GE VERNOVA

PROFICY® SOFTWARE & SERVICES

PROFICY HISTORIAN FOR CLOUD – AWS

User Guide

GE VERNOVA

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

Cloud Historian

Contents

Chapter 1. Important Product Information

What's New in Proficy Historian for Cloud 2024

Proficy Historian is a best-in-class historian software solution that collects industrial time series data. It is secure, fast, and highly efficient.

Proficy Historian for Cloud allows you to deploy the Historian server and its components on cloud destinations. It supports all the components/tools in the on-premises counterpart of Proficy Historian (such as collectors, Excel Addin). And it offers all the advantages of cloud technologies.

Proficy Historian for Cloud offers plenty of new features and enhancements to help you connect your data with other analysis tools, migrate your data from Proficy Historian on-premises, scale up Data Archiver, and many more:

- **Enhanced REST APIs:** The following Rest APIs are now available with Cloud Historian: The Tagslist API, Managing Collector Instances, Managing Collectors, Managing Datastores, and Managing Systems.
- **Support for Proficy Authentication:** Starting with Cloud Historian 2024, a common User Authentication and Authorization (Proficy Authentication) service named Proficy Authentication is now provided with Cloud Historian. You can deploy it with other Proficy products such as Operations Hub or Plant Applications, or with the Cloud Historian deployment.
- **Support for Configuration Hub:** You can create and manage Historian systems using Configuration Hub. In addition, you can manage collectors, data stores, and tags. Cloud Historian can be configured with both Windows Configuration Hub and the Containerized Configuration Hub.
- **Storage of Future Data Support:** Previously, you could store only up to 1200 seconds of future data for Cloud Historian. Now, you can store unlimited future data (that is, up to 19 January, 2038) 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 (for example, using trend charts), and revise the forecasting algorithms as needed.
- **Remote Collector Management with Configuration Hub (Windows and Containerized):** Configuration Hub provides a user-friendly interface to manage collectors running on remote servers. You can also view the status of all the collectors with visual representation.
- **Common Collectors Installer for Windows and Cloud Historian:** Common collectors installer with a utility to run for enabling encryption for collectors to connect to Cloud Historian.
- **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 APIs resources without having any of the implementation logic in place, establishing a fully interactive documentation experience using Swagger

- **Other Proficy Historian Features:** Enumerated Sets, Array Tags, Complete Messages instead of just showing Message ID's, Collector Redundancy, and Backup & Restore.

Known Issues and Limitations

Limitations

The following are the limitations with Proficy Historian for Cloud:

Description	Tracking ID
You cannot add more than 1 Cloud Historian server in Configuration Hub	DE213095 Bug 282629
User Defined Types are not supported.	DE209835 Bug ID 87729
The Stale Tags functionality is not supported.	DE209835 Bug ID 87729
The same Proficy Authentication cannot be used for more than one Proficy Historian on Cloud.	Bug 87909
Alarms and Event Archiver does not work with Cloud Historian.	DE209835 Bug ID 87729
When upgrading Cloud Historian from any version prior to 2023, you first need to upgrade to any of the 2023 versions (2023, 2023.1, 2023.1.1) and *then* upgrade to 2024.	N/A
The .NET and CA API does not work with Cloud Historian as Cloud Historian is Linux based.	N/A
Historian as an OPC HDA Server and OPC UA DA Server is not supported.	N/A
The Historian Diagnostics Manager is not suppported with Cloud Historian.	N/A

Description	Tracking ID
The File collector, the HAB collector, the Cygnet collector, and the Alarm and Events collectors are not supported with Cloud Historian.	N/A
LDAPS is not supported with containerized Proficy Authentication.	N/A
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
Import more than 50000 tags to Historian Server for the OPC UA collector using Excel Add-in is not supported. **Workaround** It is recommended to add less than 30000 tags for a collection interval of 1 second or more.	DE212906

In addition:

- The performance of Historian Administrator when connected to Proficy Historian for Cloud is not good. We recommend using Configuration Hub.
- In general, it is recommended that you use Configuration Hub instead of Historian Administrator or the Web Admin.

Known Issues

Description	Tracking ID
Cannot create archives manually from the Historian Administrator (VB Admin) **Workaround:** Use Configuration Hub to perform any configurations.	91625
Not able to create array tags of data type VariableString There is currently no workaround.	283472
Restoring of Archives is not working from Historian Administrator **Workaround:** Use Configuration Hub to perform any configurations.	97114

Description	Tracking ID
Historian Node Name is set to "historian-svc" for S2S and S2D collectors when instantiated from Configuration Hub **Workaround:** Provide the NLB DNS in the Destination Historian server field during collector instantiation.	283138
Not able to configure tags for S2D collector from Configuration Hub **Workaround:** Use the Historian Administrator (VB Admin) to configure tags for S2D collector.	283284
SDK clients does not give the correct information about collector version **Workaround:** Use Configuration Hub instead.	239806
On a 32 bit machine, Collectors are communicating to Cloud Historian on 14000 port even after encryption is enabled There is currently no workaround.	250412
In the Server Properties, some of the properties are being shown as NA & some of them as 0's.	91561
Statistics are not shown for Cloud Historian tags in Operations Hub 2024 There is currently no workaround.	408311
If you import tags, along with their CreationTime and CreationUser, that were exported from Configuration Hub, the Excel Add-in displays the message: Proficy Historian Import Tags Error. **Workaround**: Maually remove the CreationTime and CreationUser columns in the exported file before importing it.	DE207159
When you use the Python collector to convert a value to boolean type, the results are provided in a string format. **Workaround**: When you convert a value to a Boolean type, you must store the results as Result = bool(Historian.CurrentValue('TP.Simulation00001'))	N/A
The trend decimal values in Configuration Hub are different from those in Historian Administrator. There is currently no workaround.	DE212903

Description	Tracking ID
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
DA Crashes when you try to create UDT using JAVA API. UDTs are not supported with Cloud Historian. In this case, DA automatically restarts.	503389
Read Sample Rate and Receive Rate are not showing correct results in Configuration Hub.	516219
In Operations Hub 2024 Trend Card, Statistics are not shown for Cloud Historian tags.	502854
Redundancy options are enabled for some of the collectors like Server to Server, ODBC, OPCHDA, and Wonderware Collectors in Configuration Hub for which redundancy is not supported.	283537
Soft deleted tags are not showing up in the reports.	363074

Resolved Issues

Resolved Issues

Description	Tracking ID
Historian Administrator Errors Occur when Updating Calculation Collector Functions. This issue was resolved in Historian for Cloud 2024.	01076495 DE200412
Historian Data collection issues - Data Archiver no longer accepting connections. This issue was resolved in Historian for Cloud 2024.	01078435 01076667 01083611
Historian Calculation Collector keeps stopping. This issue was resolved in Historian for Cloud 2024.	DE200411 DE200408

Description	Tracking ID
The tag import not working with the Excel Add-in. This issue was resolved in Historian for Cloud 2024.	01083632

Chapter 2. Overview

About Proficy Historian for Cloud

Proficy Historian is a best-in-class historian software solution that collects industrial timeseries data. It is secure, fast, and highly efficient.

Proficy Historian for Cloud allows you to deploy the Historian server and its components on cloud destinations. Specifically, Proficy Historian for AWS allows you to deploy the Historian server on AWS.

You can deploy Proficy Historian for AWS on your own virtual private cloud (VPC) *(on page 50)*, or you can let us create one for you *(on page 44)*. You can also use the Historian license *(on page 56)* if you have one or opt for a consumption-based model (in which the bill is generated based on your data consumption).

It supports all the clients in the on-premises counterpart of Proficy Historian (such as collectors, Excel Addin). You can install them on-premises or on an EC2 instance (either on the same or a different VPC), and then connect to Data Archiver deployed on AWS.

Advantages of using Proficy Historian for AWS:

- **AWS Marketplace:** In addition to Proficy Historian for AWS, the Web Admin console and the scheduled export feature are available in the AWS marketplace.
- **Load balancing:** With the use of Amazon Network Load Balancer (NLB), load balancing is achieved.
- **Integration with CloudWatch and CloudTrail:** Using CloudWatch, you can access logs *(on page 1533)*, which help in troubleshooting issues. Using CloudTrail, you can access events *(on page 1534)*, and thus view the data consumption while reading or writing data samples.
- **File Storage in Amazon Elastic File System (EFS):** Files are stored in EFS, thus achieving scalability and security. You can access the archives and log files *(on page 1540)* created by Historian.
- **Secure and highly available:** Proficy Historian for AWS is deployed on a virtual private cloud (VPC). It uses private subnets and TLS encryption, thus making it secure.

  In addition, Proficy Historian for AWS is deployed on multiple availability zones. If Data Archiver goes down, a new one is created in less than 10 seconds, thus making it highly available.

Components of Proficy Historian for AWS

Proficy Historian for AWS contains the following components:

- **Collectors:** Collect tag data from various data sources. You can install collectors *(on page)* on multiple Windows machines. These machines can be on-premises or on a virtual private cloud (VPC).
- **Clients:** Include the following applications:
    - Web Admin console *(on page)*
    - Excel Addin *(on page 990)*
    - Excel Addin for Operations Hub
    - The OLEDB Provider *(on page 1042)*
    - Historian Administrator *(on page 1193)*
    - The Extract, Transform, and Load (ETL) tools

  You can use them to retrieve and analyze the data stored in Historian. You can install them on multiple Windows machines. These machines can be on-premises or on a VPC.
- **AWS Network Load Balancer (NLB):** Receives data read or write requests from clients or collectors. If user authentication is needed, these requests are sent to Proficy Authentication for authentication. Otherwise, these requests are sent to Data Archiver.

> **Note:**
> The AWS NLB DNS name can be quite lengthy, for CIMPLICITY sources use the AWS IP address.

- **Amazon Elastic Kubernetes Cluster (EKS):** Hosts and maintains scalability and high availability of the following components:
    - **Data Archiver:** Intercepts requests from NLB. Sends user authentication requests to Proficy Authentication. Sends data read/write requests to Amazon Elastic File System (EFS).
    - **Proficy Authentication:** An application that authenticates user credentials and grants access for Proficy applications.
    - **PostgreSQL:** A database that stores user credentials.
- **Amazon Elastic File System (EFS):** Stores tag data in .iha files and configuration data in .ihc files, along with log files and buffer files. You can access these archive files and log files *(on page 1540)*. It is fully managed, scalable, highly available, and durable.
- **CloudWatch:** Contains the logs *(on page 1533)* generated by Data Archiver. You can also access the dashboard, which contains various widgets for your analysis and monitoring.
- **CloudTrail:** Contains the events *(on page 1534)* generated by Data Archiver, containing information on the data consumption.

Compatibility with Other Proficy Products

Several Proficy products work with Proficy Historian for Cloud. The following is a general set of required versions to work with Proficy Historian for Cloud.

Product	Supported Version
CIMPLICITY	2023, 2024 pre-release
iFIX	2023, 2024 pre-release
Plant Applications	2023, 2023.1 SP2
Workflow	2.6 SP1
Operations Hub	2023.1, 2024 pre-release
On Prem Historian Clients	2022, 2023, 2024
Dream Reports	2023 Patch1

Chapter 3. Pricing and Billing

Access Pricing Information

To use Historian, you can choose between the following options:

- **License-based (BYOL):** If you want to use the BYOL model, you can use your Proficy Historian for Linux license or buy it, and then apply it *(on page 56)*. You can then proceed to work with Proficy Historian for Cloud. In this case, you can deploy and use Proficy Historian for AWS directly.
- **Consumption-based:** If you want to use the consumption-based model, you must pay for using Proficy Historian for Cloud based on the consumption (that is, the number data samples that are read/written to Data Archiver).

This topic describes how to access the price for fetching or writing data to Data Archiver per million samples. This information is available before you deploy Historian. It helps you estimate the cost of using Historian and plan accordingly.

If, however, you have already begun using Historian, you can access the cost incurred to fetch or write data to Data Archiver and make payment *(on page 29)*.

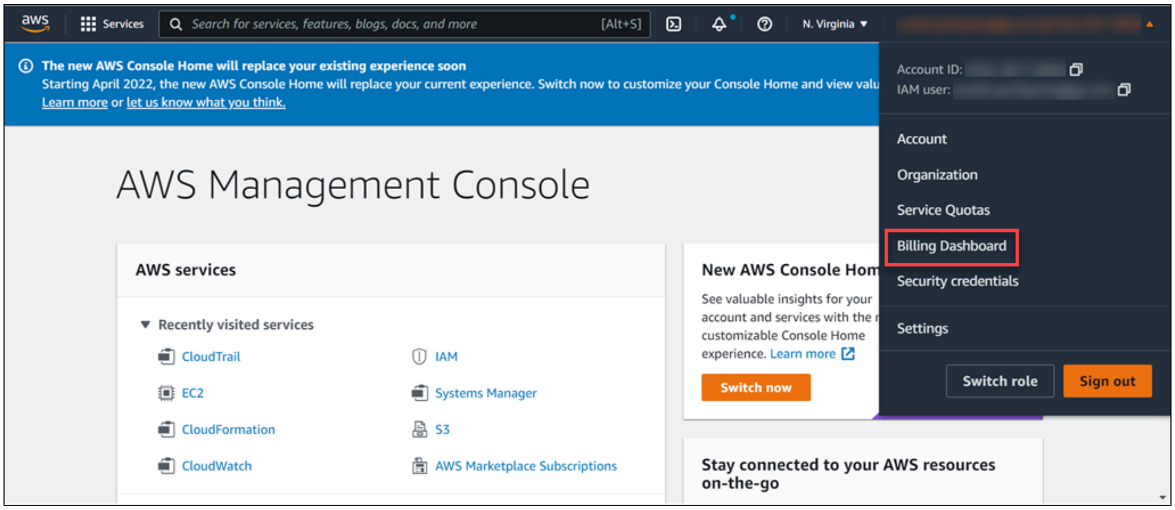
1. Log in to the AWS console.
2. Access the following URL: https://aws.amazon.com/marketplace/pp/prodview-rbmhtw3icmtqw. Or, search for Proficy Historian for AWS.
   The pricing information of Proficy Historian for AWS appears in the **Pricing Information** section.

Make Payment for Using Proficy Historian for AWS

In a consumption-based model, you can use Proficy Historian for AWS. Based on the data consumption, a bill is generated in the AWS console. You must pay the amount mentioned in the bill to continue using Historian.

This topic describes how to access your bill and make payment. To access the pricing information, refer to Access Pricing Information *(on page 29)*.

1. Log in to the AWS console.
2. In the upper-right corner of the page, select your username, and then select **Billing Dashboard**.



The new AWS Console Home will replace your existing experience soon

Starting April 2022, the new AWS Console Home will replace your current experience. Switch now to customize your Console Home and view valu

Learn more or let us know what you think.

AWS Management Console

New AWS Console Hom

See valuable insights for your account and services with the r customizable Console Home experience. Learn more

Stay connected to your AWS resources on-the-go

Account ID: [redacted]

IAM user: [redacted]

The **Billing Dashboard** page appears. You can also have the invoice emailed to you.

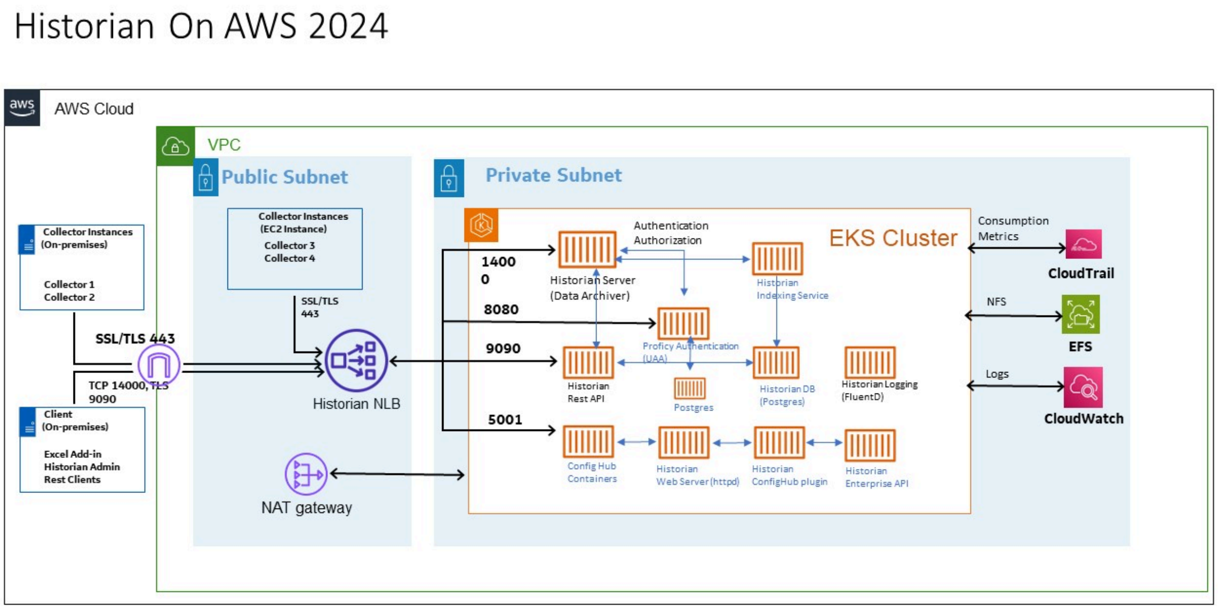
3. Make the payment.

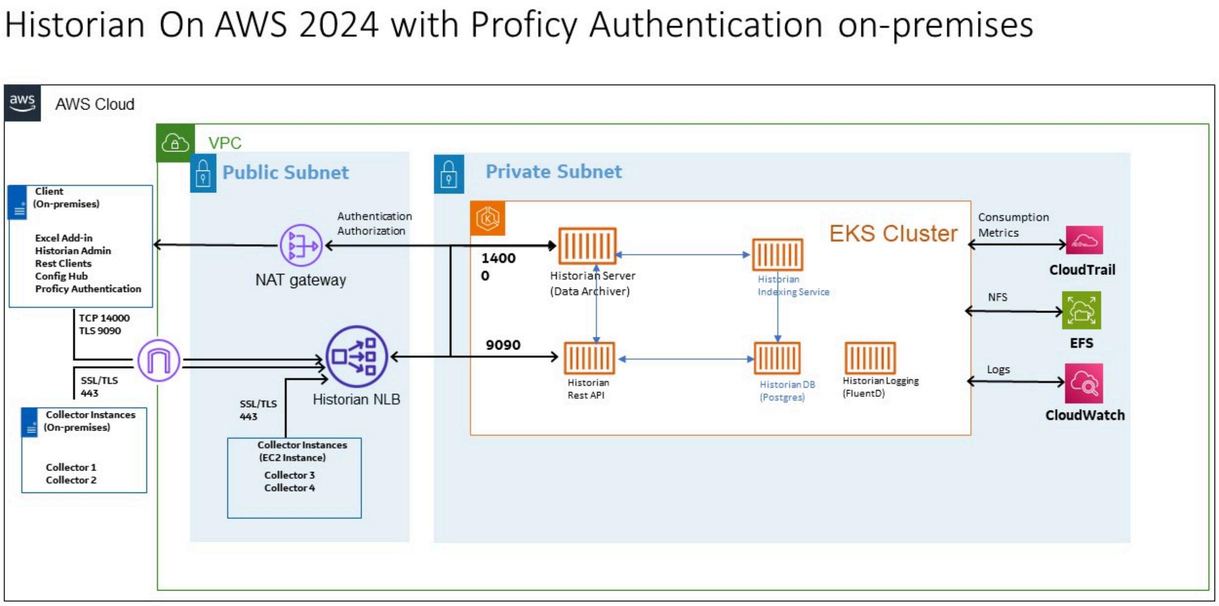
Chapter 4. Deployment

Deployment Architecture

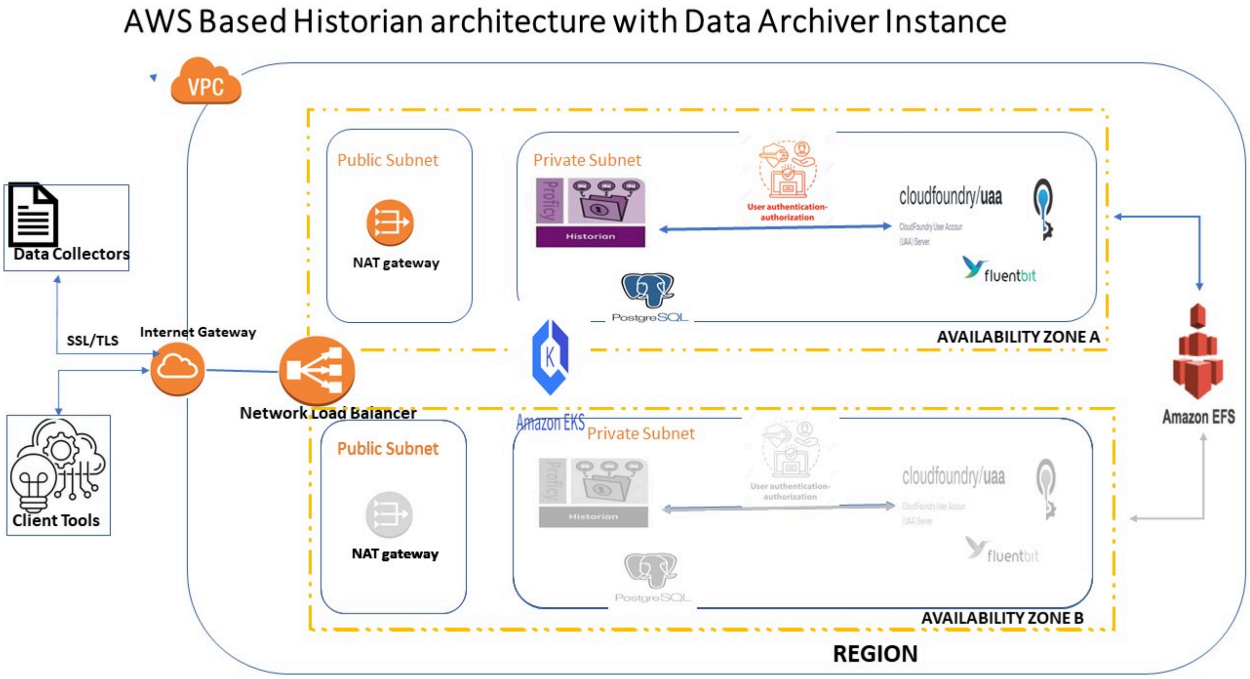
The following diagram shows the deployment architecture of Proficy Historian for AWS. In this diagram:

- Data Archiver, Proficy Authentication, PostgresSQL, Configuration Hub, and Historian Rest APIs are deployed in an Elastic Cloud Compute (EC2) instance in a private subnet inside Amazon Elastic Kubernetes Service (EKS).
- Amazon Elastic File System (EFS) is connected to Data Archiver.
- Network Load Balancer (NLB), collector instances, and the NAT Gateway are in a public subnet.
- EFS is in the Virtual Private Cloud (VPC), whereas CloudWatch and CloudTrail are outside the VPC. EFS sends archiver logs to CloudWatch, which you can use for analysis. CloudTrail is used to access events.
- Collector 1 and Collector 2 are collector instances created on an on-premises Windows machine. Similarly, Excel Addin for Historian and Historian Administrator are installed on an on-premises client machine.
- Collector 3 and Collector 4 are collector instances created on an EC2 instance in a VPC (can be a different VPC than the one in which the Historian server is deployed).





This next diagram shows the high availability architecture:



How tag data is stored if using collectors without TLS encryption:

1. Collectors send a request to AWS Network Load Balancer (NLB) to write tag data.
2. NLB sends the request to Data Archiver. If user authentication is needed, Data Archiver sends the request to Proficy Authentication, which verifies the user credentials stored in PostgreSQL. After authentication, NLB confirms to the collectors that data can be sent.
3. Data collected by the collector instances is sent to NLB.
4. NLB sends the data to Data Archiver directly. After authentication, Data Archiver stores the data in EFS in .iha files.

How tag data is stored if using collectors with TLS encryption enabled:

1. Collectors send a request to AWS NLB to write tag data. Since the request is encrypted, port 443 is used.
2. NLB decrypts the request and sends it to Data Archiver. If user authentication is needed, Data Archiver sends the request to Proficy Authentication, which verifies the user credentials stored in PostgreSQL. After authentication, NLB confirms to the collectors that data can be sent.
3. Data collected by the collector instances is encrypted and sent to NLB using port 443.
4. NLB decrypts the data and sends it to Data Archiver. After authentication, Data Archiver stores the data in EFS in .iha files.

