,804.00 |
| 5 | MY_SERVER.Simulation00013 | 24 | 1,603,918.59 | 66,829.94 | 0.00 | 195,904.40 |
| 6 | MY_SERVER.Simulation00014 | 24 | 2,062,276.05 | 85,928.17 | 0.00 | 198,425.30 |
| 7 | MY_SERVER.Simulation00015 | 24 | 2,049,800.13 | 85,408.34 | 0.00 | 194,622.60 |
| 8 | MY_SERVER.Simulation00016 | 24 | 1,645,503.09 | 68,562.63 | 0.00 | 191,515.90 |
| 9 | MY_SERVER.Simulation00017 | 24 | 1,645,930.36 | 68,580.43 | 0.00 | 191,845.50 |
| 10 | MY_SERVER.Simulation00018 | 24 | 2,095,065.18 | 87,294.38 | 0.00 | 199,377.40 |
| 11 | MY_SERVER.Simulation00019 | 24 | 2,156,987.20 | 89,874.47 | 0.00 | 189,227.00 |

### Conversion Functions

The Historian OLE DB provider generally returns data with the `VARIANT` data type. Some OLE DB clients may not understand `VARIANT` data, however, and will require the data to be returned as an integer, float, or string data type. To accommodate this, the OLE DB provider includes the functions described in the following table.

**Table 388. Conversion Functions**

| Function | Description |
|---|---|
| `to_double(column)` | Converts the specified *column* to a double float data type. |
| `to_integer(column)` | Converts the specified *column* to a single integer data type. |
| `to_string(column)` | Converts the specified *column* to a string data type. |

**Note:**

- You must edit the SQL statement manually to add conversion functions.
- You can also use the fully qualified column name (for example, `ihRawData.value`).
- Conversion functions are not available in `WHERE` or `JOIN (ON)` clauses.
- Conversion functions cannot be used within aggregate functions.

**Example: Convert Values to Double Float**

```
select timestamp, to_double(value), quality from ihRawData
```

### JOIN

A table join is an operation that combines rows from two or more tables. You can join as many tables as you want within one `JOIN` statement. When you use a table `JOIN` in a `SELECT` statement, you must specify the column name and table when selecting the columns that you want to compare. The syntax for table joins follows standard SQL language format.

**Table 389. Supported Join Operations**

| Supported Join Feature | Description |
|---|---|
| Inner Join | Combines records from two tables whenever there are matching values. |
| Left Join or Left Outer Join | Returns all of the rows from the left (first) of two tables, even if there are no matching values for records in the right (second) table. |
| Right Join or Right Outer Join | Returns all of the rows from the right (second) of two tables even if there are no matching values for records in the left (first) table. |
| Full Join or Outer Join | Returns all rows in both the left and right tables. Any time a row has no match in the other table, `SELECT` list columns from the other table contain null values. When there is a match between the tables, the entire result set row contains data values from the base tables. |
| Cross Join | Returns all rows from the left table. Each row from the left table is combined with all rows from the right table. |

**Table 389. Supported Join Operations (continued)**

| Supported Join Feature | Description |
|---|---|
| Old Join syntax | Simply selects columns from multiple tables using the `WHERE` clause without using the `JOIN` keyword. |

Table joins are a powerful tool when organizing and analyzing data. A few examples are included in this section. However, refer to the documentation for your third-party reporting software for more complete information on building more complex queries.

#### JOIN Operations Rules

The following rules apply when working with `JOIN` operations for the Historian OLE DB provider:

- You cannot join a table with itself.
- You cannot join any table with the `ihTrend` or `ihQuerySettings` tables.

The following examples display different types of joins with the `ihComments` table. Comments themselves are not usually that useful unless they are combined with data, as you do with the `JOIN` statements in the following examples.

#### Example 1: Perform an Inner Join to Retrieve Only Data With Associated Comments

```
SELECT d.timestamp, d.tagname, d.value, c.username, c.comment
FROM ihrawdata d INNER JOIN ihcomments c
ON c.tagname=d.tagname AND c.timestamp=d.timestamp
WHERE d.tagname LIKE '*0001*'
ORDER BY d.timestamp, d.tagname, c.username, c.comment
```

#### Example 2: Perform a Left Outer Join to Retrieve All Data With and Without Comments

```
SELECT d.timestamp, d.tagname, d.value, c.comment FROM ihrawdata d
LEFT OUTER JOIN ihcomments c
ON c.tagname=d.tagname AND c.timestamp=d.timestamp
WHERE d.tagname LIKE '*0001*' ORDER BY d.timestamp, d.tagname
```

### Example 3: Perform a Right Outer Join to Retrieve All Comments and Their Accompanying Data

```
SELECT d.tagname, d.timestamp, d.value, c.comment FROM ihrawdata d
RIGHT OUTER JOIN ihcomments c
ON c.tagname=d.tagname AND c.timestamp=d.timestamp
WHERE d.tagname LIKE '*0001*' ORDER BY d.tagname, d.timestamp
```

### Example 4: Perform a Cross Join

```
SELECT * FROM ihCollectors CROSS JOIN ihArchives
```

### Example 5: Perform a Cross Join (Older Syntax)

```
SELECT ihTags.Tagname, iharchives.Filename FROM ihTags, ihArchives
```

### Example 6: Join the ihMessages and ihArchives Tables

This example uses `SET StartTime` before the `SELECT` statement. The `SET` statement is necessary because the timestamp criteria in `SELECT` do not narrow down the time range for the `ihMessages` table until after the results have been collected and the join takes place.

```
SET starttime='1-jan-2000'
SELECT a.starttime, a.endtime, m.*
FROM ihmessages m JOIN iharchives a
ON m.timestamp>=a.starttime
AND m.timestamp<=a.endtime WHERE a.iscurrent=true
```

### Example 7: Interleave Data and Messages by Timestamp

```
SELECT d.timestamp, m.timestamp, d.tagname, m.messagestring,
d.value FROM ihRawData d FULL OUTER JOIN ihMessages m
ON d.timestamp=m.timestamp WHERE d.tagname=simulation00001
AND d.timestamp>='30-nov-2001 00:00:00'
AND d.timestamp<='06-dec-2001 00:00:00'
```

### Example 8: Retrieve the Greatest Values Across All Simulation Tags

In the following example, we join the `ihRawData` and `ihTags` tables, because the `ihRawData` table does not contain the `CollectorType` column.

```
SELECT TOP 300 ihRawData.tagname, ihRawData.timestamp,
ihRawData.value, ihRawData.Quality FROM ihRawData
INNER JOIN ihTags ON ihRawdata.Tagname = ihTags.Tagname
```

```
WHERE ihRawData.tagname LIKE simulation*
AND ihRawData.timestamp>=11/28/2001
AND ihRawData.timestamp<=11/29/2001
AND ihRawData.samplingmode=interpolated AND ihRawData.intervalmilliseconds=1H
AND ihTags.datatype!=FixedString
AND ihTags.datatype!=variablestring
AND ihRawData.quality>0
ORDER BY value DESC, timestamp DESC
```

### Example 9: Join the ihComments and ihRawData Tables

```
SET starttime='28-nov-2001 08:00', endtime='29-nov-2001 09:00',
samplingmode=interpolated, intervalmilliseconds=6m
SELECT d.tagname, d.timestamp, d.value, c.storedontimestamp, c.username,
c.datatypehint, c.comment FROM ihcomments c
FULL OUTER JOIN ihrawdata d ON c.tagname=d.tagname
AND c.timestamp=d.timestamp
WHERE d.tagname LIKE '*0001*'
ORDER BY d.tagname, d.timestamp,c.storedontimestamp, c.datatypehint,
c.username, c.comment
```

### Example 10: Report by Tag Description

In the following example, we join the `ihRawData` and `ihTags` tables to get the `Description` column from the `ihTags` table.

```
SELECT d.timestamp, t.description, d.value, d.quality
FROM ihrawdata d INNER JOIN ihtags t ON d.tagname=t.tagname
WHERE d.tagname LIKE '*0001' ORDER BY d.timestamp, t.description
```

### Example 11: Join Three Tables

```
SELECT ihTags.Tagname, ihTags.Description, ihRawData.TimeStamp,
ihRawData.Value, ihRawData.SamplingMode, ihComments.Comment
FROM ihTags ihTags, ihRawData ihRawData, ihComments ihComments
WHERE ihTags.Tagname = ihRawData.Tagname
AND ihRawData.Tagname = ihComments.Tagname
AND ihRawData.Timestamp = ihComments.Timestamp
AND ihRawData.TimeStamp >= {ts '2002-03-01 09:39:00.000'}
AND ihRawData.TimeStamp <= {ts '2002-03-01 09:41:00.000'}
AND ihRawData.SamplingMode = 'RawByTime'
```

```
AND ihTags.Tagname LIKE '%TestTag1%'
```

#### Example 12: Perform a Right Join (Older Syntax)

```
SELECT ihTags.Tagname, ihTags.CollectionInterval, ihCollectors.CollectorName,
ihCollectors.DefaultCollectionInterval
FROM ihTzzz|
```

#### Example 13: Perform a Left Join (Older Syntax)

```
SELECT ihTags.Tagname, ihTags.CollectionInterval, ihCollectors.CollectorName,
ihCollectors.DefaultCollectionInterval
FROM ihTags ihTags, ihCollectors ihCollectors
WHERE
ihTags.CollectionInterval *=ihCollectors.DefaultCollectionInterval
AND ihTags.Tagname LIKE '%TestTag%'
```

### Quotation Marks

You must use quotation marks when you specify a string that contains a space, a comma, or a reserved word. Reserved words are defined by the SQL-92 conventions. Single and double quotes are equivalent in queries.

#### Example: Use Quotes When a Text String Contains a Space

```
SELECT * FROM ihtags WHERE comment LIKE 'alert message'
```

### Timestamp Formats

Timestamps appear not just in the `TimeStamp` columns, but also in columns such as the `StartTime`, `EndTime`, and `LastModified` columns. You can use the date and/or time in a SQL statement that contains a timestamp. Valid date and time formats are as follows:

- System short date and time.
- SQL date and time.
- ODBC date and time.

The time format for system short timestamps is the same as the time format defined in the Windows Control Panel.

When entering a query you should use a period as the decimal separator to separate seconds from milliseconds or microseconds.

When using the SQL date and time, you should always use the English abbreviations for the desired month.

If you enter only a start time, the end time is assumed to be now. For example, if you enter `starttime > yesterday` in a `WHERE` clause, the end time for the query is now, even if you previously set an end time.

If you enter only an end time, the start time is December 31, 1969, 19:00:00.001. If you use this as the start time, you can overload the Historian server and the provider. For example, if you use `timestamp < now`, you might cause an overload.

**Example 1: Use the System Short Date and Time**

```
SET starttime='02/01/2002 11:00:00'
```

**Example 2: Use the SQL Date and Time**

```
SET starttime='14-sep-2001 11:00:00'
```

**Example 3: Use the ODBC Date and Time**

```
SET starttime={ts '2002-06-20 15:34:08'}
```

**Example 4: Set the Start Time to 4 AM Today**

```
SET starttime='04:00:00'
```

**Example 5: Set the Start Time in Milliseconds**

```
SET starttime='7/12/2011 12:03:16.183'
```

**Example 6: Set the Start Time in Microseconds**

```
SET starttime='7/12/2011 12:03:16.178439'
```

**Date and Time Shortcuts**

| Time Segment | Meaning |
|---|---|
| `now` | Now (the time and date that you execute the query) |
| `today` | Today at midnight |
| `yesterday` | Yesterday at midnight |
| `mon` | The previous Monday at midnight |
| `tues` | The previous Tuesday at midnight |
| `wed` | The previous Wednesday at midnight |

| Time Segment | Meaning |
|---|---|
| `thurs` | The previous Thursday at midnight |
| `fri` | The previous Friday at midnight |
| `sat` | The previous Saturday at midnight |
| `sun` | The previous Sunday at midnight |
| `boy` | First day of year at midnight |
| `eoy` | Last day of year at midnight |
| `bom` | First day of month at midnight |
| `eom` | Last day of month at midnight |

**Example 1: Set the Start Time to the First Day of the Month**

```
SET starttime=bom
```

**Example 2: Retrieve Messages Dated Today**

```
SELECT * FROM ihmessages WHERE timestamp>=today
```

### Relative Date and Time Shortcuts

Optionally, you can add or subtract relative time shortcuts to the absolute times.

**Table 390. Relative Date and Time Shortcuts**

| Time Segment | Meaning |
|---|---|
| `s` | Second |
| `m` | Minute |
| `h` | Hour |
| `d` | Day |
| `w` | Week |
| `micro` | Microsec-ond |

You can use relative time shortcuts when defining time intervals. For instance, use these shortcuts when you specify a value for the `IntervalMilliseconds` column.

**Note:**
You cannot use relative time shortcuts to add or subtract microseconds to or from absolute times.

### Example 1: Set the Start Time to 10 Days Before Yesterday and End Time to Today

```
SET starttime=yesterday-10d, endtime=today

SELECT * FROM ihQuerySettings
```

### Example 2: Retrieve the Previous 24 Hours of Messages

```
SELECT * FROM ihMessages WHERE timestamp>=Now-24h
```

### Example 3: Select Data Starting at 1AM Yesterday and Ending Now

```
SELECT * FROM ihrawdata WHERE timestamp>=yesterday+1h AND timestamp<=now
```

### Example 4: Retrieve Raw Data With a 1 Hour (3600000 Milliseconds) Interval Between Returned Samples

```
SELECT * FROM ihrawdata WHERE intervalmilliseconds=1h
```

### Example 5: Retrieve Raw Data With a 100 Microseconds Interval Between Returned Samples

```
SELECT * FROM ihrawdata WHERE intervalmilliseconds=100micro

and starttime>= '7/12/2011 12:03:16.100000' and endtime<='
```

### Example 6: Retrieve This Week's Output to Date

```
SET starttime=Sun, endtime=Now, intervalmilliseconds=1d, samplingmode=rawbytime

SELECT tagname, SUM(value) FROM ihRawData WHERE tagname LIKE *00* GROUP BY tagname
```

## Comparison Operators

**Table 391. Expression Comparisons**

| Comparison Symbol | Meaning |
|---|---|
| < | Less Than |
| > | Greater Than |
| <= | Less Than or Equal |
| >= | Greater Than or Equal |
| = | Equal |

**Table 391. Expression Comparisons (continued)**

| Comparison Symbol | Meaning |
|---|---|
| `!=` | Not Equal |
| `!>` | Not Greater Than |
| `!<` | Not Less Than |
| `BETWEEN x AND y` | Between the values `x` and `y` inclusive, where `x` and `y` are numeric values |

A literal on the left side of the comparison operator is not supported. For example, this statement would fail:

```
SELECT DISTINCT tagname FROM ihRawData WHERE 50>Value
```

But the following statement succeeds since the `Value` column is to the left of the `>` operator:

```
SELECT DISTINCT tagname FROM ihRawData WHERE Value>50
```

**Example 1: Retrieve Tags with a High EGU Greater Than 300**

```
SELECT DISTINCT tagname, loengineeringunits, hiengineeringunits
FROM ihTags WHERE hiengineeringunits > 300
```

**Example 2: Retrieve Tags with a Specific Description**

```
SELECT tagname, description FROM ihTags WHERE description = "aa"
```

**Example 3: Retrieve All Samples Where the Value Exceeds Query Supplied Values**

```
SELECT timestamp, tagname, value FROM ihRawData
WHERE samplingmode=rawbytime AND value>75
```

**Example 4: Retrieve All Samples Where the Value is Between Query Supplied Values**

```
SELECT timestamp, tagname, value FROM ihRawData
WHERE samplingmode=lab AND value BETWEEN 25 AND 75
```

**Example 5: Retrieve All Tag Names Starting with an A or B**

```
SELECT * FROM ihtags WHERE tagname < 'C'
```

### Logical Operators

The following logic operators are supported:

- `AND`
- `OR`
- `NOT`

**Example 1: Use the AND Logical Operator**

```
SELECT * FROM ihTags WHERE Tagname LIKE 'Simulation*'
AND CollectionInterval<3000
```

**Example 2: Use the OR Logical Operator**

```
SELECT * FROM ihTags WHERE Tagname LIKE 'ComputerName.Simulation*'
OR tagname LIKE '*String*'
```

**Example 3: Use the NOT Logical Operator**

```
SELECT * FROM ihTags WHERE NOT Datatype=SingleFloat
```

**Example 4: Use the NOT Logical Operator With a LIKE Expression**

```
SELECT * FROM ihTags WHERE Tagname NOT LIKE '*String*'
```

### Parenthetical Expressions

Parentheses control the order of evaluation of the logical operators in an expression. The OLE DB provider supports parentheses in a `WHERE` clause. You can use multiple sets of parentheses, and nest parenthetical expressions.

**Example 1: Use Parentheses**

```
SELECT * FROM ihTags
WHERE (tagname LIKE *001 AND description="aa") OR tagname LIKE *002
```

**Example 2: Use Parentheses with Logical Operators and Timestamps**

```
SELECT * FROM ihRawData WHERE tagname=Simulation00001 AND
(Timestamp=>Tu AND Timestamp<=Wed OR Timestamp>=Fri AND time
```

**Example 3: Use Multiple Sets of Parentheses**

```
SELECT * FROM ihtags
WHERE (tagname LIKE '*001*' AND description LIKE '*sim*') OR
(tagname LIKE '*02*' AND (description LIKE '*sec*' OR description LIKE '*sim*'))
```

## Supported SET Statement Syntax

The use of `SET` statements is not mandatory because you can also specify query parameters in a `WHERE` clause. However, `SET` statements can make your queries more readable. By using `SET` statements, you can

save time by simplifying `SELECT` queries, because you do not have to retype query parameters each time you issue a new `SELECT` statement. The `SET` parameters persist for the entire session.

With a `SET` statement, you can define various defaults for your queries to use, such as:

- The start date and time of the selected data
- The end date and time
- The calculation mode
- The number of rows returned
- The data sampling mode

For more information, refer to ihQuerySettings Table *(on page 2709)*.

When entering numbers, do not use a thousands separator. For example, if you want to set a collection interval to 7,000 milliseconds, use the following code:

```
SET IntervalMilliseconds = 7000
```

### Correct SET Without Comma to Separate Thousands Place

Multiple `SET` statements in the same command are not supported. Combine multiple variables in the same `SET` statement.

**Correct:**

```
SET starttime=yesterday-10d, endtime=today, samplingmode=interpolated
```

**Incorrect:**

```
SET starttime=yesterday-10d

SET endtime=today

SET samplingmode=interpolated
```

## SET Variables

The following table outlines the supported SQL variables and settings that you can use in a `SET` statement. If you do not change any variables using the `SET` statement or a `WHERE` clause in your `SELECT` statement, the default session variables are considered. You can apply any of the variables described in the following table to the current session. In turn, these settings are used when retrieving information from the Historian database tables. `SET` variables persist from statement to statement.

Some session variables that you define with the `SET` statement accept abbreviations. You must type at least the abbreviation for the statement to work. For instance, for the `CalculationMode` setting, you can

enter the abbreviation `Interp` for the `Interpolated` setting. The accepted abbreviations are highlighted in bold in the following table.

**Table 392. SET Statement Variables**

| Variable | Description | Default Setting |
|---|---|---|
| `StartTime` | A valid date and time string, such as:<br>• `StartTime = '14-sep-200111:00:00'`<br>• `StartTime = Now -1h`<br>• `StartTime = '02/01/199811:00:00'`<br>• `StartTime = {ts '2002-06-20 15:34:08'}`<br>• `StartTime = '7/12/201112:03:16.100000'` | Two hours prior to execution of the query. |
| `EndTime` | A valid date and time string, such as:<br>`EndTime = '14-sep-200112:00:00'` | The current time that you execute the query. |
| `SamplingMode` | String that represents the mode of sampling data from the archive:<br>• `CurrentValue`<br>• `Interpolated`<br>• `InterpolatedtoRaw`<br>• `RawByTime`<br>• `RawByNumber`<br>• `Calculated`<br>• `Lab`<br>• `LabtoRaw`<br>• `Trend`<br>• `TrendtoRaw`<br>• `Trend2`<br>• `TrendtoRaw2`<br>• `RawByFilterToggle` | `Calculated` |
| `Direction` | String that represents the direction of data sampling from the archive, beginning at the start time. `Direction` applies to the `RawByTime` and `RawByNumber` sampling modes: | `Forward` |

**Table 392. SET Statement Variables (continued)**

| Variable | Description | Default Setting |
|---|---|---|
| | • `Forward`<br>• `Backward` | |
| `NumberOfSamples` | Any positive integer that represents the number of samples from the archive to retrieve. Do not enter a thousands separator. For example, enter `1000` and not `1,000`.<br><br>Samples are evenly spaced within the time range defined by start and end times for most sampling modes. For the `RawByNumber` sampling mode, the `NumberOfSamples` attribute determines the maximum number of values to retrieve. For the `RawByTime` sampling mode, the `NumberOfSamples` is ignored. | `0` (use `IntervalMilliseconds`) |
| `IntervalMilliseconds` | Any positive integer that represents the interval (in milliseconds) between returned samples.<br><br>For example:<br><br>• If you run a query with `'IntervalMilliseconds = 100'`, it returns samples in 100-millisecond intervals.<br>• If you run a query with `'IntervalMilliseconds = 100micro'`, it returns samples in 100-microsecond intervals. | `60000` (one minute) |
| `CalculationMode` | The `CalculationMode` column applies only if the `SamplingMode` is set to `Calculated`. It represents the type of calculation to perform on archive data:<br><br>• `Average`<br>• `StandardDeviation`<br>• `Total`<br>• `Minimum`<br>• `MaximumCount`<br>• `RawAverage`<br>• `RawStandardDeviation`<br>• `RawTotal`<br>• `MinimumTime`<br>• `MaximumTime`<br>• `Count`<br>• `TimeGood`<br>• `FirstRawValue` | `Average` |

**Table 392. SET Statement Variables (continued)**

| Variable | Description | Default Setting |
|---|---|---|
| | • `FirstRawTime`<br>• `LastRawValue`<br>• `LastRawTime`<br>• `TagStats` | |
| `FilterTag` | A valid tagname used to define the filter. For example:<br>`FilterTag = 'SimulationString00001'`<br>You can specify only a single tag ID can be specified in the `FilterTag`. Wildcards are not supported. `FilterTag` is used in conjunction with `FilterValue`, `FilterComparisonMode`, and `FilterMode`. | An empty space (meaning `FilterTag` is not used) |
| `FilterMode` | String that represents the type of time filter:<br>• `ExactTime`<br>• `BeforeTime`<br>• `AfterTime`<br>• `BeforeAndAfterTime`<br>For example, `AfterTime` indicates that the filter condition should be `True` starting at the timestamp of the archive value that triggered the `True` condition and leading up to the timestamp of the archive value that triggered the `False` condition. `FilterMode` is used in conjunction with `FilterValue`, `FilterComparisonMode`, and `FilterTag`. | `BeforeTime` |
| `FilterComparisonMode` | String that represents the type of comparison to be made on the filter comparison value:<br>• `Equal`<br>• `EqualFirst`<br>• `EqualLast`<br>• `NotEqual`<br>• `LessThan`<br>• `GreaterThan`<br>• `LessThanEqual`<br>• `GreaterThanEqual`<br>• `AllBitsSet` | `Equal` |

**Table 392. SET Statement Variables (continued)**

| Variable | Description | Default Setting |
|---|---|---|
| | • `AnyBitSet`<br>• `AnyBitNotSet`<br>• `AllBitsNotSet`<br><br>If you enter `FilterTag` and `FilterComparisonValue` in the `SET` statement, time periods are filtered from the results where the filter condition is `False`. `FilterComparisonMode` is used in conjunction with `FilterValue`, `FilterMode`, and `FilterTag`. | |
| `FilterExpression` | An expression that includes multiple filter conditions. You can use `FilterExpression` instead of `FilterTag`, `FilterComparisonMode`, and `FilterValue`.<br><br>`FilterExpression = 'BatchID=B1'`<br><br>While using `FilterExpression`, the expression is passed within single quotes, and for complex expressions we write the conditions within parentheses. There is no maximum length for `FilterExpression`. | |
| `FilterValue` | String that represents the value with which to compare the filter tag to determine the appropriate filter times. Wildcards are not supported. Do not use a comma for the thousands separator.<br><br>For example:<br><br>`FilterValue = 'ABCD-1086031382099'`<br><br>The `FilterValue` is used in conjunction with `FilterComparisonMode`, `FilterMode`, and `FilterTag`. | An empty space (meaning filtering is not used) |
| `TimeZone` | String that represents the type of time zone that should be applied to timestamps:<br><br>• `Client`<br>• `Server`<br>• Explicit bias number (number of minutes from GMT)<br><br>For example, an explicit bias number of `300` represents 300 minutes from GMT.<br><br>**Note:**<br>Time zones are not supported on Windows 9x computers. | `Client` |

**Table 392. SET Statement Variables (continued)**

| Variable | Description | Default Setting |
|---|---|---|
| `DaylightSavingTime` | Indicates whether Daylight Saving Time logic should be applied to timestamps. Valid values:<br>• `True`<br>• `False` | Date and time settings in your Windows Control Panel |
| `RowCount` | A number that indicates the maximum number of rows that can be returned. `0` indicates there is no limit to the number of rows returned. | `5000` |

## SET Statements and Variables Examples

If you do not change any variables using the `SET` statement or a `WHERE` clause in your `SELECT` statement, the default session variables are considered. For instance, if you do not specify a start and end time for your collected data, the data output from a `SELECT` statement will be the last two hours prior to execution of the query.

For example, if you want to `SELECT` all of the messages from the `ihMessages` table for the last day, you must explicitly state that you want the messages from the last day in the query. Otherwise, only the messages from the last two hours are displayed when you run the query, since that is the default setting.

`SET` statement variables persist during a session until changed. You can combine the `SET` statement on the same line as the `SELECT` statement.

### Perform a Simple SET

```
SET samplingmode=currentvalue
```

### Perform Multiple SETs

```
SET starttime='14-sep-2001 11:00:00', endtime='14-sep-2001 12:00:00',

samplingmode=interpolated, intervalmilliseconds=
```

### Prepare for a RawByTime Query

```
SET starttime='14-sep-2001 11:00:00', endtime='14-sep-2001 12:00:00',

samplingmode=rawbytime
```

### Prepare for a RawByNumber Query

```
SET starttime='14-sep-2001 11:00:00', samplingmode=rawbynumber,

numberofsamples=10, direction=backward
```

### Prepare for One Hour Minimums

```
SET starttime='15-sep-2001 00:00:00', endtime='16-sep-2001 00:00:00',

samplingmode=calculated, intervalmilliseconds=36
```

### Prepare for a Filtered Data Query

```
SET starttime='14-sep-2001 11:00:00', endtime='14-sep-2001 12:00:00',

samplingmode=current, filtertag='MY_SERVER.simul
```

### Throttle Results with a SET Statement

```
SET ROWCOUNT = 4

SELECT Tagname FROM ihTags
```

## Combined SET and SELECT Statements

The OLE DB provider allows you to execute one `SELECT` statement and one `SET` statement per query. Enter a space or a line break to indicate the end of a statement in a query. You do not need to use a semicolon (;) at the end of the line or statement.

### Use SET and SELECT Statements on the Same Line

```
SET samplingmode=interpolated SELECT * FROM ihquerysettings
```

### Use SET and SELECT Statements on Different Lines

```
SET samplingmode=calculated, starttime=yesterday, endtime=today

SELECT * FROM ihquerysettings
```

## Parameterized SQL Queries

Parameterized SQL queries allow you to place parameters in an SQL query instead of a constant value. A parameter takes a value only when the query is executed, which allows the query to be reused with different values and for different purposes. Parameterized SQL statements are available in some analysis clients and Historian SDK.

For example, the following query contains a parameter for the collector name:

```
SELECT* FROM ihtags WHERE collectorname=? ORDER BY tagname
```

If your analysis client passes the parameter `iFIX_Albany` along with the query, it looks like follows when executed in Historian:

```
SELECT* FROM ihtags WHERE collectorname='iFIX_Albany' ORDER BY tagname
```

The advantage of using parameterized SQL queries is that you can prepare them ahead of time and reuse them for similar applications without having to create distinct SQL queries for each case. For instance, you can use the previous example in any context where you want to get tags from a collector. You can also use parameterized queries with dynamic data, where you do not know what the values will be until the statement is executed.

If your analysis client supports parameterized queries, it will automatically pass the parameter data along with a named query for Historian to process. In the case of multiple parameters, the analysis client will read the named query, and order the parameters to match.

**Note:**
You cannot use parameters to substitute table names or columns in a query.

## Multiple Parameters

To create a query with multiple parameters, place a question mark (?) for every parameter whose value you want to substitute in the query. For example, if you want an SQL query to match two `WHERE` conditions, `collectorname` and `tagname`, use the following parameterized query:

```
SELECT* FROM ihtags WHERE collectorname=? AND tagname like ? ORDER BY tagname
```

When executed, the parameterized SQL query will add the parameters as they are received from the analysis application. In the previous example, the `collectorname` parameter would be received first, followed by the `tagname` parameter. Your analysis client will order the parameters based on the query it is running.

**Note:**
If you want to enter wildcard data in your parameterized queries, include the wildcard characters as part of the parameter. For instance, in the previous example, if you want to find any tagnames with the string iFIX in them, pass it the `*iFIX*` parameter.

## Optimize the Query Performance

To optimize query performance, follow these guidelines:

- Perform `GROUP BY` on the server whenever available. For instance, Crystal Reports gives you the option to group on the server as opposed to the client.
- Use `DISTINCT` to eliminate duplicate rows.
- Be specific when specifying tag names. For instance, when using wildcards, be as specific as possible.
- Limit the duration between start and end times.
- Get as precise a data type as possible to improve storage efficiency and allow reporting tools such as Power BI or Crystal Reports to properly format the data in reports.
- Do not rely on `TOP` or `ROWCOUNT` to optimize performance because they do not change the load on the archive or network but instead they just limit what is returned to the caller.

# Troubleshooting and Frequently Asked Questions

## Troubleshooting

The following sections outline what to do if the following problems occur:


- Cannot Connect With the Historian Interactive SQL Application *(on page 2638)*
- Cannot Log Into the Historian Interactive SQL Application *(on page 2639)*
- Cannot Get Historian OLE DB provider Data *(on page 2639)*
- Samples Do Not Run *(on page 2640)*
- Time Zones Do Not Work *(on page 2640)*
- Cannot Get String Data From the ihRawData Table *(on page 2640)*
- Timestamps Include Only the Previous Two Hours *(on page 2641)*
- Row Count Less Than Expected *(on page 2641)*
- Linked Server Not Working *(on page 2641)*
- SET Not Applied to SELECT When Using a Linked Server *(on page 2641)*
- Client Crashes When Using Historian OLE DB provider *(on page 2641)*


The sections that follow the answers to this list describe frequently asked questions. These answers may help you when you are first configuring and using the Historian OLE DB provider.

## Cannot Connect With the Historian Interactive SQL Application

When the OLE DB provider connects to the archiver, a connection message is generated and logged to the archiver messages list. If you are having problems connecting with the Historian Interactive SQL application (`ihSQL.exe`), you will either not see a connection message or see a connection error instead.

If you suspect that you are having problems connecting to the archiver, follow these steps:

1. Open Historian Administrator.
2. Select **Messages**.
   The message fields appear in the main window.
3. In the **Priority** group box, select the **All** option.
4. In the **Topic** drop-down list, select **All Topics**.
5. Select **Search**.
   A list of messages appears on the right side of the window.
6. Scroll through the list of connection messages and look for any missing connections or connection errors.
   Connections denied due to security display the user name passed to the archiver. For example, the message would be similar to this:

   ```
   Unknown(\kmckenna) failed login at 03/01/2002 04:30:58.415 PM.
   ```

## Cannot Log Into the Historian Interactive SQL Application

When you use `ihSQL.exe` for the first time, you may need to select **Run As Administrator**. If you do not do this the first time you use `ihSQL.exe`, you may not be able to log in. After this, you do not need to select **Run as Administrator**.

## Cannot Get Historian OLE DB provider Data

If you cannot get data and you suspect there is a security problem with Historian, follow these steps to confirm that the Historian OLE DB provider is working:

1. Open the Historian Interactive SQL application and connect to the OLE DB provider.
2. Enter the following command:

   ```
   SELECT * FROM ihQuerySettings
   ```

3. Select the **Execute Query** button.
4. Confirm that data appears in the bottom half of the window:
   - If one row of data returns, then the provider is installed and working correctly, but you may have security problems between the provider and the server. You must use a valid Historian username and password.
   - If no rows return, then there is a connection problem between the client and the OLE DB provider.

The `ihQuerySettings` data is internal to the OLE DB provider and does not use any Historian security. Browsing the tables and columns also is unaffected by Historian security and is another way to confirm the connection between the client and provider.

## Samples Do Not Run

If you follow the recommended installation procedures, you should not have any difficulty in running the sample reports. If you do encounter any problems, they are likely to relate to the locations of files.

For example, if you are using Crystal Reports, check that you changed the server name. If the server name is incorrect, the data links will not update correctly. See Changing the Server Name *(on page 2588)* for directions on how to change it.

## Time Zones Do Not Work

If you are using Windows 9x, times zones are not supported on this operating system. Returned data displays the client time zone.

If you are expecting a server or explicit bias time zone and a client time zone displays, check the defaults in the `ihQuerySettings` table. By default, the `TimeZone` column is set to `Client`. See Supported SET Statement Syntax *(on page 2629)* for more information on setting defaults using a `SET` statement, or see WHERE Clauses for information on specifying a time zone in the `SELECT` statement.

## Cannot Get String Data From the ihRawData Table

The Historian OLE DB provider, by default, does not return string data types in the `ihRawData` table. This is because the default `SamplingMode` value is `Calculated`. You have to change the `SamplingMode` value to `Interpolated` using the `SET` statement or a `WHERE` clause.

For example, this query does not return interpolated data:

```
SELECT * FROM ihRawData

WHERE tagname = simulationstring00001
```

However, this query does:

```
SELECT * FROM ihRawData

WHERE tagname = simulationstring00001 AND

samplingmode = interpolated
```

And so does this query:

```
SET samplingmode=interpolated

SELECT * FROM ihRawData

WHERE tagname = simulationstring00001
```

## Timestamps Include Only the Previous Two Hours

By default, the data returned only includes data from two hours prior to the execution of the query. If you want to change the time frame of the data query, you need to specify a start and end time in a `SET` statement, or use a `WHERE` clause to specify a date and time period.

## Row Count Less Than Expected

By default, all queries return up to a maximum of 5,000 rows. If you want to change the maximum number of rows returned, you can specify another `RowCount` value in a `SET` statement, or use the `TOP` predicate in your `SELECT` statement.

If you specify `RowCount=0` in the `SET` statement, the `RowCount` limit is disabled. However, the `RowCount` is not actually unlimited. It can be constrained by other factors such as the time interval, or by using the `TOP` predicate in your `SELECT` statement.

## Linked Server Not Working

Check that you selected the **Select the Level Zero Only** and **Allow in Process** options in the **Provider Options** window. You may have forgotten to set them when you were creating your linked server. These are the only two options that should be selected.

## SET Not Applied to SELECT When Using a Linked Server

Make sure that the `SET` and `SELECT` statements are combined in the same query. If you open the connection and only perform the `SET`, as shown below, the `SET` parameters only get applied for the duration of the connection.

```
SELECT * FROM OPENQUERY(linkedserver, 'SET SamplingMode=interpolated')
```

The `SamplingMode` option in the previous example does not get applied to the next `OPENQUERY` that you perform with a `SELECT` statement. The `SET` statement only gets applied to the query if it is included with the `SELECT` statement. See Use OPENQUERY to Access a Linked Server for examples of how to include the `SET` statement with a `SELECT` statement.

## Client Crashes When Using Historian OLE DB provider

Ensure that your client is initializing `COM` in `Apartment` threaded mode.

## Frequently Asked Questions

The following sections outline some of the most frequently asked questions when using the Historian OLE DB provider. These questions include:

**How Are Historian Calculation Modes and SQL Aggregate Functions Different?**

You can extract calculated data from Historian by setting the `SamplingMode` column to `Calculated` and the `CalculationMode` column to the desired calculation mode type. You can use SQL aggregate functions to perform a calculation on a set of values, possibly calculated data, for the same tag or different tags and return a single value.

For instance, when comparing multiple tags you could retrieve the minimum (`MIN`) value of each tag. By setting calculation modes, Historian Administrator only calculates the minimum for each tag over a given time period. By using aggregate functions, the Historian OLE DB provider calculates the minimum value across all tags (all rows in a table), in other words, the minimum of all minimum tag values.

**How Are the ihTrend and ihRawData Tables Different?**

Typically, you use the `ihTrend` table when you want to compare multiple tags at the same time. The OLE DB provider needs to synchronize all the returned data by time, so it takes more time to query the `ihTrend` table than to query the `ihRawData` table. You can retrieve multiple tags from the `ihRawData` table, but the tags are not synchronized.

**Can I Run Multiple Applications Using the OLE DB provider?**

Yes. For instance, you can use the OLE DB provider to access data using Crystal Reports and VisiconX at the same time.

**Can I Retrieve Data From Multiple Servers?**

Yes. The OLE DB provider can have connections to multiple servers at the same time. Each is regarded as a separate session.

You cannot mix multiple servers in the same `SELECT` statement, except indirectly in a linked server in Microsoft SQL Server. Crystal Reports allows you to create subreports inside of a report. Each report gets its own data source (which would be a Historian server) and its own `SELECT` query. However, the reports cannot share data. You can have multiple VisiconX data controls in one picture, each going to a different server.

For instance, say you run iFIX and Crystal Reports at the same time. From the VisiconX page, establish a connection to the Historian OLE DB provider and perform a query on Server1. Next, run a report from Crystal Reports connecting to the same provider, but with

a connection to a different server, Server2. After you run the report and go back to the VisiconX page, you will notice that VisiconX is still connected to Server1. If you refresh the control, it uses the same settings and server as it did before. The provider maintains these two sessions separately, each with its own `SET` parameters.

So, in general, you can access multiple servers, but the data from each server remains independent. You must work with linked servers in Microsoft SQL Server to combine data from multiple servers.

**What is a Session?**

A session is defined as an OLE DB connection. You can run multiple server connections to the OLE DB provider. Each is regarded as a separate session.

You can have multiple sessions with multiple clients, such as Crystal Reports and iFIX. Multiple sessions between a client computer and a server computer count as one licensed session.

**How Do the > and >= Operators Work With Timestamps?**

The > and >= comparison operators, when used with `timestamp`, return the same values. For example, this SQL statement...

```
SELECT * FROM ihRawData WHERE tagname=simulation00001 AND
timestamp>='4/1/2001 01:50:00' AND
timestamp<='4/1/2001 04:00:00' AND
samplingmode=lab
```

...returns exactly the same first result as this statement:

```
SELECT * FROM ihRawData WHERE tagname=simulation00001 AND
timestamp > '4/1/2001 01:50:00' AND
timestamp <= '4/1/2001 04:00:00' AND samplingmode=lab
```

The first result is timestamped at 1:51:00.

**How Do I Throttle Query Results?**

The default maximum row count is 5,000. If you want to throttle the number of rows that you return in a single query, you can do one of the following:

- Use the `SET` statement to specify the `RowCount` to a specific number of rows.
- Use the `TOP` predicate to specify the top number or top percentage of rows that you want to return.
- Use the `MaxRecords` property on the `recordset` object in ADO.

**When Should I Use Excel Instead of the Historian Excel Add-In?**

Use the Excel Add-In when you want to get data into Microsoft Office 2003, 2007 or 2010 (32-bit or 64-bit). Use Excel with the Historian OLE DB provider when you want to perform advanced filtering, sorting, and joining of data. For other features that you might to perform with Excel and the Historian OLE DB provider, see Microsoft Excel *(on page 2592)*.

**Why Is the Raw Sample at the Start Time Not Returned?**

Historian OLE DB provider does not return raw samples with timestamps that match the start time. If you want to include the start time, you need to set the start time to a time earlier than the first raw sample desired.

> **Note:**
> This only applies to `RawByTime` sampling mode and not `RawByNumber`.

For example, if you want to return raw samples starting at 11/28/2001 18:25:00 you can use `1/28/2001 18:24:59` as the start time. For example, you would enter the following SQL command:

```
SELECT TimeStamp, Tagname, Value FROM ihRawData

WHERE (SamplingMode = 'RawByTime') AND

(TimeStamp >= {ts '11/28/2001 18:24:59'})

ORDER BY TimeStamp ASC
```

If your timestamps are using millisecond resolution, you can retrieve timestamps starting at 11/28/2001 18:24:59.999 to prevent any sample prior to 18:50:00 from being returned.

**What Username and Password Is Used if Not Specified in the Connect String?**

If you leave a username and password empty in the connect string, then the user that owns the process, usually the currently logged-in user, is passed to the archiver for validation. For example, this statement leaves the username and password empty:

```
ConnectionString="Provider=ihOLEDB.iHistorian.1;User Id=;Password="
```

This statement also leaves the username and password empty:

```
ConnectionString="Provider=ihOLEDB.iHistorian.1"
```

If you saved username and password information in Historian Administrator or the iFIX WorkSpace for connecting to that server, that information is not used by the OLE DB provider.

**What Is an Array Tag?**

Historian allows you to store a set of values with a single timestamp and single quality and then read the elements back individually or as an array.

On retrieval, if you specify only the tag name, then all elements are returned. If you want to retrieve only an element, you can specify *<TagName>[n]* where *n* is the element number you want to retrieve.

In an array tag:

- The size of the array tag does not need to be configured. The Data Archiver will store the number of elements that were written.
- The maximum number of elements that an array tag can store is 10000. If this limit is exceeded, Historian does not accept any further elements.
- All calculation modes except `TagStats` are supported by array tags. The calculation mode is applied on array elements and not on the array. For example, if you do a minimum on a three-element array, this works like three individual tags. The minimum of element [0] over time is computed and returned as the minimum of element [0]. The Data Archiver does not compute the minimum of element [0], [1], [2] at a single point in time and return that as the minimum of the array.
- When a normal tag is converted to an array tag, on data retrieval, the data of the normal tag cannot be retrieved.

You can query both an array tag and an element of the tag. Each element of the array tag will be displayed in a separate row and they all will have the same timestamp.

**What Is a User-Defined Type?**

Historian gives you the ability to create a new user-defined data type which includes multiple fields of any data type and then create Historian tags of that type. All the regular tag operations can be performed on this tag. You can perform raw and calculated queries on the collected data.

**What Is Not Supported?**

A frequently asked question that may also relate to troubleshooting is what functions are not supported by Historian OLE DB provider. Some of these unsupported items include:

- Concatenation in SQL statements. For example, this syntax does not work:

```
SELECT * FROM ihtags WHERE tagname= "MY_SERVER." + ihtags.Tagname
```

- Calculation in SQL statements. For example, this syntax does not work:

```
SELECT * FROM ihtags WHERE ihrawdata.value * 2 > ihtags.LoEngineeringUnits
```

- SQL inserts, updates, deletes, or commits.
- Ordering by columns not specified in the `SELECT` statement.
- The semicolon (`;`) as a separator between `SET` and `SELECT` statements (which is commonly used in DTS and Oracle). Only a space or line break is necessary.
- Nested `SELECT` statements.
- The `UCASE` macro or other similar SQL syntax.
- `ASYNC` executes in ADO and Visual Basic.
- Bookmarks in ADO and Visual Basic.
- Table creation in SQL.
- The `UNION` statement in SQL.
- The `HAVING` clause in a `SELECT` statement.
- Using comments in a query.
- The `DISTINCT` clause in aggregate functions. For example, this syntax does not work:

```
SELECT Topic, count(DISTINCT *), sum(DISTINCT messagenumber), avg(DISTINCT messagenumber) FROM
 ihmessages GROUP BY topic ORDER BY Topic
```

- A literal on the left side of a comparison operator. SQL-92 standards support this feature, but GE Intelligent Platforms does not currently support it. For example, this syntax does not work:

```
SELECT DISTINCT tagname FROM ihRawData WHERE 50>Value
```

- Analysis of the `ihTrend` table in Crystal Reports or the Microsoft SQL Server DTS application.
- Command or connect timeouts (`Connection.ConnectTimeout`, `Connection.CommandTimeout`, or `Command.CommandTimeout`) in Visual Basic. For example, this syntax does not work:

```
SET adoConn = New ADODB.Connection

adoConn.ConnectionString = "Provider=ihOLEDB.iHistorian.1;User Id=;Password="

adoConn.ConnectionTimeout = 5 ' does nothing

adoConn.CommandTimeout = 5 ' does nothing
```

```
SET cmdTestTimer = New ADODB.Command

SET cmdTestTimer.ActiveConnection = adoConn

cmdTestTimer.CommandText = "SELECT * FROM ihtags"

cmdTestTimer.CommandType = adCmdText

cmdTestTimer.CommandTimeout = 15 ' does nothing
```

# Historian Database Tables

## The Historian Database Tables

The Historian database tables contain read-only data from the Historian archive.

| Table Name | Description |
|---|---|
| ihTags Table *(on page 2651)* | Contains Historian tag configuration information. |
| ihArchives Table *(on page 2659)* | Contains Historian archive configuration information, plus performance statistics for each archive. |
| ihCollectors Table *(on page 2661)* | Contains configuration and status information for each collector connected to the Historian server. |
| ihMessages Table *(on page 2667)* | Contains Historian messages such as alerts, informational topics, and connection information contained in the audit log. |
| ihRawData Table *(on page 2670)* | Contains collected data for each tag in the Historian server. It contains not just raw data, but also calculated and interpolated data. |

| Table Name | Description |
|---|---|
| ihComments Table *(on page 2682)* | Contains the comments associated with the Historian data. |
| ihTrend Table *(on page 2693)* | Another way to look at collected data. Contains a row of data for each unique timestamp. You can use this table to look at your data at a summarized level. You would typically use this table to compare multiple tags with the same timestamp. |
| ihQuerySettings Table *(on page 2709)* | Contains a set of parameters that apply to all queries you make in that session, unless overridden by a `WHERE` clause. |
| ihCalculationDependencies *(on page 2715)* | Contains the calculation dependencies for tags. |
| ihAlarms Table *(on page 2716)* | Contains collected alarms and events data. |
| ihEnumeratedSets Table *(on page 2720)* | Contains information about enumerated sets. |
| ihEnumerated- | Contains information about enumerated states. |

| Table Name | Description |
|---|---|
| States Table *(on page 2721)* | |
| ihUser-Defined-Types Table *(on page 2722)* | Contains information about user-defined data types. |
| ihFields Table *(on page 2723)* | Contains information about fields used in user-defined types. |

The following conditions apply when using these tables:

- You cannot write/update data in these tables.
- Null values are not supported in any column. A blank space is returned when there is no value provided by the Historian server (instead of a `Null` field).
- Almost all columns in these tables support comparison operators except for the following:
  - `SamplingMode`
  - `Direction`
  - `NumberOfSamples`
  - `IntervalMilliseconds`
  - `CalculationMode`
  - `FilterTag`
  - `FilterMode`
  - `FilterComparisonMode`
  - `FilterValue`
  - `FilterExpression`
  - `TimeZone`
  - `DaylightSavingTime`
  - `RowCount`

  These columns only support the `=` comparison operator.

## Historian Security Groups and the Database Tables

A user with membership in the iH Readers security group can access any table in the Historian OLE DB provider, even the `ihArchives` and `ihCollectors` tables. Members of the iH Readers group have read-only access to these tables.

Since the Historian OLE DB provider only supports read-only access to data and does not allow `INSERT` or `UPDATE` operations, no users can make changes to the data in these tables. This includes members of the iH Readers security group and even security administrators in the iH Security Admins security group.

For more information on Historian group rights, refer to Chapter 5 in the *Getting Started with Historian* manual.

## Input Data and Historian Archive Data in Table Columns

There are two types of column data in the Historian OLE DB provider tables: input data and Historian archive data. Input data contains settings stored in the Historian OLE DB provider and has nothing to do with the data stored in the Historian archives. Historian archive data is the data retrieved from the Historian server.

While most columns contain Historian archive data, there are a few columns that contain input data. The following columns, no matter what table they appear in, contain input data and do not originate from the Historian archives:

- `SamplingMode`
- `Direction`
- `NumberOfSamples`
- `IntervalMilliseconds`
- `CalculationMode`
- `FilterTag`
- `FilterMode`
- `FilterComparisonMode`
- `FilterValue`
- `FilterExpression`
- `TimeZone`
- `DaylightSavingsTime`
- `RowCount`

The columns in the previous list are used in a `WHERE` clause to specify query parameters for retrieved data.

## About the Table Descriptions

The following sections describe each table, list each column in the table, and list the data type and description for each column. The following table outlines the data types that are used throughout this chapter.

**Table 393. Column Data Types**

| Data Type | Format of Data |
|---|---|
| `VT_BOOL` | Boolean |
| `VT_BSTR` | String |
| `VT_DBTimeStamp` | Date and Time |
| `VT_I4` | Integer |
| `VT_R4` | Float |
| `VT_R8` | Double Float |
| `VT_UI1` | Short Integer |
| `VT_VARIANT` | Numeric or String |

Also included after each table description are examples of SQL statements used with the specified database table. These examples are only provided to get you started with creating SQL statements with the Historian OLE DB provider. For more detailed information on creating SQL queries, refer to your reporting software documentation.

## ihTags Table

The `ihTags` table contains the set of tag names and the properties of each tag. Each row in the table represents one tag.

| Column Name | Data Type | Description |
|---|---|---|
| `Tagname` | `VT_-BSTR` | `Tagname` property of the tag. |

| Column Name | Data Type | Description |
|---|---|---|
| | | **Note:** There is no length limit for Historian tag names in the Data Archiver. However, different client applications may have their own limits. |
| `Description` | `VT_BSTR` | User description of the tag. |
| `EngUnits` | `VT_BSTR` | Engineering units description of the tag. |
| `Comment` | `VT_BSTR` | User comment associated with the selected tag. |
| `DataType` | `VT_BSTR` | The data type of the tag:<br>• `Scaled`<br>• `SingleFloat`<br>• `DoubleFloat`<br>• `SingleInteger`<br>• `DoubleInteger`<br>• `Quad Integer`<br>• `Unsigned Single Integer`<br>• `Unsigned Double Integer`<br>• `Unsigned Quad Integer`<br>• `Byte`<br>• `Boolean`<br>• `FixedString`<br>• `VariableString`<br>• `BLOB`<br>The data type returned in this column is the data type that you defined in Historian Administrator application. |
| `FixedStringLength` | `VT_UI1` | Zero unless the data type is `FixedString`. If the data type is `FixedString`, this number represents the maximum length of the string value. |
| `CollectorName` | `VT_BSTR` | Name of the collector responsible for collecting data for the specified tag. |

| Column Name | Data Type | Description |
|---|---|---|
| `SourceAddress` | `VT_-BSTR` | Address used to identify the tag at the data source. For iFIX systems, this is the NTF (`Node.Tag.Field`). |
| `CollectionType` | `VT_-BSTR` | Type of collection used to acquire data for the tag:<br>• **Unsolicited:** The collector accepts data from the source whenever the source presents the data.<br>• **Polled:** The collector acquires data from a source on a periodic schedule determined by the collector.<br>**Note:**<br>Not all collectors support unsolicited collection. |
| `CollectionInter-val` | `VT_I4` | The time interval, in milliseconds, between readings of data from this tag.<br>For polled collection, this field represents the time between samples. For unsolicited collection, this field represents the minimum time allowed between samples. |
| `CollectionOffset` | `VT_I4` | The time shift from midnight, in milliseconds, for collection of data from this tag. |
| `LoadBalancing` | `VT_-BOOL` | Indicates whether the data collector should automatically shift the phase of sampling to distribute the activity of the processor evenly over the polling cycle. This is sometimes called phase shifting. |
| `TimeStampType` | `VT_-BSTR` | The timestamp type applied to data samples at collection time:<br>• **Source:** The source delivers the timestamp along with the data sample.<br>• **Collector:** The collector delivers the timestamp along with the collected data. |
| `HiEngineeringU-nits` | `VT_R8` | The high end of the engineering units range. Used only for scaled data types and input scaled tags. |
| `LoEngineeringU-nits` | `VT_R8` | The low end of the engineering units range. Used only for scaled data types and input scaled tags. |

| Column Name | Data Type | Description |
|---|---|---|
| `InputScaling` | `VT_BOOL` | Indicates whether the measurement should be converted to an engineering units value. When set to `False`, the measurement is interpreted as a raw measurement.<br><br>When set to `True`, the system converts the value to engineering units by scaling the value between the `HiScale` and `LoScale` columns. If not enabled, the system assumes the measurement is already converted into engineering units. |
| `HiScale` | `VT_R8` | The high-end value of the input scaling range used for the tag. |
| `LoScale` | `VT_R8` | The low-end value of the input scaling range used for the tag. |
| `CollectorCompression` | `VT_BOOL` | Indicates whether collector compression is enabled for the tag.<br><br>Collector compression applies a smoothing filter to incoming data by ignoring incremental changes in values that fall within a deadband centered around the last collected value. The collector passes (to the archiver) any new value that falls outside the deadband and then centers the deadband around the new value. |
| `CollectorDeadbandPercentRange` | `VT_R4` | The current value of the compression deadband. |
| `ArchiveCompression` | `VT_BOOL` | Indicates whether archive collector compression is enabled for the tag. |
| `ArchiveDeadbandPercentRange` | `VT_R4` | The current value of the archive compression deadband. |
| `CollectorGeneral1` | `VT_BSTR` | The general (or spare) configuration fields for the tag. |
| `CollectorGeneral2` | `VT_BSTR` | The general (or spare) configuration fields for the tag. |
| `CollectorGeneral3` | `VT_BSTR` | The general (or spare) configuration fields for the tag. |
| `CollectorGeneral4` | `VT_BSTR` | The general (or spare) configuration fields for the tag. |

| Column Name | Data Type | Description |
|---|---|---|
| `CollectorGeneral5` | `VT_BSTR` | The general (or spare) configuration fields for the tag. |
| `ReadSecurityGroup` | `VT_BSTR` | The name of the Windows security group that controls the reading of data for the tag.<br><br>Refer to "Implementing Historian Security" in the *Getting Started with Historian* manual for definitions of the various security levels and groups. |
| `WriteSecurityGroup` | `VT_BSTR` | The name of the Windows security group that controls the writing of data for the tag.<br><br>Refer to "Implementing Historian Security" in the *Getting Started with Historian* manual for definitions of the various security levels and groups. |
| `AdministratorSecurityGroup` | `VT_BSTR` | The name of the Windows security group responsible for controlling configuration changes for the tag. |
| `Calculation` | `VT_BSTR` | The equation for the calculation performed for the tag. |
| `LastModified` | `VT_DBTimeStamp` | The date and time that the tag configuration was last modified. The time structure includes milliseconds. |
| `LastModifiedUser` | `VT_BSTR` | The username of the Windows user who last modified the tag configuration. |
| `CollectorType` | `VT_BSTR` | The type of collector responsible for collecting data for the tag:<br><br>• `Undefined`<br>• `iFIX`<br>• `Simulation`<br>• `OPC`<br>• `File`<br>• `iFIXLabData`<br>• `ManualEntry`<br>• `Simulation`<br>• `Other` |

| Column Name | Data Type | Description |
|---|---|---|
| `StoreMilliseconds` | `VT_BOOL` | Indicates whether milliseconds are recorded in timestamps.<br><br>If not enabled, the time resolution is in seconds instead of milliseconds. Maximum data compression is achieved when this option is set to `False`. This is the optimum setting for most applications.<br><br>**Note:**<br>`StoreMilliseconds` returns `False` in Historian v4.5 and later. |
| `TimeResolution` | `String` | Indicates the timestamp resolution in seconds, milliseconds, or microseconds. |
| `UTCBias` | `VT_I4` | The time zone bias for the tag. Time zone bias is used to indicate the natural time zone of the tag expressed as an offset from UTC (Universal Time Coordinated) in minutes.<br><br>UTC is the international time standard, the current term for what was commonly referred to as Greenwich Mean Time (GMT). |
| `AverageCollectionTime` | `VT_I4` | The average time it takes to execute the calculation tag since you started the Calculation collector. |
| `CollectionDisabled` | `VT_I4` | Indicates whether collection is enabled (`0`) or disabled (`1`) for the tag. The default setting is enabled (`0`). |
| `CollectorCompressionTimeout` | `VT_I4` | Indicates the maximum amount of time the collector will wait between sending samples to the archiver. This time is kept per tag, as different tags report to the archiver at different times.<br><br>This value should be in increments of your collection interval, and not less.<br><br>Ideally, this field is used for polled data values. It can be used with unsolicited data, but when you do so, you are dependent on the data source for the value to change. With unsolicited data, since Historian only records the value when it changes, the actual time before the timeout might exceed the compression timeout. |

| Column Name | Data Type | Description |
|---|---|---|
| ArchiveCompressionTimeout | VT_I4 | Indicates the maximum amount of time from the last stored point before another point is stored, if the value does not exceed the archive compression deadband.<br><br>The data archiver treats the incoming sample after the timeout occurs as if it exceeded compression. It then stores the pending sample. |
| TimeZone | VT_BSTR | The type of time zone used:<br><br>• Client<br>• Server<br>• Explicit bias number (number of minutes from GMT) |
| DaylightSavingTime | VT_BOOL | Indicates whether Daylight Saving Time logic should be applied to timestamps. |
| RowCount | VT_I4 | Indicates the maximum number of rows that can be returned. A value of 0 indicates there is no limit to the number of rows returned. |
| InterfaceAbsoluteDeadbanding | VT_BOOL | Indicates whether absolute collector deadband is enabled for this tag. |
| InterfaceAbsoluteDeadband | VT_R8 | Indicates the value for absolute collector deadband. |
| ArchiveAbsoluteDeadbanding | VT_BOOL | Indicates whether absolute archive deadband is enabled for this tag. |
| ArchiveAbsoluteDeadband | VT_R8 | Indicates the value for absolute archive deadband. |
| SpikeLogic | VT_BOOL | Indicates whether Spike Logic is enabled for the tag. |
| SpikeLogicOverride | VT_BOOL | Indicates whether the Spike Logic setting for this tag overrides the collector. |
| StepValue | VT_BOOL | Indicates whether the StepValue property is enabled for the tag. |
| EnumeratedSetName | VT_BSTR | Indicates the enumerated set name associated with a tag. You can get more information about the set via the ihEnumeratedSet table. |

| Column Name | Data Type | Description |
|---|---|---|
| DataStoreName | VT_-BSTR | Indicates the name of the data store the tag belongs to. |
| NumberOfElements | VT_I4 | Indicates whether the tag is an array tag.<br>If set to -1, the tag is an array tag. If set to 0, the tag is not an array tag. Since the size of the array is dynamic, there is no single number of elements that can be returned. |
| CalcType | Enum | Indicates whether the tag is an analytical tag or a normal tag. |
| IsAlias | VT_-BOOL | Indicates whether the tag has an alias or not. |

## ihTags Examples

Tasks that you might want to perform on the ihTags table are outlined in the following examples.

### Example 1: Find All Tags That Belong to a Specific Collector

```
SELECT * FROM ihtags WHERE collectorname=MYCOMPUTER_Simulation ORDER BY tagname
```

### Example 2: Find All Tags With a Specific Poll Rate, a Range of Poll Rates, or Polling Disabled

```
SELECT * FROM ihtags WHERE CollectionInterval=500
OR (CollectionInterval>=1000 AND CollectionInterval<=1200)
OR CollectionInterval=0
```

### Example 3: Retrieve All Tags Collected by Each Collector

```
SELECT collectorname, tagname FROM ihTags ORDER BY collectorname
```

### Example 4: Retrieve All Tags With a Specific Poll Rate

```
SELECT tagname FROM ihtags WHERE collectioninterval=1000
```

### Example 5: Retrieve All Tags With Subsecond Collection

```
SELECT tagname FROM ihtags
WHERE collectioninterval BETWEEN 1 AND 999
```

### Example 6: Retrieve All Tags with Polling Disabled

```
SELECT tagname, collectioninterval FROM ihtags

WHERE collectioninterval=0
```

### Example 7: Count the Number of Tags and Group by Collector Name

```
SELECT collectorname, COUNT(*) FROM ihTags GROUP BY collectorname
```

### Example 8: Count the Number of Tags and Group by Collector Type

```
SELECT ihCollectors.collectortype, COUNT(*)

FROM ihTags INNER JOIN ihCollectors

WHERE ihTags.collectorname=ihCollectors.collectorname

GROUP BY ihcollectors.collectortype
```

### Example 9: Retrieve Tags Associated With a Specific Enumerated Set

```
SELECT * FROM ihtags

WHERE EnumeratedSetName='ExampleSet’
```

## ihArchives Table

Historian archives are stored as data files, each of which contains data gathered during a specific period of time.

The `ihArchives` table contains Historian archive configuration information and performance statistics for each archive. Each row in this table represents one archive. The following table describes the columns of the `ihArchives` table.

**Table 394. ihArchives Table**

| Column Name | Data Type | Description |
|---|---|---|
| `Archive-Name` | `VT_BSTR` | Name of the archive for the current server if the authenticated user is a member of Historian Administrators group. |
| `ArchiveS-tatus` | `VT_BSTR` | The status of the specified archive:<br>• `Undefined`<br>• `Empty`<br>• `NotEmpty` |

**Table 394. ihArchives Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| FileName | VT_BSTR | The file name for the specified archive. The file name must be specified in the context of the Historian server drives and directories. |
| IsCurrent | VT_BOOL | Indicates whether the specified archive is the newest archive that new data currently flows into. |
| IsReadOnly | VT_BOOL | Indicates whether the read-only status is set for the specified archive. |
| FileSize-Current-Disk | VT_I4 | The actual space on the hard disk (in MB) for the specified archive. |
| FileSize-Current | VT_I4 | The size of the archive file that is currently being used (in MB) for the specified archive. |
| FileSize-Target | VT_I4 | The target size of the specified archive file (in MB). |
| StartTime | VT_DB-TimeS-tamp | The start time of the specified archive. This represents the earliest timestamp (including date and time) for any tag contained in the archive. |
| EndTime | VT_DB-TimeS-tamp | The end time of the specified archive. This represents the latest timestamp (including date and time) for any tag contained in the archive. |
| LastBackup | VT_DB-TimeS-tamp | The date and time the most recent online backup was performed on this archive. |
| LastBacku-pUser | VT_BSTR | The name of the user who performed the most recent online backup. |
| LastModi-fied | VT_DB-TimeS-tamp | The date and time that the archive was last modified. The time structure includes milliseconds. |
| LastModi-fiedUser | VT_BSTR | The username of the Windows user who last modified the archive. |
| TimeZone | VT_BSTR | The type of time zone used: |

**Table 394. ihArchives Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| | | • Client<br>• Server<br>• Explicit bias number (number of minutes from GMT). |
| `Daylight-SavingTime` | `VT_BOOL` | Indicates whether Daylight Saving Time logic should be applied to timestamps. |
| `RowCount` | `VT_I4` | Indicates the maximum number of rows that can be returned. A value of `0` indicates there is no limit to the number of rows returned. |
| `DataStore-Name` | `VT_BSTR` | Indicates the name of the data store the tag belongs to. |

### ihArchives Examples

A task that you might want to perform on the `ihArchives` table is retrieving and recording the state of the archives and archive sizes when an event happens. Recording conditions when an event happens is useful in troubleshooting.

Sample SQL statements for the `ihArchives` table are outlined in the following examples.

### Example 1: Retrieve the Archive List Sorted by StartTime

```
SELECT archivename, starttime, endtime

FROM iharchives ORDER BY starttime
```

### Example 2: Retrieve All Properties of the Current Archive

```
SELECT * FROM iharchives WHERE iscurrent=true
```

## ihCollectors Table

The `ihCollectors` table contains the configuration and status information for each collector connected to the Historian server. Each row in this table represents a collector that is connected to the archiver. The following table describes the columns of the `ihCollectors` table.

**Table 395. ihCollectors Table**

| Column Name | Data Type | Description |
|---|---|---|
| `CollectorName` | `VT_BSTR` | The name of the collector. The collector name is unique in a specific Historian server. |
| `CollectorDescription` | `VT_BSTR` | The user description for the collector. |
| `Comment` | `VT_BSTR` | The user comment associated with the collector. |
| `ComputerName` | `VT_BSTR` | The name of the Windows computer on which the collector is running. |
| `Status` | `VT_BSTR` | The status of the specified collector:<br>• `Unknown`<br>• `Starting`<br>• `Running`<br>• `Stopping`<br>• `Stopped` |
| `CollectorType` | `VT_BSTR` | The type of collector responsible for collecting data for the tag:<br>• `Undefined`<br>• `iFIX`<br>• `Simulation`<br>• `OPC`<br>• `OPC AE`<br>• `File`<br>• `iFIXLabData`<br>• `ManualEntry`<br>• `Simulation`<br>• `Calculation`<br>• `ServerToServer`<br>• `Other` |

**Table 395. ihCollectors Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| `MaximumDiskFreeBufferSize` | `VT_I4` | The maximum size (in MB) of the disk buffer for outgoing data. |
| `MaximumMemoryBufferSize` | `VT_I4` | The maximum size of the memory buffer (in MB) for outgoing data.<br><br>The memory buffer stores data during short-term or momentary interruptions of the server connection. The disk buffer handles long-duration outages. |
| `ShouldAdjustTime` | `VT_BOOL` | If the data source supplies the timestamps, this value is `False`. If the collector supplies the timestamps, this value is `True`.<br><br>**Note:**<br>This column does not change collector times to match the server time. It indicates whether an increment of time is added or subtracted to compensate for the relative difference between the server and collector clocks, independent of time zone differences. |
| `ShouldQueueWrites` | `VT_BOOL` | Indicates whether queue writes are allowed. |
| `CanBrowseSource` | `VT_BOOL` | If `True`, this column indicates that the collector can browse its source for tags. |
| `CanSourceTimestamp` | `VT_BOOL` | Indicates whether the data source can provide timestamps along with the data. |
| `StatusOutputAddress` | `VT_BSTR` | An address or tagname in the data source to output current collector status. |
| `RateOutputAddress` | `VT_BSTR` | An address or tagname in the data source into which the collector writes the current value of the events per minute output. |
| `HeartbeatOutputAddress` | `VT_BSTR` | The address in the source database into which the collector writes the heartbeat signal output. |
| `CollectorGeneral1` | `VT_BSTR` | The general (or spare) configuration fields for the collector. The `CollectorGeneral1` column is not user-defined, and is different for each collector. |

**Table 395. ihCollectors Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| CollectorGeneral2 | VT_BSTR | The general (or spare) configuration fields for the collector. The CollectorGeneral2 column is not user-defined, and is different for each collector. |
| CollectorGeneral3 | VT_BSTR | The general (or spare) configuration fields for the collector. The CollectorGeneral3 column is not user-defined, and is different for each collector. |
| CollectorGeneral4 | VT_BSTR | The general (or spare) configuration fields for the collector. The CollectorGeneral4 column is not user-defined, and is different for each collector. |
| CollectorGeneral5 | VT_BSTR | The general (or spare) configuration fields for the collector. The CollectorGeneral5 column is not user-defined, and is different for each collector. |
| LastModified | VT_DBTimeStamp | The date and time that the collector configuration was last modified. The time structure includes milliseconds. |
| LastModifiedUser | VT_BSTR | The username of the Windows user who last modified the collector configuration. |
| SourceTimeInLocalTime | VT_BOOL | For data source timestamps only. Indicates whether the timestamps use local time. If the value is False, UTC time is used. |
| CollectionDelay | VT_I4 | The length of time, in seconds, that the collector should delay collection at startup (to allow the data source time to initialize). |
| DefaultTagPrefix | VT_BSTR | The prefix that is automatically applied to all tagnames added by the specified collector. |
| DefaultCollectionInterval | VT_I4 | The collection interval, in milliseconds, for tags added by the collector. |
| DefaultCollectionType | VT_BSTR | Type of collection used to acquire data for tags added by the collector:<br>• **Unsolicited:** The collector accepts data from the source whenever the source presents the data.<br>• **Polled:** The collector acquires data from a source on a periodic schedule determined by the collector. |

**Table 395. ihCollectors Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| | | **Note:**<br>Not all collectors support unsolicited type collection. |
| `DefaultTimeStampType` | `VT_BSTR` | Type of timestamp applied to data samples at collection time for tags added by the collector:<br>• **Source:** The source delivers the timestamp along with the data sample.<br>• **Collector:** The collector delivers the timestamp along with the collected data. |
| `DefaultCollectorCompression` | `VT_BOOL` | Indicates whether default collector compression is enabled for tags added by the collector. |
| `DefaultCollectorCompressionDeadband` | `VT_R4` | The default collector compression deadband for tags added by the collector. |
| `DefaultCollectorCompressionTimeout` | `VT_I4` | The default collector compression timeout value. |
| `DisableOnTheFlyChanges` | `VT_BOOL` | Indicates whether a user can make on-the-fly changes to this tag. When enabled (`True`) you can make changes to this tag without having to restart the collector.<br>When disabled (`False`), any changes you make to this tag do not affect collection until you restart the collector. |
| `DefaultSpikeLogic` | `VT_BOOL` | Indicates whether Spike Logic is enabled. |
| `DefaultSpikeMultiplier` | `VT_R4` | The default Spike Logic multiplier. |
| `DefaultSpikeInterval` | `VT_I4` | The default Spike Logic interval. |

**Table 395. ihCollectors Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| `RedundancyEnabled` | `VT_BOOL` | Indicates whether collector redundancy is enabled. |
| `PrincipalCollector` | `VT_BSTR` | Indicates the primary collector. |
| `IsActiveRedundantCollector` | `VT_BOOL` | Indicates whether the current collector is active. |
| `FailoverOnCollectorStatus` | `VT_BOOL` | Indicates whether the collector is set to fail over on an unknown collector status. |
| `FailoverOnBadQuality` | `VT_BOOL` | Indicates whether the collector is set to fail over on bad data quality received from the watchdog tag. |
| `FailoverOnValue` | `VT_BOOL` | Indicates whether the collector is set to fail over on a change in value. |
| `FailoverValueChangeType` | `VT_I4` | The value for the `FailoverOnValue` option. |
| `WatchdogValueMaxUnchangedPeriod` | `VT_I4` | The maximum period for an unchanged value. |
| `WatchdogTagName` | `VT_BSTR` | The watchdog tag name. |
| `TimeZone` | `VT_BSTR` | The type of time zone used:<br>• Client<br>• Server<br>• Explicit bias number (number of minutes from GMT) |
| `DaylightSavingTime` | `VT_BOOL` | Indicates whether Daylight Saving Time logic should be applied to timestamps. |
| `RowCount` | `VT_I4` | The maximum number of rows that can be returned. A value of `0` indicates there is no limit to the number of rows returned. |

### ihCollectors Examples

One task that you might want to perform on the `ihCollectors` table could be retrieving and recording the state of the collectors when an event happens. Recording conditions when an event happens is useful in troubleshooting.