How data is retrieved:

1. Clients (that is, Excel Addin, the Web Admin console, the REST Query service, or Historian Administrator) send a request to NLB to retrieve data.
2. NLB sends the request to Data Archiver, which retrieves data from EFS. If, however, user authentication is needed, Data Archiver sends the request to Proficy Authentication, which verifies the user credentials stored in PostgreSQL. After authentication, data is retrieved from EFS.

Prerequisites

Hardware Requirements

Proficy Historian for AWS

Proficy Historian for AWS is deployed in an Elastic Cloud Compute (EC2) instance in Elastic Kubernetes Cluster (EKS) inside a Virtual Private Cloud (VPC). During deployment choose an instance type based on your requirement. Our benchmarking results suggest that M5.2XLarge supports up to 15 million samples per minute. You can choose an instance of lower capacity based on the rate of collection.

Collectors

You can install collectors on an on-premises Windows machine. Or, you can install them on a machine in the same VPC where the Historian server is deployed, or you can choose to deploy in a different VPC.

The following table provides the hardware requirements for installing collectors on an on-premises Windows machine.

Hardware Component	Recommendation
RAM	8 GB
Disk size	80 GB

Sizing Recommendations

We recommend the following configuration of EC2 instances depending on the type of environment or the volume of data.

Environment or Volume of Data	Recommendation
Production Environment with write rate < 3MM samples/min	C5.XLarge
Production Environment with write rate 3 - 10MM samples/min	M5.XLarge
Production Environment with write rate 10 – 15MM samples/min	M5.2XLarge
Production Environment with Scheduled Export job running	M5 series (M5.Large for smaller systems and M5.Xlarge and M5.2Xlarge for bigger systems respectively).
Test environment	C5.XLarge

Software Requirements

Collectors

Install any of the following operating systems:

- 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.

The other requirements, such as Microsoft®.NET Framework, are installed automatically.

> **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).

Excel Add-in for Historian

You can install Excel Add-in for Historian on an on-premises machine or on an EC2 instance in a VPC. To use Excel Add-in for Historian, install any of the following versions of Microsoft® 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)

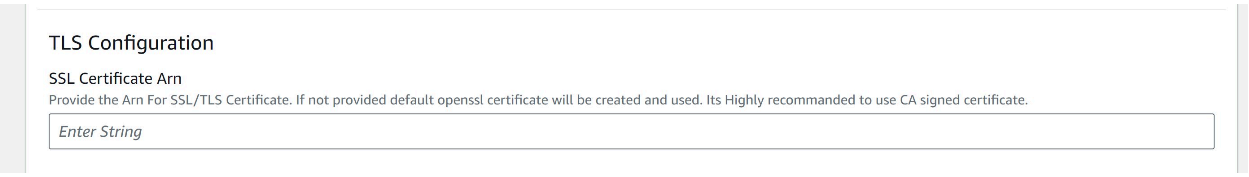
Create Self-Signed OpenSSL Certificate for NLB

Overview

For secure traffic between Historian collectors and to access containerized Configuration Hub and Proficy Authentication with Cloud Historian, an SSL certificate is required to be used with NLB.

When you deploys Historian on AWS 2024 with the Cloud Formation Template, there are two options related to SSL certificates:

- Leave SSL Certificate Arn parameter empty: In this case, a self-signed SSL certificate will be created by the CFT using OpenSSL command in an AWS Lambda function.
- Provide arn of SSL certificate already uploaded to either AWS ACM / IAM: In this case, CFT will not create a self-signed SSL certificate; instead you must provide the SSL certificate to be used with NLB listeners.

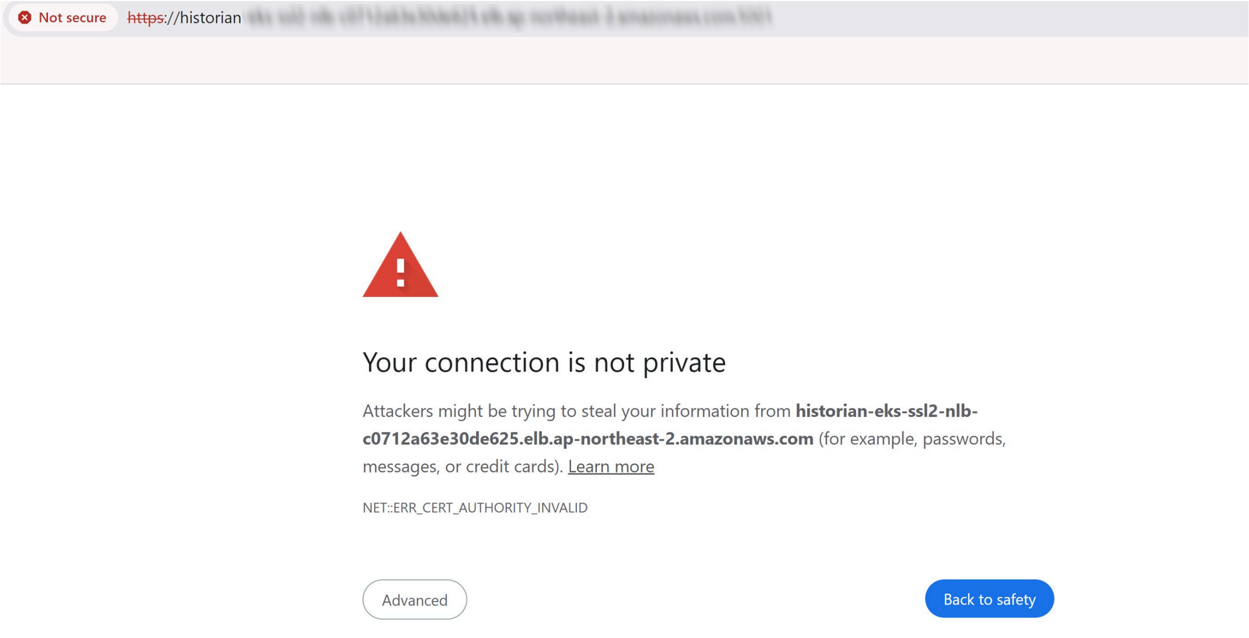


TLS Configuration

SSL Certificate Arn

Provide the Arn For SSL/TLS Certificate. If not provided default openssl certificate will be created and used. Its Highly recommanded to use CA signed certificate.

In both the cases, the SSL certificates are created without any knowledge of the NLB DNS. The NLB DNS will only be known when CFT deployment completes. Due to this, when you access Configuration Hub or Proficy Authentication deployed with Historian CFTs, a Non-secure field will be displayed on the web browser, as shown in the following figure.



Your connection is not private

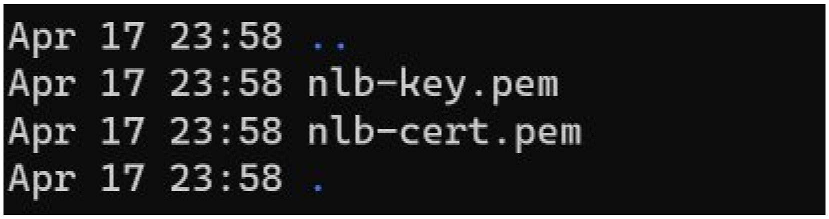
Attackers might be trying to steal your information from **historian-eks-ssl2-nlb-c0712a63e30de625.elb.ap-northeast-2.amazonaws.com** (for example, passwords, messages, or credit cards). Learn more

NET::ERR_CERT_AUTHORITY_INVALID

To prevent browser from displaying the Non-secure field, you need to create an SSL certificate by providing NLB DNS in SAN (Subject Alternative Names) field. To create such an SSL certificate, you can use the openssl package in Linux based systems, by issuing the following commands:

```
openssl genpkey -algorithm RSA -out ./nlb-key.pem -pkeyopt rsa_keygen_bits:2048

openssl req -x509 -new -nodes -key ./nlb-key.pem -sha256 -out nlb-cert.pem -subj

 "/O=ProficyHistorian/OU=MFG/CN=CloudHistorian" -addext "subjectAltName = DNS:<NLB_DNS>" -days 3600
```

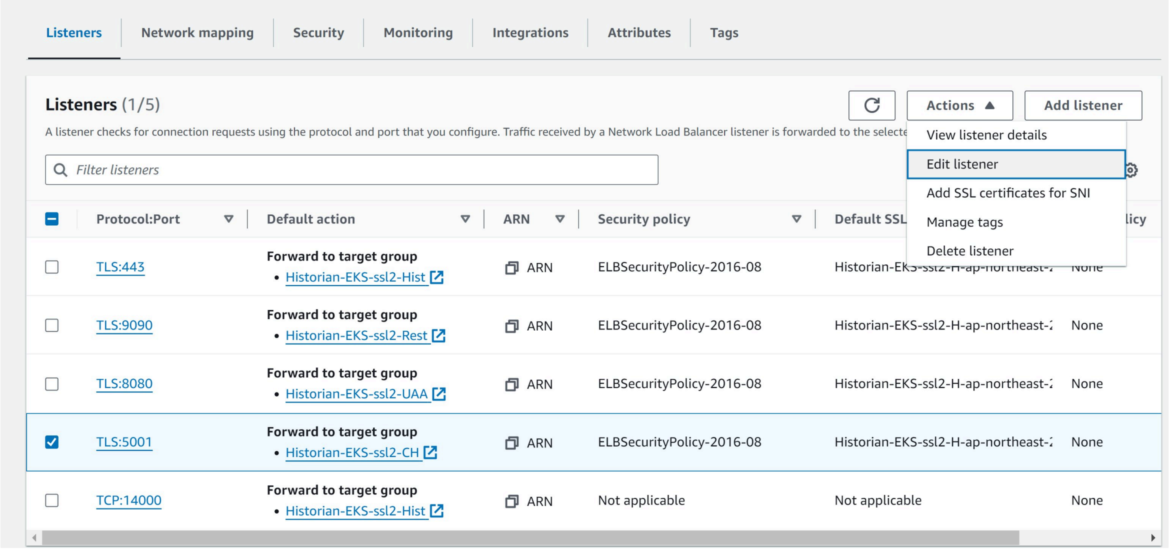
After executing the previous two commands, the private key (nlb-key.pem) and the certificate (nlb-cert.pem) will be created. The contents of these two files will be required later when uploading to AWS IAM / ACM:



```
Apr 17 23:58 ..
Apr 17 23:58 nlb-key.pem
Apr 17 23:58 nlb-cert.pem
Apr 17 23:58 .
```

Steps from the AWS Management Console

1. Access the **AWS Management Console** page.
2. From the AWS Management console, select the Region where Historian is deployed, and then EC2 > Load Balancers > Select Historian NLB > Go to Listeners.
3. Edit the Listener with port 5001 (Config Hub), as shown in the following figure:



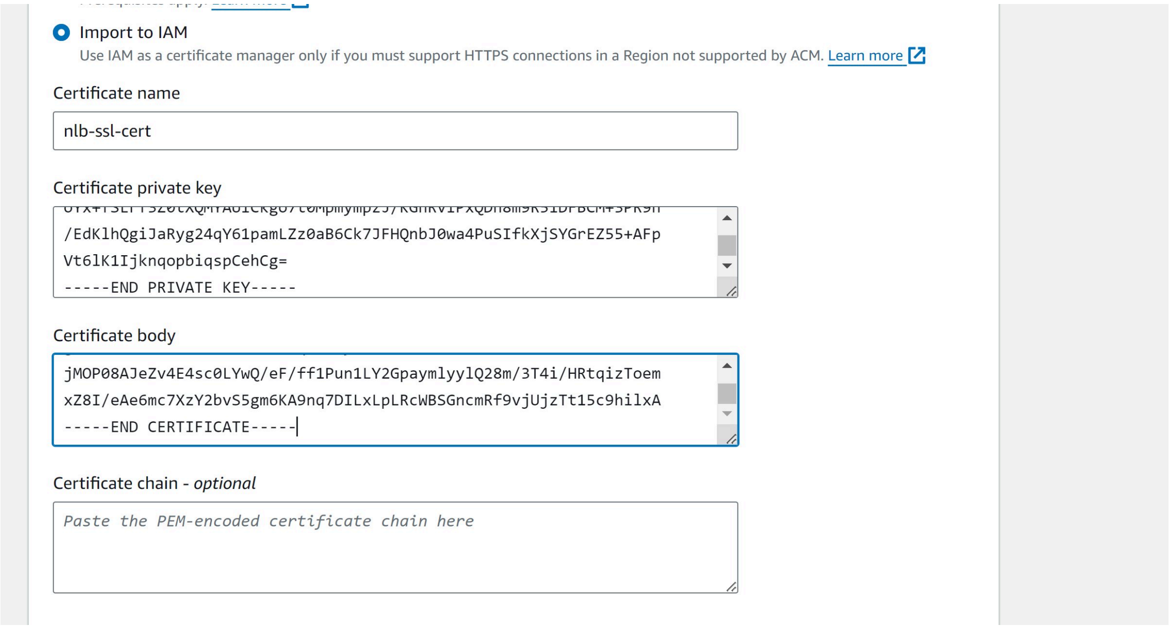
Listeners (1/5)

A listener checks for connection requests using the protocol and port that you configure. Traffic received by a Network Load Balancer listener is forwarded to the selecte

☐	Protocol:Port	Default action	ARN	Security policy	Default SSL	licy
☐	TLS:443	Forward to target group • Historian-EKS-ssl2-Hist	ARN	ELBSecurityPolicy-2016-08	Historian-EK	None
☐	TLS:9090	Forward to target group • Historian-EKS-ssl2-Rest	ARN	ELBSecurityPolicy-2016-08	Historian-EKS-ssl2-H-ap-northeast-2	None
☐	TLS:8080	Forward to target group • Historian-EKS-ssl2-UAA	ARN	ELBSecurityPolicy-2016-08	Historian-EKS-ssl2-H-ap-northeast-2	None
☑	TLS:5001	Forward to target group • Historian-EKS-ssl2-CH	ARN	ELBSecurityPolicy-2016-08	Historian-EKS-ssl2-H-ap-northeast-2	None
☐	TCP:14000	Forward to target group • Historian-EKS-ssl2-Hist	ARN	Not applicable	Not applicable	None

4. Under the Default SSL/TLS server certificate, click on **Import certificate**. Select either import to ACM or IAM.

5. Provide the certificate private key (nlb-key.pem) and the certificate body (nlb-cert.pem) created in the previous steps and then **Save Changes**.



● Import to IAM

Use IAM as a certificate manager only if you must support HTTPS connections in a Region not supported by ACM. Learn more

Certificate name

nlb-ssl-cert

Certificate private key

```
/EdKlhQgiJaRyg24qY61pamLZz0aB6Ck7JFHQnbJ0wa4PuSIfkXjSYGrEZ55+AFp
Vt6lK1IjknqopbiqspCehCg=
-----END PRIVATE KEY-----
```

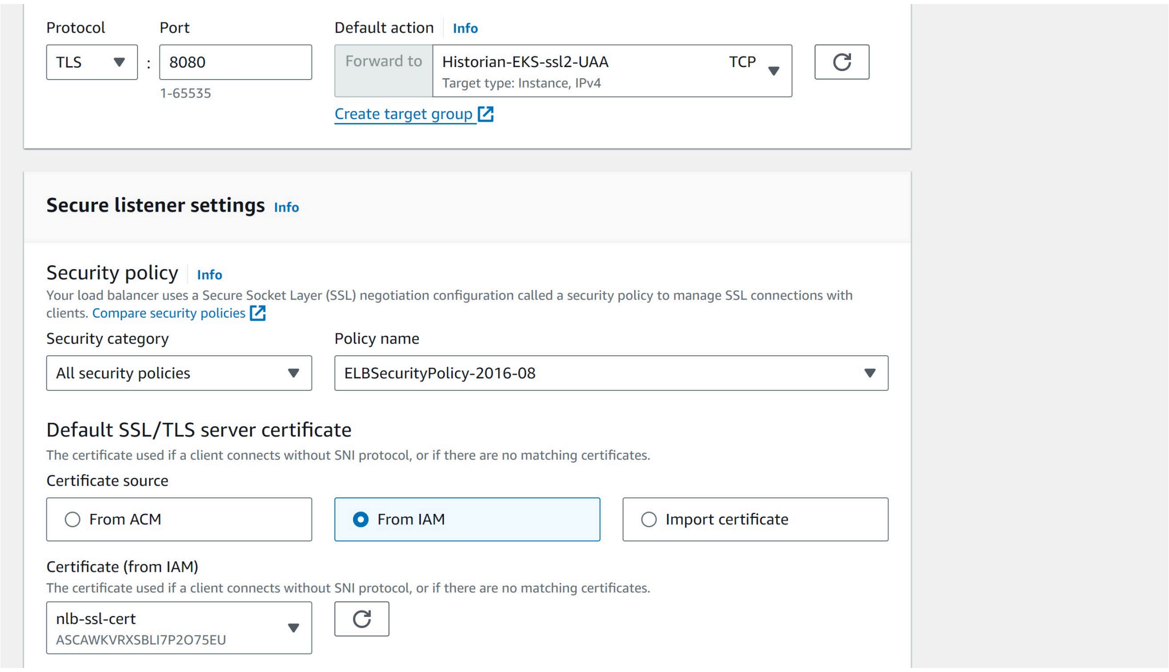
Certificate body

```
jMOP08AJeZv4E4sc0LYwQ/eF/ff1Pun1LY2Gpaymlyy1Q28m/3T4i/HRtqizToem
xZ8I/eAe6mc7XzY2bvS5gm6KA9nq7DILxLpLRcWBSGncmRf9vjUjzTt15c9hilxA
-----END CERTIFICATE-----
```

Certificate chain - *optional*

Paste the PEM-encoded certificate chain here

6. Repeat these steps for the Proficy Authentication listener (8080), except this time, use the already uploaded SSL certificate:



Protocol	Port	Default action Info
TLS	8080	Forward to Historian-EKS-ssl2-UAA TCP Target type: Instance, IPv4
	1-65535	Create target group

Secure listener settings Info

Security policy Info

Your load balancer uses a Secure Socket Layer (SSL) negotiation configuration called a security policy to manage SSL connections with clients. Compare security policies

Security category: All security policies

Policy name: ELBSecurityPolicy-2016-08

Default SSL/TLS server certificate

The certificate used if a client connects without SNI protocol, or if there are no matching certificates.

Certificate source

○ From ACM ● From IAM ○ Import certificate

Certificate (from IAM)

The certificate used if a client connects without SNI protocol, or if there are no matching certificates.

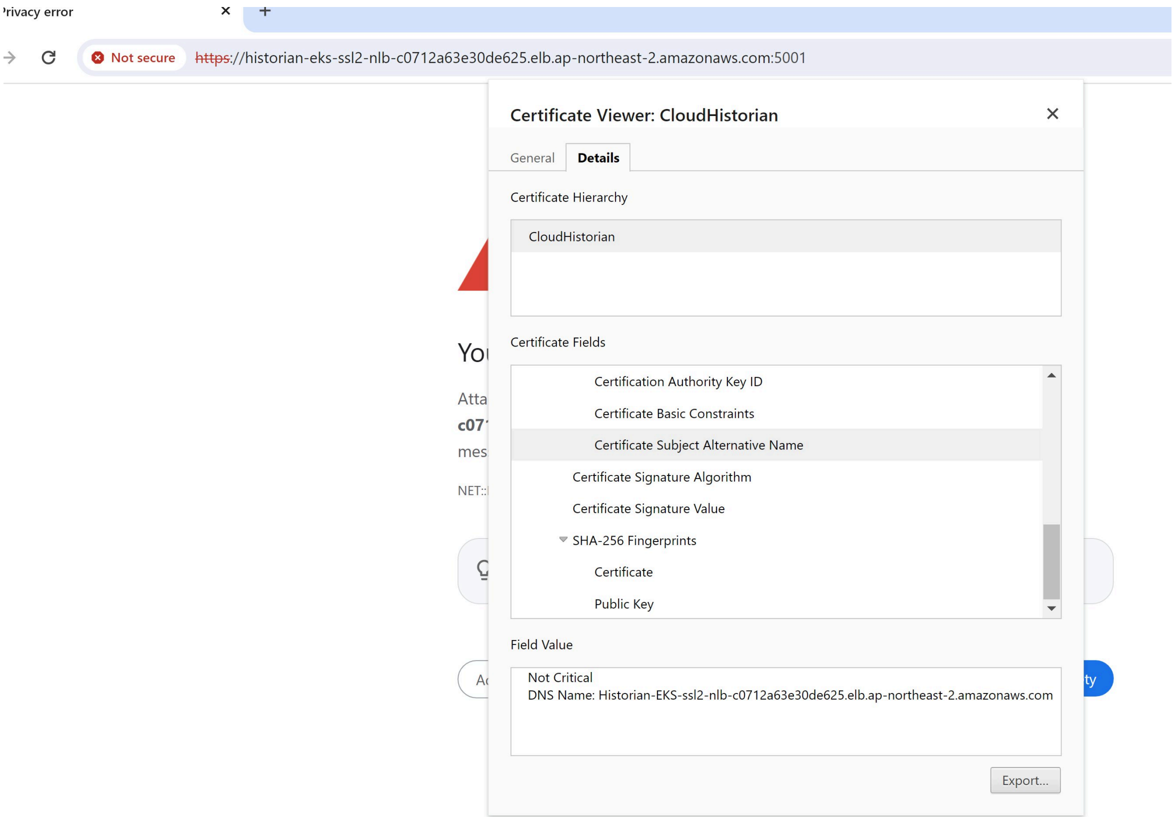
nlb-ssl-cert
ASCAWKVRXSBLI7P2O75EU

Note:

The following steps are for windows based machines:

7. Access the NLB with the 5001 port from a web browser.

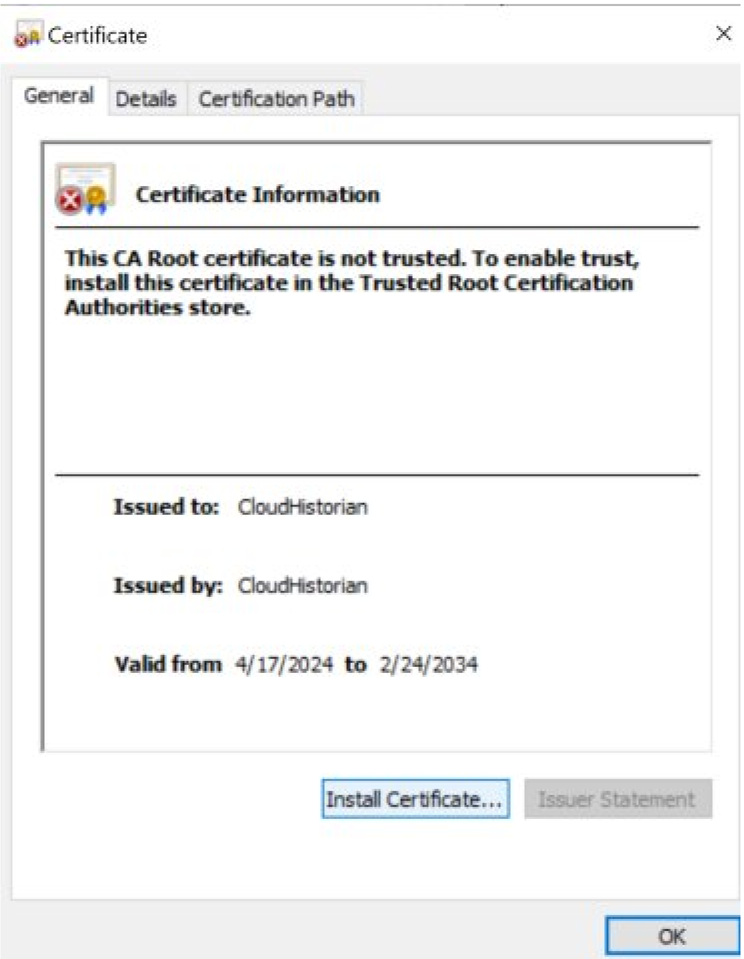
8. View the certificate, and notice under the **Certificate Subject Alternative Name** field, that the NLB DNS is set.



'rivacy error
Not secure
https://historian-eks-ssl2-nlb-c0712a63e30de625.elb.ap-northeast-2.amazonaws.com:5001
Certificate Viewer: CloudHistorian
General
Details
Certificate Hierarchy
CloudHistorian
Certificate Fields
Certification Authority Key ID
Certificate Basic Constraints
Certificate Subject Alternative Name
Certificate Signature Algorithm
Certificate Signature Value
SHA-256 Fingerprints
Certificate
Public Key
Field Value
Export...