Sample SQL statements for the `ihCollectors` table are outlined in the following examples.

### Example 1: Retrieve All Collectors With Status Information

```
SELECT collectorname, collectordescription AS desc, status

FROM ihcollectors
```

### Example 2: Retrieve All Collectors Not Running

```
SELECT collectorname, collectordescription AS desc, status

FROM ihcollectors WHERE status!=running
```

## ihMessages Table

The `ihMessages` table contains Historian messages such as alerts, informational topics, and connection information contained in the audit log. Each row in this table represents a message. The following table describes the columns of the `ihMessages` table.

**Table 396. ihMessages Table**

| Column Name | Data Type | Description |
|---|---|---|
| `TimeStamp` | `VT_DBTimeStamp` | The date and time that the message was created. |
| `TimeStampSeconds` | `VT_DBTimeStamp` | The date and time that the message was created. |
| `Microseconds` | `VT_I4` | The microsecond portion of the date and time for the message. |

**Table 396. ihMessages Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| `Topic` | `VT_-BSTR` | The topic name of the message:<br>• `AlertTopics`<br>• `AllTopics`<br>• `ConfigurationAudit`<br>• `Connections`<br>• `General`<br>• `MessageTopicMax`<br>• `MessageTopics`<br>• `Performance`<br>• `ServiceControl`<br>• `Security`<br>• `Undefined` |
| `User-name` | `VT_-BSTR` | Name of the Windows user who generated the message, or who the message is associated with. |
| `Mes-sage-Number` | `VT_I4` | Message number for the message. A message number is a unique identifier associated with the message template. |
| `Mes-sage-String` | `VT_-BSTR` | Translated text of the message, including any substitutions. Messages generally include translated fixed text and variable substitutions such as timestamps, usernames, and tagnames. |
| `Time-Zone` | `VT_-BSTR` | The type of time zone used:<br>• Client<br>• Server<br>• Explicit bias number (number of minutes from GMT) |
| `Day-light-Saving-Time` | `VT_-BOOL` | Indicates whether Daylight Saving Time logic should be applied to timestamps. |

**Table 396. ihMessages Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| `Row Count` | `VT_I4` | Indicates the maximum number of rows that can be returned. A value of `0` indicates there is no limit to the number of rows returned. |

## ihMessages Examples

One task that you might want to perform on the `ihMessages` table is retrieving a history of alerts and messages, with timestamps and user information. For instance, you might want to query the alerts for a day, or all messages associated with a particular username.

Sample SQL statements for the `ihMessages` table are outlined in the following examples.

### Example 1: Retrieve All Messages and Alerts for Today

```
SELECT * FROM ihmessages WHERE timestamp>=today
```

### Example 2: Retrieve All Alert Messages for a Specific User and Time

```
SELECT * FROM ihmessages
WHERE timestamp>'12-sep-2001 02:00:00'
AND topic=AlertTopics
AND username='DataArchiver' ORDER BY timestamp
```

### Example 3: Retrieve All Messages in Your Archive

```
SELECT * FROM ihMessages WHERE timestamp <= Now
```

### Example 4: Retrieve All Messages for a Specific User

```
SELECT * FROM ihMessages WHERE username=operator1
AND timestamp<=Now
```

### Example 5: Count All Messages by a Specific User

```
SELECT username, COUNT(*) FROM ihMessages
WHERE timestamp <=Now GROUP BY username
```

## ihRawData Table

The `ihRawData` table contains any collected data for each tag contained in the Historian server. It contains not just raw data, but also calculated data and interpolated data. This table is the one typically used for reporting.

There is one row in the `ihRawData` table for each combination of tagname and timestamp. For instance, you can have two rows for the same tag, each with different timestamps. You can retrieve data for more than one tag name in a simple query.

The following table describes the columns of the `ihRawData` table.

**Table 397. ihRawData Table**

| Column Name | Data Type | Description |
|---|---|---|
| `Tagname` | `VT_BSTR` | Tagname property of the tag.<br><br>**Note:**<br>There is no length limit for Historian tag names in the Data Archiver. However, different client applications may have their own limits. |
| `TimeStamp` | `VT_DBTimeStamp` | The date and time for the data sample. |
| `TimeStampSeconds` | `VT_DBTimeStamp` | The date and time for the data sample. |
| `MicrosecondSeconds` | `VT_DBTimeStamp` | The microsecond interval for the data sample. |
| `Value` | `VT_VARIANT` | The value of the data. |
| `Quality` | `VT_VARIANT` | For non-raw sampled data, this column displays the percentage of good quality samples in the interval. For instance, a value of `100` means all samples in the interval are good. |

**Table 397. ihRawData Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| | | For raw sampled data, data values are:<br>• `Good`<br>• `Bad`<br>• `Uncertain`<br>• `Not Available`<br>• `Really Unknown`<br>This column also includes the subquality of the data value, if it exists:<br>• `NonSpecific`<br>• `ConfigError`<br>• `NotConnected`<br>• `DeviceFail`<br>• `SensorFail`<br>• `LastKnownValue`<br>• `CommFailure`<br>• `OutOfService`<br>• `ScaledOutOfRange`<br>• `OffLine`<br>• `NoValue`<br>• `Really Unknown` |
| `OPC-Quality-Valid` | `VT_BSTR` | Indicates whether the `OPCQuality` column contains valid real OPC quality. A value of `0` indicates that you should ignore the `OPCQuality` field, and a value of `1` indicates that the `OPCQuality` column contains valid real OPC quality. |
| `OPC-Quality` | `VT_I4` | Indicates the OPC quality as delivered by the OPC server to the Historian OPC collector. The exact meaning of the bits depends on the OPC specification and the OPC server documentation. Typically, a value of `0` represents bad quality, and a value of `192` represents good quality. |

**Table 397. ihRawData Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| `SamplingMode` | `VT_BSTR` | The mode used to sample data from the archive:<br>• `CurrentValue`: Retrieves the current value. Time frame criteria are ignored.<br>• `Interpolated`: Retrieves evenly spaced interpolated values based on interval or `NumberOfSamples` and the time frame criteria.<br>• `RawByTime`: Retrieves raw archive values based on time frame criteria.<br>• `RawByNumber`: Retrieves raw archive values based on the `StartTime`, `NumberOfSamples`, and `Direction` criteria.<br>**Note:** `EndTime` criteria are ignored for this sampling mode.<br>• `RawByFilterToggle`: Returns filtered time ranges. The values returned are `0` and `1`. If the value is `1`, then the condition is true and `0` means false. This sampling mode is used with the time range and filter tag conditions. The result starts with a starting timestamp and ends with an ending timestamp.<br>• `Calculated`: Retrieves evenly spaced calculated values based on `NumberOfSamples`, interval, the time frame criteria, and the `CalculationMode` criteria.<br>• `Lab`: Returns actual collected values without interpolation.<br>• `Trend`: Returns raw minimum and raw maximum values for each specified interval. Use this sampling mode to maximize performance when retrieving data points for plotting. If the sampling period does not evenly divide by the interval length, Historian ignores any leftover values at the end, rather than putting them into a smaller interval.<br>• `Trend2`: Returns raw minimum and raw maximum values for each specified interval. Use this sampling mode to maximize performance when retrieving data points for plotting. Also, if the sampling period does not evenly divide by the interval length, Historian creates as many intervals of the interval length as will fit into the sampling period, and then creates a remainder interval from whatever time is left. This sampling mode is more suitable than the `Trend` mode for analysis of minimums and maximums and for plotting programs that can handle unevenly spaced data. |

**Table 397. ihRawData Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| | | • `InterpolatedtoRaw`: Provides raw data in place of interpolated data when the number of samples is less than the available samples.<br>• `TrendtoRaw`: This sampling mode almost always produces the same results as the `Trend` mode. The exception is that when the number of samples requested is greater than the number of raw data points, this mode returns all available raw data points with no further processing. `TrendtoRaw` is therefore used instead of `Trend` when the number of actual data samples is less than the requested number of samples.<br>• `TrendtoRaw2`: This sampling mode almost always produces the same results as the `Trend2` mode. The exception is that when the number of samples requested is greater than the number of raw data points, this mode returns all available raw data points with no further processing. `TrendtoRaw2` is therefore used instead of `Trend2` when the number of actual data samples is less than the requested number of samples.<br>• `LabtoRaw`: Provides raw data for the selected calculated data when the number of samples is less than the available samples. |
| `Direction` | `VT_BSTR` | The direction (forward or backward from the start time) of data sampling from the archive. |
| `NumberOfSamples` | `VT_I4` | Number of samples from the archive to retrieve.<br><br>Samples will be evenly spaced within the time range defined by the start and end times for most sampling modes. For the `RawByNumber` mode, this column determines the maximum number of values to retrieve. For the `RawByTime` mode, this value is ignored.<br><br>**Note:**<br>The `NumberofSamples` and `IntervalMilliseconds` columns are mutually exclusive. If `NumberofSamples` is used, `IntervalMilliseconds` is not used. |
| `IntervalMil-` | `VT_I4` | For non-raw sampled data, this column represents a positive integer for the time interval (in milliseconds) between returned samples. |

**Table 397. ihRawData Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| `liseconds` | | **Note:** The `NumberofSamples` and `IntervalMilliseconds` columns are mutually exclusive. If `NumberofSamples` is used, `IntervalMilliseconds` is not used. |
| `CalculationMode` | `VT_BSTR` | This column applies only if the `SamplingMode` is set to `Calculated`. It represents the type of calculation to perform on archive data:<br>• `Average`<br>• `Count`<br>• `Maximum`<br>• `MaximumTime`<br>• `Minimum`<br>• `MinimumTime`<br>• `StandardDeviation`<br>• `Total`<br>• `RawAverage`<br>• `RawStandardDeviation`<br>• `RawTotal`<br>• `TimeGood`<br>• `FirstRawValue`<br>• `FirstRawTime`<br>• `LastRawValue`<br>• `LastRawTime`<br>• `TagStats` |
| `FilterTag` | `VT_BSTR` | Tagname used to define the filter, if specified. Only a single tag can be specified, and wildcards are not supported. |
| `FilterMode` | `VT_BSTR` | The type of time filter:<br>• `ExactTime`: Retrieves data for the exact times that the filter condition is `True`.<br>• `BeforeTime`: Retrieves data from the time of the last `False` filter condition to the time of the next `True` condition. |

**Table 397. ihRawData Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| | | • `AfterTime`: Retrieves data from the time of the last `True` filter condition to the next `False` condition.<br>• `BeforeAndAfterTime`: Retrieves data from the time of the last `False` filter condition to the next `False` condition.<br><br>This mode defines how time periods before and after transitions in the filter condition should be handled.<br><br>For example, `AfterTime` indicates that the filter condition should be `True` starting at the timestamp of the archive value that triggered the `True` condition and leading up to the timestamp of the archive value that triggered the `False` condition. |
| `FilterComparisonMode` | `VT_BSTR` | The type of comparison to be made on the filter comparison value:<br><br>• `Equal`: Filter condition is `True` when the `FilterTag` value is equal to the comparison value.<br>• `EqualFirst`: Filter condition is `True` when the `FilterTag` value is equal to the first comparison value.<br>• `EqualLast`: Filter condition is `True` when the `FilterTag` value is equal to the last comparison value.<br>• `NotEqual`: Filter condition is `True` when the `FilterTag` value is not equal to the comparison value.<br>• `LessThan`: Filter condition is `True` when the `FilterTag` value is less than the comparison value.<br>• `GreaterThan`: Filter condition is `True` when the `FilterTag` value is greater than the comparison value.<br>• `LessThanEqual`: Filter condition is `True` when the `FilterTag` value is less than or equal to the comparison value.<br>• `GreaterThanEqual`: Filter condition is `True` when the `FilterTag` value is greater than or equal to the comparison value.<br>• `AllBitsSet`: Filter condition is `True` when the binary `FilterTag` value is equal to all the bits in the condition. It is represented as `^` to be used in `FilterExpression`. |

**Table 397. ihRawData Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| | | • `AnyBitSet`: Filter condition is `True` when the binary `FilterTag` value is equal to any of the bits in the condition. It is represented as `~` to be used in `FilterExpression`.<br>• `AnyBitNotSet`: Filter condition is `True` when the binary `FilterTag` value is not equal to any one of the bits in the condition. It is represented as `!~` to be used in `FilterExpression`.<br>• `AllBitsNotSet`: Filter condition is `True` when the binary `FilterTag` value is not equal to all the bits in the condition. It is represented as `!^` to be used in `FilterExpression`.<br><br>This column defines how archive values for the `FilterTag` value should be compared to the `FilterValue` value to establish the state of the filter condition. If `FilterTag` and `FilterComparisonValue` values are specified, time periods are filtered from the results where the filter condition is `False`. |
| `FilterValue` | `VT_BSTR` | The value with which to compare the `FilterTag` value to determine appropriate filter times. |
| `FilterExpression` | `VT_BSTR` | An expression which includes one or more filter conditions. The type of conditions used are:<br><br>• `AND`<br>• `OR`<br>• Combination of `AND` and `OR`<br><br>`FilterExpression` can be used instead of the `FilterTag`, `FilterComparisonMode` and `FilterValue` parameters. While using `FilterExpression`, the expression is passed within single quotes. For complex expressions, write the conditions within parentheses. There is no maximum length for the `FilterExpression` value, but if called using OLE DB or Excel, those tools may have their own limitations. |
| `TimeZone` | `VT_BSTR` | The type of time zone used: |

**Table 397. ihRawData Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| | | • Client<br>• Server<br>• Explicit bias number (number of minutes from GMT) |
| `DaylightSavingTime` | `VT_BOOL` | Indicates whether Daylight Saving Time logic should be applied to timestamps. |
| `RowCount` | `VT_I4` | Indicates the maximum number of rows that can be returned. A value of `0` indicates that there is no limit to the number of rows returned. |

The `ihRawData` table can generate a large number of rows if not used with caution. You can easily generate queries which take a very long time to complete and put stress on the archiver and generate network traffic.

## ihRawData Examples

Tasks that you might want to perform on the `ihRawData` table are outlined in the following examples.

### Example 1: Retrieve All Samples With a Value Outside the Query Supplied Values

```
SELECT * FROM ihRawData WHERE value<140000 OR value>150000
```

### Example 2: Retrieve All Bad Samples (Raw Data)

```
SELECT * FROM ihRawData WHERE quality NOT LIKE good*

AND samplingmode=RawbyTime
```

### Example 3: Count Bad Samples (Raw Data)

```
SELECT COUNT(*) FROM ihRawData WHERE quality NOT LIKE good*

AND samplingmode=RawbyTime
```

## Example 4: Retrieve All Bad Samples Over the Last Day (Interpolated Data)

```
SELECT timestamp, tagname, value, quality FROM ihRawData

WHERE samplingmode=rawbytime

AND Quality NOT LIKE good*

AND timestamp>=Now-24H
```

## Example 5: Use an Explicit Time Zone

```
SELECT * FROM ihRawData WHERE timezone=300
```

## Example 6: Perform a Simple Sequence of Events

```
SELECT timestamp, tagname, value, quality FROM ihrawdata

WHERE samplingmode=rawbytime ORDER BY timestamp
```

## Example 7: Report the Busiest Tags

```
SELECT tagname, value FROM ihRawData

WHERE samplingmode=calculated

AND calculationmode=count

AND numberofsamples=1

AND timestamp>='07/30/2002 10:00:00'

AND timestamp<='07/30/2002 11:00:00' order by value descending
```

## Example 8: Retrieve All Bad Samples Over the Last Day

```
SELECT timestamp, tagname, value, quality FROM ihRawData

WHERE samplingmode=rawbytime

AND Quality NOT LIKE good*

AND timestamp>=Now-24H
```

## Example 9: Retrieve All Bad Samples, Ignore End of Collection Markers

```
SELECT timestamp, tagname, value, quality FROM ihRawData

WHERE samplingmode=rawbytime

AND Quality NOT LIKE good*

AND quality NOT LIKE 'bad offline' AND timestamp>=Now-24H
```

### Example 10: Count Bad Samples, Ignore End of Collection Markers

```
SELECT COUNT(*) FROM ihRawData WHERE samplingmode=rawbytime

AND Quality NOT LIKE good* and Quality NOT LIKE 'bad offline'

AND timestamp>=Now-24H
```

### Example 11: Obtain All Raw Samples With Comments From Yesterday

```
SELECT ihRawData.Tagname, ihRawData.TimeStamp, ihRawData.Value

FROM ihRawData

INNER JOIN ihComments ON ihComments.Tagname = ihRawData.Tagname

AND ihComments.Timestamp = ihRawData.Timestamp

AND ihComments.Comment = "The comment" WHERE samplingmode=rawbytime

AND ihComments.Timestamp > Yesterday

AND ihComments.Timestamp < Today
```

### Example 12: Determine the Number of Milliseconds Per Interval With Good Data

```
SELECT timestamp, tagname, value as TimeGood, quality, intervalmilliseconds FROM ihRawData

WHERE tagname=Denali.Simulation00001

AND samplingmode=calculated

AND calculationmode=timegood

AND intervalmilliseconds=10s

AND timestamp>='1/20/2003 13:18:00'

AND timestamp<='1/20/2003 13:20:00'
```

### Example 13: Retrieve Raw Minimum and Maximum Values Per Interval

In this example, you use the data retrieved from the query (with the `Trend` sampling mode) to plot points.

```
SELECT timestamp, tagname, value, quality

FROM ihRawData

WHERE tagname=dFloatTag5

AND samplingmode=trend

AND intervalmilliseconds=24h

AND timestamp>='1/01/2003 07:00:00'

AND timestamp<='1/10/2003 12:00:00'
```

### Example 14: Retrieve Data with Native Values and Tags Associated With Enumerated Sets

If the `enumnativevalue` query modifier is not set, the data is retrieved with string values by default. If it is set, the raw values are retrieved. These values are then retrieved by default for the current session and will only change when you open a new session.

```
SELECT * from ihrawdata

WHERE samplingmode='rawbytime' and tagname=mytag AND criteriastring='#enumnativevalue'


SELECT timestamp,value,quality from ihrawdata WHERE tagname = MyTag AND samplingmode=Interpolated and numberofsamples=6

 and criteriastring='#enumnativevalue'


SET criteriastring='#enumnativevalue'

SELECT * from ihrawdata

WHERE samplingmode='rawbytime' and tagname=mytag
```

### Example 15: Retrieve Average Values for Enumerated Sets

```
SET criteriastring='#enumnativevalue'


SELECT * from ihrawdata

WHERE tagname LIKE Call AND samplingmode=calculated AND calculationmode=average
```

## ihHabAlarms Table

The `ihHabAlarms` table contains alarm data collected from Habitat by the HAB collector. This data is stored in the Historian archive files.

| Column Name | Data Type | Description |
|---|---|---|
| `tagname` | `VT_BSTR` | The tagname property of the tag. |
| `time-stamp` | `VT_DB-TimeS-tamp` | The date and time for the data sample (based on the timestamp of collector) |
| `time-stampsec-onds` | `VT_I4` | The date and time for the data sample. |

| Column Name | Data Type | Description |
|---|---|---|
| microseconds | VT_I4 | The microsecond interval for the data sample. |
| sampling-mode | VT_BSTR | The mode used to retrieve data from the archive. Only the RawByTime sampling mode is used for alarms. It retrieves raw archive values for a time period. |
| quality | VT_BSTR | The quality of the tag data. For raw sampled data, the valid data values is Good. |
| text | VT_BSTR | The alarms message |
| location | VT_BSTR | The substation or location as defined in the Habitat database |
| priority | VT_BSTR | The alarm priority as defined in the Habitat database |
| category | VT_BSTR | The alarm category as defined in the Habitat database |
| exception | VT_BSTR | The alarm exception as defined in the Habitat database |
| area | VT_BSTR | The alarm area as defined in the Habitat database |
| fieldtime | VT_DBTimeStamp | The field time of the alarm message (if available) |
| fmillisec | VT_I4 | The milliseconds part of the field time |
| fnanosec | VT_I4 | The nanoseconds part of the field time |
| time | VT_DBTimeStamp | The time of the alarm generated in Habitat (mapped to TIME_CIRCLG) |
| timefmt | VT_I4 | |
| tnanosec | VT_I4 | The nanoseconds part of the alarms time |
| rowcount | VT_I4 | The maximum number of rows that can be returned. A value of 0 indicates that there is no limit to the number of rows returned. |

## ihComments Table

The `ihComments` table contains the annotations associated with the collected data. There is a separate row of data in the `ihComments` table for each comment associated with a tag. For instance, you can have five rows that contain the same tag and timestamp, but each contain a different comment value.

It is possible to have different data types of annotations. Comments are most often strings, but can be binary numbers or BLOBs. Only string comments are returned in the `ihComments` table.

The following table describes the columns of the `ihComments` table.

**Table 398. ihComments Table**

| Column Name | Data Type | Description |
|---|---|---|
| `Tagname` | `VT_BSTR` | Tagname property of the tag.<br><br>**Note:**<br>There is no length limit for Historian tag names in the Data Archiver. However, different client applications may have their own limits. |
| `TimeStamp` | `VT_DBTimeStamp` | The date and time that the data was generated. |
| `TimeStampSeconds` | `VT_DBTimeStamp` | The date and time that the data was generated. |
| `Microseconds` | `VT_I4` | The microsecond portion of the date and time. |
| `StoredOnTimeStamp` | `VT_DBTimeStamp` | The date and time that the comment was generated. |
| `StoredOnTimeStamp` | `VT_DBTimeStamp` | The time that the comment was added to the archive. |
| `SuppliedUsername` | `VT_BSTR` | The username of the currently logged-in Windows user at the |

**Table 398. ihComments Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| | | time that the comment was entered. |
| `Username` | `VT_BSTR` | Username provided along with the comment. |
| `Comment` | `VT_BSTR` | The actual comment. |
| `DataTypeHint` | `VT_BSTR` | Name of the data type for the comment:<br>• `String`<br>• `Read-only`<br>• `Optional` |
| `SamplingMode` | `VT_BSTR` | The mode used to sample data from the archive:<br>• `CurrentValue`: Retrieves the current value. Time frame criteria are ignored.<br>• `Interpolated`: Retrieves evenly spaced interpolated values based on interval or `NumberOfSamples` and time frame criteria.<br>• `RawByTime`: Retrieves raw archive values based on time frame criteria.<br>• `RawByNumber`: Retrieves raw archive values based on the `StartTime`, `NumberOf-Samples`, and `Direction` criteria. |

**Table 398. ihComments Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| | | **Note:** `EndTime` criteria are ignored for this sampling mode.<br>• `RawByFilterToggle`: Returns filtered time ranges. The values returned are `0` and `1`. If the value is `1`, then the condition is true and `0` means false. This sampling mode is used with the time range and `FilterTag` conditions. Results have starting and ending timestamps.<br>• `Calculated`: Retrieves evenly spaced calculated values based on `NumberOfSamples`, interval, time frame, and `CalculationMode` criteria.<br>• `Lab`: Returns actual collected values without interpolation.<br>• `Trend`: Returns raw minimums and maximums for each specified interval. Use this mode to maximize performance when retrieving data points for plotting. If the sampling |

**Table 398. ihComments Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| | | period does not evenly divide by the interval length, Historian ignores any left-over values at the end instead of putting them into a smaller interval.<br>• `Trend2`: Returns raw minimum and maximum values for each specified interval. Use this mode to maximize performance when retrieving data points for plotting. If the sampling period does not evenly divide by the interval length, Historian creates as many intervals of the interval length as will fit into the sampling period, and then creates a remainder interval from whatever time is left. This sampling mode is more suitable than the `Trend` mode for analysis of minimums and maximums and for plotting programs that can handle unevenly spaced data.<br>• `InterpolatedtoRaw`: Provides raw data in place of interpolated data when the number of samples requested is less than the |

**Table 398. ihComments Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| | | number of available samples.<br>• `TrendtoRaw`: This mode almost always produces the same results as the `Trend` mode. The exception is that when a greater number of samples are requested than the number of raw data points, this mode returns all available raw data points with no further processing. This mode is therefore used instead of `Trend` when the number of actual data samples is less than the requested number of samples.<br>• `TrendtoRaw2`: This sampling mode almost always produces the same results as the `Trend2` mode. The exception is that when a greater number of samples are requested than the number of raw data points, this mode returns all available raw data points with no further processing. This mode is therefore used instead of `Trend2` when the number of actual data samples is |

**Table 398. ihComments Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| | | less than the requested number of samples.<br>• `LabtoRaw`: Provides raw data for the selected calculated data when the number of samples is less than the number of available samples. |
| `Direction` | `VT_BSTR` | The direction (forward or backward from the start time) of data sampling from the archive. |
| `NumberOfSamples` | `VT_I4` | Number of samples from the archive to retrieve.<br><br>Samples will be evenly spaced within the time range defined by start and end times for most sampling modes. For the `RawByNumber` mode, this column determines the maximum number of values to retrieve. For the `RawByTime` mode, this value is ignored.<br><br>**Note:**<br>The `NumberofSamples` and `IntervalMilliseconds` columns are mutually exclusive. If `NumberofSamples` is used, `IntervalMilliseconds` is not used. |
| `IntervalMilliseconds` | `VT_I4` | For non-raw sampled data, this column represents a positive integer for the time interval (in |

**Table 398. ihComments Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| | | milliseconds) between returned samples.<br><br>**Note:**<br>The `NumberofSamples` and `IntervalMilliseconds` columns are mutually exclusive. If `NumberofSamples` is used, `IntervalMilliseconds` is not used. |
| `CalculationMode` | `VT_BSTR` | The calculation mode, if used. |
| `FilterTag` | `VT_BSTR` | Tagname used to define the filter, if specified. Only a single tag can be specified, and wildcards are not supported. |
| `FilterMode` | `VT_BSTR` | The type of time filter:<br><br>• `ExactTime`: Retrieves data for the exact times that the filter condition is `True`.<br>• `BeforeTime`: Retrieves data from the time of the last `False` filter condition to the time of the next `True` condition.<br>• `AfterTime`: Retrieves data from the time of the last `True` filter condition to the time of the next `False` condition.<br>• `BeforeAndAfterTime`: Retrieves data from the time |

**Table 398. ihComments Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| | | of the last `False` filter condition to the time of the next `False` condition.<br><br>The `FilterMode` defines how time periods before and after transitions in the filter condition should be handled.<br><br>For example, `AfterTime` indicates that the filter condition should be `True` starting at the timestamp of the archive value that triggered the `True` condition and ending at the timestamp of the archive value that triggered the `False` condition. |
| `FilterComparisonMode` | `VT_BSTR` | The type of comparison to be made on the filter comparison value:<br><br>• `Equal`: Filter condition is `True` when the `FilterTag` value is equal to the comparison value.<br>• `EqualFirst`: Filter condition is `True` when the `FilterTag` value is equal to the first comparison value.<br>• `EqualLast`: Filter condition is `True` when the `FilterTag` value is equal to the last comparison value.<br>• `NotEqual`: Filter condition is `True` when the `FilterTag` |

**Table 398. ihComments Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| | | value is not equal to the comparison value.<br>• `LessThan`: Filter condition is `True` when the `FilterTag` value is less than the comparison value.<br>• `GreaterThan`: Filter condition is `True` when the `FilterTag` value is greater than the comparison value.<br>• `LessThanEqual`: Filter condition is `True` when the `FilterTag` value is less than or equal to the comparison value.<br>• `GreaterThanEqual`: Filter condition is `True` when the `FilterTag` value is greater than or equal to the comparison value.<br>• `AllBitsSet`: Filter condition is `True` when the binary `FilterTag` value is equal to all the bits in the condition. It is represented as `^` to be used in `FilterExpression`.<br>• `AnyBitSet`: Filter condition is `True` when the binary `FilterTag` value is equal to any of the bits in the condition. It is represented as `~` to be used in `FilterExpression`. |

**Table 398. ihComments Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| | | • `AnyBitNotSet`: Filter condition is `True` when the binary `FilterTag` value is not equal to any one of the bits in the condition. It is represented as `!~` to be used in `FilterExpression`.<br>• `AllBitsNotSet`: Filter condition is `True` when the binary `FilterTag` value is not equal to all the bits in the condition. It is represented as `!^` to be used in `FilterExpression`.<br><br>This column defines how archive `FilterTag` values should be compared to `FilterValue` values to establish the state of the filter condition. If `FilterTag` and `FilterComparisonValue` values are specified, time periods are filtered from the results where the filter condition is `False`. |
| `FilterValue` | `VT_BSTR` | The value with which to compare the `FilterTag` value to determine appropriate filter times. |
| `FilterExpression` | `VT_BSTR` | An expression which includes one or more filter conditions. The type of conditions used are:<br><br>• `AND`<br>• `OR`<br>• Combination of `AND` and `OR` |

**Table 398. ihComments Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| | | This column can be used instead of the `FilterTag`, `FilterComparisonMode`, and `FilterValue` columns. While using `FilterExpression`, the expression is passed within single quotes, and for complex expressions you write the conditions within parentheses. There is no maximum length for `FilterExpression`, but if called using OLE DB or Excel, these tools may have their own limitations. |
| `TimeZone` | `VT_BSTR` | The type of time zone used:<br>• Client<br>• Server<br>• Explicit bias number (number of minutes from GMT) |
| `DaylightSavingTime` | `VT_BOOL` | Indicates whether Daylight Saving Time logic should be applied to timestamps. |
| `RowCount` | `VT_I4` | Indicates the maximum number of rows that can be returned. A value of `0` indicates that there is no limit to the number of rows returned. |

## ihComments Examples

Example SQL statements for the `ihComments` table are outlined in the following examples.

### Example 1: Retrieve All Comments for a Specific Tag for This Month

```
SELECT * FROM ihcomments WHERE tagname LIKE '*001'

AND timestamp>bom
```

### Example 2: Retrieve Comments That Contain a Substring

```
SELECT * FROM ihcomments WHERE comment LIKE '*abc*'
```

### Example 3: Retrieve All Comments in an Archive

```
SELECT * FROM ihComments WHERE timestamp<=Now

AND samplingmode=rawbytime
```

## ihTrend Table

The `ihTrend` table allows you to compare multiple tags for the same timestamp. It contains a row of data for each unique timestamp, but with columns from one or more tags. The column names are dynamic and determined by the returned tag names. The `ihTrend` table is similar to a pivot table or, for instance, a cross-tab report that you can create in Crystal Reports.

The `ihTrend` table can store up to 100 columns in a returned set. This allows you to compare `Value` columns with up to 99 tags for a single timestamp, or `Value` and `Quality` columns with up to 49 tags.

**Note:**
Currently, you cannot analyze the `ihTrend` table in Crystal Reports or the Microsoft SQL Server DTS application.

The following table describes the columns of the `ihTrend` table, including all possible tag columns. Different queries on this table can produce different column results.

**Note:**
In all column names in the following table, *TagID* is used as a placeholder for the actual tag name.

**Table 399. IhTrend Table**

| Column Name | Data Type | Description |
|---|---|---|
| `TimeStamp` | `VT_DBTimeStamp` | The date and time that the trend was generated. |
| `TimeStampSeconds` | `VT_DBTimeStamp` | The date and time for the data sample. |
| `Microseconds` | `VT_I4` | The microsecond interval for the data sample. |
| `SamplingMode` | `VT_BSTR` | The mode of sampling data from the archive:<br>• `CurrentValue`: Retrieves the current value. Time frame criteria are ignored.<br>• `Interpolated`: Retrieves evenly spaced interpolated values based on interval or `NumberOfSamples` and time frame criteria.<br>• `RawByTime`: Retrieves raw archive values based on time frame criteria.<br>• `RawByNumber`: Retrieves raw archive values based on the `StartTime`, `NumberOfSamples`, and `Direction` criteria.<br>**Note:** `EndTime` criteria are ignored for this mode.<br>• `RawByFilterToggle`: Returns filtered time ranges. The values returned are `0` and `1`. If the value is `1`, then the condition is true and `0` means false. This mode is used with the time range and `FilterTag` conditions. Results start and end with timestamps.<br>• `Calculated`: Retrieves evenly spaced calculated values based on `NumberOfSamples`, interval, time frame, and `CalculationMode` criteria.<br>• `Lab`: Returns actual collected values without interpolation.<br>• `Trend`: Returns the raw minimums and maximums for each specified interval. Use this mode to maximize performance when retrieving data points for plotting. If the sampling period does not evenly divide by the interval length, His- |

**Table 399. IhTrend Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| | | torian ignores any leftover values at the end instead of putting them into a smaller interval.<br>• `Trend2`: Returns the raw minimums and maximums for each specified interval. Use this mode to maximize performance when retrieving data points for plotting. If the sampling period does not evenly divide by the interval length, Historian puts leftover values into a remainder interval. This mode is more suitable than the `Trend` mode for analysis of minimums and maximums and for plotting programs that can handle unevenly spaced data.<br>• `InterpolatedtoRaw`: Provides raw data in place of interpolated data when the number of samples is less than the number of available samples.<br>• `TrendtoRaw`: This mode almost always produces the same results as the `Trend` mode. The exception is that when the number of samples requested is greater than the number of raw data points, this mode returns all available raw data points with no further processing. This mode is therefore used instead of `Trend` when the number of actual data samples is less than the requested number of samples.<br>• `TrendtoRaw2`: This mode almost always produces the same results as the `Trend2` mode. The exception is that when the number of samples requested is greater than the number of raw data points, this mode returns all available raw data points with no further processing. This mode is therefore used instead of `Trend2` when the number of actual data samples is less than the requested number of samples.<br>• `LabtoRaw`: Provides raw data for the selected calculated data when the number of requested samples is less than the number of available samples. |
| `Direction` | `VT_BSTR` | The direction (forward or backward from the start time) of data sampling from the archive. |
| `NumberOfSamples` | `VT_I4` | Number of samples to retrieve from the archive.<br><br>Samples will be evenly spaced within the start and end times defined for most sampling modes. For the `RawByNumber` mode, this column determines the maximum number of values to retrieve. For the `RawByTime` mode, this column is ignored. |

**Table 399. IhTrend Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| | | **Note:**<br>The `NumberofSamples` and `IntervalMilliseconds` columns are mutually exclusive. If `NumberofSamples` is used, `IntervalMilliseconds` is not used. |
| `IntervalMilliseconds` | `VT_I4` | For non-raw sampled data, this column represents a positive integer for the time interval (in milliseconds) between returned samples.<br>**Note:**<br>The `NumberofSamples` and `IntervalMilliseconds` columns are mutually exclusive. If `NumberofSamples` is used, `IntervalMilliseconds` is not used. |
| `CalculationMode` | `VT_BSTR` | This column applies only if the `SamplingMode` is set to `Calculated`. It represents the type of calculation to perform on archive data:<br>• `Average`<br>• `Count`<br>• `Maximum`<br>• `MaximumTime`<br>• `Minimum`<br>• `MinimumTime`<br>• `StandardDeviation`<br>• `Total`<br>• `RawAverage`<br>• `RawStandardDeviation`<br>• `RawTotal`<br>• `TimeGood`<br>• `FirstRawValue`<br>• `FirstRawTime`<br>• `LastRawValue`<br>• `LastRawTime`<br>• `TagStats` |
| `FilterTag` | `VT_BSTR` | Tagname used to define the filter, if specified. Only a single tag can be specified. Wildcards are not supported. |

**Table 399. IhTrend Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| `Filter-Mode` | `VT_BSTR` | The type of time filter:<br>• `ExactTime`: Retrieves data for the exact times that the filter condition is `True`.<br>• `BeforeTime`: Retrieves data from the time of the last `False` filter condition to the time of the next `True` condition.<br>• `AfterTime`: Retrieves data from the time of the last `True` filter condition to the time of the next `False` condition.<br>• `BeforeAndAfterTime`: Retrieves data from the time of the last `False` filter condition to the time of the next `False` condition.<br>This value defines how time periods before and after transitions in the filter condition should be handled.<br>For example, `AfterTime` indicates that the filter condition should be `True` starting at the timestamp of the archive value that triggered the `True` condition and ending at the timestamp of the archive value that triggered the `False` condition. |
| `Filter-ComparisonMode` | `VT_BSTR` | The type of comparison to be made on the filter comparison value:<br>• `Equal`: Filter condition is `True` when the `FilterTag` value is equal to the comparison value.<br>• `EqualFirst`: Filter condition is `True` when the `FilterTag` value is equal to the first comparison value.<br>• `EqualLast`: Filter condition is `True` when the `FilterTag` value is equal to the last comparison value.<br>• `NotEqual`: Filter condition is `True` when the `FilterTag` value is not equal to the comparison value.<br>• `LessThan`: Filter condition is `True` when the `FilterTag` value is less than the comparison value.<br>• `GreaterThan`: Filter condition is `True` when the `FilterTag` value is greater than the comparison value.<br>• `LessThanEqual`: Filter condition is `True` when the `FilterTag` value is less than or equal to the comparison value.<br>• `GreaterThanEqual`: Filter condition is `True` when the `FilterTag` value is greater than or equal to the comparison value. |

**Table 399. IhTrend Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| | | • `AllBitsSet`: Filter condition is `True` when the binary `FilterTag` value is equal to all the bits in the condition. It is represented as `^` to be used in `FilterExpression`.<br>• `AnyBitSet`: Filter condition is `True` when the binary `FilterTag` value is equal to any of the bits in the condition. It is represented as `~` to be used in `FilterExpression`.<br>• `AnyBitNotSet`: Filter condition is `True` when the binary `FilterTag` value is not equal to any one of the bits in the condition. It is represented as `!~` to be used in `FilterExpression`.<br>• `AllBitsNotSet`: Filter condition is `True` when the binary `FilterTag` value is not equal to all the bits in the condition. It is represented as `!^` to be used in `FilterExpression`.<br><br>This column defines how archive values for the `FilterTag` value should be compared to the `FilterValue` value to establish the state of the filter condition. If `FilterTag` and `FilterComparisonValue` values are specified, time periods are filtered from the results where the filter condition is `False`. |
| `FilterValue` | `VT_BSTR` | The value with which to compare the `FilterTag` value to determine appropriate filter times. |
| `FilterExpression` | `VT_BSTR` | An expression which includes one or more filter conditions. The type of conditions used are:<br><br>• `AND`<br>• `OR`<br>• Combination of `AND` and `OR`<br><br>This column can be used instead of the `FilterTag`, `FilterComparisonMode`, and `FilterValue` columns. While using `FilterExpression`, the expression is passed within single quotes. For complex expressions, you write the conditions within parentheses. There is no maximum length for `FilterExpression`, but if called using OLE DB or Excel, these tools may have their own limitations. |
| `TimeZone` | `VT_BSTR` | The type of time zone used: |

**Table 399. IhTrend Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| | | • Client<br>• Server<br>• Explicit bias number (number of minutes from GMT) |
| `Day-light-Saving-Time` | `VT_BOOL` | Indicates whether Daylight Saving Time logic should be applied to timestamps. |
| `RowCount` | `VT_I4` | Indicates the maximum number of rows that can be returned. A value of `0` indicates that there is no limit to the number of rows returned. |
| *`Tag-ID`*`.Value` | `VT_-VARIANT` | The value of the data for the specified tag ID. |
| *`Tag-ID`*`.Quality` | `VT_-VARIANT` | For non-raw sampled data, this column displays the percentage of good quality samples in the interval. For instance, a value of `100` means all samples in the interval are good.<br><br>For raw sampled data, data values are:<br><br>• `Good`<br>• `Bad`<br>• `Uncertain`<br>• `Not Available`<br>• `Really Unknown`<br><br>This column also includes the subquality of the data value, if it exists:<br><br>• `NonSpecific`<br>• `ConfigError`<br>• `NotConnected`<br>• `DeviceFail`<br>• `SensorFail`<br>• `LastKnownValue`<br>• `CommFailure`<br>• `OutOfService` |

**Table 399. IhTrend Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| | | • `ScaledOutOfRange`<br>• `OffLine`<br>• `NoValue`<br>• `Really Unknown` |
| *`Tag-ID`*`.Tag-name` | `VT_BSTR` | Tagname property of the specified tag ID. |
| *`Tag-ID`*`.De-scrip-tion` | `VT_BSTR` | User description for the specified tag ID. |
| *`Tag-ID`*`.EngU-nits` | `VT_BSTR` | Engineering unit description for the specified tag ID. |
| *`Tag-ID`*`.Com-ment` | `VT_BSTR` | User comment associated with the specified tag ID. |
| *`Tag-ID`*`.Data-Type` | `VT_BSTR` | The data type for the specified tag ID:<br>• `Scaled`<br>• `SingleFloat`<br>• `DoubleFloat`<br>• `SingleInteger`<br>• `DoubleInteger`<br>• `QuadInteger`<br>• `UnsignedSingleInteger`<br>• `UnsignedDoubleInteger`<br>• `UnsignedQuadInteger`<br>• `FixedString`<br>• `VariableString`<br>• `Byte`<br>• `Boolean` |

**Table 399. IhTrend Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| | | • `BLOB`<br>• `Time`<br>• `Undefined`<br><br>The data type returned in this column is the data type that you defined in Historian Administrator application. |
| *Tag-ID*.Fixed-String-Length | VT_UI1 | This value is `0` unless the data type is `FixedString`. If the data type is `FixedString`, this number represents the maximum length of the string value. |
| *Tag-ID*.Collector-Name | VT_BSTR | The name of the collector responsible for collecting data for the specified tag ID. |
| *Tag-ID*.Source-Address | VT_BSTR | The address used to identify the specified tag ID at the data source. For iFIX systems, this is the NTF (`Node.Tag.Field`). |
| *Tag-ID*.Collection-Type | VT_BSTR | Type of collection used to acquire data for the tag:<br><br>• **Unsolicited:** The collector accepts data from the source whenever the source presents the data.<br>• **Polled:** The collector acquires data from a source on a periodic schedule determined by the collector.<br><br>**Note:**<br>Not all collectors support unsolicited collection. |
| *Tag-ID*.Collection-Interval | VT_I4 | The time interval, in milliseconds, between readings of data from this tag.<br><br>For polled collection, this field represents the time between samples. For unsolicited collection, this field represents the minimum time allowed between samples. |

**Table 399. IhTrend Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| *Tag-ID*.Collection-Offset | VT_I4 | The time shift from midnight, in milliseconds, for collection of data from this tag. |
| *Tag-ID*.Load-Balancing | VT_BOOL | Indicates whether the data collector should automatically shift the phase of sampling to distribute the activity of the processor evenly over the polling cycle for the specified tag ID. This is sometimes called phase shifting. |
| *Tag-ID*.Time-Stamp-Type | VT_BSTR | The timestamp type applied to data samples at collection time:<br>• Source: The source delivers the timestamp along with the data sample.<br>• Collector: The collector delivers the timestamp along with the collected data. |
| *Tag-ID*.Hi-Engineering-Units | VT_R8 | The high end of the engineering units range. Used only for scaled data types and input scaled tags. |
| *Tag-ID*.Lo-Engineering-Units | VT_R8 | The low end of the engineering units range. Used only for scaled data types and input scaled tags. |
| *Tag-ID*.InputScaling | VT_BOOL | Indicates whether the measurement should be converted to an engineering units value. When set to False, the measurement is interpreted as a raw measurement.<br>When set to True, the system converts the value to engineering units by scaling the value between the HiScale and LoScale values. If not enabled, the system assumes the measurement is already converted into engineering units. |
| *Tag-ID*.HiScale | VT_R8 | The high-end value of the input scaling range used for the tag. |

**Table 399. IhTrend Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| *Tag-ID*.LoScale | VT_R8 | The low-end value of the input scaling range used for the tag. |
| *Tag-ID*.Collector-Compression | VT_BOOL | Indicates whether collector compression is enabled for the specified tag ID.<br><br>Collector compression applies a smoothing filter to incoming data by ignoring incremental changes in values that fall within a deadband centered around the last collected value. The collector passes (to the archiver) any new value that falls outside the deadband and then centers the deadband around the new value. |
| *Tag-ID*.Collector-Deadband-Percent-Range | VT_R4 | The current value of the compression deadband. |
| *Tag-ID*.Archive-Compression | VT_BOOL | Indicates whether archive collector compression is enabled for the tag. |
| *Tag-ID*.Archive-Deadband-Percent-Range | VT_R4 | The current value of the archive compression deadband. |
| *Tag-ID*.Collector-General1 | VT_BSTR | The general (or spare) configuration fields for the specified tag ID. |

**Table 399. IhTrend Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| *Tag-ID*.Collector-General2 | VT_BSTR | The general (or spare) configuration fields for the specified tag ID. |
| *Tag-ID*.Collector-General3 | VT_BSTR | The general (or spare) configuration fields for the specified tag ID. |
| *Tag-ID*.Collector-General4 | VT_BSTR | The general (or spare) configuration fields for the specified tag ID. |
| *Tag-ID*.Collector-General5 | VT_BSTR | The general (or spare) configuration fields for the specified tag ID. |
| *Tag-ID*.ReadSecurityGroup | VT_BSTR | The name of the Windows security group that controls the reading of data for the specified tag ID.<br><br>Refer to "Implementing Historian Security" in the *Getting Started with Historian* manual for definitions of the various security levels and groups. |
| *Tag-ID*.WriteSecurityGroup | VT_BSTR | The name of the Windows security group that controls the writing of data for the specified tag ID.<br><br>Refer to "Implementing Historian Security" in the *Getting Started with Historian* manual for definitions of the various security levels and groups. |
| *Tag-ID*.AdministratorSecurityGroup | VT_BSTR | The name of the Windows security group responsible for controlling configuration changes for the specified tag ID. |

**Table 399. IhTrend Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| *Tag-ID*.Calculation | VT_BSTR | The equation for the calculation performed for the specified tag ID. |
| *Tag-ID*.LastModified | VT_DBTimeStamp | The date and time that the tag configuration was last modified. The time structure includes milliseconds. |
| *Tag-ID*.LastModifiedUser | VT_BSTR | The username of the Windows user who last modified the tag configuration. |
| *Tag-ID*.CollectorType | VT_BSTR | The type of collector responsible for collecting data for the specified tag ID:<br>• Undefined<br>• iFIX<br>• Simulation<br>• OPC<br>• File<br>• iFIXLabData<br>• ManualEntry<br>• Simulation<br>• Other |
| *TagID*.StoreMilliseconds | VT_BOOL | Indicates whether time resolution in milliseconds is enabled for the specified tag ID.<br><br>If not enabled, time resolution is in seconds instead of milliseconds. Maximum data compression is achieved when this value is set to False. This is the optimum setting for most applications. |
| *Tag-ID*.UTCBias | VT_I4 | The time zone bias for the specified tag ID. Time zone bias is used to indicate the natural time zone of the tag expressed as an offset from UTC (Universal Time Coordinated) in minutes.<br><br>UTC is the international time standard, the current term for what was commonly referred to as Greenwich Mean Time (GMT). |

**Table 399. IhTrend Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| *Tag-ID*.AverageCollectionTime | VT_I4 | The average time it takes to execute the calculation tag since you started the Calculation collector for the specified tag ID. |
| *Tag-ID*.CollectionDisabled | VT_I4 | Indicates whether collection is enabled (0) or disabled (1) for the specified tag ID. The default setting is enabled (0). |
| *Tag-ID*.CollectorCompressionTimeout | VT_I4 | Indicates the maximum amount of time the collector will wait between sending samples to the archiver. This time is kept per tag, as different tags report to the archiver at different times.<br>This value should be in increments of your collection interval, and not less.<br>Ideally, this field is used for polled data values. It can be used with unsolicited data, but when you do so, you are dependent on the data source for the value to change. With unsolicited data, since Historian only records the value when it changes, the actual time before the timeout might exceed the compression timeout. |
| *Tag-ID*.ArchiveCompressionTimeout | VT_I4 | Indicates the maximum amount of time from the last stored point before another point is stored, if the value does not exceed the archive compression deadband for the specified tag ID. |
| *Tag-ID*.InterfaceAbsoluteDeadbanding | VT_BOOL | Indicates whether absolute collector deadband is enabled for the specified tag ID. |
| *Tag-ID*.In- | VT_R8 | Indicates the value for absolute collector deadband. |

**Table 399. IhTrend Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| `terface-Absolut-eDead-band` | | |
| *`Tag-ID`*`.Archive-Absolut-eDead-banding` | `VT_BOOL` | Indicates whether absolute archive deadband is enabled for the specified tag ID. |
| *`Tag-ID`*`.Archive-Absolut-eDead-band` | `VT_R8` | Indicates the value for absolute archive deadband. |
| *`Tag-ID`*`.Spike-Logic` | `VT_BOOL` | Indicates whether Spike Logic is enabled on the collector. |
| *`Tag-ID`*`.Spike-LogicOv-erride` | `VT_BOOL` | Indicates whether the Spike Logic setting for the specified tag ID overrides the collector setting (`True`) or the collector setting is used (`False`). |

Use care when building queries against the `ihTrend` table. Because a query to this table compares multiple tags at the same time, it takes longer to query the `ihTrend` table than it does the `ihRawData` table. The `ihTrend` table can be quite large, so be sure to either use the default start and end times, or define a specific time interval. See Query Performance Optimization *(on page 2611)* for more ideas on how to optimize your query of the `ihTrend` table.

## ihTrend Examples

Example SQL statements for the `ihTrend` table are outlined in the following examples.

### Example 1: Retrieve Value and Quality of the First 50 Tags

```
SELECT timestamp, *.value, *.quality FROM ihtrend
```

### Example 2: Retrieve Value of the First 100 Tags

```
SELECT timestamp, *.value FROM ihTrend
```

### Example 3: Retrieve Values of All Tags That Match a Specific Pattern

```
SELECT timestamp,*0001.value FROM ihtrend ORDER BY MY_SERVER.Simulation00001.Value
```

### Example 4: Retrieve Hourly Interpolated Values of TagNames That Match *0001

```
SET samplingmode=interp, intervalmilliseconds=1h

SELECT timestamp, *0001.value FROM ihtrend

ORDER BY Simulation00001.value DESC, timestamp DESC
```

### Example 5: Retrieve Maximum Values of All TagNames That Match *0001

The following example shows how to use a `TagName` (`simulation.00001.Value`) in a `WHERE` clause.

```
SELECT timestamp, *0001.value FROM ihtrend

WHERE timestamp>='28-nov-2001 00:00'

AND timestamp<='29-nov-2001 00:00:00'

AND samplingmode=calc

AND intervalmilliseconds=1h

AND calculationmode=max

AND simulation00001.Value > 1000 ORDER BY timestamp
```

### Example 6: Select Interpolated Values for All Single Float Tags

The following example shows how to select interpolated values for all single float tags, without doing a `JOIN` with the `ihTags` table to retrieve the `DataType` property.

```
SELECT timestamp, *.value,*.description FROM ihtrend

WHERE timestamp>>='28-nov-2001 00:00'

AND timestamp<='29-nov-2001 00:00:00'

AND samplingmode=calculated

AND intervalmilliseconds=2h

AND *.datatype = singlefloat ORDER BY timestamp
```

### Example 7: Select Interpolated Data for TagNames That Match sim*

The following example shows how to sort the returned rows by a `TagName` , `simulation.00001.Value`.

```
SET starttime='28-nov-2001 00:00', endtime='29-nov-2001 00:00:00', samplingmode=interp, intervalmilliseconds=1h

SELECT timestamp, sim*.*, sim*.description, sim*.lastmodifieduser FROM ihtrend

WHERE sim*.description LIKE '*sim*'

AND sim*.description like '*first*'

AND *.datatype = singlefloat

ORDER BY simulation00001.value DESC, timestamp
```

## ihQuerySettings Table

The `ihQuerySettings` table contains the current session settings. These settings are applied to all queries you make in a session, unless overridden with a `WHERE` clause. This table displays settings stored in the provider, and has nothing to do with the data stored in the archives.

The `ihQuerySettings` table provides a convenient way to display all your session settings. You cannot, however, write or update settings in this table. This table contains only one row with the settings for the current session. The only way to change these parameters is by using the `SET` statement.

The following table describes the columns of the `ihQuerySettings` table.

**Table 400. ihQuerySettings Table**

| Column Name | Data Type | Description |
|---|---|---|
| `StartTime` | `VT_DB_Time_Stamp` | The start time of the query. This represents the earliest timestamp for any tag contained in the query.<br>If no `StartTime` value is specified, the start time is two hours prior to execution of the query. |
| `EndTime` | `VT_DB_Time_Stamp` | The end time of the query. This represents the latest timestamp for any tag contained in the query.<br>If no `EndTime` value is specified, the end time is the time that you execute the query. |

**Table 400. ihQuerySettings Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| `SamplingMode` | `VT_BSTR` | The mode of sampling data from the archive:<br>• `CurrentValue`: Retrieves the current value. Time frame criteria are ignored.<br>• `Interpolated`: Retrieves evenly spaced interpolated values based on interval or `NumberOfSamples` and time frame criteria.<br>• `RawByTime`: Retrieves raw archive values based on time frame criteria.<br>• `RawByNumber`: Retrieves raw archive values based on the `StartTime`, `NumberOfSamples`, and `Direction` criteria.<br>**Note:**<br>`EndTime` criteria are ignored for this mode.<br>• `RawByFilterToggle`: Returns filtered time ranges. The values returned are `0` and `1`. If the value is `1`, then the condition is true and `0` means false. This mode is used with the time range and `FilterTag` conditions. Results start and end with timestamps.<br>• `Calculated`: Retrieves evenly spaced calculated values based on `NumberOfSamples`, interval, time frame, and `CalculationMode` criteria.<br>• `Lab`: Returns actual collected values without interpolation.<br>• `Trend`: Returns the raw minimums and maximums for each specified interval. Use this mode to maximize performance when retrieving data points for plotting. If the sampling period does not evenly divide by the interval length, Historian ignores any leftover values at the end instead of putting them into a smaller interval.<br>• `Trend2`: Returns the raw minimums and maximums for each specified interval. Use this mode to maximize performance when retrieving data points for plotting. If the sampling period does not evenly divide by the interval length, Historian puts leftover values into a remainder interval. This mode is more suitable than the `Trend` mode for analysis of minimums and maximums and for plotting programs that can handle unevenly spaced data.<br>• `InterpolatedtoRaw`: Provides raw data in place of interpolated data when the number of samples is less than the number of available samples.<br>• `TrendtoRaw`: This mode almost always produces the same results as the `Trend` mode. The exception is that when the number of samples requested is greater than the number of raw data points, this mode returns all available raw data points with no |

**Table 400. ihQuerySettings Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| | | further processing. This mode is therefore used instead of `Trend` when the number of actual data samples is less than the requested number of samples.<br>• `TrendtoRaw2`: This mode almost always produces the same results as the `Trend2` mode. The exception is that when the number of samples requested is greater than the number of raw data points, this mode returns all available raw data points with no further processing. This mode is therefore used instead of `Trend2` when the number of actual data samples is less than the requested number of samples.<br>• `LabtoRaw`: Provides raw data for the selected calculated data when the number of requested samples is less than the number of available samples.<br><br>`Calculated` is the default setting. |
| `Direction` | `VT_BSTR` | The direction (`Forward` or `Backward` from the start time) of data sampling from the archive. The default value is `Forward`. |
| `NumberOfSamples` | `VT_I4` | Number of samples to retrieve from the archive.<br><br>Samples will be evenly spaced within the specified start and end times defined for most sampling modes. For the `RawByNumber` mode, this column determines the maximum number of values to retrieve. For the `RawByTime` mode, this column is ignored.<br><br>**Note:**<br>The `NumberofSamples` and `IntervalMilliseconds` columns are mutually exclusive. If `NumberofSamples` is used, `IntervalMilliseconds` is not used. |
| `IntervalMilliseconds` | `VT_I4` | For non-raw sampled data, this column represents a positive integer for the time interval (in milliseconds) between returned samples.<br><br>**Note:**<br>The `NumberofSamples` and `IntervalMilliseconds` columns are mutually exclusive. If `IntervalMilliseconds` is used, `NumberofSamples` is not used. |
| `Calcula-` | `VT_BSTR` | This column applies only if the `SamplingMode` is set to `Calculated`. It represents the type of calculation to perform on archive data: |

**Table 400. ihQuerySettings Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| `tion-Mode` | | • `Average`<br>• `Count`<br>• `Maximum`<br>• `MaximumTime`<br>• `Minimum`<br>• `MinimumTime`<br>• `OPCQOr` and `OPCQAnd`<br>• `StandardDeviation`<br>• `StateCount`<br>• `StateTime`<br>• `Total`<br>• `RawAverage`<br>• `RawStandardDeviation`<br>• `RawTotal`<br>• `TimeGood`<br>• `FirstRawValue`<br>• `FirstRawTime`<br>• `LastRawValue`<br>• `LastRawTime`<br>• `TagStats`<br><br>The default value is `Average`. |
| `Fil-ter-Tag` | `VT_-BSTR` | Tagname used to define the filter, if specified. Only a single tag can be specified. Wildcards are not supported. |
| `Fil-ter-Mode` | `VT_-BSTR` | The type of time filter:<br><br>• `ExactTime`: Retrieves data for the exact times that the filter condition is `True`.<br>• `BeforeTime`: Retrieves data from the time of the last `False` filter condition to the time of the next `True` condition. |

**Table 400. ihQuerySettings Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| | | • `AfterTime`: Retrieves data from the time of the last `True` filter condition to the time of the next `False` condition.<br>• `BeforeAndAfterTime`: Retrieves data from the time of the last `False` filter condition to the time of the next `False` condition.<br><br>This value defines how time periods before and after transitions in the filter condition should be handled.<br><br>For example, `AfterTime` indicates that the filter condition should be `True` starting at the timestamp of the archive value that triggered the `True` condition and ending at the timestamp of the archive value that triggered the `False` condition. |
| `FilterComparisonMode` | `VT_BSTR` | The type of comparison to be made on the filter comparison value:<br><br>• `Equal`: Filter condition is `True` when the `FilterTag` value is equal to the comparison value.<br>• `EqualFirst`: Filter condition is `True` when the `FilterTag` value is equal to the first comparison value.<br>• `EqualLast`: Filter condition is `True` when the `FilterTag` value is equal to the last comparison value.<br>• `NotEqual`: Filter condition is `True` when the `FilterTag` value is not equal to the comparison value.<br>• `LessThan`: Filter condition is `True` when the `FilterTag` value is less than the comparison value.<br>• `GreaterThan`: Filter condition is `True` when the `FilterTag` value is greater than the comparison value.<br>• `LessThanEqual`: Filter condition is `True` when the `FilterTag` value is less than or equal to the comparison value.<br>• `GreaterThanEqual`: Filter condition is `True` when the `FilterTag` value is greater than or equal to the comparison value.<br>• `AllBitsSet`: Filter condition is `True` when the binary `FilterTag` value is equal to all the bits in the condition. It is represented as `^` to be used in `FilterExpression`.<br>• `AnyBitSet`: Filter condition is `True` when the binary `FilterTag` value is equal to any of the bits in the condition. It is represented as `~` to be used in `FilterExpression`. |

**Table 400. ihQuerySettings Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| | | • `AnyBitNotSet`: Filter condition is `True` when the binary `FilterTag` value is not equal to any one of the bits in the condition. It is represented as `!~` to be used in `FilterExpression`.<br>• `AllBitsNotSet`: Filter condition is `True` when the binary `FilterTag` value is not equal to all the bits in the condition. It is represented as `!^` to be used in `FilterExpression`.<br><br>This option defines how archive values for the `FilterTag` value should be compared to the `FilterValue` value to establish the state of the filter condition. If `FilterTag` and `FilterComparisonValue` values are specified, time periods are filtered from the results where the filter condition is `False`. |
| `FilterValue` | `VT_BSTR` | The value with which to compare the `FilterTag` value to determine appropriate filter times. |
| `FilterExpression` | `VT_BSTR` | An expression which includes one or more filter conditions. The type of conditions used are:<br><br>• `AND`<br>• `OR`<br>• Combination of `AND` and `OR`<br><br>This column can be used instead of the `FilterTag`, `FilterComparisonMode`, and `FilterValue` columns. While using `FilterExpression`, the expression is passed within single quotes. For complex expressions, you write the conditions within parentheses. There is no maximum length for this value, but if called using OLE DB or Excel, these tools may have their own limitations. |
| `TimeZone` | `VT_BSTR` | The type of time zone used:<br><br>• Client<br>• Server<br>• Explicit bias number (number of minutes from GMT) |
| `Daylight` | `VT_BOOL` | Indicates whether Daylight Saving Time logic should be applied to timestamps. |

**Table 400. ihQuerySettings Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| `SavingTime` | | |
| `RowCount` | `VT_I4` | Indicates the maximum number of rows that can be returned. A value of `0` indicates that there is no limit to the number of rows returned.<br><br>If the query result contains more rows than the `RowCount` value, the Historian OLE DB provider truncates the extra rows. The truncation is performed last. For instance, if you use `ORDER BY` in your `SELECT` statement, the truncation occurs after the rows are ordered. |
| `AlarmType` | `VT_BSTR` | Indicates the alarm type:<br><br>• `Alarms`<br>• `Alarm_History`<br>• `Events` |

### ihQuerySettings Examples

Example SQL statements for the `ihQuerySettings` table are outlined in the following examples.

### Example 1: Show All Settings for the Current Session

```
SELECT * FROM ihquerysettings
```

### Example 2: Show the Selected Session Settings

```
SELECT starttime, endtime FROM ihquerysettings
```

## ihCalculationDependencies Table

The `ihCalculationDependencies` table contains the calculation and server-to-server tags and their triggers. The following table describes the columns of the `ihCalculationDependencies` table.

**Table 401. ihCalculationDependencies Table**

| Column Name | Data Type | Description |
|---|---|---|
| `Tagname` | `VT_-BSTR` | A calculation or server-to-server tag with unsolicited collection and at least one dependent tag. |
| `Dependent-Tag-name` | `VT_-BSTR` | A dependent tagname. If a tag has multiple dependent tags, there are multiple rows in the table for that tagname. |
| `RowCount` | `VT_I4` | Indicates the maximum number of rows that can be returned. A value of `0` indicates that there is no limit to the number of rows returned. |

### ihCalculationDependencies Examples

Example SQL statements for the `ihCalculationDependencies` table are outlined in the following examples.

### Example 1: Show the Dependencies for a Specific Tag

```
SELECT * FROM ihcalculationdependencies WHERE tagname = c1
```

### Example 2: Show the Dependencies for a Specific Dependent Tag

```
SELECT * FROM ihcalculationdependencies
WHERE dependenttagname=brahms.ai1.f_cv
```

## ihAlarms Table

The `ihAlarms` table contains collected alarms and events data. The following table describes the columns of the `ihAlarms` table.

**CAUTION:**
When you perform joins of the `ihRawData` and `ihAlarms` tables, you can easily construct queries that temporarily consume all your system resources. Although this scenario typically does not affect data collection, it can interfere with data analysis. To avoid this issue, always define a start and end time for the query to limit the number of rows returned.

**Table 402. ihAlarms Table**

| Column Name | Data Type | Description |
|---|---|---|
| `AlarmID` | `VT_I4` | The unique ID of the alarm or event in the Historian alarm database. |
| `ItemID` | `VT_BSTR` | The OPC `ItemID` of the alarm. This contains the source address of the data access tag with which the alarm is associated. This can contain a `NULL` value if an alarm is not associated with a tag. |
| `Source` | `VT_BSTR` | The unique identifier used by the OPC AE Collector for the alarm or event. |
| `DataSource` | `VT_BSTR` | The collector interface name associated with the alarm or event. |
| `Tagname` | `VT_BSTR` | The Historian tag name associated with the alarm. This value is `NULL` unless the tag is also collected by Historian. |
| `AlarmType` | `VT_BSTR` | The alarm type:<br>• `Alarms`: In Historian, the full life cycle of an alarm is stored as a single record in the alarm archive.<br>• `Alarm_History`: The separate transitions for all alarms. One row per transition is returned.<br>• `Events`: The simple and tracking events. |
| `EventCategory` | `VT_BSTR` | The OPC event category of the alarm or event. |
| `Condition` | `VT_BSTR` | The OPC condition of the alarm. Does not apply to event data. This value combined with the `Source` value comprises an alarm. |
| `SubCondition` | `VT_BSTR` | The OPC subcondition of the alarm. Does not apply to event data. This value represents the state of the alarm. |
| `StartTime` | `VT_DBTimeStamp` | The start time or timestamp of the alarm or event. |
| `EndTime` | `VT_DBTimeStamp` | The end time of the alarm. Does not apply to event data. |

**Table 402. ihAlarms Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| `AckTime` | `VT_DB-TimeS-tamp` | The time the alarm was acknowledged. Does not apply to event data. |
| `Microsec-onds` | `VT_I4` | The microsecond portion of the date and time. |
| `Message` | `VT_BSTR` | The message attached to the alarm or event. |
| `Acked` | `VT_BOOL` | Stores the acknowledgement status of the alarm. If the alarm is acknowledged, this is set to `TRUE`. |
| `Severity` | `VT_I4` | The severity of the alarm or event. Stored as an integer value with a range of 1–1000. |
| `Actor` | `VT_BSTR` | The operator who acknowledged the alarm, or caused the tracking event. |
| `Quality` | `VT_-VARIANT` | The quality of the alarm or event. Stored as a string, with values of `GOOD` or `BAD`. |
| `TimeZone` | `VT_BSTR` | The type of time zone used:<br>• Client<br>• Server<br>• Explicit bias number (number of minutes from GMT) |
| `Daylight-SavingTime` | `VT_BOOL` | Indicates whether Daylight Saving Time logic should be applied to timestamps. |
| `RowCount` | `VT_I4` | The maximum number of rows returned by the current query. |
| `User-De-fined Variable #X` | `VT_-VARIANT` | User-defined variables. This is a dynamic list of columns that varies based on the collectors running on the Historian system. |

**Note:**
Additional fields may be added by third-party products such as iFIX. Please consult the relevant product documentation for further information.