Not Critical
DNS Name: Historian-EKS-ssl2-nlb-c0712a63e30de625.elb.ap-northeast-2.amazonaws.com

9. Export this certificate and install the .crt file:



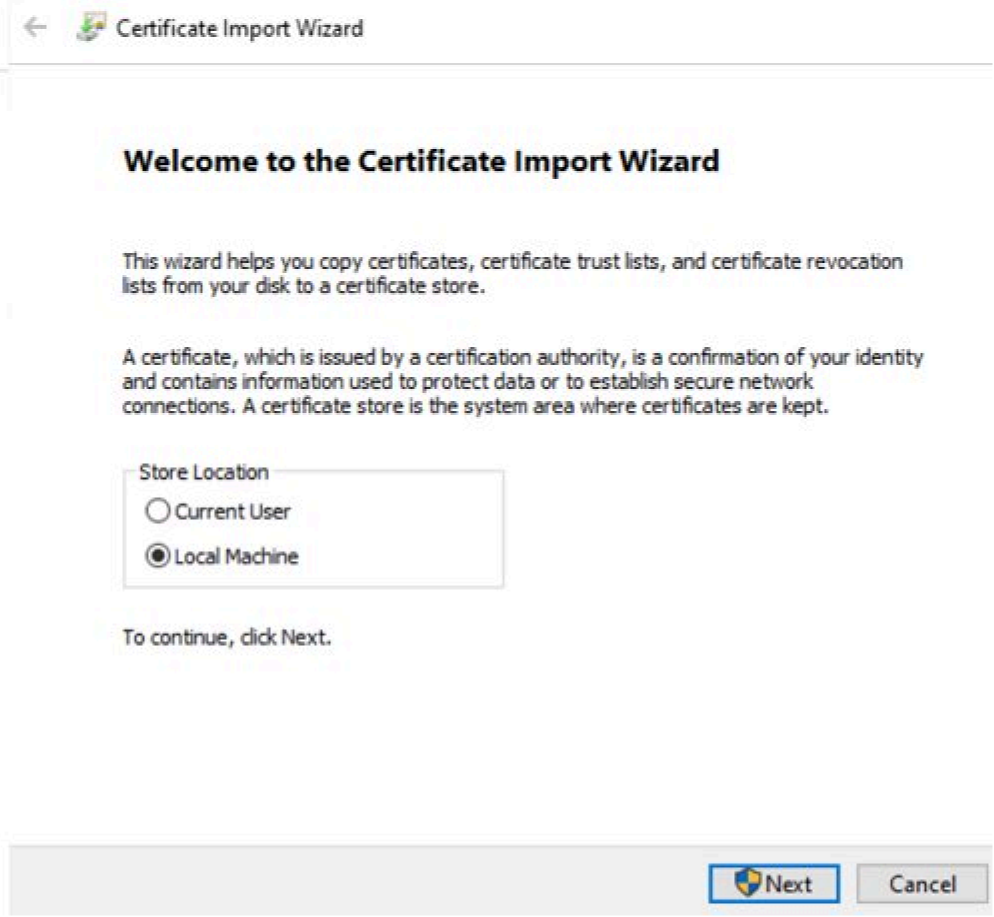
This CA Root certificate is not trusted. To enable trust, install this certificate in the Trusted Root Certification Authorities store.

Issued to: CloudHistorian

Issued by: CloudHistorian

Valid from 4/17/2024 to 2/24/2034

10. Select the **Local Machine** for the location, as shown in the following figure.



Certificate Import Wizard

Welcome to the Certificate Import Wizard

This wizard helps you copy certificates, certificate trust lists, and certificate revocation lists from your disk to a certificate store.

A certificate, which is issued by a certification authority, is a confirmation of your identity and contains information used to protect data or to establish secure network connections. A certificate store is the system area where certificates are kept.

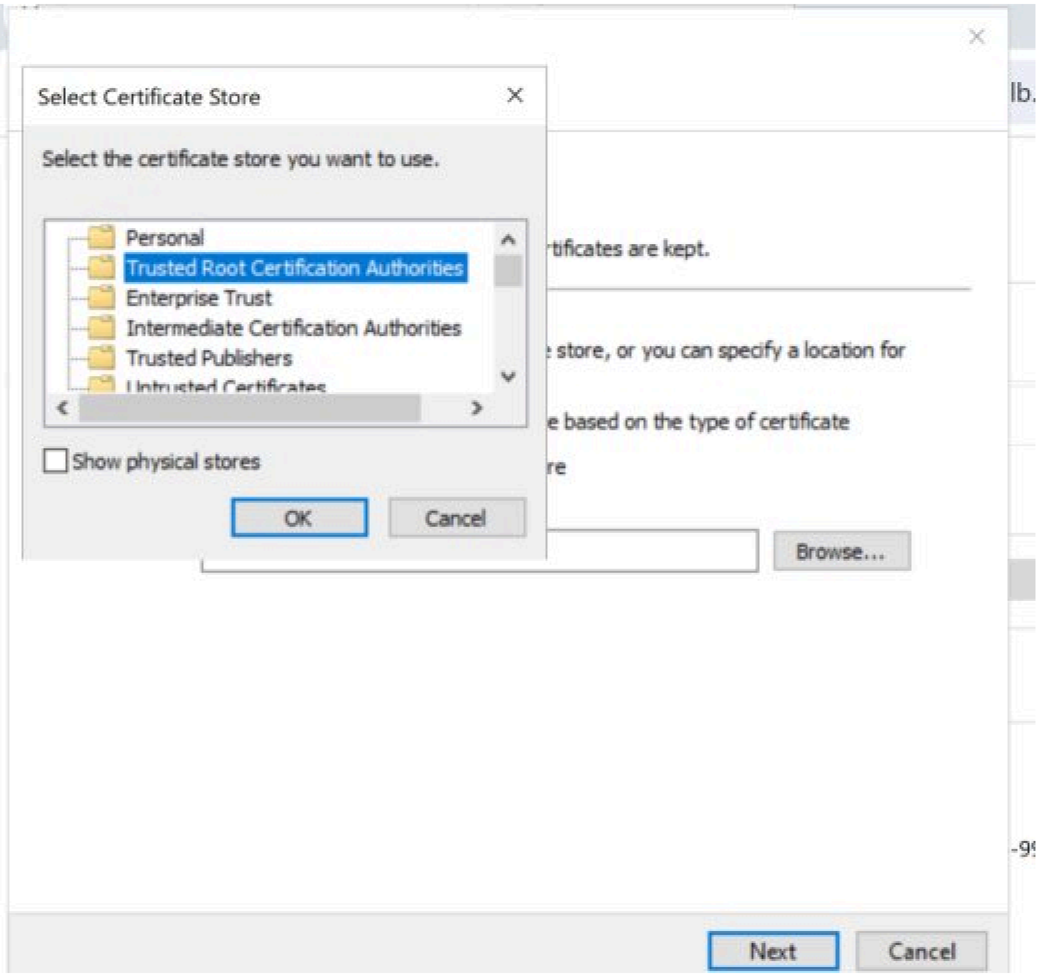
Store Location

○ Current User

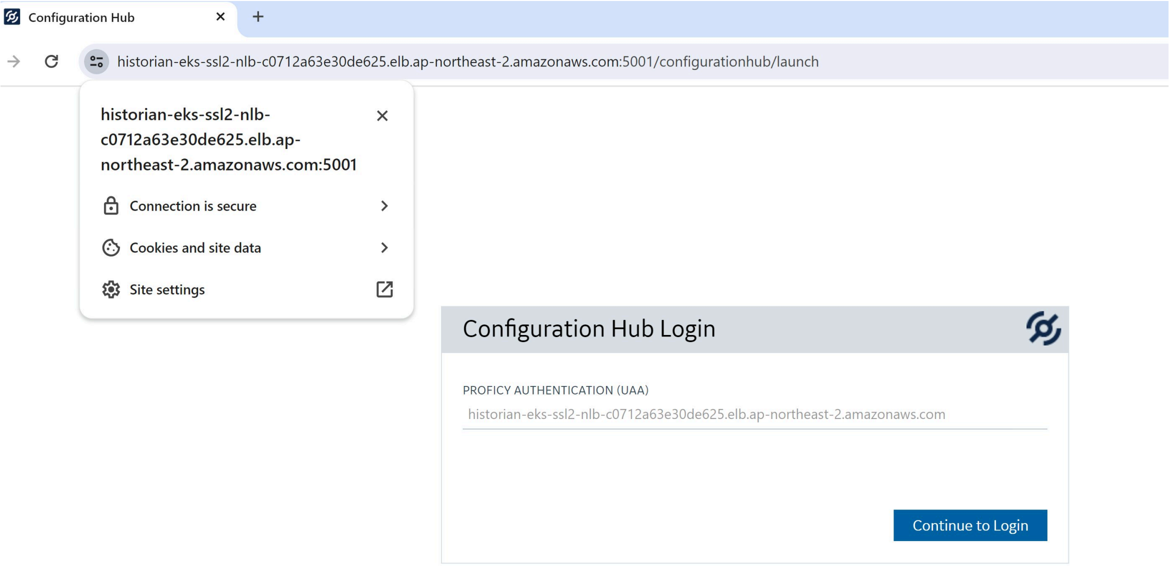
● Local Machine

To continue, click Next.

11. When prompted, select the Windows **Trusted Root Certification Authorities** store and complete the install.


Select Certificate Store
Select the certificate store you want to use.
Personal
Trusted Root Certification Authorities
Enterprise Trust
Intermediate Certification Authorities
Trusted Publishers
Untrusted Certificates
☐ Show physical stores
OK
Cancel
rtificates are kept.
: store, or you can specify a location for
e based on the type of certificate
re
Browse...
lb.
-9
Next
Cancel

12. Now, close all web browser tabs, restart the browser, and access the Historian NLB on port 5001.


Configuration Hub
historian-eks-ssl2-nlb-c0712a63e30de625.elb.ap-northeast-2.amazonaws.com:5001/configurationhub/launch
historian-eks-ssl2-nlb-c0712a63e30de625.elb.ap-northeast-2.amazonaws.com:5001
Connection is secure
Cookies and site data
Site settings
Configuration Hub Login
PROFICY AUTHENTICATION (UAA)
historian-eks-ssl2-nlb-c0712a63e30de625.elb.ap-northeast-2.amazonaws.com
Continue to Login

The Connection should now display as secure and a valid certificate will be displayed, as shown in the previous figure.

About Deploying Proficy Historian for AWS

To deploy Proficy Historian for AWS, you need a virtual private cloud (VPC), which allows you to deploy Proficy Historian into a virtual network that you have defined. For more information, refer to https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html.

Applying the license: You can use Proficy Historian for AWS with or without the Historian license. Accordingly, the following products are available in the AWS marketplace:

- **Bring your own license (BYOL):** Use this product if you have the Historian for Linux license (or if you plan to get one). After you deploy Proficy Historian for AWS, apply the license *(on page 56)*.
- **Consumption-based:** Use this product if you do not have the Historian for Linux license. A bill is generated based on your data consumption. For information on the pricing, refer to Access Pricing Information *(on page 29)*.

Deployment options:

- By creating a new VPC *(on page 44)*: Use this option if you do not have a VPC, or if you want to create a separate VPC for Proficy Historian. If you choose this option, a new VPC is created, along with the public and private subnets (one for each availability zone), the NAT Gateway, and IGW. Proficy Historian for AWS is then deployed in that VPC.
- By using an existing VPC *(on page 50)*: Use this option if you already have a VPC in which you want to deploy Proficy Historian for AWS as well. If you choose this option, Proficy Historian, along with the required resources, is deployed in your VPC.

After you deploy Proficy Historian for AWS, the following resources are deployed in the stack:

- Elastic Kubernetes Service (EKS): Contains a cluster of EC2 instances.
- Elastic File System (EFS): Stores the tag data and the tag configuration details.
- Network Load Balancer (NLB): Receives requests from collector/client machines and directs them to the Proficy Authentication service. The NLB DNS is also called the public IP address of the NLB. It is used to establish a connection between the Historian server and collectors/clients deployed on an on-premises machine or on a different VPC.
- EC2 Instance: Contains the Data Archiver, Proficy Authentication, and PostgreSQL containers.

For more information on these components, refer to Deployment Architecture *(on page 31)*.

Deploy Proficy Historian for AWS in a New VPC

Ensure that you are in the region in which you want to deploy Proficy Historian for AWS. The following regions are supported:

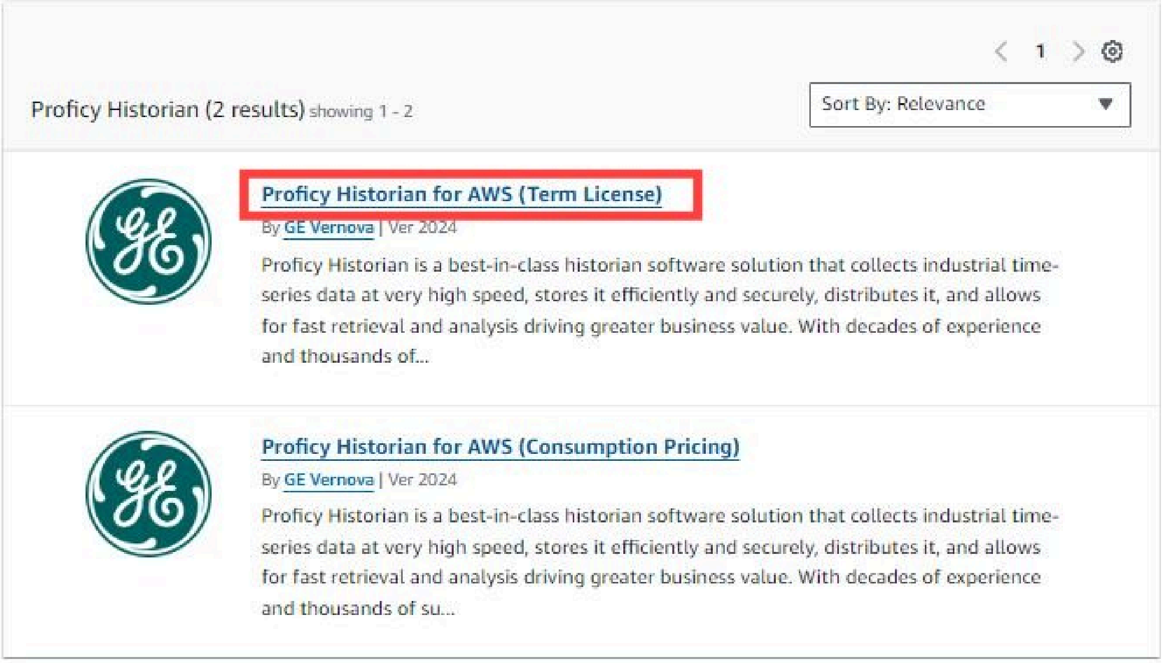
- Asia Pacific (Hong Kong)
- Asia Pacific (Tokyo)
- Asia Pacific (Seoul)
- Asia Pacific (Mumbai)
- Asia Pacific (Singapore)
- Asia Pacific (Sydney)
- Canada (Central)
- Europe (Frankfurt)
- Europe (Stockholm)
- Europe (Ireland)
- Europe (London)
- Europe (Paris)
- Middle East (Bahrain)
- South America (Sao Paulo)

- US East (N. Virginia)
- US East (Ohio)
- US West (N. California)
- US West (Oregon)

Note:
By default, each region can contain up to five VPCs. Therefore, if the region in which you want to deploy Proficy Historian for AWS has already reached the limit, you can deploy it in an existing VPC *(on page 50)*. Or you can choose a different region.

This topic describes how to deploy Proficy Historian for AWS in a new VPC. Alternatively, you can deploy Proficy Historian for AWS in an existing VPC *(on page 50)*.

1. Log in to the AWS marketplace.
2. Search for Proficy Historian.
3. In the list of products that appear, if you have the Historian for Linux license, select **Proficy Historian for AWS**.



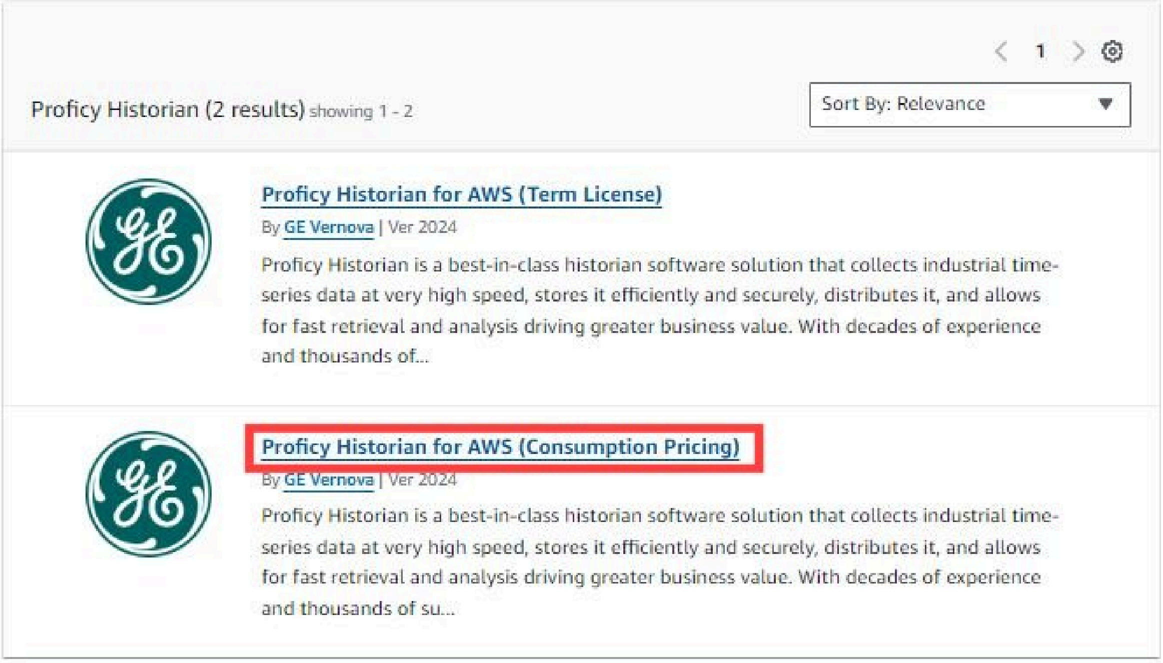
Proficy Historian (2 results) showing 1 - 2

Sort By: Relevance

Proficy Historian for AWS (Term License)
By GE Vernova | Ver 2024
Proficy Historian is a best-in-class historian software solution that collects industrial time-series data at very high speed, stores it efficiently and securely, distributes it, and allows for fast retrieval and analysis driving greater business value. With decades of experience and thousands of...

Proficy Historian for AWS (Consumption Pricing)
By GE Vernova | Ver 2024
Proficy Historian is a best-in-class historian software solution that collects industrial time-series data at very high speed, stores it efficiently and securely, distributes it, and allows for fast retrieval and analysis driving greater business value. With decades of experience and thousands of su...

Or, if you want to use the consumption model, select **Proficy Historian for AWS (Consumption Pricing)**.


Proficy Historian (2 results) showing 1 - 2

Sort By: Relevance

Proficy Historian for AWS (Term License)

By GE Vernova | Ver 2024

Proficy Historian is a best-in-class historian software solution that collects industrial time-series data at very high speed, stores it efficiently and securely, distributes it, and allows for fast retrieval and analysis driving greater business value. With decades of experience and thousands of...

Proficy Historian for AWS (Consumption Pricing)

By GE Vernova | Ver 2024

Proficy Historian is a best-in-class historian software solution that collects industrial time-series data at very high speed, stores it efficiently and securely, distributes it, and allows for fast retrieval and analysis driving greater business value. With decades of experience and thousands of su...

4. Select **Continue to Subscribe**.

   The terms and conditions appear.
5. Select **Continue to Configuration**.

   The **Configure this software** page appears.
6. Enter values as described in the following table.

Field	Description
Fulfillment option	Select **CloudFormation Template**.
Software version	Select **2024**.

7. Select **Continue to Launch**.

   The **Launch this software** page appears.
8. Under **Deployment template**, select **Deploy Proficy Historian including the VPC Creation**.

   The **Quick create stack** page appears. All the fields in the **Parameters** section are populated automatically.
9. Enter values as described in the following table.

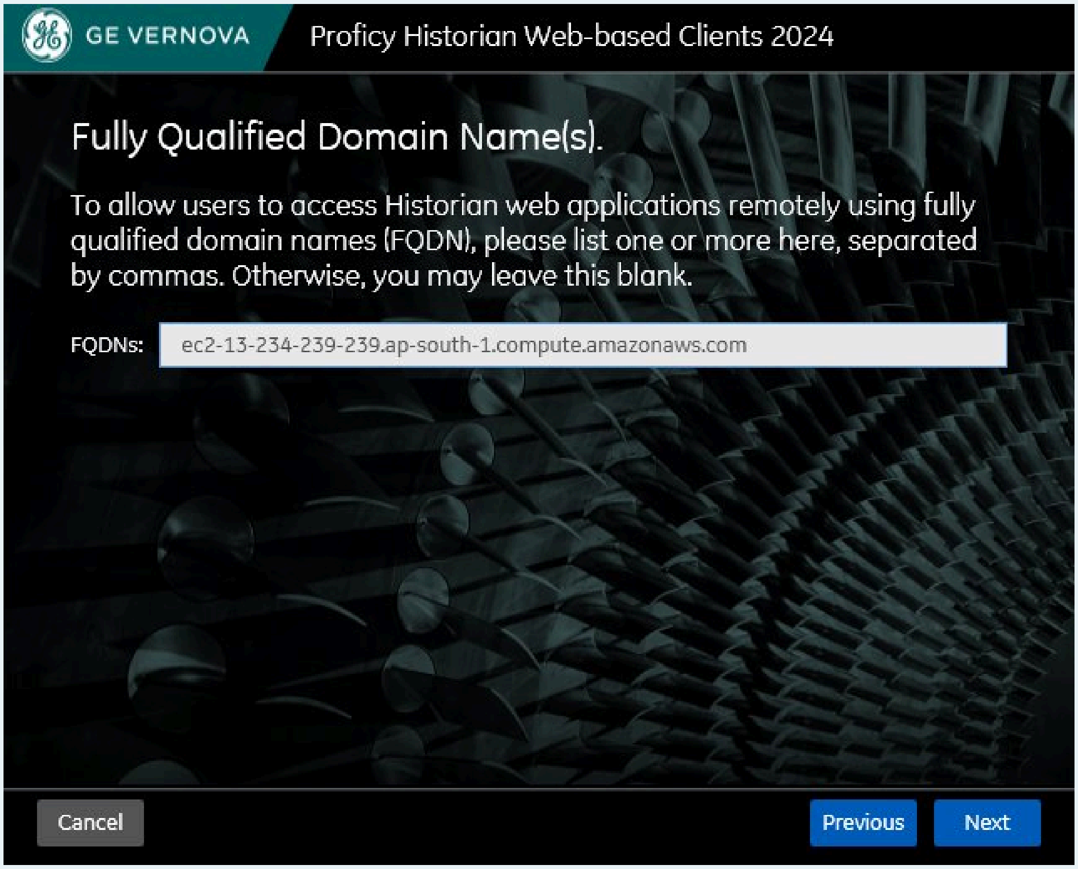
Field/Section	Description
Stack name	Enter a name for the stack. A value is required and must be unique. The value can include all alphanumeric characters and

Field/Section	Description
	dashes. It must begin with an alphabetic character and cannot exceed 128 characters.
New VPC Configuration	Modify values in the section, or leave the default values as is. You can also choose to disable the client VPN endpoint if needed.
EKS cluster name	Enter a unique name for the Elastic Kubernetes Service (EKS) cluster. A value is required, must be unique, and must be less than 28 characters.
Instance type	Choose an instance type based on your requirement. Our benchmarking results suggest that M5.Xlarge supports up to 15 million samples per minute. You can choose an instance of lower or higher capacity based on the rate of collection.
TLS Configuration	Provide the ARN of an SSL certificate. If you leave this field blank, we generate an openSSL certificate. However, we recommend that you provide a trusted certificate.
Use Existing Proficy Auth and Proficy Config Hub	Select Yes to use an existing instance of Proficy Authentication, and enter the URL below. Select No if you want to create a new instance of Proficy Authentication.
URL of Proficy Authentication	Provide the existing Proficy Authentication URL in the following format: https://<machinename>:443. For example: `htttps://ec2-15-237-65-143.eu-west-3.compute.amazonaws.com:443`
Proficy Authentication Admin user secret	Provide the admin user secret for Proficy Authentication. If using an existing Proficy Authentication, specify the admin user secret for the existing Proficy Authentication.
Historian Admin User Password	Provide a password to use as the admin secret and password for default users. The password must contain a minimum of eight characters, at least one uppercase letter, one lowercase letter, one number, and one special character. The password must start with a letter.
Config Hub Admin User Password	Provide the Configuration Hub user password. If not using an existing Configuration Hub instance, a password is required.