## ihAlarms Examples

### Example 1: Show All Alarms for the Last Two Hours, Including Vendor Attributes

```
SELECT * FROM ihAlarms

SELECT * FROM ihAlarms WHERE alarmtype = alarms //same as above
```

### Example 2: Show Alarm History

```
SELECT * FROM ihAlarms WHERE alarmtype = alarm_history
```

### Example 3: Show Tracking and System Events

```
SELECT * FROM ihAlarms WHERE alarmtype = events
```

### Example 4: Return All Closed Events and Associated Tag Data

```
SELECT

alarmid, ihalarms.tagname, ihalarms.starttime, ihalarms.endTime, ihrawdata.timestamp, ihrawdata.value

FROM ihalarms, ihrawdata

WHERE ihalarms.tagname=ihrawdata.tagname

AND ihalarms.starttime <= ihrawdata.timestamp

AND ihalarms.endtime >= ihRawdata.timestamp

AND ihalarms.subcondition == "OK"

OR ihalarms.quality = "Bad"

ORDER BY ihalarms.starttime
```

**Note:**

When you join data from the `ihRawData` and `ihAlarms` tables, be sure to specify a timestamp range.

### Example 5: Return All Open Alarms and Associated Tag Data

```
SELECT

alarmid, ihalarms.tagname, ihalarms.starttime, ihalarms.endTime, ihrawdata.timestamp, ihrawdata.value

FROM ihalarms, ihrawdata

WHERE ihalarms.tagname=ihrawdata.tagname

AND ihalarms.starttime <= ihrawdata.timestamp

AND ihalarms.endtime >= ihRawdata.timestamp

AND ihalarms.subcondition <> "OK"

AND ihalarms.quality = "Good"
```

```
ORDER BY ihalarms.starttime
```

**Note:**
When you join data from the `ihRawData` and `ihAlarms` tables, be sure to specify a timestamp range.

## ihEnumeratedSets Table

The `ihEnumeratedSets` table contains information about enumerated sets that are defined in the system. The following table describes the columns of the `ihEnumeratedSets` table.

**Table 403. ihEnumeratedSets Table**

| Column Name | Data Type | Description |
|---|---|---|
| `SetName` | `VT_BSTR` | The name of the set. |
| `Description` | `VT_BSTR` | The description of the set. |
| `NumberofStates` | `VT_I4` | The number of states a set contains. |
| `NumberofTag-References` | `VT_I4` | The number of tags with which a set is associated. |
| `SetDataType` | `VT_BSTR` | The data type of the set. |
| `Administrator-SecurityGroup` | `VT_BSTR` | The security group to which the set belongs. |
| `LastModified-User` | `VT_BSTR` | Indicates which user last modified the set. |
| `LastModified-Time` | `VT_DB-TimeS-tamp` | Indicates the last time the set was modified. |
| `RowCount` | `VT_I4` | Indicates the maximum number of rows that can be returned. A value of `0` indicates that there is no limit to the number of rows returned. |

### ihEnumeratedSets Examples

Sample SQL statements for the `ihEnumeratedSets` table are outlined in the following examples.

### Example 1: Retrieve All Sets By Using Integer States

```
SELECT * FROM ihEnumeratedSets

WHERE SetDataType='integer'
```

### Example 2: Retrieve a Set By Name From Sets

```
SELECT * FROM ihEnumeratedSets

WHERE setname like PLC1
```

## ihEnumeratedStates Table

The `ihEnumeratedStates` table contains information about enumerated sets that are defined in the system. The following table describes the columns of the `ihEnumeratedStates` table.

**Table 404. ihEnumeratedStates Table**

| Column Name | Data Type | Description |
|---|---|---|
| `SetName` | `VT_BSTR` | The name of the set. |
| `Description` | `VT_BSTR` | The description of the set. |
| `NumberofStates` | `VT_I4` | The number of states a set contains. |
| `NumberofTag-References` | `VT_I4` | The number of tags with which a set is associated. |
| `SetDataType` | `VT_BSTR` | The data type of the set. |
| `Administrator-SecurityGroup` | `VT_BSTR` | The security group to which the set belongs. |
| `LastModified-User` | `VT_BSTR` | Indicates which user last modified the set. |
| `LastModified-Time` | `VT_DB-TimeS-tamp` | Indicates the last time the set was modified. |
| `RowCount` | `VT_I4` | Indicates the maximum number of rows that can be returned. A value of `0` indicates that there is no limit to the number of rows returned. |

### ihEnumeratedStates Examples

Sample SQL statements for the `ihEnumeratedStates` table are outlined in the following examples.

### Example 1: Retrieve All States That Belong to a Specific Set

```
SELECT * FROM ihEnumeratedStates

WHERE setname=plcset1 order by statelowvalue ascending
```

### Example 2: Retrieve All States From a Specific Set

```
SELECT * FROM ihEnumeratedStates

WHERE setname = 'setname'
```

## ihUserDefinedTypes Table

The `ihUserDefinedTypes` table contains information about user-defined data types in the system.

Use this table to see the set of types and get information about each field in the data type.

The following table describes the columns of the `ihUserDefinedTypes` table.

**Table 405. ihUserDefinedTypes Table**

| Column Name | Data Type | Description |
|---|---|---|
| `TypeName` | `VT_BSTR` | The name of the user-defined type. |
| `DataType` | `VT_BSTR` | The data type of the user-defined type. |
| `Description` | `VT_BSTR` | The description of the user-defined type. |
| `StoreField-Quality` | `VT_BOOL` | Indicates whether the field-level quality is stored. |
| `NumberofFields` | `VT_I4` | The number of fields a user-defined type contains. |
| `NumberofTag-References` | `VT_I4` | The number of tags with which a user-defined type is associated. |
| `Administrator-SecurityGroup` | `VT_BSTR` | The security group to which the user-defined type belongs. |
| `LastModified-User` | `VT_BSTR` | Indicates which user last modified the user-defined type. |

**Table 405. ihUserDefinedTypes Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| `LastModifiedTime` | `VT_DBTimeStamp` | Indicates the last time the user-defined type was modified. |
| `RowCount` | `VT_I4` | Indicates the maximum number of rows that can be returned. A value of `0` indicates that there is no limit to the number of rows returned. |

### ihUserDefinedTypes Examples

Sample SQL statements for the `ihUserDefinedType` table are outlined in the following examples.

### Example 1: Retrieve All User-Defined Types

```
SELECT * FROM ihuserdefinedtypes
```

### Example 2: Retrieve a User-Defined Type By Name

```
SELECT * FROM ihuserdefinedtypes WHERE typename LIKE New
```

## ihFields Table

The `ihFields` table contains information about field elements that are specified in user-defined data types. The following table describes the columns of the `ihFields` table.

**Table 406. ihFields Table**

| Column Name | Data Type | Description |
|---|---|---|
| `TypeName` | `VT_BSTR` | The name of the user-defined type. |
| `FieldName` | `VT_BSTR` | The name of the field. |
| `Description` | `VT_BSTR` | The description of the field. |
| `FieldValueDataType` | `VT_BSTR` | The data type of the field. |

**Table 406. ihFields Table (continued)**

| Column Name | Data Type | Description |
|---|---|---|
| `Master-Field` | `VT_-BOOL` | Indicates whether the field is a master field. |
| `RowCount` | `VT_I4` | Indicates the maximum number of rows that can be returned. A value of `0` indicates that there is no limit to the number of rows returned. |

## ihFields Examples

Sample SQL statements for the `ihFields` table are outlined in the following examples.

## Example: Retrieve All Fields for a Specific Type

```
SELECT * FROM ihfields WHERE typename='MyUserDefinedType'
```

# Chapter 39. The Excel Add-In for Historian

## Overview of the Excel Add-In for Historian

The Excel Add-In for Historian enhances the power and benefits of using the Historian data archiving and retrieval system.

**Features:**

- You can add tags to Historian by generating a tag worksheet using the standard Excel tools, editing the parameters, and then importing the information in bulk directly into Historian.
- You can export tag parameters from Excel, make bulk changes using similar techniques, and then import the changes back into Historian.
- You can retrieve selected data from any archive file and include it in a customized report.
- You can plot the data in any of the standard chart formats.
- You can calculate derived variables from raw data values.
- You can perform mathematical functions to smooth or characterize data.
- You can import, export, and modify tags, data, and messages — all with familiar Excel commands, macros, and computational techniques.
- You can create dynamic reports that you can share among users.

### Excel Add-In window Conventions

The Excel Add-In uses several conventions in its windows that allow you to take full advantage of the features of the Historian Excel Add-In:

- You can select tags, times, and events either by cell references or by manually entering the values.
- Many windows support selecting multiple statistics or attributes. You can select multiple items in a list using one of the following methods:
    - Dragging the mouse over multiple items.
    - Pressing the Shift key and selecting the ends of a contiguous range.
    - Pressing the Control key and selecting multiple individual items.
- Specifying an output cell is optional. If you do not specify an output cell, the active cell is used as the starting point for output. When you specify an output cell, that cell is used as the starting point for output. If you select a range for an output cell, the top left cell in the range is used as the starting point for output.
- Specifying an output range determines how many data points are retrieved from a given query. It is important for these functions to specify whether you want the data points to be sorted in ascending or descending order by selecting the appropriate option.

- When you specify an output range or an output cell, ensure that the active cells are not the same cells that you specified with tag name cell references. Otherwise, it will lead to circular cell referencing and incorrect values.
- Specifying data retrieval into rows or columns determines how multiple attributes or statistics are displayed in the worksheet.
- Specifying data retrieval into rows or columns only applies when the window inserts a single function into the worksheet. When you select a multi-cell output range, the orientation of that range determines whether the requested data is returned into rows or columns.
- Excel does not support the use of the right and left arrow keys of the keyboard to move between characters in text boxes and fields in the windows.
- If no parameters in an Excel formula change, the formula does not recalculate unless you edit the formula. For example, if you change a Hi Scale value from 100 to 50 and then import a tag, the Hi Scale field will still display 100 when looking at the tag information.
- When retrieving data, leave at least one blank line at the top of the output display for the column header labels. If you do not, the header labels will not appear.
- When you retrieve data for more than one tag, if you choose to display the timestamp in the output, then the timestamp will be displayed only once and the parameter values of the selected tags will be shown based on the orientation selected.
- In several fields, an underscore appears at the right side of the field. If you select the underscore, the window instantly changes to a minimized display. You can return to the original display by selecting the box again. The purpose of this feature is to allow you to see an unobstructed view of your worksheet or other windows as you work your way through the window and to allow you to select a cell or range of cells in the worksheet.

# Setting Up

## Install the Historian Excel Add-in Using the Installer

Install one of the following 32-bit or 64-bit Microsoft® Excel® applications:

- Microsoft® Excel® 2019
- Microsoft® Excel® 2016
- Microsoft® Excel® 2013
- Microsoft® Excel® 2010

You can install Excel Add-In separately or during Client Tools installation. This topic describes how to install Excel Add-In separately using the installer. You can also install it at a command prompt *(on page*

*195)*. However, if installing via a command prompt, do not install the Excel Add-In on the machine that you have installed the Historian Administrator or data archiver.

1. Run the `InstallLauncher.exe` file.
2. Select **Install Excel Add-in for Historian**.
   The installer runs through the installation steps.

   > **Note:**
   > If using certain versions of Windows (like Windows 10 or Windows 2019), you may receive an error message, stating that some of the DLL files are not registered. You can ignore these messages.

3. When prompted to reboot your system, select **Yes**.

Excel Add-In is installed.

Activate Excel Add-In *(on page 197)*.

## Install the Historian Excel Add-in at a Command Prompt

1. Install one of the following 32-bit or 64-bit Microsoft® Excel® applications:
   - Microsoft® Excel® 2019
   - Microsoft® Excel® 2016
   - Microsoft® Excel® 2013
   - Microsoft® Excel® 2010
2. Install Excel Add-in using the installer *(on page 195)* on your machine:
   a. Open the Command Prompt as an Administrator. (From the Start menu, right-click the Command Prompt app, and select Run as Administrator.)
   b. On the Historian install media, navigate to the Excel folder.
   c. Run the following command line: setup.exe -r.
   d. When the install runs, select the Excel option and leave the other options greyed out as they cannot be removed.

      When you run this install, a template file named setup.iss is created at `C:\Windows`. This file stores the installation options that you have provided during the installation.

      After completing the installation, you can then use this `.iss` file to install Excel Add-in at a command prompt on other machines.

On the other Client machine where you want to install the Excel Add-in, place a copy of the **setup.iss** on the local directory and open a Command Prompt (Run as Administrator). Run the silent install with following command:

```
setup.exe /s
```

If your iss file is not called **setup.iss** or is in different directory than setup.exe you need to add the flag: /f1. For example:

```
setup.exe /s /sms /f1"C:\Temp\setup.iss
```

Wait for few minutes (probably 3-5 minutes) for the install to complete.

You can install Excel Add-In separately or during Client Tools installation. However, do not install Excel Add-In on the machine on which you have installed Historian Administrator or data archiver.

1. Copy the `setup.iss` file to each machine on which you want to install Excel Add-in at a command prompt.
2. In the folder that contains the `setup.iss` file, run the following command: `setup.exe /s /sms`
   The installer runs through the installation steps.

   **Note:**
   If using certain versions of Windows (like Windows 10 or Windows 2019), you may receive an error message, stating that some of the DLL files are not registered. You can ignore these messages.

3. When prompted to reboot your system, select **Yes**.

Excel Add-In is installed.

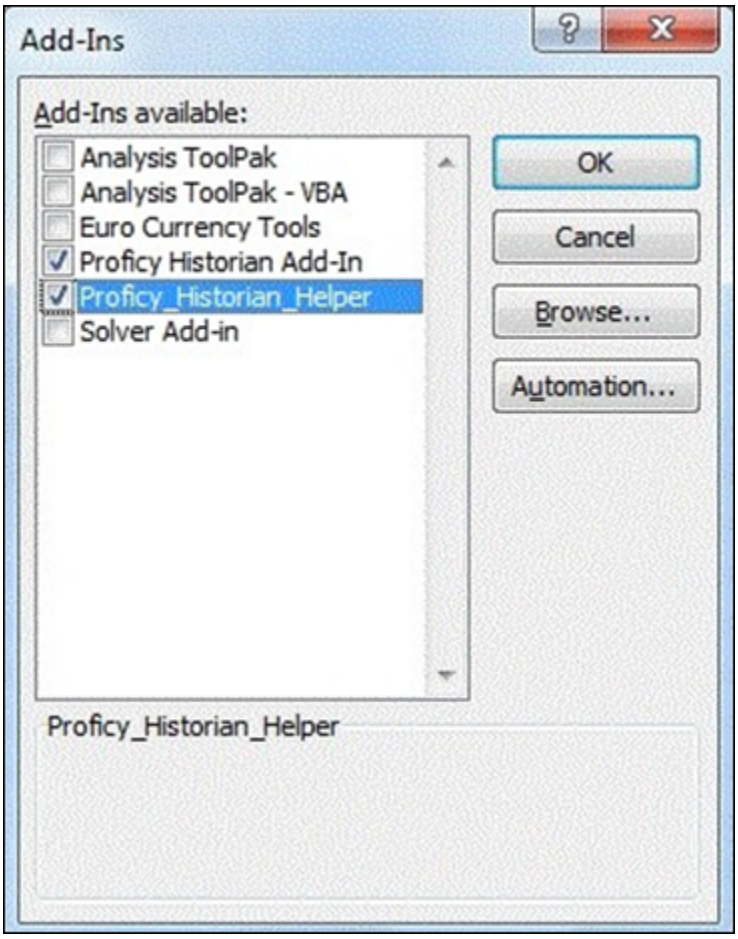
Activate Excel Add-In *(on page 197)*.

## Activate Excel Add-In

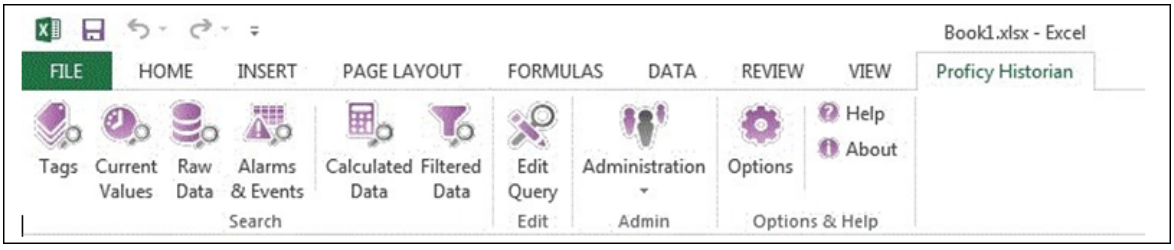
Install Excel Add-In *(on page 195)*.

1. Open a new Microsoft Excel worksheet.
2. Select **File > Options**.
   The **Excel Options** window appears.
3. Select **Add-Ins**.
4. In the **Manage** box, select **Excel Add-ins**, and then select **Go**.

The **Add-Ins** window appears.




5. Select the **Proficy Historian Add-In** and **Proficy_Historian_Helper** check boxes, and then select **OK**.
   If the **Proficy Historian Add-In** and **Proficy_Historian_Helper** check boxes do not appear, select **Browse** to locate the `Historian.xla` file for the check boxes to appear. This file is created if you have installed Microsoft Excel after installing Excel Add-In. By default, the `Historian.xla` file is located in the `C:\Program Files\Proficy\Historian or C:\Program Files (x86)\Proficy\Historian` folder.
   Excel Add-In is now ready to use and the **Proficy Historian** menu is now available in the Microsoft Excel toolbar.



# Querying Data

## Query Current Values

You can query the following types of data using the add-in:

- **Current values:** Retrieves the most recently updated value of one or more tags or process variables.

  > **Note:**
  > If you attempt to perform a query with two worksheets open, the add-in may become unstable and unresponsive. This is a known Microsoft Excel issue. To avoid this issue, work with only one Excel spreadsheet at a time.

- **Raw data:** Raw data values are the values actually stored in the archive, after applying collector and archive compression, but before applying any interpolation, smoothing, or other signal processing calculations. Querying raw data retrieves these values for a selected tag.

In addition, you can query filtered data *(on page 2732)* and calculated data *(on page 2734)*.

1. Open an Excel worksheet.
2. If you want to query current values, select **Historian > Query Current Value**. If you want to query raw data, select **Historian > Query Raw Data**.
   The **Historian Current Value Query** or the **Historian Raw Data Query** window appears.
3. Select the Historian server from the drop-down list box. If you do not specify a server, the default server is considered.

   > **Tip:**
   > To set the selected server as default, ensure that the **Set Server to Default** option is enabled.

4. Select a tag on your worksheet, and then place the cursor in the **Tag Name** field.
   Optionally, you can select the tag from the **Advance Tag Search** window. For more information, refer to Advanced Tag Search *(on page 2744)*.
   The tag name is automatically entered. You can also enter a tag name manually in the **Tag Name** field.
5. Enter values as described in the following table.

| Field | Description |
|---|---|
| **Query Type** | Select the type of data search:<br>◦ **By Time**: Using this option, you can search for data values between a start time and an end time. You can also use relative time entries to this field.<br>◦ **By Number Forward**: Using this option, you can search for a number of values after a specified time. Enter values into the **After Time** and **Number of Values** fields.<br>◦ **By Number Backward**: Using this option, you can search for a number of values before a specified time. Enter values in the **Values Before Time** and **Number of Values** fields. |
| **Query Criteria String** | Enter the query criteria along with the # symbol. For example, if the query criteria string is to retrieve only good data quality values, enter #ONLYGOOD. For more information, see Query Modifiers *(on page 2738)*. |
| **Output Display** | Select one or more parameters for the output. |
| **Output Range** | Select a range of cells in a single row or column to determine where the returned data is placed. |
| **Rows** or **Columns** | Select either **Columns** or **Rows** for the output display. Selecting **Columns** displays a table of values with parameters arranged in columns with header labels at the top. Selecting **Rows** rotates the table 90 degrees. |
| **Ascending** or **Descending** | Specify the order of the retrieved data. |

6. Select **OK**.
   The query returns a number of data points based on the number of rows or columns specified in the output range. If all the data points do not appear, select enough rows or columns to display all the data.

**Note:**
For an array tag, each element is displayed in separate rows with the tag name and index.

## Query Filtered Data

You can filter tag data based on a specific batch ID, lot number, or product code. You can also filter data that meets certain limits (for example, all the data points in which the temperature exceeds a certain value).

When querying filtered data, you can use a **Filter Expression** instead of **FilterTag**, **FilterMode**, and **FilterValue** parameters. You can use multiple filter conditions in the filter expression. For more information and examples on filter expression, refer to *Advanced Topics*.

**Note:**
Do not use the **Desc** option for the **Output Range** in the **Filtered Data Query** window. Using this option may cause the Excel Add-In to become unstable. If you use this option and find that Excel is unstable, try minimizing the Excel application window, expose the **Filtered Data Query** window, and close the window. Excel should then function normally.

1. Open an Excel worksheet.
2. Select **Historian > Query Filtered Data**.
   The **Historian Filtered Data Query** window appears.
3. Select the Historian server from the drop-down list box. If you do not specify a server, the default server is considered.

   **Tip:**
   To set the selected server as default, ensure that the **Set Server to Default** option is enabled.

4. Select a tag on your worksheet, and then place the cursor in the **Tag Name** field.
   If entering multiple tag names manually, separate each tag name with a colon. If your tag name has a colon within it, then select the tag names via cell references only.

   Do not use wildcards in this field. If you use a tag mask instead of a tagname, Historian only returns the first possible match.

   Optionally, you can select the tag from the **Advance Tag Search** window. For more information, refer to Advanced Tag Search *(on page 2744)*.
   The tag name is automatically entered. You can also enter a tag name manually in the **Tag Name** field.
5. Enter values as specified in the following table.

| Field | Description |
|---|---|
| **Query Time** | Enter the start time and end time for the query. You can also use relative time entries. |
| **Query Criteria String** | Enter the query criteria along with the # symbol. For example, if the query criteria string is to retrieve only good data quality values, enter #ONLYGOOD. For more information, see Query Modifiers *(on page 2738)*. |
| **Sampling Type** | Select the sampling type. For information, refer to Sampling Types *(on page 2781)*. |
| **Calculation Field** | Select a calculation algorithm. This field is enabled only if you select **Calculated Sampling** in the **Sampling Type** field. |
| **Sampling Interval** | Select one of the following options:<br>◦ **By Interval**: Using this option, you can query the data for a specific interval. option displays two entry fields, and . Enter values in both. For example, if you want to query the data for 10-minute intervals, enter 10 in the **Interval** field, and select **Minutes** in the **Time Unit** field.<br>◦ **By Samples**: Using this option, you can query the data for a specific number of samples. For example, to query 100 samples, enter 100 in the **Number of Samples** field. |
| **State Value** | Enter the state value. This field is enabled only if you selected **Calculated** in the **Sampling Type** field and if you selected **State Count** or **State Time** in the **Calculation Field** field. |
| **Output Display** | Select one or more parameters for the output. |
| **Filter Definition** | Enter the filter parameters in the available fields. |
| **Output Range** | Select a range of cells in a single row or column to determine where the returned data is placed. |
| **Rows** or **Columns** | Select either **Columns** or **Rows** for the output display. Selecting **Columns** displays a table of values with parameters arranged in columns with header labels at the top. Selecting **Rows** rotates the table 90 degrees. |
| **Ascending** or **Descending** | Specify the order of the retrieved data. |

6. Select **OK**.
   The query returns a number of data points based on the number of rows or columns specified in the output range. If all the data points do not appear, select enough rows or columns to display all the data.

   **Note:**
   - For an array tag, each element is displayed in separate rows with the tag name and index.
   - The **TagStats Calculation** mode is not supported.

## Querying Calculated Data

You can query data that is the result of performing calculations on raw data.

**Note:**
If you attempt to perform a query with two worksheets open, the add-in may become unstable and unresponsive. This is a known Microsoft Excel issue. To avoid this issue, work with only one Excel spreadsheet at a time.

1. Open an Excel worksheet.
2. Select **Historian > Query Calculated Value**.
   The **Historian Calculated Query** window appears.
3. Select the Historian server from the drop-down list box. If you do not specify a server, the default server is considered.

   **Tip:**
   To set the selected server as default, ensure that the **Set Server to Default** option is enabled.

4. Select a tag on your worksheet, and then place the cursor in the **Tag Name** field.
   Optionally, you can select the tag from the **Advance Tag Search** window. For more information, refer to Advanced Tag Search *(on page 2744)*.
   The tag name is automatically entered. You can also enter a tag name manually in the **Tag Name** field.
5. Enter values as described in the following table.

| Field | Description |
|---|---|
| **Query Time** | Enter the start time and end time for the query. You can also use relative time entries. |
| **Query Criteria String** | Enter the query criteria along with the # symbol. For example, if the query criteria string is to retrieve only good data quality values, enter #ONLYGOOD. For more information, see Query Modifiers *(on page 2738)*. |
| **Sampling Type** | Select the sampling type. For information, refer to Sampling Types *(on page 2781)*. |
| **Calculation** | Select a calculation algorithm. This field is enabled only if you select **Calculated Sampling** in the **Sampling Type** field. |
| **Sampling Interval** | Select one of the following options:<br>◦ **By Interval**: Using this option, you can query the data for a specific interval. option displays two entry fields, and . Enter values in both. For example, if you want to query the data for 10-minute intervals, enter 10 in the **Interval** field, and select **Minutes** in the **Time Unit** field.<br>◦ **By Samples**: Using this option, you can query the data for a specific number of samples. For example, to query 100 samples, enter 100 in the **Number of Samples** field. |
| **State Value** | Enter the state value. This field is enabled only if you selected **Calculated** in the **Sampling Type** field and if you selected **State Count** or **State Time** in the **Calculation Field** field. |
| **Output Display** | Select one or more parameters for the output. |
| **Output Range** | Select a range of cells in a single row or column to determine where the returned data is placed. |
| **Rows** or **Columns** | Select either **Columns** or **Rows** for the output display. Selecting **Columns** displays a table of values with parameters arranged in columns with header labels at the top. Selecting **Rows** rotates the table 90 degrees. |
| **Ascending** or **Descending** | Specify the order of the retrieved data. |

6. Select **OK**.

The query returns a number of data points based on the number of rows or columns specified in the output range. If all the data points do not appear, select enough rows or columns to display all the data.

**Note:**

- For an array tag, each element is displayed in separate rows with the tag name and index.
- The **TagStats Calculation** mode is not supported.

## Query Alarms and Events Data Using the Excel Add-In

Querying Alarms and Events data in the Excel Add-In retrieves alarms and events data according to your Query Criteria. Three query types are available: Alarm, Alarm history, and Events.

**Note:**
You cannot use an OPC alarms and events server with FIX32 SCADA systems. To collect alarm data from these systems, you can use a proxy for alarms through an iFIX node with an OPC alarms and events server. The OPC Classic Alarms and Events collector of Proficy Historian can then collect the data. For more information, refer to Using the OPC AE Collector with FIX32 SCADA Systems *(on page 1225)*.

1. Open an Excel worksheet.
2. Select **Historian > Query Alarms & Events**.
   The **Query Alarms & Events** window appears.
3. Enter values as described in the following table.

| Field | Description |
|---|---|
| **Server** | Select the SQL server that contains the alarms and events data. |
| **Query Type** | Select one of the following values:<br>◦ **Alarms**: In Historian, an alarm's entire life cycle is stored as a single record in the alarm archive. Thus, when retrieving from the archive, the entire life cycle of an alarm will be returned in a single record.<br>◦ **Alarm History**: Each change in the alarm's state will be returned in a single record.<br>◦ **Events**: One row per event is returned. |

| Field | Description |
|---|---|
| **Query Criteria** | Enter the criteria that you want to use in the query. For example, you may want to include alarms where the Alarm ID is equal to a specific Alarm ID occurring after a specific start time. In addition, you can specify which attributes must be displayed and how the results must be sorted in the spreadsheet. |
| **Output Range** | Select a range of cells in a single row or column to determine where the returned data must be placed. |
| **Output Orientation** | Select either **Columns** or **Rows** for the output display. Selecting **Columns** displays a table of values with parameters arranged in columns with header labels at the top. Selecting **Rows** rotates the table 90 degrees. |
| **Maximum Results** | Enter the maximum number of results for the query to return.<br><br>**Note:**<br>An Excel spreadsheet can display up to 255 columns and 32,767 rows. |
| **Output Display** | Specify the attributes that you want to display in the spreadsheet. |
| **Output Sorting** | Select the parameter using which you want to sort the results:<br>◦ **Alarm Time**: Sorts the data by the start time. The results appear in the reverse chronological order.<br>◦ **Custom Sort**: Allows you to select the field using which you want to sort the results.<br>◦ **None**: The results are not sorted at all. They are returned in the order they are received from the alarms and events database. |

4. Select **OK**.
   The query results appear.

## Modify a Query

You can change query parameters such as tag name, start time, end time, and so on. You cannot, however, narrow down the output range. For example, you cannot reduce the number in the

**NumberOfSamples** field, or you cannot change the **Output Orientation** to values that result in fewer rows or columns.

1. Open an Excel worksheet.
2. Access the query that you want to modify.
3. In the **Add-In** drop-down list box, select **Edit Query** or icon. Or you can double-click any cell that has the query formula.
   The **Edit Query** window appears.
4. Modify the query, and then select **OK**.

## Query Modifiers

Query modifiers are used to retrieve data that has been stored in the archive. They are used along with sampling and calculation modes to get a specific set of data.

If you want to use a query modifier, when you create or modify a query, in the **Query Criteria String** field, enter #, and then enter the query modifier. For example, if you want to retrieve only good data quality values, enter #ONLYGOOD.

| Query Modifier | Results |
|---|---|
| `ONLYGOOD` | Excludes bad and uncertain data quality values from retrieval and calculations.<br><br>Although you can use this modifier with any sampling or calculation mode, it is most useful with raw and current value queries. All the calculation modes such as minimum or average exclude bad values by default, so this modifier is not required in those cases. |
| `INCLUD-EREPLACED` | Normally, when you query raw data, any values that have been replaced with a different value for the same timestamp are not returned.<br><br>The `INCLUDEREPLACED` modifier is used to specify that you want replaced values to be returned, in addition to the updated values. However, you cannot query only the replaced data and the retrievable values that have replaced the other values. You can query all currently visible data and get the data that has been replaced.<br><br>This modifier is only useful with the rawbytime or rawbynumber retrieval. Do not use it with any other sampling or calculation mode. |

| Query Modifier | Results |
|---|---|
| `INCLUDEDELETED` | Retrieves the value that was previously deleted. Data that has been deleted from the archiver is never actually removed but is marked as hidden. Use the `INCLUDEDELETED` modifier to retrieve the values that were deleted, in addition to the current values.<br><br>This modifier is only useful with the rawbytime or rawbynumber retrieval. Do not use it with any other sampling or calculation mode. |
| `ONLYIFCONNECTED` and `ONLYIFUPTODATE` | Retrieve bad data if the collector is not currently connected and sending data to the archiver. You can use these modifiers with any sampling or calculation mode.<br><br>The bad data is not stored in the IHA file but is only returned in the query. If the collector reconnects and flushes data and you run the query again, the actual stored data is returned in the following situations:<br><br>• Collector loses connection to the archiver<br>• Collector crashes<br>• Collector compression is used and no value exceeds the deadband |
| `ONLYRAW` | Retrieves only the raw samples. It does not add interpolated or lab sampled values at the beginning of each interval during calculated retrieval such as average, minimum, or maximum.<br><br>Normally, a data query for minimum value will interpolate a value at the start of each interval and use that together with any raw samples to determine the minimum value in the interval. Interpolation is necessary because some intervals may not have any raw samples stored.<br><br>Use this query modifier with calculation modes only, not with raw or sampled retrieval like interpolated modes. |
| `LABSAMPLING` | Affects the calculation modes that interpolate a value at the start of each interval. Instead of using interpolation, lab sampling is used. When querying highly compressed data, you may have intervals with no raw samples stored.<br><br>For example, an average from 2 pm to 6 pm on a one-hour interval will interpolate a value at 2 pm, 3 pm, 4 pm, and 5 pm, and uses those in addition to any stored samples to compute averages. When you specify `LABSAMPLING`, the lab sampling mode is used instead of the interpolated sampling mode to determine these hourly values. A lab sampled average is used when querying a tag that never ramps up but changes in a step pattern such as a state value or a set point. Use this query modifier with calculation modes only, not raw or sampled retrieval like interpolated modes. |

| Query Modifier | Results |
|---|---|
| `ENUM-NATIVE-VALUE` | Retrieves the native, numeric values such as 1 or 2 instead of string values such as on/off for the data that has enumerated states associated with it.<br><br>You can use this modifier with any sampling or calculation mode. |
| `INCLUDE-BAD` | Normally, when you query calculated data from Historian, only good data quality raw samples are considered. `INCLUDEBAD` modifier includes bad data quality values in calculations.<br><br>You can use this modifier with any sampling or calculation mode. |
| `FILTERIN-CLUDEBAD` | Normally, while filtering, we use only good data quality values. When we use `FILTERINCLUDE-BAD`, the bad data quality values are considered when filtering to determine time ranges. This modifier is not always recommended. |
| `USE-MASTER-FIELDTIME` | Returns the value of all the fields at the same timestamp of the master field time, in each interval returned. It is used only for the multi-field tags. |
| `HON-ORENDTIME` | Normally, a query keeps searching through archives until the required number of samples has been located, or until it gets to the first or last archive. However, there are cases where you would want to specify a time limit as well. For example, you may want to output the returned data for a `RawByNumber` query in a trend page, in which case there is no need to return data that would be off page.<br><br>If you want to specify a time limit, provide an end time in your `RawByNumber` query and include the `HONORENDTIME` query modifier. Since `RawByNumber` has direction (backward or forward), the end time must be older than the start time for a backward direction or later than the start time for a forward direction. Use this query modifier only with the `RawByNumber` sampling mode. |
| `EXAMINE-FEW` | Queries using calculation modes normally loop through every raw sample, between the given start time and end time, to compute the calculated values.<br><br>When using `FirstRawValue`, `FirstRawTime`, `LastRawValue`, and `LastRawTime` calculation modes, we can use only the raw sample near each interval boundary and achieve the same result. The `EXAMINEFEW` query modifier enables this. If you are using one of these calculation modes, you may experience better read performance using the `EXAMINEFEW` query modifier.<br><br>Using this query modifier is recommended when: |

| Query Modifier | Results |
|---|---|
| | • The time interval is great than 1 minute.<br>• The collection interval is greater than 1 second.<br>• The data node size is greater than the default 1400 bytes.<br>• The data type of the tags is String or Blob. Query performance varies depending on all of the above factors.<br><br>Use this query modifier only with `FirstRawValue`, `FirstRawTime`, `LastRawValue`, and `LastRawTime` calculation modes. |

## Access Archive Statistics

You can access a list of selected statistics about an archive file. You can specify the server, the archive file name, and the type of information you want to access (such as start time, end time, file name, target file size, current file size, current or read-only status, last backup time, and last backup users). You can also specify a range of cells for the display.

1. Open an Excel worksheet.
2. Select **Historian > Administration > List Archives**.
   The **Historian Archive List** window appears.
3. Select a server from the drop-down list. If you do not specify a server, the default server is considered.
4. Enter values as described in the following table.

| Field | Description |
|---|---|
| **Archive Name** | Enter a archive name. Do not use wildcards in this field.<br><br>**Tip:**<br>To return details for more than one item, specify a substring in the **Archive Name** field that exists in each archive you want listed. For example, if you have archive files named from `Hero5_-Archive001` to `Hero5_Archive010`, enter |

| Field | Description |
|---|---|
| | `Hero5_Archive` to return the details for all those archives. |
| **Output Display** | Select one or more parameters for the output display. |
| **Output Range** | Select a range of cells in a single row or column to determine where the returned data is placed. |

5. Select **Asc** or **Desc** to sort the archives in ascending or descending order.
6. Select either **Columns** or **Rows** for the output display.

> **Note:**
> When selecting multiple tags, the orientation of the return data is based on the orientation of the selected tags and the Row/Col selection is ignored.

7. Select **OK**.
   The statistics of the selected archives appear.

## Access Collector Statistics

You can access a list of selected statistics of a collector instance. You can specify the server, the collector instance, and the type of information you want to access. You can also specify the range of cells for the display.

1. Open an Excel worksheet.
2. Select **Historian > Administration > List Collectors**.
   The **Historian Collector List** window appears.
3. Select a server from the drop-down list. If you do not specify a server, the default server is considered.
4. Enter values as described in the following table.

| Field | Description |
|---|---|
| **Collector Name** | Enter a collector instance name. Do not use wildcards in this field. |

| Field | Description |
|---|---|
| | **Tip:** To return details for more than one item, specify a substring in the **Collector Name** field that exists in each collector you want listed. For example, if you have collectors named from `Hero5_Collector0001` to `Hero5_Collector010`, enter `Hero5_Collector` to return the details for all those collectors. |
| **Output Display** | Select one or more parameters for the output display. |
| **Output Range** | Select a range of cells in a single row or column to determine where the returned data is placed. |

5. Select either **Columns** or **Rows** for the output display.

> **Note:**
> When selecting multiple tags, the orientation of the return data is based on the orientation of the selected tags and the Row/Col selection is ignored.

6. Select **OK**.
   The statistics of the selected collector instances appear.

# Managing Tags

## Search for a Tag (Basic)

You can search for tags and perform actions on them.

This topic describes how to perform a basic search of tags. You can also perform an advanced search *(on page 2744)*.

1. Open an Excel worksheet.
2. Select **Historian > Search Tags**.
   The **Historian Tag Search** window appears.

3. In the **Server** field, select a server from the drop-down list. If you do not specify a server, the default server is considered.
4. In the **Tag Mask**, enter a wildcard character to search for tags (for example, *).
5. Select **Search**.
   The **Historian Tag Search** window is populated with a tag list.
6. Move tags from the left section to the right section to add them to the search query.
7. Use the **Search Display** section to choose whether you want to display tag names or tag description. It also displays the number of tags returned.
8. Use the **Output With** to choose whether the output shows the names of the selected tags or the cell computation formulas.
   You can use the **Output with Formula** to place a dynamic formula in the worksheet instead of just copying the selected tag names. When you do so, the list of tags returned are dynamic based on the tag mask criteria. This is useful when selecting a cell reference for the tag mask as opposed to typing in a tag mask directly in the window.
9. Use the **Output Range** field to determine where in the worksheet the output data must appear.
10. Use the **Output Display** section to select the type of data to be displayed.
11. Select **OK** to apply your choices and initiate the query.
    A list of tags appears based on your search criteria.

## Search for a Tag (Advanced)

You can search for tags and perform actions on them.

This topic describes how to perform a advanced search of tags. You can also perform a basic search *(on page 2743)*.

When you perform an advanced search, the most recently used search criteria are saved in a file named `DefaultSearchCriteria.xml` in `c:\user- s\<username>\AppData`. These criteria are automatically loaded into the window the next time you access the Excel worksheet. You can reuse or modify the criteria rather than entering them each time. If you want to reset your criteria, delete the XML file.

While performing an advanced search, you can:

- Add multiple search criteria.
- Modify the existing criteria.
- Delete the unwanted search criteria from the list.
- Save the criteria to a file and reuse it.
- View the details of a tag in the search results.

1. Open an Excel worksheet.
2. Select **Historian > Search Tags > Advanced Tag Search**.
3. In the **Tag Criteria** field, specify one or more tag criteria *(on page 2784)*.
4. Provide values in the **Tag Criteria Value** field.
5. Select **Add Criteria**.
   The criteria are listed in the **Search Criteria** section.
6. Select **Search**.
   All tags that satisfy the query criteria are displayed in the **Available** section.
7. Move tags from the **Available List** section.
8. To modify the **Tag Criteria Value** already entered:
   a. Double-click the criteria from the list.
   b. Change the **Tag Criteria Value**.
   c. Select **Update Criteria**. The criteria value is updated with the new value.
9. To delete the search criteria from the list, select the criteria from the list, and then select **Delete**.
10. To save a search criteria list to be reused:

    a. Create your search criteria list.

    b. Select **Save**.
       The **Save As** window appears.

    c. Enter the file name, and select **Save**.
       Your criteria list is saved.

11. To load an existing criteria list:

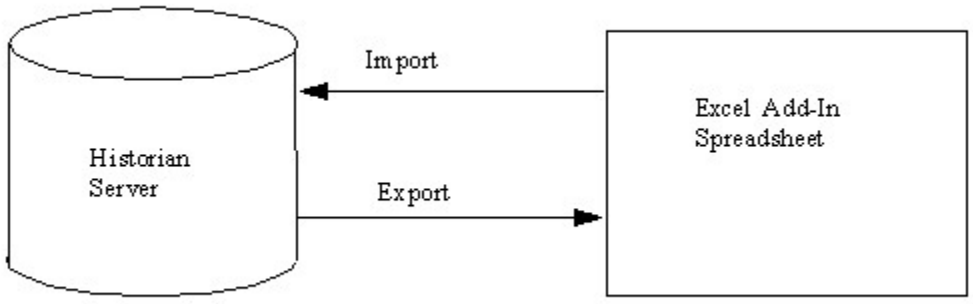
    a. Select **Load**.
       The **Open** window appears.

    b. Choose the XML file you saved earlier, and then select **Open**.
       The criteria list is loaded to the **Advanced Tag Search** window.

12. To view the tag attributes, double-click the tag from the available section or from the selected section.
    The **Tag Attributes** window appears with the attribute details.
13. Select **OK**.

## Export Tags

You can export tags from a Historian server into an Excel worksheet or to another system (either local or remote). After you export tags into an Excel worksheet, you can add/modify tags *(on page 2747)* in bulk, and then import them *(on page 2748)*.




**Note:**
You cannot enter more than 32,767 characters in a single cell in an Excel worksheet.

1. Open an Excel worksheet.
2. Select **Historian > Administration > Export Tags**.
   The **Export Tags from Historian** window appears.
3. Select a server from the drop-down list. If you do not select a server, the add-in uses the default server.
4. Enter values as described in the following table.

| Field | Description |
|---|---|
| **Filter Criteria** | Enter the name or description of the tag you want to export. You can use a tag mask to select a group of tags. To select a tag, use cell references instead of manually typing them.<br>**Note:**<br>You cannot export multiple tags when tagnames are read from multiple cells. If you specify a range of tagnames to read from multiple cells in the **Tag** |

| Field | Description |
|---|---|
| | **Mask** or **Tag Name(s)** fields, only the first tag in the range will be exported. |
| **Collector** | Enter the collector name. |
| **Data Type** | Enter the data type. |

5. Select one or more field names from the list in the right hand window. Always include tag names in the list of fields to export.
6. In the **Export Options** section, specify whether you want to export tags into a new Excel worksheet, a CSV file, or an XML file. If you select CSV or XML, you must also enter a path and file name for the destination file.
7. Select **OK**.
   The data is exported.

## Add/Modify Tags

Using the add-in, you can add tags to Historian or modify existing tags. To do so, include the tags in an Excel worksheet either automatically or manually, and then import them in bulk into Historian.

This can be a very convenient mechanism when you are working with large numbers of tags. If any conflicting names or parameters occur, an error occurs; you can then resolve the conflict and try again.

1. Create a tags worksheet in Excel either manually or automatically (using macros or any other tools).
   Since Historian requires information about each tag that varies with the type of the tag, ensure that you have included all the required information in the worksheet before attempting to import it into Historian. To determine what specific tag information is required, refer to the documentation provided with your SCADA application.
2. Import the tags into Historian *(on page 2748)*.

**Note:**
If any errors on the import occur, a window appears, specifying the issues encountered during the import. If an error occurs with any line of the import, the whole import is aborted.

## Import Tags

In an Excel worksheet, add/modify the tags that you want to import *(on page 2747)*.

Using the add-in, you can add tags to Historian or modify existing tags. To do so, include the tags in an Excel worksheet either automatically or manually, and then import them in bulk into Historian (either local or remote).

For example, with the Excel Add-In you can successfully import unsolicited tags without a calculation dependency (trigger), which you cannot do using Historian Administrator. Similarly, you can import circular references, which you cannot do using Historian Administrator.

**Note:**
Do not add or update the following spare configurations as the data may get corrupted or overwritten:

- The **Spare 1** field for OSI PI Distributor. OSI PI distributor reads data from the Historian tag displayed in the **Tag Source Address** field and sends it to the OSI PI tag name displayed in the **Spare 1** field.
- The **Spare 5** field for the Server-to-Server collector and the Server-to-Server distributor because it is only used for internal purposes.

1. Open an Excel worksheet.
2. Select **Historian > Administration > Import Tags**.
   A message appears.
3. Select **Yes** to initiate the operation.
   A message appears, confirming that the import is complete.
4. Select **OK**.
   If errors occur, a window appears detailing the issues encountered during the import. If an error occurs with any line of the import, the whole import operation is aborted.
   f you export all the fields and attempt to import the read-only fields **LastModified** and **LastModifiedUser**, you may receive the following error message: Import failed, Error with Import Header. To avoid this issue, export the tags without selecting the read-only fields, and then import the tags.

## Rename Tags

To rename a tag, you must be a member of the administrator's group with tag-level security.

When you rename a tag, you can choose between the following options:

- **Rename using an alias:** In this case, the old name is called the tag alias. You can retrieve tag data using the tag alias as well. When you copy a tag, the tag alias is captured as well to aid in an audit trail.
- **Rename permanently:** In this case, the old name is no longer captured. Therefore, you can create another tag with this old name. You cannot store and forward data using the old name. This implies that data for the tag is collected separately for the new name.

1. Export the tags *(on page 2746)* that you want to rename.

   > **Important:**
   > You must only include tag name in the list of fields to export.

2. In the Excel worksheet, to the right of the **Tagname** column, insert a column named New Tagname.
3. For each tag that you want to rename, enter the new name in the **New Tagname** column.

   > **Important:**
   > You must specify a tag name in all the rows of the **New Tagname** column. If you do not want to rename any of those exported tags, you must delete that row.

4. If you want to rename the tags permanently, to the right of the **New Tagname** column, insert a column named **Permanent Rename**.
5. For each tag that you want to rename permanently, enter `TRUE` in the **Permanent Rename** column. For the remaining tags, enter `FALSE`.
6. Select **Historian > Administration > Rename Tags**.
   A message appears, asking you to confirm that you want to rename the tags.
7. Select **Yes**.
   The tags are renamed.

## Working with Array Tags

Historian allows you to store a set of values with a single timestamp and then read the elements back individually or as an array tag. In Historian, we can modify a tag to an array tag by specifying the `NumberOfElements` as `-1`. Where `NumberOfElements` indicates the tag is an array tag. If the `NumberOfElements` is `-1`, then the tag is an array tag.

Tags with zero `NumberOfElements` are not array tags. Since the size of the array is dynamic there is no single number of elements that can be returned. In Excel Add-in each element of the array tag is displayed in separate rows with tagname with index (`Tag-name[]`) and values. You can perform all operations that you use for a tag on an Array tag. You can export, import and query an array tag or an array element.

**Note:**

- Array tags do not support Enumerated set.
- **TagStats Calculation Mode** is not supported.

# Importing and Exporting Data

## Import Tag Data

The **Import Data** command is the converse of the **Export Data** command. It moves selected information from your current worksheet into the specified Server in the same way the **Import Tags** command functions.

**Note:**
If you use the **Active Hours** setting while importing data using the Excel Add-In, note that if the first tags imported are not within the **Active Hours** settings, no subsequent tags will be returned on that import (even if they are within the set active hours).

1. Select **Administration** and then select **Import Data** from the **Historian** menu.
   A message box appears.
2. Select **Yes** to initiate the operation. If successful, a window appears confirming the completion of the import function.
   Select **OK** to close the window. If errors occur on the import, a window appears detailing the issues encountered in the import. If an error occurs in any line of the import, the whole import is aborted.

## Import Alarms and Events Data into a Spreadsheet

Using the Historian Excel add-in, you can import alarms and events data into a spreadsheet. It helps you include alarms and events data into the Historian archive that is not normally collected by Historian or when you are migrating data from an older system into Historian.

**Note:**

- The Excel worksheet must contain at least the source and timestamp columns.
- If an error occurs while importing data, the whole operation is aborted.

1. Open an Excel worksheet.
2. Select **Historian > Administration > Import Alarms**.
   The alarms and events data is imported.

## Export Tag Data

The Export Data function allows you to move values from the Historian Server to your Excel worksheet or to another system in the same way you move tag information with Export Tags.

**Note:**
Before importing or exporting tags, data, or messages, you should be aware of a convention used with the Historian application. The Server is the reference point for all import and export functions. If you want to move tag information from the Server into your worksheet, you must use the **Export Tags** command. Conversely, if you want to move data from your worksheet to the server, you must use the **Import Data** command.

1. Open an Excel worksheet.
2. Select **Historian > Administration > Export Raw Data**.
   The **Export Data from Historian** window appears.
3. If you want to specify a server, select a server from the drop down list. If you do not specify a server, the Add-In uses the default server.
4. Select a tag on your worksheet or enter the tag names manually.

   **Note:**
   If your tag name has a colon within it, then you should select the tag names via cell references only.

5. Optionally, you can select the tag name from the **Advance Tag Search** window.
   See Search for a Tag (Advanced) *(on page 2744)*
6. In the **Query Criteria String**, enter the query criteria along with the # symbol.

For example, if the query criteria string is to retrieve only good data quality values, then you should specify `#ONLYGOOD` as the **Query Criteria String**. See Query Modifiers *(on page 2738)*.

7. In the **Query Time** section enter values of time in the **Start Time** and **End Time** fields.
   You can also use relative time entries to this field. See Relative Time Entries *(on page 2777)*.
8. In the **Sampling Type** section, select a type from the drop-down list.
9. The **Calculation** field is active only after you select **Calculated Sampling** as the **Sample Type**.
   Select a **Calculation Algorithm** type from the drop-down list.
10. In the **Sampling Interval** section, select either the **By Interval** or **By Samples** option.
    The **By Interval** option displays two entry fields, **Interval** and **Time Unit**. Enter values in both. For example, to sample at 10 minute intervals, enter 10 in the interval field and select Minutes in the Time Unit field. The **By Samples** option displays a **Number of Samples** field.

    To specify a number of samples for the data query, enter a number in this field. For example, to query 100 samples, enter 100 in this field.
11. In the **Filter Definition** section, enter filter parameters in the fields for **Filter Tag**, **Filter Comparison**, **Include Date Where Value Is Equal To**, and **Include Times**.
    These fields are optional. If you do not enter any values, the query returns all values without filtering.
12. In the **Fields To Export** section, select one or more fields.
    To select multiple individual tags, press the **Control** key and select the tagnames. To select a sequence of tags, press the **Shift** key and select the first and last tagname of the sequence.
13. In the **Export Options** section, select one of three options:
    - **To New Worksheet**
    - **To CSV File** or
    - **To XML File**
14. If you select **To CSV File** or **To XML File**, you must enter a file name and path for the new file in the **File Name** field.
15. Select **OK** to initiate the export. Select **Cancel** to abort the operation and close the window.

## Export Alarms and Events Data from a Spreadsheet

Using the Historian Excel add-in, you can export alarms and events data into a new worksheet or into a CSV or an XML file.

1. Open an Excel worksheet.
2. Select **Historian > Administration > Export Alarms**.
   The **Historian Alarm Export** window appears.
3. Enter values as described in the following table.

| Field | Description |
|---|---|
| **Server** | Select the SQL server that contains the alarms and events data. |
| **Query Type** | Select one of the following values:<br>◦ **Alarms**: In Historian, an alarm's entire life cycle is stored as a single record in the alarm archive. Thus, when retrieving from the archive, the entire life cycle of an alarm will be returned in a single record.<br>◦ **Alarm History**: Each change in the alarm's state will be returned in a single record.<br>◦ **Events**: One row per event is returned. |
| **Query Criteria** | Enter the criteria that you want to use in the query. For example, you may want to include alarms where the Alarm ID is equal to a specific Alarm ID occurring after a specific start time. In addition, you can specify which attributes must be displayed and how the results must be sorted in the spreadsheet. |
| **Output Range** | Select a range of cells in a single row or column to determine where the returned data must be placed. |
| **Output Orientation** | Select either **Columns** or **Rows** for the output display. Selecting **Columns** displays a table of values with parameters arranged in columns with header labels at the top. Selecting **Rows** rotates the table 90 degrees. |
| **Maximum Results** | Enter the maximum number of results for the query to return.<br> **Note:**<br>An Excel spreadsheet can display up to 255 columns and 32,767 rows. |
| **Output Display** | Specify the attributes that you want to display in the spreadsheet. |
| **Output Sorting** | Select the parameter using which you want to sort the results: |

| Field | Description |
|---|---|
| | ◦ **Alarm Time**: Sorts the data by the start time. The results appear in the reverse chronological order.<br>◦ **Custom Sort**: Allows you to select the field using which you want to sort the results.<br>◦ **None**: The results are not sorted at all. They are returned in the order they are received from the alarms and events database. |
| **Export Options** | Select the format in to which you want to export the data:<br>◦ **To New Worksheet**: Exports the data to a new Excel worksheet.<br>◦ **To CSV File**: Exports the data with comma separated values to a new file.<br>◦ **To XML File**: Exports the data to a new XML file. |

4. Select **OK**.
   The data is exported.

# Working with Messages

## Search for Messages

You can search the archive for selected types of messages generated during a specific time period and to display selected fields from those messages. When you do so, a dynamic formula is placed in the worksheet, using which you can build a dynamic message report that you can build, save, and reuse.

1. Open an Excel worksheet.
2. Select **Historian > Administration > Search Messages**.
   The **Historian Message Search** window appears.
3. Select a server from the drop-down list. If you do not specify a server, the default server is used.
4. Enter values as described in the following table.

| Field | Description |
|---|---|
| **Topic** | Select one of the message types from the drop-down list. |
| **Query Times** | Enter values for the start time and end time. |

| Field | Description |
|---|---|
| **Search String** | Enter a search string for the text of messages. You need not enter * for wildcards. |
| **Output Display** | Select one or more parameters for the output display. Select a name to select it. |
| **Output Range** | Select a range of cells in a single row or column to determine where the returned data is placed. |
| **Asc** or **Desc** | Specify whether you want to sort the messages in ascending or descending order.<br><br>**Note:**<br>When selecting multiple tags, the orientation of the return data is based on the orientation of the selected tags and the Row/Col selection is ignored. |

5. Select **OK**.
   A list of messages appear based on the search criteria.

## Import Messages

You can import all the messages from an Excel worksheet into a Historian server. You can, however, only add messages to Historian; you cannot modify or remove them.

1. Open an Excel worksheet.
2. Select **Historian > Administration > Import Messages**.
   A message appears, asking you to confirm that you want to import messages.
3. Select **Yes** to execute the import. A window appears when the operation is complete. Select **OK** to close the window.
   A message appears, stating that the import is successful. If, however, an error occurs in any line of the import, the whole import is aborted.

## Export Messages

You can export messages from a Historian server to your worksheet, a CSV file, or an XML file.

You can specify which fields of the messages are exported, such as timestamp, topic, message string, message number, substitutions, or username.

1. Open an Excel worksheet.
2. Select **Historian > Administration > Export Messages**.
   The **Export Messages From Historian** window appears.
3. Select a server from the drop-down list box. If you do not specify a server, the default server is considered.
4. Enter values as described in the following table.

| Field | Description |
|---|---|
| **Topic** | Select one of six types of messages from the drop-down list box. |
| **Filter Criteria** | Enter values for the start time, end time, and search text string in the corresponding fields. |
| **Fields to Export** | Select one or more field names from the list that you want to export. |
| **Export Options** | Specify whether you want to export messages to a new worksheet, a CSV file, or an XML file.<br><br>For a CSV or an XML file, you must enter a file name and path for the new file in the **File Name** field. |

5. Select **OK**.
   The messages that meet the search criteria are exported.

# Managing Enumerated Sets

## Before You Begin

Export the enumerated sets into an Excel worksheet *(on page 2758)*.

Before importing enumerated sets into Historian from an Excel worksheet, you can perform the following actions:


- Add sets *(on page 2757)*
- Delete sets *(on page 2757)*
- Modify set description *(on page 2757)*
- Add states *(on page 2757)*
- Modify states *(on page 2758)*
- Delete states *(on page 2758)*

### To add sets:

1. Enter details in the following columns in the Excel worksheet.
    - **SetName**
    - **SetDescription**
    - **StateName**
    - **StateDescription**
    - **StateLowValue**
    - **StateHighValue**
    - **StateRawValueDataType**
    - **NumberOfStatesInThisSet**

    We recommend that you fill the columns in the aforementioned sequence.
2. Select **Import Enumerated Sets**.

### To delete sets:

1. Select the row that contains the state you want to delete.
2. Right-click the row, and then select **Delete**.

### To modify the description of a set:

Place the cursor in the **StateDescription** cell and modify the description.

You cannot modify the name of the set. If you change the name of the set, it is considered a new set.

### To add states:

1. Select the set to which you want to add a set.
2. Add the name of the set in the **SetName** column.

   The name match the set you selected.
3. Enter the details in the **SetDescription**, **StateName**, **StateDescription**, **StateLowValue**, **StateHighValue**, and **StateRawValueDataType** columns.
4. Enter the total number of states in the **NumberOfStatesInThisSet** column.

   Ensure that this value is the same for the current state and existing states in the set. For example, if a state has two states already and you are adding a third state, the number of states for all the three states must be changed to three.

   A new state is added to the set.

### To modify states:

1. Modify the values of each state.

   The states are modified.

### To delete states:

1. Select the row that contains the state you want to delete.
2. Right-click the row, and then select **Delete**.

   The state is deleted.

### What to do Next

Import the enumerated sets *(on page 2759)*.

## Export Enumerated Sets

You can export enumerated sets in bulk into an Excel worksheet, add/modify/delete them *(on page 2756)*, and then import them *(on page 2759)* into Historian.

1. Open an Excel worksheet.
2. Select **Historian > Administration > Export Enumerated Sets**.
   The **Historian Export Tags** window appears.
3. Select a server from the drop-down list box. If you do not select a server, the default server is considered.
4. Enter values as described in the following table.

| Field | Description |
|---|---|
| **Filter Criteria** | In the **EnumeratedSet Mask/EnumeratedSet Name** field, search for the sets set you want to export. You can enter a set name or name mask (or description and a description mask (*) criteria). All the sets matching the criteria appear. |
| **Export Options** | Specify whether you want to export the enumerated sets into an Excel worksheet, a CSV file, or an XML file. For a CSV or XML file, you must also enter a path and file name for the destination file. |

5. Select **OK**.
   A message appears, stating that the export is successful.
6. Select **OK**.

Add/modify/delete enumerated sets *(on page 2756)* as needed, and then import them *(on page 2759)* into Historian.

## Import Enumerated Sets

Export the enumerated sets *(on page 2758)* and add/modify/delete them *(on page 2756)* as needed.

You can create or modify enumerated sets in the Historian server by importing them from an Excel worksheet.

1. Open an Excel worksheet.
2. Select **Historian > Administration > Import Enumerated Sets**.
   A message appears, asking you to confirm that you want to import the enumerated sets.
3. Select **Yes**.
   A message appears, confirming that importing the enumerated sets is successful.
4. Select **OK** .

## Rename Enumerated Sets

1. Export the enumerated sets *(on page 2758)* that you want to rename.
2. In the exported worksheet, remove all the fields except **setname**.
3. Change **setname** to **EnumeratedSetName** in the header.
4. To the right of the **EnumeratedSetName** column, insert a column named **NewEnumeratedSetName**.
5. In the **NewEnumeratedSetName** column, enter a new name for each enumerated set that you want to rename.

   **Important:**
   If you do not want to rename any of the exported enumerated sets, delete that row from the spreadsheet and rename the remaining enumerated sets.

6. Select **Historian > Administration > Rename Enumerated Sets**.
   The **Proficy Historian Rename Enumerated Sets** window appears.
7. Select **Yes**.
   The enumerated sets are renamed.

# Managing User-Defined Types

Before importing a User Defined Type set into Historian from an Excel Worksheet, you can perform the following actions:


- Add a User Defined Type *(on page 2760)*
- Modify a User Defined Type *(on page 2760)*
- Add fields *(on page 2760)*
- Modify fields *(on page 2761)*
- Delete fields *(on page 2761)*


### To add User Defined Types:

1. Enter details into the columns in the excel worksheet.

   **Note:**
   The columns are listed here according to the way they appear in the Excel worksheet. However, it is recommended to fill the columns in the following sequence: UserDefinedTypeName, User- DefinedTypeDescription, FieldName, FieldDescription, FieldDatatype, IsMasterField, NumberOfFields, StoredFieldQualities and AdminSecurityGroup.

2. Select **Import UserDefinedTypes** to import the types.

### To modify the description of a User Defined Type:

1. Select in the **User Defined Type Description** cell and modify the description.

   **Note:**
   You cannot modify the name of the type. If you change the name of the type, it is considered as a new type.

2. Select the **Import User Defined Types** to import the types.

### To add fields:

1. Select the UserDefinedType to which you wish to add a field.
2. Add the name of the type in the UserDefinedTypeName column. The name should be same as the type selected by you.

3. Enter the details in the UserDefinedTypeDescription, FieldName, FieldDescription, FieldDatatype, IsMasterField, NumberOfFields, StoredFieldQualities and AdminSecurityGroup columns.
4. In the NumberOfFields column, enter the total number of fields. Ensure that this value is the same for the current field and existing fields in the user defined type.

   For example, if a UserDefinedType has two fields already and you are adding a third field, the number of fields for all the three fields should be changed to three. A new field is added to the UserDefinedType on import.

### To modify fields:

1. Select the field you wish to modify by selecting the row in the Excel worksheet.
2. Modify the values by selecting in the respective columns. The field/fields are modified on import.

### To delete fields:

1. Select the row that has the field you wish to delete.
2. Right-select the row and select Delete. Alternatively, you can also use the Delete key on your key-board.
3. Select the **Import UserDefinedTypes** to update your changes.

## Export User-Defined Types

You need to have appropriate security permissions to import and export a user defined type. For more information, refer to *Getting Started with Historian Guide* > *Implementing Historian Security* for the definition of the various security levels and groups.

You can export user-defined types in bulk into an Excel worksheet, add/modify/delete them *(on page 2760)*add/modify/delete them *(on page 2756)*, and then import them *(on page 2762)* into Historian.

1. Open an Excel worksheet.
2. Select **Historian > Administration > Export User Defined Types**.
   The **Historian Export User Defined Types** window appears.
3. Select a server from the drop-down list. If you do not select a server, the default server is considered.
4. Enter values as described in the following table.

| Field | Description |
|---|---|
| **Filter Criteria** | In the **UserDefinedType Mask/UserDefined-Type Name** field, search for the sets set you |

| Field | Description |
|---|---|
| | want to export. You can use * for a wildcard search. |
| **Export Options** | Specify whether you want to export the user-defined types to an Excel worksheet, a CSV file, or an XML file. For a CSV or an XML file, you must also enter a path and file name for the destination file. |

5. Select **OK**.
   The user-defined types are exported.

## Import User-Defined Types

Export the user-defined types *(on page 2761)* and add/modify/delete them *(on page 2760)*.

You can create or modify user-defined data types in an Excel worksheet and then import them into Historian.

1. Open an Excel worksheet.
2. Select **Historian > Administration > Import MultiField Source Addresses**.
   A message box appears asking you to confirm whether to import the sets.
   A message appears, asking you confirm that you want to import the user-defined types.
3. Select **Yes**.
   The user-defined types are imported.

# Reference

## Excel Add-In Options

| Field | Description |
|---|---|
| **Internal vs. External References** | Choosing **Use External References** allows your application to reference cells in other worksheets and workbooks in addition to the current one. If you choose **Use Internal References** instead, you can only access cells in the current worksheet. The default setting is **Use External References**. |
| **Automatically Update Links to Add- In (Yes/ No)** | Add-In functions are maintained as worksheet links. If users who share worksheets do not have Microsoft Office installed the same way, it is necessary to turn this feature on. When on, this feature automatically re-establishes |

| Field | Description |
|---|---|
| | any formula links that may be broken due to differences among users in Microsoft Office installation. The default setting enables this feature.<br><br>The Auto Update feature allows sharing of worksheets. You must, however, install the Excel Add-In in the exact same Microsoft Office Library Path as the other worksheets if you want to use the sharing feature.<br><br>When opening a worksheet with links to another worksheet, you may receive a message prompting you to update all linked information in the workbook (Yes) or keep the existing information (No). It is recommended that you select No and keep the existing information. The links will be automatically updated for your worksheet. Save your worksheet after the links have been updated. |
| **Show/Hide Header Labels** | This option lets you display or suppress the column header labels that are automatically placed in the worksheet when entering formulas throughout the Historian windows. The default setting is **Show Labels**. |
| **Color** | Allows you to select the header name color from the drop-down list: black, blue, red, green, magenta, cyan, or yellow. |
| **Assign Default Server** | This window shows the current server assignment. You can modify the setting by selecting the **Edit** button and accessing the **Historian Server Managers** window. This window allows you to save user connection information, add or connect to a new server, delete a server, and modify the default server. |
| **Adjust Column Widths** | This option lets you automatically adjust the width of columns in your worksheet as formulas are inserted by Historian windows. Select **Adjust Header Column Width** to modify the width of header labels; select **Adjust Data Column Width** to modify the data column widths to accommodate the data values. Enabling these options usually makes the worksheet much more readable. However, doing so can sometimes make the worksheet calculate too much when building a large report. In such cases, disable the automatic feature and adjust individual columns manually. |
| **Save/Default/Cancel** | These action buttons let you apply your choices of options. Select **Save** to apply the settings you entered, select **Default** to select default settings for all options, and select **Cancel** to close the window. |

## Reports

You can generate a wide range of custom reports. You can use all the standard, familiar Excel tools and techniques to access the Historian archives and build reports and charts of all types to fit your specific needs. You can use the sample reports included with Historian almost as is — just change the tags to fit your application. As an alternative, use the setup worksheets as a starting point and adapt them to your particular situation.

**Defining Reports**

You can define a report so that Excel recalculates the worksheet whenever the contents of specific cells, such as start times or dates, change. In this way, the report generates a dynamic snapshot of process performance, updated regularly in real time. You can also manually initiate recalculation at anytime.

**Building Dynamic Reports**

The primary rule to follow in building a dynamic report is to use formulas with cell references that contain variable information rather than fixed data, so that recalculation produces new data each time it occurs. You then initiate recalculation by changing certain inputs manually or automatically.

**Sharing Reports**

You can share any Excel reports you develop with the Historian Excel Add-In as you would any other Excel workbook. For each client using the worksheets, set up the Excel Add-In for Historian.

**Using the Sample Reports**

The Historian application includes three typical sample reports that demonstrate the power and ease-of-use of the Excel Add-In. Use them directly in your application or modify them to fit your requirements.