Field/Section	Description
CloudWatch Logging	By default, the option to send logs to CloudWatch is enabled. This will allow you to send Data Archiver logs to CloudWatch; you can later access the logs *(on page 1533)* and monitor them. And, the retention period is set to 30 days, after which the logs are deleted. If needed, you can change the retention period. We strongly recommend that you do not disable logging.

10. If needed, you can choose to enable the client VPN endpoint for additional security.
11. Select the check boxes to acknowledge the capabilities required by CloudFormation, and then select **Create stack**.

> **Note:**
> If you want to use windows Proficy Authentication with Historian on AWS 2024, also be sure to do the following:
>
> - Ensure that Windows VM or computer where Proficy Authentication is installed is accessible from the Internet.
> - When installing Historian Web Based Clients from the install media (ISO), specify the FQDN (Fully Qualified Domain Name). The following figure shows an example of how to specify the FQDN of a Windows EC2 machine.



Fully Qualified Domain Name(s).

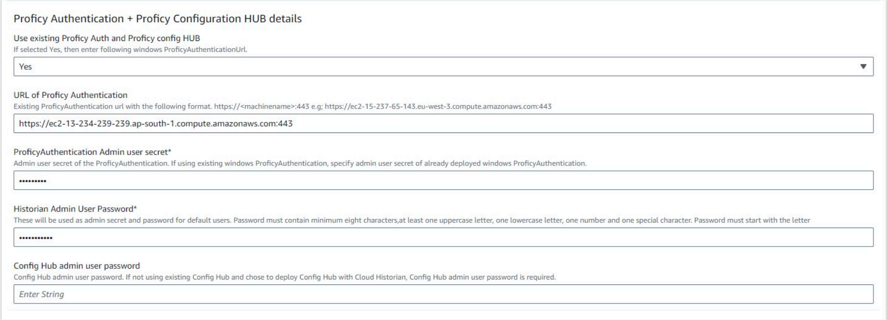
To allow users to access Historian web applications remotely using fully qualified domain names (FQDN), please list one or more here, separated by commas. Otherwise, you may leave this blank.

FQDNs: ec2-13-234-239-239.ap-south-1.compute.amazonaws.com

Be aware that the FQDN should be accessible from the public Internet.

- If the user already installed Proficy Authentication on a Windows computer with Historian Web Based Clients without specifying a FQDN which is accessible from public Internet, then a **re-install without purging databases of Historian Web Based Clients is needed**. Specify public FQDN while re-installing.
- While deploying Historian on AWS, in the CloudFormationTemplate parameters specify the following:
  - Yes for the *'Use existing Proficy Auth and Proficy Config Hub* field
  - In the *URL of Proficy Authentication* area, the FQDN with the format: https://<FQDN>:443

- The '*ProficyAuthentication Admin user*' secret



Proficy Authentication + Proficy Configuration HUB details

Use existing Proficy Auth and Proficy config HUB
If selected Yes, then enter following windows ProficyAuthenticationUrl.
Yes ▼

URL of Proficy Authentication
Existing ProficyAuthentication url with the following format. https://<machinename>:443 e.g; https://ec2-15-237-65-143.eu-west-3.compute.amazonaws.com:443
https://ec2-13-234-239-239.ap-south-1.compute.amazonaws.com:443

ProficyAuthentication Admin user secret*
Admin user secret of the ProficyAuthentication. If using existing windows ProficyAuthentication, specify admin user secret of already deployed windows ProficyAuthentication.
•••••••••

Historian Admin User Password*
These will be used as admin secret and password for default users. Password must contain minimum eight characters,at least one uppercase letter, one lowercase letter, one number and one special character. Password must start with the letter
•••••••••••

Config Hub admin user password
Config Hub admin user password. If not using existing Config Hub and chose to deploy Config Hub with Cloud Historian, Config Hub admin user password is required.
Enter String

The stack is created, along with a VPC with two private and two public subnets.

CAUTION:

Do not update or delete the stack; you can lose data.

1. Apply the Proficy Historian for Linux license if you have one *(on page 56)*.
2. Based on your requirement, install collectors *(on page)*.

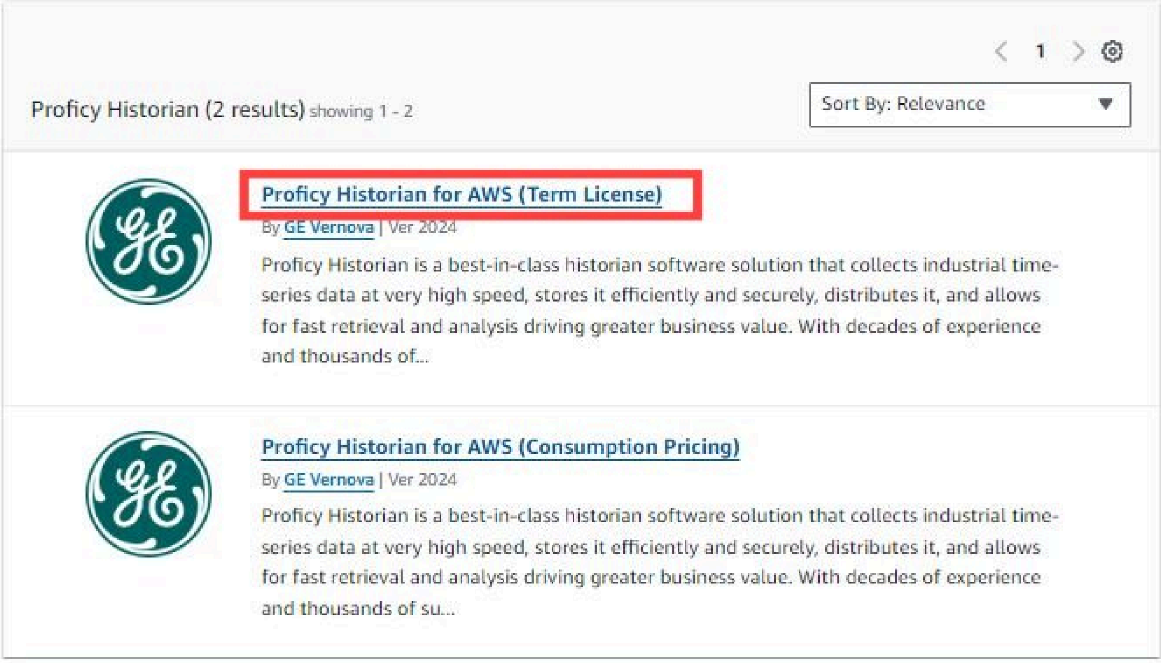
Deploy Proficy Historian for AWS in an Existing VPC

1. Create a virtual private cloud (VPC), and make a note of the VPC configuration (that is, the IDs of the VPC, subnets, and so on). For instructions, refer to https://docs.aws.amazon.com/managedservices/latest/userguide/find-vpc.html and https://docs.aws.amazon.com/managedservices/latest/userguide/find-subnet.html.
2. Ensure that you are in the region in which you want to deploy Proficy Historian for AWS. The following regions are supported:
    - Asia Pacific (Hong Kong)
    - Asia Pacific (Tokyo)
    - Asia Pacific (Seoul)
    - Asia Pacific (Mumbai)
    - Asia Pacific (Singapore)
    - Asia Pacific (Sydney)
    - Canada (Central)
    - Europe (Frankfurt)
    - Europe (Stockholm)
    - Europe (Ireland)
    - Europe (London)

- Europe (Paris)
- Middle East (Bahrain)
- South America (Sao Paulo)
- US East (N. Virginia)
- US East (Ohio)
- US West (N. California)
- US West (Oregon)

This topic describes how to deploy Proficy Historian for AWS in an existing VPC. Alternatively, you can deploy Proficy Historian for AWS in a new VPC *(on page 44)*.

1. Log in to the AWS marketplace.
2. In the list of products that appear, if you have the Historian for Linux license, select **Proficy Historian for AWS**.



Proficy Historian (2 results) showing 1 - 2

Sort By: Relevance

Proficy Historian for AWS (Term License)

By GE Vernova | Ver 2024

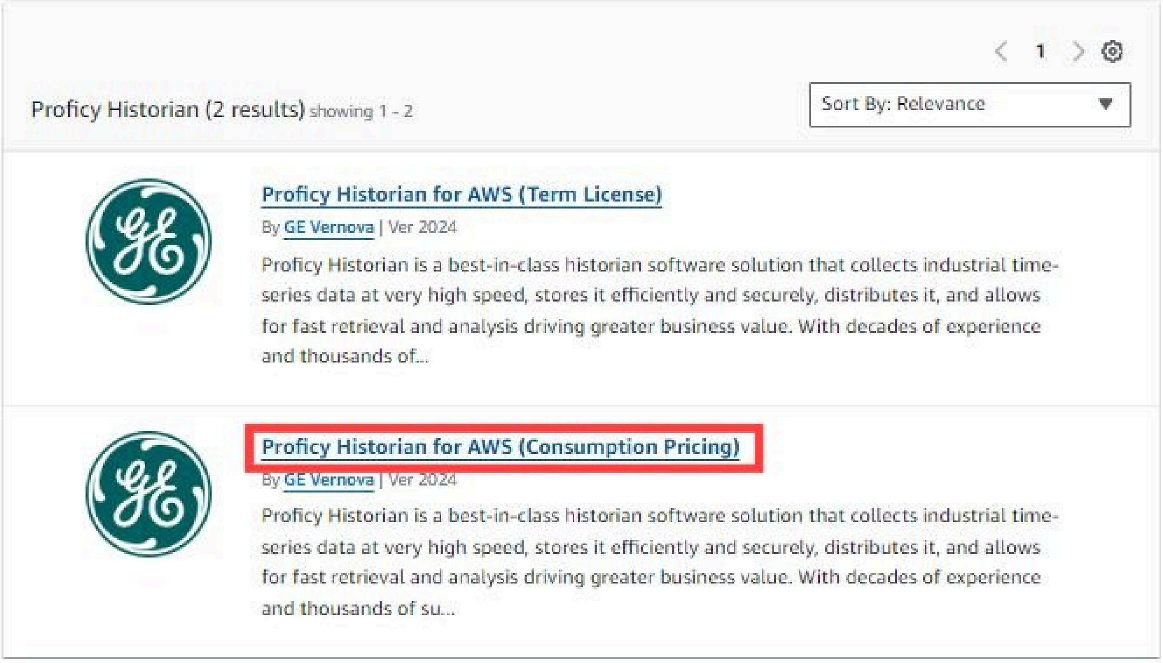
Proficy Historian is a best-in-class historian software solution that collects industrial time-series data at very high speed, stores it efficiently and securely, distributes it, and allows for fast retrieval and analysis driving greater business value. With decades of experience and thousands of...

Proficy Historian for AWS (Consumption Pricing)

By GE Vernova | Ver 2024

Proficy Historian is a best-in-class historian software solution that collects industrial time-series data at very high speed, stores it efficiently and securely, distributes it, and allows for fast retrieval and analysis driving greater business value. With decades of experience and thousands of su...

Or, if you want to use the consumption model, select **Proficy Historian for AWS (Consumption Pricing)**.



Proficy Historian (2 results) showing 1 - 2

Proficy Historian for AWS (Term License)

By GE Vernova | Ver 2024

Proficy Historian is a best-in-class historian software solution that collects industrial time-series data at very high speed, stores it efficiently and securely, distributes it, and allows for fast retrieval and analysis driving greater business value. With decades of experience and thousands of...

Proficy Historian for AWS (Consumption Pricing)

By GE Vernova | Ver 2024

Proficy Historian is a best-in-class historian software solution that collects industrial time-series data at very high speed, stores it efficiently and securely, distributes it, and allows for fast retrieval and analysis driving greater business value. With decades of experience and thousands of su...

3. Select **Continue to Subscribe**.

   The terms and conditions appear.
4. Select **Continue to Configuration**.

   The **Configure this software** page appears.
5. Enter values as described in the following table.

Field	Description
Fulfillment option	Select **CloudFormation Template**.
Software version	Select **2024**

6. Select **Continue to Launch**.

   The **Launch this software** page appears.
7. Under **Deployment template**, select **Deploy Proficy Historian in the Existing VPC**.

   The **Quick create stack** page appears.
8. Enter values as described in the following table.

Field/Section	Description
Stack name	Enter a name for the stack. A value is required and must be unique. The value can include all alphanumeric characters and dashes. It must begin with an alphabetic character and cannot exceed 128 characters.

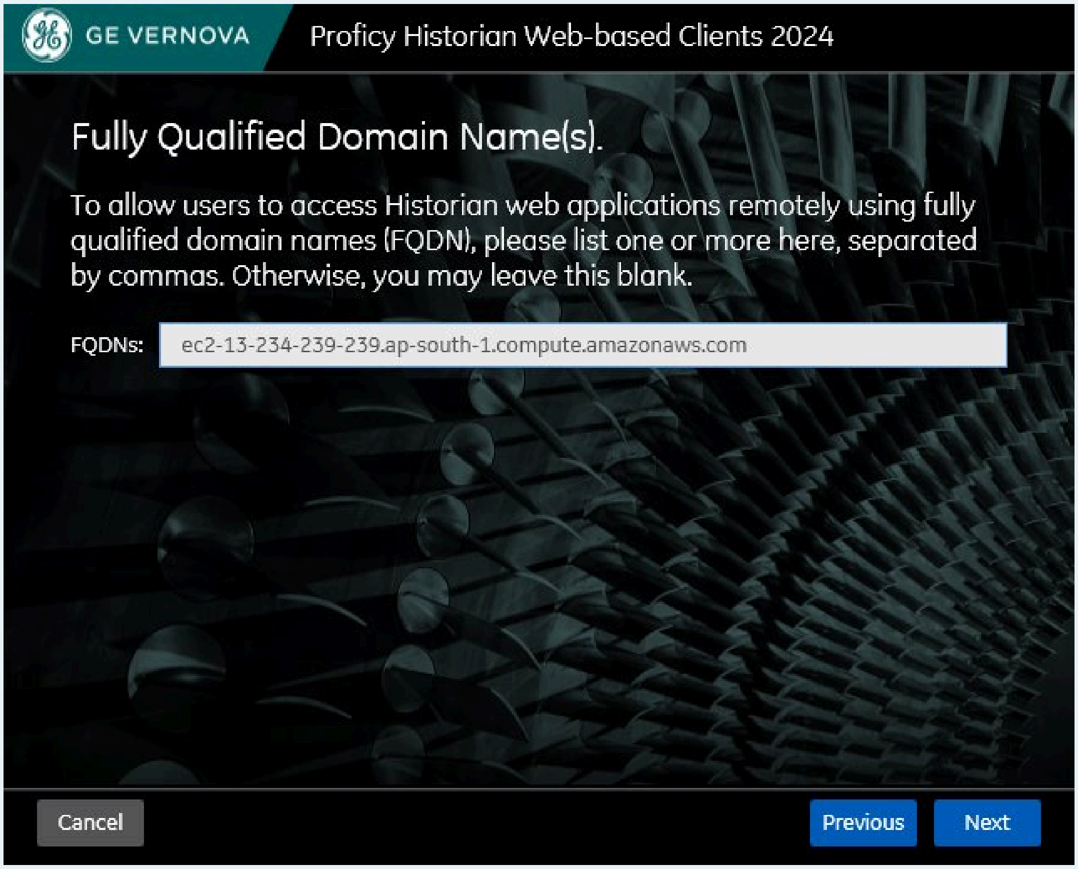
Field/Section	Description
VPC ID	Enter the ID of the VPC on which you want to deploy Proficy Historian for AWS. For instructions on how to the find the VPC ID, refer to https://docs.aws.amazon.com/managedservices/latest/userguide/find-vpc.html.
Private Subnets and Public Subnets IDs	Enter the IDs of the two private and the two public subnets in your VPC. For instructions on how to find these IDs, refer to https://docs.aws.amazon.com/managedservices/latest/userguide/find-subnet.html.
EKS cluster name	Enter a unique name for the Elastic Kubernetes Service (EKS) cluster. A value is required, must be unique, and must be less than 28 characters.
Instance type	Choose an instance type based on your requirement. Our benchmarking results suggest that M5.Xlarge supports up to 15 million samples per minute. You can choose an instance of lower or higher capacity based on the rate of collection.
TLS Configuration	Provide the ARN of an SSL certificate. If you leave this field blank, we generate an openSSL certificate. However, we recommend that you provide a trusted certificate.
Use Existing Proficy Auth and Proficy Config Hub	Select Yes to use an existing instance of Proficy Authentication, and enter the URL below. Select No if you want to create a new instance of Proficy Authentication.
URL of Proficy Authentication	Provide the existing Proficy Authentication URL in the following format: https://<machinename>:443. For example: `https://ec2-15-237-65-143.eu-west-3.compute.amazonaws.com:443`
Proficy Authentication Admin user secret	Provide the admin user secret for Proficy Authentication. If using an existing Proficy Authentication, specify the admin user secret for the existing Proficy Authentication.
Historian Admin User Password	Provide a password to use as the admin secret and password for default users. The password must contain a minimum of eight characters, at least one uppercase letter, one lowercase letter, one number, and one special character. The password must start with a letter.

Field/Section	Description
Config Hub Admin User Password	Provide the Configuration Hub user password. If not using an existing Configuration Hub instance, a password is required.
CloudWatch Logging	By default, the option to send logs to CloudWatch is enabled. This will allow you to send Data Archiver logs to CloudWatch; you can later access the logs *(on page 1533)* and monitor them. And, the retention period is set to 30 days, after which the logs are deleted. If needed, you can change the retention period. We strongly recommend that you do not disable logging.

9. If needed, you can choose to enable the client VPN endpoint for additional security.
10. Select the check boxes to acknowledge the capabilities required by CloudFormation, and then select **Create stack**.

Note:
If you want to use windows Proficy Authentication with Historian on AWS 2024, also be sure to do the following:

- Ensure that Windows VM or computer where Proficy Authentication is installed is accessible from the Internet.
- When installing Historian Web Based Clients from the install media (ISO), specify the FQDN (Fully Qualified Domain Name). The following figure shows an example of how to specify the FQDN of a Windows EC2 machine.



Fully Qualified Domain Name(s).

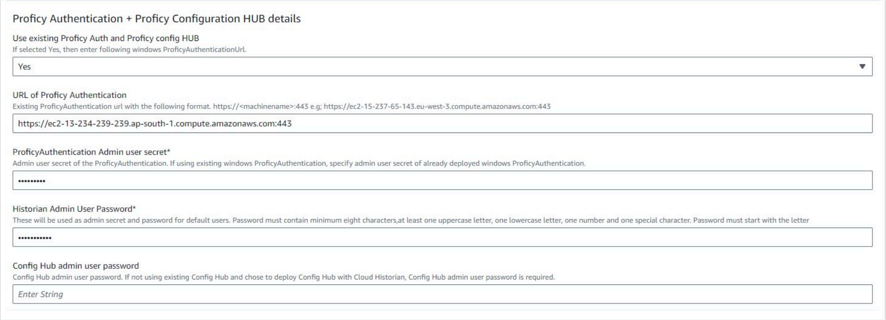
To allow users to access Historian web applications remotely using fully qualified domain names (FQDN), please list one or more here, separated by commas. Otherwise, you may leave this blank.

FQDNs: ec2-13-234-239-239.ap-south-1.compute.amazonaws.com

Be aware that the FQDN should be accessible from the public Internet.

- If the user already installed Proficy Authentication on a Windows computer with Historian Web Based Clients without specifying a FQDN which is accessible from public Internet, then a **re-install without purging databases of Historian Web Based Clients is needed**. Specify public FQDN while re-installing.
- While deploying Historian on AWS, in the CloudFormationTemplate parameters specify the following:
  - Yes for the *'Use existing Proficy Auth and Proficy Config Hub* field
  - In the *URL of Proficy Authentication* area, the FQDN with the format: https://<FQDN>:443

- The '*ProficyAuthentication Admin user*' secret



Proficy Authentication + Proficy Configuration HUB details

Use existing Proficy Auth and Proficy config HUB
If selected Yes, then enter following windows ProficyAuthenticationUrl.
Yes

URL of Proficy Authentication
Existing ProficyAuthentication url with the following format. https://<machinename>:443 e.g; https://ec2-15-237-65-143.eu-west-3.compute.amazonaws.com:443
https://ec2-13-234-239-239.ap-south-1.compute.amazonaws.com:443

ProficyAuthentication Admin user secret*
Admin user secret of the ProficyAuthentication. If using existing windows ProficyAuthentication, specify admin user secret of already deployed windows ProficyAuthentication.
•••••••••

Historian Admin User Password*
These will be used as admin secret and password for default users. Password must contain minimum eight characters,at least one uppercase letter, one lowercase letter, one number and one special character. Password must start with the letter
•••••••••••

Config Hub admin user password
Config Hub admin user password. If not using existing Config Hub and chose to deploy Config Hub with Cloud Historian, Config Hub admin user password is required.
Enter String

The stack is created, along with a VPC with two private and two public subnets.

CAUTION:

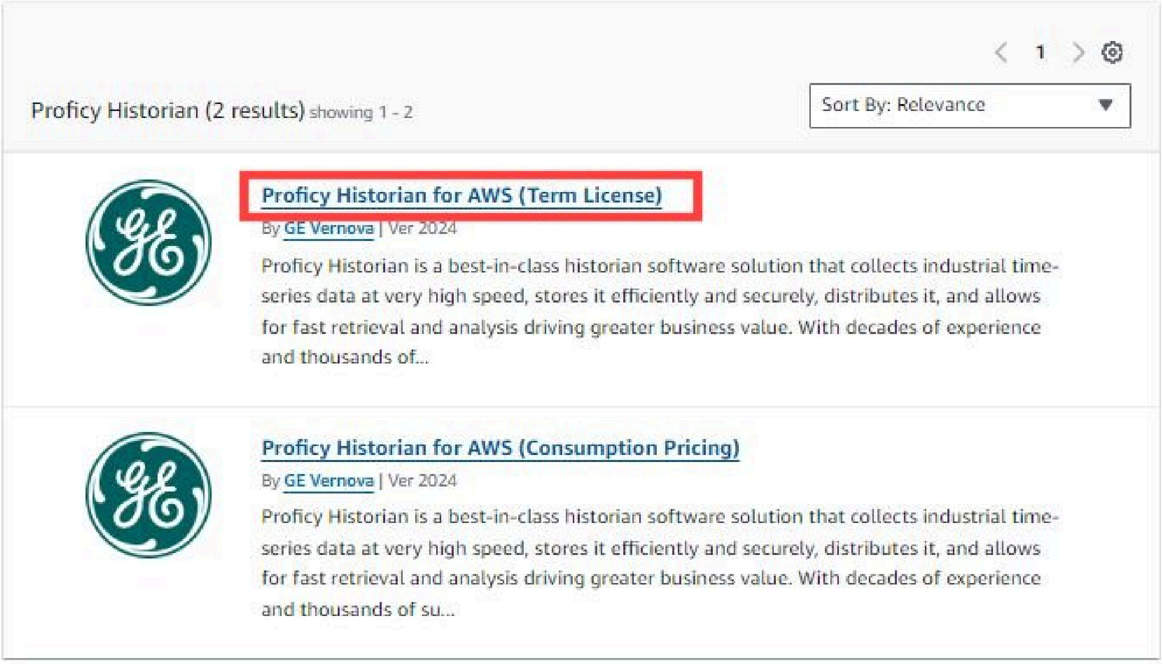
Do not update or delete the stack; you can lose data.

1. Apply the Proficy Historian for Linux license if you have one *(on page 56)*.
2. Based on your requirement, install collectors *(on page)*.

Applying the Historian License (BYOL)

- Upload the Proficy Historian for Linux license in Amazon S3.
- Deploy Proficy Historian for AWS *(on page 43)*.

1. Log in to the AWS marketplace. Ensure that you are in the same region in which you have deployed Proficy Historian for AWS.
2. Search for Proficy Historian.
3. In the list of products that appear, select **Proficy Historian for AWS**.



Proficy Historian (2 results) showing 1 - 2

Proficy Historian for AWS (Term License)

By GE Vernova | Ver 2024

Proficy Historian is a best-in-class historian software solution that collects industrial time-series data at very high speed, stores it efficiently and securely, distributes it, and allows for fast retrieval and analysis driving greater business value. With decades of experience and thousands of...

Proficy Historian for AWS (Consumption Pricing)

By GE Vernova | Ver 2024

Proficy Historian is a best-in-class historian software solution that collects industrial time-series data at very high speed, stores it efficiently and securely, distributes it, and allows for fast retrieval and analysis driving greater business value. With decades of experience and thousands of su...

4. Select **Continue to Subscribe**.

   The terms and conditions appear.
5. Select **Continue to Configuration**.