The three sample Excel reports are built using tags from the Simulation collector. You must create an instance of the Simulation collector and start it in order for these reports to work. The `Historian Batch Report Sample.xls` file also uses Batch ID and Product ID tags from the Simulation collector. These are Simulation Collector points that are configured to store string data types.

To ensure that the sample reports work correctly, you must add the string tags. These are the last five tags in the tag collector list. Add the string tags in Historian Administrator by browsing the Simulation collector and adding all of the tags by selecting the **Add All Tags** check box. Alternatively, you can run the `Add Tags to Simulation Collector.bat` batch file in the `Historian\Server` directory of the machine that has the Simulation collector.

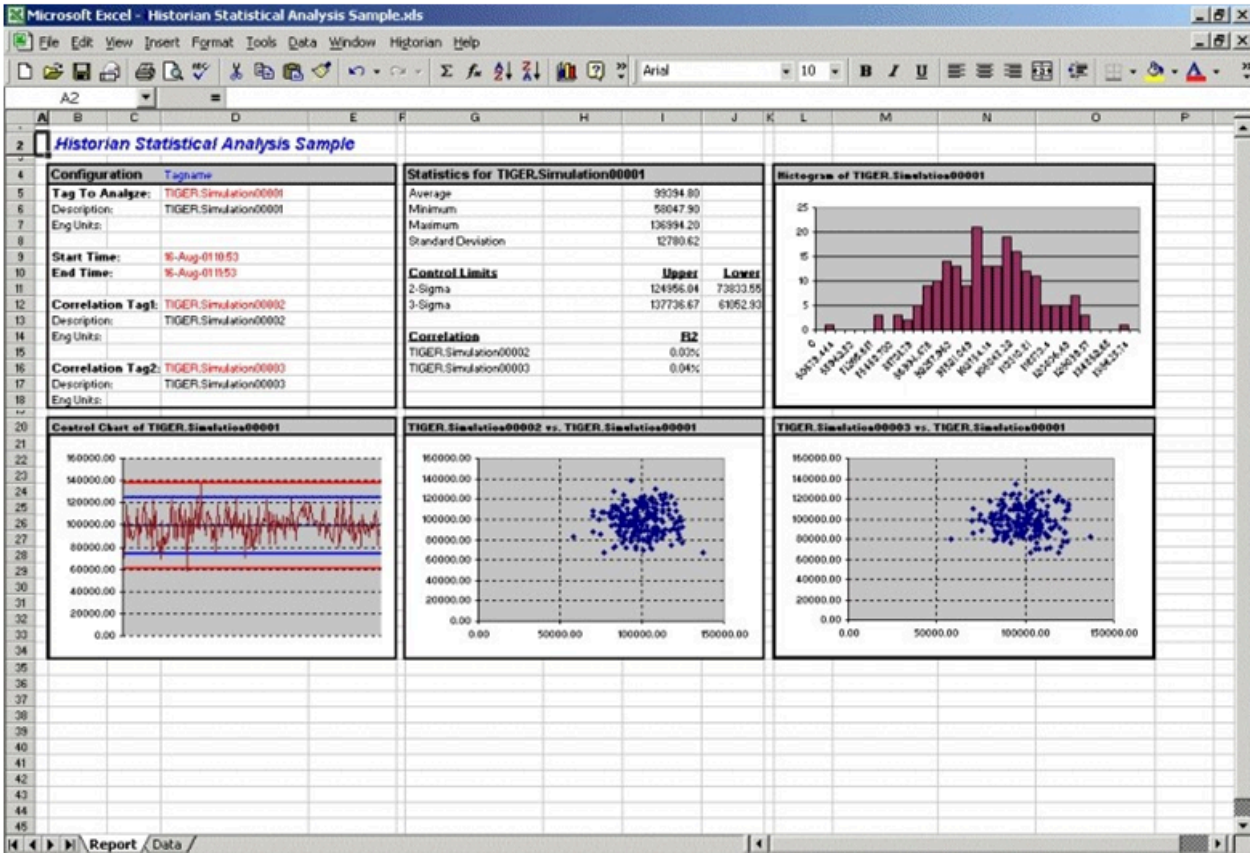
In addition, when you create an instance of the Simulation collector, it prompts you for the number of simulation tags it should create (but you must still add the tags for collection using one of the two methods above). The default is 1000. Do not enter a value less than 30.

When opening a sample Excel report, you may receive a message prompting you to update all linked information in the workbook or keep the existing information. It is recommended that you select No (that is, keep the existing information). The links will be automatically updated for your worksheet. Save your worksheet after the links have been updated.

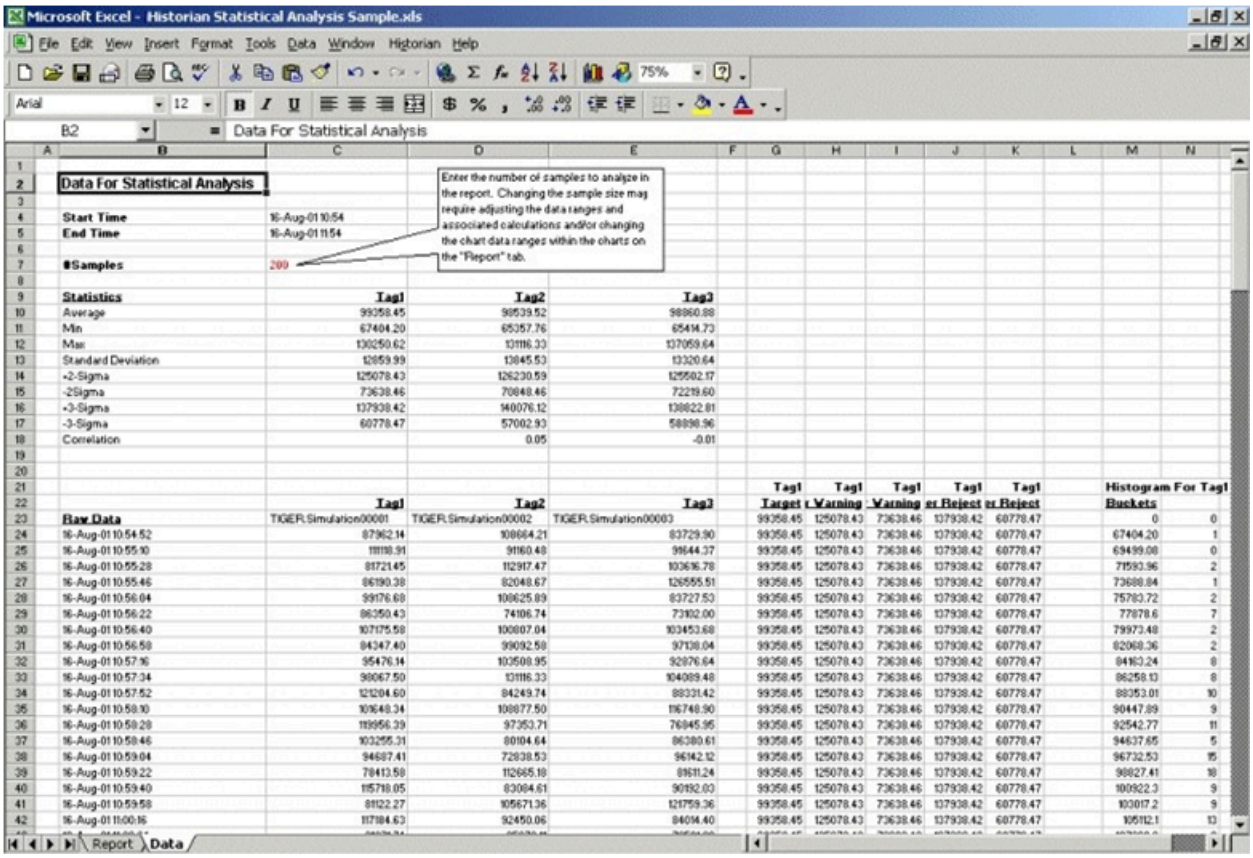
**Historian Statistical Analysis Sample Report**

For a specific duration, this report calculates a number of statistical properties of a tag, such as the average, maximum, minimum, standard deviation, 2 sigma and 3 sigma control limits, and correlation coefficients for other tags. It displays charts of various types for several of these variables.

The chart at the lower left is a plot of the main variable vs. time with sigma control limits indicated by the straight lines. The two charts to the right are scatter diagrams that show the correlation between the main variable and two other variables. The chart at the top right is a histogram of data values of the main variable that shows how the data points are distributed.

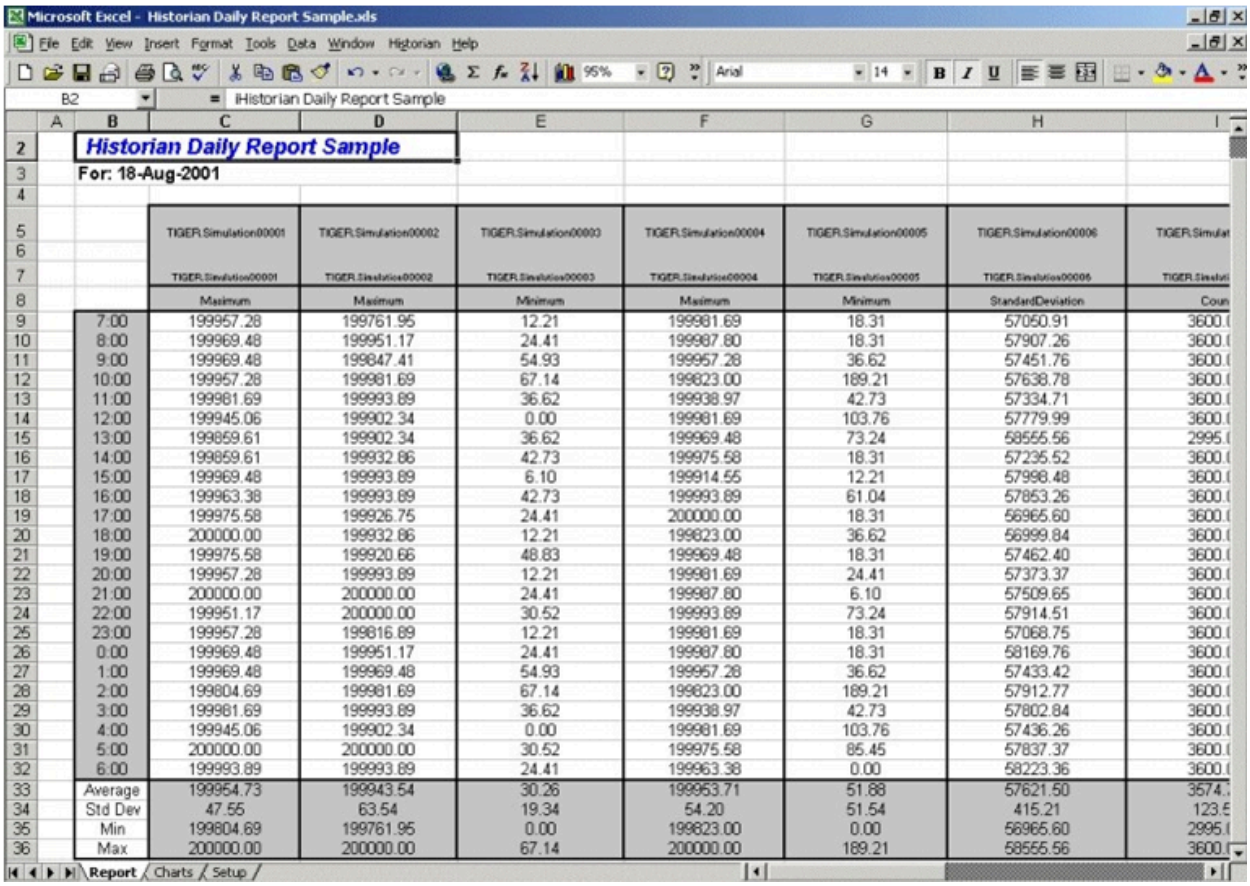


The following figure shows the worksheet associated with the sample report that contains the data used to generate the report.




**Daily Performance Sample Report**

This sample report shows how the measured values and selected statistical properties of specified tags have varied in the last 24 hours. This sample is an example of a typical daily performance report in an industrial plant.



Microsoft Excel - Historian Daily Report Sample.xls

File Edit View Insert Format Tools Data Window Historian Help

B2 = Historian Daily Report Sample

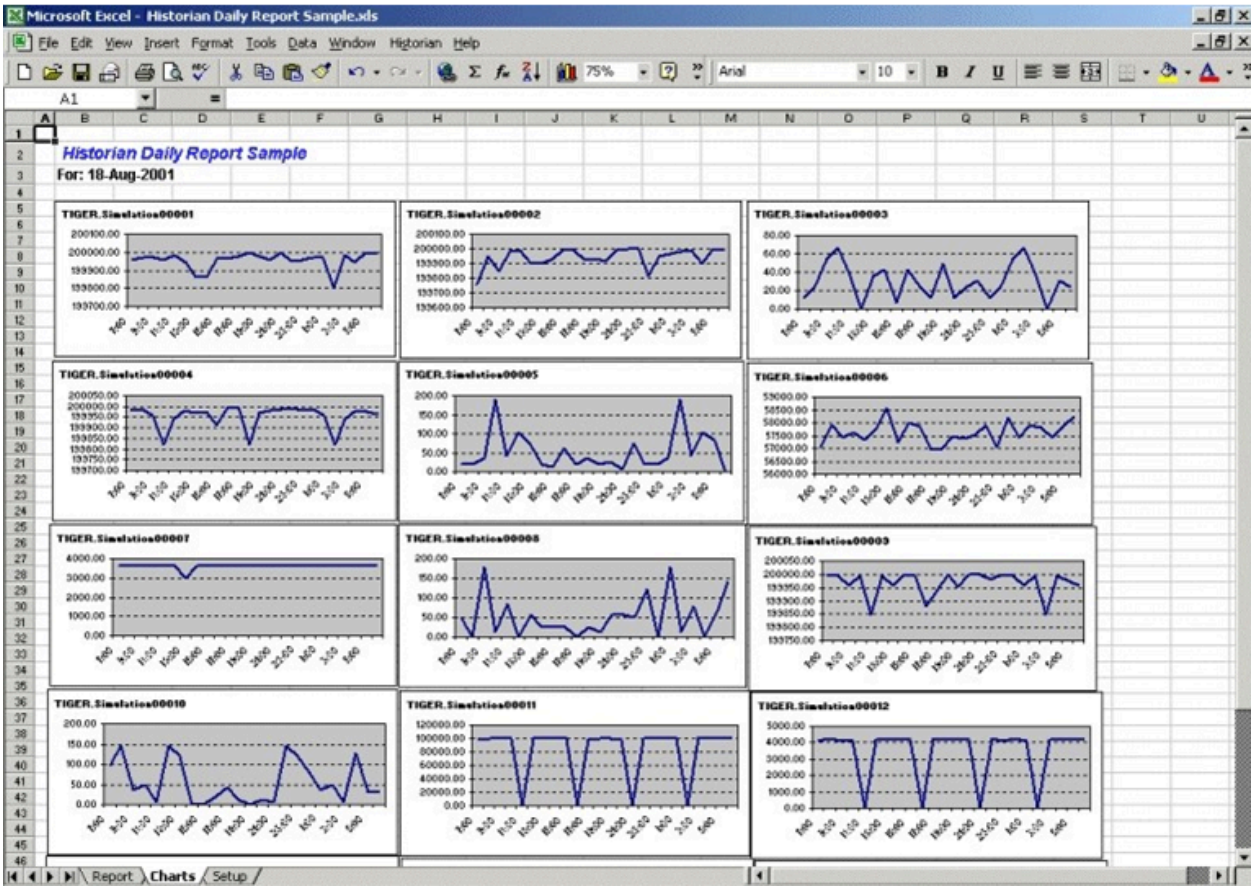
**Historian Daily Report Sample**

For: 18-Aug-2001

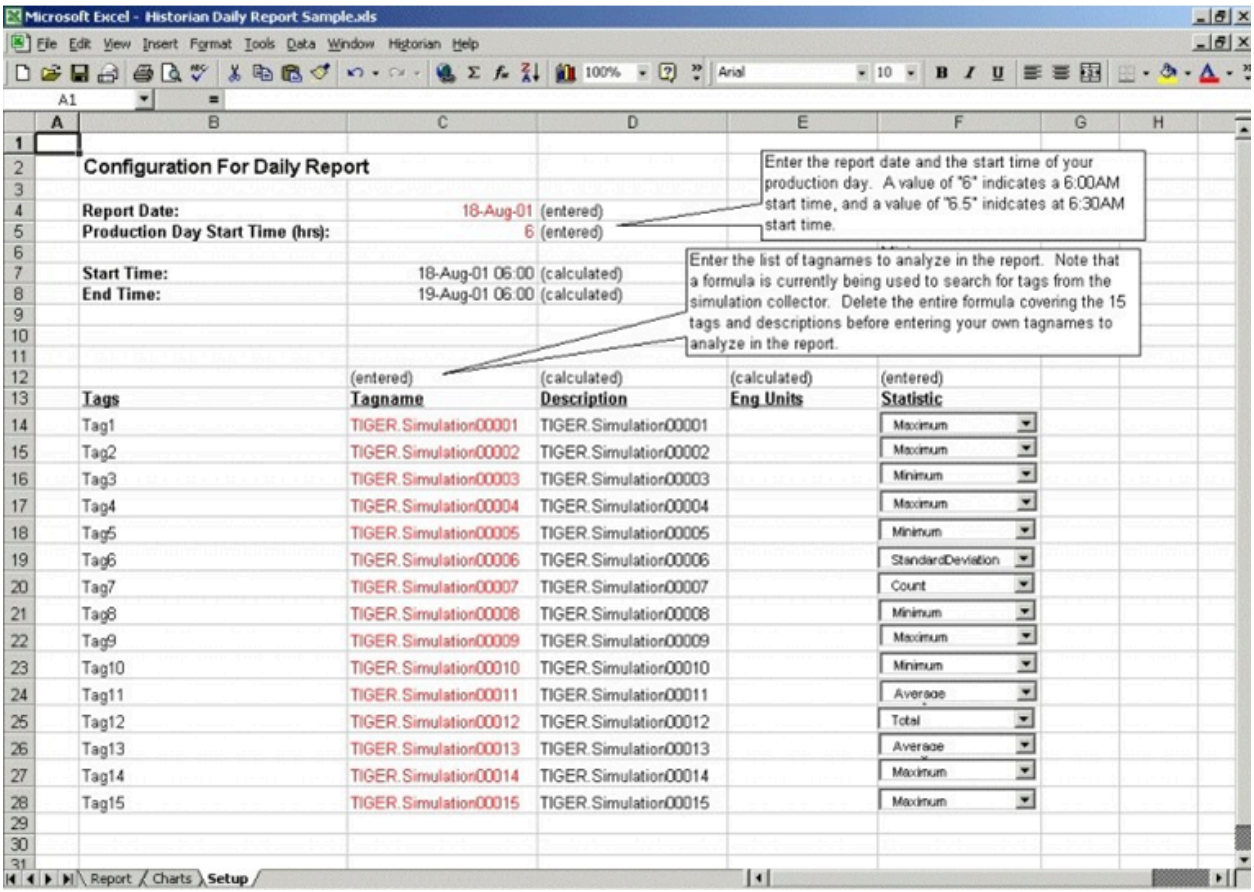
| | TIGER.Simulation00001 | TIGER.Simulation00002 | TIGER.Simulation00003 | TIGER.Simulation00004 | TIGER.Simulation00005 | TIGER.Simulation00006 | TIGER.Simulat |
|---|---|---|---|---|---|---|---|
| | TIGER.Simulation00001 | TIGER.Simulation00002 | TIGER.Simulation00003 | TIGER.Simulation00004 | TIGER.Simulation00005 | TIGER.Simulation00006 | TIGER.Simulat |
| | Maximum | Maximum | Minimum | Maximum | Minimum | StandardDeviation | Coun |
| 7:00 | 199957.28 | 199761.95 | 12.21 | 199981.69 | 18.31 | 57050.91 | 3600.( |
| 8:00 | 199969.48 | 199951.17 | 24.41 | 199987.80 | 18.31 | 57907.26 | 3600.( |
| 9:00 | 199969.48 | 199847.41 | 54.93 | 199957.28 | 36.62 | 57451.76 | 3600.( |
| 10:00 | 199957.28 | 199981.69 | 67.14 | 199823.00 | 189.21 | 57638.78 | 3600.( |
| 11:00 | 199981.69 | 199993.89 | 36.62 | 199938.97 | 42.73 | 57334.71 | 3600.( |
| 12:00 | 199945.06 | 199902.34 | 0.00 | 199981.69 | 103.76 | 57779.99 | 3600.( |
| 13:00 | 199859.61 | 199902.34 | 36.62 | 199969.48 | 73.24 | 58555.56 | 2995.( |
| 14:00 | 199859.61 | 199932.86 | 42.73 | 199975.58 | 18.31 | 57235.52 | 3600.( |
| 15:00 | 199969.48 | 199993.89 | 6.10 | 199914.55 | 12.21 | 57998.48 | 3600.( |
| 16:00 | 199963.38 | 199993.89 | 42.73 | 199993.89 | 61.04 | 57853.26 | 3600.( |
| 17:00 | 199975.58 | 199926.75 | 24.41 | 200000.00 | 18.31 | 56965.60 | 3600.( |
| 18:00 | 200000.00 | 199932.86 | 12.21 | 199823.00 | 36.62 | 56999.84 | 3600.( |
| 19:00 | 199975.58 | 199920.66 | 48.83 | 199969.48 | 18.31 | 57462.40 | 3600.( |
| 20:00 | 199957.28 | 199993.89 | 12.21 | 199981.69 | 24.41 | 57373.37 | 3600.( |
| 21:00 | 200000.00 | 200000.00 | 24.41 | 199987.80 | 6.10 | 57509.65 | 3600.( |
| 22:00 | 199951.17 | 200000.00 | 30.52 | 199993.89 | 73.24 | 57914.51 | 3600.( |
| 23:00 | 199957.28 | 199816.89 | 12.21 | 199981.69 | 18.31 | 57068.75 | 3600.( |
| 0:00 | 199969.48 | 199951.17 | 24.41 | 199987.80 | 18.31 | 58169.76 | 3600.( |
| 1:00 | 199969.48 | 199969.48 | 54.93 | 199957.28 | 36.62 | 57433.42 | 3600.( |
| 2:00 | 199804.69 | 199981.69 | 67.14 | 199823.00 | 189.21 | 57912.77 | 3600.( |
| 3:00 | 199981.69 | 199993.89 | 36.62 | 199938.97 | 42.73 | 57802.84 | 3600.( |
| 4:00 | 199945.06 | 199902.34 | 0.00 | 199981.69 | 103.76 | 57436.26 | 3600.( |
| 5:00 | 200000.00 | 200000.00 | 30.52 | 199975.58 | 85.45 | 57837.37 | 3600.( |
| 6:00 | 199993.89 | 199993.89 | 24.41 | 199963.38 | 0.00 | 58223.36 | 3600.( |
| Average | 199954.73 | 199943.54 | 30.26 | 199953.71 | 51.88 | 57621.50 | 3574.( |
| Std Dev | 47.55 | 63.54 | 19.34 | 54.20 | 51.54 | 415.21 | 123.( |
| Min | 199804.69 | 199761.95 | 0.00 | 199823.00 | 0.00 | 56965.60 | 2995.( |
| Max | 200000.00 | 200000.00 | 67.14 | 200000.00 | 189.21 | 58555.56 | 3600.( |

Report / Charts / Setup /

The report shown in the following image is a collection of chart plots of the data displayed in the report of the previous image.

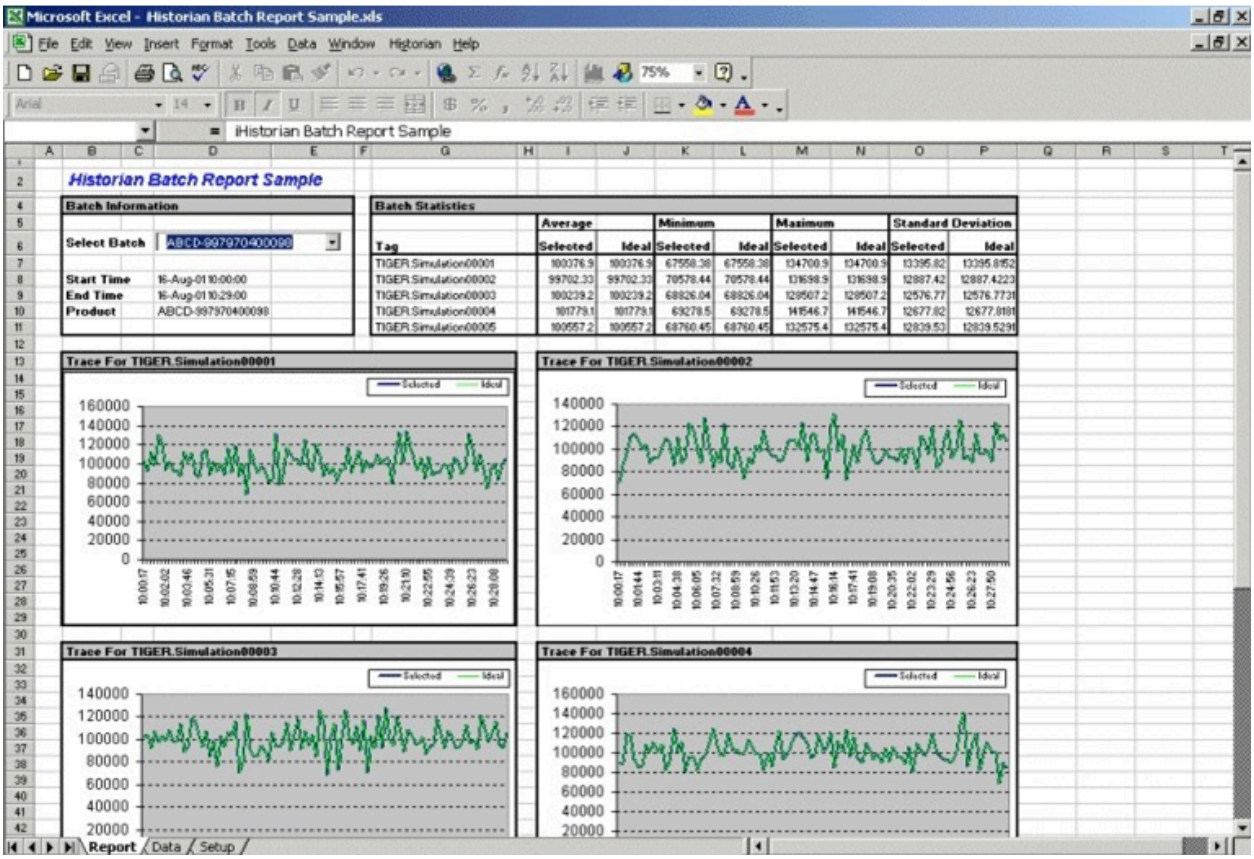



The following figure shows the worksheet used to set up the Daily Sample Report. Edit the worksheet to adapt this report to your application.

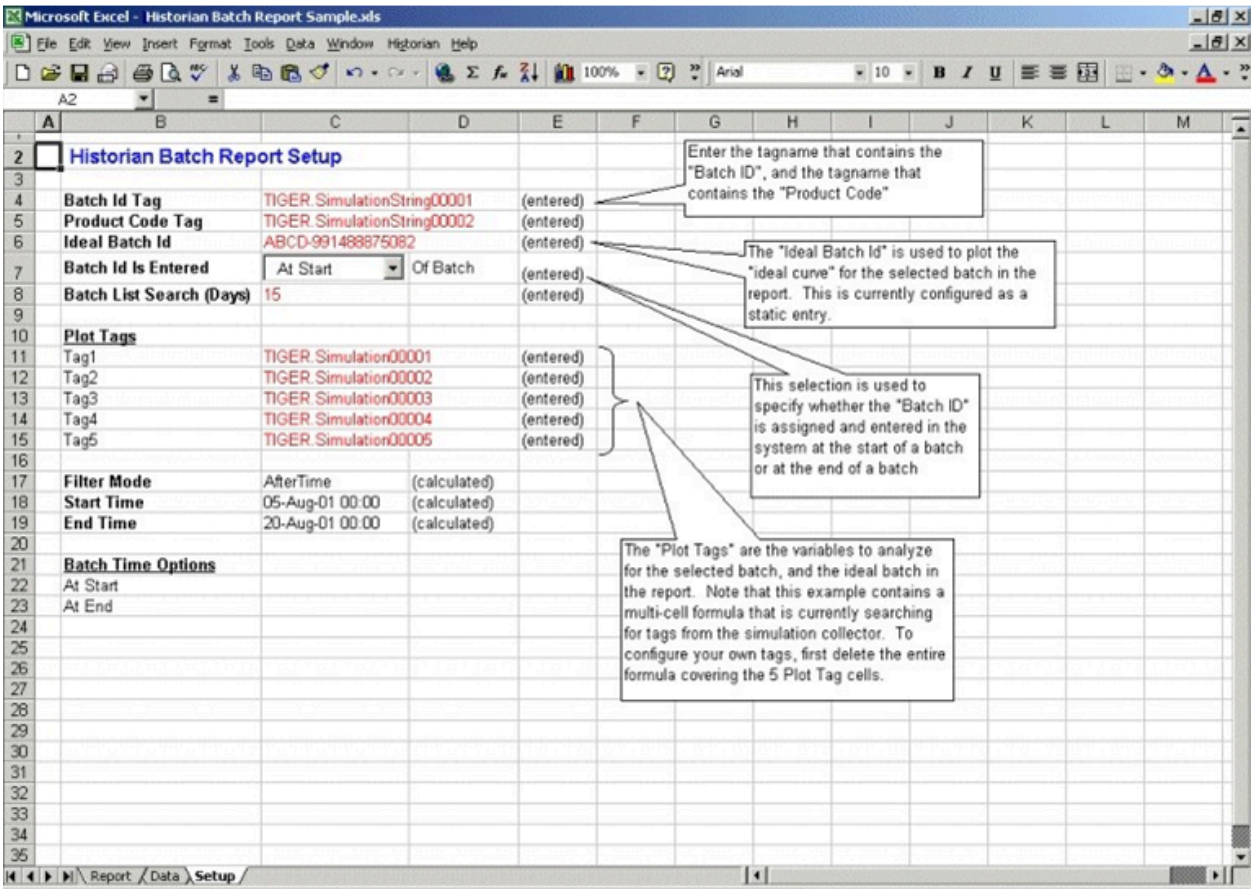



**Batch Sample Report**

This is an example of a report that might be used with a batch type of industrial process. The table at the top of the report shows the batch identification, the start and end times, product name, and computed statistics for several process variables. The charts show how selected process parameters varied during the batch cycle.




This is the configuration worksheet used to generate the report shown in the previous image. Modify this worksheet to adapt it to your requirements.




**Troubleshooting the Excel Add-In Sample Reports**

If you follow the recommended installation procedures, you should not have any difficulty in running the sample reports. If you do encounter any problems, they are likely to relate to the locations of files and the links to those files.

When opening a sample Excel report, you may receive a message prompting you to update all linked information in the workbook or keep the existing information. We recommend that you select No (that is, keep the existing information). The links will be automatically updated for your worksheet. Save your worksheet after the links have been updated.

For problems in the worksheets themselves, refer to Excel online Help for assistance.

**Running a Report Using Visual Basic**

The following Visual Basic example shows you how to create a hidden instance of Microsoft Excel, open a preconfigured Historian report in that instance, and then print the report to the default printer. To use the example, you must modify the path of the `.XLA` and `.XLS` files. The paths that you need to edit are in bold font in the following example.

To use this example, you must have the privileges to run the collector as a Windows service and a default printer must be installed. If Historian security is enabled, you must be a member of the iH Readers group. Tag-level security can override this privilege.

You can trigger this example to run on an event basis or on a polled basis. Most likely, you would run this example on an event basis. However, you can run it on a polled basis using Windows Task Scheduler.

```
Sub CreateExcelObjects()

Dim xlApp As Excel.Application Dim wkbNewBook As Excel.Workbook Dim wksSheet As Excel.Worksheet Dim strBookName As

 String

' Create new hidden instance of Excel. Set xlApp = New Excel.Application

' Open the preconfigured Historian Excel Add-in report.

Workbooks.Open "C:\Program Files\Microsoft Office\Office11\Library\iHistorian.xla"

Set wkbNewBook = Workbooks.Open("c:\testih.xls", 0, False)

'xlApp.Visible = True

With wkbNewBook

For Each wksSheet In .Worksheets

Select Case wksSheet.Name Case "tag1" wksSheet.Select

.RefreshAll

.PrintOut End Select Next wksSheet

.Close False

End With

Set wkbNewBook = Nothing xlApp.Quit

Set xlApp = Nothing

End Sub
```

### Array Formulas for the Historian Excel Add-In

In Excel, an array formula is a data request that inputs a set of parameters and returns results. The Historian Excel Add-In uses the following array formulas:

```
ihSearchTags

(pServer,pTagMask,pDescriptionMask,pCollector,pArraySize,pSort,pRowCol,Parameters())


ihQueryData

(pServer-,pTagName,pStartTime,pEndTime,pSamplingMode,pCalculationMode,pSamplingInterval,pNumberOfSamples,pDirection,pFi

lterTag,pFilterMode,pFilterComparisonMo ())


ihQueryData3
```

```
(pServer,pTagName,pStartTime,pEndTime,pSamplingMode,pCalculationMode,pSamplingInterval,pNumberOfSamples,pDirection,pFil
terTag,pFilterMode,pFilterComparisonMo ())


ihQueryMessages
(pServer,pTopic,pStartTime,pEndTime,pSearchText,pArraySize,pSort,pRowCol,Parameters())


ihListArchives
(pServer,pArchiveNameMask,pArraySize,pSort,pRowCol,Parameters())


ihListCollectors
(pServer,pCollectorNameMask,pArraySize,pSort,pRowCol,Parameters())
```

When inserting an array formula, you cannot overwrite part of the range of another array formula in your worksheet. The range includes cells without data displayed. An error message appears if you try to do so. Reselect a different output range to insert the formula.

**Array Formula Parameters**

The following table describes the parameters for the array formulas for the add-in.

| Parameter | Description |
|---|---|
| `pArchiveNameMask` | A search mask you can use to browse the archivers. Use standard Windows wildcard characters. |
| `pArraySize` | The number of cells that the array spans. |
| `pCalculationMode` | The type of the calculation mode *(on page 1073)*. |
| `pCollector` | The collector or collector mask that you want to query. |
| `pCollectorNameMask` | A search mask for browsing collectors. Use standard Windows wildcard characters. |

| Parameter | Description |
|---|---|
| `pDescriptionMask` | A search mask for browsing tag descriptions. Use standard Windows wildcard characters. |
| `pDirection` | The direction (forward/backward from the start time) of data sampling from the archive. |
| `pEndTime` | The end time used to refine your query. |
| `pFilterComparisonMode` | The type of comparison to be made on the filter comparison value:<br>• `Equal`: Filter condition is True when the `FilterTag` is equal to the comparison value.<br>• `EqualFirst`: Filter condition is True when the `FilterTag` is equal to the first comparison value.<br>• `EqualLast`: Filter condition is True when the `FilterTag` is equal to the last comparison value.<br>• `NotEqual`: Filter condition is True when the `FilterTag` is NOT equal to the comparison value.<br>• `LessThan`: Filter condition is True when the `FilterTag` is less than the comparison value.<br>• `GreaterThan`: Filter condition is True when the `FilterTag` is greater than the comparison value.<br>• `LessThanEqual`: Filter condition is True when the `FilterTag` is less than or equal to the comparison value.<br>• `GreaterThanEqual`: Filter condition is True when the `FilterTag` is greater than or equal to the comparison value.<br>• `AllBitsSet`: Filter condition is True when the binary value of the `FilterTag` is equal to all the bits in the condition. It is represented as `^` to be used in Filter Expression.<br>• `AnyBitSet`: Filter condition is True when the binary value of the `FilterTag` is equal to any of the bits in the condition. It is represented as `~` to be used in Filter Expression.<br>• `AnyBitNotSet`: Filter condition is True when the binary value of the `FilterTag` is not equal to any one of the bits in the condition. It is represented as `!~` to be used in Filter Expression.<br>• `AllBitsNotSet`: Filter condition is True when the binary value of theFilterTag is not equal to all the bits in the condition. It is represented as `!^` to be used in Filter Expression. |

| Parameter | Description |
|---|---|
| `pFilterComparisonValue` | The value to compare the filter tag with when applying the appropriate filter to the DataRecordset query (to determine the appropriate filter times). |
| `pFilterExpression` | An expression that includes multiple filter conditions. The type of conditions used are:<br>• `AND` condition<br>• `OR` condition<br>• Combination of both `AND` and `OR`<br><br>You can use a filter expression instead of `FilterTag`, `FilterComparisonMode` and `FilterValue` parameters. While using `FilterExpression`, the expression is passed within single quotes, and for complex expressions, enclose the conditions in parentheses. There is no maximum length for a filter expression. |
| `pFilterMode` | The type of the time filter:<br>• `ExactTime`: Retrieves data for the exact times that the filter condition is True (only True).<br>• `BeforeTime`: Retrieves data from the time of the last False filter condition up until the time of the True condition (False until True).<br>• `AfterTime`: Retrieves data from the time of the True filter condition up until the time of the next False condition (True until False).<br>• `BeforeAndAfterTime`: Retrieves data from the time of the last False filter condition up until the time of next False condition (While True).<br>• The `FilterMode`: Defines how time periods before and after transitions in the filter condition should be handled.<br><br>For example, `AfterTime` indicates that the filter condition should be True starting at the timestamp of the archive value that triggered the True condition and leading up to the timestamp of the archive value that triggered the False condition. |
| `pFilterTag` | The single tagname used when applying the filter criteria. |

| Parameter | Description |
|---|---|
| `pNumberOfSamples` | Number of samples from the archive to retrieve.<br><br>Samples will be evenly spaced within the time range defined by start time and end time for most sampling modes. For the `RawByNumber` sampling mode, the `NumberOfSamples` column determines the maximum number of values to retrieve. For the `RawByTime` sampling mode, the `NumberOfSamples` is ignored. |
| `pRowCol` | The sorting criteria used: 0 for columns and 1 for rows. |
| `pSamplingInterval` | For non-raw sampled data, this column represents a positive integer for the time interval (in milliseconds) between returned samples. |
| `pSamplingMode` | The type of the sampling mode *(on page 1044)* used by the query. |
| `pSearchText` | The text or mask that you want to search for in the message. |
| `pServer` | Name of the server from which you are retrieving data. If you are running Excel on the same server from which you are retrieving data, you need not enter a string, as the default server is used. |
| `pSort` | The sorting criteria used for the rows or columns: 0 for descending and 1 for ascending. |
| `pStartTime` | The start time used to refine your query. |
| `pTagMask` | A search mask for browsing tagnames. Use standard Windows wildcard characters. |
| `pTagName` | The tagname or tagname mask that you want to query. |
| `pTopic` | The message topic:<br><br>• Connections<br>• Configuration |

| Parameter | Description |
|---|---|
| | • General<br>• Services<br>• Performance<br>• Security |
| `Parameters()` | Output display of the array formula. This field can include be one or more parameters. |

## Relative Time Entries

When entering the Start and End times for Excel Add-in queries and exports, you can already enter them as exact literal dates and times such as *"1/28/14 09:00:00"* in the query windows like **Query Calculated Data**, or you can use a cell reference to an exact time, or use an Excel function such as `=Now()` or `=Today()`. Apart from the mentioned ways, you can use relative time entries using a base value and an offset value as described in the following tables.

For example, you can use `Yesterday+8H` for 8am yesterday or `Now-15m` for 15 minutes before the current time. The typical use of a relative time entry, is to type the time values using a base and an offset into the start and end time of the **Query** window or the **Export** window, instead of having to put `=Now()` or `=Today()` in a cell and making a cell reference to that, or use the base `Monday` to produce weekly reports.

### Base Values

| Base Value | Description |
|---|---|
| `Now` | The current date and time. |
| `Today` | The current date at midnight. |
| `Yesterday` | The previous day at midnight. |
| `Sunday` | Today or the most recent Sunday at midnight. |
| `Monday` | Today or the most recent Monday at midnight. |
| `Tuesday` | Today or the most recent Tuesday at midnight. |
| `Wednesday` | Today or the most recent Wednesday at midnight. |

| Base Value | Description |
|---|---|
| `Thursday` | Today or the most recent Thursday at midnight. |
| `Friday` | Today or the most recent Friday at midnight. |
| `Saturday` | Today or the most recent Saturday at midnight. |

### Offset Values

| Offset Value | Description |
|---|---|
| `d` | One 24 hour day |
| `h` | One hour |
| `m` | One minute |
| `s` | One second |

## Filter Parameters for Data Queries

| Parameters | Description |
|---|---|
| `FilterTag` | The single tag name used when applying the filter criteria.<br><br>**Note:**<br>You can enter your filter conditions using Filter tag, Filter Comparison Mode, and Filter Comparison Value or you can put that all that information in a single Filter Expression. |
| `FilterExpression` | An expression that includes one or more filter conditions. The types of conditions used are:<br><br>• AND Condition<br>• OR Condition<br>• Combination of both AND and OR<br><br>FilterExpression can be used instead of FilterTag, FilterComparisonMode and FilterValue parameters. There is no maximum length for a filter expression. |
| `FilterMode` | The type of time filter: |

| Parameters | Description |
|---|---|
| | • **ExactTime** — Retrieves data for the exact times that the filter condition is True (only True).<br>• **BeforeTime** — Retrieves data from the time of the last False filter condition up until the time of the True condition (False until True).<br>• **AfterTime** — Retrieves data from the time of the True filter condition up until the time of next False condition (True until False).<br>• **BeforeAndAfterTime** — Retrieves data from the time of the last False filter condition up until the time of next False condition (While True).<br><br>The Filter Mode defines how time periods before and after transitions in the filter condition should be handled.<br><br>For example, AfterTime indicates that the filter condition should be True starting at the timestamp of the archive value that triggered the True condition and leading up to the timestamp of the archive value that triggered the False condition. |
| `FilterComparisonMode` | The type of comparison to be made on the filter comparison value:<br><br>• **Equal** — Filter condition is True when the Filter Tag is equal to the comparison value.<br>• **EqualFirst** — Filter condition is True when the Filter Tag is equal to the first comparison value.<br>• **EqualLast** — Filter condition is True when the Filter Tag is equal to the last comparison value.<br>• **NotEqual** — Filter condition is True when the Filter Tag is NOT equal to the comparison value.<br>• **LessThan** — Filter condition is True when the Filter Tag is less than the comparison value.<br>• **GreaterThan** — Filter condition is True when the Filter Tag is greater than the comparison value.<br>• **LessThanEqual** — Filter condition is True when the Filter Tag is less than or equal to the comparison value.<br>• **GreaterThanEqual** — Filter condition is True when the Filter Tag is greater than or equal to the comparison value.<br>• **AllBitsSet** — Filter condition is True when the binary value of the Filter Tag is equal to all the bits in the condition. It is represented as `^` to be used in Filter Expression. |

| Parameters | Description |
|---|---|
| | • **AnyBitSet** — Filter condition is True when the binary value of the Filter Tag is equal to any of the bits in the condition. It is represented as `~` to be used in Filter Expression.<br>• **AnyBitNotSet** — Filter condition is True when the binary value of the Filter Tag is not equal to any one of the bits in the condition. It is represented as `!~` to be used in Filter Expression.<br>• **AllBitsNotSet** — Filter condition is True when the binary value of the Filter Tag is not equal to all the bits in the condition. It is represented as `!^` to be used in Filter Expression.<br>• **Alarm Condition** — Specifies an alarm condition to filter data by. For example, Level.<br>• **Alarm SubCondition** — Specifies an alarm sub-condition to filter data by. For example, HIHI.<br><br>The Filter Comparison Mode defines how archive values for the Filter Tag should be compared to the Filter Value to establish the state of the filter condition. If a Filter Tag and Filter Comparison Value are supplied, time periods are filtered from the results where the filter condition is False.<br><br>**Note:**<br>Filter Comparison Mode is only used if Filter Tag is filled in. |
| `FilterComparisonValue` | The value to compare the filter tag with when applying the appropriate filter to the data record set query (to determine the appropriate filter times).<br><br>**Note:**<br>Filter Comparison Value is only used if Filter Tag is filled in. |

## Batch IDs

If you had a `BatchID` going into a Historian tag, that `BatchID` will either have a timestamp at the beginning of the batch or at the end of the batch. Different batch systems report the `BatchID` as the batch is started, and other systems do not report the `BatchID` until the batch is finished.

If your `BatchID` is reported at the beginning of a batch, you would need to use the **AfterTime** option because you would want to include all data for a particular `BatchID` after the time the BatchID was

reported up until the next `BatchID` was reported. If your `BatchID` was being reported at the end of the batch, you would want to use the **BeforeTime** option because you would want to include all data for a particular `Batch ID` before the time the `Batch ID` was reported back to the previous `BatchID` being reported.

## Sampling Types

**Interpolated Sampling**

Calculates values between two data points using a linear interpolation algorithm.

**Calculated Sampling**

Computes values using an algorithm selected in the Calculation field.

**Lab Sampling**

Computes intermediate values between two data points by using the last actual value. This type of sampling displays as a stair step type of curve.

**Trend Sampling**

Returns the raw minimum and raw maximum value for each specified interval. Use the Trend sampling mode to maximize performance when retrieving data points for plotting. For the Trend sampling mode, if the sampling period does not evenly divide by the interval length, **Historian** ignores any leftover values at the end, rather than putting them into a smaller interval.

**InterpolatedtoRaw Sampling**

Provides raw data in place of interpolated data when the number of samples fall lesser than the available samples.

**TrendtoRaw Sampling**

The TrendtoRaw sampling mode almost always produces the same results as the Trend sampling mode. The exception is that, when more samples are requested than there are raw data points, the TrendtoRaw sampling mode returns all of the available raw data points with no further processing. TrendtoRaw is therefore used rather than Trend when the number of actual data samples are fewer than the requested number of samples.

**LabtoRaw Sampling**

Provides raw data for the selected calculated data over the plot, when the number of samples fall lesser than the available samples.

**RawByFilterToggle Sampling**

Returns filtered time ranges with values 0 and 1. If the value is 1, then the filter condition is true and 0 means false. This sampling mode is used with the time range and filter tag conditions. The result starts with a starting time stamp and ends with an ending timestamp.

**Trend2 Sampling**

Returns the raw minimum and raw maximum value for each specified interval. Use the Trend2 sampling mode to maximize performance when retrieving data points for plotting. Also, if the sampling period does not evenly divide by the interval length, Historian creates as many intervals of the interval length as will fit into the sampling period, and then creates a remainder interval from whatever time is left. Trend2 sampling mode is more suitable than Trend sampling mode for analysis of mins and maxes and for plotting programs that can handle unevenly spaced data.

**TrendtoRaw2 Sampling**

The TrendtoRaw2 sampling mode almost always produces the same results as the Trend2 sampling mode. The exception is that, when more samples are requested than there are raw data points, the TrendtoRaw2 sampling mode returns all of the available raw data points with no further processing. TrendtoRaw2 is therefore used rather than Trend2 when the number of actual data samples are fewer than the requested number of samples.

## Calculation Algorithm Types

**Average**

A time weighted arithmetic mean.

**Minimum**

The lowest value in the group.

**Maximum**

The highest value in the group.

**Standard Deviation**

The square root of the arithmetic mean of deviations from the time- weighted arithmetic mean of all values in the group.

**Total**

The time-weighted total of all values in the group. Note that Engineering Units are assumed to be in Units/Day. If your Engineering Units were not measured in Units/Day, you must scale your total to the actual time units of the measurement. For example, if the measurement

were in Units/Minute (such as GPM), you would multiply the total number by 1440 (minutes in a day) to scale the value into the correct time units.

**Count**

The total number of values in the group.

**Raw Average**

The unweighted arithmetic mean of all values in the group.

**Raw Standard Deviation**

The square root of the arithmetic mean of deviations from the unweighted arithmetic mean of all values in the group.

**Raw Total**

The unweighted total of all values in the group.

**Time of Minimum Value**

The time at which the minimum value occurred. l Time of Maximum Value - the time at which the maximum value occurred.

**Time Good**

The amount of time (in milliseconds) during the interval when the data quality is good.

**State Count**

Displays the number of times a tag has transitioned to another state from a previous state. A state transition is counted when the previous good sample is not equal to the state value and the next good sample is equal to state value.

**State Time**

Displays the duration that a tag was in a given state within an interval.

**First Raw Value**

Returns the first good raw sample value in the given time interval.

**First Raw Time**

Returns the time stamp of the first good raw sample in the given time interval.

**Last Raw Value**

Returns the last good raw sample value in the given time interval.

**Last Raw Time**

Returns the time stamp of the last good raw sample in the given time interval.

**TagStats**

Returns the values of multiple calculation modes in a single query.

**Delta Positive, Delta Negative, and Delta**

Return the delta over a time interval. For more information, refer to:

- DELTAPOS *(on page 1116)*
- DELTANEG *(on page 1132)*
- DELTA *(on page 1141)*

## Tag Criteria List

The following table outlines the tag criteria available:

| Criteria | Description |
|---|---|
| Tagname | Tagname or tag mask property of the tag. |
| Description | User description of the tag. |
| Data Type | The data type of the tag. |
| Collector Name | Name of the collector responsible for collecting data for the specified tag. |
| Collector Type | The type of collector responsible for collecting data for the tag.<br>**Note:**<br>Do not use wildcards in this field. |
| Collection Type | Type of collection used to acquire data for the tag. |
| Data Store Name | Indicates the name of the data store to which the tag belongs to. |
| EGU Description | Indicates the engineering units assigned to the tag.<br>**Note:**<br>Do not use wildcards in this field. |
| Comment | Comments that is applied to the tag. |

| Criteria | Description |
|---|---|
| | **Note:** Do not use wildcards in this field. |
| Source Address | The address for the selected tag in the data store.<br>**Note:** Do not use wildcards in this field. |
| Collection Interval | The time interval between the readings of data. The value entered is in milliseconds. |
| Collector Compression | Whether or not collector compression is enabled as a default setting. |
| Archive Compression | Indicates the current effect of archive data compression. |
| Last Modified User | The name of the person who last modified the tag configuration parameters. |
| Enumerated Set Name | Indicates the enumerated set name associated with the tag. |

# Troubleshooting Issues with the Add-In

**Troubleshooting General Imports**

- Review the `HistorianSDKErrors.log` file. This file is usually located in the `LogFiles` folder in your Historian program folder. Historian records additional information for some errors in this file. Sometimes, by reviewing this file, you can determine the cause of the error.
- If using Historian security, verify that the user has the appropriate security rights. If the rights are incorrect, log in as a user with the correct privileges or change the rights for the current user.
- Verify that there are no empty rows between valid rows in your spreadsheet. These empty rows can cause issues.
- Note if any errors occur. If an error occurs with any line of the import, Historian aborts the whole import.

**Troubleshooting Tag Imports**

- If you remove or add Historian servers, and then if you attempt to search for tags, the add-in may not recognize the default server, and may display a message, stating that the default server has not been set. To avoid this issue, close and reopen the **Search Tags** window.
- Make sure that you are not trying to import the Calculation Execution Time, Last Modified, or Last Modified User fields for each tag. These fields are read-only. As such, you can export them but cannot import them.
- Verify that your collector does not contain any duplicate tagnames.
- Verify that the number of tags that you want to import does not exceed the maximum licensed tag count. If it does, you will not be able to import the tags.

**Troubleshooting Data Imports**

- Ensure that the time stamps of any online archives are not prior to the start time of the oldest online archive.
- Ensure that the time stamps are within the active hours setting in the **Data Store Maintenance** page of Historian Administrator.
- Ensure that the time stamps are not for a time greater than 15 minutes ahead of the system time on the Historian server.
- Ensure that the tags are valid Historian tags. To do this, import your tags before importing their associated data.

**Troubleshooting Data or Tag Exports**

You cannot export data or tags to a remote path using the add-in.

You can export a 64-bit tag, include it in a report and perform calculations on it. However, there will be a minor precision loss while retrieving the data due to a Visual Basic limitation.

**Importing Tags Fails**

**Description:** If you export all the fields and attempt to import the read-only fields **LastModified** and **LastModifiedUser**, you may receive an error message.

**Error Message:** Import failed, Error with Import Header.

**Workaround:** Export the tags without selecting the read-only fields, and then import the tags.

**Unable to Run Sample Reports**

**Description:** If you follow the recommended installation procedures, you should not have any difficulty in running the sample reports. If you do encounter any problems, they are likely to relate to the locations of files and the links to those files.

**Workaround:** When opening a sample Excel report, you may receive a message prompting you to update all linked information in the workbook or keep the existing information. We recommend that you select No (that is, keep the existing information). The links will be automatically updated for your worksheet. Save your worksheet after the links have been updated.

For problems in the worksheets themselves, refer to Excel online Help for assistance.

**Error Occurs While Inserting an Array Formula**

When inserting an array formula, you cannot overwrite part of the range of another array formula in your worksheet. The range includes cells without data displayed. An error message appears if you try to do so. Reselect a different output range to insert the formula.

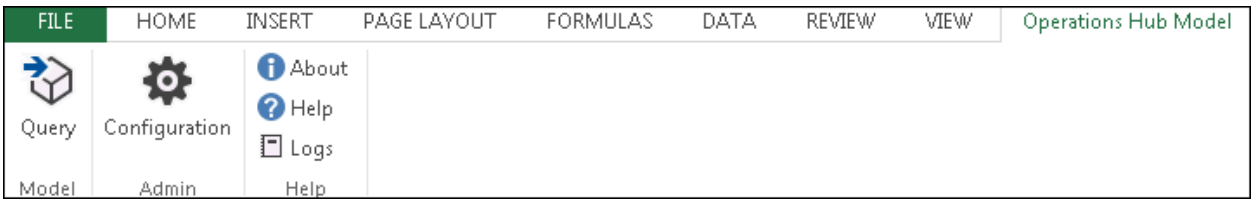
# Chapter 40. The Excel Addin for Operations Hub

## Overview of the Excel Add-In for Operations Hub

The Excel Add-in for Operations Hub enhances the benefits of using Operations Hub. It enables you to retrieve the object data that can be used to perform further analysis using Excel features.

**Features:**

- **Query:** Enables you to perform the model query and to import the data into Excel. You can query historical data based on object types, objects, and data variables.
- **Configuration:** Enables you to connect to Operations Hub and save the server details.
- **About:** Displays the version information of the add-in.
- **Help:** Displays the product documentation for the add-in.
- **Logs:** Opens the folder where you can view the logs.




In addition, you can:

- View the metadata of the selected data variables and objects in Excel.
- Select the data filter options with which you can filter the data to be retrieved.
- Select the output display options for which you want to view the data.

**Limitations:**

- If you query the data variables of data type string, byte, or array, a null value is returned
- You cannot query current value for an array tag using the Excel Add-In for Operations Hub. .
- If you query the following calculation modes, an error occurs:
    - Minimum Time
    - Maximum Time
    - First Raw Time
    - Last Raw Time
    - Time Good

## About Operations Hub

Operations Hub is an end-to-end solution for developing, managing, and delivering applications to leverage the capabilities of big data analytics and the internet of things. Using Operations Hub, you can create applications that will collect and analyze data from a machine or a server, and trigger actions based on certain events.

Operations Hub provides you a user-friendly interface to create components of an application such as queries, database tables (called entities), events, email templates, users, and so on without the need to use your programming skills. You can also design pages and dashboards using these components.

### Advantages of using Operations Hub

- Operations Hub is quick, easy, and cost-effective. You do not need programming skills to develop an application.
- The Operations Hub applications use HTML5 and CSS3, and hence, they are platform-independent.
- You can access an application using a computer or a mobile device.
- You can provide controlled access to an application and data, based on user roles.
- You can create entities and queries for a relational database.

For more information about Operations Hub, refer to

https://www.ge.com/digital/documentation/opshub/windows/windows/c_overview_of_app_designer.html

# Setting Up

## Software Requirements

The following components are required to use Excel Add-in for Operations Hub:

| Component | Version | Description |
|---|---|---|
| Operations Hub | 2.0 and above. | If you have purchased the standard or enterprise license of Historian, you receive a no-cost license for Operations Hub, which enables you to: |

| Component | Version | Description |
|---|---|---|
| | | • Access to the Historian Analysis run-time application, which is an in-built HTML5 application in Operations Hub.<br>• Perform advanced trend analysis, including inserting annotations.<br>• Define an asset model including tag mapping. |
| Microsoft Excel | 2016 and 2019 (32 bit or 64 bit) | |
| Historian REST APIs | | Historian REST APIs are required to integrate between Historian and Operations Hub. Historian REST APIs are installed automatically when you install Historian Web-based Clients *(on page 155)*. |

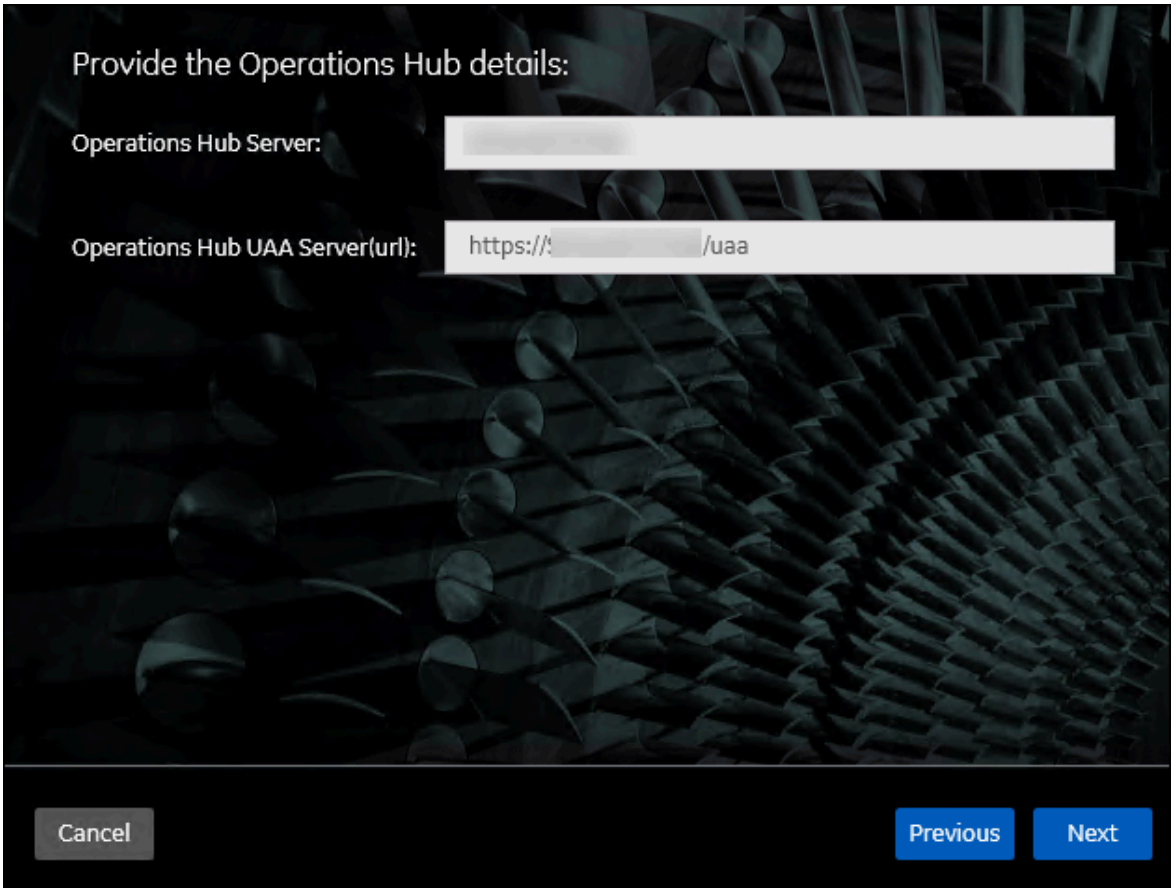
## Install Excel Add-In for Operations Hub

Install the Historian server *(on page 104)* and other software requirements *(on page 198)*.

1. Run the `InstallLauncher.exe` file.
2. Select **Install Excel Add-in for Operations Hub**.
   The welcome page appears.
3. Select **Next**.
4. Read and accept the license agreement, and then select **Next**.
5. Select the available disk to install the Excel Add-in for Operations Hub, and then select **Next**.

   **Note:**
   We recommend that you select the drive where Microsoft Excel is installed.

6. Provide the details of Operations Hub, and then select **Next**.




The **You are ready to install** page appears.

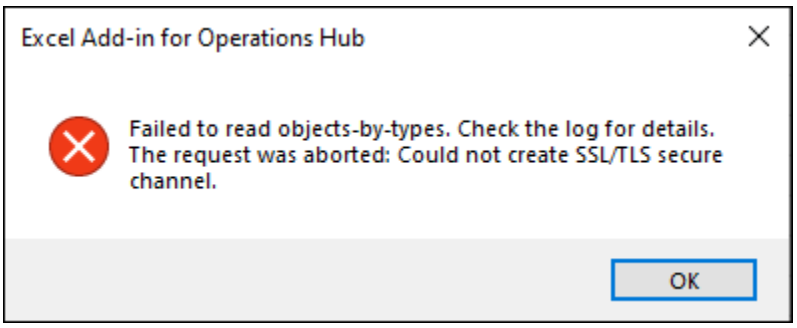
7. Select **Install**.
   Excel Add-In for Operations Hub is installed.

Copy/export the issuer certificate *(on page 200)*, and then install/import it *(on page 201)*.

## Copy or Export the Issuer Certificate on Server

Install Excel Add-In for Operations Hub *(on page 199)*.

This topic describes how to copy or export the issuer certificate on Server. If this is not done, you will get an error while querying data as shown in the image below.




1. Navigate to the machine where Operations Hub is installed.
2. Select **Site Information (Not secure)**.
3. Select **Certificate (invalid)**.
   The **Certificate** window appears.
4. Select **Certificate Path**.
5. Select the Root CA certificate.
6. Select **Details**.
7. Select **Copy to file**.
   The **Certificate Export Wizard** window appears.
8. Select **DER encoded binary X.509(.CER)** format and select **Next**.
9. Select **Browse** to save the certificate file at desired location.
10. Complete the certificate export.

Install or import the certificate *(on page 201)*.

## Install/Import the Issuer Certificate

Copy or export the issuer certificate *(on page 200)* on the machine on which Excel Add-In for Operations Hub is installed.

1. Right-click the certificate, and then select **Install Certificate**.
   The **Certificate Import Wizard** page appears
2. Select **Local Machine**, and then, select **Next**.
3. Select **Place all certificates in the following store**.
4. Select **Trusted Root certification Authorities**, and then select **OK**.
5. Select **Next**, and then select **Finish**.
   The certificate is imported.

Configure the Operations Hub server *(on page 202)*.

## Connect to Operations Hub

To query a model defined in Operations Hub, you must first connect to the Operations Hub server. You will then receive a token from the server, which will be used for authentication.

1. Select **Configuration** menu in Admin.
   The **Operations Hub Configuration** window appears.
2. Provide values as described in the following table.

| Field | Details |
|---|---|
| Operations Hub Server | The Operations Hub server name to which you want to connect and get the data. |
| Operations Hub Proficy Authentication Server (url) | The URL of the Proficy Authentication service of Operations Hub.<br><br>Example: https://<ophubservername>/uaa |

> **Note:**
> The **Token Status** field indicates the status of the connection with Operations Hub server.

3. Select **Connect**.
   The login page appears.
4. Provide the **User Identifier** and **Password** to connect to Operations Hub.
5. Select **Open UaaAuthSchemeHandler**.
   Operations Hub Server to which you are connected and the status of the token appears.
6. Select **Save** to save the Operations Hub server details. The configuration will be retained and used when you open excel add-in again.

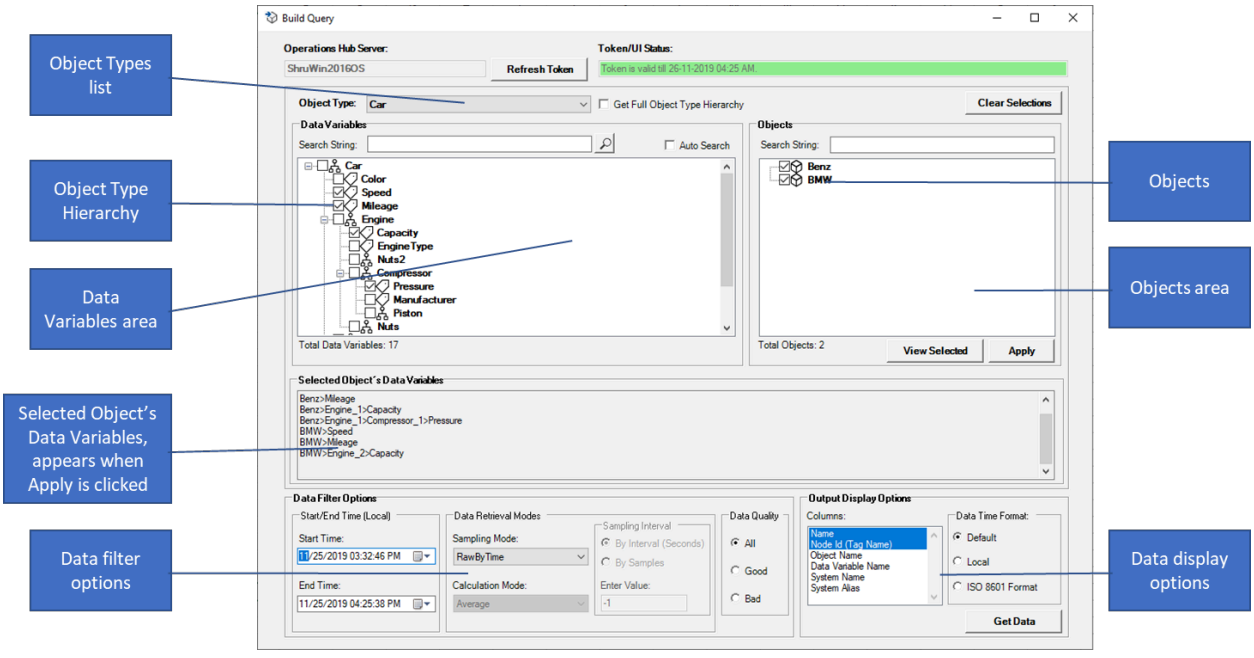
# Querying an Operations Hub Model

You can query an Operations Hub model using the object types, data variables, and objects:

| Item | Description |
|---|---|
| Object types | Define the structure of the equipment within your model (for example, a car).<br><br>Object types are represented by . |

| Item | Description |
|---|---|
| Data variables | Define the actual data that is received from a data source - that is, how to use a property in the Views feature. For example, you can define a property to appear as a trend line on a trend chart. For each object type, such as a car, you set up all the data variable names (such as color, speed). Any object associated with this type can reuse them in its own definition.<br><br>Data variables are represented by . |
| Objects | Instances of object types or pieces of equipment (for example, BMW, BENZ).<br><br>Objects are represented by . |

The following image shows the object types, data variables, and objects in the add-in.




## Query Operations Hub Model

1. Access an Excel worksheet.
2. Select the object type from the **Object Type** drop-down list box. Example: Car.

The corresponding data variables and contained types are displayed in the **Data Variables** list for the selected Object Type.

3. Optional: Select the **Get Full Object Type Hierarchy** check box to get the complete object type hierarchy for the selected object type in the data variables

> **Note:**
> Selecting this option might impact the performance and you may experience the delay in retrieving and browsing the object type hierarchy.

4. Select the data variables from the **Data Variables** list.

> **Note:**
> Only one level of the object type hierarchy is displayed. Double-click the object types to browse through the other levels.

> **Tip:**
> You can right select in the **Data Variables** section to perform the following actions:
> - **Check All**: Selects all the data variables.
> - **Collapse All**: Collapses the hierarchy.
> - **Expand All**: Expands the hierarchy displaying all the data variables.
> - **Uncheck All**: Deselects all the data variables.

5. Optional: Enter a search criterion in the **Search String** text box and then, select the Search icon to filter the data variables based on the string entered.

> **Note:**
> '*' can be used as wild card character in the search text box.

The resultant data variables matching the filter string appear in the data variables list.

6. Optional: Select the **Auto Search** check box to filter the data variables as you type the string in the **Search String** text box. Note: Auto search option is recommended only when your model has less number data variables.
7. Select the objects from the **Objects** list. You can right-select in the **Objects** section to perform the following actions:
   - **Check All**: Selects all the objects.
   - **Uncheck All**: Deselects all the objects.

8. Select **Apply** to apply the selection to query the model.
9. Select **View Selected** to view the metadata of selected data variables and objects. The details will appear on Excel.

> **Note:**
> You can continue with building the query returning to the Build Query window by selecting the Query button.

| Name | ObjectName | DataVariable Name | SystemAlias | NodeId | BindingType | Schema Version | SystemName | SystemType | ObjectTypeHierarchy |
|---|---|---|---|---|---|---|---|---|---|
| BMW>Speed | BMW | Speed | HistMachine1 | HistMachine1.Ramp_1%Noise | Historical | 1 | HistMachine1 | Historian | Car>Speed |
| BMW>Mileage | BMW | Mileage | HistMachine1 | HistMachine1.Simulation00005 | Historical | 1 | HistMachine1 | Historian | Car>Mileage |
| BMW>Engine_2>Capacity | Engine_2 | Capacity | HistMachine2 | HistMachine2.Simulation00003 | Historical | 1 | HistMachine2 | Historian | Car>Engine>Capacity |
| BMW>CEAT Tyre>Durability | CEAT Tyre | Durability | HistMachine3 | HistMachine3.Simulation00008 | Historical | 1 | HistMachine3 | Historian | Car>Tyres>Durability |

10. Optional: You can select the **Clear Selections** button to clear all selections.
11. Select the **Data Filter Options** to retrieve the data based on the sampling and calculation modes:
    - Start/End Time: The duration for which you want to retrieve the data.
    - Sampling modes: The sampling mode for the data. It specifies the way data will be retrieved from Historian. Example: CurrentValue, Interpolated, Calculated and RawByTime. For more information, refer to Sampling Modes *(on page 1044)*.
    - Calculation modes: Calculation modes are used when the sampling mode is set to Calculated. The data type of all calculated values will be DoubleFloat except for MinimumTime, MaximumTime, FirstRawTime and LastRawTime which will be a Date. The datatype of the values of FirstRawValue and LastRawValue will be the same as that of the selected tag.

      The Calculation Field is active only after you select Calculated Sampling as the Sample Type. You can select a Calculation Algorithm type from the drop-down list.

      The calculation modes supported by Historian are supported by Excel Add-in for Operations Hub except Minimum Time, Maximum Time, First Raw Time, Last Raw Time, Time Good, State Count, State Time.

      > **Note:**
      > Minimum Time, Maximum Time, First Raw Time, Last Raw Time, Time Good calculation modes are listed in the Calculation Modes drop-down but are not supported.

      For more information, refer to Calculation Modes *(on page 1085)*.

> **Note:**
>
> Some of the calculation modes such as Minimum Time,Maximum Time, First Raw Time, Last Raw Time, Time Good are returning bad data upon query.

12. Select the **Data Quality** based on which you want to export the samples.
13. Select the **Data Display Options**
    - **Columns**: Select the Columns for which you want the values to be displayed. By default, Node, Node Id (Tag Name) will be selected.
    - **Data time to display**: Choose the timestamp format to be displayed.
        - Default: To display the time format as in Operations Hub.
        - Local: To display the time in local time zone.
        - ISO 8601 Format: To display time in ISO 8601 readable format. (Result will include T for the time designator and Z for the zero UTC offset)

        > **Note:**
        > When you query for a data variable in a duration which has no data, the Default mode selection displays time value as 0 and Local and ISO 8601 format displays time as 1970 year because, the 0 epoch time converted to local time is 1970.

14. Select **Get Data** to read and display data for the selected data variables in the excel sheet.

| Name | Node Id (Tag Name) | Time | Value | Quality |
|---|---|---|---|---|
| Benz>Mileage | HistMachine1.Simulation00004 | 1575369343000 | 173491.625 | Good |
| Benz>Engine_1>Compressor_1>Pressure | HistMachine1.Simulation00004 | 1575369343000 | 173491.625 | Good |
| BMW>CEAT Tyre>Durability | HistMachine2.Simulation00008 | 1575369343000 | 87215.79688 | Good |
| Benz>MRF Tyre>Durability | HistMachine2.Simulation00003 | 1575369334000 | 48927.27344 | Good |
| BMW>Mileage | HistMachine3.Simulation00005 | 1575369334000 | 122385.3281 | Good |

# Troubleshooting Issues with the Add-In

## Query Error

**Error Message:** Failed to read objects-by-types. Check the log for details. The request was aborted: Could not create SSL/TLS secure channel.

**Workaround:** Provide values as described in the following table:

| Field | Details |
|---|---|
| Name | SchUseStrongCrypto |
| Value | '1' |
| Type | DWord |
| Operations Hub Proficy Authentication Server (url) | HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\.NetFramework\v4.0.30319 |

### Error Occurs if you Query an Object Type with many properties and containment objects

**Error Message:** Proxy server could not handle the request.

**Workaround:** Use the on-demand approach.

### Blank Login Page Appears

**Issue:** When Operations Hub Proficy Authentication is opened (logged in) in browser and then if you try to connect to Operations Hub Proficy Authentication from Excel Add-in, sometimes, a blank login page appears.

**Workaround:** Refresh the browser window which pops up authorize window and then, select **Open UaaAuthSchemeHandler**.

### Data Variable Selections are not Retained

**Issue:** When the data variables are searched and selected from the Object Type hierarchy, previously made selections of data variables (before search) are not retained.

**Workaround:** Remove the text entered in the **Search** text box and select **Apply** for all the previous selections to be applied.

### Error Occurs Even Before Performing an Action

**Error Message:** Failed to get access_token.

**Workaround:** Get the token from the Operations Hub Proficy Authentication server by selecting **Configuration** in the add-in.

# Chapter 41. Trend Client

## Overview of Trend Client

Trend Client provides a simple and intuitive interface to analyze your process and equipment data to:

- Troubleshoot and improve your processes.
- Save operational cost, money, and risk.

Using Trend Client, you can plot trend charts and current-value tables, which help you visualize timeseries data.

To install Trend Client, install Web-based Clients.

### Access Trend Client

Web-based Clients.

1. On the desktop, select . Alternatively, you can access the following URL: `https://<Trend Client web server name>/uaa/login#/home`
2. Log in with the credentials you provided while installing Web-based Clients.
   The Trend Client home page appears.

### Access Help

In the upper-right corner of Trend Client, select .
The Help documentation for Trend Client appears.

## Access a Tag

When you search for a tag, you can narrow down the search results further by performing a basic search or an advanced search.

**Note:**
By default, maximum one million tags are retrieved. If the Historian clients are configured to retrieve more than a million tags, to retrieve all of them, add the `MaxTagsToRetrieve` registry key under `HKEY_LOCAL_MACHINE\SOFTWARE\Intellution, Inc.\iHistorian\Services`

`\DataArchiver\`, and then set the maximum number of tags that you want to retrieve. Restart the Historian Data Archiver service for the change to reflect.

1. Access Trend Client *(on page 2799)*.
2. If you want to access all tags, select **Tags > Search**.
   The **Tags Search** window appears.
3. If you want to access all tags, select .
   A list of all the tags in the Historian server appears. You can filter the list of tags by entering the name or description in the text box.
4. If you want to perform an advanced search:

   a. On the **Tags** page, select **Advance Search**.
      The **Advanced Tag Search** window appears.

   b. Expand **Select Tag Criteria**, and enter values in the search criteria.

      **Note:**
      Supported wildcard characters in search are * and ?. The following table provides examples of searching using wildcard characters.

| Search String | Result |
|---|---|
| W* | All tags starting with letter W |
| *e | All tags ending with letter e |
| W*e | All tags starting and ending with W and e respectively |
| Tag? | All four letter named tags, where the last letter can be any-thing |
| W*0000? | All tag names staring with W and ending with 0000 followed by any single letter |

   c. Select **Find Tags**.
      A list of tags that match the search criteria appears.

      **Note:**
      Do not add or update the following spare configurations as the data may get corrupted or over written:

- The **Spare 1** field for OSI PI Distributor. OSI PI distributor reads data from the Historian tag displayed in the Tag Source Address field and sends it to the OSI PI tag name displayed in the **Spare 1** field.
- The **Spare 5** field for Server to Server Collector and Server to Server Distributor as it is only used for internal purposes.

To analyze the tag data, add it in a trend chart *(on page 2802)*, current value table *(on page 2803)*, value card *(on page 2803)*, or a text box *(on page 2804)*.

## Add Tags for Analysis

Before you access the trend chart or value table of a tag, you must add it to the **Tags** section.

1. On the **Tags** page, select **Advance Search**.
   The **Advanced Tag Search** window appears.
2. Expand **Select Tag Criteria**, and enter values in the search criteria.

**Note:**
Supported wildcard characters in search are * and ?. The following table provides examples of searching using wildcard characters.

| Search String | Result |
|---|---|
| W* | All tags starting with letter W |
| *e | All tags ending with letter e |
| W*e | All tags starting and ending with W and e respectively |
| Tag? | All four letter named tags, where the last letter can be anything |
| W*0000? | All tag names staring with W and ending with 0000 followed by any single letter |

3. Select **Find Tags**.
   A list of tags that match the search criteria appears.
4. In the row containing each tag that you want to add, select , and then select **Apply**.
   The tags appear in the **Summary** and **Tags** sections.

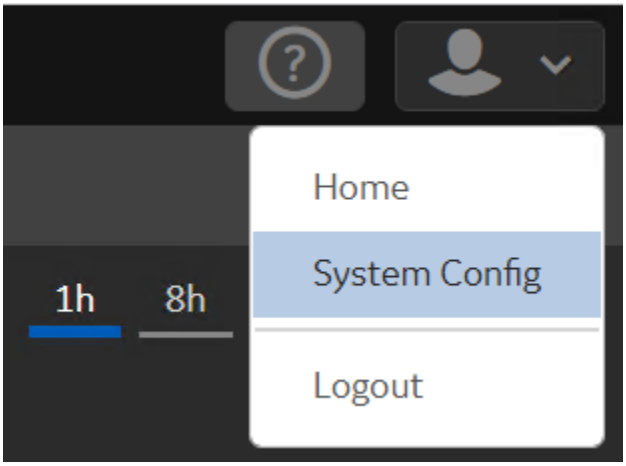
To analyze the tag data, add it in a trend chart *(on page 2802)*, current value table *(on page 2803)*, value card *(on page 2803)*, or a text box *(on page 2804)*.

# Creating a Display

## Add a Trend Chart

By default, the sample size limit that is plotted on a trend chart is 2000. If you want to change this value:

1. In the upper-right corner of Trend Client, select the drop-down list box, and then select **System Config**.




   The **System Configuration** page appears.

2. In the **Analysis Sample Size Limit** field, enter the sample size, and press Enter.

   The changes are automatically saved.

SImilarly, maximum 30 days of data is plotted on a trend chart; you can change this value.

Trend charts are graphical representations for showing how the value of one or more items changes over time. Trend charts can show information as Line, Area, Scatter, and Statistics Charts.

1. Access Trend Client *(on page 2799)*.
2. Select .
3. Add tags for analysis *(on page 2801)*.
4. Select the tags, and drag and drop them into the chart area.

A trend chart is plotted for the selected tags. A blue dot appears in the row containing each tag in the **Tags** section, indicating that you have added it to the chart.

> **Tip:**
> By default, the title of the table is Chart <number>. You can rename the title as needed.

You can add more trend charts, or you can add a current value table *(on page 2803)*, value card *(on page 2803)*, or a text box *(on page 2804)* to the display.

## Add a Current Value Table

A current value table provides the current value of each tag, along with the quality, description, units of measurement, and so on.

1. Access Trend Client *(on page 2799)*.
2. Select .
   A blank current value table appears.
3. Add tags for analysis *(on page 2801)*.
4. In the **Tags** section, select each tag that you want to include in the current value table, and drag and drop it to the table.
   The tags are added to the current value table, displaying the current value, quality, description, and so on.

> **Tip:**
> By default, the title of the table is Current Values <number>. You can rename the title as needed.

You can add more current value tables, or you can add a trend chart *(on page 2802)*, value card *(on page 2803)*, or a text box *(on page 2804)* to the display.

## Add a Value Card

The value card is a snapshot of the most current value of a tag. Each value card is limited to a single tag.

1. Access Trend Client *(on page 2799)*.
2. Select .
   A blank current value card appears.

3. Add the tag for analysis *(on page 2801)*.
4. In the **Tags** section, select the tag that you want to include in the value card, and drag and drop it to the value card.
   The tags are added to the value card, displaying the current value, quality, description, timestamp, and so on.

> **Tip:**
> By default, the title of the table is Current Values <number>. You can rename the title as needed.

You can add more value cards, or you can add a trend chart *(on page 2802)*, current value table *(on page 2803)*, or a text box *(on page 2804)* to the display.

## Add a Text Box

You can add a text box to enter free-form text to your display. The text box contains standard editing icons.

1. Access Trend Client *(on page 2799)*.
2. Select .
   A blank text box appears.
3. Enter text and apply formatting as needed.

You can add more text boxes, or you can add a trend chart *(on page 2802)*, current value table *(on page 2803)*, or a value card *(on page 2803)* to the display.

## Access a Display

1. In the upper-right corner of Trend Client, select .
   A list of displays that you have saved appears.
2. Select the display that you want to access.

The display appears in the main section.

> **Tip:**
> If you want to remove the legends and statistics, and arrange the items in the display vertically with two elements in each row, select and then **Column View**. If you want to view only the trend charts, select and then **Stacked View**.

## Provide a Title to a Display

1. Access Trend Client *(on page 2799)*.
2. In the **Click to add title** box, enter the title of the display, and then press Enter.
   The name must be unique and must not contain commas.
   The title of the display is saved.

## Filter Data

After adding tags to a trend chart, table, or a value card, you can filter the data based on certain criteria. You can apply the filter only to a selected item in a display or to all the items.

> **Note:**
> You cannot filter data from multiple Historian servers.

This topic describes how to filter data based on tag values. You can also filter data based on the duration *(on page 2812)*.

1. Access the display in which you want to filter data.
2. Select **Filter**.
3. Select the tag, operator, and value in the corresponding drop-down list boxes. For example, suppose you have plotted a tag named Tag1 on a trend chart. If you want to plot only the values greater than 150, select **Tag1**, **is greater than (>)**, and **150**, respectively.
4. Ensure that the **Enabled** toggle is switched on.
5. As needed, add more conditions by selecting **Add Condition**.
6. If you want to apply the filter only to the selected item, select **Apply**. If you want to apply the filter to all the items in the display, select **Apply to All**.
   The data is filtered based on the criteria you have specified.

## Change the Sampling Mode

By default, the sampling mode of items in a display is interpolated (1000 samples over the last hour). You can change the sampling mode of data in a display. You can change the mode only to a selected item in a display or to all the items.

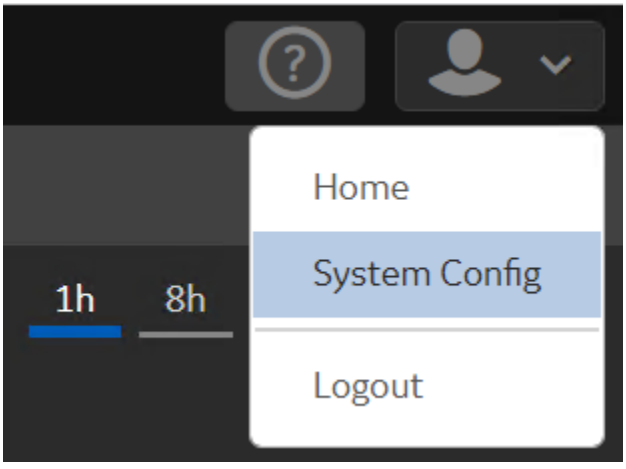
You can choose from various sampling modes.

1. Access the display for which you want to change the mode.
2. Select **Mode**.
3. In the **Sampling Mode** field, select the sampling mode that you want to apply.
4. In the **Sample Increment** field, select one of the following options:
    - **By Size**: Select this option if you want each sample to contain a specific number of data points, and then enter the size. You must enter a value less than the value in the **System Configuration** page. Otherwise, an error message appears.
    - **By Time**: Select this option if you want each sample to contain the data points collected in a fixed duration of time.
5. If you want to use the sampling mode only to the selected item, select **Apply**. If you want to use the sampling mode to all the items in the display, select **Apply to All**.
   The sampling mode is changed.

## Change the Time Zone

By default, the local time zone is used for a display. You can change it to UTC.

1. In the upper-right corner of Trend Client, select the drop-down list box, and then select **System Config**.

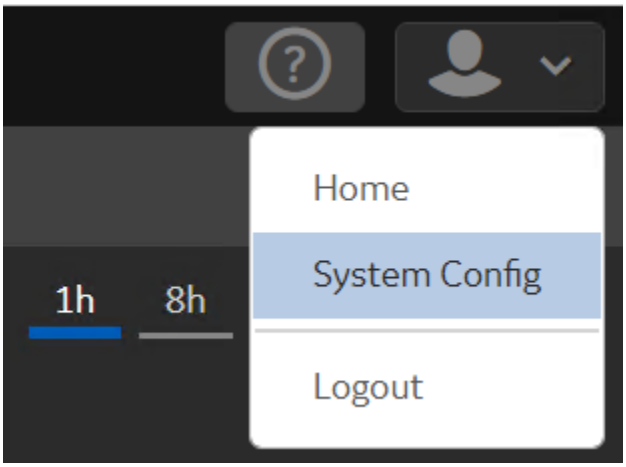



   The **System Configuration** page appears.

2. Under **Time Display**, select the time zone you want to use.
   A message appears, confirming that the time zone has been changed.
3. Log out of Trend Client, and log in again.
   The time zone is changed.

## Export Data

- Install Microsoft Excel 2007 or 2010.
- By default, the sample size limit that is plotted on a trend chart is 2000. In addition, you can export data for a maximum duration of 30 days (43200 minutes) or 100,000 data points, whichever option contains less data. And, the default delimiter is a semicolon. If you want to change these default values:
  1. In the upper-right corner of Trend Client, select the drop-down list box, and then select **System Config**.

     The **System Configuration** page appears.
  2. In the **Export Download Settings** section, change the default values as needed. For the delimiter, do not enter a character that is used in xml tags (such as <, >, &).

     The changes are automatically saved.

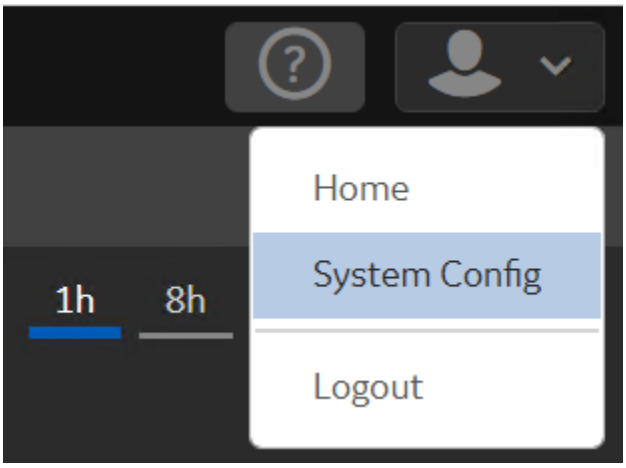
You can export raw tag data or the data in a trend chart.

1. Select the charts with the tag data you want to export.
2. If you want to export raw data, select , and then select **Export Raw Data**. If you want to export trend data, select , and then select **Export Trended Data**.
   The data is exported as a CSV file.

## Set the Refresh Interval

By default, the content in a display is refreshed every 30 seconds. You can change this value. Remember that decreasing the interval can increase the server load.

1. In the upper-right corner of Trend Client, select the drop-down list box, and then select **System Config**.



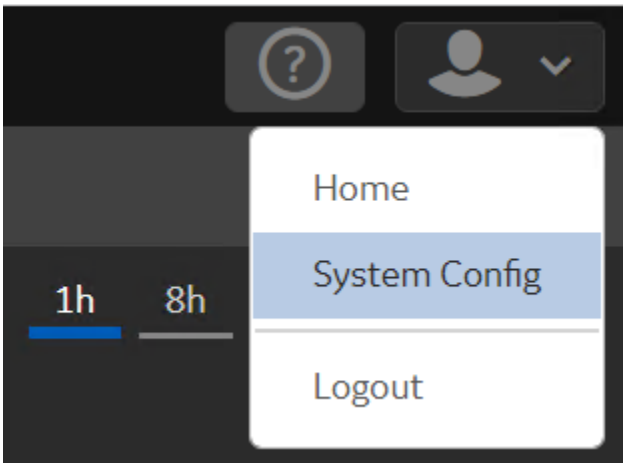

   The **System Configuration** page appears.
2. In the **Content Refresh Interval**, select the duration, and press Enter.
   The refresh interval is changed.

# Working with a Trend Chart

## Add a Trend Chart

By default, the sample size limit that is plotted on a trend chart is 2000. If you want to change this value:

1. In the upper-right corner of Trend Client, select the drop-down list box, and then select **System Config**.




   The **System Configuration** page appears.

2. In the **Analysis Sample Size Limit** field, enter the sample size, and press Enter.

   The changes are automatically saved.

SImilarly, maximum 30 days of data is plotted on a trend chart; you can change this value.

Trend charts are graphical representations for showing how the value of one or more items changes over time. Trend charts can show information as Line, Area, Scatter, and Statistics Charts.

1. Access Trend Client *(on page 2799)*.
2. Select .
3. Add tags for analysis *(on page 2801)*.
4. Select the tags, and drag and drop them into the chart area.
   A trend chart is plotted for the selected tags. A blue dot appears in the row containing each tag in the **Tags** section, indicating that you have added it to the chart.

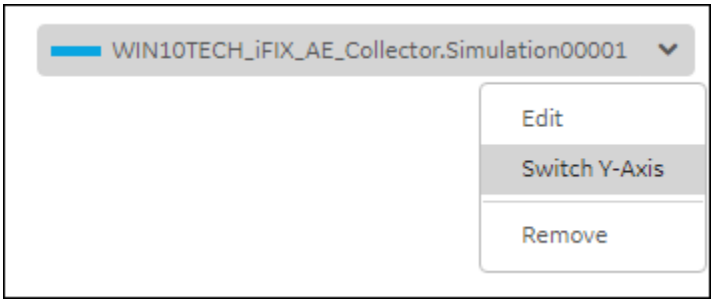
> **Tip:**
> By default, the title of the table is Chart <number>. You can rename the title as needed.

You can add more trend charts, or you can add a current value table *(on page 2803)*, value card *(on page 2803)*, or a text box *(on page 2804)* to the display.

## Switch the Y-Axis of a Trend Chart

You can switch the y-axis to the other side of a trend chart.

In the trend chart, select the drop-down list box that is labelled after the tag name, and then select **Switch Y-Axis**.



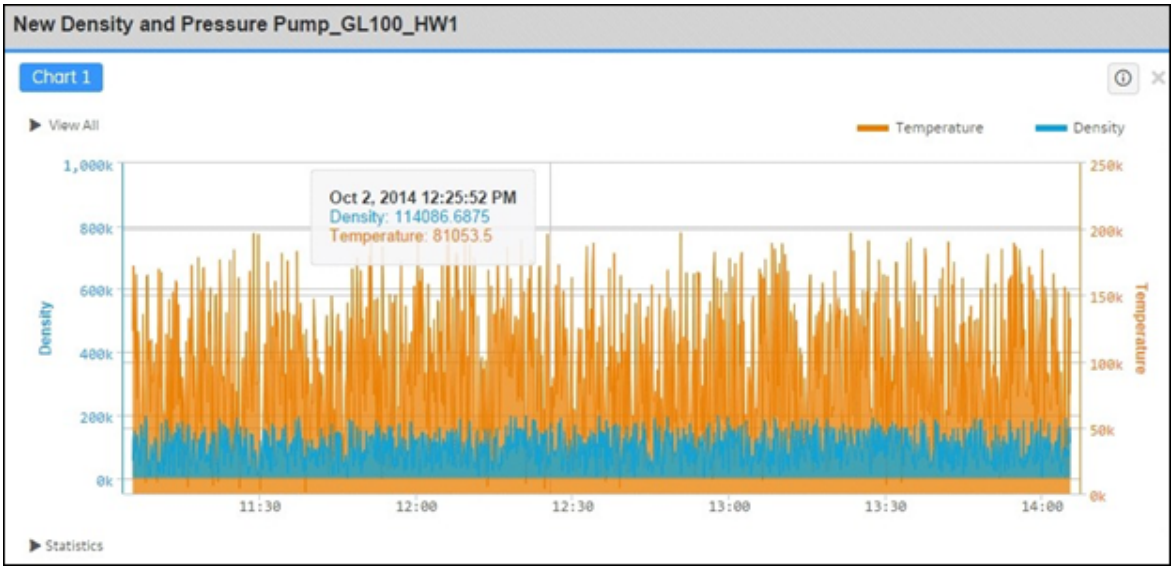

The y-axis for the tag is shifted to the other side of the trend chart.

## Change the Format of a Trend Chart

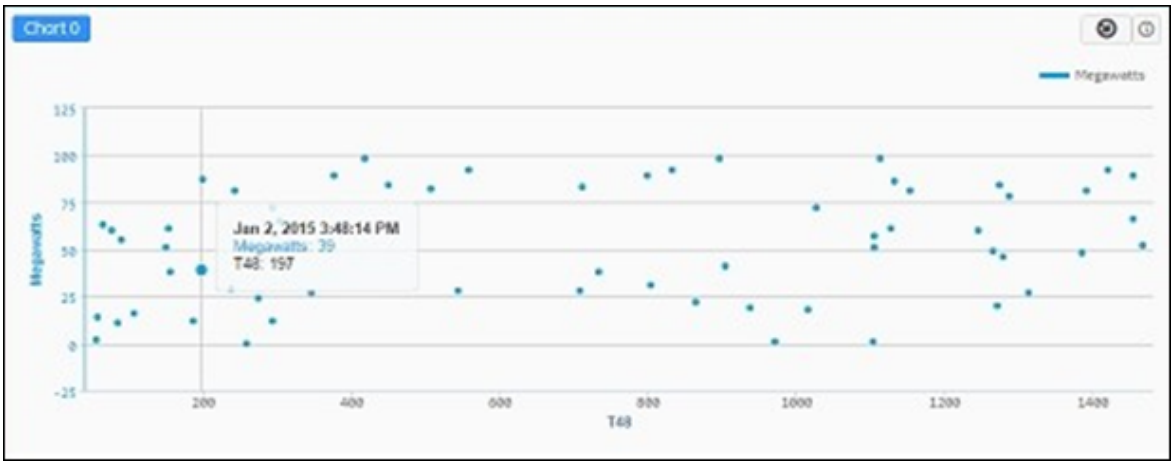
You can choose to plot each tag in any of the following formats:

- **Line:**

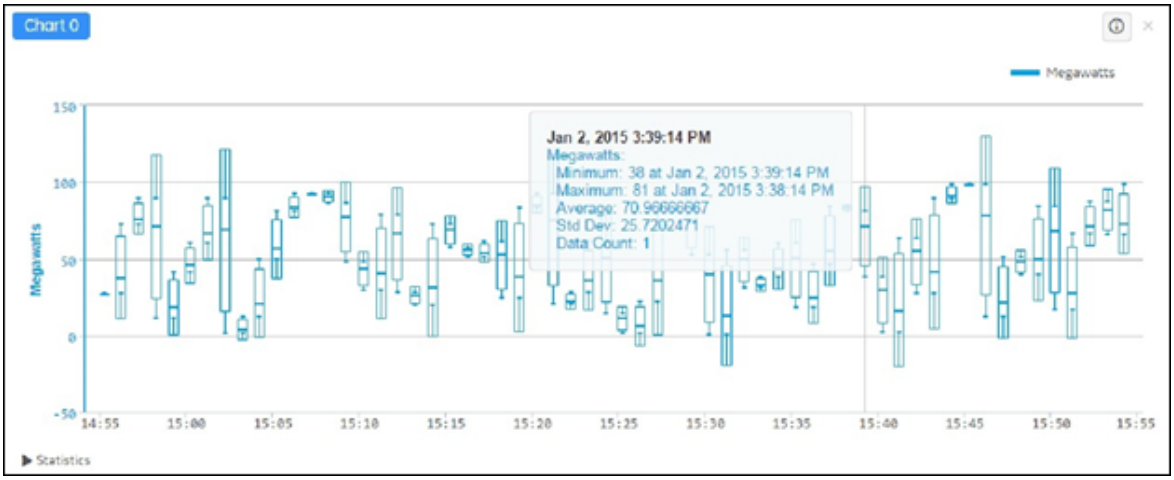
- **Area:**




- **Scatter:**

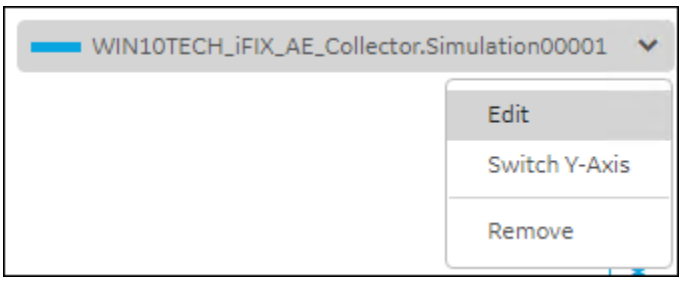



- **Statistics:**




If you select **Statistics**, a box and whisker chart are displayed in the trend chart. When you pause over, the minimum, maximum, average, standard deviation, and data sample counts within the interval as determined by the Historian Sampling Mode for the chart appear.

1. In the trend chart, select the drop-down list box that is labelled after the tag name, and then select **Edit**.




   The **Chart Editor** window appears.
2. For the tag selected in the **Select Series** field, in the **Type** field, select the trend chart format that you want to use.
3. Select **OK**.
   The format of the trend chart is changed for the selected tags.

## Change the Duration of a Display

By default, the duration used for items in a display is one hour up to the current time. You can choose of the following durations:

- One hour
- Eight hours
- One day

- One week
- One month

In addition, you can set the start and end dates and timezone.

1. Select **Tags**, and then select one of the following durations:
    - **1h** (one hour)
    - **8h** (eight hours)
    - **1d** (one day)
    - **1w** (one week)
    - **1m** (one month)

   The items in the display are plotted for the selected duration.
2. If you want to change the start and end dates:
    a. Select **Time**.
    b. Enter values in the **Start** and **End** fields.
    c. If you want to apply the start and end dates only for the selected item, select **Apply**. If you want to apply the changes to all the items in the display, select **Apply to All**.

## Access the Statistics of a Trend Chart

You can access the following statistics of a trend chart:

- First raw value
- Last raw value
- Minimum value
- Maximum value
- Count
- Raw total
- Raw average
- Raw standard deviation

1. Select the trend chart whose statistics you want to access.
2. Select **Statistics**.

   The statistics of the trend chart appear.

## Change the Sampling Mode of a Trend Chart

1. Select the trend chart whose sampling mode you want to change.
2. After you have added a chart(s), select **Mode** at the top of the page.

3. Select your Historian Sampling Mode from the drop-down menu.
    - If your selection for **Sampling Mode** is **Calculated**, chose your calculation from the drop-down menu.
4. Select the **Sample Increment** from the drop-down menu as either **By Size** or **By Time**.

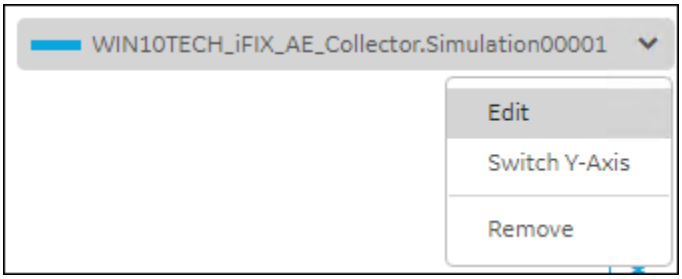
> **Note:**
> Number of samples by size must be equal to or less than the limit defined by your System Administrator; otherwise, a warning message informs you that the data will NOT be retrieved.

5. Select **Apply** to apply to a specific chart, or select **Apply to All** to apply to all the charts in this analysis session.

## Change the Scale of a Trend Chart

By default, depending on the minimum and maximum tag values plotted in a trend chart, the upper scale and lower scale of the y-axis are considered. You can, however, choose to enter these scales manually.

1. In the trend chart, select the drop-down list box that is labelled after the tag name, and then select **Edit**.




   The **Chart Editor** window appears.
2. For the tag selected in the **Select Series** field, switch the **Auto Scale** toggle.
   The **Min** and **Max** fields appear.
3. Enter the lower scale and upper scale values in the **Min** and **Max** field respectively, and then select **OK**.
   The scale of the trend chart is changed.

# Managing Favorites

## Access a Favorite

1. In the upper-right corner of Trend Client, select .
   A list of favorites appears.
2. Select the favorite that you want to access.
   The favorite appears in the main section.

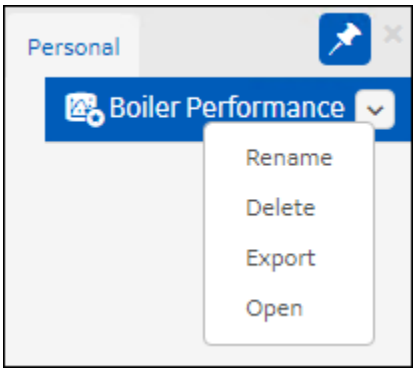
## Export a Favorite

You can share your favorites with another uses by exporting the favorites. When you do so, a JSON file is downloaded, containing the details of the favorites. Another user can then import this JSON file *(on page 2816)* to access your favorites.

- Open the **Action** menu for a single favorite and select on **Export**.

  Your JSON file will be saved in your Downloads folder.

1. In the upper-left corner of Trend Client, select .
   A list of favorites appears.
2. In the row containing the favorite that you want to export, select , and then select **Export**.




   If you want to export all your favorites, select .
   The favorites are exported as a JSON file.

Share the JSON file with the user who wants to import the favorites *(on page 2816)*.

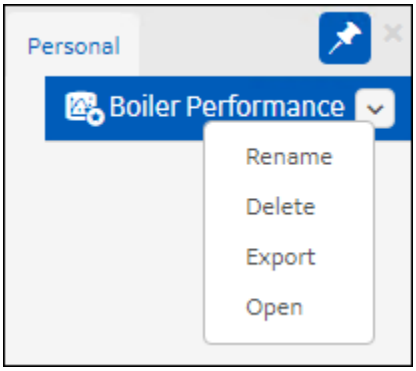
## Import a Favorite

1. Copy the JSON file created by exporting the favorites that you want to import. For information, refer to Export a Favorite *(on page 2815)*.
2. If you want to import the favorites to Trend Client installed on a different Historian server, update the Historian server name in all the tag names in the JSON file.

1. In the favorites section, select .
   The **Import Favorites** window appears.
2. Select **Browse**, and select the JSON file that contains the favorites you want to import.
3. Select **OK**.
   The favorites are imported.

## Delete a Favorite

1. In the upper-left corner of Trend Client, select .
   A list of favorites appears.
2. In the row containing the favorite that you want to delete, select , and then select **Delete**.




   A message appears, asking you to confirm that you want to delete the favorite.
3. Select **Yes**.
   The favorite is deleted.

# Chapter 42. Historian Web Admin Console

## Overview

### Overview of the Web Admin Console

The Web Admin console is a web-based user interface, which you can use to monitor, supervise, archive, retrieve, and control data gather functions from the Historian server, a client, or one or more remote web-based nodes. It contains a diagnostic dashboard and Configuration Manager.

Using the Web Admin console, you can:

- Monitor and troubleshoot the system performance.
- Maintain and configure the Historian System.
- Retrieve and analyze archived information.
- Set up and maintain configuration and other parameters for tags, collectors, and archives.
- Perform specific supervisory and security tasks for the Historian system.

The **WhereTo** page is displayed when you logout and login to the Web Admin console without closing tab or browser.

**Workaround:** Logout and close the tab or browser and reopen it.

### Difference Between the Web Admin Console and Historian Administrator

The following features are available in the Web Admin consoleand not in Historian Administrator:

- Diagnostic Manager
- Ability to add or configure mirror nodes

The following features are available in Historian Administrator and not in the Web Admin console:

- A single interface to access all the servers.
- Ability to create a calculated tag.
- Ability to assign read/write/admin groups to a tag or a set of tags.
- Ability to define enumerated sets.
- Ability to define user-defined types.
- Ability to configure an OPC HDA server.

**Note:**
Regardless of whether you perform a task using the Web Admin console or Historian Administrator, the changes are reflected in both the applications.

## Actions You Can Perform Using the Web Admin Console

- Examine key operating statistics for archives and collectors, and displays them in an interactive user interface.
- Mirror stored data on multiple nodes to provide high levels of data reliability and redundancy. With Data Mirror, you can have continuous data read and write functionality. Data Mirroring gives you:
    - High availability of Historian Server as any of the mirrored nodes can answer read requests.
    - Data Redundancy because data is stored in multiple locations.
- Perform archive maintenance, including:
    - Set archive size.
    - Select options and parameters.
    - Display security parameters.
    - Add and restore archives.
    - Perform routine backup and restoration tasks.
- Perform tag maintenance, including:
    - Add, delete, and copy tags.
    - Search for tags in a data source or in the Historian Database.
    - Start and stop collection on a tag.
    - Configure display, and edit tag parameters and options.
    - Display trend data for selected tags.
- Perform data collector maintenance, including:
    - Add or delete collectors.
    - Configure, display, and edit parameters for all types of collectors.
    - Create calculation formulas.
    - Display performance trends for selected collectors.

## Access the Web Admin Console

Install Historian Web-based Clients *(on page 154)*.

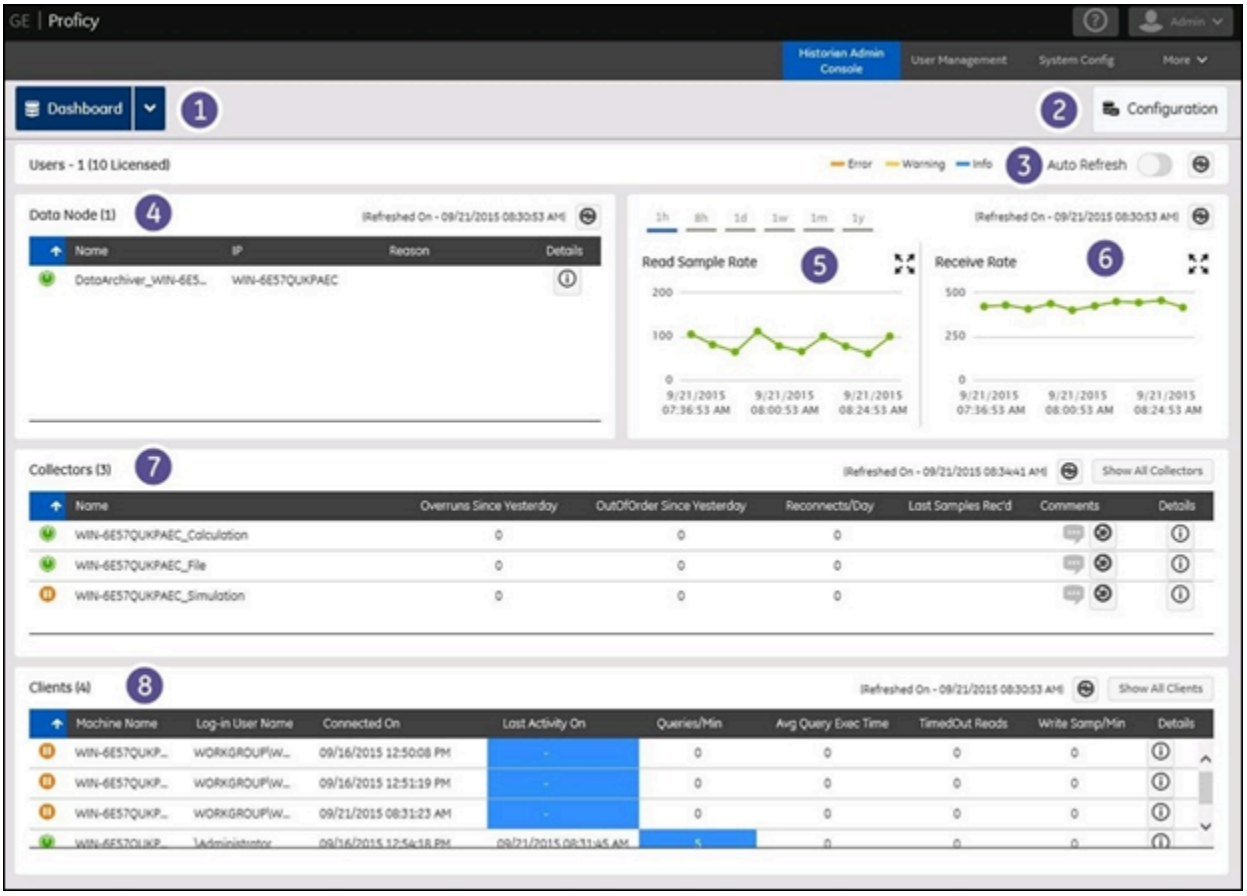
1. In a web browser, enter a URL in the following format: `https://<web server name>/historian-visualization/hwa#/home`, where *<web server name* is the computer name on which you have installed the Web Admin console.
   The login page appears.

2. Log in with your username and password.
   The Web Admin console appears.

# Understanding the Interface

## Understand the Historian Interface

The main interface consists of several panels or windows and appears as shown in the following image:




The elements for this interface are defined as follows:

**Table 407. Interface Descriptions**

| Number | Item | Description |
|---|---|---|
| 1 | Dashboard | This link at the top left of the page opens the Historian Dashboard page – the one that you are currently viewing, which displays the overall picture of the system health. |
| 2 | Configuration | This link opens the configuration panel. From there, you can view and modify the details of collectors, client services, data stores, tags |

**Table 407. Interface Descriptions (continued)**

| Number | Item | Description |
|---|---|---|
| | | and active jobs. For more information, refer to the Configuration Panel *(on page 2830)* topic. |
| 3 | Auto / Manual Refresh | If Auto Refresh is ON, then the page is refreshed automatically. If Auto Refresh is OFF, then you can manually refresh the page or the individual section of the dashboard by selecting the icon. |
| 4 | Data Node | This panel displays the basic information of the nodes (Primary Node & Mirror Nodes) currently available. To view the details of a particular node, select the Details button of the node. The Data Node page appears. |
| 5 | Read Sample Rate | This panel gives you the trend of the average read sample rate across all archives in the data store per sample per minute. You can choose the time scales by selecting on the time options provided on the top right area of the page. To scale the panel, select the icon. Receive Rate This panel gives you the trend of the recent rate at which the samples have been received per minute. You can choose the time scales by selecting on the time options provided on the top right area of the page. To scale the panel, select the icon. |
| 6 | Collectors | This panel displays the details of all the unhealthy collectors connected to the system. To view the details of a particular collector, select the Details button. The Collector Detail Diagnostics window appears. To view all the collectors in the system select the Show All Collectors button. The Configuration Page appears. For more information on the collector panel, refer to the Collector Statistics topic. |
| 7 | Clients | This panels displays the client statistics of the top five read and write clients in the order of the load that they impose on the server. For more information, refer to the Client Statistics topic. |
| 8 | Color codes | The Error, Warning, and Information color codes are displayed based on the status of the Data Node, Collectors or Clients. |

## Client Panel

The Client panel in the Dashboard shows the details of all the connected clients in the system in the order of their fault levels.

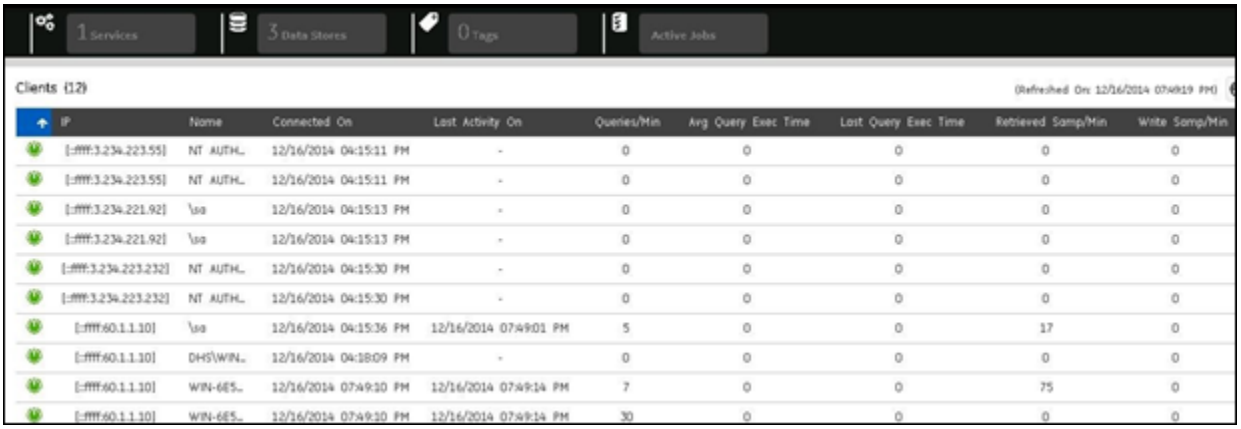
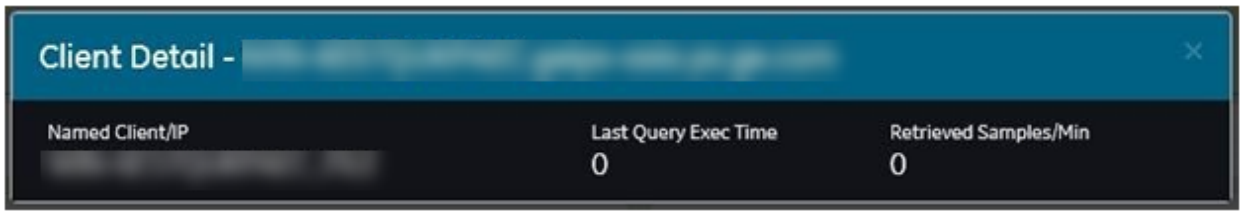



**Table 408. Client Statistics**

| Field | Description |
|---|---|
| Connection | Indicates the status of the current connection. |
| Machine Name | The host name from where the client is connected. |
| Log-In User Name | The user name with which the client is connected. |
| Connected On | The date and time when the connection was established. |
| Last Activity | On The time when the last read and write request was made. |
| Queries/Min | The current rate at which the query requests are made. |
| Average Query Execution Time | The average time taken to execute a query. |
| Timed Out Reads | The number of timed out read requests made by clients. This counter increments when a query made by a client is taking longer time than "Max Query Time" or returning more samples than specified in the "Max Query Intervals". |
| Samples Written/Min | The current rate at which the data samples are written. |
| Details | Select this button to view more client details. To view all the clients, select the Show All Clients button. The All Clients page appears |

**Client Details**

This page, which is accessed from the Client Statistics panel in the dashboard, shows the following additional detail for a client.




| Field | Description |
|---|---|
| Named Client/IP | The name of the client or IP address from where the client is connected. |
| Last Query Exec Time | The time taken to execute the last query. |
| Retrieved Samples/Min | The current rate at which the data samples are received. For more information, refer to the Client Statistics topic. |

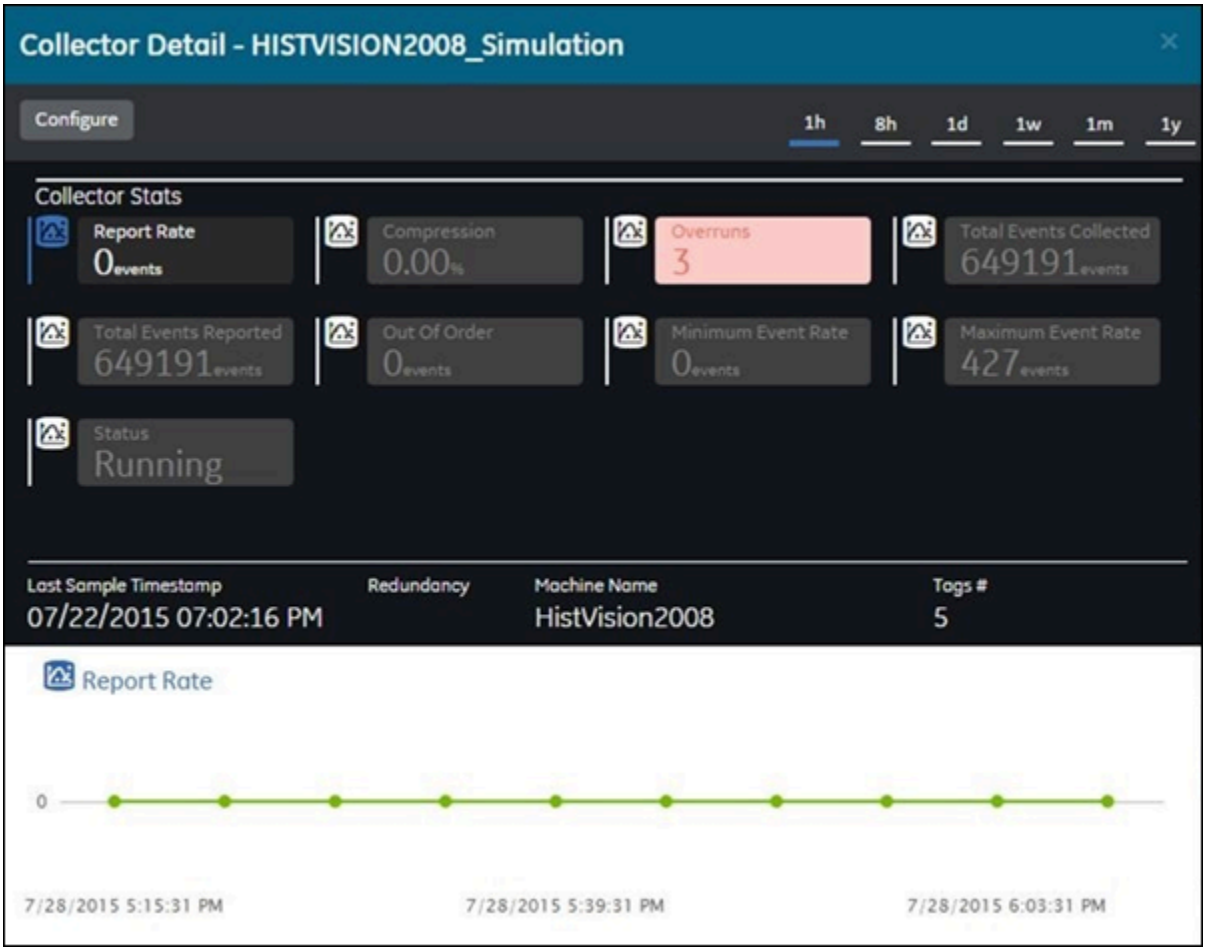
## Collector Panel

The Collector panel in the Dashboard shows details of all the collectors whose performance does not meet the required performance status. The collector statistics panel displays data described in the following table. Select the Show All Collectors button to view all the collectors connected to the system. The View All Collectors page is displayed.

| Field | Description |
|---|---|
| Connection | Indicates the status of the current connection. "Running" (Green) indicates that the collector is operating. "Stopped" (Red) indicates that it is in pause mode and not collecting data. "Unknown" indicates that status information about the collector is unavailable at present, perhaps as a result of a lost connection between collector and server. |
| Name | The name of the computer the collector is running on. |
| Overruns Since Yesterday | The overruns in relation to the total events collected for the past 24 hours. This value is calculated by using the following equation: `OVERRUN_PCT = OVER-` |

| Field | Description |
|---|---|
| | `RUNS / ( OVERRUNS + TOTAL_EVENTS_COLLECTED )`. Overruns are a count of the total number of data events not collected on their scheduled polling cycle. In normal operation, this value should be zero. You may be able to reduce the number of overruns on the collector by increasing the tag collection intervals (per tag). |
| Out of Order Since Yesterday | The number of samples within a series of timestamped data values normally transmitted in sequence have been received out of sequence for the past 24 hours. This field applies to all collectors. Even though events are still stored, a steadily increasing number of out of order events indicates a problem with data transmission that you should investigate. For instance, a steadily increasing number of out of order events when you are using the OPC Collector means that there is an out of order between OPC Server and the OPC Collector. This may also cause an out of order between the OPC Collector and the Data Archiver, but that is not what this statistic indicates. |
| Reconnects/Day | Displays the number of reconnections that happened in the last 24 hours. |
| Last Samples Received | Displays the delayed data sample duration. |
| Comments | Displays the comments if any. |
| Details | Select this button to view the Collector Detail Diagnostics |

Selecting the Details option for a particular collector pops up a display that shows the current statistics on the operation of that selected data collector.




To view or modify the configuration details of the collector, select the **Configure** button. The statistics displayed on this page are computed independently on various time scales and schedules. As a result, they may update at different times. You can choose the time scales by selecting on the time options provided on the top right area of the page.

The collector detail diagnostics page has the following types of fields.

- Trendable Fields
  - These fields can be trended.
  - These fields can be distinguished by the trend icon next to the field name.
  - To graphically view a particular trendable field based on the timestamp, select on the field name.
- Non-Trendable Fields
  - These fields cannot be trended.
  - These fields have no trend icon next to the field name.

**Table 409. Trendable Fields**

| Field | Description |
|---|---|
| Report Rate | This display is a trend chart that displays the average rate at which data is coming into the server from the selected collector. This is a general indicator of load on the Historian collector. Since this chart displays a slow trend of compressed data, it may not always match the instantaneous value of Report Rate displayed in the Collector panel of the System Statistics page. |
| Compression | This display is a trend chart that displays the effectiveness of collector compression. If the chart displays a low current value, you can widen the compression deadbands to pass fewer values and increase the effect of compression. |
| Overruns Percent | This trend chart displays the value at which data overruns are occurring. This value is calculated by the following equation: `OVERRUN_PCT =OVERRUNS / ( OVERRUNS + TOTAL_EVENTS_COLLECTED )` Overruns are a count of the total number of data events not collected. Under normal conditions, the current value should always be zero. If the current value is not zero, which indicates that data is being lost, you should take steps to reduce peak load on the system, by increasing the collection interval. |
| Total Events Collected | Counts the total number of events collected from the data source by the collector. |
| Total Events Reported | Counts the total number of events reported to the Historian archive from the collector. This number may not match the Total Events Collected field due to collector compression. |
| Out of Order | The number of samples within a series of timestamped data values normally transmitted in sequence have been received out of sequence since collector startup. This field applies to all collectors. Even though events are still stored, a steadily increasing number of out of order events indicates a problem with data transmission that you should investigate. For instance, a steadily increasing number of out of order events when you are using the OPC Collector means that there is an out of order between the OPC Server and the OPC Collector. This may also cause an out of order between the OPC Collector and the data archiver but that is not what this statistic indicates. |
| Minimum Event Rate | Specifies the minimum number of data samples per minute sent to the archiver from all sources. |

**Table 409. Trendable Fields (continued)**

| Field | Description |
|---|---|
| Maximum Event Rate | Specifies the maximum number of data samples per minute sent to the archiver from all sources. |
| Status | The current status of collection. "Running" indicates that the collector is operating. "Stopped" indicates that it is in pause mode and not collecting data. "Unknown" indicates that status information about the collector is unavailable at present, perhaps as a result of a lost connection between collector and server |

**Table 410. Non-Trendable Fields**

| Field | Description |
|---|---|
| Last Sample Timestamp | Displays when the last data sample was written. |
| Redundancy | Indicates whether collector redundancy is enabled or disabled. |
| Machine Name | Displays the machine name where the collector is running. |
| Tag# | Displays the number of tags added to the collector. |

## Data Node Panel

The Data Node panel displays the basic information of the Historian Data Archivers (Primary Node plus Mirrors) currently available.

The Data Node panel displays data as described in the following table:

| Field | Description |
|---|---|
| Connection | Indicates the status of the current connection. "Running" (Green) indicates that the Data Archiver is active. "Stopped" (Red) indicates that the Data Archiver is inactive. "Pause" indicates that the user has manually paused it. |

| Field | Description |
|---|---|
| Name | The logical name of the service. |
| IP | The IP address or the host name of the service. |
| Reason | Displays why the Data Node is not performing as expected. The reason is a warning message for the disk running out of space or memory. To view the details of the problem, move the mouse over the reason. |
| Details | Select this button to view the Data Node Detail Diagnostics. |

**Data Node Detail Diagnostics**

The Data Node Detail diagnostics displays the data described in the following table. For each field, you can see the graphical representation of the values based on the time scales selected.

**Note:**
The statistics displayed on this page are computed independently on various time scales and schedules. As a result, they may update at different times.

You can choose the time scales by selecting on the time options provided on the top right area of the page. To graphically view a particular parameter based on the timestamp, select the chart icon of the parameter. Select the data store available with the selected data node to view its disk free space and its statistics.

**Table 411. Utilization Stats**

| Field | Description |
|---|---|
| Memory Utilization | Indicates how much server memory is being consumed. |
| Write Cache Hit Ratio | The hit ratio of the write cache in percent of total writes. It is a measure of how efficiently the system is collecting data and should typically range from 95 to 99.99%. If the data is changing rapidly over a wide range, however, the hit percentage drops significantly because current values differ from recently cached values. More regular sampling may increase the hit percentage. Out of order data also reduces the hit ratio. |
| Archive Compression | The current effect of archive data compression. If the value is zero, it indicates that archive compression is either ineffective or turned off. To increase the effect of data compression, increase the value of archive compression deadbands on individual tags in the Tag Maintenance page to activate compression. In computing the effect of archive compression, Historian counts internal system tags as well as data source tags. Therefore, when working with a very small number of tags and with compression disabled on data source tags, this field may indicate a value other than zero. If you use a realistic number of tags, however, system tags will constitute a very small percentage of total tags and will therefore not cause a significant error in computing the effect of archive compression on the total system. |
| Failed Writes | The number of samples that failed to be written. Since failed writes are a measure of system malfunctions or an indication of offline archive problems, the value shown in the display should be zero. If you observe a non-zero value, investigate the cause of the problem and take corrective action. The Historian also generates a message if a write fails. Note that the message only appears once per tag, for a succession of failed writes associated with that tag. For example, if the number displayed in this field is 20, but they all pertain to one Historian tag, you will only receive one message until the Historian tag is healthy again. |
| Out of Order | The number of samples within a series of timestamped data values normally transmitted in sequence have been received out of sequence since collector startup. This field applies to all collectors. Even though events are still stored, a steadily increasing number of out of order events indicates a problem with data transmission that you should investigate. For example, a steadily increasing number of out of order events when you are using the OPC Collector |

**Table 411. Utilization Stats (continued)**

| Field | Description |
|---|---|
| | means that there is an out of order between the OPC Server and the OPC Collector. This may also cause an out of order between the OPC Collector and the data archiver but that is not what this statistic indicates. |

**Table 412. Process Rate**

| Field | Description |
|---|---|
| Read Queue Rate | Specifies the number of read requests processed per minute, that came into the archiver from all clients. A read request can return multiple data samples. |
| Write Queue Rate | Specifies the number of write requests processed per minute, that came into the archiver from all clients. A write request can contain multiple data samples. |
| Message Queue Rate | Specifies the number of messages processed per minute. |

**Table 413. Queue Size**

| Field | Description |
|---|---|
| Read Queue Size | Displays the total number of messages present in the Read queue. |
| Write Queue Size | Displays the total number of messages present in the Write queue. |
| Message Queue Size | Displays the total number of messages present in the Message queue. |

**Table 414. Disk Usage**

| Field | Description |
|---|---|
| Disk Free Space | The amount of disk space (in MB) left in the current archive. |
| Est. Days to Fill | The amount of time left before the archive is full, based on the current consumption rate. At that point, a new archive must be opened (could be automatic). To increase the days to full, you must reduce the Consumption Rate as noted above.<br><br>To ensure that collection is not interrupted, you should make sure that the Automatically Create Archives option is enabled in the **Data Store Maintenance** page (the **Global Options** section). |

**Table 414. Disk Usage (continued)**

| Field | Description |
|---|---|
| | You may also want to enable Overwrite Old Archives if you have limited disk capacity. Enabling overwrite, however, means that some old data will be lost when new data overwrites the data in the oldest online archive. Use this feature only when necessary.<br><br>The Estimated Days Until Full field is dynamically calculated by the server and becomes more accurate as an archive gets closer to completion. This number is only an estimate and will vary based on a number of factors, including the current compression effectiveness. The System sends messages notifying you at 5, 3, and 1 days until full. |
| Consumption Rate | The speed at which you are using up archive disk space. If the value is too high, you can reduce it by slowing the poll rate on selected tags or data points or by increasing the filtering on the data (widening the compression deadband to increase compression). |

## Configuration Panel

The Configuration page displays the current system status and performance statistics of Collectors *(on page 2822)* and Clients *(on page 2820)*, and allows configuration of the various components of the Historian system. This page displays the data described in the following table.

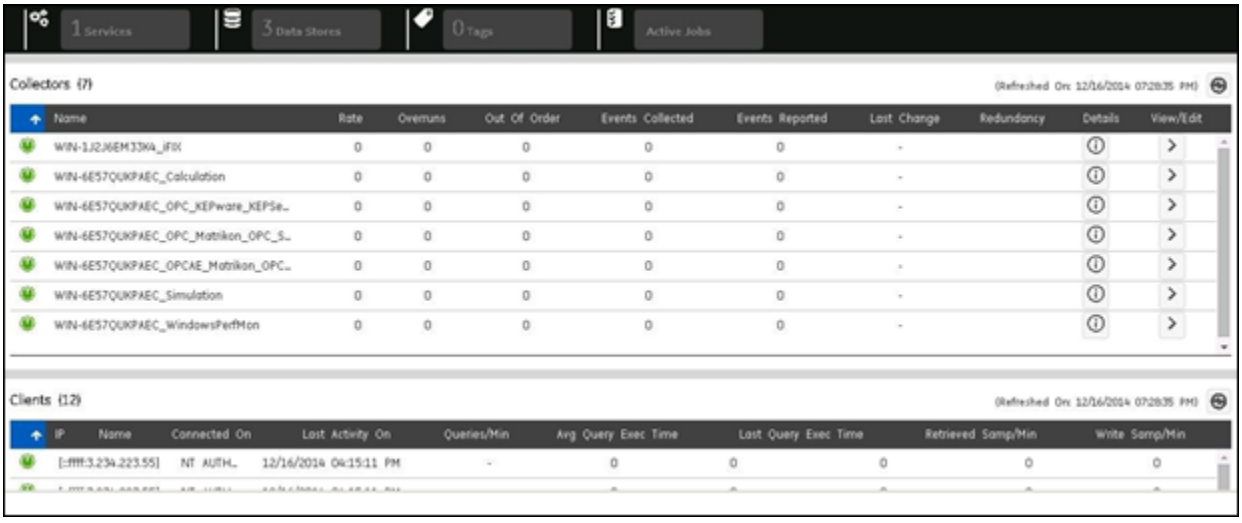


**Table 415. Configuration Page Fields**

| Field | Description |
|---|---|
| Services | Displays the number of services running. For more information, refer to the Configure Services *(on page 2834)* section. |
| Data Stores | Displays the number of data stores configured in the system. For more information, refer to the Data Stores *(on page 2833)* section. |
| Tags | The number of tags available with the data archiver. For more information, refer to the Tags *(on page 2835)* section. |
| Active Jobs | Displays the number of current active jobs in the system. For more information, refer to the Jobs Page *(on page 2834)* section. |
| Collectors | Displays the details of all the collectors available. For configuration of Collectors *(on page 2822)*, refer to the Configure Collectors section. |
| Clients | Displays the details of all the clients present in the system. For more information, refer to the Clients *(on page 2820)* section. |

**Configure Collectors**

The Collectors panel in the Configuration page displays a mix of similar and additional data as the Dashboard, as described in the following table.

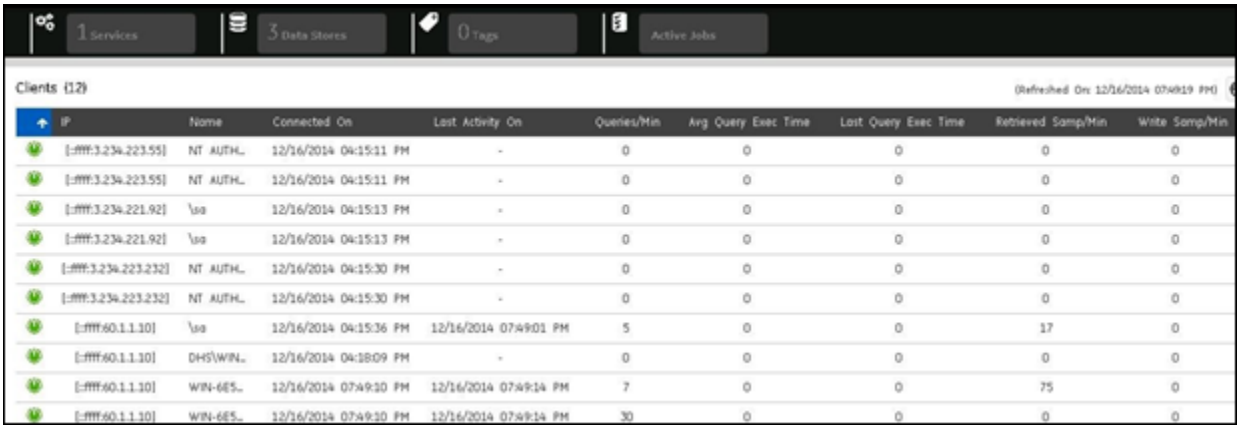
| Field | Description |
|---|---|
| Connection | Indicates the status of the current connection. "Running" indicates that the collector is operating. "Stopped" indicates that it is in pause mode and not collecting data. "Unknown" indicates that status information about the collector is unavailable at present, perhaps as a result of a lost connection between the collector and server. |
| Name | The collector ID, which is used to identify the collector in an Historian system. |
| Rate | The current rate in number of samples/minute at which the server is receiving data from the collector. It is a measure of the collection rate and also of data compression activity. A value equal to the data acquisition rate, when Collector Compression Percent is zero, indicates that every data value received from the data source is being reported to the server. This means that the collector is not performing any data compression. You can lower the report rate, and make the system more efficient, by increasing the data compression at |

| Field | Description |
|---|---|
| | the collector. To do this, widen the collection compression deadbands for selected tags. |
| Overruns | The overruns in relation to the total events collected since startup. This value is calculated by using the following equation: `OVERRUN_PCT = OVERRUNS / ( OVERRUNS + TOTAL_EVENTS_COLLECTED )` Overruns are a count of the total number of data events not collected on their scheduled polling cycle. In a normal operation, this value should be zero. You may be able to reduce the number of overruns on the collector by increasing the tag collection intervals (per tag). |
| Out of Order | The number of samples within a series of timestamped data values normally transmitted in sequence that have been received out of sequence since collector startup. This field applies to all collectors. Even though events are still stored, a steadily increasing number of out of order events indicates a problem with data transmission that you should investigate. For example, a steadily increasing number of out of order events when you are using the OPC Collector means that there is an out of order between the OPC Server and the OPC Collector. This may also cause an out of order between the OPC Collector and the data archiver but that is not what this statistic indicates. |
| Events Collected | Counts the total number of events collected from the data source by the collector. |
| Events Reported | Counts the total number of events reported to the Historian archive from the collector. This number may not match the Total Events Collected field due to collector compression. |
| Last Change | The timestamp when the last collection happened. |
| Redundancy | Displays the current redundancy status of the collector. "Active" state indicates that the collector is currently collecting data and "Standby" indicates that the collector is the standby for the primary collector.<br> **Note:**<br>This status will be displayed only when the Redundant Collector property of the collector is Enabled. |
| Details | Select this button to view the Collector Detail Diagnostics. . |

| Field | Description |
|---|---|
| View/Edit | Select this button to examine or modify configuration parameters for any collector in your system. For more information, refer Collector Details Page. |

**Configure Clients**

This page shows all clients in the system. For more information on the fields, refer to the field and description table in the Client Statistics *(on page 2820)* topic.