   The **Configure this software** page appears.
6. Enter values as described in the following table.

Field	Description
Fulfillment option	Select **CloudFormation Template**.
Software version	Select **2024**.

7. Select **Continue to Launch**.
8. Under **Deployment template**, select **Proficy Historian License - BYOL**.

   The **Quick create stack** page appears.
9. Enter values as described in the following table.

Field/Section	Description
Stack name	Enter a name for the stack. A value is required and must be unique. The value can include all alphanumeric characters and dashes. It must begin with an alphabetic character and cannot exceed 128 characters.

Field/Section	Description
EKS cluster name	Enter the EKS cluster name in which you deployed Proficy Historian for AWS. A value is required. **Tip:** You can find it in the list of EKS clusters.
S3 URL	Enter the URL of the Historian license that you have uploaded in Amazon S3. A value is required.

10. Select **Create stack**.
    The license is applied.

Based on your requirement, install collectors *(on page)* or the Web Admin console and the REST Query service *(on page)*.

Upgrading

Upgrade to Cloud Historian 2024

Before proceeding to upgrade, it is recommended that you to take a backup of the Historian archive files and the .ihc configuration file. For more information, refer to the Access Archives *(on page 1540)* topic.

This topic describes how to upgrade to the latest version of Cloud Historian. Be aware that Direct Upgrades from a version prior to 2023 are not supported. When upgrading from any version prior to 2023, you first need to upgrade to any of the 2023 versions (2023, 2023.1, 2023.1.1), and then upgrade to 2024.

> **Note:**
> Be aware that AWS introduced NLB with security and this change has been incorporated with the new deployments but for the existing ones (before 2024 released versions), the NLB will not be replaced with upgrade. If you need NLB with security for existing deployments, then you need to manually delete the NLB, perform the data migration, and then the DNS will change.

1. Access the **AWS Management Console** page.
2. Search for **Proficy Historian**.
3. In the list of products that appear, if you have the Historian for Linux license, select **Proficy Historian for AWS**. Or, if you want to use the consumption model, select **Proficy Historian for AWS (Consumption Pricing)**.

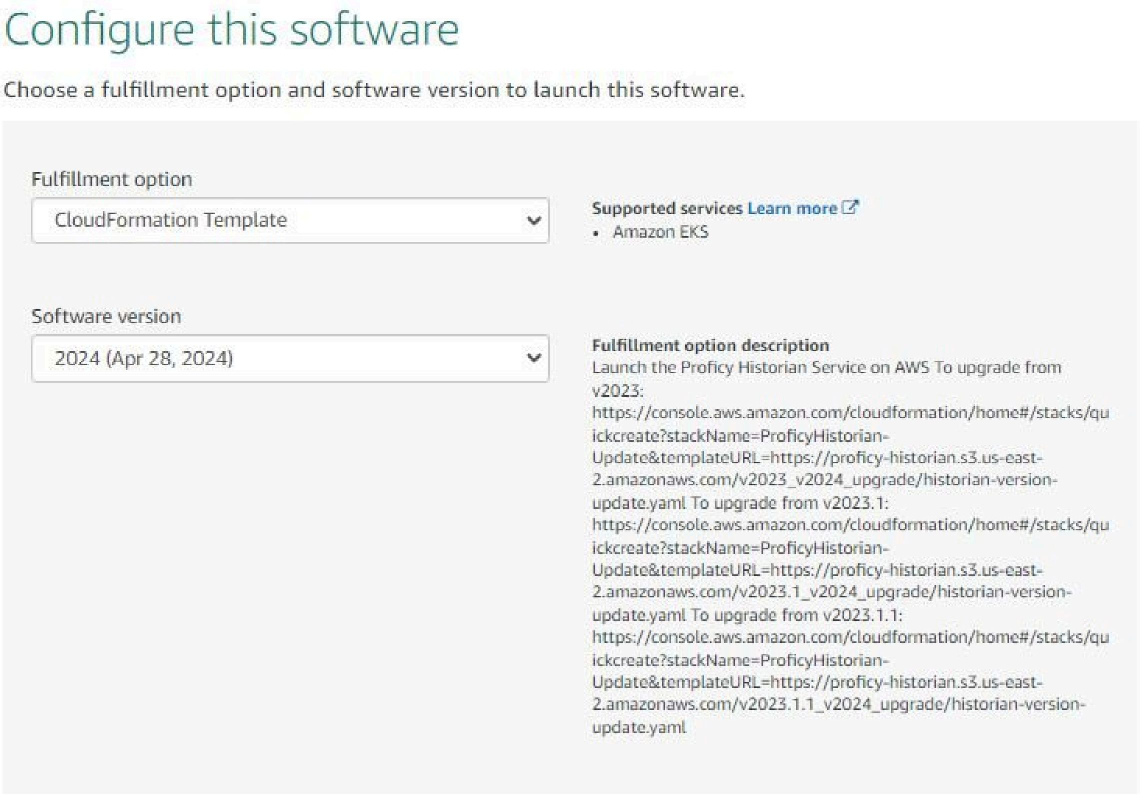
4. Select **Continue to Subscribe**.

   The terms and conditions appear.
5. Select **Continue to Configuration**.

   The Configure this software page appears.
6. Enter values as described in the following table.

Field	Select...
Fulfillment Option	CloudFormation Template
Softeware version	2024

7. In the **Fulfillment option description**, copy the upgrade link, and then access the link from a web browser.



Configure this software

Choose a fulfillment option and software version to launch this software.

Fulfillment option: CloudFormation Template

Software version: 2024 (Apr 28, 2024)

Supported services Learn more

- Amazon EKS

Fulfillment option description

Launch the Proficy Historian Service on AWS To upgrade from v2023: https://console.aws.amazon.com/cloudformation/home#/stacks/quickcreate?stackName=ProficyHistorian-Update&templateURL=https://proficy-historian.s3.us-east-2.amazonaws.com/v2023_v2024_upgrade/historian-version-update.yaml To upgrade from v2023.1: https://console.aws.amazon.com/cloudformation/home#/stacks/quickcreate?stackName=ProficyHistorian-Update&templateURL=https://proficy-historian.s3.us-east-2.amazonaws.com/v2023.1_v2024_upgrade/historian-version-update.yaml To upgrade from v2023.1.1: https://console.aws.amazon.com/cloudformation/home#/stacks/quickcreate?stackName=ProficyHistorian-Update&templateURL=https://proficy-historian.s3.us-east-2.amazonaws.com/v2023.1.1_v2024_upgrade/historian-version-update.yaml

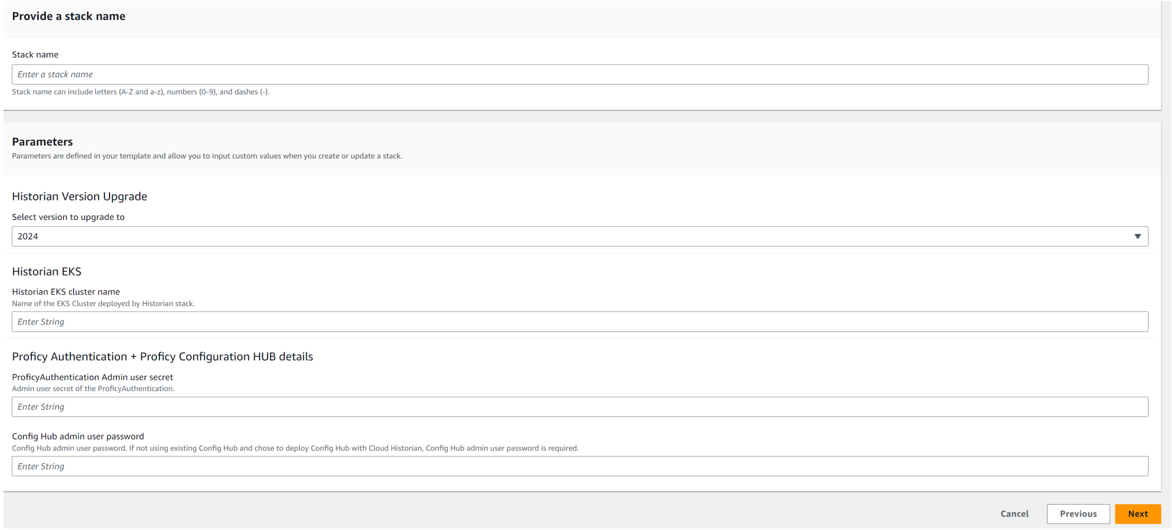
Link to upgrade from v2023: https://console.aws.amazon.com/cloudformation/home#/stacks/quickcreate?stackName=ProficyHistorian-Update&templateURL=https://proficy-historian.s3.us-east-2.amazonaws.com/v2023_v2024_upgrade/historian-version-update.yaml

Link to upgrade from v2023.1: https://console.aws.amazon.com/cloudformation/home#/stacks/quickcreate?stackName=ProficyHistorian-Update&templateURL=https://proficy-historian.s3.us-east-2.amazonaws.com/v2023.1_v2024_upgrade/historian-version-update.yaml

Link to upgrade from v2023.1.1: https://console.aws.amazon.com/cloudformation/home#/stacks/quickcreate?stackName=ProficyHistorian-Update&templateURL=https://proficy-historian.s3.us-east-2.amazonaws.com/v2023.1.1_v2024_upgrade/historian-version-update.yaml

8. Provide the information as shown in the following table and figure.

Field	Description
Stack Name	Enter the Historian server name.
Historian Version Upgrade	Enter the version to upgrade to. For instance: 2024.
Historian EKS	Enter the name of the EKS Cluster deployed by the Historian stack.
Proficy Authentication Admin User Secret	The password for the Proficy Authentication admin user.
Config Hub Admin User Password	The password for the Configuration Hub admin user.



Provide a stack name

Stack name

Enter a stack name

Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-).

Parameters

Parameters are defined in your template and allow you to input custom values when you create or update a stack.

Historian Version Upgrade

Select version to upgrade to

2024

Historian EKS

Historian EKS cluster name

Name of the EKS Cluster deployed by Historian stack.

Enter String

Proficy Authentication + Proficy Configuration HUB details

ProficyAuthentication Admin user secret

Admin user secret of the ProficyAuthentication.

Enter String

Config Hub admin user password

Config Hub admin user password. If not using existing Config Hub and chose to deploy Config Hub with Cloud Historian, Config Hub admin user password is required.

Enter String

9. Acknowledge and create stack.

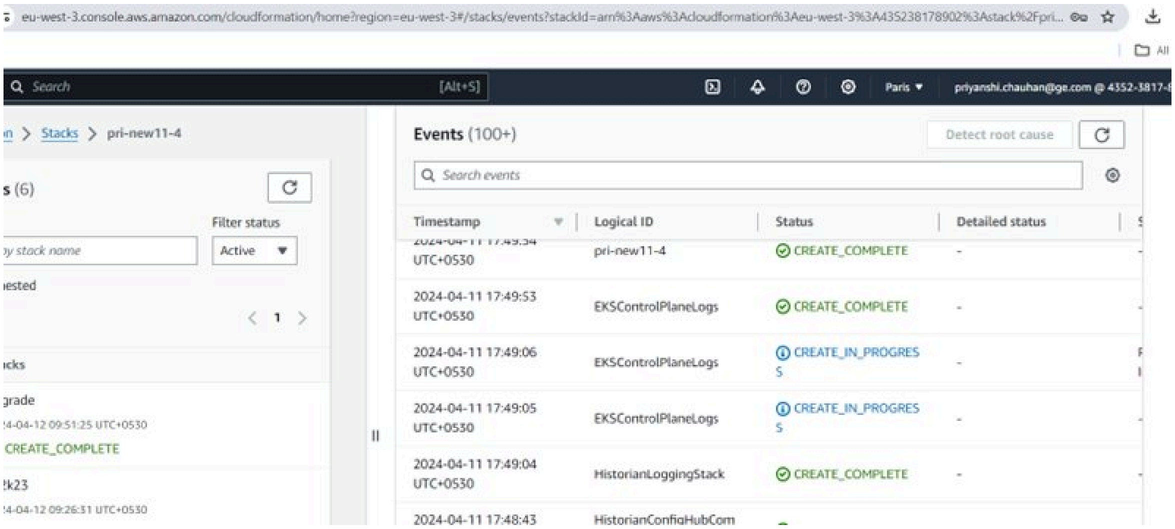
> **Note:**
> Please be aware that the upgrade to 2024 will take approximately 30 minutes, as the following changes are occurring with the upgrade:
>
> - Updating Cloud Foundry UAA to Proficy Authentication which also includes migrating the users data.
> - Deploying the Configuration Hub containers.

10. Perform post upgrade steps:
    - If the write rate is greater than 3MM Samples/minute, change the EC2 instance type to **M5.Xlarge**. See Change EC2 Instance Type for an Existing Deployment *(on page 61)*. Also, be sure to reference the Sizing recommendations in the Hardware Requirements *(on page 33)* topic.
    - If you notice any performance issues, update your write threads to 4 (if it is <4) using the **Advanced Options** of Server from the Configuration Hub.

Change EC2 Instance Type for an Existing Deployment

This topic describes how to change the EC2 instance type for an existing deployment.

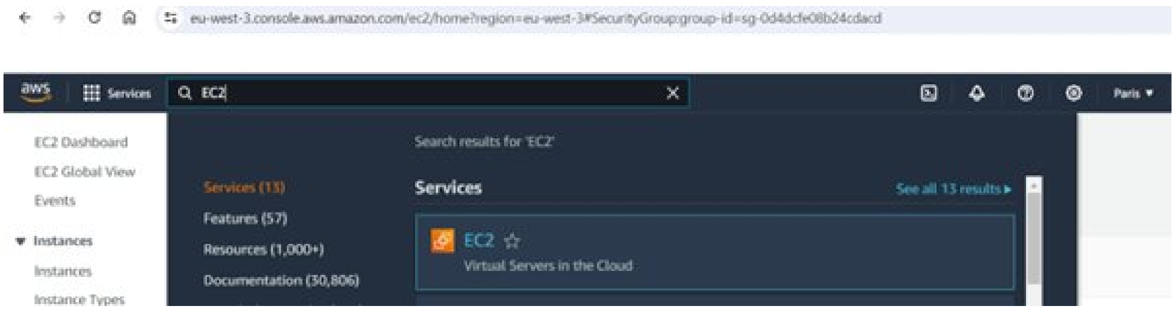
1. Access the **AWS Management Console** page.
2. Locate the new deployment in the CloudFormation Stack, as shown in the following figure.



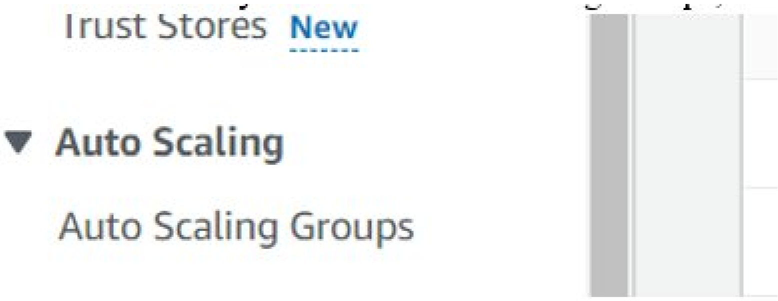
eu-west-3.console.aws.amazon.com/cloudformation/home?region=eu-west-3#/stacks/events?stackId=arn%3Aaws%3Acloudformation%3Aeu-west-3%3A435238178902%3Astack%2Fpri...

Timestamp	Logical ID	Status	Detailed status
2024-04-11 17:49:54 UTC+0530	pri-new11-4	CREATE_COMPLETE	-
2024-04-11 17:49:53 UTC+0530	EKSControlPlaneLogs	CREATE_COMPLETE	-
2024-04-11 17:49:06 UTC+0530	EKSControlPlaneLogs	CREATE_IN_PROGRESS	-
2024-04-11 17:49:05 UTC+0530	EKSControlPlaneLogs	CREATE_IN_PROGRESS	-
2024-04-11 17:49:04 UTC+0530	HistorianLoggingStack	CREATE_COMPLETE	-
2024-04-11 17:48:43	HistorianConfigHubCom		

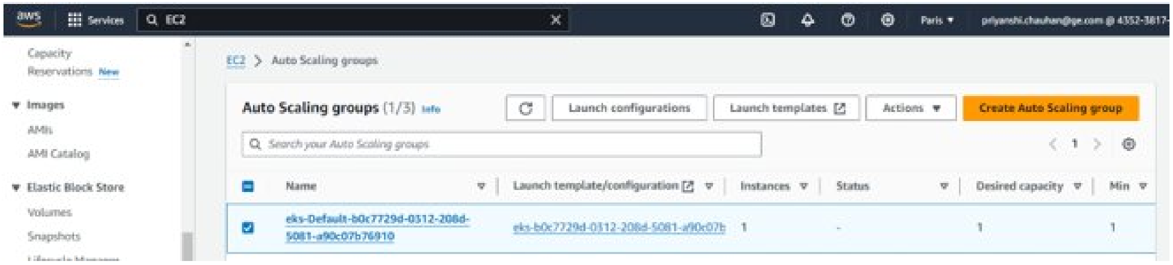
3. Seach for EC2, and select it:



4. On the bottom left of the screen, you will see Auto Scaling Groups; select it.

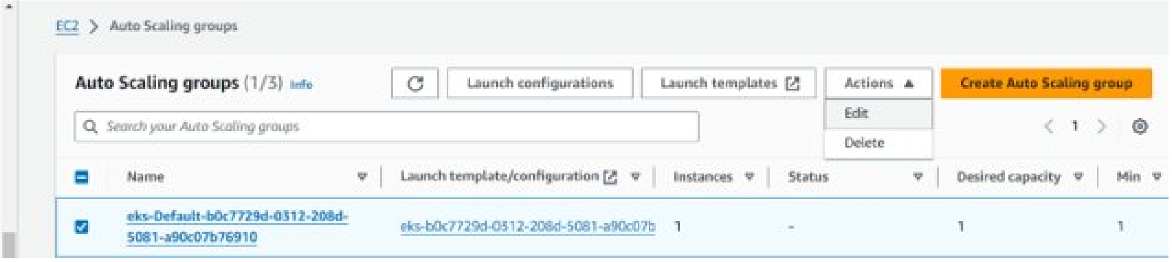


5. Locate the ASG with the value of 1 for the running instance, as shown in the following figure.


aws
Services
EC2
Paris ▾
priyanshi.chauhan@ge.com @ 4352-3817-
Capacity Reservations New
▾ Images
AMIs
AMI Catalog
▾ Elastic Block Store
Volumes
Snapshots
EC2 > Auto Scaling groups
Auto Scaling groups (1/3) Info
Launch configurations
Launch templates
Actions ▾
Create Auto Scaling group
Search your Auto Scaling groups

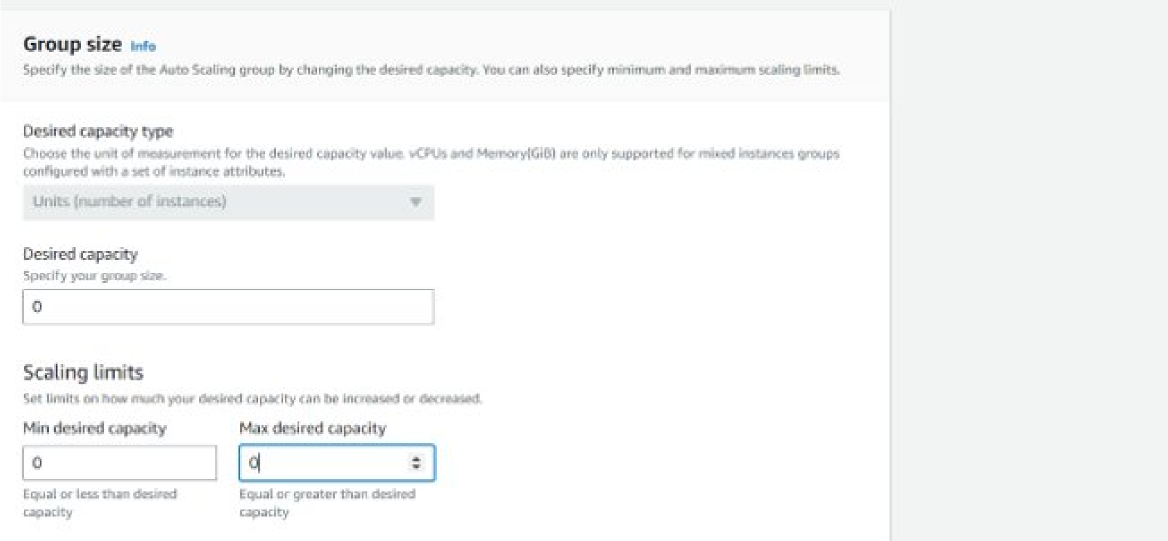
	Name	Launch template/configuration	Instances	Status	Desired capacity	Min
☑	eks-Default-b0c7729d-0312-208d-5081-a90c07b76910	eks-b0c7729d-0312-208d-5081-a90c07b	1	-	1	1

6. Click the **Actions** button, and select **Edit**.


EC2 > Auto Scaling groups
Auto Scaling groups (1/3) Info
Launch configurations
Launch templates
Actions ▴
Edit
Delete
Create Auto Scaling group
Search your Auto Scaling groups

	Name	Launch template/configuration	Instances	Status	Desired capacity	Min
☑	eks-Default-b0c7729d-0312-208d-5081-a90c07b76910	eks-b0c7729d-0312-208d-5081-a90c07b	1	-	1	1

7. In the screen that appears, make sure the **Desired Capacity**, **Min Desired Capacity**, and **Max Desired Capacity** are set to 0.



Group size Info

Specify the size of the Auto Scaling group by changing the desired capacity. You can also specify minimum and maximum scaling limits.

Desired capacity type

Choose the unit of measurement for the desired capacity value. vCPUs and Memory(GiB) are only supported for mixed instances groups configured with a set of instance attributes.

Units (number of instances)

Desired capacity

Specify your group size.

0

Scaling limits

Set limits on how much your desired capacity can be increased or decreased.

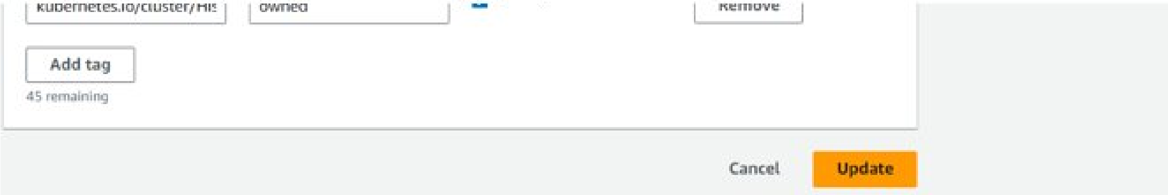
Min desired capacity: 0

Equal or less than desired capacity

Max desired capacity: 0

Equal or greater than desired capacity

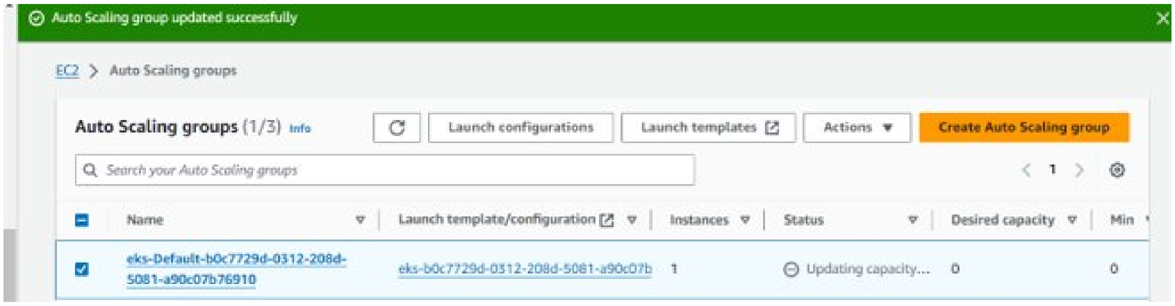
8. Click the **Update** button at the bottom of the screen.



kubernetes.io/cluster/His | owned

45 remaining

9. Observe that the ASG will start the count as 0, as shown in the following figure.



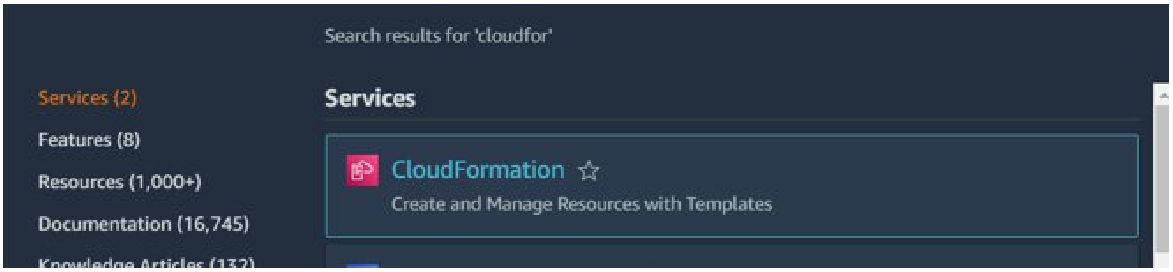
Auto Scaling group updated successfully

EC2 > Auto Scaling groups

Auto Scaling groups (1/3) Info

	Name	Launch template/configuration	Instances	Status	Desired capacity	Min
☑	eks-Default-b0c7729d-0312-208d-5081-a90c07b76910	eks-b0c7729d-0312-208d-5081-a90c07b	1	Updating capacity...	0	0

10. Wait appoximately 10 minutes to make the instance count to 0.
11. After the count completes, search for CloudFormation and then search for the deployed stack.