## Data Stores Page

The data store Information page lets you read, add, rename, and delete the data stores. To view the data stores, select the Configuration link, and then select **Data Stores**. The **Data Stores** page appears and displays the list of available data stores and their details. For more information, refer to the **Configure Data Stores** section.

The following table describes the fields available in the **Data Stores** page:

| Field | Description |
|---|---|
| Data Store Name | Displays the name of the data store. |
| Type | Indicates whether the storage type is Historical or SCADA buffer. |
| Description | Displays the description of the data store. |
| Is Default | Indicates whether the data store is the default store. Select Yes if you want to set the data store as the default data store. |
| Number of Tags | Displays the number of tags the data store contains. |

| Field | Description |
|---|---|
| State | The state that the data store is in. The data store state is always running until you delete the data store. |
| Edit | Select this button to edit the data store configuration. |
| Delete | Select this button to delete the data store. |
| Details | Select this button to view the archive details of the data store. |

## Jobs Page

When you add a mirror node, replicate a node, re-synch an archive file, back up an archive file or restore an archive file then a job is initiated with a Job ID, which will be seen on the user interface of the Historian Web Admin console.

To view the jobs, you can select the Active Jobs link in any of the Collector, Client, or Configuration pages. You can view the number of jobs that are currently running on the system from this page and can also search a job based on its Job ID by entering the Job ID and selecting the search button in the search field.

| Field | Description |
|---|---|
| Status | Displays the current status of the job (Succeeded, Failed and In Progress) in different color codes. If you see a Failed job, you can expand it to see the description of the failure. |
| Job ID | The unique ID given to a particular job. |
| Type | The type of job. |
| Description | The description of the job. |
| Percentage | The percentage of completion of the job. |
| Start Time | The time at which the job started. |
| Complete Time | The time at which the job was completed. |

## Services Page

The Services page displays all the services running in the system. You can also add a mirror node from the Services page. To refresh the services page, select the Refresh icon.

By default, a node will have four services installed during creation:

- Client Manager
- Configuration Manager
- Data Archiver
- Diagnostic Manager

**Table 416. Service Configuration**

| Field | Description |
|---|---|
| Computer Name | The name of the computer the service is currently running. |
| Service Name | The name of the service and its current status. |
| Port | The port number. After you change the port number of a service, ensure that it is updated in the following registry key: `HKEY_LOCAL_MACHINE\SOFTWARE\Intellution, Inc.\iHistorian\Services`, and restart the service. |
| Type | The type of the node service. |

## Editing a Service

1. Select the **Edit** icon.
   The Edit Service Configuration window appears.
2. Modify the configuration details.
3. Select **Save**.

## Tags Page

To display the Tags page, select the Tags link in any of the Collectors, Clients, or Configuration page. The Tags page lets you read and modify all tag parameters of the Historian system. To access information on a specific tag or group of tags, however, you must first search for the tags.

You can search for tags in the Historian Tag Database by selecting the Search button or the Advanced Search button. You can also add tags manually or from the collector by selecting the appropriate icons on the Tags page.

The Tags page has two sections: Tag Viewer and Tag Editor.

**Tag Viewer**

Displays all the tags available in the system. You can choose to display the total number of tags you want to view by selecting the Show entries. To view all the tags, select the page numbers available at the bottom right of the section.

From this section, you can perform the following operations on the tags:

- Add Tag
- Delete Tag
- Rename Tag
- Search for Tags
- Filtering Tags
- Copy Tag
- Display a Tag

To select a tag, select on the tag name on the page. You can select multiple tags at a time. To clear your selection, select the Clear Selections button. To edit the tag parameters, select the tag and select the Edit button. The tag details are displayed in the Tag Editor. If you select multiple tags and select the Edit button, then the details of the first tag are displayed and when you update, **all** of the common parameters of the selected tags are updated.

**Tag Editor**

In this section, you can view and edit specific tag parameters and options. To view all the tags, select the page numbers available at the bottom right of the section.

The Tag Editor allows you to edit specific tag parameters and options for one or more selected tags. To modify the values, enter new values in the appropriate fields and then select the Update button at the bottom of the page to apply the changes. The Update button, when selected, applies all parameter changes you have made on any tabs in this page. If you want to cancel changes and return to the original values or settings, open a different page and then return to the Tag Details page without selecting the Update button. For more information, refer to the following sections.

## The Advanced Section

To display or edit advanced parameters, select **Advanced**. To modify the values, enter new values in the appropriate fields and then select the Update button at the bottom of the page to apply the changes. Until you select the Update button, entering a new value changes the display of the field name to blue. The fields in the **Advanced** section contain the following information:

**Table 417. Data Collection Options**

| Field | Description |
|---|---|
| Time Assigned By | The source of the timestamp for a data value is either the collector or the data source. All tags, by default, have their time assigned by the collector. When you configure a tag for a polled collection rate, the tag is updated based on the collection interval. For example, if you set the collection interval to 5 seconds with no compression, the archive is updated with a new data point and timestamp every 5 seconds, even if the value is not changing. However, if you change the Time Assigned By field to Source for the same tag, the archive updates only when the device timestamp changes. For example, if the poll time is still 5 seconds, but the timestamp on the device does not change for 10 minutes, no new data is added to the archive for 10 minutes.<br><br>**Note:**<br>This field is disabled for Calculation and Server-to-Server tags. |
| Time Zone Bias | The number of minutes from GMT that should be used to translate timestamps when retrieving data from this tag. For example, the time zone bias for Eastern Standard time is -300 minutes (GMT-5). This property is not used during collection. Use this option if a particular tag requires a time zone adjustment during retrieval other than the client or server time zone. For example, you can retrieve data for two tags with different time zones by using the tag time zone selection in the iFIX chart. |
| Time Adjustment | If the Server-to-Server collector is not running on the source computer, select the Adjust for Source Time Difference option if you want to compensate for the time difference between the source archiver computer and the collector computer. The Time Adjustment field applies only to Server-to-Server tags that use a polled collection type. |
| Data Store | The data store to which the tag belongs. |

## The Collection Section

To display or edit collection parameters, select **Collection**. To modify the values, enter new values in the appropriate fields and then select the Update button at the bottom of the page to apply the changes. Until you select the Update button, entering a new value changes the display of the field name to blue. The fields in the **Collection** section contain the following information:

**Table 418. Data Source**

| Field | Description |
|---|---|
| Collector | The name of the collector for the selected tag. Select the drop-down arrow to display a list of all collectors. |
| Source Address | The address for the selected tag in the data source. Select the Browse button (...) to display a browse window. Leave the Source Address field blank for Calculation and Server-to-Server tags.<br><br>**Note:**<br>When exporting or importing tags using the EXCEL Add-In, the Calculation column, not the SourceAddress column, holds the formulas for the Calculation or Server-to-Server tags. |
| Data Type | A list of data types.<br><br>**Note:**<br>If you change the data type of an existing tag between a numeric and a string or binary data type (and vice versa), the tag's compression and scaling settings will be lost. |
| Data Length | The number of bytes for a fixed string data type. This field is active only for fixed string data types. This field is adjacent to the Data Type field. |
| Enumerated Set Name | The name of the Enumerated Set that can be assigned to the tags. Select the Browse button (...) to display the Define Enumerated Set window. |
| Is Array Tag | Indicates the tag is an array tag. |
| Calc Type | Indicates the type of tag. Analytic Tag, Raw tag or Expression Tag. |

**Choosing a Data Type**

The main use of the scaled data type is to save space. Instead of using 4 bytes of data, it uses only 2 bytes. The scaled data type accomplishes this by storing the data as a percentage of the EGU limit. This saving of space results in a loss of precision. Because of the way that the scaled data type stores data, changing of the EGU limits will result in a change in the values that are displayed. For example, if the original EGU values were 0 to 100 and a value of 20 was stored using the scaled data type and if the EGUs are changed to 0 to 200 at a later date, that value of 20 will be represented as 40.

**Table 419. Collection Options**

| Field | Description |
|---|---|
| Collection | Select the appropriate option to enable or disable collection for this tag. The default setting is Enabled. If you disable collection for the tag, Historian stops collecting data for the tag, but does not delete the tag or any data. |
| Collection Type | Select the type of data collection used for this tag, which can be polled or unsolicited. Polled means that the data collector requests data from the data source at the collection interval specified in the polling schedule. Unsolicited means that the data source sends data to the collector whenever necessary (independent of the data collector polling schedule). |
| Collection Interval | Enter the time interval between readings of data from this tag. With Unsolicited Collection Type, this field defines the minimum interval at which unsolicited data will be sent by the data source. |
| Collection Offset | Used with the collection interval to schedule collection of data from a tag.<br><br>**Note:**<br>The minimum value you can enter in this field is 1000 ms. If you enter a value in milliseconds, note that the value must be in intervals of 1000 ms. For example, 1000, 2000, and 3000 ms are valid values, but 500 and 1500 ms are invalid values. For example, if you want to collect a value for a tag every hour at thirty minutes past the hour (12:30, 1:30, 2:30, and so on), you would enter a collection interval of 1 hour and an offset of 30 minutes. As another example, if you want to collect a value each day at 8am, you would enter a collection interval of 1 day and an offset of 8 hours. |
| Time Resolution | Select the precision for timestamps, which can be either seconds, milliseconds or microseconds. |

## Condition-Based Collection

Condition-based collection is a method to control the storage of data for data tags by assigning a condition. Data is always collected but it is written to the Data Archiver only if the condition is true; otherwise, the collected data is discarded. This condition is driven by a trigger tag; a tag collected by the collector evaluating the condition. Ideally, condition-based collection should be used only with tags that are updating faster than the trigger tag.

**Note:**
Condition-based collection is supported only by the archiver and collectors of Historian version 4.5 and above.

Condition-based collection can be used to archive only the specific data that is required for analysis, rather than archiving data at all times, as the collector is running. For example, if a collector has tags for multiple pieces of equipment, you can stop collection of tags for one piece of equipment during its maintenance.

It is typically used on tags that use fast polled collection but you don't want to use collector compression. While the equipment is running, you want all the data but when the equipment is stopped, you don't want any data stored. The trigger tag will also typically use polled collection. But, either tag could use unsolicited collection.

The condition is evaluated every time data is collected for the data tag. When a data sample is collected, the condition is evaluated and data is either queued for sending to the archiver, or discarded. If the condition cannot be evaluated as true or false (for example, if the trigger tag contains a bad data quality or the collector is not collecting the trigger tag), the condition is considered true and the data is queued for sending.

No specific processing occurs when the condition becomes true or false. If the condition becomes true, no sample is stored to the data tag using that condition, but the data tag will store a sample the next time it collects. When the condition becomes false, no end of the collection marker is stored until the data tag is collected. For example, if the condition becomes false at 1:15 and the data tag gets collected at 1:20, the end of collection marker is created at 1:20 and has a timestamp of 1:20, not 1:15.

This condition-based collection is applicable only to the following collectors:

- Simulation Collector
- OPC Collector
- iFIX collector
- PI Collector

Condition-based collection does not apply to alarm collectors.

| Field | Description |
|---|---|
| Condition Based | Select the appropriate option to enable or disable condition-based collection for a tag. The default setting is Disabled. |

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>Trigger Tag</td><td>The name of the tag used in the condition. Use the browse button to select a trigger tag from the list of tags associated with the collector.</td></tr>
<tr><td>Comparison</td><td>Select the appropriate comparison operator from the drop-down list. The following is a list of comparison operator parameters:<br><table><tr><td>< =</td><td>Setting condition as Trigger Tag value less than or equal to the Compare Value.</td></tr><tr><td>> =</td><td>Setting condition as Trigger Tag value greater than or equal to the Compare Value.</td></tr><tr><td><</td><td>Setting condition as Trigger Tag value less than the Compare Value.</td></tr><tr><td>></td><td>Setting condition as Trigger Tag value greater than the Compare Value</td></tr><tr><td>=</td><td>Setting condition as Trigger Tag value equals the Compare Value.</td></tr><tr><td>!=</td><td>Setting condition as Trigger Tag value not the same as the Compare Value.</td></tr></table><br>Collection will resume only when the value of the triggered Undefined tag changes. This is considered as an incomplete configuration, so condition-based collection is turned off and all of the collected data is sent to the archiver.</td></tr>
<tr><td>Compare Value</td><td>Enter an appropriate target value which is compared against the value of the triggered tag. Make sure while using '=' and '!=' comparison parameters, that the format of the compared value and triggered tag are similar. For example, for a float type trigger tag, the compare value must be a float value; otherwise, the condition result is an invalid configuration, condition-based collection is disabled, and all the data is sent to the archiver.</td></tr>
<tr><td>End of Collection Markers</td><td>Select the appropriate option to enable or disable end of collection markers. The default setting is enabled. When the condition becomes false, all the tag’s values are marked as "Bad", and subquality as "ConditionCollectionHalted." Trending and reporting applications can use this information to indicate that the real-world value is unknown after this time until the condition</td></tr>
</table>

| Field | Description |
|---|---|
| | becomes true and a new sample is collected. If disabled, a bad data marker is not inserted when the condition becomes false. |

## Compression Tab

To display or edit compression parameters, select **Compression**. To modify the values, enter new values in the appropriate fields and then select the Update button at the bottom of the page to apply the changes. Until you select the Update button, entering a new value changes the display of the field name to blue.

**Note:**
The **Compression** section is disabled for Array Tags.

The fields in the **Compression** section contain the following information:

**Table 420. Collector Compression**

| Field | Description |
|---|---|
| Collector Compression (Enabled, Disabled) | Select the appropriate option to enable or disable compression at the collector level. Collector compression applies a smoothing filter to incoming data by ignoring incremental changes in values that fall within a deadband centered on the last reported value. The collector reports any new value that falls outside the deadband to the Historian archive and then centers the deadband on the new value. |
| Collector Deadband | The current value of the compression deadband. This value can be computed as a percentage of the span, centered on the data value or given as an absolute range around the data value.<br>**Note:**<br>With some OPC Servers, the whole deadband value is added to **and** subtracted from the last data value. This effectively doubles the magnitude of the deadband compared to other OPC Servers. |
| Engineering Unit | Converts the deadband percentage into engineering units and displays the result. This value establishes the deadband range that is centered around the new value. When enabling Archive Compression or Collector Compres- |

**Table 420. Collector Compression (continued)**

| Field | Description |
|---|---|
| | sion, the Engineering Units field represents a calculated number created to give an idea of how large a deadband you are creating in Engineering Units. The deadband is entered in percent (%) and Historian multiplies that percent by the range (Hi Engineering Units-Lo Engineering Units) to calculate the percentage in Engineering Units. |
| Collector Compression Timeout | Indicates the maximum length of time the collector will wait between sending samples for a tag to the archiver. This time is kept per tag, as different tags report to the archiver at different times. For polled tags, the Collector Compression Timeout value should be in multiples of your collection interval. After the timeout value is exceeded, the tag stores a value at the next scheduled collection interval, and not when the timeout occurred. For example, if you have a 10 second collection interval, a 1 minute compression timeout, and a collection that started at 2:14:00, then if the value has not changed, the value is logged at 2:15:10 and not at 2:15:00. For unsolicited tags, a value is guaranteed in, at most, twice the compression timeout interval. A non-changing value is logged on each compression timeout. For example, an unsolicited tag with a 1 second collection interval and a 30 second compression timeout is stored every 30 seconds. A changing value for the same tag may have up to 60 seconds between raw samples. In this case, if the value changes after 10 seconds, then that value is stored, but the value at 30 seconds (if unchanged) will not be stored. The value at 60 seconds will be stored. This leaves a gap of 50 seconds between raw samples which is less than 60 seconds. Compression timeout is supported in all collectors except the PI collector. |
| Archive Compression (Enabled, Disabled) | Select the appropriate option to enable or disable compression at the Historian archive level. If enabled, Historian applies the archive deadband settings against all reported data from the collector. |
| Archive Deadband | The current value of the archive deadband expressed as a percent of span or an absolute number. Each time the system reports a new value, it computes a line between this data point and the last archived value. The deadband is calculated as a tolerance centered about the slope of this line. When the next data point is reported, the line between the new point and the last archived point is tested to see if it falls within the deadband tolerance calcu- |

**Table 420. Collector Compression (continued)**

| Field | Description |
|---|---|
| | lated for the previous point. If the new point passes the test, it is reported and is not archived. This process repeats with subsequent points. When a value fails the tolerance test, the last reported point is archived and the system computes a line between the new value and the newly archived point, and the process continues. |
| Engineering Unit | Converts the deadband percentage into engineering units and displays the result. This value establishes the deadband range that is centered on the new value. When enabling Archive Compression or Collector Compression, the Engineering Units field represents a calculated number created to give an idea of how large a deadband you are creating in Engineering Units. The deadband is entered in percent (%) and Historian multiplies that percentage by the range (Hi Engineering Units-Lo Engineering Units) to calculate the percentage in Engineering Units. |
| Archive Compression Timeout | Indicates the maximum length of time from the last stored point before another point is stored, if the value does not exceed the archive compression deadband. The data archiver treats the incoming sample after the timeout occurs as if it exceeded compression. It then stores the pending sample. For more information on Collector and Archive Corruption, refer to the Notes on Collector and Archive Compression topic. |

To determine how your specific server handles deadband, refer to the documentation of your OPC Server. Example: The engineering units are 0 to 200. The deadband value is 10%, which equals 20 units. If the deadband value is 10% and the last reported value is 50, the value will be reported when the current value exceeds 50 + 10 = 60 or is less than 50-10 = 40. Note that the deadband (20 units) is split around the last data value (10 on either side.) Alternatively, you could specify an absolute deadband of 5. In this instance, if the last value was 50, a new data sample will be reported when the current value exceeds 55 or drops below 45. If compression is enabled and the deadband is set to zero, the collector ignores data values that do not change and records any that do change. If you set the deadband to a non-zero value, the collector records any value that lies outside the deadband. If the value changes drastically, a pre-spike point may be inserted. See the Spike Logic section for more details.

## Understand Collector and Archive Compression

This section describes the behavior of collector and archive compression. Understanding these two Historian features will help you apply them appropriately to reduce the storage of unnecessary data. Smaller archives are easier to maintain and allow you to keep a greater time span of historical data online.

Collector compression applies a smoothing filter, inside the collector, to data retrieved from the data source. By ignoring small changes in values that fall within a deadband centered on the last reported value, only significant changes are reported to the archiver. Fewer samples reported, yields less work for the archiver and less archive storage space used. The definition of significant changes is determined by the user by setting the collector compression deadband value.

For convenience, the Historian Web Admin console calculates and shows the deadband in engineering units if you enter a deadband percentage. If you later change the high and low EGU limits, the deadband is still a percentage, but of the new limits. A 20% deadband on 0 to 500 EGU span is 100 engineering units. If you change the limits to 100 and 200, then the 20% is now 20 engineering units.

The deadband is centered on the last reported sample, not simply added to it or subtracted. If your intent is to have a deadband of 1 unit between reported samples, you need a compression deadband of 2, so that it is one to each side of the last reported sample. In an example of 0 to 500 EGU range, with a deadband of 20%, the deadband is 100 units, and the value has to change by more than 50 units from the last reported value.

Changes in data quality from good to bad, or bad to good, automatically exceed collector compression and are reported to the archiver. Any data to that comes to the collector out of time order will also automatically exceed collector compression.

It is possible for collected tags with no compression to appear in Historian as if the collector or archive compression options are enabled. If collector compression occurs, you will notice an increase in the percentage of the Compression value from 0% in the Collectors panel of the System Statistics page in Historian Administrator. When archive compression occurs, you will notice the Archive Compression value and status bar change on the System Statistics page.

For all collectors, except the File collector, you may observe collector compression occurring for your collected data (even though it is not enabled) if bad quality data samples appear in succession. When a succession of bad data quality samples appears, Historian collects only the first sample in the series. No new samples are collected until the data quality changes. Historian does not collect the redundant bad data quality samples, and this is reflected in the Collector Compression percentage statistic.

For a Calculation or Server-to-Server Collector, you may possibly observe collector compression (even though it is not enabled) when calculations fail, producing no results or bad quality data.

The effect of Collector Compression Timeout is to behave, for one poll cycle, as if the collector compression feature is not being used. The sample collected from the data source is sent to the archiver. Then the compression is turned back on, as configured, for the next poll cycle with new samples being compared to the value sent to the archiver.

**Archive Compression**

Archive compression can be used to reduce the number of samples stored when data values for a tag form a straight line in any direction. For a horizontal line (non-changing value), the behavior is similar to collector compression. But, in archive compression, it is not the values that are being compared to a deadband, but the slope of line those values produce when plotted against time.

Archive compression logic is executed in the data archiver and, therefore, can be applied to tags populated by methods other than collectors. Archive compression can be used on tags where data is being added to a tag by migration, or by the File collector, or by an SDK program for example.

Each time the archiver receives a new value for a tag, the archiver calculates a line between this incoming data point and the last archived value. The deadband is calculated as a tolerance centered about the slope of this line. The slope is tested to see if it falls within the deadband tolerance calculated for the previous point. If the new point does not exceed the tolerance, it is held by the archiver rather than being archived to disk. This process repeats with subsequent points. When an incoming value exceeds the tolerance, the value held by the archiver is written to disk and the incoming sample becomes held.

The effect of the archive compression timeout is that the incoming sample is automatically considered to have exceeded compression. The held sample is archived to disk and the incoming sample becomes the new held sample. If the Archive Compression value on the System Statistics page indicates that archive compression is occurring, and you did not enable archive compression for the tags, the reason could be because of internal statistics tags with archive compression enabled.

**Note:**
Array tags do not support Archive and Collector Compression. If the tag is an array tag, then the **Compression** tab is disabled.

## General Tab

To display or edit the general parameters listed below, select **General**. To modify the values, enter new values in the appropriate fields, and then select the Update button at the bottom of the page to apply the changes. Until you select the Update button, entering a new value changes the display of the field name to blue.

**Table 421. Description Panel**

| Field | Description |
|---|---|
| Description | The tag description of the selected tag. |
| EGU Description | The engineering units, if any, assigned to the selected tag. Often referred to as Unit of Measure, or UoM. |
| Comment | Comments, if any, that apply to the selected tag. |
| StepValue | This tag property is used to indicate that the actual measured value changes in a sharp step instead of a smooth linear interpolation. This option should be selected only for numeric data. Enabling this option affects only data retrieval; it has no effect on data collection or storage |
| Spare Configuration | The Spare 1 through Spare 5 fields list any configuration information stored in these fields. |

**Note:**
Do not add or update the following spare configurations as the data may get corrupted or over written:

- The **Spare 5** field for Server to Server Collector and Server to Server Distributor.
- The **Spare 1** field for OSI PI Distributor.

## The **Scaling** Section

Scaling converts a data value from a raw value expressed in an arbitrary range of units, such as a number of counts, to one in engineering units, such as gallons per minute or pounds per square inch. The scaled data type can serve as a third form of data compression, in addition to collector compression and archive compression, if it converts a data value from a data type that uses a large number of bytes to one that uses fewer bytes. To display or edit scaling parameters, select **Scaling**. To modify the values, enter new values in the appropriate fields and then select the Update button at the bottom of the page to apply the changes. Until you select the Update button, entering a new value changes the display of the field name to blue. The fields in the **Scaling** section contain the following information:

**Table 422. Input Scaling**

| Field | Description |
|---|---|
| Input Scaling | Select the appropriate option to enable or disable input scaling, which converts an input data point to an engineering units value. |
| Hi Scale Value | The upper limit of the span of the input value. Lo Scale Value The lower limit of the span of the input value. |

For example, to rescale and save a 0-4096 input value to a scaled range of 0-100, enter 0 and 4096 as the low and high input scale values and 0 and 100 as the low and high engineering units values, respectively. If a data point exceeds the high or low end of the input scaling range, then Historian logs a bad data quality point with a ScaledOutOfRange subquality. In the previous example, if your input data is less than 0, or greater than 4096, then Historian records a bad data quality for the data point. For example, a value of 4097, in this case, yields a bad data quality.

**OPC Servers and TRUE Values**

Some OPC Servers return a TRUE value as -1. If your OPC Server is returning TRUE values as -1, modify the following scaling settings in the **Tag Maintenance** page of Historian Administrator:

```
Hi Engineering Units = 0

Lo Engineering Units = 1

Hi Scale Value = 0

Lo Scale Value = -1

Input Scaling = Enabled
```

# Configure General Collector Options

## Configure General Collector Options

You can modify collector configurations from the following sections of the Collector Configuration Screen.

- Action Buttons
- Configuration Tab
- Defaults Tab
- Performance Tab
- Redundancy Tab

**Table 423. Action Buttons**

| Button | Action |
|---|---|
| | Resumes collection of the collector. |
| | Pauses the collection of the collector |
| | Refreshes the page |
| Update | Updates the changes made. |

**Table 424. The Performance Section**

| Field | Description |
|---|---|
| Report Rate | This display is a trend chart that displays the average rate at which data is coming into the server from the selected collector. This is a general indicator of load on the Historian collector. Since this chart displays a slow trend of compressed data, it may not always match the instantaneous value of Report Rate displayed in the Collector panel of the System Statistics page. |
| Out of Order | The number of samples within a series of timestamped data values normally transmitted in sequence that have been received out of sequence since collector startup. This field applies to all collectors. |
| Compression | This display is a trend chart that displays the effectiveness of collector compression. If the chart displays a low current value, you can widen the compression deadbands to pass fewer values and increase the effect of compression. |
| Total Events Collected | Displays the number of events collected from the data source. |
| Total Events Reported | Displays the total number of events reported to the Historian archive from the collector. |
| Overruns | This trend chart displays the value at which data overruns are occurring. This value is calculated by following equation: `OVERRUN_PCT =OVERRUNS / ( OVERRUNS + TOTAL_EVENTS_COLLECTED )` Overruns are a count of the total number of data events not collected. Under normal conditions, the current value should **always** be zero. If the current value is not zero, which indicates that data is |

**Table 424. The Performance Section (continued)**

| Field | Description |
|---|---|
| | being lost, you should take steps to reduce peak load on the system by increasing the collection interval. |

**Table 425. The Defaults Section**

| Field | Description |
|---|---|
| Prefix for Tag Names | Displays a prefix, if any, that is automatically added to all tag names when you browse and pick the specified collector. To change the prefix, enter a new text string and select the Update button at the bottom of the page. This field applies to all collectors except the File and Calculation collectors |
| Collection Type | Indicates whether this collector is configured for polled data collection or unsolicited collection. |
| Collection Interval | The time in milliseconds, seconds, minutes, or hours required to complete a poll of a given tag on the selected collector. It is also used in unsolicited collection. In effect, it specifies how frequently data can be read from a tag. The collection interval can be individually configured for each tag. To change it, enter a new value. NOTE: To avoid collecting repeat values with the OPC Collector when using device timestamps, specify a collection interval that is greater than the OPC Server update rate. |
| Time Assigned By | Indicates whether the timestamp for the data value is supplied by the collector or the data source. To change it, select a different type. |
| | Compression Indicates whether or not collector compression is enabled as a default setting. To change it, select the other option. This option is overridden by Tag settings. |
| Deadband | The default setting of the collector compression deadband in absolute or percentage range values. |
| Timeout | The default setting for the collector compression time-out for tags added through the Add Multiple Tags From Collector window. You must enable the Collector Compression option to use this field. |
| Spike Logic Control | Spike logic monitors incoming data samples for spikes in a tag's values. If spike logic is enabled, a sample of equal value to the previously archived sample is inserted into the archive in front of the spike value. |

**Table 425. The Defaults Section (continued)**

| Field | Description |
|---|---|
| Multiplier | The Multiplier option specifies how much larger a spike value must be than the deadband range before spike logic is invoked. For example, if a value of 3 is entered in the Multiplier field and the deadband percentage is set to 5%, spike logic will not be invoked until the difference between the spike value and the previously archived data point is 15% of the EGU range. |
| Interval | The Interval option specifies how many samples must have been compressed before spike logic is invoked. For example, if the Interval field is set to 4, and 6 values have been compressed since the last archived data sample, spike logic will be invoked. |
| On-Line Tag Config Changes | Enabling this feature allows you to make on the fly changes to tags without having to restart the collector. If you disable this option, any changes you make to tags do not affect collection until you restart the collector executable. |
| Sync Timestamp for Server | Enabling this feature automatically adjusts all outgoing data timestamps to match the server clock. This feature is not active when you configure timestamps to be supplied by the data source. Note, that this does not change collector times to match the server time, it adds or subtracts an increment of time to compensate for the relative difference between the clocks of the server and collector, independent of time zone or day light saving time (DST) differences. If the collector system clock is greater than 15 minutes ahead of the archiver system clock, and the Synchronize Timestamps to Server option is disabled, data will not be written to the archive. |
| Delay Collection @ Startup (Sec) | Permits you to enter the number of seconds to delay collection on startup (after loading its tag configuration). The default is 2 seconds. |

**Note:**
Not all options in this section are available to all collectors. If an option is disabled, it doesn't apply to the current type of collector.

**Table 426. Configuration Tab**

| Field | Description |
|---|---|
| Description | The name of the selected collector. |
| Number of Tags | Displays the number of tags currently added to the Simulation Collector. Edit this field to modify the number of Simulation tags available for addition to the Historian System. |
| Function Period (Seconds) | The period, in seconds, of the SIN, STEP, and RAMP functions implemented in the Simulation collector. The default is 60 seconds. |
| Computer Name | The machine name of the computer that the collector is installed on. |
| Memory Buffer Size | The size of the memory buffer currently assigned to the store and forward function. The memory buffer stores data during short-term or momentary interruptions of the server connection; the disk buffer handles long duration outages. To estimate the size you need for this buffer, you need to know how fast the collector is trying to send data to the server and how long the server connection is likely to be down. With those values and a safety margin, you can calculate the required size of the buffer. NOTE: If you enter a new value for this parameter, the change is effective the next time you restart the collector. |
| Minimum Free Space | The minimum free disk space that must be available on the computer. If the minimum space required is not available when the collector starts, the collector will shut down. |
| Heartbeat Output Address | The address in the source database into which the collector writes the heartbeat signal output. This address should be connected to a writable analog field. For an iFIX data collector, use an iFIX tag for the output address. Enter the address as NODE.TAG.FIELD (for example, MyNode.MyCollector_AO.F_CV). For an OPC Collector, use the OPC address in the server. Refer to your OPC documentation for more information. The data collector writes the value of 1 to this location every 60 seconds while it is running. You could program the iFIX database to generate an alarm if the Heartbeat Output Address is not written to once every 60 seconds, notifying you that the data collector has stopped. |
| Status Output Address | Address in the source database into which the collector writes the current value of the collector status (running, stopping, stopped, unknown, or start- |

**Table 426. Configuration Tab (continued)**

| Field | Description |
|---|---|
| | ing) output, letting an operator or the HMI/SCADA application know the current status of the collector.<br><br>This address should be connected to a writable text field of at least 8 characters. This value is updated only upon a change in status of the collector. For an iFIX data collector, use a TX tag for the output address. Enter the address as NODE.TAG.FIELD (for example, MyNode.MyCollector_TX.A_CV).<br><br>For an OPC Collector, use an OPC address in the server. Refer to your OPC documentation for more information. The text string usually displays Running, Stopped, or Unknown, matching the Status column value displayed in the collector pane in the System Statistics page of Historian Administrator. |
| Rate Output Address | The address in the source database into which the collector writes the current value of the events/minute output, letting an operator or the HMI/SCADA application know the performance of the collector. This should be connected to a writable analog field. The value is written once per minute.<br><br>For an iFIX data collector, use an iFIX tag for the output address. Enter the address as NODE.TAG.FIELD (for example, MyNode.MySIM_AO.F_CV).<br><br>For an OPC Collector, use a writable OPC address in the server. Refer to your OPC documentation for more information. This value displays the same value as the Report Rate field in the collector pane in the System Statistics page of Historian Administrator. |

**Redundancy Tab**

Historian includes support for collector redundancy, which decreases the likelihood of lost data due to software or hardware failures. Implementing collector redundancy ensures that collection of your data remains uninterrupted. Collector redundancy makes use of two or more collectors, gathering data from a single source. Two or more collectors may be configured in a redundant group. All collectors in the group actively gather the same tags from a data source but only the "active" collector forwards its samples to the Historian server. The non-active collectors buffer their data against failover of the active collector. The Historian server actively monitors the health of the redundant collectors and will automatically switch to a backup if certain user-configurable trigger conditions are met.

The **Redundancy** section displays the following information:

| Field | Description |
|---|---|
| Backup Collector | If enabled, specifies that this is a redundant collector. |
| Backup For | Specifies the primary collector.<br><br>**Note:**<br>This configuration will be preserved if you disable collector redundancy. This allows you to temporarily take a redundant collector offline without losing its configuration. |
| Backup Collector Status | The current status of this collector. |
| Backed Up By | The name of the collector providing redundancy for the selected collector. |
| Watchdog Tag | Specifies a tag to use to determine the status of the collector. If the watchdog tag meets any of the conditions specified below, the secondary collector will be brought on line to replace it. |
| Failover on Bad Quality | If enabled, the secondary collector is promoted when a data sample from the watchdog tag is received with bad quality. Failover happens on every write of a bad data sample to the watchdog, not just on the transition from good to bad quality |
| Failover When Value Transitions from Zero | If selected, the secondary collector is promoted when a data sample from the watchdog tag with a non-zero value is received from the primary collector. Failure happens every time a non-zero value is received, not just when the value promotes from zero to a non-zero value |
| Failover When No Value Changes for __ Seconds | If selected, the secondary collector is promoted when no data value changes have been received within the time period specified. This could be tied into a heartbeat status indicator where the value is checked every 5 seconds. To prevent failure, there must be a value change. |

# Maintain, Operate, and Monitor Historian

## Plan For Data Recovery

Planning for data recovery means always having up-to-date backup files for important information to call up and restore quickly when the need arises.

**Key Files**

- The .IHC file contains all configuration information.
- The *.IHA file is the current online archive files.

The .IHC file is automatically backed up when, and only when, you back up the **current** archive .IHA file. The .IHC uses the following naming convention: `ComputerName_Config-Backup.ihc`

By default, the .IHC backup path is the same as the archives path. If the default backup path is different than the archives path, the .IHC file is copied to the backup folder with the standard .IHC naming convention `ComputerName_Config.ihc`

Restoring the IHC file restores the system configuration (tag, archive, and collector configuration) to the state it was in before the event occurred. If you restore the archive file (IHA) along with the configuration file (IHC), you can quickly pick up where you left off when the event occurred with minimum loss of data.

## Develop a Maintenance Plan

The primary goal of a maintenance plan is to maintain integrity of the data collected. If you are successful in this regard, you will always be able to recover from a service interruption and continue operations with minimal or no loss of data. Since you can never ensure 100% system uptime, you must frequently and regularly back up current data and configuration files, and maintain non-current archive files in a read-only state. It is recommended that you use the following guidelines for backup and routine maintenance.

**Daily Maintenance**

On a daily schedule, perform the following backup operations unless you use ihArchiveBackup.exe to back up archives automatically.

1. Use the Historian Web Admin console to back up the current archive and most recent .IHA archived data file. This preserves data collected up to this moment in time. You do not need to back up any read-only archive files after they have been backed up once.
2. Use Windows Explorer to back up the .IHC file if it has been modified by anyone (unless it is backed up automatically). This file contains all current configuration information (tag configuration, archive configuration, and collector configuration). Using this file, you can restore the system configuration after an unplanned shutdown.

## Monitor Historian Performance

Historian provides a variety of performance counters and performance tags that can be used to monitor how well the Historian components are performing. These performance tags or counters can also be used to determine the resource usage on the computer that runs the Historian application.

Use performance tags to view information in an Excel report or SDK program, possibly along with other Historian tags. Use performance counters to view information in Windows Performance monitor, possibly along with non-Historian counter information.

Performance counters are useful when the Historian Web Admin console is not installed or cannot connect. Like any Windows performance counter, you must add the counters for collection to view history. Performance tags are always being collected and you can view past data any time.

Performance counters are updated in real time. Performance tags are updated once per minute with the activity over the last minute.

Performance counters contain more information than tags. Any counter can be collected to a tag using the Historian Windows Performance Collector. Those tags will count against your licensed tag count.

## Historian Performance Tags

The following table provides information about the various Historian Server Performance tags.

**Table 427. Server Performance Tags**

| Tag Name | Description |
|---|---|
| PerfTag_CompressionRatio | Specifies the current effect of archive data compression. |
| PerfTag_MinimumCompression-Ratio | Specifies the minimum compression ratio. |
| PerfTag_MaximumCompression-Ratio | Specifies the maximum compression ratio. |
| PerfTag_TotalEvents | Specifies the total number of data samples reported to the Historian archive from all sources. . |
| PerfTag_TotalOutOfOrder | Specifies the total out of order data samples. |
| PerfTag_AverageEventRate | Specifies the average number of data samples per minute sent to archiver from all sources |
| PerfTag_MinimumEventRate | Specifies the minimum number of data samples per minute sent to archiver from all sources. |
| PerfTag_MaximumEventRate | Specifies the maximum number of data samples per minute sent to archiver from all sources. |

**Table 427. Server Performance Tags (continued)**

| Tag Name | Description |
|---|---|
| PerfTag_WriteCacheHitRatio | Specifies the hit ratio of the write cache in percent of the total writes. |
| PerfTag_TotalFailedWrites | Specifies the total number of samples since startup that failed to be written. |
| PerfTag_TotalMessages | Specifies the total messages (for example, connection or audit messages) received by the archiver since startup |
| PerfTag_TotalAlerts | Specifies the total number of alerts received by the data archiver since startup. |
| PerfTag_FreeSpace | Indicates the free disk space left in the current archive. |
| PerfTag_SpaceConsumptionRate | Specifies an archive disk space consumption rate in megabytes per day. |
| PerfTag_PredictedDaysToFull | Indicates the approximate number of days required for an archive to fill. |
| PerfTag_MemoryUsage | Specifies the amount of RAM used by the Data Archiver. |
| PerfTag_MemoryVMSize | Specifies the amount of virtual memory used by the Data Archiver. |
| PerfTag_TotalAlarms | Specifies the total number of alarms received by the Data Archiver since starting up. |
| PerfTag_AverageAlarmRate | Specifies the average alarm rate in alarms per minute received by Data Archiver. |
| PerfTag_TotalFailedAlarms | Specifies the total number of alarms since startup that failed to be written. |
| Perftag_ReadQueueSize | Specifies the total number of messages present in the Read queue. |
| Perftag_AverageReadRate | Specifies the total number of data samples per minute returned from the Data Archiver for all read requests. |
| Perftag_ ReadQueuePushRate | Specifies the number of read requests per minute that came into the archiver from all clients. A read request can return multiple data samples. |

**Table 427. Server Performance Tags (continued)**

| Tag Name | Description |
|---|---|
| Perftag_ WriteQueuePushRate | Specifies the number of write requests per minute that came into the archiver from all clients. A write request can contain multiple data samples. |

The following table provides information about the various Historian Collector Performance Tags.

**Note:**
Replace the placeholder `%CollectorName%` with the name of a Collector.

**Table 428. Collector Performance Tags**

| Tag Name | Description |
|---|---|
| PerfTag_`%CollectorName%`_InterfaceStatus | Specifies the status of an interface. |
| PerfTag_`%CollectorName%`_InterfaceTotalEventsCollected | Specifies the total number of events collected by an interface. |
| PerfTag_`%CollectorName%`_InterfaceTotalEventsReported | Specifies the total number of events reported by an interface. |
| PerfTag_`%CollectorName%`_InterfaceOutOfOrderEvents | Specifies the total out of order events. |
| PerfTag_`%CollectorName%`_InterfaceAverageEventRate | Specifies the average event rate of an interface. |
| PerfTag_`%CollectorName%`_InterfaceMinimumEventRate | Specifies the minimum event rate of an interface. |
| PerfTag_`%CollectorName%`_InterfaceMaximumEventRate | Specifies the maximum event rate of an interface. |
| PerfTag_`%CollectorName%`_InterfaceOverruns | Specifies the interface overruns. |
| PerfTag_`%CollectorName%`_InterfaceCompression | Specifies the compression of an interface. |
| PerfTag_`%CollectorName%`_InterfaceOverrunsPercent | Specifies the number of overruns in relation to the total events collected since startup. |

## Historian Server Performance Counters

The Windows performance counters are exposed as objects with counters. In the table below, you can see each counter and the object to which it belongs. Each object has one or more instances as shown in the Windows Performance Monitor.

**Table 429. Historian Archive Object Counters**

| Archive Counter | Description |
|---|---|
| Cache Priority | Relative priority of items from the archive stored to the Windows Cache. A higher priority means the item is more likely to stay in cache. |
| Disk Read Time (usec) | Duration of last disk read in microseconds. |
| Disk Read Time Max (usec) | Maximum duration across all disk reads from the archive in microseconds. |
| Disk Reads | Number of disk reads from archive. |
| Disk Write Time (usec) | Time of last disk write in microseconds. |
| Disk Write Time Max (usec) | Maximum duration of all disk writes to the archive in microseconds. |
| Disk Writes | Number of disk writes to archive. |
| File Size (MB) | Size of the archive file in MB. |
| Read Calls | Number of read calls to the archive since startup. |
| Read Rate (Calls/min) | Number of read calls to the archive per minute. |
| Write Count | Number of data samples written to archive since startup. |
| Write Count Rate | Number of data samples written to archive per minute. |
| Writes Compressed | Number of data samples since startup that were compressed writes to the archive. |
| Writes Expensive | Number of data samples since startup that were expensive or slow. |
| Writes Failed | Number of data samples that were failed writes to the archive. |
| Writes OutofOrder | Number of data samples that were out of time order writes to the archive. |

**Table 430. Historian Cache Object Counters**

| Cache Counter | Description |
|---|---|
| Hit Percentage | Hit rate percentage (0-100) for successful data retrieval calls to the cache. Higher numbers represent more efficiency. |
| Hits | Number of hits in the cache since startup. To reset the count, restart the Archiver. |
| Misses | Number of misses in the cache. |
| Num Adds | Total number objects added to cache. |
| Num Deletes | Total number of objects deleted from cache. |
| Num High Prio Objs | Number of high priority objects available for deletion. |
| Num Low Prio Objs | Number of low priority objects available for deletion. |
| Num Med Prio Objs | Number of medium priority objects available for deletion. |
| Obj Count | Number of objects in the cache. |
| Size (KB) | Size of cache in KB. |

**Table 431. Historian DataStores Object Counters**

| DataStores Counter | Description |
|---|---|
| Compression Ratio (Average) | Archive compression ratio for this data store. |
| Compression Ratio (Max) | Maximum archive compression ratio for the data store. |
| Compression Ratio (Min) | Minimum archive compression ratio for the data store. |
| Messages (Total Alerts) | Total alerts since startup |
| Messages (Total) | Total messages since startup. |
| Read Calls | Number of read calls to the data store. |
| Read Rate (Calls/min) | Average read rate across all archives in the data store. (Read Calls/Minute) |
| Read Samp Rate (Samp/min) | Average read rate across all archives in the data store. (Samples/Minute) |

**Table 431. Historian DataStores Object Counters (continued)**

| DataStores Counter | Description |
|---|---|
| Space (Consumption MB/day) | Disk space consumption rate. (MB/day) |
| Space (Days To Full) | Number of days until current archive is full. |
| Space (Free in MB) | Free disk space in the current archive. |
| Write Rate (Average) | Average event rate across all archives. (Samples/Minute) |
| Write Rate (Max) | Minimum event rate across all archives. (Samples/Minute) |
| Write Rate (Min) | Minimum event rate across all archives. (Samples/Minute) |
| Writes (Cache Hit Ratio) | Write Cache hit ratio. |
| Writes (Compressed) | Total number of compressed data samples since startup. |
| Writes (Total Failed) | Total failed data sample writes since startup. |
| Writes (Total OutOf-Order) | Total out of order data samples since startup. |
| Writes (Total) | Total data samples across all archives since startup. |

**Table 432. Historian Overview Object Counters**

| Overview Counter | Description |
|---|---|
| Compression Ratio (Average) | Average archive compression ratio of all data stores. |
| Compression Ratio (Max) | Average maximum compression ratio of all data stores. |
| Compression Ratio (Min) | Average minimum compression ratio of all data stores. |
| Memory Usage (KB) | Private bytes memory usage for Data Archiver. |
| Memory VM Size (KB) | Virtual Bytes memory usage for Data Archiver. |
| Messages (Total Alerts) | Sum of total alerts of all data stores since startup. |
| Messages (Total) | Sum of total messages of all data stores since startup. |
| Read Rate (Calls/min) | Sum of all average read rates of all data stores. (Samples/Minute) |
| Read Samp Rate (Samp/min) | Average read rate across all archives. (Samples/Minute) |
| Space (Consumption MB/day) | Sum of space consumption rate (MB/day) of all data stores. |

**Table 432. Historian Overview Object Counters (continued)**

| Overview Counter | Description |
|---|---|
| Space (Days To Full) | Minimum number of days until current archive is full for all data stores. |
| Space (Free in MB) | Sum of all free space in the current archive of all data stores. |
| Write Rate (Average) | Sum of all average event rates of all data stores. (Samples/Minute) |
| Write Rate (Max) | Sum of all maximum event rates of all data stores. |
| Write Rate (Min) | Sum of all minimum event rates of all data stores. |
| Writes (Cache Hit Ratio) | Average write Cache hit ratio of all data stores. |
| Writes (Compressed) | Sum of total number of compressed data samples of all data stores. |
| Writes (Expensive) | Sum of total number of expensive writes data samples of all data stores. One of the reasons for expensive writes is out-of-order data. |
| Writes (Total Failed) | Sum of total failed data sample writes of all data stores. |
| Writes (Total OutOfOrder) | Sum of total out of order data samples of all data stores. |
| Writes (Total) | Sum of total data samples across all archives of all data stores. |

**Table 433. Historian Queue Object Counters**

| Queue Counters | Description |
|---|---|
| ClientQueues with Msgs | The number of client queues with messages current on them. A lower number means all clients are up to date. A higher number means that the archiver is not up to date with incoming network traffic |
| Count (Max) | Maximum number of messages on the queue. (memory and disk) |
| Count (Total) | Number of messages on the queue. (memory and disk) |
| Disk Buf Msg Reads | Number of messages read from the disk buffer file. |
| Disk Buf Msg Writes | Number of messages written to the disk buffer file. |
| Processed Count | Number of messages processed from the queue since startup. |
| Processed Rate (msg/min) | Recent rate at which messages have been processed for the queue. |
| Processing Time (Ave) | Average time in milliseconds to process a message. |

**Table 433. Historian Queue Object Counters (continued)**

| Queue Counters | Description |
|---|---|
| Processing Time (Last) | Time in milliseconds to process the last message. |
| Processing Time (Max) | Maximum time in milliseconds to process a message. |
| Recv Count (msgs) | Number of messages received into the queue. |
| Recv Rate (msgs/min) | Recent rate at which messages have been received for the queue. |
| Size Kb (Mem&Disk Max) | Max size of messages in Kb on the queue. (memory and disk). |
| Size Kb (Mem&Disk) | Size of messages in Kb on the queue. (memory and disk) |
| Size Kb (Mem) | Size of messages in Kb on the queue. (memory only) |
| Threads | Number of worker threads allocated to process this queue. This is the number of created threads but they may be idle. |
| Threads Working | Number of queue processing worker threads currently processing messages. |
| Time in Queue (Ave) | Average time in milliseconds that a message was in the queue, waiting to be processed. |
| Time in Queue (Last) | Time in milliseconds that the last message was in the queue, waiting to be processed. |
| Time in Queue (Max) | Maximum time in milliseconds that a message was in the queue, waiting to be processed. |

**Table 434. Historian Config Counters**

| Config Counters | Description |
|---|---|
| File Size | The size of the Configuration File in MB |
| Hist Tags(Actual) | Number of the Historical tags in the system |
| Hist Tags (Licensed) | Total licensed Historian tags. |
| Hist Tags(Used) | Effective number of Historical Licensed Tags in the system. Can be greater than the number of tags because some tags count as more than one Licensed Tag. |
| Hist Tags (UsedBy-Arrays) | Effective number of Historical Licensed Array Tags in the system (Not the raw tag count, the effective licensed count). |

**Table 434. Historian Config Counters (continued)**

| Config Counters | Description |
|---|---|
| Hist Tags (UsedByUser-Def) | Effective number of Historical Licensed User Defined Tags in the system (Not the raw tag count, the effective licensed count). |
| Number of Collectors | Number of collectors defined on the system. |
| Number of EnumSets | Number of enumerated sets defined on the system. |
| Number of UserDefTypes | Number of user defined types defined on the system. |
| SCADA Tags (Actual) | Number of SCADA Tags in the system. |
| SCADA Tags (Licensed) | Total Licensed SCADA tags. |
| SCADA Tags (Used) | Effective number of SCADA Licensed Tags in the system. Can be greater than the number of tags because some tags count as more than one Licensed Tag. |
| SCADA Tags (UsedBy-Arrays) | Effective number of SCADA Licensed Array Tags in the system (Not the raw tag count, the effective licensed count). |
| SCADA Tags (UsedBy-UserDef) | Effective number of SCADA Licensed User Defined Tags in the system (Not the raw tag count, the effective licensed count). |

## Adding a Performance Tag

1. In the **Tag Maintenance** page, select the **Tags** link on the toolbar.
   The Tag Maintenance page appears.
2. Select the **Add Tag Manually** link on the toolbar.
   The Add Tag window appears.
3. Enter a name for the Performance Tag.
4. Select **OK**.
   The Tag Maintenance page displays with the specified tag properties.

## Viewing Tag or Counter Trend Data

How to display a trend of data for a selected tag or performance counter

1. On the **Tag Maintenance** page, select a tag.
2. Right-select the tag and select **Trend**. The trend for the selected tag displays.

## Evaluate Data Compression Performance

You can determine how effectively data compression is functioning at any given time by examining the Collector Detail Diagnostics on the Collectors section of the Dashboard, as shown in the Understanding the Interface topic.

The compression field at the top of the page shows the current effect of archive compression. If the value is zero, it indicates that compression is either ineffective or turned off. If it shows a value other than zero, it indicates that archive compression is operating and effective. The value itself indicates how well it is functioning. To increase the effect of data compression, increase the value of the archive compression deadband so that compression becomes more active. Values for this parameter should typically range from 0 to 9%.

## Handling Value Step Changes with Collector Data Compression

**Note:**
Individual tags can be configured to retrieve step value changes. Refer to the **General** section for more information.

If you enable collector compression, the collector does not send any new input values to the archiver if the value remains within its compression deadband. Occasionally, after several sample intervals inside the deadband, an input makes a rapid step change in value during a single sample interval. Since there have been no new data points recorded for several intervals, an additional sample is stored one interval before the step change with the last reported value to prevent this step change from being viewed as a slow ramp in value. This value marks the end of the steady-state, non-changing value period, and provides a data point from which to begin the step change in value. The collector uses an algorithm that views the size of the step change and the number of intervals since the last reported value to determine if a marker value is needed. The following is an example of the algorithm:

```
BigDiff=abs(HI_EGU-LO_EGU)*(CompressionDeadbandPercent/(100.0*2.0))*4.0

If ( Collector Compression is Enabled )

If ( Elapsed time since LastReportedValue>=( SampleInterval * 5 ) )

If ( abs(CurrentValue-LastReportedValue) > BigDiff )

Write LastReportedValue,Timestamp=(CurrentTime-SampleInterval)
```

In the example above, if a new value was not reported for at least the last 4 sample intervals, and the new input value is at least 4 deltas away from the old value (where a single delta is equal to half of the compression deadband), then a marker value is written.

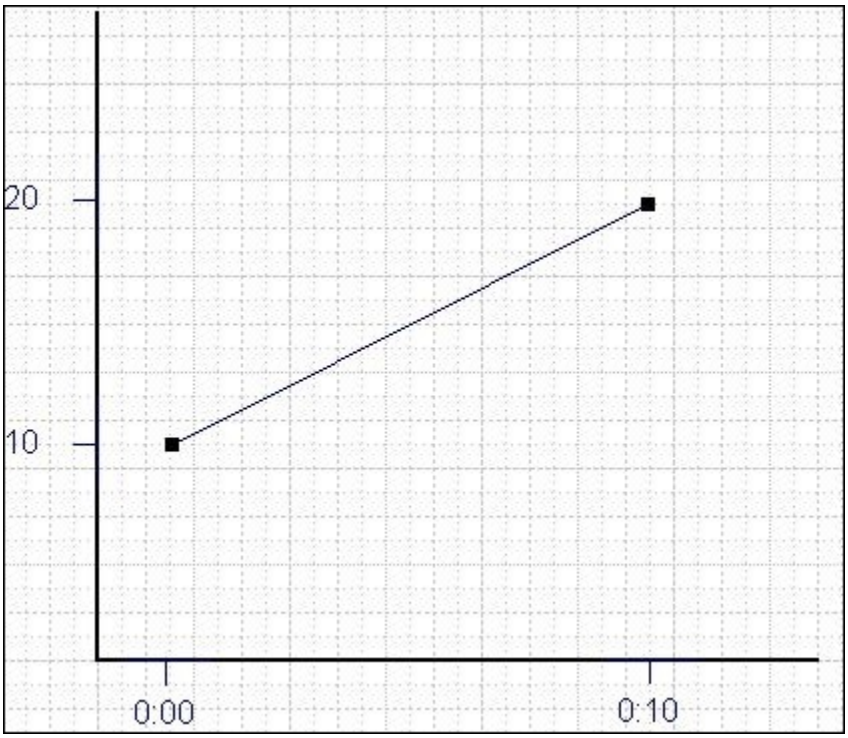
**Note:**
These settings are also adjustable from the Registry. Please contact technical support for more information.

### Example: Value Spike with Collector Compression

A collector reads a value of X once per second, with a compression deadband of 1.0. If the value of X is 10.0 for a number of seconds starting at 0:00:00 and jumps to 20.0 at 0:00:10, the data samples read would be:

| Time | X |
|---|---|
| 0:00:00 | 10.0 (steady state value) |
| 0:00:10 | 20.0 (new value after step change) |

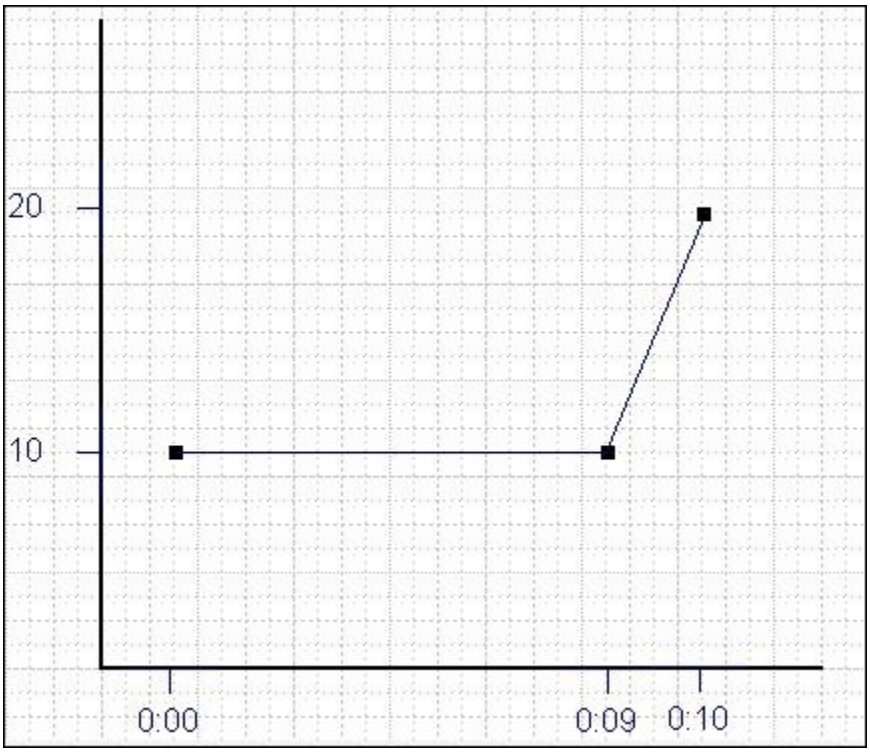
However, without the marker value, if this data were to be put into a chart, it would look like the data value **ramped** over 10 seconds from a value of 10.0 to 20.0, as shown in the following chart.




The addition of the marker value to the data being stored results in the following data values:

| Time | X |
|---|---|
| 0:00:00 | 10.0 (steady state value) |
| 0:00:09 | 10.0 (inserted Marker value) |
| 0:00:10 | 20.0 (new value after step change) |

If you chart this data, the resulting trend accurately reflects the raw data and likely real world values during the time period as shown in the following chart.




## Historian Data Types

Historian uses the following data types.

**Table 435. Data Types**

| Data Type | Size | Description |
|---|---|---|
| Single Float | 4 bytes | The single float data type stores decimal values up to 6 places. Valid ranges for the single float data type are 1.175494351e-38F to 3.402823466e+38F |

**Table 435. Data Types (continued)**

| Data Type | Size | Description |
|---|---|---|
| Double Float | 8 bytes | The double float data type stores decimal values up to 15 places. Valid values for the double float data type are 2.2250738585072014e-308 to 1.7976931348623158e+308 |
| Single Integer | 2 bytes | The single integer data type stores whole numbers, without decimal places. Valid values for the single integer data type are -32767 to +32767. |
| Double Integer | 4 bytes | The double integer data type stores whole numbers, without decimal places. Valid values for the double integer data type are - 2147483648 to +2147483648. |
| Quad Integer | 8 bytes | The quad integer data type stores whole numbers without decimal places. Valid values for the quad integer data type are -9,223,372,036,854,775,808 (negative 9 quintillion) to +9,223,372,036,854,775,807 (positive 9 quintillion). |
| Unsigned Single Integer | 2 bytes | The unsigned single integer data type stores whole numbers without decimal places. Valid values for the unsigned single integer data type are 0 to 65535. |
| Unsigned Double Integer | 4 bytes | The unsigned double integer data type stores whole numbers without decimal places. Valid values for the unsigned double integer data type are 0 to 4,294,967, 295 (4.2 billion). |
| Unsigned Quad Integer | 8 bytes | The unsigned quad integer data type stores whole numbers without decimal places. Valid values for the unsigned quad integer data type are 0 to 18,446,744,073,709,551,615 (19 quintillion). |
| Byte | 1 byte | The Byte data type stores integer values. Valid values for the byte data type are -128 to +127. |
| Boolean | 1 byte | The Boolean data type stores boolean values. Valid values for the boolean data type are 0=FALSE and 1=TRUE. If the user sends zero, the value is taken as zero. **Anything** other than zero is treated as one. |
| Fixed String | Configured by user | The fixed string data type stores string data of a fixed size. Valid values are between 0 and 255 bytes. |

**Table 435. Data Types (continued)**

| Data Type | Size | Description |
|---|---|---|
| Variable String | No fixed size | The variable string data type stores string values of undetermined size. This data type is useful if you cannot rely on a constant string length from your data source. |
| Binary Object | No fixed size | The binary object data type stores binary data. This is useful for capturing data that can not be classified by any other data type. |
| Scaled | 2 bytes | The scaled data type lets you store a 4-byte float as a 2-byte integer in the Historian archive. The scaled data type saves disk space but sacrifices data precision as a result. |

## Additional Notes on Data Types

**Quad Integer**

If a tag is associated with Quad Integer, Unsigned Double Integer, or Unsigned Quad Integer data types and if you are retrieving data using Non-Web Admin, Excel Add-in, Calculation, ihSQL, and ihSDK, then there may be a loss of precision value due to a Visual Basic limitation.

Calculation collector supports only the calculations performed using the current value calculation. It does not support other calculations due to a Visual Basic script limitation.

The high and low EGU limits for Quad Integer, Unsigned Single Integer, Unsigned Double Integer, and Unsigned Quad Integer are between 2.2250738585072014e-308 to 1.7976931348623158e+308.

**Fixed String Data Types**

The fixed string data type lets you store string data of a fixed size. This is useful when you know exactly what data will be received by Historian. If a value is larger than the size specified in the Data Length field, it will be truncated.

**Scaled Data Types**

Historian uses the high and low EGU values to both store and retrieve archived values for the scaled data type. This allows you to store 4 byte floats as 2 byte integers in the Historian archive. Though this saves disk space, it also sacrifices data precision as a result. The smaller the span is between the high and low EGU limits, the more precise the retrieved value will be.

When calculating the value of a scaled data type, use this formula:

```
ArchivedValue = (((RealWorldValue - EngUnits->Low) / (EngUnits->High - EngUnits->Low) * (float) HR_SCALED_MAX_VALUE)
 + .5);
```

For example: A value of 12.345 was stored in a scaled tag whose high EGU was 200 and low EGU was 0. When later retrieved from the Historian archive, a value of 12.34473 would be returned.

**Note:**
Values that are outside of the EGU range of a scaled data type tag get stored as "bad" or "scaledoutofrange" in Historian. You **cannot** correct values for scaled data types that were inserted while EGUs were incorrect. Changing either the High or Low EGU tags does not affect existing data, but only affects the new data with new timestamps. If necessary, contact technical support for additional information.

### Setting a Value For the Fixed String Data Type

1. In the Admin App, select **Tags**.
2. Select the tag you want to configure.
3. Select **Collection**.
4. In the **Data Type** drop-down list, select **Fixed String**.
5. Enter a value in bytes in the adjacent field. This field is enabled only when the data type selected is Fixed String.

# Managing Tags

## Access a Tag

1. Access the Web Admin console *(on page 2818)*.
2. Select **Configuration** or **Show All Collectors** or **Show All Clients**.
3. Select **Tags**.
   The **Tags** page appears.
4.

## Add a Tag to a Data Source

To display the Tags page, select the Tags link in any of the Historian Web Admin console pages. The Tags page lets you read and modify all tag parameters for the Historian system. To access information on a specific tag or group of tags, however, you must first search for the tags. You can search for the tags using the Search for Tags button.

You can add tags manually through the tags page or choose the tags from the listed collectors. Typically, you add tags to Historian by browsing the data source. You can also add tags manually or add tags from the collector by selecting the appropriate link in the second line of the display.

If you add a tag with a tag name greater than 25 characters in length, the characters beyond 25 are not visible in the Tags list on the Tags page. To see the entire tag name, place the mouse cursor over the tag to see a ToolTip that displays the complete tag name.

**Note:**
Do not add or update the following spare configurations as the data may get corrupted or over written:

- The **Spare 1** field for OSI PI Distributor. OSI PI distributor reads data from the Historian tag displayed in the Tag Source Address field and sends it to the OSI PI tag name displayed in the **Spare 1** field.
- The **Spare 5** field for Server to Server Collector and Server to Server Distributor as it is only used for internal purposes.

## Add a Tag Manually

**Note:**
If you manually add a Server-to-Server tag that uses the polled collection type, make sure that you set the **Time Adjustment** field for the tag to the **Adjust for Source Time Difference** option after you add the tag. The **Time Adjustment** field is located in the **Advanced** section in the **Tag Maintenance** page.

1. Select the icon link in the **Tag Details** page and select **Add Tags Manually**.
   The Add Tag window appears.
2. Select a collector from the drop-down list in the **Collector Name** field. This associates the new tag with a specific collector.
3. Enter the **Source Address** and **Tag Name** in the appropriate fields.
4. Select the data store in the **Data Store** field.
5. Select a **Data Type** from the drop-down list.
6. For fixed string data types only, enter a value in the field adjacent to the Data Type field.
7. Select Seconds, Milliseconds, or Microseconds in the **Time Resolution** field.

8. If the tag is an Array Tag, select the **Is Array Tag** option.
9. Select **Add** to add the tag.

## Add a Source Address to a Tag

1. Select a collector from the drop-down list in the **Collector Name** field.
   This associates the new tag with a specific collector.
2. Enter the **Source Address** or select **Browse**.
   The Add Tags from Collectors window appears.
3. Select the tag you want to associate with the source address.
   You can select only one tag.
4. Select **OK**.
   The source address of the tag is added.

## Adding OPC Tags from a Collector

1. Select the link in the **Tag Details** page.
2. Select **Add Tags from Collector**.
   The Add Tags from Collector window appears.
3. Select the collector from the **Collector** name list.
4. Enter the **Source Tag Name** or select **Browse**.
   The list of folders available with the collector is displayed.
5. Expand the folder to select the desired tags.
   The > symbol indicates that you need to navigate further within the folder. You can select a single tag or multiple tags. If you want a series of tags, press and hold the Shift key and select the series.
6. Select **Add** or **Add All** to add tags.
   The selected tags appear in the right hand section.
7. To preview the selected tag details, select **Preview**.
8. To add the selected tags from the collectors, select **Add Selected Tags**.

## Adding Simulation Tags from a Collector

1. Select the link in the Tag page and then select **Add Tags** from Collector.
   The Add Tags from Collector window appears.
2. Select the collector from the **Collector** name list.
3. Enter the **Source Tag Name** or select **Browse**.
   The list of available tags is displayed.
4. Select the desired tags and select **Add** or select **Add All** to add tags.

You can select a single tag or multiple tags. If you want a series of tags, press and hold the Shift key and select the series.

The selected tags appear in the right hand section.

5. To preview the selected tag details, select **Preview**.
6. To add the selected tags from the collectors, select the **Add Selected Tags**.

## Filter and Search Tags

Using the **Search** window you can:

- Add multiple search criteria based on the tag criteria and the criteria value.
- Modify the existing criteria value.
- Delete unwanted search criteria from the list.
- Automatically load the most recently used criteria for re-use.
- Search the Historian database based on search criteria.
- View the details of a tag.

## Filtering Tags

By default, the Tags page displays all available tags. To filter the tags based on a set of criteria, use the Filter option.

1. On the **Tags** page, select the **Filter** button.
   The section expands to show the filter criteria.

2. Select the criteria and enter a value to filter by.
3. Select **Add Criteria** .
4. Select **Apply** to filter the tags based on the criteria.

Select **Remove** to remove the criteria selected or select **Reset** to reset your input.

## Searching for Tags: Simple Search

1. Select the **Search** button in the Tags page.
   The Search box appears.

2. Enter a tag mask in the **Search** field using either the full/partial tagname or standard Windows wildcard characters.

   This will help you to filter the search query more precisely. If wildcard characters are not used, then the search will result in all the tags containing the given search string.

   > **Note:**
   > Supported wildcard characters in simple search are * and ?.

   Example for using wildcard characters in search strings:

| Search String | Result |
|---|---|
| W* | All tags starting with letter W |
| *e | All tags ending with letter e |
| W*e | All tags starting and ending with W and e respectively |
| Tag? | All four letter named tags, where the last letter can be anything |
| W*0000? | All tag names staring with W and ending with 0000 followed by any single letter |

3. Select **Enter**.
   The relevant tag(s) are listed.

## Searching for Tags: Advanced Search

The Advanced Tag Search window allows you to search for a set of tags that match the search criteria and then perform actions on one or more tags that you select from the list.