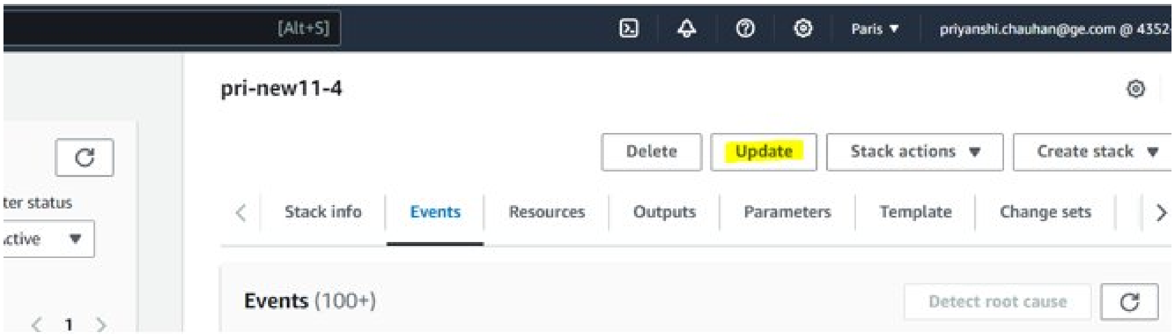
Search results for 'cloudfor'

Services

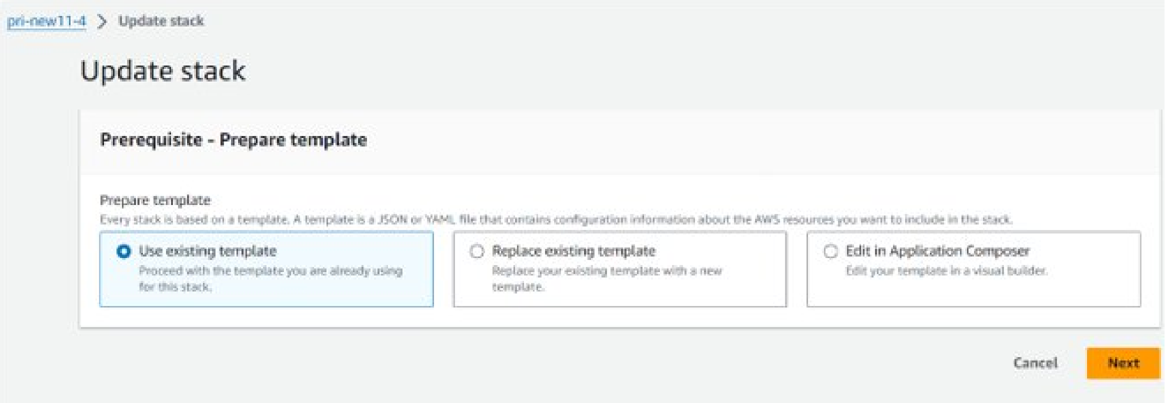
CloudFormation

Create and Manage Resources with Templates

12. On the page for the stack, click the **Update** button located at the top of the stack.



13. On the next screen that appears, select the **Use existing template** option, and then click **Next**.



Prerequisite - Prepare template

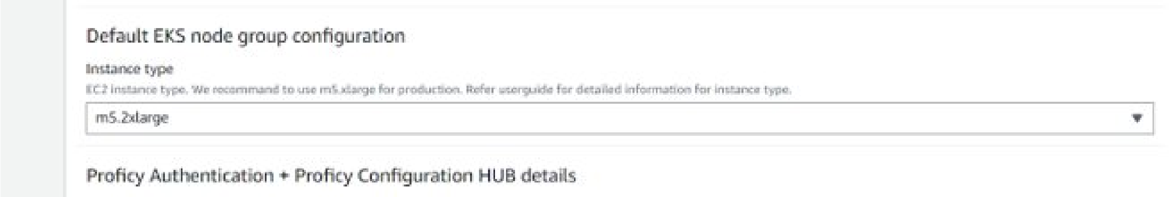
Prepare template
Every stack is based on a template. A template is a JSON or YAML file that contains configuration information about the AWS resources you want to include in the stack.

● Use existing template
Proceed with the template you are already using for this stack.

○ Replace existing template
Replace your existing template with a new template.

○ Edit in Application Composer
Edit your template in a visual builder.

14. On the next screen, change the instance type to whatever EC2 instance type you want.



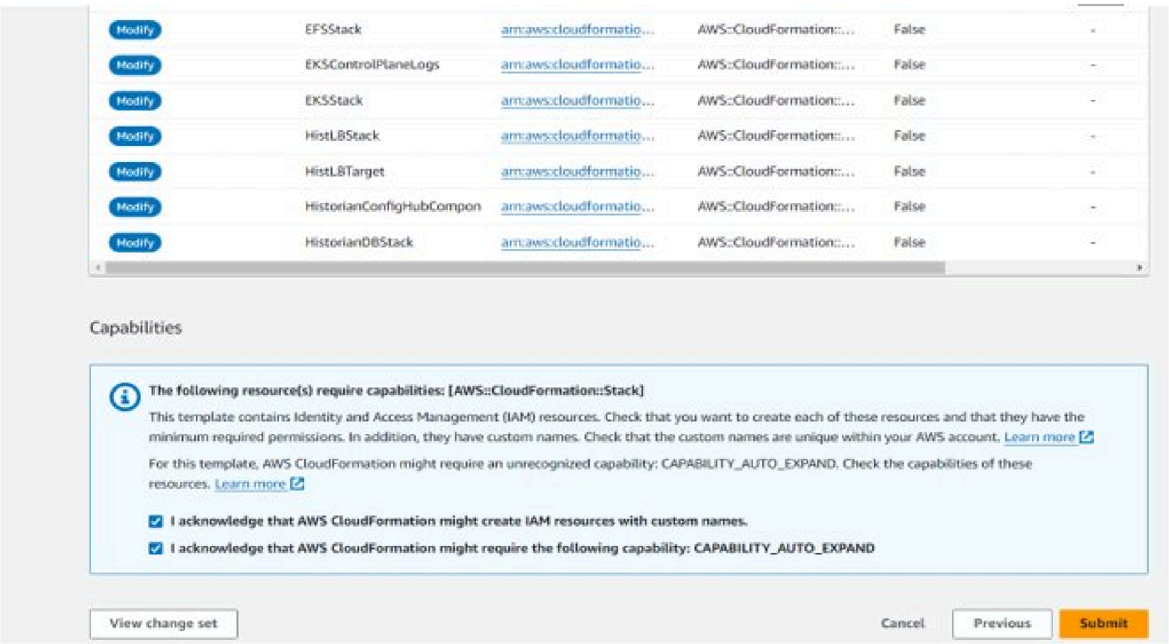
Default EKS node group configuration

Instance type
EC2 instance type. We recommand to use m5.xlarge for production. Refer userguide for detailed information for instance type.
m5.2xlarge ▼

Proficy Authentication + Proficy Configuration HUB details

15. Leave the defaults for all other options, and click **Next**.

16. On the final screen, select the acknowledgements to continue, and then click **Submit**.



Modify	EFSStack	arn:aws:cloudformatio...	AWS::CloudFormation::...	False	-
Modify	EKSControlPlaneLogs	arn:aws:cloudformatio...	AWS::CloudFormation::...	False	-
Modify	EKSStack	arn:aws:cloudformatio...	AWS::CloudFormation::...	False	-
Modify	HistLBStack	arn:aws:cloudformatio...	AWS::CloudFormation::...	False	-
Modify	HistLBTarget	arn:aws:cloudformatio...	AWS::CloudFormation::...	False	-
Modify	HistorianConfigHubCompon	arn:aws:cloudformatio...	AWS::CloudFormation::...	False	-
Modify	HistorianDBStack	arn:aws:cloudformatio...	AWS::CloudFormation::...	False	-

Capabilities

The following resource(s) require capabilities: [AWS::CloudFormation::Stack]

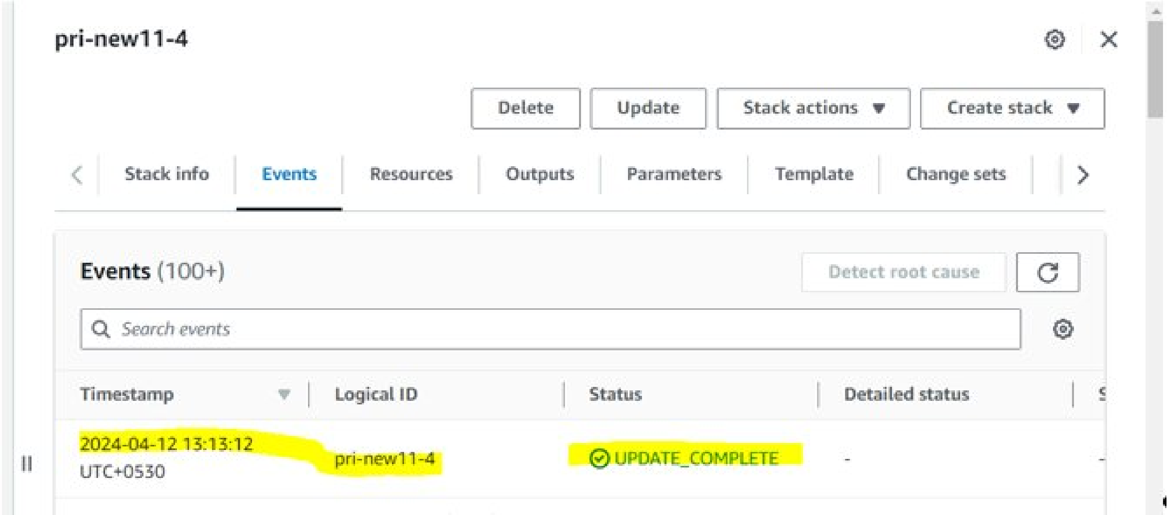
This template contains Identity and Access Management (IAM) resources. Check that you want to create each of these resources and that they have the minimum required permissions. In addition, they have custom names. Check that the custom names are unique within your AWS account. Learn more

For this template, AWS CloudFormation might require an unrecognized capability: CAPABILITY_AUTO_EXPAND. Check the capabilities of these resources. Learn more

☑ **I acknowledge that AWS CloudFormation might create IAM resources with custom names.**

☑ **I acknowledge that AWS CloudFormation might require the following capability: CAPABILITY_AUTO_EXPAND**

17. After the stack update completes, observe the updated status, as shown in the following figure:

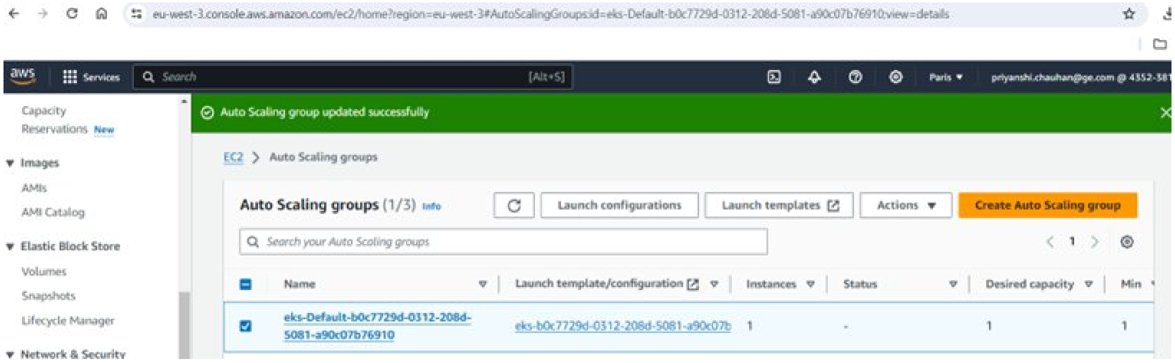


pri-new11-4

Events (100+)

Timestamp	Logical ID	Status	Detailed status
2024-04-12 13:13:12 UTC+0530	pri-new11-4	UPDATE_COMPLETE	-

18. After the update is completed, change the Auto Scaling Groups to 1. From the EC2, click Edit and change the to **Desired Capacity**, **Min Desired Capacity**, and **Max Desired Capacity** 1.



eu-west-3.console.aws.amazon.com/ec2/home?region=eu-west-3#AutoScalingGroups:id=eks-Default-b0c7729d-0312-208d-5081-a90c07b76910;view=details

Auto Scaling group updated successfully

EC2 > Auto Scaling groups

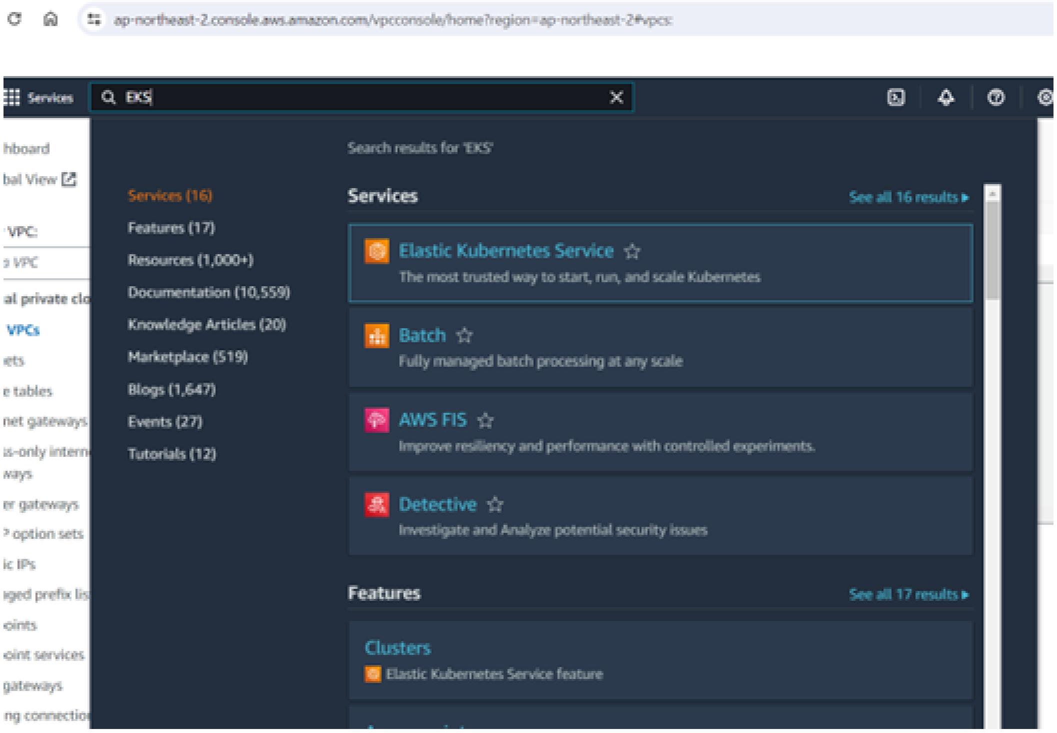
Auto Scaling groups (1/3) Info

	Name	Launch template/configuration	Instances	Status	Desired capacity	Min
☑	eks-Default-b0c7729d-0312-208d-5081-a90c07b76910	eks-b0c7729d-0312-208d-5081-a90c07b	1	-	1	1

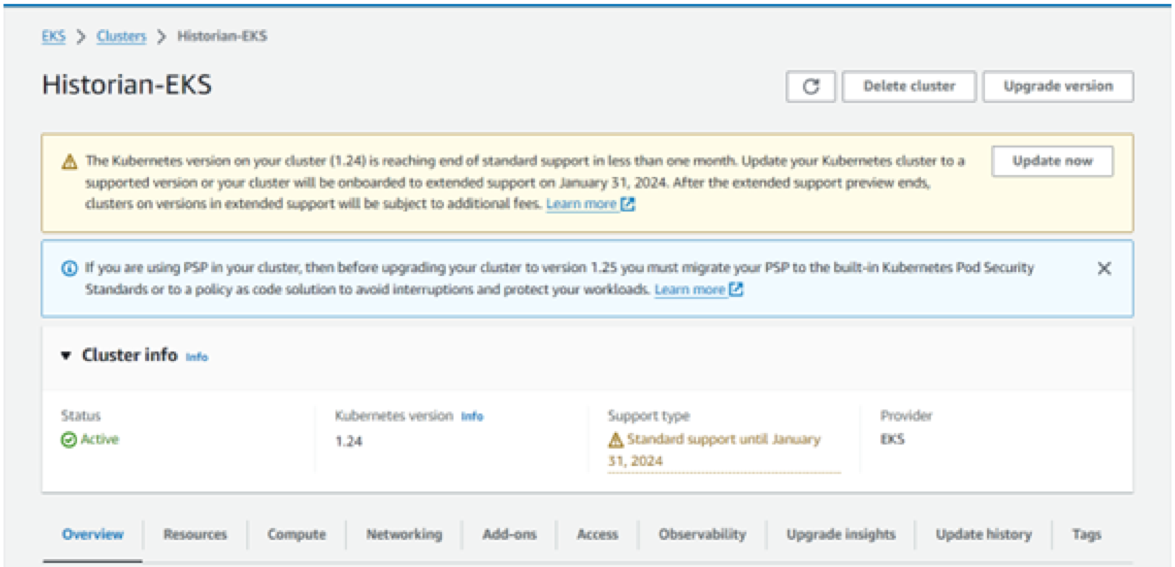
Upgrade an Existing EKS Cluster

This topic describes how to upgrade an existing EKS Cluster. For more information, see: https://docs.aws.amazon.com/eks/latest/userguide/update-cluster.html.

1. On the search bar in the AWS console, type EKS and select Elastic Kubernetes Service as shown in the following figure:



2. In the EKS section, you can see view the cluster eligible for upgrade. For example:

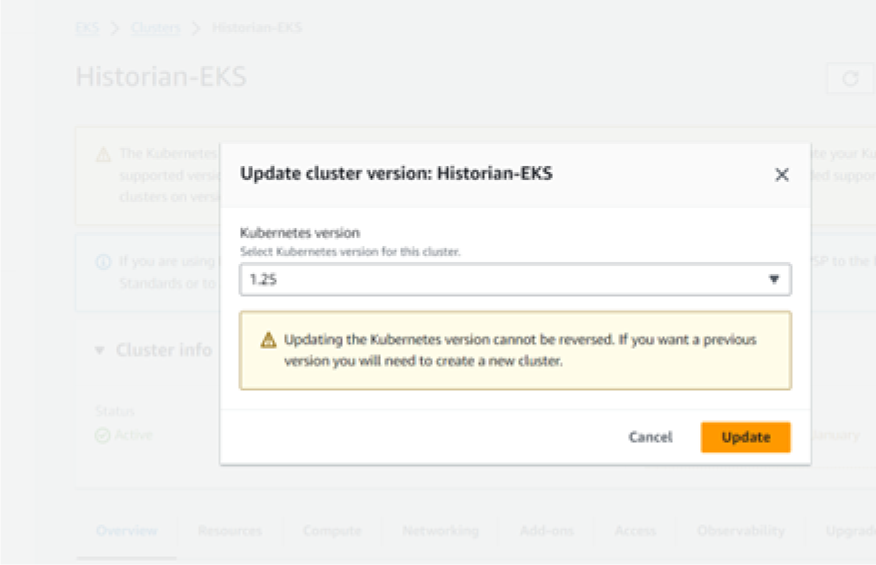


The Kubernetes version on your cluster (1.24) is reaching end of standard support in less than one month. Update your Kubernetes cluster to a supported version or your cluster will be onboarded to extended support on January 31, 2024. After the extended support preview ends, clusters on versions in extended support will be subject to additional fees. Learn more

If you are using PSP in your cluster, then before upgrading your cluster to version 1.25 you must migrate your PSP to the built-in Kubernetes Pod Security Standards or to a policy as code solution to avoid interruptions and protect your workloads. Learn more

Status: Active
Kubernetes version Info: 1.24
Support type: Standard support until January 31, 2024
Provider: EKS

3. Click the **Update now** button. This action will allow you to update to the next possible EKS version.



Update cluster version: Historian-EKS

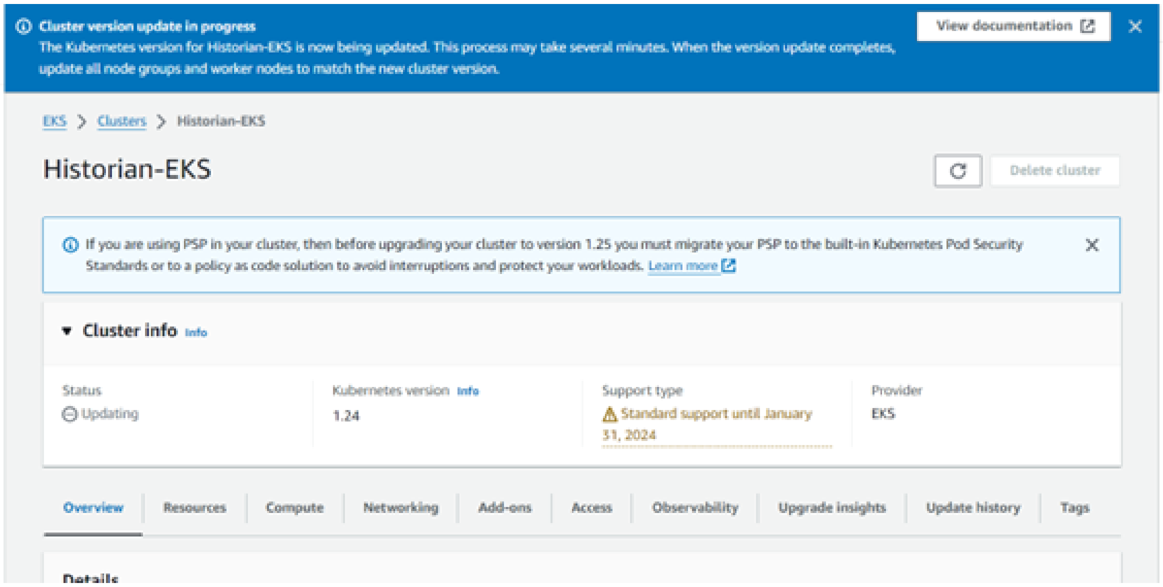
Kubernetes version

Select Kubernetes version for this cluster.

1.25

⚠ Updating the Kubernetes version cannot be reversed. If you want a previous version you will need to create a new cluster.

4. Click **Update** and the status will change to updating, as shown in the following figure.



Cluster version update in progress

The Kubernetes version for Historian-EKS is now being updated. This process may take several minutes. When the version update completes, update all node groups and worker nodes to match the new cluster version.

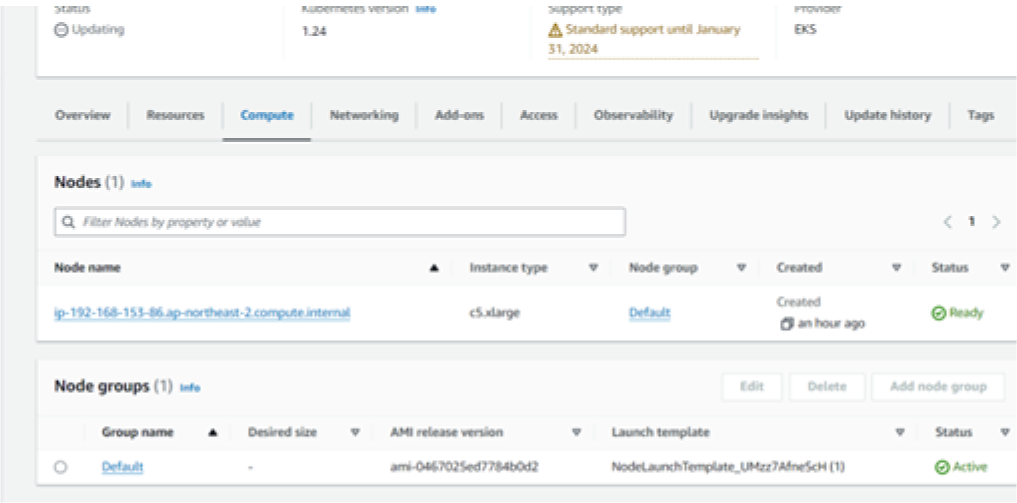
Historian-EKS

ⓘ If you are using PSP in your cluster, then before upgrading your cluster to version 1.25 you must migrate your PSP to the built-in Kubernetes Pod Security Standards or to a policy as code solution to avoid interruptions and protect your workloads. Learn more

Cluster info Info

Status	Kubernetes version Info	Support type	Provider
Updating	1.24	⚠ Standard support until January 31, 2024	EKS

5. Go to the **Compute** tab, and select the node group as shown in the following figure.



Status	Kubernetes version	Support type	Provider
Updating	1.24	Standard support until January 31, 2024	EKS

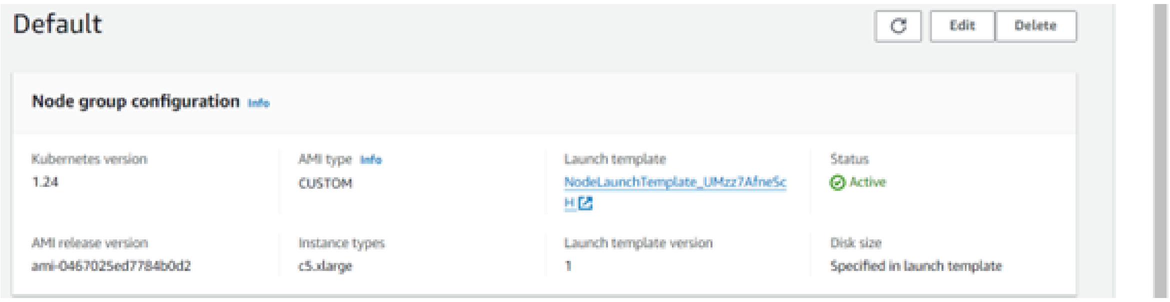
Nodes (1)

Node name	Instance type	Node group	Created	Status
ip-192-168-153-86.ap-northeast-2.compute.internal	c5.xlarge	Default	Created an hour ago	Ready

Node groups (1)

	Group name	Desired size	AMI release version	Launch template	Status
○	Default	-	ami-0467025ed7784b0d2	NodeLaunchTemplate_UMzz7AfneScH (1)	Active

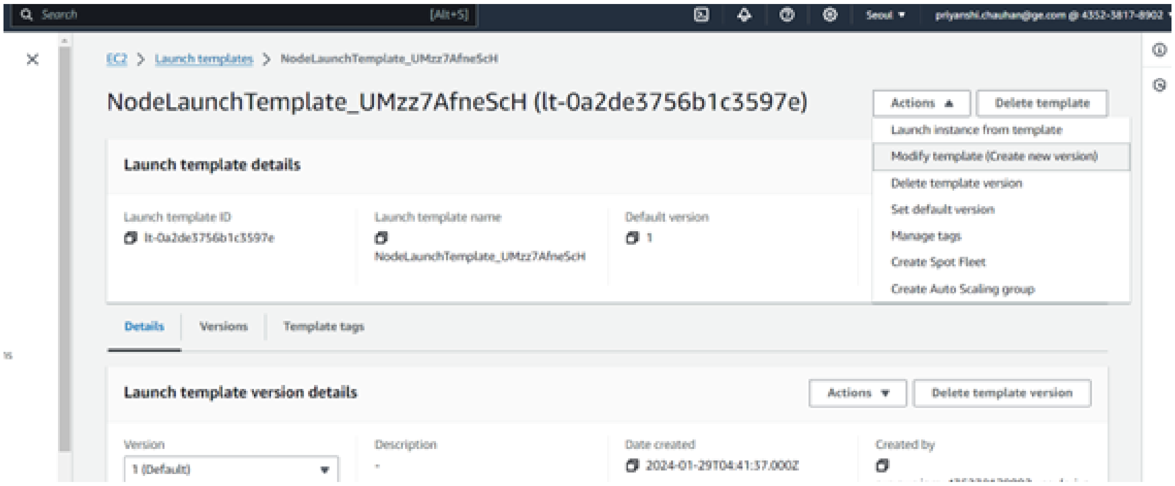
6. Select the **Default** node group and click **Launch template**. The default node group configuration appears.



Node group configuration

Kubernetes version 1.24	AMI type CUSTOM	Launch template NodeLaunchTemplate_UMzz7AfneScH	Status Active
AMI release version ami-0467025ed7784b0d2	Instance types c5.xlarge	Launch template version 1	Disk size Specified in launch template

7. Click **Actions** and select **Modify template** (create new version).


Search
[Alt+S]
Seoul
priyanshi.chauhan@ge.com @ 4352-3817-8902
EC2 > Launch templates > NodeLaunchTemplate_UMzz7AfneScH
Actions
Delete template
Launch instance from template
Modify template (Create new version)
Delete template version
Set default version
Manage tags
Create Spot Fleet
Create Auto Scaling group
Details
Versions
Template tags
Actions
Delete template version

NodeLaunchTemplate_UMzz7AfneScH (lt-0a2de3756b1c3597e)

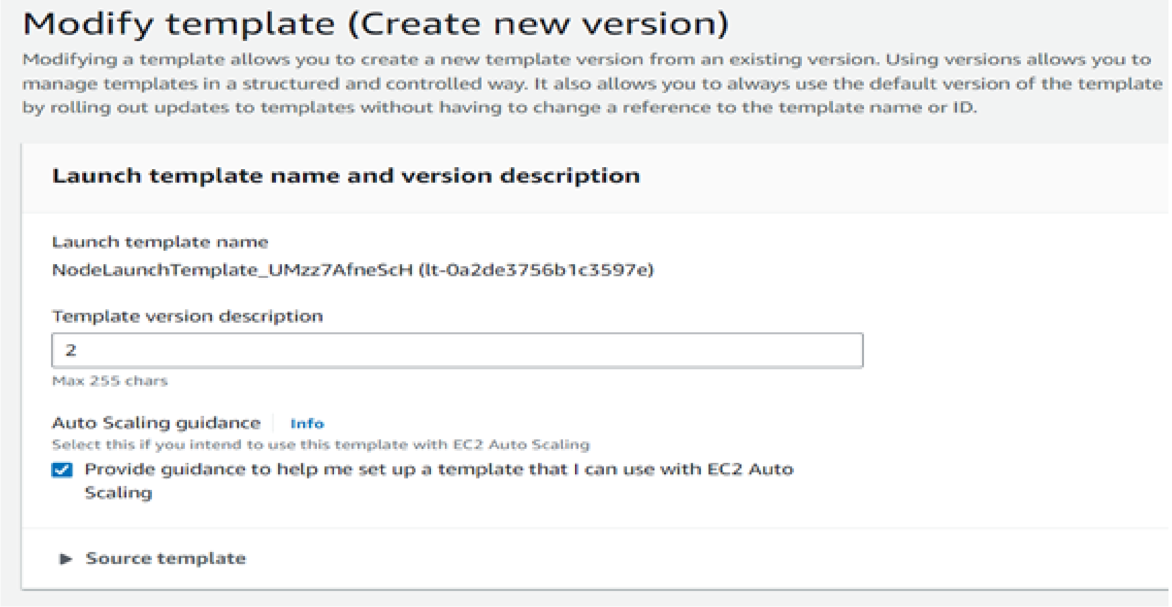
Launch template details

Launch template ID	Launch template name	Default version
lt-0a2de3756b1c3597e	NodeLaunchTemplate_UMzz7AfneScH	1

Launch template version details

Version	Description	Date created	Created by
1 (Default)	-	2024-01-29T04:41:37.000Z	

8. In the **Template version description**, enter 2 as it will be the second version of the template, and also select the **Auto Scaling guidance** check box as shown in the following figure.



Modify template (Create new version)

Modifying a template allows you to create a new template version from an existing version. Using versions allows you to manage templates in a structured and controlled way. It also allows you to always use the default version of the template by rolling out updates to templates without having to change a reference to the template name or ID.

Launch template name and version description

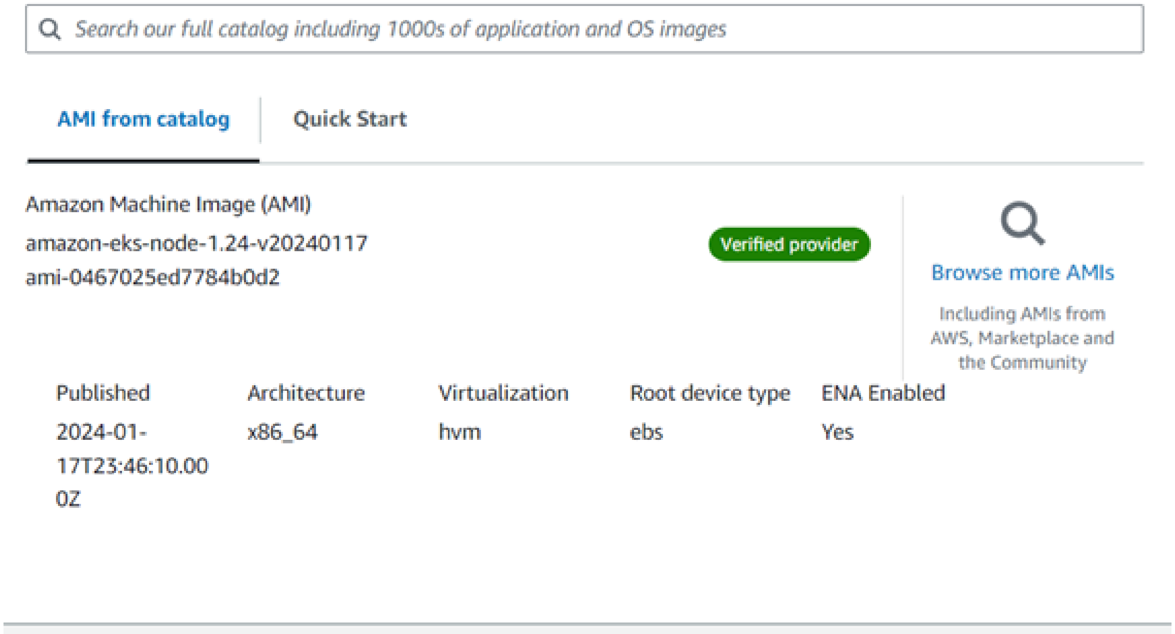
Launch template name
NodeLaunchTemplate_UMzz7AfneScH (lt-0a2de3756b1c3597e)

Template version description
2
Max 255 chars

Auto Scaling guidance | Info
Select this if you intend to use this template with EC2 Auto Scaling
☑ Provide guidance to help me set up a template that I can use with EC2 Auto Scaling

▶ Source template

9. While creating the new version, click the **Browse more AMIs** section.

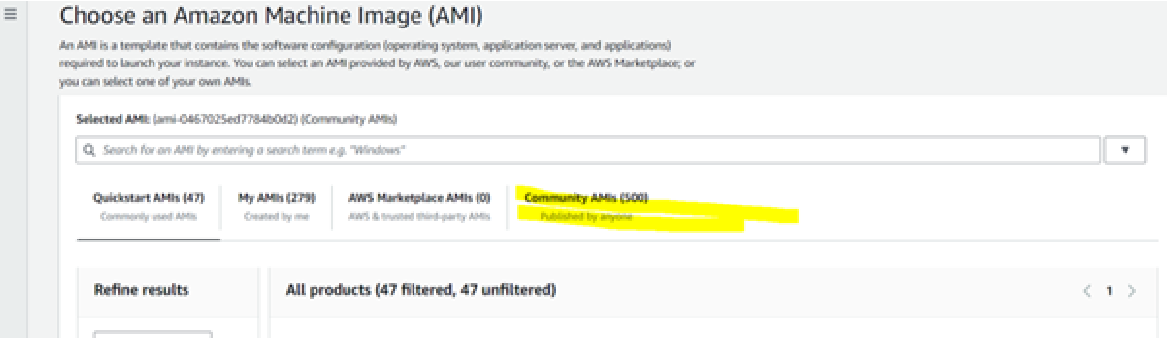


Search our full catalog including 1000s of application and OS images

Amazon Machine Image (AMI)
amazon-eks-node-1.24-v20240117
ami-0467025ed7784b0d2

Published	Architecture	Virtualization	Root device type	ENA Enabled
2024-01-17T23:46:10.000Z	x86_64	hvm	ebs	Yes

10. Click the fourth tab, **Community AMI**.



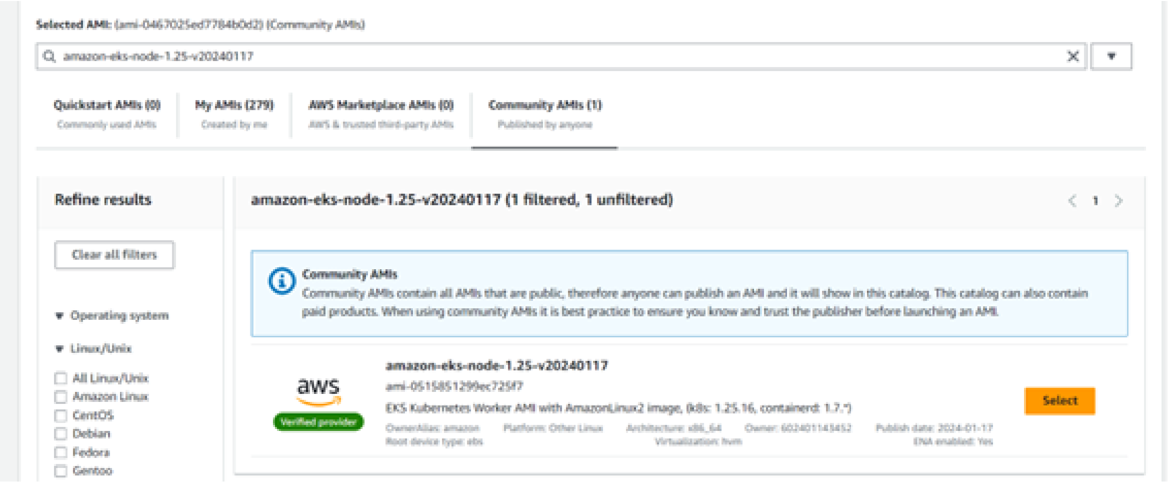
Choose an Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Selected AMI: (ami-0467025ed7784b0d2) (Community AMIs)

All products (47 filtered, 47 unfiltered)

11. Search the AMI of the latest version. For example, the following figure shows: **amazon-eks-node-1.25-v20240117**.



Selected AMI: (ami-0467025ed7784b0d2) (Community AMIs)

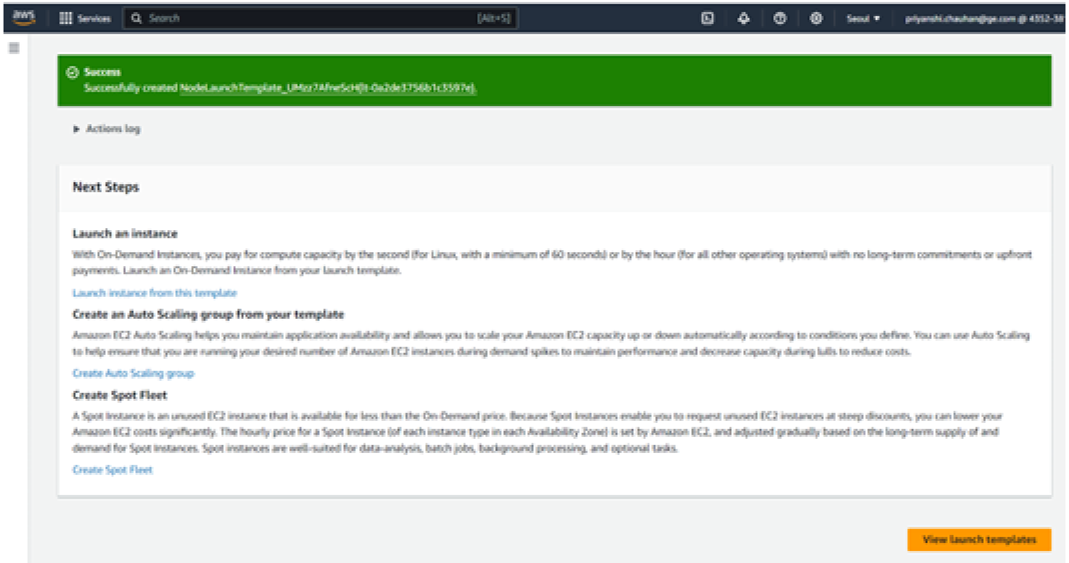
amazon-eks-node-1.25-v20240117

amazon-eks-node-1.25-v20240117 (1 filtered, 1 unfiltered)

Community AMIs
Community AMIs contain all AMIs that are public, therefore anyone can publish an AMI and it will show in this catalog. This catalog can also contain paid products. When using community AMIs it is best practice to ensure you know and trust the publisher before launching an AMI.

amazon-eks-node-1.25-v20240117
ami-0515851299ec725f7
EKS Kubernetes Worker AMI with AmazonLinux2 image, (k8s: 1.25.16, containerd: 1.7.*)
OwnerAlias: amazon Platform: Other Linux Architecture: x86_64 Owner: 602401143452 Publish date: 2024-01-17
Root device type: ebs Virtualization: hvm ENA enabled: Yes

12. Leave the defaults for all other options as it is. Click **Create template version**.



Success
Successfully created NodeLaunchTemplate_UMzz7AfneScH0t-0a2de3756b1c3597e).

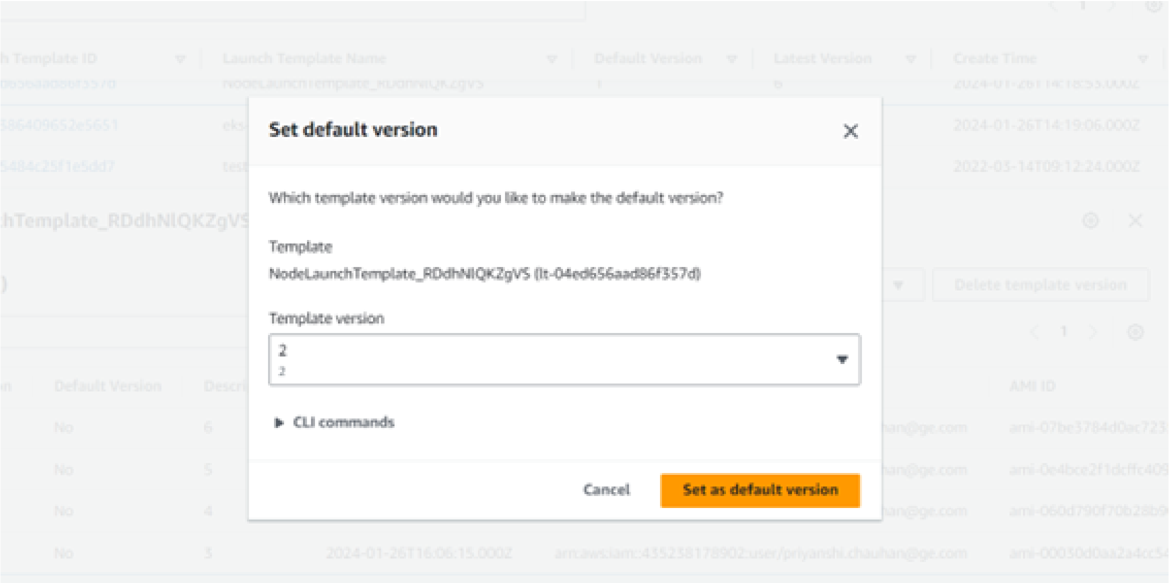
Next Steps

Launch an instance
With On-Demand Instances, you pay for compute capacity by the second (for Linux, with a minimum of 60 seconds) or by the hour (for all other operating systems) with no long-term commitments or upfront payments. Launch an On-Demand Instance from your launch template.
Launch instance from this template

Create an Auto Scaling group from your template
Amazon EC2 Auto Scaling helps you maintain application availability and allows you to scale your Amazon EC2 capacity up or down automatically according to conditions you define. You can use Auto Scaling to help ensure that you are running your desired number of Amazon EC2 instances during demand spikes to maintain performance and decrease capacity during lulls to reduce costs.
Create Auto Scaling group

Create Spot Fleet
A Spot Instance is an unused EC2 instance that is available for less than the On-Demand price. Because Spot Instances enable you to request unused EC2 instances at steep discounts, you can lower your Amazon EC2 costs significantly. The hourly price for a Spot Instance (of each instance type in each Availability Zone) is set by Amazon EC2, and adjusted gradually based on the long-term supply of and demand for Spot Instances. Spot instances are well-suited for data-analysis, batch jobs, background processing, and optional tasks.
Create Spot Fleet

13. After creating the template, set it as the default version by clicking on actions and selecting version 2 as default version, as shown in the following figure. Click **Set as default version**.



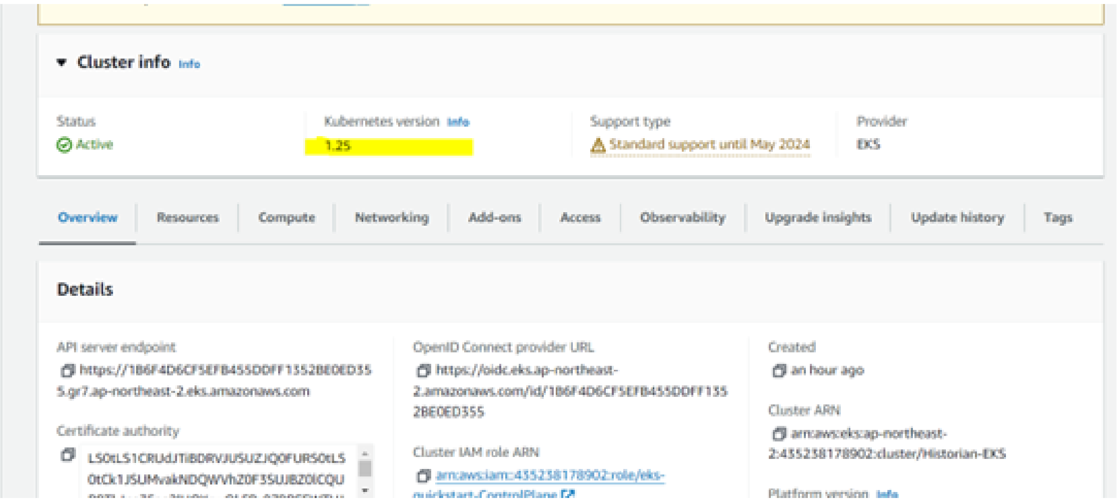
Set default version

Which template version would you like to make the default version?

Template
NodeLaunchTemplate_RDdhNIQKZgVS (lt-04ed656aad86f357d)

Template version
2

14. Go back to the EKS page, and check if it is upgraded to the version you chose to upgrade. After it is done, you can then upgrade the node group with new template that you just created.