It saves the most recently used search criteria to a file named **DefaultSearchCriteria.xml** in the Excel App Data path, which is: `c:\users\<username>\AppData`, and this criteria is automatically loaded into the window the next time it is opened. This allows you to re-use or modify the criteria rather than entering them each time. To reset your criteria, delete the XML file.

1. Select the **Advanced Search** button.
   The Advanced Search window appears.
2. In the **Step 1** section, select the tag criteria from the list.
3. Enter or select the **Tag Criteria Value**.
   If you leave the field blank, the search returns all of the available tags.

4. (optional) Select the **Add Criteria** button to add more criteria to narrow your search.
5. Select the **Find Tags** button.
   All the tags that satisfy the query criteria are displayed in the Step 2 section.
6. Select tags from the list by selecting **Add** .

   To select all of the tags, select the **Add All** button. To remove a selected tag, select Remove .
7. Select **Apply** to return the list of tags on the parent window.
   Select **Reset** to clear your search criteria. Select **Cancel** to close the window.

## Access the Trend Chart of Tag Values

This topic describes how to access the values of a tag in a trend chart. The difference in the timestamp of consecutive values depends on the time resolution of the tag. For example, if the time resolution is seconds, the timestamp of consecutive values of the tag will be one second apart.

You can plot the values of multiple tags in a single trend chart.

1. Access the Web Admin console *(on page 2818)*.
2. Select one or more tags on the **Tag Details** page.
3. Select the button at the bottom of the page and select the **Trend** option.
   The Trend page displays the trend of the last 10 minutes with an interval of 1 minute and interpolated sampling mode. By default, all selected tag trends are shown.

- To zoom in on the trend, select and drag to select the region to zoom in on.
- To zoom out on the trend, select the Reset Zoom button.
- To see the tag name, date/time, quality, and value at an instance, hover your mouse over the trend.
- You can change the type of trend you want to view: Line, Column, or Area.
- Selected tag names are shown as legends; select to see a particular tag's trend and select again to hide the trend.
- Select the Refresh button to refresh the page.
- Select the Close button to cancel the operation.

## Displaying Raw Data Samples

You can view the most recent ten raw samples for one or more selected tags.

1. Select one or more tags on the **Tag Details** page.

   **Note:**
   If you choose **Select All**, only the data for the first 10 tags selected will be displayed.

2. Select the button at the bottom of the page and select the **Last 10 Raw Values** option.
   The Last 10 Raw Values for selected Tag(s) window appears, displaying the tag name, timestamp, value, and quality for each selected tag.

- For each selected tag, values are sorted by timestamp in descending order.
- Where necessary, tag names are truncated and indicated with an ellipsis "..."
- To view the full name, hover over the tag name. A tooltip displays the full tag name.
- Panes can be collapsed and expanded to see the last 10 raw values for more selected tags.
- Select **Refresh** button to refresh the page.
- Select **Close** button to cancel the operation.

## Dynamic Collector Updates

The dynamic collector update feature ensures that any modifications done to the tag configuration do not affect all the tags in a collector. Only the tags that stop data collection will record zero data and bad quality without restarting the collector. In other words, the tags that do not stop data collection do not record bad data samples to the collection.

Whenever you add tags, delete tags, or modify certain tag properties, the following collectors reload only the modified tag(s) without restarting the collectors.

- OPC Collector
- iFIX collector
- Calculation collector
- Simulation Collector
- Server to Server Collector
- PI Collector
- PI Distributor

If you disable On-line Tag Configuration Changes option, any changes you make to the tags do not affect collection until after you restart the collector and the data archiver. You can restart the data archiver from Windows services. To restart the collector, stop and start the collector service or executable (or use Configuration Hub *(on page 639)*). Restarting the collector stops and restarts the tag(s) collection and records bad data samples to the collection. If the modified tags get zero bad markers and available

runtime values at the same time, then precedence is given to available runtime values instead of zero bad markers.

All the collector configuration changes done within a 30 second time frame are batched together. When possible, update/modify a small set of tags at a time to collect the modified data faster. However, when updating large sets of tags at the same time, best practice is to disable On-line Tag Configuration Changes and restart the collector after you are finished.

## Starting or Stopping Data Collection For a Tag

For a tag to stop and restart collection without restarting the collector, the On-line Tag Configuration Changes option must be enabled. By default, the On-line Tag Configuration Changes option is enabled. If necessary, enable the On-line Tag Configuration Changes option on the **Advanced** section of the **Collector Maintenance** page.

1. In the **Tag Viewer** section of the **Tags** page, select the tag from the list.
2. In the **Tag Editor** section, select **Collection**.
3. Scroll down to the **Collection** field and either:
    - **Disable** the collection option.
    - **Enable** the collection option.
4. Select **Update**.

## Reload Tag Parameters

Whenever you modify certain tag parameters, the following collectors reload only the modified tags without restarting the collectors.

- OPC Collector
- iFIX collector
- Calculation collector
- Simulation Collector
- Server to Server Collector
- PI Collector
- PI Distributor

For a tag to stop and restart the collection without restarting the collector, you must select the **On-line Tag Configuration Changes** option on the **Advanced** section of the **Collector Maintenance** page. By default, the On-line Tag Configuration Changes option is enabled.

If you disable the On-line Tag Configuration Changes option, any changes you make to the tags do not affect collection until after you restart the collector. To restart the collector you must stop and start the collector service or executable. Restarting the collector stops and restarts the tag(s) collection and records bad data samples to the collection. All the collector configuration changes done within a 30 second time frame are batched together and applied to the collector. If the modified tags get zero bad markers and available runtime values at the same time, then precedence is given to available runtime values instead of zero bad markers.

It is recommended that you update/modify a small set of tags at a time to collect the modified data faster. It is recommended to you disable the On-line Tag Configuration Changes option while updating large sets of tags at the same time, and restart the collector after modification.

### Tag Properties that Cause the Tag Collection to Stop and Restart

Changes to the following tag properties cause an updated/modified tag to stop and restart the collection, when the On-Line Tag Configuration Changes option is enabled:

- Collector Name
- Collector Type
- SourceAddress
- Spare 1 – 5
- Data Type
- Collection Interval
- Collection Offset
- Collection Disabled/Enabled (CollectionDisabled in SDK)
- Collection Type
- TimeStampType
- Calculation Dependencies (in SDK) or Calculation Triggers (in Historian Administrator) – applies to Server-to-Server and Calculation collectors only

### Tag Properties that Do Not Cause the Tag Collection to Stop and Restart

The following tag properties do not cause the tag collection to stop and restart, but the collectors use these new values immediately (when the On-Line Tag Configuration Changes option is enabled):

- High Engineering Units
- Low Engineering Units
- Input Scaling
- High Scale
- Low Scale

- Collector Compression
- Collector Deadband Percent Range
- Collector Compression Timeout

## Rename Tags

> **Note:**
> To rename tags, you must be a member of the administrator's group with tag level security.

New tag names are called **active** tag names and old tag names are called **aliases**.

You can also rename or permanently rename tags using the Non-Web Historian Administrator, the Excel Add-in, or the Application Program Interfaces (ihSDK, ihUAPI, or ihAPI).

If you modify the properties of a renamed tag, the properties of all of its aliases will also be updated.

Whenever you rename a tag, only the active (newest) tag name will be visible in the Tags list on Historian Administrator Tag page. You can rename tags multiple times, but only the latest, active tag name will be visible in the Tags list.

You can also retrieve the data using any alias (that is, the new or old tag names). However, the Tag page will display only an active tag name (that is, the new tag name). Whenever you change or copy a tag name, the information about the old tag name, new tag name, and time stamps are all recorded in the audit trail.

Be aware of the following when you are using the Tag Rename feature:

- If you modify a renamed tag property, then all of the alias' tag properties will also be updated.
- If you delete a renamed tag, then all the aliases will also be deleted.
- You can rename tags multiple times, but only the latest active tag name (renamed tag name) will be visible in the Tags list.
- If you rename a tag, the tag count does not increase.
- If you copy a tag, then the tag count increases.
- An alias can be queried, but cannot be modified or deleted.

### Permanently Rename a Tag

You can permanently rename a tag if you no longer want to read from and write to a tag by its previous name. Permanently renaming makes the previous tag name available for new usage. For example, if you had permanently renamed Tag A to Tag B, you can create a new tag with the tag name Tag A with no linkage to the previous tag.

Things you need to know when you are using the Permanent Rename feature:

- If you permanently rename a tag, the tag name will be updated with the new tag name and the old tag name will be lost.
- You can permanently rename tags multiple times, but only the latest tag name (new tag name) will be visible in the Tags list.
- If you permanently rename a tag, the tag count does not increase.
- Store and forward data will be lost if you do a permanent rename and the data is sent using the old tag name.
- There will be loss of data during the process of permanently renaming a tag. If you are going to perform permanent rename, it is recommended to stop the collector and then permanently rename the tag.
- If a tag is permanently renamed and is a trigger tag to other tags, then you need to re-assign the new trigger name to the tags.

## Copying a Tag

If you copy a tag, then the tag count increases.

1. Select the **Copy** button in the **Tags** page.
   The **Copy Tag** window appears.
2. Enter a tag name for the new tag.
3. Select **OK**.

## Renaming a Tag

Renaming a tag creates an alias for the tag. Only the name changes; none of the tag's properties are changed. The tag data can be referred to using any alias for that tag, including the current name. Renaming a tag does not increase the tag count.

1. Double-click the **Tag** link in the **Tag Details** page.
   The tag name becomes editable.
2. Enter a new tag name and press the Enter key.

If you are connecting to Historian 3.5 collectors, then you must restart the respective collectors before browsing the tags.

## Renaming a Tag Permanently

Tag renaming only changes the tag's name without changing any of the tag's properties.

1. In the **Tag** page, select the tag you want to rename.
2. Select the arrow button in the right-hand section and select **Rename**.
   The Rename Tag window appears.
3. Enter a new tag name.
4. Select **OK**.
   A message box appears confirming that you want to permanently rename the tag.
5. Select **Yes** to permanently rename the tag.

If you are connecting to Historian 3.5 collectors, then you must restart the respective collectors before browsing the tags.

## Stale Tag Management

Stale tags are tags that have no new data samples within a specified period of time. These tags add to system overhead and slow down user queries. The **Data Store Configuration** option allows system administrators to configure the time period after which tags are considered stale and how often the system should check for stale tags.

Under the default configuration, tags are never considered stale. This effectively disables stale tag management.

To see the names and descriptions of all currently stale tags, use the **IsStale** criteria in a tag search.

To improve performance, permanently delete *(on page 2882)* unused tags. Perform a tag search using the **IsStale** criteria and select **Permanently Delete Tag**.

## Delete Tags

There are key differences between deleting and permanently deleting a tag.

| Delete | Permanently Delete |
|---|---|
| The tag is removed from the tag database but any data for that tag is retained in the archive | All data associated with the tag is completely lost and the tag name is available for reuse. |
| Tag data is still available from the archive, so you can still reference that tag, for example, from within a calculation formula or by using the Excel Add-In. | Tag data is no longer available from the archives and you will **not** be able to query the existing data for that tag. |

## Deleting a Tag

1. In the **Tag** page, select a tag from the list.
2. Select the button at the bottom right of the page. The Delete Tag window appears.
3. Select the **Remove Tag(s) from System** option and select **OK**. This removes the tag from the Tag Database but retains any data for that tag in the archive.
4. A message box appears asking you to confirm the deletion. Select **Yes** to delete the tag.

## Deleting a Tag Permanently

1. In the **Tag** page, select a tag from the list.
2. Select the **Delete** button. The Delete Tag window appears.
3. Select the **Permanently Remove Tag(s) from System** option and select **OK**. This permanently removes the tag from the Tag Database.
4. A message box appears asking you to confirm the deletion. Select **Yes** to delete the tag.

# Managing Data Stores

## About Data Stores

A data store is a logical collection of tags. It is used to store, organize, and manage tags according to the data source and storage requirements. A data store can have multiple archive files (*.IHA), and includes both logical and physical storage definitions.

Tags can be segregated into separate archives through the use of data stores. The primary use of data stores is to segregate tags by data collection intervals. For example, you can put a name plate or static tags where the value rarely changes into one data store, and your process tags into another data store. This can improve query performance.

Historian data stores are stored as archive files that contain data gathered from all data sources during a specific period of time. You can write and read data from the archive files.

You can define two types of data stores:

- **Historical Data Store**: Tags stored under historical data store will store data as long as the disk space is available. Depending on your license, you may be able to create multiple historical data stores. The maximum number of historical data stores supported depends on the license.
- **SCADA Buffer Data Store**: Tags stored under the SCADA buffer data store will store data for a specific duration of time based on license.

When you install the Historian server, two historical data stores are installed by default.

- **System**: Stores Historian messages and performance tags. This is only for internal usage within Historian, and you cannot add tags to this data store. You cannot rename or delete the system data store.
- **User**: Stores tag data. This is a default data store. You can rename and delete a user data store as long as there is another default data store set for tag addition.

Based on your license, a SCADA Buffer data store may also be installed. It stores short-term tags and data.

## Moving Tags Between Data Stores

You can move tags from one data store to another. After a tag is moved, the incoming data is stored in the new data store.

When you move a tag, only the tag itself is moved; the data already associated with that tag does not automatically move with it. Moving the old data is optional, but if the old data is not moved to the new data store, it cannot be retrieved.

1. In the **Tags** page, select a single tag or multiple tags.
2. In the **Tag Editor** section, select **Advanced**.
3. Select **Data Store** to select the tag's new data store.
   A message appears asking you to confirm the change.
4. Select **Yes**.
5. Select **Update**.
   A message appears confirming that the tag's data store has changed and reminding you that the data has to be manually moved using MigrateIHA.exe.

(optional) Move the data associated with the tags you just moved with the migration utility tool (MigrateIHA.exe). If you moved multiple tags, you can move the data for all, some, or none of the tags.

Before moving any data, best practice is to back up the archive file(s) that contain the old data.

**Note:**
When migrating tags using MigrateIHA.exe, ensure that you select the **Migrate using the tag mask** option and specify the tag name or wildcard mask to migrate the tag you want to include. Refer to the How to use the IHA Migration Tool section in *Important Product Information*.

## Adding a Data Store

The number of data stores you can create depends on your license.

1. Access the Web Admin console *(on page 2818)*.
2. Select **Data Stores**.
   A list of data stores appears.
3. Enter values as described in the following table.

| Field | Description |
|---|---|
| **Data Store Name** | Enter a unique name for the data store. A value is required. You can use all alphanumeric characters and special characters except / \ * ? < > \| |
| **Description** | Enter a description for the data store. |
| **Is Default** | Switch the toggle on if you want to set this data store as the default one. A default data store is the one that is considered if you do not specify a data store while adding a tag. You can set only one data store as default. |

4. Select .
   The data store is created.

   When you add tags to the data store, it will have its own set of .IHA (iHistorian Archive) files. Ensure that you back up the new data store archives periodically.

When you add the tags to the new data store, it will have its own set of .IHA (iHistorian Archive) files. Ensure that you back up *(on page 2896)* the new data store archives periodically.

## Deleting a Data Store

You can delete a data store if it is no longer needed. You cannot delete the System data store. You cannot delete the last User store; at least one User store must exist.

1. If there are tags assigned to the data store, reassign them to other data stores or delete them.
2. In the **Data Stores** page, select the unwanted data store and then select **Delete**.
   A message appears asking you to confirm deletion
3. Select **OK** to delete the data store.

## Editing a Data Store

The following special characters cannot be used in data store file names: `/ \ * ? < > |`

1. Select the **Edit** button.
   The **Edit Data Store Configuration** window appears.
2. Make the desired change(s):
   a. To rename the data store, enter the new name in the **Data Store New Name** field.
   b. Enter or modify the **Description**.
   c. Toggle the **Is Default** option.
3. Select **Save** to save the changes.

# Managing Data Archives

## Configure Data Archives

### About Data Archives

Historian archives are stored data files, each of which contains data gathered from all data sources during a specific period of time. There are two types of archive files in an Historian archives directory:

`Machinename_Config.ihc` – the single .IHC file contains information about the archiver, tag configuration, and collector configuration.
`Machinename_ArchiveXXX.iha` – archive data files where x is a number indicating the place of the file in a time-based sequence.

Since archived data files can be quite large, adjust system parameters carefully so that you limit data collection to meaningful data only and so that you minimize the required size of system storage. This chapter describes techniques you can use in your application to accomplish these goals.

**Note:**
You must have a minimum of 10 GB free space available for the Data Archiver to start.

Historian now supports a maximum archive size of 256 GB per archive. When you start Historian, it may take a longer time to start an archiver depending on the number of archives online, number of tags, and number of connections.

**Note:**
The limit for the number of LUNs is 100.

### Archive Creation

Archive files are created to store data as the Historian server receives it until they reach their duration limit. When the limit is reached, a new archive is created and the data is loaded into that archive. These archives can be created based on number of days or hours. You can observe the archives on the left side of the **Data Store** page, under **Archives**, with the name of the archive and start time for each archive

**Note:**
As of Historian 7.0, Historian archives are time-based only. Historian will asynchronously create a new empty archive when data starts loading into an existing archive. If the option to automatically create archives is not enabled, however, you must open a new archive manually.

To create archives based on **days**:

1. Open the **Data Stores** page and then select **Edit**. The Archive Configuration page appears.
2. Select **Configuration**.
3. In the **Archive Duration** field, select the **Days** option from the drop-down list.
4. Enter the number of days for which you want to create archives.
5. Select **Update**.

Setting Days to 1 means that a new archive will be created every day starting from the time your first archive is created. The next archive is created after one day (24 hours) from the time the first archive was created.

To create archives based on **hours**:

1. Open the **Data Stores** page, and then select **Edit**. The **Archive Configuration** page appears.
2. Select **Configuration**.
3. In the **Archive Duration** field, select the **Hours** option from the drop-down list.
4. Enter the number of hours for which you want to create archives.
5. Select **Update**.

Setting **Hours** to 1 means that a new archive will be created after every hour starting from the time your first archive has been created.

## The Archive Configuration Screen

To access the **Archive Configuration** page, select **Edit** on the **Data Stores** page.

The Archive Configuration page lets you read and modify the parameters of archives and data stores. In this page, you can see the list of all archives of the selected data store.

To examine a particular archive, select the archive and then select **Edit**. The details of the archive are displayed in the Archive Details section.

- Action Buttons
- Statistics Section
- Archives Section
- Archive Details Section

## Points To Remember

- You may need to add an archive when the current archive is almost full and you have not enabled automatic creation of archives.
- You may need to restore an archive when you start up after an unplanned shutdown or when you need to retrieve data from an old, inactive archive.
- You may need to back up an archive before a planned Historian software product upgrade.
- You may need to manually resynchronize archives when the archives in the mirrored environment are not synchronized.

## Action Buttons

Select a button to perform the action indicated by the name. The following table describes these buttons.

| Button | Action |
|---|---|
| Add Archive | Select this button to add a new archive to the data store. |
| Remove Archive | Remove an archive. First select an archive name to select it, and then select Remove. Selecting OK removes the archive file from the list of archives for the system, and places it in the \Archives\Offline directory. This does not delete the archive file from the system. An archive must be closed before it can be removed. |
| Close Archive | Manually close the current archive. An archive must be closed before it can be removed. |
| Backup Archive | Back up a selected archive. Verify the file name and path and then select OK to save the file. |

| Button | Action |
|---|---|
| Restore Archive | Restores an archive from backup. |
| Update | Apply all parameter changes that you have made in this page. If you want to cancel changes and return to the original values or settings, open a different page and then return to the Archive Configuration page. |
| Edit | Select this to edit the Archive configuration details. |

### Statistics Section

Indicates the current status of the collector.

- **Running** indicates that the collector is operating.
- **Stopped** indicates that it is in pause mode and not collecting data.
- **Unknown** indicates that status information about the collector is unavailable at present, perhaps as a result of a lost connection between collector and server.

**Table 436. Configuration Tab**

| Field | Description |
|---|---|
| Default Archive Path | The path name that will be used for any newly created archives. If you change the path, the change takes effect the next time a new archive is created.<br><br>**Note:**<br>Do not use a period in the default archive path field. If a period is present in the default archive path, you will not be able to specify a default archive name. |
| Default Backup Path | The location to which the backup file will be saved. |
| Archive Duration (Days/Hours) | Specifies the duration of a newly created archive in days or hours. A new archive will be created after the selected number of days or hours. |

**Table 436. Configuration Tab (continued)**

| Field | Description |
|---|---|
| | **Note:**<br>When the **Archive Duration** property is changed in a mirrored environment, the changes will take effect only after a time gap of 15 minutes. |
| Data is Read-only After (Hours) | The number of hours, prior to now, for which data can be stored in a read/write archive. After the time expires, that portion of the archive file is automatically made read-only. Incoming data values with timestamps prior to this time are rejected.<br><br>A single archive file, therefore, may contain a read-only section, another read-write section containing recently written data, and unused free space.<br><br>**Note:**<br>A read-only archive file cannot be moved using Windows Explorer. To move a read-only archive file, select the file and select the Remove button on the **Details** section of the **Archive Maintenance** page. The Archiver then releases its locks, which permits you to move the file at will. |
| Base Archive Name | A prefix that is automatically added to the file name of all created archives. To change the prefix, enter a new text string and select **Update**. |
| Free Space Required (MB) | Indicates the remaining disk space required after a new archive is created. If the available space is less than the requirement, a new archive is not created. The default is 5000 MB.<br><br>The Free Space Required field does not apply to alarms and events archives. The alarms and events archiver will continue writing to the alarms and events archive until the drive is full. If this occurs, the alarms and events archiver will buffer incoming alarms and events data until the drive has free space. An error will also be written to the Historian message log. |

**Table 436. Configuration Tab (continued)**

| Field | Description |
|---|---|
| Automatically Create Archives (Enable/Disable) | Select the appropriate button to enable or disable this function. When enabled, the server automatically starts a new archive in the default path directory whenever the current archive fills up. If disabled, no new data will be written to the archives once the default size has been reached.<br><br>**Note:**<br>To create multiple archives at the same time, **Automatically Create Archives** must be *Disabled*. |
| Overwrite Old Archives (Enable/Disable) | Select the appropriate button to enable or disable this function. When enabled, the system replaces the oldest archived data with new data when the default size has been reached.<br><br>• To create multiple archives at the same time, **Overwrite Old Archives** must be *Disabled*.<br>• If you enable both **Automatically Create Archives** and **Overwrite Old Archives**, then you must set `ihArchiveFreeSpaceHardLimit` to TRUE using APIs.<br><br>**CAUTION:**<br>Since this action deletes historical data, exercise caution in using this feature. Be sure that you have a backup of the archive so that you can restore it later. |
| SCADA Buffer Duration (Days) | Indicates the maximum number of days the trend data can be stored. The maximum number of days is 200 days.<br><br>This field applies only to SCADA buffer data stores. |
| Use Caching (Enabled/Disabled) | When reading data from the archiver, some data is saved in the system memory and retrieved using caching. This results in faster retrieval as the data is already stored in the buffer. Enable the **Use Caching** option to retrieve data faster.<br><br>This option is not available for SCADA buffer data stores. |

**Table 436. Configuration Tab (continued)**

| Field | Description |
|---|---|
| Generate Message on Data Update (Enabled/Disabled) | If this option is enabled, an audit log entry will be made any time the value of a previously archived data point in the Historian archive is overwritten. This log entry will contain both the original and new values.<br><br>To create multiple archives at the same time, **Generate Message on Data Update** must be *Disabled*.<br><br>This option is not available for SCADA buffer data stores. |
| Store OPC Quality (Enabled/Disabled) | Stores the OPC data quality.<br><br>To create multiple archives at the same time, **Store OPC Quality** must be *Disabled*. |
| Stale Period | Specifies the time period after which tags are considered stale for this data store. The value is defined in days. Valid values are:<br><br>• 0 (zero): This default value means that tags are never considered stale. This effectively disables stale tag management.<br>• 7 days (1 week) to 36500 days (100 years) |
| Stale Period Check | Specifies the frequency with which the staleness of the tag is checked. The value is defined in days. Valid values are 1 day (the default) to 30 days. |

## Archives Section

The Archives section displays the list of archives available with the selected data store. To edit an archive, select the archive and select the **Edit** button. The details of the archive are displayed in the Archive Details section.

| Field | Description |
|---|---|
| Name | The name of the archive. |
| Start Time | The time of the oldest sample in the archive. |
| End Time | The time the archive is automatically or manually closed. |

### Archive Details Section

Archive Details section of the page lets you read and modify all archiving parameters for the Historian system.

**Table 437. Archiving Parameters**

| Field | Description |
|---|---|
| Status | The current operating state of the archive: Active, Current, Empty.<br>• Current: Archive is actively accepting data.<br>• Active: Archive contains data but is not currently accepting data.<br>• Empty: Archive was created but has never accepted data. |
| Start Time | The time of the oldest sample in the archive. |
| End Time | The time the archive is automatically or manually closed. |
| Last Backup On | The date and time the last backup was performed on this archive. |
| Backup By | User name (at time of login to Historian Administrator) of the person who performed the last backup of the archive. |
| File Location | The path and name of the archive file. |
| File Size (MB) | The size (in MB) of the archive file.<br>**Note:**<br>Historian now supports a maximum Archive Size of 256 GB per archive. |
| File Attribute | The attribute to set a closed archive to Read-only or Read/Write.<br>**Note:**<br>If you plan to create multiple archives at the same time, then you **must** set **File Attribute** to **Read/Write**. |

## Calculate Required Archive Size

Historian will asynchronously create a new empty archive when data starts loading into an existing archive. Whenever the current archive becomes full, Historian will immediately serve data to a newly created archive. This significantly reduces archive creation and transition time. If the option to

automatically create archives is not enabled, however, you must open a new archive manually. As of Historian 7.0, Historian archives are time-based only.

**CAUTION:**
When the default size limit is reached, if automatic archive creation is disabled and you do not manually create a new archive, new data will not be written to the archives .

**CAUTION:**
If the available disk space is less than the configured amount of free disk space, Historian cannot automatically create new archives.

If you enable the Overwrite Old Archives option, the system replaces the oldest archived data with new data. Since this action deletes historical data, exercise caution in using this feature. Be sure that you have a backup of the archive so that you can restore it later.

If you enable the Overwrite Old Archives option and if you want to retrieve time-based information, create an additional archive to overcome the early loss of data due to archive preparedness. For example, if you want to save 12 months of data into 12 archives, create 13 archives.

During archiver startup and every 60 seconds while the server is running, Historian checks to make sure that you have configured enough free disk space to save the archives, buffer files, and log files. If there is insufficient disk space, the Data Archiver shuts down and a message is logged into the log file. For each archive, you need approximately 1MB of archive space for every 1000 tags, for tag information.

By default, you can view the Historian archiver log file in `Historian Data\LogFiles`

```
[03/03/10 15:28:41.398] Insufficient space available in [d:\Historian\Archives\]
[03/03/10 15:28:41.399] The server requires a minimum of [5000 MB] to continue
[03/03/10 15:28:41.679] USER: DataArchiver TOPIC: ServiceControl MSG: DataArchiver(DataArchiver) Archiver s
[03/03/10 15:28:41.807] DataArchiver Service Stopped.
[03/03/10 15:28:41.809] [d:\Historian\LogFiles\DataArchiver-34.log] Closed.
```

Archive size is a function of the rate at which you archive data and the time period that you want the archive to cover. You may want the archive to cover a time period of perhaps, 30 days.

Factors that affect the rate at which you archive data are:

- Number of tags – a large number of tags increases the data rate.
- Polling frequency of each tag – a high polling frequency increases the data rate.

- Compression settings – disabling compression or setting narrow deadband parameters increases the data rate.
- Data types – choosing data types that increase the number of bytes per value increases the data rate.

The following is an example of a manual calculation of required archive size, using typical parameter values.

**Table 438. Assumptions**

| | |
|---|---|
| Number of tags | 5000 |
| Polling rate | 1 value/5 seconds |
| % Pass Compression | 5% (Pass Compression is the number of data values archived relative to the number of values read, expressed as percent.) |
| Bytes/value: | 4 |
| Duration: | 1 month (30 days) |

**Calculation**

$$\#Tags \times \frac{Values}{Tag} \times \frac{Tags}{Second} \times \%PassComp \times \frac{Bytes}{Value} \times \frac{Seconds}{Hour} \times \frac{Hours}{Day} \times \frac{MB}{Bytes} = \frac{MB}{Day}$$

$$5000 \times \frac{1}{1} \times \frac{1}{5} \times \frac{5}{100} \times \frac{4}{1} \times \frac{3600}{1} \times \frac{24}{1} \times \frac{1}{1024 \times 1024} \times 30 = 494 \frac{MB}{Month}$$

The calculation shows that a file size of 500 MB is adequate for archiving one month of data for this application.

If you believe the calculated size is too large for your application, you can modify parameters as follows:

- Decrease the polling frequency.
- Increase compression deadband, reducing the pass percentage.
- Reduce the number of tags.
- Add more disk capacity to your computer.

### Archive Size Calculator

An Archive Size Calculator tool is available to estimate archive size based on your input and estimates the archive size and collector compression based upon a tag that has already been configured. Log on to http://support.ge-ip.com/devsupport/ to download this and other GE Intelligent Platforms freeware product solutions.

## Prepare for Multiple Archive Creation

If you plan to create multiple archives at the same time, set the following parameters. These parameters apply **only** when creating multiple archives at the same time.

- In the **Archive Details** section, set **File Attribute** to **Read/Write**.
- In the **Configuration** Tab:
  - set **Automatically Create Archives** to **Disabled**.
  - set **Overwrite Old archives** to **Enabled**.
  - set **Store OPC Quality** to **Disabled**.
  - set **Data is Readonly After (Hours)** to **1 month**.
  - set **Generate Message on Data Update** to **Disabled**.

Before you begin creating multiple archives on a remote machine, ensure that you have enough hard disk space on that machine. The Allocate Space slider does **not** display a remote machine's hard disk space; the `r;percentage of available disk space will be used` message displayed by the Allocate Space slider will be inaccurate if it appears at all.

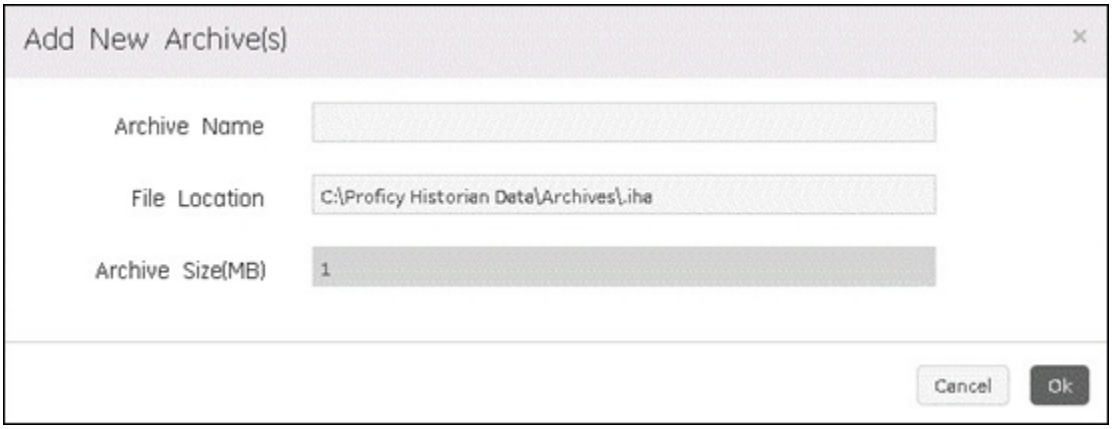
If you receive the error message `Runtime error 330 Invalid Property Value` while creating multiple archives on a remote machine, it is probably because you did not have enough hard disk space on that machine. When you select **OK** on the error message, Historian Administrator may disappear. You must now clean up the remote machine's hard disk space and restart Historian Administrator.

## Adding One or More Archives

You may need to add an archive when the current archive is almost full and you have not enabled automatic creation of archives.

**Note:**
Historian now supports a maximum archive size of 256 GB per archive. When the current archive is full, the system will write to the next archive in the sequence in which it was created. As of Historian 7.0, Historian archives are time-based only.

1. In the **Archives** section, select the icon.
   The Add New Archive(s) window appears.




2. In the **Archive Name** field, enter the name of the archive. The archive name must be the same as the filename.
3. In the **File Location** field, enter the path of the archive from a local drive or specify a UNC path.
4. In the **Archive Size (MB)** field, enter the size of the file in MB that you want to create.
5. Select **OK**.
6. Select **Cancel** to stop the operation.
   If you cancel the operation, any archives already created during this operation will be deleted.

## Back up Historian Archive Files

Back up your Historian archive files periodically to ensure your data is protected. Historian bundles alarms and events data with tag data in its backup files, and stores them as ZIP files. After an archive has been backed up, it can be stored to a shared network location, stored off-site, or written to physical media.

**Note:**

- Use Microsoft® Volume Shadow Copy Service to back up archives more than 2 GB in size. *(on page 2898)*.
- Ensure that you have enough hard drive space on your default backup location before backing up your archives.
- For Historian 6.0 or later clients, you can only back up time-based archives.

The .IHC file is automatically backed up when, and only when, you back up the "current" archive .IHA file. By default, the .IHC backup path is the same as the archives path. The .IHC uses the following naming convention: `ComputerName_Config-Backup.ihc` If the default backup path is different than the

archives path, the .IHC file is copied to the backup folder with the standard .IHC naming convention `ComputerName_Config.ihc`.

If you back up an archive more than once, the backup tool will (by default) attempt to use the same name for the backup file and will detect that an archive with the same name already exists. Rename the backup archive file or move the original backup archive file from the target backup directory.

## Backing up Archives using Historian

Best practice is to store archive **backups** in a different location than the archive **files**.

1. Open the **Archive Configuration** page.
2. In the **Archives** section, select an existing archive.
3. Select the **Backup** button.
   The Backing up Archive window appears.
4. Enter the **Archive Name**.
5. Save the backup file to the archive backup file location.
   A new Job Id is created and the details of the status are displayed in the Jobs Page.

## Including Alarm Data in Archive Backups

When backing up your Historian archives, any alarms that have a life cycle that overlaps the data archive being backed up will be included. This means that an alarm with a long life cycle can be included up in multiple backups. For example, say the following alarm and archive dates were the following:

| Alarm/Data Archive | Start Time | End Time |
|---|---|---|
| Alarm1 | 09/02/2004 | 09/06/2004 |
| Archive1 | 09/01/2004 | 09/03/2004 |
| Archive2 | 09/03/2004 | 09/04/2004 |
| Archive3 | 09/04/2004 | 09/06/2004 |

If any or all of these archives are backed up, Alarm1 will go into the backup for each one. When the archives are restored, Historian will analyze the included alarm data and, if the data is already in the Historian archive, is intelligent enough to know it already has the alarm.

Use the following procedure to change alarm timestamp checking.

1. From the **Start** menu, select **Run** and enter `Regedit`.
2. Open the following key folder `HKEY_LOCAL_MACHINE\SOFTWARE\Intellution, Inc.\iHistorian\Services\DataArchiver\`
3. Create a new DWORD called **AlarmTimestampCheck** and set its value to 1.
   Set **AlarmTimestampCheck** to 2 for slower timestamp checking. Set **AlarmTimestampCheck** to 0 to disable timestamp checking entirely.
4. Select **OK**.
5. Close the Registry Editor
6. Open Historian Administrator.
7. Restart the Data Archiver for the changes to take effect.

## Backing up Archives Using Volume Shadow Copy Service

Historian can use the Microsoft® Volume Shadow Copy Service to back up and restore large archive files reliably and in a short period of time without affecting the data collection. The Historian Data Archiver uses ihArchiverBackup.exe as the default backup system. If you want to back up your archiver files regularly, you can set the scheduler to back up your files automatically.

VSS provides fast volume capture of the state of a disk which is called a snapshot or shadow copy. When the snapshot is taken, disk writes are suspended for a brief period of time, typically on the order of milliseconds. After the snapshot, disk writes can resume, but the original state of the files are maintained by a difference file. The different file allows the state of the original file at the time of the snapshot to be reconstructed. This behavior allows files to be backed up while new data is being written to files.

If you are using ihArchiveBackup.exe before the upgrade, your backup will continue to work in the same or similar manner as it did before the upgrade. There is no change in the backup procedure and the Auto Recovery Backup Files option remains unchanged.

**Note:**
You can use both ihArchiveBackup.exe or VSS for backup, however, VSS is a better choice for larger archives to reduce the load on the Data Archiver service.

The Volume Shadow Copy feature is provided by Windows Operating System, and the instructions to use backup and restore vary depending on the backup application that is used in the Windows operating system.

Historian supports using the Volume Shadow Copy Service in the following operating systems.

- Microsoft® Windows® Server 2019 (64-bit)
- Microsoft® Windows® Server 2016 (64-bit)
- Microsoft® Windows® Server 2012 R2 (64-bit)
- Microsoft® Windows® 10 IoT (32-bit or 64-bit)
- Microsoft® Windows® 10 (32-bit or 64-bit)
- Microsoft® Windows® 8.1 Professional (32-bit or 64-bit)

Microsoft uses a backup format called Virtual Hard Disk (VHD) to back up files. If you create an archive backup using Microsoft® Volume Shadow Copy Service, you must first restore the archives files (that is, convert **.bkf** or **.vhd** into **.iha**) using the Windows Restore wizard, and then restore the archives (.iha) into Historian. For more information on restoring an archive (.iha) into Historian, refer to the Restoring an Archive *(on page 2899)* topic.
In addition to using the Backup and Restore wizard, you can also use the command line utilities:

- Refer to How to use command line parameters with the Ntbackup command: http://support.microsoft.com/kb/814583
- Refer to How to use command line parameters with the Wbadmin command: http://technet.microsoft.com/en-us/library/cc754015(WS.10).aspx
- If you need additional assistance about using Windows Server Backup Wizard, refer to Microsoft's website at: http://technet.microsoft.com/en-us/library/cc753528.aspx

## Restore Historian Archive Files

Under certain circumstances, you may want to restore tag and alarms and events data to Historian. This may be after an unplanned shutdown, or you may need to retrieve data from an old, inactive archive. You can restore only time-based archives.

**Warning:**
**Never** restore an archive to a production Historian server without a current archive already online.

**CAUTION:**
Restoring an archive is a resource-intensive operation and should be scheduled for non-peak usage times.

## Restoring Archives from Historian Backup Files

If it is not already present, **copy** the archive file to the default archive path. Leave the original backup file where it is. Archives that have been previously removed from Historian can be found in the `\Archives\Offline` directory. Archives on removable media must also be copied into the default archive path.

1. Open the **Archive Configuration** page.
2. Select the **Restore Archive** icon.
   The Restore Archive window appears.
3. In the **Archive Name** field, enter the name of the archive you want to restore.
4. In the **File Location** field, enter the path name of the archive from a local drive or specify a UNC path.
5. Verify that the file name and path are correct.
6. In the **Data Store** field, select the name of the data store to load the archive file into.
   If Historian is unable to find the specified data store, the file will be loaded to the default data store.
7. Select **OK**.
   The restored archive is moved to the `\Archive` directory and is made available for querying.

# Chapter 43. Extract, Transform, and Load (ETL)

## Overview of the Historian ETL Tools

Transferring data from one Historian server to another is typically performed by Proficy Historian collectors. These collectors provide a connected streaming data transfer mechanism (except the Calculation and File collectors). In a system where a steady network connection is not possible or not cost-effective, a periodic file-oriented data transfer is preferred. The Historian Extract, Transform, and Load (ETL) tools consist of a comprehensive set of file-oriented data extraction, transfer, and loading tools.

**Potential ways of using ETL tools:**

- Data transfer from any of the following sources:
  - An eDNA server
  - An ODBC data source
  - Proficy Historian
  - PI Historian
- Data transfer via radio or low bandwidth cellular connection
- Data transfer where there is no connectivity (read and write using portable media)
- Data transfer for periodic connectivity applications (for example, ships can transfer data when they arrive at a port)
- Data migration from OSI PI Server to Proficy Historian
- Data extraction to import into other applications
- Data import from other applications

**Components of Historian ETL:**

- **Extract:** Using this tool, you can extract time series data from an eDNA server, an ODBC data source, Proficy Historian or PI Server. For Proficy Historian, you can also extract alarms and events data, perform scaling and absolute deadband compression.
- **Transform:** Using this tool, you can transfer data from an onsite Historian server or an ODBC data source to the destination Historian server using a file-sharing application such as FTP, BITS, and so on.
- **Load:** Using this tool, you can load data into Proficy Historian. This tool monitors a file directory, unzips the files, and processes them.

Depending on the use case, you can use these tools independently or together.

**Limitations:**

- The ETL tools do not support array data.

## Workflow for Transferring Data from eDNA

To transfer data from an eDNA server, you must perform the following steps.

| Step Number | Description | Notes |
|---|---|---|
| 1 | Install Historian ETL *(on page 202)*. | This step is required. You must install ETL on both the source and destination machines of the data transfer.<br><br>**Note:**<br>If you want to upgrade ETL:<br>1. Uninstall the existing version of ETL.<br>2. Backup the configuration files, and delete them.<br>3. Install the latest version of ETL. |
| 2 | Specify the tags whose data you want to extract. You can do so using the utility *(on page 2911)*, creating the configuration file manually *(on page 2912)*, using a template *(on page 2913)*, or using a blank spreadsheet *(on page 2914)*. | This step is required. It involves providing a list of tags whose data you want to extract. |
| 3 | Configure the extract settings *(on page 2933)*. | This step is required. It involves providing the .xml files that contain the tags list and the tables list, along with other parameters. |
| 4 | Start the data extraction *(on page 2937)*. | This step is required. It involves extracting tag data and compressing it so that it can |

| Step Number | Description | Notes |
|---|---|---|
| | | be transferred to the destination Historian server. |
| 5 | Transfer the data using BITS *(on page 2957)*, FTP *(on page 2959)*, or any other file-sharing application. | This step is required. It involves setting up the file-sharing application that you want to use and then transferring the data to the machine on which the destination Historian server is installed. |
| 6 | Load the data *(on page 2961)* into the destination Historian server. | This step is optional. It involves extracting the .zip files transferred by the file-sharing application and then loading the data into the destination Historian server. |

**Note:**
Depending on the use case, you can use these tools independently or together.

## Workflow for Transferring Data from an ODBC Data Source

To transfer data from an ODBC data source, you must perform the following steps.

| Step Number | Description | Notes |
|---|---|---|
| 1 | Install Historian ETL *(on page 202)*. | This step is required. You must install ETL on both the source and destination machines of the data transfer.<br>**Note:**<br>If you want to upgrade ETL: |

| Step Number | Description | Notes |
|---|---|---|
| | | 1. Uninstall the existing version of ETL.<br>2. Backup the configuration files, and delete them.<br>3. Install the latest version of ETL. |
| 2 | Specify the tags and tables whose data you want to extract. You can do so manually *(on page 2924)*, using a template *(on page 2926)*, or using a blank spreadsheet *(on page 2928)*. | This step is required. It involves providing a list of tags whose data you want to extract and the tables from which you want to extract data. |
| 3 | Configure the extract settings *(on page 2933)*. | This step is required. It involves providing the .xml files that contain the tags list and the tables list, along with other parameters. |
| 4 | Start the data extraction *(on page 2937)*. | This step is required. It involves extracting tag data and compressing it so that it can be transferred to the destination Historian server. |
| 5 | Transfer the data using BITS *(on page 2957)*, FTP *(on page 2959)*, or any other file-sharing application. | This step is required. It involves setting up the file-sharing application that you want to use and then transferring the data to the machine on which the destination Historian server is installed. |
| 6 | Load the data *(on page 2961)* into the destination Historian server. | This step is optional. It involves extracting the .zip files transferred by the file-sharing application and then loading the data into the destination Historian server. |

**Note:**
Depending on the use case, you can use these tools independently or together.

## Workflow for Transferring Data from Proficy Historian

To transfer data from Proficy Historian, you must perform the following steps.

| Step Number | Description | Notes |
|---|---|---|
| 1 | Install Historian ETL *(on page 202)*. | This step is required. You must install ETL on both the source and destination machines of the data transfer.<br><br>**Note:**<br>If you want to upgrade ETL:<br>1. Uninstall the existing version of ETL.<br>2. Backup the configuration files, and delete them.<br>3. Install the latest version of ETL. |
| 2 | Specify the tags and tables whose data you want to extract. You can do so manually *(on page 2938)*, using a template *(on page 2939)*, or using a blank spreadsheet *(on page 2940)*. | This step is required. It involves providing a list of tags whose data you want to extract and the tables from which you want to extract data. |
| 3 | Configure the extract settings *(on page 2942)*. | This step is required. It involves providing the .xml files that contain the tags list and the tables list, along with other parameters. You can also extract alarms and events data. |
| 4 | Start the data extraction *(on page 2947)*. | This step is required. It involves extracting tag data and com- |

| Step Number | Description | Notes |
|---|---|---|
| | | pressing it so that it can be transferred to the destination Historian server. |
| 5 | Transfer the data using BITS *(on page 2957)*, FTP *(on page 2959)*, or any other file-sharing application. | This step is required. It involves setting up the file-sharing application that you want to use and then transferring the data to the machine on which the destination Historian server is installed. |
| 6 | Load the data *(on page 2961)* into the destination Historian server. | This step is optional. It involves extracting the .zip files transferred by the file-sharing application and then loading the data into the destination Historian server. |

**Note:**
Depending on the use case, you can use these tools independently or together.

## Workflow for Transferring Data from PI Historian

To transfer data from PI Historian, you must perform the following steps.

| Step Number | Description | Notes |
|---|---|---|
| 1 | Install Historian ETL *(on page 202)*. | This step is required. You must install ETL on both the source and destination machines of the data transfer.<br>**Note:**<br>If you want to upgrade ETL: |

| Step Number | Description | Notes |
|---|---|---|
| | | 1. Uninstall the existing version of ETL.<br>2. Backup the configuration files, and delete them.<br>3. Install the latest version of ETL. |
| 2 | Specify the tags and tables whose data you want to extract. You can do so manually *(on page 2948)*, using a template *(on page 2949)*, or using a blank spreadsheet *(on page 2950)*. | This step is required. It involves providing a list of tags whose data you want to extract and the tables from which you want to extract data. |
| 3 | Configure the extract settings *(on page 2952)*. | This step is required. It involves providing the .xml files that contain the tags list and the tables list, along with other parameters. |
| 4 | Start the data extraction *(on page 2957)*. | This step is required. It involves extracting tag data and compressing it so that it can be transferred to the destination Historian server. |
| 5 | Transfer the data using BITS *(on page 2957)*, FTP *(on page 2959)*, or any other file-sharing application. | This step is required. It involves setting up the file-sharing application that you want to use and then transferring the data to the machine on which the destination Historian server is installed. |
| 6 | Load the data *(on page 2961)* into the destination Historian server. | This step is optional. It involves extracting the .zip files transferred by the file-sharing application and then loading the data in- |

| Step Number | Description | Notes |
|---|---|---|
| | | to the destination Historian server. |

**Note:**
Depending on the use case, you can use these tools independently or together.

# Install the Historian ETL Tools

- If you want to use the Historian Extract, Transform, and Load (ETL) tools to transfer data from a PI Historian server, install the PI SDK package.
- If you want to use the ETL tools to transfer data from an eDNA server, copy the following eDNA binaries to the `<installation drive>\Program Files\GE Digital\Historian ETL eDNA Extract` folder:
  - `EzDnaApi.dll`
  - `EzDNAApiNet.dll`

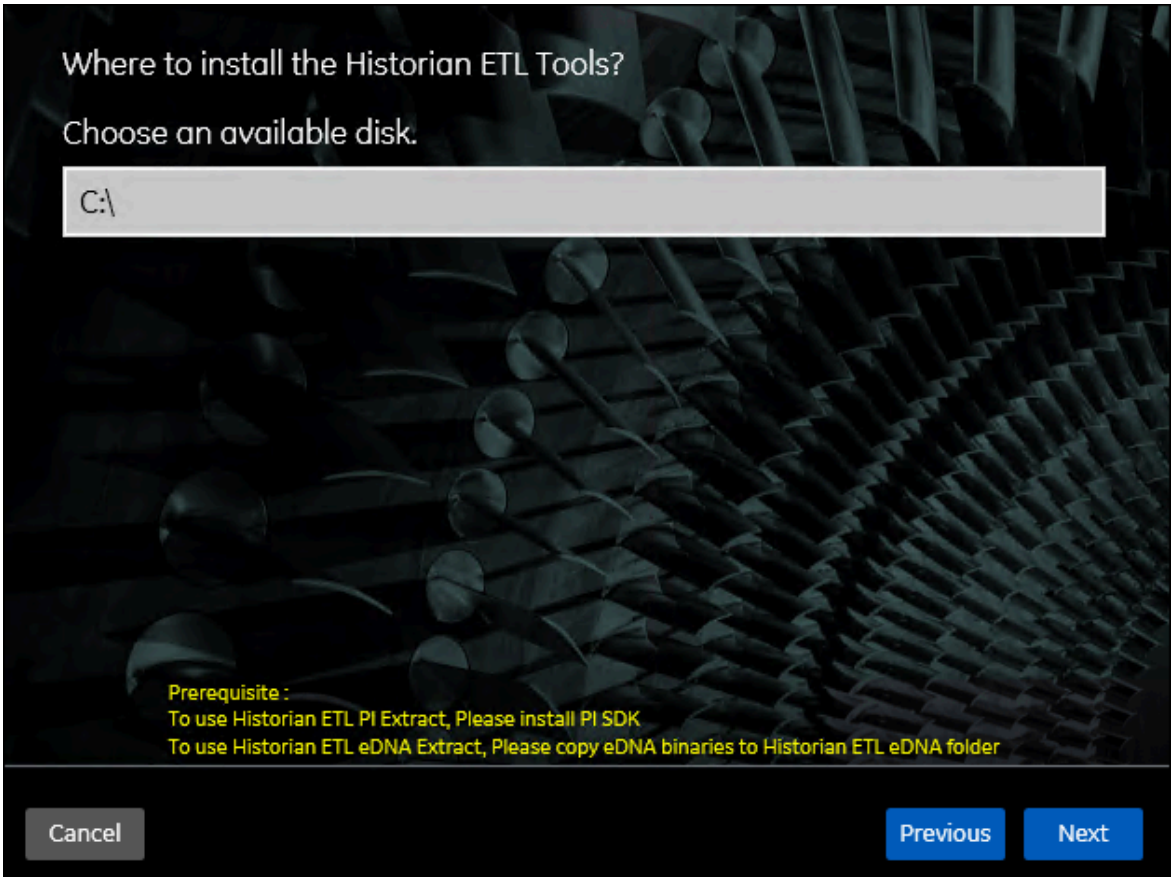
  By default, these files are available in the following folder on the machine on which the eDNA server is installed: `C:\Program Files(x86)\eDNA`

Installing ETL installs the following tools:

- The Extract tool
- The Transform tool
- The Load tool

This topic describes how to install ETL to extract, transform, and load data from an onsite Historian machine to the destination Historian server. You must install Historian ETL on both the onsite Historian machine and the destination Historian server (that is, the source and destination machines for data transfer).

1. Run the `InstallLauncher.exe` file.
2. Select **Install Historian ETL Tools**.
   The welcome page appears.
3. Select **Next**.
   The license agreement appears.
4. Select the **Accept** check box, and then select **Next**.
   The default installation drive appears.




5. If required, modify the installation drive for Historian ETL, and then select **Next**.
   A message appears, stating that you are ready to install ETL.
6. Select **Install**.

The Historian ETL tools are installed on your machine.

- The following folders are created in the `<installation drive>/Program Files/GE Digital` folder:
  - `Historian ETL eDNA Extract`
  - `Historian ETL Extract`
  - `Historian ETL Load`
  - `Historian ETL ODBC Extract`
  - `Historian ETL PI Extract`
  - `Historian ETL Transform`
- The following services are installed:
  - Historian ETL eDNA Extract
  - Historian ETL Extract
  - Historian ETL ODBC Extract_x64

    - Historian ETL ODBC Extract_x86
    - Historian ETL Load
    - Historian ETL PI Extract
- The following registry paths are created:
    - `HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\GE Digital\Historian ETL eDNA Extract`
    - `HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\GE Digital\Historian ETL ODBC Extract`
    - `HKEY_LOCAL_MACHINE\SOFTWARE\GE Digital\Historian ETL Extract`
    - `HKEY_LOCAL_MACHINE\SOFTWARE\GE Digital\Historian ETL ODBC Extract`
    - `HKEY_LOCAL_MACHINE\SOFTWARE\GE Digital\Historian ETL PI Extract`
    - `HKEY_LOCAL_MACHINE\SOFTWARE\GE Digital\Historian ETL Load`

1. If you want to extract data from an eDNA server, copy the `Customdna.ini` file in the eDNA server into the `C:\Windows` folder on the machine on which you have installed ETL.
2. Extract data from an eDNA server *(on page 2910)*, ODBC data source *(on page 2924)*, Proficy Historian *(on page 2938)*, or PI Historian *(on page 2948)*.

## Upgrade the ETL Tools

1. Uninstall the existing version of Historian ETL.
2. Backup the configuration files, and delete them.
3. Install the latest version of Historian ETL *(on page 202)*.

# About Extracting Data from an eDNA Server

The Historian ETL eDNA Extract tool extracts data as follows:

1. Extracts data related to tags into text files, which are named in the following format: `YYYYDDMMHHRR_<OSM name>.txt`. These files are stored in the following folder: `<installation drive>\Historian ETL eDNA Extract\Hist Files`.

   You can only extract tag data; you cannot extract alarms and events data.

2. After a specified number of files are extracted (by default, 6), the files are compressed into a .zip file, which is named in the following format: `YYYYDDMMHHRR_<OSM name>.zip`. These files are stored in the following folder: `<installation drive>\Program Files\GE Digital\Historian ETL eDNA Extract\Zip Files`.
3. Deletes the text files in the `<installation drive>\Program Files\GE Digital\Historian ETL eDNA Extract\Hist Files` folder after they are compressed.

## Specify Tags for the eDNA Data Extraction Using a Utility

Before you extract data, you must specify the tags whose data you want to extract.

This topic describes how to specify the tags by generating the tag configuration file automatically using the Historian ETL eDNA Extract utility. Alternatively, you can specify the tags by creating a configuration file manually *(on page 2912)*, using a template spreadsheet *(on page 2913)*, or using a blank spreadsheet *(on page 2914)*.

1. Run the `HistorianETLeDNAExtractConfigTool` file located in the `<installation drive>\Program Files\GE Digital\Historian ETL eDNA Extract` folder.

   **Tip:**
   You can also enter ETL eDNA Extract in Windows Run.

   The **Historian ETL eDNA Extract Settings** window appears, displaying the **Basic Configuration** section.
2. Select **Files**, and then provide values as specified in the following table.

| Field | Description |
|---|---|
| **Service Name** | Enter the service name used by the Historian ETL eDNA Extract tool to connect to the eDNA server. You can enter multiple values separated by commas. |
| **Tag Name** | Specify whether you want to use the extended ID or the long ID in the tag name. If you select **ExtendedID**, the name of the tag created in Historian will contain the extended ID (and the spare field of the tag will contain the long ID). Similarly, if you select **LongID**, the name of the tag created in Historian will contain the long ID (and the spare field of the tag will contain the extended ID). |
| **OSM Name** | Enter a name to identify the Original Equipment/Sales Manufacturer (OSM). This name is included in the .zip files that are extracted on this machine. |
| **Zip TagConfig** | Select **True** if you want to create a .zip file of the metadata of eDNA tags. It is a good practise to create a .zip file of the metadata. |

3. Select **Tag Config File**.
   A file named `TagConfig.xml` is generated in the `<installation drive>:\Program Files\GE Digital\Historian ETL eDNA Extract` folder. This file contains the list of tags and their properties.

Configure the eDNA Data Extraction settings *(on page 2916)*.

## Specify Tags for eDNA Data Extraction Manually

Before you extract data, you must specify the tags whose data you want to extract.

This topic describes how to specify the tags by creating a configuration file manually. Alternatively, you can specify the tags using the utility *(on page 2911)*, a template spreadsheet *(on page 2913)*, or a blank spreadsheet *(on page 2914)*.

1. Run the `HistorianETLeDNAExtractConfigTool` file located in the `<installation drive>\Program Files\GE Digital\Historian ETL eDNA Extract` folder.

   **Tip:**
   You can also enter ETL eDNA Extract in Windows Run.

   The **Historian ETL eDNA Extract Settings** window appears, displaying the **Basic Configuration** section.
2. Select **Files**, and then select **Sample Tag Config**.
   A sample tag configuration .xml file is created.
3. Include the tag properties *(on page 2916)* for all the tags whose data you want to extract.
   Tag properties for a tag named voltage used in an eDNA server:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Taglist>
    <Tag Name="SUBSTN.TATANAGR.BUS.6B1.MES1.KV">
        <LocalName>A0000001</LocalName>
        <RemmoteName>SUBSTN.TATANAGR.BUS.6B1.MES1.KV</RemoveName>
        <ServiceName>JHARMP.SCADA1</ServiceName>
  <DataType>Float</DataType>
        <RemoteNameDesc>SUBSTN.TATANAGR.BUS</RemoteNameDesc>
        <RemoteNameSpare4>SUBSTN.TATANAGR.BUS</RemoteNameSpare4>
        <RemoteNameSpare5>SUBSTN.TATANAGR.BUS</RemoteNameSpare5>
        <RemoteNameEnggUnits>MW</RemoteNameEnggUnits>
```

```
        <TagName>ExtendedID</TagName>
    </Tag>
</Taglist>
```

4. In the `HistorianETLeDNAExtractConfigTool` file located at `<installation drive>\Program Files\GE Digital\Historian ETL eDNA Extract`, select **Files**, and then in the **Tag Configuration File** field, provide the path to the configuration file that you have created, and then select **Save**.
   The tag configuration file is specified.

Configure the eDNA Data Extraction settings *(on page 2916)*.

## Specify Tags for eDNA Data Extraction Using a Template

Before you extract data, you must specify the tags whose data you want to extract.

This topic describes how to specify the tags using a template spreadsheet, which is provided with the Historian ETL package. Alternatively, you can specify the tags using the utility *(on page 2911)*, using a blank spreadsheet *(on page 2914)*, or by creating a configuration file manually *(on page 2912)*.

1. Access the `eDNAHistTagConfigGenerateExcel.xlsx` file located in the `Historian eDNA ETL Extract` folder.
2. **Optional:** If you want to modify an existing tag configuration file, you can import the data from that .xml file by selecting **Developer > Import**.

   **Tip:**
   If the **Developer** tab is not available, right-click the menu bar of the spreadsheet, select **Customize the Ribbon**, and then select the **Developer** check box.

   Data from the .xml file is imported into the spreadsheet.

   **Note:**
   If an error occurs, stating that the data is not valid according to the schema, verify that:
   - You have provided values in all the red-colored columns.
   - There are no blank rows.
   - The values that you have entered are valid and of the same data type as defined for each property.

   For details, refer to Tag Properties *(on page 2916)*.

3. For each tag, enter or modify values in the columns for the tag properties *(on page 2916)*. A value is required in the red-colored columns.
4. Save the file.
5. Select **Developer > Export**.

> **Tip:**
> If the **Developer** tab is not available, right-click the menu bar of the spreadsheet, select **Customize the Ribbon**, and then select the **Developer** check box.

6. Enter a name and location for the .xml file.
   The tag configuration file is created with the list of tags and their properties to be extracted.
   If an error occurs, stating that the data is not valid according to the schema, verify that:
   - You have provided values in all the red-colored columns.
   - Values in the Name column are unique.
   - There are no blank rows.
   - The values that you have entered are valid and of the same data type as defined for each property.

   For details, refer to Tag Properties *(on page 2916)*.
7. In the `HistorianETLeDNAExtractConfigTool` file located at `<`*`installation drive`*`>\Program Files\GE Digital\Historian ETL eDNA Extract`, select **Files**, and then in the **Tag Configuration File** field, provide the path to the configuration file that you have created, and then select **Save**.
   The tag configuration file is specified.

Configure the eDNA Data Extraction settings *(on page 2916)*.

## Specify Tags for eDNA Data Extraction Using a Blank Spreadsheet

Before you extract data, you must specify the tags whose data you want to extract.

This topic describes how to specify the tags manually by creating a configuration file. Alternatively, you can specify the tags using the utility *(on page 2911)*, a template spreadsheet *(on page 2913)* or a blank spreadsheet *(on page 2914)*.

1. Create a Microsoft Excel file.
2. Enter column names matching the names of the tag properties *(on page 2916)*.
3. **Optional:** If you want to modify an existing tag configuration file, you can import the data from that .xml file by selecting **Developer > Import**.

> **Tip:**
> If the **Developer** tab is not available, right-click the menu bar of the spreadsheet, select **Customize the Ribbon**, and then select the **Developer** check box.

> **Note:**
> If a message appears, stating that the .xml file is not linked to schema, ignore the message.

4. Select **Developer > Source > XML Maps > Add**.
5. Select the `eDNAHistTagConfigSchema.xsd` file located in the `<installation drive>:\Program Files\GE Digital\Historian ETL eDNA Extract` folder.
6. Map each column name with each entry under **Taglist** in the **XML Source** section. To do so, select each column name, and then double-click the corresponding property under **Taglist**.
   Each property under **Taglist** changes to bold formatting, indicating that it is mapped to the corresponding column.
7. For each tag, enter or modify values in the columns.
8. Select **Design > Properties**, select the **Validate data against schema for import and export** check box, and then select **OK**.
9. Save the file.
10. Select **Developer > Export**.
11. Enter a name and location for the .xml file.
    The tag configuration file is created with the list of tags and their properties to be extracted.
    If an error occurs, stating that the data is not valid according to the schema, verify that:
    - You have provided values in all the red-colored columns.
    - Values in the Name column are unique.
    - There are no blank rows.
    - The values that you have entered are valid and of the same data type as defined for each property.

    For details, refer to Tag Properties *(on page 2916)*.
12. In the `HistorianETLeDNAExtractConfigTool` file located at `<installation drive>\Program Files\GE Digital\Historian ETL eDNA Extract`, select **Files**, and then in the **Tag Configuration File** field, provide the path to the configuration file that you have created, and then select **Save**.
    The tag configuration file is specified.

Configure the eDNA Data Extraction settings *(on page 2916)*.

## Tag Properties for eDNA Data Extraction

This topic provides a list of tag properties that you can define for each tag that you want to extract from an eDNA server.

| Column Name | Data Type | Description |
|---|---|---|
| Name | String | Enter the name of the tag. A value is required and must be unique. |
| LocalName | String | Enter a unique identifier for the tag. A value is required and must be unique. |
| RemoteName | String | Enter the name of the tag that you want to use after it is imported to Historian. A value is required. |
| ServiceName | String | Enter the service name used to connect to the eDNA server. You can enter multiple values separated by commas. A value is required. |
| DataType | String | Enter the data type of the tag. A value is required. |
| RemoteNameDesc | String | Enter the description of the remote name. A value is required. |
| RemoteNameSpare4 | String | Enter the longID or extendedID of the tag. A value is required. |
| RemoteNameSpare5 | String | Enter a description for the value in the RemoteNameSpare4 parameter. A value is required. |
| RemoteNameEnggUnits | String | Enter the engineering units of the tag values. A value is required. |
| TagName | String | Specify whether you want to use the extended ID or the long ID in the tag name. |

## Configure the eDNA Data Extraction Settings

Specify the tags whose data you want to extract from an eDNA server. You can do so by using the utility *(on page 2911)*, creating a configuration file manually *(on page 2912)*, using a template *(on page 2913)*, or using a blank spreadsheet *(on page 2914)*.

This topic describes how to configure the settings for eDNA data extraction. These settings are saved in the `eDNAHistorianETLExtract.exe.config` file.

1. Run the `HistorianETLeDNAExtractConfigTool` file located in the `<installation drive>\Program Files\GE Digital\Historian ETL eDNA Extract` folder.

   > **Tip:**
   > You can also enter ETL eDNA Extract in Windows Run.

   The **Historian ETL eDNA Extract Settings** window appears, displaying the **Basic Configuration** section.
2. If the configuration details are stored in a file, select **Import Config** to import the settings, and skip to step 6.
3. Provide values as specified in the following table.

| Field | Description | Default Value |
|---|---|---|
| **Service Name** | Enter the service name used by the Historian ETL eDNA Extract tool to connect to the eDNA server. You can enter multiple values separated by commas. | |
| **Tag Name** | Specify whether you want to use the extended ID or the long ID in the tag name. If you select **ExtendedID**, the name of the tag created in Historian will contain the extended ID (and the spare field of the tag will contain the long ID). Similarly, if you select **LongID**, the name of the tag created in Historian will contain the long ID (and the spare field of the tag will contain the extended ID). | |
| **Run Interval (Seconds)** | Enter the interval, in seconds, at which the Historian ETL eDNA Extract tool will extract data. You must enter a value greater than or equal to 60. | 150 (that is, a text file is created for data that is extracted in 150 seconds) |
| **Min # of Files to Compress** | Enter the number of files that must be compressed into a single .zip file. | 6 (that is, a .zip file is created for every six text files) |
| **Extract Type** | Specifies whether you want to extract only the current data or historical data or both. | Both (Historical and Current) |

| Field | Description | Default Value |
|---|---|---|
| **OSM Name** | Enter a name to identify the Original Equipment/Sales Manufacturer (OSM). This name is included in the .zip files that are extracted on this machine. | Host name of the local machine |
| **Zip TagConfig** | Select **True** if you want to create a .zip file of the metadata of eDNA tags. | |

4. Select **Files**, and then provide values as specified in the following table.

| Field | Description | Default Value |
|---|---|---|
| **Historian Export Path** | Enter the path to the folder in which the text files containing the extracted data must be stored. | `<installation drive>\Program Files\GE Digital\Historian ETL eDNA Extract\Hist Files` |
| **Tag Configuration File** | Enter the path to the tag configuration that you have created. You can create it automatically *(on page 2911)*, manually *(on page 2912)*, a template spreadsheet *(on page 2913)*, or using a blank spreadsheet *(on page 2914)*. | `<installation drive>\Program Files\GE Digital\Historian ETL eDNA Extract\TagConfiguration.xml` |
| **Zip Export Path** | Enter the path to the folder in which the compressed files must be stored. | `<installation drive>\Program Files\GE Digital\Historian ETL eDNA Extract\Zip Files` |
| **State File** | Enter the path to the file that the Historian ETL eDNA Extract tool will create to store the timestamp of the last successful export. This timestamp is used to identify the start time for the next iteration of extraction. This ensures that there is no loss of data during extraction. | `<installation drive>\Program Files\GE Digital\Historian ETL eDNA Extract\State.xml` |

| Field | Description | Default Value |
|---|---|---|
| | For example, suppose the current time is 11am, and data has been extracted only till 9am. The state file contains the timestamp for 9am. Therefore, when data extraction is resumed, it is extracted from 9am.<br><br>The state file is created after you apply the Historian ETL eDNA Extract settings. It is updated each time .zip files are transferred to the destination Historian server or when the Historian ETL eDNA Extract tool is stopped. For a sample state file, refer to Example of a State File *(on page 2968)*. | |
| **Regen File** | Enter the path to the regeneration file. You can use this field to extract data.<br><br>If the **Gen Sample Regen File** field is set to **True**, a sample regeneration file will be created after you apply the settings. You can modify this file as needed, and specify the path of the same file in the **Regen File** field. For a sample regeneration file, refer to Example of a Regeneration File *(on page 2966)*. | `<installation drive>\Program Files\GE Digital\Historian ETL eDNA Extract\Regen.xml` |
| **Gen Sample Regen File** | Specify whether you want to generate the sample regeneration file. You can then modify this file as needed. | False |
| **Sample Regen File** | Enter the path to the sample regeneration file that will be created if the **Gen Sample Regen File** field is set to **True**. You can modify this file as needed. | `<installation drive>\Program Files\GE Digital\Historian ETL eDNA Extract\Sample_Regen.xml` |

| Field | Description | Default Value |
|---|---|---|
| **Tag Config File** | Select this button to generate a file with the list of tags from which you want to extract data. For instructions, refer to Specify Tags for the eDNA Data Extraction Using a Utility *(on page 2911)*. | |
| **Sample Tag Config** | Select this button to generate a sample tag configuration file. You can then include the list of tags from which you want to extract data. For instructions, refer to Specify Tags for eDNA Data Extraction Manually *(on page 2912)*. | |

5. Select **Advanced Configuration**, and then provide values as specified in the following table.

| Field | Description | Default Value |
|---|---|---|
| **Log Level** | Select the log level to indicate the amount of information to be logged. The following options are available:<br>◦ Info<br>◦ Error<br>◦ Debug | Info |
| **Delay Interval** | The duration, in seconds, by which the data retrieval time will be reduced. For example, if data will be retrieved for 10 minutes, and if you enter 60 in this field, data for the last 60 seconds will not be retrieved in that batch; it will be retrieved in the next batch. This will ensure the retrieval of any dynamic records that were updated in that duration. | 60 |
| **Catch Up Interval (Minutes)** | Enter the catchup interval, in minutes, used to size files when catching up to the current time. | 10 |

| Field | Description | Default Value |
|---|---|---|
| **Save Limit** | Enter the number of files to be exported after which the `State.xml` file must be updated. | 20 |
| **Catch Up Time Limit (Hours)** | Enter the maximum time, in hours, to go back when catching up after a restart. If you want to extract historical data as well, enter the duration for which you want to extract the data. For example, if you want to extract data for the past one day, enter 24. | 168 |
| **Quality Conversion Good** | Enter the mapping details for good quality between eDNA hexadecimal codes and SCADA binary codes.<br><br>For example, if the hexadecimal code for good quality is 8203, and if the binary codes in a SCADA system for good quality are 1, 27, and 30, enter 8203=1,27,30.<br><br>Use a semicolon to separate multiple entries (for example, 8203=27,30,1;8003=30;203=34).<br><br>If this field is blank, all the values are considered as bad quality, without storing any binary digits. | |
| **Quality Conversion Bad** | Enter the mapping details for bad quality between eDNA hexadecimal codes and SCADA binary codes.<br><br>For example, if the hexadecimal code for bad quality is C003, and if the binary codes in a SCADA system for good quality are 2, 4, and 6, enter C003=2,4,6.<br><br>Use a semicolon to separate multiple entries (for example, C003=2,4,6;C406=8,9). | |

| Field | Description | Default Value |
|---|---|---|
| | If this field is blank, all the values are considered as bad quality, without storing any binary digits. | |

6. Select **Save**.
   The changes to the settings are applied and saved in the `HistorianETLeDNAExtract.exe.config` file.

Start the data extraction *(on page 2923)*.

## Extract Historical eDNA Data

By default, the ETL tools extracts both current and historical data. This topic describes how to configure the settings to extract historical data using the ETL tools.

1. Create a regeneration file, specifying the start time, end time, and interval for which you want to capture the historical data. For a sample regeneration file, refer to Example of a Regeneration File *(on page 2966)*.
2. Run the `HistorianETLeDNAExtractConfigTool` file located in the `<installation drive>\Program Files\GE Digital\Historian ETL eDNA Extract` folder.

   **Tip:**
   You can also enter ETL Historian eDNA Extract in the Windows Start menu.

   The **Historian ETL Extract Configuration** window appears, displaying the **Basic Configuration** section.
3. Select **Advanced Configuration**, and then provide values as specified in the following table.

| Field | Description | Default Value |
|---|---|---|
| **Regen File** | Enter the path to the regeneration file that you have created. | `<installation drive>\Program Files\GE Digital\Historian ETL eDNA Extract\Regen.xml` |
| **Catch Up Time Limit** | Enter the duration, in hours, for which you want to extract historical data. | 168 |

| Field | Description | Default Value |
|---|---|---|
| | For example, if you want to extract data for the past one day, enter 24. | |
| **Temp Regen File** | Enter the path to the temporary regeneration file. | `<installation drive>\Program Files\GE Digital\Historian ETL eDNA Extract\Temp_Regen.xml` |
| **Generate Temp Regen File** | Select **True**. | False |

4. As needed, provide values in the remaining fields *(on page 2942)*.

Start the data extraction *(on page 2923)*.

## Start the eDNA Data Extraction

- Configure the Historian ETL eDNA Extract settings *(on page 2916)*.
- If the machine on which you want to extract the data is different from the eDNA sever, access the `hosts` file located in `C:\Windows\System32\drivers\etc`, and add the host name and IP address of the eDNA server in the following format:

  ```
  <IP address> <eDNA service directory name>
  ```

  For example: 100.300.50.7 dna_svcdir_01

1. Run the `HistorianETLeDNAExtractConfigTool` file located in the `<installation drive>\Program Files\GE Digital\Historian ETL eDNA Extract` folder.

   **Tip:**
   You can also enter ETL Historian eDNA Extract in the Windows Start menu.

   The **Historian ETL eDNA Extract Configuration** window appears.
2. Select **Start Service**.
   Data extraction from the eDNA server begins.

   **Tip:**
   If the tool does not start as expected, access the logs using Windows Event Viewer.

Transfer data using BITS *(on page 2957)*, FTP *(on page 2959)*, or any other file-sharing application.

# About Extracting Data from an ODBC Data Source

The Historian ETL ODBC Extract tool extracts data as follows:

1. Extracts data related to tags into text files, which are named in the following format: `YYYYDDMMHHRR_<onsite Historian computer name>.txt`. These files are stored in the following folder: `<installation drive>\Program Files\GE Digital\Historian ETL ODBC Extract\Hist Files`.

   You can only extract tag data; you cannot extract alarms and events data.
2. After a specified number of files are extracted (by default, 6), the files are compressed into a .zip file, which is named in the following format: `YYYYDDMMHHRR_<DSN>.zip`. These files are stored in the following folder: `<installation drive>\Program Files\GE Digital\Historian ETL ODBC Extract\Zip Files`.
3. Deletes the text files in the `<installation drive>\Program Files\GE Digital \Historian ETL ODBC Extract\Hist Files` folder after they are compressed.

## Specify Tags and Tables for ODBC Data Extraction Manually

Before you extract data, you must specify the tags and tables whose data you want to extract.

This topic describes how to specify the tags and tables manually by creating a configuration file. Alternatively, you can specify them using a template spreadsheet *(on page 2926)* or a new spreadsheet *(on page 2928)*.

1. Create an .xml file to include the tag properties *(on page 2931)* for all the tags whose data you want to extract.
   Tag properties for a tag named Pressure used in an ODBC data source:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Taglist>
    <Tag Name="Pressure">
        <LocalName>ValvePressure</LocalName>
        <RemoteName>ValvePressure</RemoteName>
        <DataType>int</DataType>
        <TableName>Valve_Pressure_Table</TableName>
    </Tag>
</Taglist>
```

2. Create another .xml file to include the column names of the table for each tag whose data you want to extract. This file is used to identify the column names in the database table whose data you want to extract. For a list of table properties that you can specify, refer to Table Properties for an ODBC Data Source *(on page 2932)*.
   Suppose you want to extract values of a tag named Pressure from a table named Valve_Pressure_Table. Suppose this table contains the following columns in the database:
   - TagName
   - TagQuality
   - TagTimestamp
   - TagValue

   In this case, provide these column names as follows:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TableList>
    <Table>
        <TableName>Valve_Pressure_Table</TableName>
        <TagName>TagName</TagName>
        <Timestamp>TagTimestamp</Timestamp>
        <Value>TagValue</Value>
        <Quality>TagQuality</Quality>
        <Condition></Condition>
    </Table>
</TableList>
```

   In addition, you can specify other properties to determine the quality of each value. For example, suppose you want to define the quality values as follows:

| Tag Value | Quality |
|---|---|
| 0-50 | Good |
| 51-100 | Bad |
| 101 and 102 | Uncertain |
| 103, 104, and 105 | Not applicable |

   In that case, provide these values as follows:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TableList>
    <Table>
```

```
        <TableName>PressureTable</TableName>
        <TagName>TagName</TagName>
        <Timestamp>TagTimestamp</Timestamp>
        <Value>TagValue</Value>
        <Quality>TagQuality</Quality>
        <QualityStatus>
            <ihOPCBad>0-50</ihOPCBad>
            <ihOPCGood>51-100</ihOPCGood>
            <ihOPCUncertain>101,102</ihOPCUncertain>
            <ihOPCNA>103,104,105</ihOPCNA>
        </QualityStatus>
        <Condition></Condition>
    </Table>
</TableList>
```

When you do so, if the tag value is, say, 56, the quality of the value is stored as Good in Historian. However, if this value does not match the value in the Quality column, the value is stored as Bad in Historian.

3. Verify that all the tags and tables that you specify in the first file contain the corresponding entries in the second file.

Configure the Historian ETL ODBC Extract settings *(on page 2933)*.

## Specify Tags and Tables ODBC Data Extraction Using a Template

Before you extract data, you must specify the tags and tables whose data you want to extract.

This topic describes how to specify the tags using a template spreadsheet, which is provided with the Historian ETL package. Alternatively, you can specify the tags using a new spreadsheet *(on page 2928)* or by creating a configuration file manually *(on page 2924)*.

1. Access the `ETLODBCConfigXMLGenerate.xlsx` file located in the `Historian ETL ODBC Extract` folder.
2. **Optional:** If you want to modify an existing tag configuration file, you can import the data from that .xml file by selecting **Developer > Import**. Similarly, if you want to modify an existing table configuration file, import the data into a separate worksheet.

> **Tip:**
> If the **Developer** tab is not available, right-click the menu bar of the spreadsheet, select **Customize the Ribbon**, and then select the **Developer** check box.

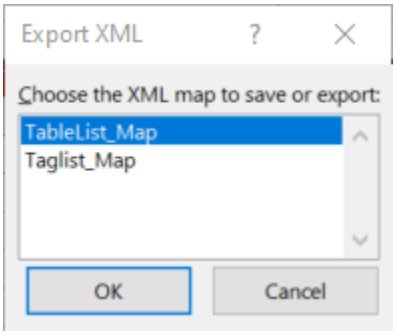
Data from the .xml file is imported.

> **Note:**
> If an error occurs, stating that the data is not valid according to the schema, verify that:
> - You have provided values in all the red-colored columns.
> - There are no blank rows.
> - The values that you have entered are valid and of the same data type as defined for each property.
>
> For details, refer to the Tag Properties *(on page 2931)* and Table Properties *(on page 2932)* topics.

3. For each tag, enter or modify values in the columns in the **ETLODBCTagConfig** worksheet. A value is required in the red-colored columns. For a list of tag properties that you can provide, refer to Tag Properties for ODBC Data Extraction *(on page 2931)*.
4. Select the **ETLODBCTableConfig** worksheet.
5. For each tag that you specified in the **ETLODBCTagConfig** worksheet, specify the table properties *(on page 2932)*. A value is required in the red-colored columns.
6. Save the file.
7. To create a tag configuration file:

    a. Select the **ETLODBCTagConfig** worksheet.

    b. Select **Developer > Export**.
    The **Export XML** window appears, asking you to choose whether you want to export the tag configuration or the table configuration.



c. Select **Taglist_Map**, and then select **OK**.

d. Provide a file name and location.

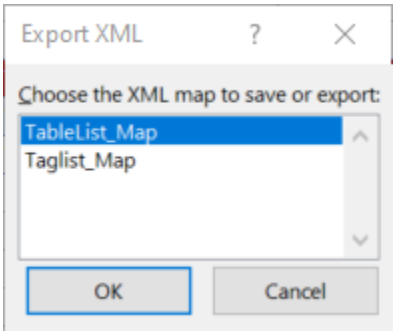
The tag configuration file is created with the list of tags whose data you want to extract.

8. To create a table configuration file:

a. Select the **ETLODBCTableConfig** worksheet.

b. Select **Developer > Export**.

The **Export XML** window appears, asking you to choose whether you want to export the tag configuration or the table configuration.




c. Select **TableList_Map**, and then select **OK**.

d. Provide a file name and location.

The table configuration file is created with the list of tables from which you want to extract data.

Configure the Historian ETL ODBC Extract settings *(on page 2933)*, providing the paths to the two .xml files you have created.

## Specify Tags and Tables for ODBC Data Extraction Using a Blank Spreadsheet

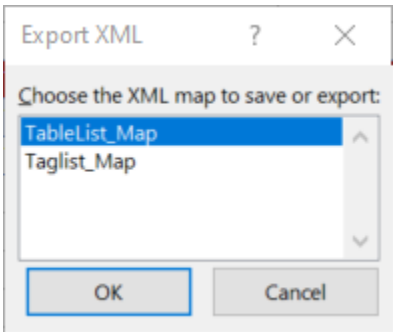
Before you extract data, you must specify the tags and tables whose data you want to extract.

This topic describes how to specify the tags using a new spreadsheet. Alternatively, you can specify the tags using a template spreadsheet *(on page 2926)* or by creating a configuration file manually *(on page 2924)*.

1. Create a Microsoft Excel file, and create the following worksheets:
    - **ETLODBCTagConfig**
    - **ETLODBCTableConfig**
2. In the **ETLODBCTagConfig** worksheet, enter column names matching the names of the tag properties *(on page 2931)*. Similarly, in the **ETLODBCTableConfig** worksheet, enter column names matching the names of the table properties *(on page 2932)*.
3. Import the tag configuration schema:
    a. In the **ETLODBCTagConfig** worksheet, select **Developer > Source > XML Maps > Add**.
    b. Select the `ETLODBCTagConfigSchema` file located in the `Historian ETL ODBC Extract` folder.
    c. Map each column name with each entry under **Taglist** in the **XML Source** section. To do so, select each column name, and then double-click the corresponding property under **Taglist**. Each property under **Taglist** changes to bold formatting, indicating that it is mapped to the corresponding column.
4. For each tag, enter or modify values in the columns.
5. Import the table configuration schema:
    a. In the **ETLODBCTableConfig** worksheet, select **Developer > Source > XML Maps > Add**.
    b. Select the `ETLODBCTableConfigSchema` file located in the `Historian ETL ODBC Extract` folder.
    c. Map each column name with each entry under **TableList** in the **XML Source** section. To do so, select each column name, and then double-click the corresponding property under **TableList**.
    Each property under **TableList** changes to bold formatting, indicating that it is mapped to the corresponding column.
6. For each table, enter or modify values in the columns.
7. In both the worksheets, select **Design > Properties**, select the **Validate data against schema for import and export** check box, and then select **OK**.
8. Save the file.
9. Select **Developer > Export**.
10. Create a tag configuration file:

a. Select the **ETLODBCTagConfig** worksheet.

b. Select **Developer > Export**.
The **Export XML** window appears, asking you to choose whether you want to export the tag configuration or the table configuration.




c. Select **Taglist_Map**, and then select **OK**.

d. Provide a file name and location.

The tag configuration file is created with the list of tags whose data you want to extract.
If an error occurs, stating that the data is not valid according to the schema, verify that:

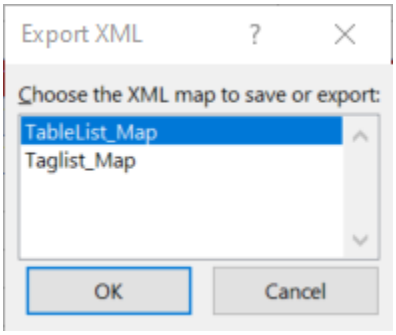
- You have provided values in all the red-colored columns.
- Values in the Name column in the **ETLODBCTagConfig** worksheet are unique.
- There are no blank rows.
- The values that you have entered are valid and of the same data type as defined for each property.

For information, refer to Tag Properties *(on page 2931)*.

11. Create a table configuration file:

a. Select the **ETLODBCTableConfig** worksheet.

b. Select **Developer > Export**.
The **Export XML** window appears, asking you to choose whether you want to export the tag configuration or the table configuration.




c. Select **TableList_Map**, and then select **OK**.

d. Provide a file name and location.

The table configuration file is created with the list of tables from which you want to extract data.

If an error occurs, stating that the data is not valid according to the schema, verify that:

- You have provided values in all the red-colored columns.
- There are no blank rows.
- The values that you have entered are valid and of the same data type as defined for each property.

For information, refer to Table Properties *(on page 2932)*.

Configure the Historian ETL ODBC Extract settings *(on page 2933)*, providing the paths to the two .xml files you have created.

## Tag Properties for ODBC Data Extraction

This topic provides a list of tag properties that you can define for each tag that you want to extract from an ODBC data source.

| Column Name | Data Type | Description |
|---|---|---|
| Name | String | Enter the name of the tag. A value is required and must be unique. |
| LocalName | String | Enter the local name of the tag. A value is required. |
| RemoteName | String | Enter the remote name of the tag. This will be used to name the tag after it is imported into Historian. A value is required. |
| DataType | String | Select the data type of the tag. A value is required. |

| Column Name | Data Type | Description |
|---|---|---|
| TableName | String | Enter the table that contains the tag data you want to import. A value is required and must match the name of a table in the list of table names you provide. Otherwise, an error occurs. |

## Table Properties for an ODBC Data Source

This topic provides a list of table properties that you can define for each tag that you want to extract from an ODBC data source.

| Column Name | Data Type | Description |
|---|---|---|
| TableName | String | Enter the name of the table. A value is required and must match the name of a table you specify in the list of tags. |
| TagNameColumn | String | Enter the name of the tag whose data you want to extract from the table. A value is required and must match the name of a tag you specify in the list of tags. |
| TimestampColumn | String | Enter the name of the column in the database table from which you want to extract the timestamp values. |
| ValueColumn | String | Enter the name of the column in the database table from which you want to extract the tag values. |
| QualityColumn | String | Enter the name of the column in the database table from which you want to extract the quality values. |
| ihOPCBad | String | Enter the values or range of values whose quality must be considered bad. For example, if a tag value is considered bad when the value is between 0 and 50, enter `<ihOPCBad>0-50</ihOPCBad>`. You can also enter words and values separated by commas (for example, `<ihOPCBad>bad, uneven</ihOPCBad>`).<br><br>A value is required. |
| ihOPCGood | String | Enter the values or range of values whose quality must be considered good. For example, if a tag value is considered good when the value is between 51 and 100, enter `<ihOPCGood>51-100</ihOPCGood>`. You can also enter words and val- |

| Column Name | Data Type | Description |
|---|---|---|
| | | ues separated by commas (for example, `<ihOPCGood>good, better, excellent</ihOPCGood>`).<br><br>A value is required. |
| ihOPCUncertain | String | Enter the values or range of values whose quality must be considered uncertain. For example, if a tag value is considered uncertain when the value is 101 or 102, enter `<ihOPCUncertain>101,102</ihOPCUncertain>`. You can also enter a range of values. |
| ihOPCNA | String | Enter the values or range of values whose quality must be considered not applicable. For example, if a tag value is considered not applicable when the value is 103, 104, or 105, enter `<ihOPCNA>103,104,105</ihOPCNA>`. You can also enter a range of values. |
| Condition | String | Enter the condition that you want to use to filter data. For example, if you enter Historian = 1, you can use this condition in a WHERE clause when querying data. |

## Configure the ODBC Data Extraction Settings

Specify the tags whose data you want to extract from an ODBC data source. You can do so by creating a configuration file manually *(on page 2924)*, using a template *(on page 2926)*, or using a blank spreadsheet *(on page 2928)*.

1. Run the `HistorianETLODBCExtractConfigTool_x64` file (for Windows 32 bit) or the `HistorianETLODBCExtractConfigTool_x86` file (for Windows 64 bit) located in the `<installation drive>\Program Files\GE Digital\Historian ETL ODBC Extract` folder.

   > **Tip:**
   > You can also enter ETL ODBC Extract in Windows Run.

   The **Historian ETL ODBC Extract Settings** window appears, displaying the **Basic Configuration** section.
2. If the configuration details are stored in a file, select **Import Config** to import the settings, and skip to step 6.

3. Provide values as specified in the following table.

| Field | Description | Default Value |
|---|---|---|
| **Data Source Name (DSN)** | Enter the name of the data source from which you want to extract data. It is the saved collection of settings required to connect to the ODBC data source.<br><br>You can use only a 32-bit or a 64-bit DSN depending on whether your application is 32-bit or 64-bit.<br><br>If you do not have a DSN, select **ODBC DataSource**, and create a system DSN. | |
| **Username** | Enter the username of the user to connect to ODBC data source. A value is required only if using SQL authentication. | |
| **Password** | Enter the password of the user to connect to ODBC data source. A value is required only if using SQL authentication. | |
| **Unit ID** | Enter the unit ID of the machine from which you want to transfer data. | |
| **Min # of Files to Compress** | Enter the number of files that must be compressed into a single .zip file. | 6 (that is, a .zip file is created for every six text files) |
| **Run Interval (Seconds)** | Enter the interval, in seconds, at which the Historian ODBC ETL Extract tool will extract data. You must enter a value greater than or equal to 60. | 150 (that is, a text file is created for data that is extracted in 150 seconds) |
| **Delay Interval** | The duration, in seconds, by which the data retrieval time will be reduced. For example, if data will be retrieved for 10 minutes, and if you enter 60 in this field, data for the last 60 seconds will not be retrieved in that batch; it will be retrieved in the next batch. This will ensure the re- | |

| Field | Description | Default Value |
|---|---|---|
| | trieval of any dynamic records that were updated in that duration. | |

4. Select **File Configuration**, and then provide values as specified in the following table.

| Field | Description | Default Value |
|---|---|---|
| **Data Extract Path** | Enter the path to the folder in which the text files containing the extracted data must be stored. | `<installation drive>\Program Files\GE Digital\Historian ETL ODBC Extract\Hist Files` |
| **Zip Extract Path** | Enter the path to the folder in which the compressed files must be stored. | `<installation drive>\Program Files\GE Digital\Historian ETL ODBC Extract\Zip Files` |
| **Tag Configuration File** | Enter the path to the tag configuration file *(on page 2924)* that you have created. | `<installation drive>\Program Files\GE Digital\Historian ETL ODBC Extract\OSM_OSMName.xml` |
| **Table Configuration File** | Enter the path to the table configuration file *(on page 2924)* that you have created. | `<installation drive>\Program Files\GE Digital\Historian ETL ODBC Extract\TableConfig.xml` |
| **State File** | Enter the path to the file that the Historian ODBC ETL Extract tool will create to store the timestamp of the last successful export. This timestamp is used to identify the start time for next iteration of extraction. This ensures that there is no loss of data during extraction.<br><br>For example, suppose the current time is 11am, and data has been extracted only till 9am. The state file contains the timestamp for 9am. | `<installation drive>\Program Files\GE Digital\Historian ETL ODBC Extract\State.xml` |

| Field | Description | Default Value |
|---|---|---|
| | Therefore, when data extraction is resumed, it is extracted from 9am.<br><br>The state file is created after you apply the Historian ODBC ETL Extract settings. It is updated each time .zip files are transferred to the destination Historian server or when the Historian ODBC ETL Extract tool is stopped. For a sample state file, refer to Example of a State File *(on page 2968)*. | |
| **Regen File** | Enter the path to the regeneration file. You can use this field to extract data.<br><br>If the **Generate Sample Regen File** field is set to **True**, a sample regeneration file will be created after you apply the settings. You can modify this file as needed, and specify the path of the same file in the **Regen File** field. For a sample regeneration file, refer to Example of a Regeneration File *(on page 2966)*. | `<installation drive>\Program Files\GE Digital\Historian ETL ODBC Extract\Regen.xml` |
| **Sample Regen File** | Enter the path to the sample regeneration file that will be created if the **Generate Sample Regen File** field is set to **True**. You can modify this file as needed. | `<installation drive>\Program Files\GE Digital\Historian ETL ODBC Extract\Sample_Regen.xml` |

5. Select **Advanced Configuration**, and then provide values as specified in the following table.

| Field | Description | Default Value |
|---|---|---|
| **Log Level** | Select the log level to indicate the amount of information to be logged. The following options are available: | Info |

| Field | Description | Default Value |
|---|---|---|
| | ◦ Info<br>◦ Error<br>◦ Debug | |
| **Extract Type** | Specifies whether you want to extract only the current data or historical data or both. However, we recommend that you extract only current data. | Current Data |
| **Generate Sample Regen File** | Select **True** if you want to generate the sample regeneration file. You can then modify this file as needed. | False |
| **Query Timeout** | The time, in seconds, after which a query to the database times out. | 30 |
| **Save Limit** | Enter the number of files to be exported after which the `State.xml` file must be updated. | 20 |
| **Catch Up Interval (Seconds)** | Enter the catch up interval, in seconds, used to size files when catching up to the current time. | 10 |
| **Catch Up Time Limit (Hours)** | Enter the maximum time, in hours, to go back when catching up after a restart.<br><br>You can use this field to extract data. | 168 |

6. Select **Save**.
   The changes to the settings are applied and saved in the `HistorianETLODBCExtract.exe.config` file.

Start the data extraction *(on page 2947)*.

## Start the ODBC Data Extraction

Configure the Historian ETL ODBC Extract settings *(on page 2933)*.

1. Run the `Historian ETL ODBC Extract Configuration` file located in the `Historian ETL ODBC Extract` folder.
   The **Historian ETL ODBC Extract Configuration** window appears.
2. Select **Start Service**.

Data extraction from the ODBC data source begins.

> **Tip:**
> If the tool does not start as expected, access the logs using Windows Event Viewer.

Transfer data using BITS *(on page 2957)*, FTP *(on page 2959)*, or any other file-sharing application.

# About Extracting Data from Proficy Historian

The Historian ETL Extract tool extracts data as follows:

1. Extracts data related to tags into text files, which are named in the following format: `YYYYDDMMHHRR_<onsite Historian computer name>.txt`. These files are stored in the following folder: `<Historian ETL installation location>/Historian ETL Extract/HistFiles`.

   > **Note:**
   > Data related to alarms and events is stored in .lax files. You can choose not to extract data related to alarms and events.

2. After a specified number of files are extracted (by default, 6), the files are compressed into a .zip file, which is named in the following format: `YYYYDDMMHHRR_<onsite Historian computer name>.zip`. These files are stored in the following folder: `<Historian ETL installation location>/Historian ETL Extract/ZipFiles`.
3. Deletes the text files in the `<Historian ETL installation location>/Historian ETL Extract/HistFiles` folder after they are compressed.

## Specify Tags for Proficy Historian Data Extraction Manually

Before you extract data, you must specify the tags whose data you want to extract.

This topic describes how to specify the tags manually by creating a configuration file. Alternatively, you can specify the tags using a template spreadsheet *(on page 2939)* or a blank spreadsheet *(on page 2940)*.

Create an .xml file to include the tag properties *(on page 2942)* for all the tags whose data you want to extract.
Tag properties for a tag named Pressure used in Proficy Historian:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Taglist>
    <Tag Name="Pressure">
        <LocalName>ValvePressure</LocalName>
        <RemoteName>ValvePressure</RemoteName>
        <Compression>1</Compression>
        <DeadbandRange>2.5</DeadbandRange>
        <DeadbandTimeout>2</DeadbandTimeout>
        <RequireRescale>1</RequireRescale>
        <HiEng>10</HiEng>
        <LowEng>8</LowEng>
        <HiScale>10</HiScale>
        <LowScale>8</LowScale>
    </Tag>
</Taglist>
```

Configure the Historian ETL Extract settings *(on page 2942)*.

## Specify Tags for Proficy Historian Data Extraction Using a Template

Before you extract data, you must specify the tags whose data you want to extract.

This topic describes how to specify the tags using a template spreadsheet, which is provided with the Historian ETL package. Alternatively, you can specify the tags using a new spreadsheet *(on page 2940)* or by creating a configuration file manually *(on page 2938)*.

1. Access the `ProficyHistTagConfigGenerateExcel.xlsx` file located in the `Historian ETL Extract` folder.
2. **Optional:** If you want to modify an existing tag configuration file, you can import the data from that .xml file by selecting **Developer > Import**.

   **Tip:**
   If the **Developer** tab is not available, right-click the menu bar of the spreadsheet, select **Customize the Ribbon**, and then select the **Developer** check box.

   Data from the .xml file is imported into the spreadsheet.

   **Note:**
   If an error occurs, stating that the data is not valid according to the schema, verify that:

- You have provided values in all the red-colored columns.
- There are no blank rows.
- The values that you have entered are valid and of the same data type as defined for each property.

For details, refer to the Tag Properties *(on page 2942)* topic.

3. For each tag, enter or modify values in the columns for the tag properties *(on page 2942)*. A value is required in the red-colored columns.
4. Save the file.
5. Select **Developer > Export**.

**Tip:**
If the **Developer** tab is not available, right-select the menu bar of the spreadsheet, select **Customize the Ribbon**, and then select the **Developer** check box.

6. Enter a name and location for the .xml file.
   The tag configuration file is created with the list of tags and their properties to be extracted.

**Note:**
If an error occurs, stating that the data is not valid according to the schema, verify that:
- You have provided values in all the red-colored columns.
- Values in the Name column are unique.
- There are no blank rows.
- The values that you have entered are valid and of the same data type as defined for each property.

For details, refer to the Tag Properties *(on page 2942)* topic.

Configure the Historian ETL Extract settings *(on page 2942)*.

## Specify Tags for Proficy Historian Data Extraction Using a Blank Spreadsheet

Before you extract data, you must specify the tags whose data you want to extract.

This topic describes how to specify the tags using a new spreadsheet. Alternatively, you can specify the tags using a template spreadsheet *(on page 2939)* or by creating a configuration file manually *(on page 2938)*.

1. Create a Microsoft Excel file.
2. Enter column names matching the names of the tag properties *(on page 2942)*.
3. **Optional:** If you want to modify an existing tag configuration file, you can import the data from that .xml file by selecting **Developer > Import**.

> **Tip:**
> If the **Developer** tab is not available, right-select the menu bar of the spreadsheet, select **Customize the Ribbon**, and then select the **Developer** check box.

> **Note:**
> If a message appears, stating that the .xml file is not linked to schema, ignore the message.

4. Select **Developer > Source > XML Maps > Add**.
5. Select the `ProficyHistTagConfigSchema` file located in the `Historian ETL Extract` folder.
6. Map each column name with each entry under **Taglist** in the **XML Source** section. To do so, select each column name, and then double-click the corresponding property under **Taglist**.
   Each property under **Taglist** changes to bold formatting, indicating that it is mapped to the corresponding column.
7. For each tag, enter or modify values in the columns.
8. Select **Design > Properties**, select the **Validate data against schema for import and export** check box, and then select **OK**.
9. Save the file.
10. Select **Developer > Export**.
11. Enter a name and location for the .xml file.
    The tag configuration file is created with the list of tags and their properties to be extracted.

> **Note:**
> If an error occurs, stating that the data is not valid according to the schema, verify that:
> - You have provided values in all the red-colored columns.
> - Values in the Name column are unique.
> - There are no blank rows.
> - The values that you have entered are valid and of the same data type as defined for each property.
>
> For details, refer to the Tag Properties *(on page 2942)* topic.

Configure the Historian ETL Extract settings *(on page 2942)*.

## Tag Properties for Proficy Historian Data Extraction

This topic provides a list of tag properties that you can define for each tag that you want to extract from Proficy Historian.

| Column Name | Data Type | Description |
|---|---|---|
| Name | String | Enter the name of the tag. A value is required and must be unique. |
| LocalName | String | Enter the local name of the tag. A value is required. |
| RemoteName | String | Enter the remote name of the tag. This will be used to name the tag after it is imported into Historian. A value is required. |
| Compression | Boolean | • Enter 1 if you want to enable collector compression.<br>• Enter 0 if you do not want to enable collector compression. |
| DeadbandRange | Numeric | Enter the collector deadband range (only absolute values, not in percentage). |
| DeadbandTimeout | Numeric | Enter the collector compression timeout for the tag. |
| RequireRescale | Boolean | • Enter 1 if you want to enable scaling, which converts an input data point to an engineering units value.<br>• Enter 0 if you do not want to enable scaling. |
| HiEng | Numeric | Enter the upper limit in engineering units for the tag. |
| LowEng | Numeric | Enter the lower limit in engineering units for the tag. |
| HiScale | Numeric | Enter the upper limit for the tag value if scaling is enabled. |
| LowScale | Numeric | Enter the lower limit for the tag value if scaling is enabled. |

## Configure the Proficy Historian Data Extraction Settings

Specify the tags whose data you want to extract from Proficy Historian. You can do so by creating a configuration file manually *(on page 2938)*, using a template *(on page 2939)*, or using a blank spreadsheet *(on page 2940)*.

This topic describes how to configure the Historian ETL Extract tool to modify the default folders to store the extracted data, to specify whether data related to alarms and events must be extracted, and so on.

**Note:**
These settings are saved in the `HistorianETLExtract.exe.config` file.

1. Run the `Historian ETL Extract Configuration` file located in the `Historian ETL Extract` folder.

   **Tip:**
   You can also enter ETL Historian Extract in Windows Run.

   The **Historian ETL Extract Configuration** window appears, displaying the **Basic Configuration** section.
2. If the configuration details are stored in a file, select **Import Config** to import the settings. Otherwise, skip to the next step.
3. Provide values as specified in the following table.

| Field | Description | Default Value |
|---|---|---|
| **Historian Server** | Enter the host name or IP address of the onsite Historian machine. If you leave it blank, the local host name is considered. | |
| **Historian User** | Enter the ID of the user to connect to the Historian server on the onsite Historian machine. A value is required only if security is enabled for the Historian server. | |
| **Historian Password** | Enter the password of the user to connect to the Historian server on the onsite Historian machine. A value is required only if security is enabled for the Historian server. | |
| **Unit ID** | Enter the unit ID of the machine from which you want to transfer data. | OSMName |
| **Run Interval** | Enter the interval, in seconds, at which the Historian ETL Extract tool will extract data. You must enter a value greater than or equal to 60. | 300 (that is, a text file is created for data that is extracted in 300 seconds) |

| Field | Description | Default Value |
|---|---|---|
| **Min # of Files to Compress** | Enter the number of files that must be compressed into a single .zip file. | 6 (that is, a .zip file is created for every six text files) |
| **Alarms & Events** | Select **False** if you do not want to extract data related to alarms and events. | True |

4. Select **Files**, and then provide values as specified in the following table.

| Field | Description | Default Value |
|---|---|---|
| **Historian Export Path** | Enter the path to the folder in which the text files containing the extracted data must be stored. | `<Installation folder of Historian ETL>/Historian ETL Extract/HistFiles` |
| **Tag Configuration File** | Enter the path to the tag configuration file that you have created. | `<Installation folder of Historian ETL>/Historian ETL Extract/OSM_OSMName.xml` |
| **Zip Export Path** | Enter the path to the folder in which the compressed files must be stored. | `<Installation folder of Historian ETL>/Historian ETL Extract/ZipFiles` |
| **State File** | Enter the path to the file that the Historian ETL Extract tool will create to store the timestamp of the last successful export. This timestamp is used to identify the start time for next iteration of extraction. This ensures that there is no loss of data during extraction.<br><br>For example, suppose the current time is 11am, and data has been extracted only till 9am. The state file contains the timestamp for 9am. Therefore, when data extraction is resumed, it is extracted from 9am. | `<Installation folder of Historian ETL>/Historian ETL Extract/State.xml` |

| Field | Description | Default Value |
|---|---|---|
| | The state file is created after you apply the Historian ETL Extract settings. It is updated each time .zip files are transferred to the destination Historian server or when the Historian ETL Extract tool is stopped. For a sample state file, refer to Example of a State File *(on page 2968)*. | |
| **Regen File** | Enter the path to the regeneration file. You can use this field to extract historical data *(on page 2946)*.<br><br>If the **Generate Sample Regen File** field is set to **True**, a sample regeneration file will be created after you apply the settings. You can modify this file as needed, and specify the path of the same file in the **Regen File** field. For a sample regeneration file, refer to Example of a Regeneration File *(on page 2966)*. | `<installation folder of Historian ETL>\Regen.xml` |

5. Select **Advanced Configuration**, and then provide values as specified in the following table.

| Field | Description | Default Value |
|---|---|---|
| **Log Level** | Select the log level to indicate the amount of information to be logged. The following options are available:<br>◦ Info<br>◦ Error<br>◦ Debug | Info |
| **Catch Up Interval** | Enter the catch up interval, in seconds, used to size files when catching up to the current time. | 10 |

| Field | Description | Default Value |
|---|---|---|
| **Save Limit** | Enter the number of files to be exported after which the `State.xml` file must be updated. | 20 |
| **Catch Up Time Limit** | Enter the maximum time, in hours, to go back when catching up after a restart.<br><br>You can use this field to extract historical data *(on page 2946)*. | 168 |
| **Delay Interval** | The duration, in seconds, by which the data retrieval time will be reduced. For example, if data will be retrieved for 10 minutes, and if you enter 60 in this field, data for the last 60 seconds will not be retrieved in that batch; it will be retrieved in the next batch. This will ensure the retrieval of any dynamic records that were updated in that duration. | 60 |
| **Sample Regen File** | Enter the path to the sample regeneration file that will be created if the **Generate Sample Regen File** field is set to **True**. You can modify this file as needed. | `<installation folder of Historian ETL>\Sample_Regen.xml` |
| **Generate Sample Regen File** | Select **True** if you want to generate the sample regeneration file. You can then modify this file as needed. | False |

6. Select **Save**.
   The changes to the settings are applied and saved in the `HistorianETLExtract.exe.config` file.

Start the data extraction *(on page 2947)*.

## Extract Historical Data from Proficy Historian

By default, the ETL tools extract current data, starting from the time you have configured the tags for data extraction. This topic describes how to extract historical data using the ETL tools.

1. Create a regeneration file, specifying the start time, end time, and interval for which you want to capture the historical data. For a sample regeneration file, refer to Example of a Regeneration File *(on page 2966)*.
2. Run the `Historian ETL Extract Configuration` file located in the `Historian ETL Extract` folder.

   > **Tip:**
   > You can also enter ETL Historian Extract in the Windows Start menu.

   The **Historian ETL Extract Configuration** window appears, displaying the **Basic Configuration** section.
3. Select **Advanced Configuration**, and then provide values as specified in the following table.

| Field | Description | Default Value |
|---|---|---|
| **Regen File** | Enter the path to the regeneration file that you have created. | `<installation drive>\Program Files\GE Digital\Historian ETL Extract\Regen.xml` |
| **Catch Up Time Limit** | Enter the duration, in hours, for which you want to extract historical data. For example, if you want to extract data for the past one day, enter 24. | 168 |
| **Temp Regen File** | Enter the path to the temporary regeneration file. | `<installation drive>\Program Files\GE Digital\Historian ETL Extract\Temp_Regen.xml` |
| **Generate Temp Regen File** | Select **True**. | False |

4. As needed, provide values in the remaining fields *(on page 2942)*

Start the data extraction *(on page 2947)*.

## Start the Proficy Historian Data Extraction

Configure the Historian ETL Extract settings *(on page 2942)*.

1. Run the `Historian ETL Extract Configuration` file located in the `Historian ETL Extract` folder.
   The **Historian ETL Extract Configuration** window appears.
2. Select **Start Service**.
   Data extraction from the Proficy Historian server begins.

> **Tip:**
> If the tool does not start as expected, access the logs using Windows Event Viewer.

Transfer data using BITS *(on page 2957)*, FTP *(on page 2959)*, or any other file-sharing application.

# Extracting Data from PI Historian

The Historian ETL PI Extract tool extracts data as follows:

1. Extracts data related to tags into text files, which are named in the following format: `YYYYDDMMHHRR_<onsite Historian computer name>.txt`. These files are stored in the following folder: `<Historian ETL installation location>/Historian ETL PI Extract/HistFiles`.
2. After a specified number of files are extracted (by default, 6), the files are compressed into a .zip file, which is named in the following format: `YYYYDDMMHHRR_<onsite Historian computer name>.zip`. These files are stored in the following folder: `<Historian ETL installation location>/Historian ETL PI Extract/ZipFiles`.

## Specify Tags for PI Historian Data Extraction Manually

Before you extract data, you must specify the tags whose data you want to extract.

This topic describes how to specify the tags manually by creating a configuration file. Alternatively, you can specify the tags using a template spreadsheet *(on page 2949)* or a blank spreadsheet *(on page 2950)*.

Create an .xml file to include the tag properties *(on page 2951)* for all the tags whose data you want to extract.
Tag properties for a tag named Pressure used in an ODBC data source:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Taglist>
    <Tag Name="Pressure">
        <LocalName>ValvePressure</LocalName>
```

```
        <RemoteName>ValvePressure</RemoteName>
    </Tag>
</Taglist>
```

Configure the Historian ETL PI Extract settings *(on page 2952)*.

## Specify Tags for PI Historian Data Extraction Using a Template

Before you extract data, you must specify the tags whose data you want to extract.

This topic describes how to specify the tags using a template spreadsheet, which is provided with the Historian ETL package. Alternatively, you can specify the tags using a new spreadsheet *(on page 2950)* or by creating a configuration file manually *(on page 2948)*.

1. Access the `PIHistTagConfigGenerateExcel.xlsx` file located in the `Historian PI ETL Extract` folder.
2. **Optional:** If you want to modify an existing tag configuration file, you can import the data from that .xml file by selecting **Developer > Import**.

   **Tip:**
   If the **Developer** tab is not available, right-click the menu bar of the spreadsheet, select **Customize the Ribbon**, and then select the **Developer** check box.

   Data from the .xml file is imported into the spreadsheet.

   **Note:**
   If an error occurs, stating that the data is not valid according to the schema, verify that:
   - You have provided values in all the red-colored columns.
   - There are no blank rows.
   - The values that you have entered are valid and of the same data type as defined for each property.

   For details, refer to the Tag Properties *(on page 2951)* topic.
3. For each tag, enter or modify values in the columns for the tag properties *(on page 2951)*. A value is required in the red-colored columns.
4. Save the file.
5. Select **Developer > Export**.

> **Tip:**
> If the **Developer** tab is not available, right-select the menu bar of the spreadsheet, select **Customize the Ribbon**, and then select the **Developer** check box.

6. Enter a name and location for the .xml file.
   The tag configuration file is created with the list of tags and their properties to be extracted.

> **Note:**
> If an error occurs, stating that the data is not valid according to the schema, verify that:
> - You have provided values in all the red-colored columns.
> - Values in the Name column are unique.
> - There are no blank rows.
> - The values that you have entered are valid and of the same data type as defined for each property.
>
> For details, refer to the Tag Properties *(on page 2951)* topic.

Configure the Historian ETL PI Extract settings *(on page 2952)*.

## Specify Tags for PI Historian Data Extraction Using a Blank Spreadsheet

Before you extract data, you must specify the tags whose data you want to extract.

This topic describes how to specify the tags using a new spreadsheet. Alternatively, you can specify the tags using a template spreadsheet *(on page 2949)* or by creating a configuration file manually *(on page 2948)*.

1. Create a Microsoft Excel file.
2. Enter column names matching the names of the tag properties *(on page 2951)*.
3. **Optional:** If you want to modify an existing tag configuration file, you can import the data from that .xml file by selecting **Developer > Import**.

> **Tip:**
> If the **Developer** tab is not available, right-select the menu bar of the spreadsheet, select **Customize the Ribbon**, and then select the **Developer** check box.

> **Note:**
> If a message appears, stating that the .xml file is not linked to schema, ignore the message.

4. Select **Developer > Source > XML Maps > Add**.
5. Select the `PIHistTagConfigSchema` file located in the `Historian ETL PI Extract` folder.
6. Map each column name with each entry under **Taglist** in the **XML Source** section. To do so, select each column name, and then double-click the corresponding property under **Taglist**.
   Each property under **Taglist** changes to bold formatting, indicating that it is mapped to the corresponding column.
7. For each tag, enter or modify values in the columns.
8. Select **Design > Properties**, select the **Validate data against schema for import and export** check box, and then select **OK**.
9. Save the file.
10. Select **Developer > Export**.
11. Enter a name and location for the .xml file.
    The tag configuration file is created with the list of tags and their properties to be extracted.

> **Note:**
> If an error occurs, stating that the data is not valid according to the schema, verify that:
> - You have provided values in all the red-colored columns.
> - Values in the Name column are unique.
> - There are no blank rows.
> - The values that you have entered are valid and of the same data type as defined for each property.
>
> For details, refer to the Tag Properties *(on page 2951)* topic.

Configure the Historian ETL PI Extract settings *(on page 2952)*.

## Tag Properties for PI Historian Data Extraction

This topic provides a list of tag properties that you can define for each tag that you want to extract from PI Historian.

| Column Name | Data Type | Description |
|---|---|---|
| Name | String | Enter the name of the tag. A value is required and must be unique. |
| LocalName | String | Enter the local name of the tag. A value is required. |
| RemoteName | String | Enter the remote name of the tag. This will be used to name the tag after it is imported into Historian. A value is required. |

## Configure the PI Historian Data Extraction Settings

Specify the tags whose data you want to extract from PI Historian. You can do so by creating a configuration file manually, using a template, or using a blank spreadsheet.

This topic describes how to configure the Historian ETL PI Extract tool to modify the default folders to store the extracted data.

**Note:**
These settings are saved in the `PIHistorianETLExtract.exe.config` file.

1. Run the `Historian ETL PI Extract Configuration` file located in the `Historian ETL PI Extract` folder.

   **Tip:**
   You can also enter ETL PI Extract in Windows Run.

   The **Historian ETL PI Extract Configuration** window appears, displaying the **Basic Configuration** section.
2. If the configuration details are stored in a file, select **Import Config** to import the settings. Otherwise, skip to the next step.
3. Provide values as specified in the following table.

| Field | Description | Default Value |
|---|---|---|
| **Historian Server** | Enter the host name or IP address of the PI Server machine. | |

| Field | Description | Default Value |
|---|---|---|
| **Historian User** | Enter the ID of the user to connect to the PI Server machine. A value is required only if security is enabled for the Historian server. | |
| **Historian Password** | Enter the password of the user to connect to the PI Server machine. A value is required only if security is enabled for the Historian server. | |
| **Unit ID** | Enter the unit ID of the machine from which you want to transfer data. | OSMName |
| **Run Interval** | Enter the interval, in seconds, at which the Historian ETL PI Extract tool will extract data. You must enter a value greater than or equal to 60. | 150 (that is, a text file is created for data that is extracted in 150 seconds) |
| **Min # of Files to Compress** | Enter the number of files that must be compressed into a single .zip file. | 6 (that is, a .zip file is created for every six text files) |

4. Select **Files**, and then provide values as specified in the following table.

| Field | Description | Default Value |
|---|---|---|
| **Historian Export Path** | Enter the path to the folder in which the text files containing the extracted data must be stored. | `<Installation folder of Historian ETL>/Historian ETL PI Extract/HistFiles` |
| **Tag Configuration File** | Enter the path to the tag configuration file that you have created. | `<Installation folder of Historian ETL>/Historian ETL PI Extract/OSM_OSMName.xml` |
| **Zip Export Path** | Enter the path to the folder in which the compressed files must be stored. | `<Installation folder of Historian ETL>/Historian ETL PI Extract/ZipFiles` |
| **State File** | Enter the path to the file that the Historian ETL PI Extract tool will create to store the timestamp of the last successful export. This timestamp is | `<Installation folder of Historian ETL>/Historian ETL PI Extract/State.xml` |

| Field | Description | Default Value |
|---|---|---|
| | used to identify the start time for next iteration of extraction. This ensures that there is no loss of data during extraction.<br><br>For example, suppose the current time is 11am, and data has been extracted only till 9am. The state file contains the timestamp for 9am. Therefore, when data extraction is resumed, it is extracted from 9am.<br><br>The state file is created after you apply the settings. It is updated each time .zip files are transferred to the destination Historian server or when the Historian PI ETL Extract tool is stopped. For a sample state file, refer to Example of a State File *(on page 2968)*. | |

5. Select **Advanced Configuration**, and then provide values as specified in the following table.

| Field | Description | Default Value |
|---|---|---|
| **Log Level** | Select the log level to indicate the amount of information to be logged. The following options are available:<br>◦ Info<br>◦ Error<br>◦ Debug | Info |
| **Catch Up Interval** | Enter the catch up interval, in seconds, used to size files when catching up to the current time. | 10 |
| **Regen File** | Enter the path to the regeneration file. You can use this field to extract historical data *(on page 2946)*. | `<installation folder of Historian ETL>\Regen.xml` |

| Field | Description | Default Value |
|---|---|---|
| | If the **Generate Sample Regen File** field is set to **True**, a sample regeneration file will be created after you apply the settings. You can modify this file as needed, and specify the path of the same file in the **Regen File** field. For a sample regeneration file, refer to Example of a Regeneration File *(on page 2966)*. | |
| **Save Limit** | Enter the number of files to be exported after which the `State.xml` file must be updated. | 20 |
| **Catch Up Time Limit** | Enter the maximum time, in hours, to go back when catching up after a restart.<br><br>You can use this field to extract historical data *(on page 2946)*. | 168 |
| **Delay Interval** | The duration, in seconds, by which the data retrieval time will be reduced. For example, if data will be retrieved for 10 minutes, and if you enter 60 in this field, data for the last 60 seconds will not be retrieved in that batch; it will be retrieved in the next batch. This will ensure the retrieval of any dynamic records that were updated in that duration. | 60 |
| **Sample Regen File** | Enter the path to the sample regeneration file that will be created if the **Generate Sample Regen File** field is set to **True**. You can modify this file as needed. | `<installation folder of Historian ETL>\Sample_Regen.xml` |
| **Generate Sample Regen File** | Select **True** if you want to generate the sample regeneration file. You can then modify this file as needed. | False |

6. Select **Save**.
   The changes to the settings are applied and saved in the `PIHistorianETLExtract.exe.config` file.

Start the data extraction *(on page 2947)*.

## Extract Historical Data from PI Historian

By default, the ETL tools extract current data, starting from the time you have configured the tags for data extraction. This topic describes how to extract historical data using the ETL tools.

1. Create a regeneration file, specifying the start time, end time, and interval for which you want to capture the historical data. For a sample regeneration file, refer to Example of a Regeneration File *(on page 2966)*.
2. Run the `Historian ETL PI Extract Configuration` file located in the `Historian ETL PI Extract` folder.

   > **Tip:**
   > You can also enter ETL PI Extract in the Windows Start menu.

   The **Historian ETL PI Extract Configuration** window appears, displaying the **Basic Configuration** section.
3. Select **Advanced Configuration**, and then provide values as specified in the following table.

| Field | Description | Default Value |
|---|---|---|
| **Regen File** | Enter the path to the regeneration file that you have created. | `<`*`installation drive`*`>\Program Files\GE Digital\Historian ETL Extract\Regen.xml` |
| **Catch Up Time Limit** | Enter the duration, in hours, for which you want to extract historical data. For example, if you want to extract data for the past one day, enter 24. | 168 |
| **Temp Regen File** | Enter the path to the temporary regeneration file. | `<`*`installation drive`*`>\Program Files\GE Digital\Historian ETL PI Extract\Temp_Regen.xml` |
| **Generate Temp Regen File** | Select **True**. | False |

4. As needed, provide values in the remaining fields *(on page 2952)*.

Start the data extraction *(on page 2957)*.

## Start the PI Historian Data Extraction

Configure the Historian ETL PI Extract settings *(on page 2952)*.

1. Run the `Historian ETL PI Extract Configuration` file located in the `Historian ETL PI Extract` folder.
   The **Historian ETL PI Extract Configuration** window appears.
2. Select **Start Service**.
   Data extraction from PI Server begins.

> **Tip:**
> If the tool does not start as expected, access the logs using Windows Event Viewer.

# About Transferring Data Using Background Intelligent Transfer Service (BITS)

After you extract data, you must transfer it to the destination machine. To do so, you can use BITS, FTP, or any other file-sharing application.

To transfer data using BITS, perform the following steps:

1. Install the BITS IIS server extension.
2. Configure the BITS settings *(on page 2957)*.
3. Transfer data to the destination machine *(on page 2958)*.

## Configure BITS

Install the BITS IIS server extension.

1. Using IIS Management Console, navigate to the default website node, select **Add Virtual Directory**, and create a virtual directory named `MD_BITS`.
2. In the `MD_BITS` folder, create a folder named `OSMUploads`.
3. Enable the BITS IIS server extension:
   a. Navigate to the virtual directory in IIS Manager.
   b. From the list of features in the virtual directory, double-click **BITS Uploads**.
   c. Select the **Allow clients to upload files** check box, and then select **Apply**.
4. Change the port number of the default website in IIS. By default, the port number is 80.

a. In IIS Manager, in the **Connections** section, under the computer name > **Sites**, select **Default Web Site**.
b. In the **Actions** pane, under **Edit Site**, select **Bindings**.
c. In the **Site Bindings** window, select **http > Edit**.
d. In the **Edit Site Bindings** window, in the **Port** field, enter the new port number (for example, 6150), and then select **OK**.
e. In the **Site Bindings** window, select **Close**.
f. In the **Actions** section, under **Manage Web Site**, select **Stop**, and then select **Start**.

Verify the data transfer settings *(on page 2958)*.

## Verify the Data Transfer Settings

This topic provides a list of tasks that you can perform to verify that the data transfer settings are correct.

- From you onsite Historian machine, using Internet Explorer, verify that you can access a web page created on the destination machine.
- Verify that the Historian ETL Load tool running on the destination machine is configured to watch the virtual folder in IIS for the incoming .zip files.

## Transfer Data using BITS

Configure BITS *(on page 2957)*.

You can transfer data using BITS by performing one of the following steps:

1. If you want to use the `OSM_LBW_Transfer.vbs` file to transfer files, perform the following steps:
    a. Verify that the `OSM_LBW_Transfer.vbs` file is configured to watch the folder in the destination machine in which the .zip files will be placed.
    b. Run the `OSM_LBW_Transfer.vbs` script by running the `OSM_LBW_Transfer.cmd` file. The script transfers files in the `<Historian ETL installation location>/Historian ETL Extract/ZipFiles` folder to the destination machine.
2. If you want to use the `OSMBitsDownload.vbs` file to download files, perform the following steps.
    a. Access the `OSMBitsDownload.vbs` file, and verify that the path to the files and folders specified in the file is correct.
    b. Run the `OSMBitsDownload.vbs` script by running the following command: `cscript OSMBitsDownload.vbs`.

The script uses the `DownloadFilesToOSM.txt` file to fetch the names of the files that must be transferred.

**Important:**

- Ensure that the `DownloadFilesToOSM.txt` file exists in the same location as the `OSMBitsDownload.vbs` file.
- In the `DownloadFilesToOSM.txt` file, enter the file names that you want to download using the `OSMBitsDownload.vbs` script. If you want to transfer all zip files, enter `*.zip`.

**Tip:**
To access the jobs created by BITS while transferring data, access the `BITSADMIN/LIST/ALLUSERS` folder. If the tool does not start as expected, access the logs using Windows Event Viewer.

Load the data into the destination Historian server *(on page 2961)*.

# About Transferring Data Using File Transfer Protocol (FTP)

After you extract data, you must transfer it to the destination Historian server. To do so, you can use BITS, FTP, or any other file-sharing application.

**Note:**
The minimum bandwidth required to transfer data using FTP is 2 KBps.

To transfer data using FTP, perform the following steps:

1. Install an FTP server on the destination Historian server.
2. Configure the FTP settings *(on page 2959)*.
3. Transfer data to the destination Historian server *(on page 2960)*.

## Configure FTP

Install FTP on the destination Historian server.

1. Access the `FTPFileTransfer` file located in the `<Installation folder of Historian ETL>/Historian ETL Transform` folder.
   The **FTP File Transfer Tool** window appears.
2. If the FTP configuration details are saved in a file, select **File > Load**, and then select the configuration file that contains the details. Otherwise, skip to the next step.
3. Provide values as specified in the following table.

   **Tip:**
   You can select **File > New** to create a new instance of the settings.

| Field | Description | Default Value |
|---|---|---|
| **FTP Server Address** | Enter the address of the FTP server. | ftp://127.0.0.1/ftpserver |
| **FTP UserID** | Enter the user ID to log in to the FTP server. | Blank |
| **FTP Password** | Enter the password to log in to the FTP server. | Blank |
| **Directory to Monitor** | Provide the folder from which files must be transferred. | `C:\` |
| **File Mask** | Enter the file mask for the FTP server. | *.zip |

4. **Optional:** Select **Test Connection** to test the FTP server connection.
5. Select **File > Save** to save the configuration details, which you can reuse.
6. Select **File > Detail Log** if you want a detailed logging, which will help you in troubleshooting.

The FTP server settings are configured.

Transfer data using FTP *(on page 2960)*.

## Transfer Data Using FTP

Configure FTP *(on page 2959)*.

1. Access the `FTPFileTransfer` file located in the `<Installation folder of Historian ETL>/Historian ETL Transform` folder.
   The **FTP File Transfer Tool** window appears.

2. Select **Start**.
   Data is transferred from the `ZipFiles` folder to the machine on which the destination Historian is installed.

Load the data into the destination Historian server *(on page 2961)*.

# About Loading Data

After the data is transferred to the machine on which the destination Historian is installed, you must load it into the destination Historian server using the Historian ETL Load tool. This tool loads data as follows:

1. Extracts the .zip files in the `<Historian ETL installation location>/Historian ETL Load/ZipImportFiles` folder and stores the text files in the `<Historian ETL installation location>/Historian ETL Load/ImportFiles` folder in the destination Historian server.
2. Loads the data into the destination Historian server.
3. Deletes the .zip files in the `<Historian ETL installation location>/Historian ETL Load/ZipImportFiles` folder, and imports the text files to the destination Historian server.
4. Deletes the text files in the `<Historian ETL installation location>/Historian ETL Load/ImportFiles` folder after importing them to the destination Historian server.

## Configure the Historian ETL Load tool

This topic describes how to configure the Historian ETL Load service to modify the default folders to store the extracted data, to specify whether data related to alarms and events must be transferred, and so on.

**Note:**
These settings are saved in the `HistorianETLLoad.exe.config` file.

1. Run the `Historian ETL Load Configuration` file located in the `Historian ETL Load` folder.

   **Tip:**
   You can also enter ETL Historian Load in Windows Run.

   The **Historian ETL Load Configuration** window appears, displaying the **Basic Configuration** section.
2. If the configuration details are stored in a file, select **Import Config** to import the settings. Otherwise, skip to the next step.

3. Provide values as specified in the following table.

| Field | Description | Default Value |
|---|---|---|
| **Historian Server** | Enter the host name or IP address of the destination Historian server. If you leave this field blank, the local host name is considered. | Blank |
| **Historian User** | Enter the ID of the user to connect to the destination Historian server. A value is required only if security is enabled for the destination Historian server. | Blank |
| **Historian Password** | Enter the password of the user to connect to the destination Historian server. A value is required only if security is enabled for the destination Historian server. | Blank |
| **IH Status Tagname** | Enter the name of the tag that is used to check the state of the destination Historian server. This tag must exist in Historian. | IHStatusTag |
| **Alarms & Events** | Select **False** if you do not want to load data related to alarms and events. | True |
| **Add Tag Automatically** | Select **True** to create a tag automatically in the destination Proficy Historian server if the tag name that you entered does not exist in the server. | False |

4. Select **Files**, and then provide values as specified in the following table.

| Field | Description | Default Value |
|---|---|---|
| **Historian File Path** | Enter the path to the folder in which the text files containing the extracted data are stored. | `<Installation folder of Historian ETL>/Historian ETL Load/ImportFiles` |
| **Zip File Path** | Enter the path to the folder in which the compressed files are available. | `<Installation folder of Historian ETL>/Historian ETL Load/ZipImportFiles` |
| **Error File Path** | Enter the path to the folder in which file that could not be loaded or deleted must be stored. | `<Installation folder of Historian ETL>/Historian ETL Load/ErrorFiles` |
| **Error File Life Time (days)** | Enter the amount of time, in days, error files must be stored before deletion. | 3.5 |

5. Select **Advanced Configuration**, and then provide values as specified in the following table.

| Field | Description | Default Value |
|---|---|---|
| **Log Level** | Select the log level to indicate the amount of information to be logged. The following options are available:<br>◦ INFO<br>◦ ERROR<br>◦ DEBUG | INFO |
| **Write Batch Size** | Enter the number of records to write to the Historian server at one time.<br>**Tip:** A large batch size can produce faster tag value writes, but can lead to poor performance | 1000 |

| Field | Description | Default Value |
|---|---|---|
| | if multiple collectors share the same Historian server. A 1000 to 2000 batch size is ideal for a shared Historian server. | |
| **Wait Interval (seconds)** | Enter the amount to time, in seconds, to wait to check for new files. | 90 |
| **Wait For Reply** | Select **False** if you do not want the Historian ETL Load tool to wait for the acknowledgement from Historian that files are transferred. | True |
| **Error On Replace** | Select **True** if you want an error to be logged if duplicate data is transferred. | False |
| **Max Retries** | Enter the maximum number of times the Historian ETL Load tool must try to resend data in case of an error or failure. | 2 |
| **Retry Timeout (seconds)** | Enter the amount of time, in seconds, to wait before trying to resend data. | 5 |

6. Select **Save**.
   The changes to the settings are applied and saved in the `HistorianETLLoad.exe.config` file.

Load data into the destination Historian server *(on page 2965)*.

## Load Data into the Destination Historian Server

1. Configure the ETL Load tool *(on page 2961)*.
2. Verify that the Historian ETL Load tool running on the destination Historian server is configured to watch the IIS Virtual folder for the incoming .zip files.

1. Run the `Historian ETL Load Configuration` file located in the `Historian ETL Load` folder.
   The **Historian ETL Load Configuration** window appears.
2. Select **Start Service**.

   **Tip:**
   If the tool does not start as expected, access the logs using Windows Event Viewer.

   The data is loaded in the destination Historian server.

# Data File Format

This topic provides the format of the content in a text file that contains tag data extracted by the Historian ETL Extract tool. Each line in the text file contains the following parameters, separated by commas:

| Parameter | Description | Valid Values |
|---|---|---|
| Time | The time at which the data is captured. It is displayed in the following format: <epoch format>:<nanoseconds, human-readable><br><br>For example, if the value is 1601890328:76000000, then:<br><br>• 1601890328 indicates the time in the epoch format.<br>• 76000000 indicates the time in nanoseconds, human-readable.<br><br>The value after the colon is greater than 0 only if the time | N/A |

| Parameter | Description | Valid Values |
|---|---|---|
| | resolution of the tag is milliseconds or microseconds. | |
| Tag name | The name of the tag for which the data is captured. | Any tag that exists in the destination Historian server<br><br>**Note:** If the AddTagAutomatically parameter is set to True, when you add a tag that does not exist in the Historian server, it will be created. |
| Data value | The value of the data that is captured. | N/A |
| Data type | The data type for the value that is captured. | • short<br>• int<br>• float<br>• Double<br>• Single Byte String |
| Quality | The quality of the data that is captured. | • GOOD<br>• BAD |

**Note:**

- The BLOB data type is not supported.
- Comments are not captured.
- Data quality other than GOOD and BAD is not supported.

## Example of a Regeneration File

The following lines of code represent the content of a regeneration file, which provides the start time, end time, and interval to export data. After all the requests are processed, the regeneration file is deleted.

**Note:**
The regeneration file has three sections for setting the start time, end time, and interval. These multiple sections act as a backup. That is, if the values provided in the first section are not valid (for example, the start time is later than the end time, interval has a negative value), the values from the next section are considered.

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:clr="http://schemas.microsoft.com/soap/encoding/clr/1.0"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body>
<a1:ArrayList id="ref-1" xmlns:a1="http://schemas.microsoft.com/clr/ns/System.Collections">
<_items href="#ref-2"/>
<_size>3</_size>
<_version>3</_version>
</a1:ArrayList>

<SOAP-ENC:Array id="ref-2" SOAP-ENC:arrayType="xsd:anyType[4]">
<item href="#ref-3"/>
<item href="#ref-4"/>
<item href="#ref-5"/>
</SOAP-ENC:Array>

<a3:RegenRequest id="ref-3"
 xmlns:a3="http://schemas.microsoft.com/clr/nsassem/ETLExtract/HistorianETLExtract%2C
%20Version%3D1.9.0.0%2C%20Culture%3Dneutral%2C%20PublicKeyToken%3Dnull">
<startTime>2020-07-02T09:39:37.3384336+05:30</startTime>
<endTime>2020-07-02T10:39:37.3384336+05:30</endTime>
<interval>3</interval>
</a3:RegenRequest>

<a3:RegenRequest id="ref-4"
 xmlns:a3="http://schemas.microsoft.com/clr/nsassem/ETLExtract/HistorianETLExtract%2C
%20Version%3D1.9.0.0%2C%20Culture%3Dneutral%2C%20PublicKeyToken%3Dnull">
<startTime>2020-07-02T09:39:37.3384336+05:30</startTime>
<endTime>2020-07-02T10:39:37.3384336+05:30</endTime>
```

```
<interval>4.5</interval>
</a3:RegenRequest>

<a3:RegenRequest id="ref-5"
 xmlns:a3="http://schemas.microsoft.com/clr/nsassem/ETLExtract/HistorianETLExtract%2C
%20Version%3D1.9.0.0%2C%20Culture%3Dneutral%2C%20PublicKeyToken%3Dnull">
<startTime>2020-07-02T09:39:37.3384336+05:30</startTime>
<endTime>2020-07-02T10:39:37.3384336+05:30</endTime>
<interval>14.5</interval>
</a3:RegenRequest>

</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

## Example of a State File

The following lines of code represent the content of a state file indicating that the last successful export occurred on June 6, 2010 at 5:00:04 PM in the UTC-4 time zone:

```
<SOAP-ENV:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:clr="http://schemas.microsoft.com/soap/encoding/clr/1.0"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">

<SOAP-ENV:Body>

<a1:Persist id="ref-1"
 xmlns:a1="http://schemas.microsoft.com/clr/nsassem/LBExport/LBExport%2C%20Version%3D1.0.1.0%2C%20Culture%3Dneutral%2C
%20PublicKeyToken%3Dnull">

<lastExport>2010-06-07T17:00:04.2017462-04:00</lastExport>

</a1:Persist>

</SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

# Troubleshooting ETL Issues

## Recommendations

Some overall recommendations when working with ETL to minimize issues:

- Extract Historian data for 30000 tags with 18MM samples per day.
- Set the Regen Interval to be less than 10 minutes.

## Unable to Start the ETL Tools

**Issue:** When you try to start the Historian ETL Extract, Historian ETL PI Extract, or the Historian ETL Load service, an error occurs.

**Diagnostics:** If there is a limited bandwidth of the internet, the certificate publisher service times out after 30 seconds. To avoid this issue, disable the publisher service:

1. Access the configuration file for the service that you want to start. For example, for the Historian ETL Extract service, access the `HistorianETLExtract.exe.config` file.
2. Set the element **generatePublisherEvidence** to disabled.

## An Error Occurs When Importing Tag Data into an Excel Spreadsheet

**Issue:** To specify tags using a template spreadsheet, when you import the tags stored in an .xml file, an error occurs, stating that the data is not valid according to the schema.

**Diagnostics:** Verify that:

- You have provided values in all the red-colored columns.
- Values in the Name column are unique.
- There are no blank rows.
- The values that you have entered are valid and of the same data type as defined for each property.

For details, refer to the Tag Properties *(on page 2942)* topic.

**Issue:** When you attempt to import tag data to an Excel spreadsheet from an .xml file, an error occurs, stating that the .xml file is not linked to schema.

**Workaround:** Ignore the message.