Cluster info Info

Status: Active
Kubernetes version: 1.25
Support type: Standard support until May 2024
Provider: EKS

Details

API server endpoint
https://1B6F4D6CF5EFB455DDFF1352BE0ED355.gr7.ap-northeast-2.eks.amazonaws.com

Certificate authority
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUMvakNDQWVhZ0F3SUJBZ0lCQU

OpenID Connect provider URL
https://oidc.eks.ap-northeast-2.amazonaws.com/id/1B6F4D6CF5EFB455DDFF1352BE0ED355

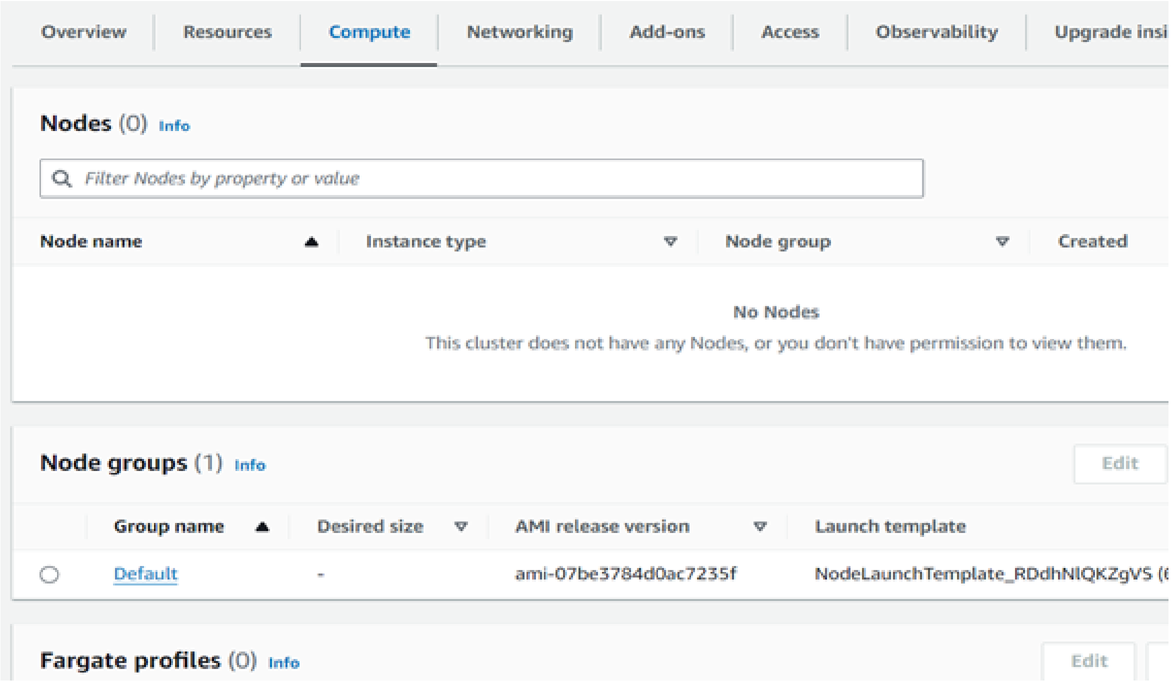
Cluster IAM role ARN
arn:aws:iam::435238178902:role/eks-quickstart-ControlPlane

Created
an hour ago

Cluster ARN
arn:aws:eks:ap-northeast-2:435238178902:cluster/Historian-EKS

Platform version Info

15. To update the node group, go to **Compute** tab and select node group with name **Default** as shown in the following figure:



Nodes (0) Info

Filter Nodes by property or value

Node name	Instance type	Node group	Created

No Nodes

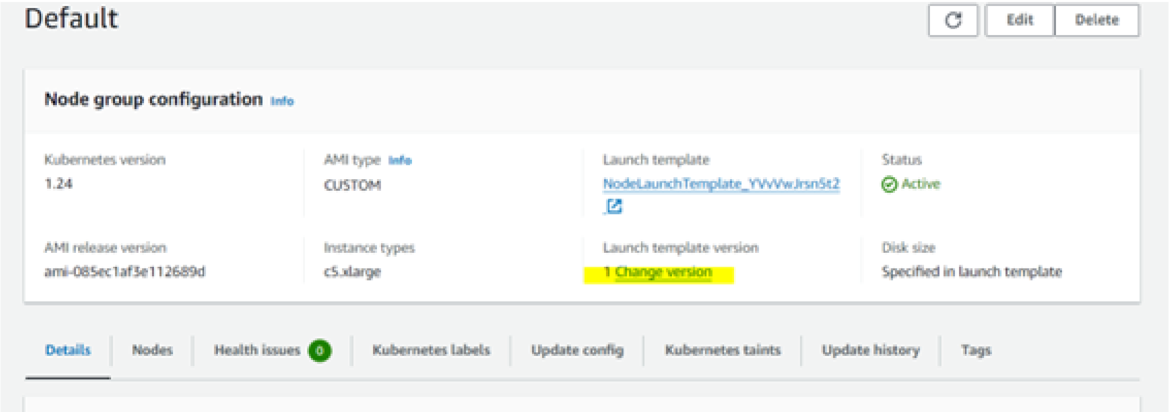
This cluster does not have any Nodes, or you don't have permission to view them.

Node groups (1) Info

	Group name	Desired size	AMI release version	Launch template
○	Default	-	ami-07be3784d0ac7235f	NodeLaunchTemplate_RDdhNlQKZgVS ((

Fargate profiles (0) Info

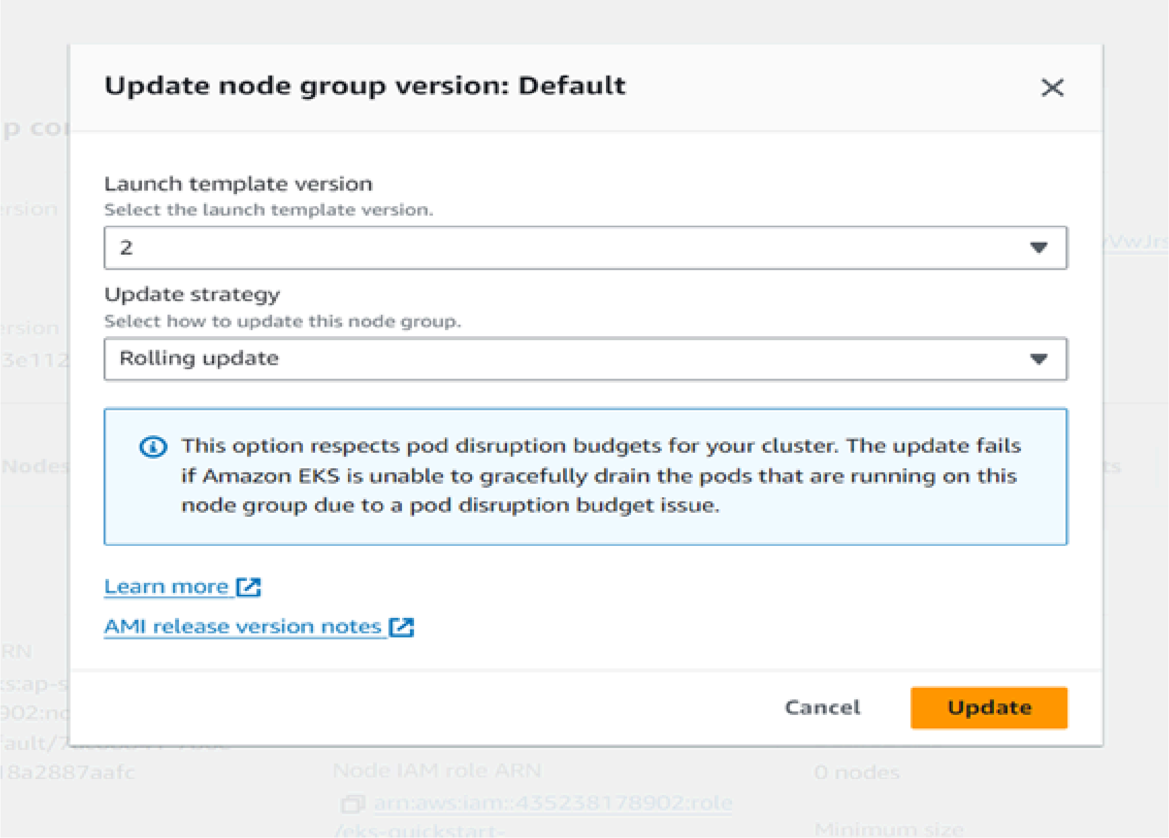
16. In the node group configuration, you can see **Change version** in Launch template version as shown in the following figure:


Default

Node group configuration Info

Kubernetes version	AMI type Info	Launch template	Status
1.24	CUSTOM	NodeLaunchTemplate_YVvVwJrsn5t2	Active
AMI release version	**Instance types**	**Launch template version**	**Disk size**
ami-085ec1af3e112689d	c5.xlarge	1 Change version	Specified in launch template

17. Click **Change version** and select the template you created in the previous steps and then click **Update**:



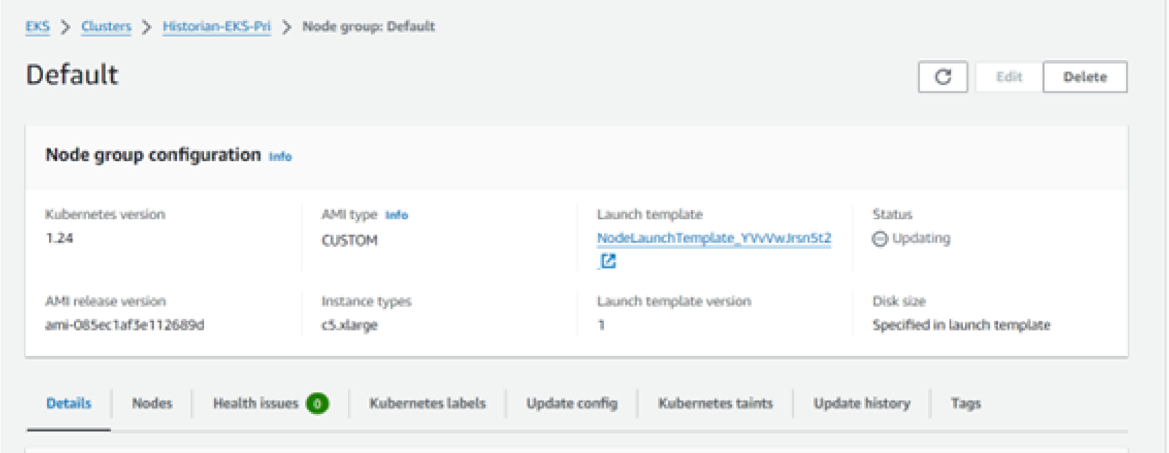
Update node group version: Default

Launch template version
Select the launch template version.
2

Update strategy
Select how to update this node group.
Rolling update

This option respects pod disruption budgets for your cluster. The update fails if Amazon EKS is unable to gracefully drain the pods that are running on this node group due to a pod disruption budget issue.

18. After performing this step, notice that the status changes to **Updating** as shown in the following figure.



Default

Node group configuration

Kubernetes version 1.24	AMI type CUSTOM	Launch template NodeLaunchTemplate_YVvVwJrsn5t2	Status Updating
AMI release version ami-085ec1af3e112689d	Instance types c5.xlarge	Launch template version 1	Disk size Specified in launch template

19. Wait a few mintues to complete the upgrade. After it is done, check whether the cluster version and node group version is same or not.
20. Repeat steps 4-13 and create new template version for every upgrade and apply it. The AMI names for every K8s version are as follows:

Kubernetes 1.25: amazon-eks-node-1.25-v20240117

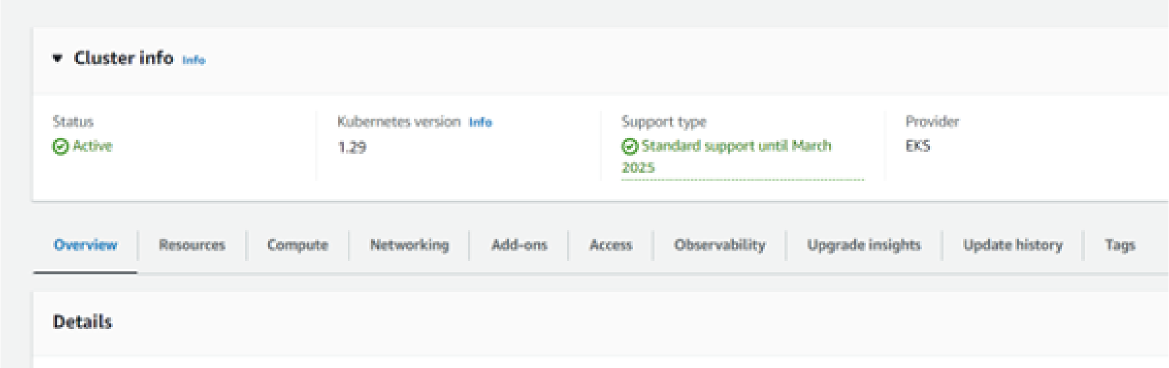
Kubernetes 1.26: amazon-eks-node-1.26-v20240117

Kubernetes 1.27: amazon-eks-node-1.27-v20240117

Kubernetes 1.28: amazon-eks-node-1.28-v20240117

Kubernetes 1.29: amazon-eks-node-1.29-v20240117

21. After complete, confirm both the EKS version and node group version is at latest.



▼ Cluster info Info

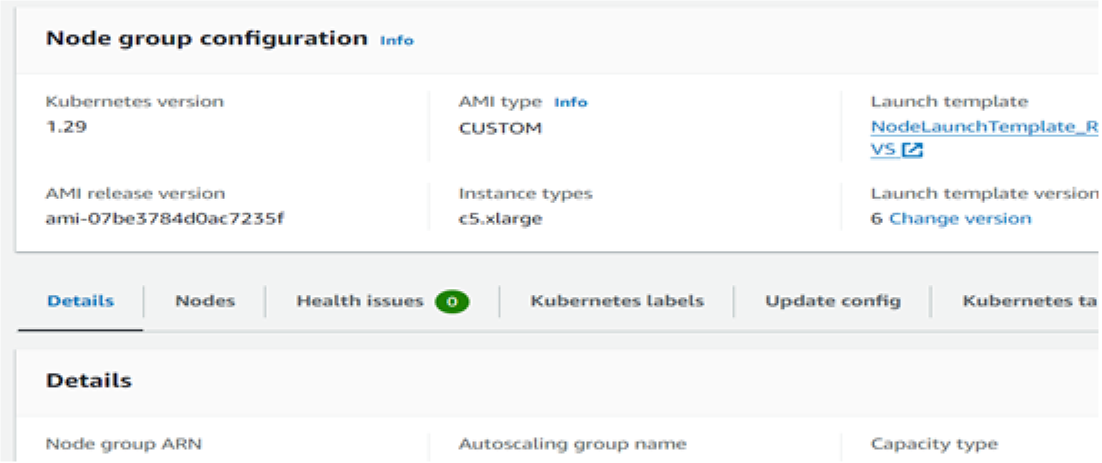
Status: Active

Kubernetes version Info: 1.29

Support type: Standard support until March 2025

Provider: EKS

Details



Node group configuration Info

Kubernetes version: 1.29

AMI type Info: CUSTOM

Launch template: NodeLaunchTemplate_R VS

AMI release version: ami-07be3784d0ac7235f

Instance types: c5.xlarge

Launch template version: 6 Change version

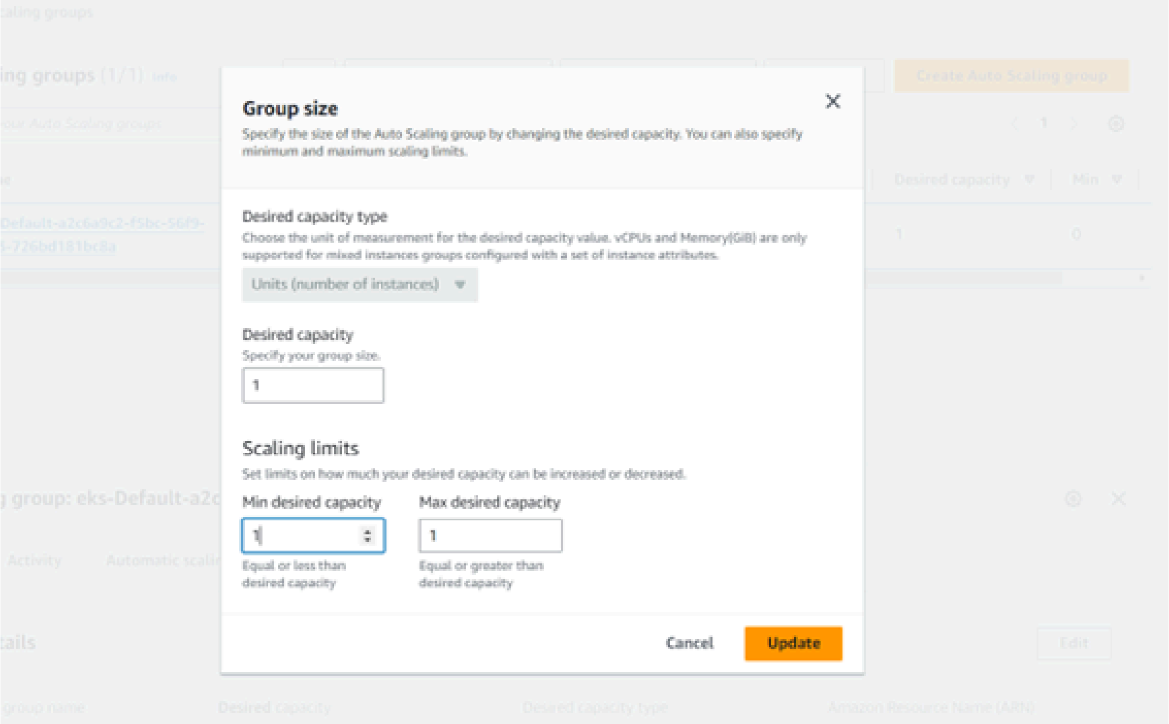
Details

Node group ARN

Autoscaling group name

Capacity type

22. Finally, change the number of instances in auto scaling groups to 1 and wait for 3-4 mins until its in a running state and all pods running as well.



Group size

Specify the size of the Auto Scaling group by changing the desired capacity. You can also specify minimum and maximum scaling limits.

Desired capacity type

Choose the unit of measurement for the desired capacity value. vCPUs and Memory(GiB) are only supported for mixed instances groups configured with a set of instance attributes.

Units (number of instances)

Desired capacity

Specify your group size.

1

Scaling limits

Set limits on how much your desired capacity can be increased or decreased.

Min desired capacity: 1

Equal or less than desired capacity

Max desired capacity: 1

Equal or greater than desired capacity

Installing Configuration Hub

About Setting up Configuration Hub

To set up Configuration Hub, you must perform the following steps:

1. Deploy Proficy Historian for AWS. *(on page 43)*
2. Install Web-based Clients *(on page 75)*.
3. Install collectors *(on page 90)*.
4. Perform the post-installation tasks *(on page 95)*.

If you want to upgrade Configuration Hub, refer to Upgrade Configuration Hub *(on page 96)*.

After you install or upgrade the required components, you can access Configuration Hub *(on page 97)*.

Install Web-based Clients

Deploy Proficy Historian for AWS *(on page 43)*.

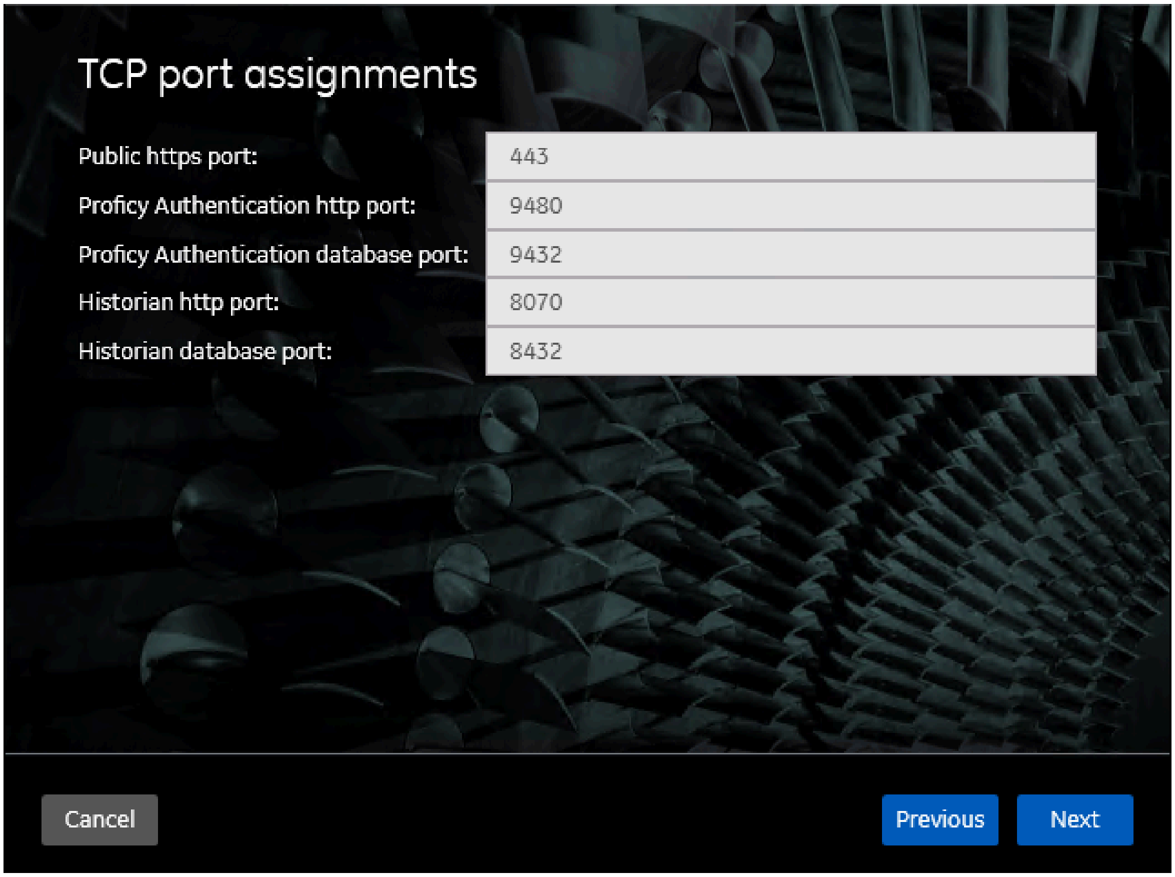
This topic describes how to install Web-based Clients using a GUI-based installer. Use these steps if you want to use Windows Configuration Hub with Cloud Historian.

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.

   TCP port assignments

   Public https port: 443

   Proficy Authentication http port: 9480

   Proficy Authentication database port: 9432

   Historian http port: 8070

   Historian database port: 8432

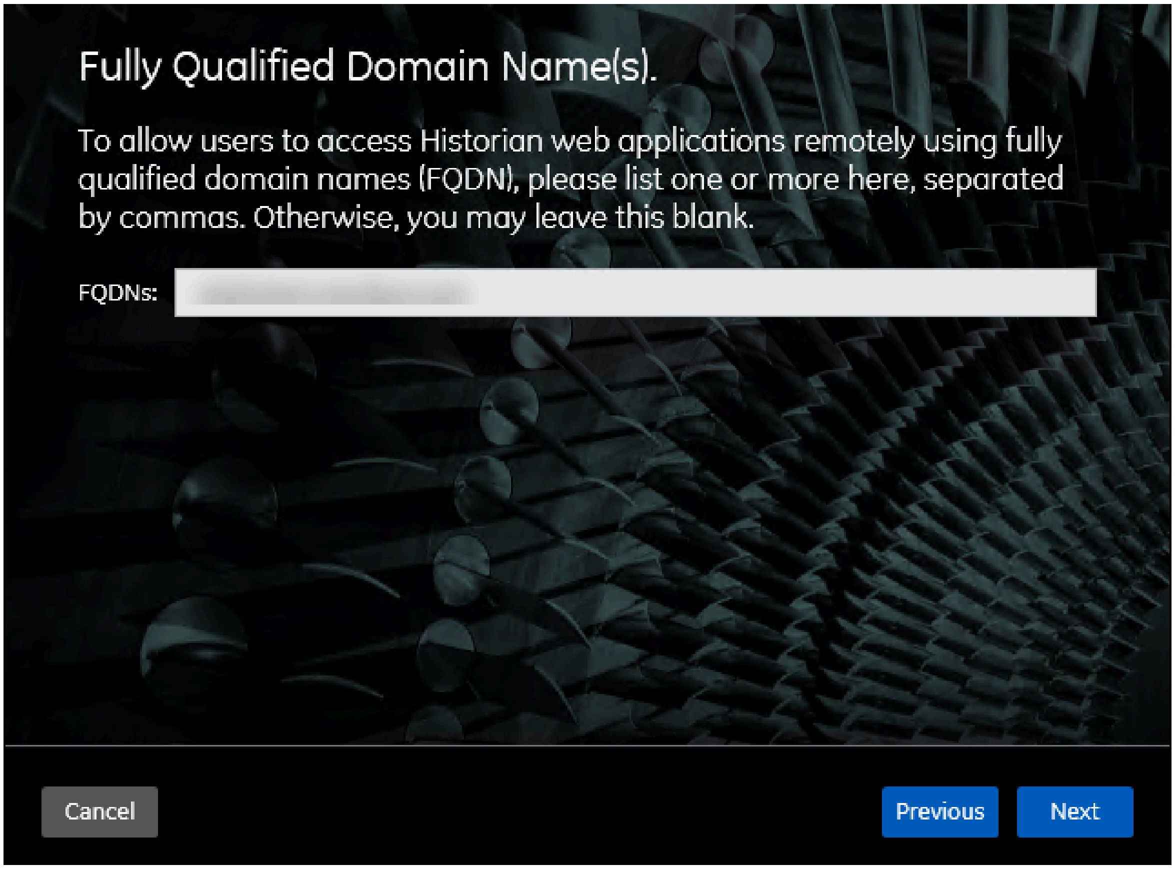
5. As needed, change the values for TCP port assignments as described in the following table, and then select **Next**.

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:

Field	Description
	◦ 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. ◦ 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.
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.



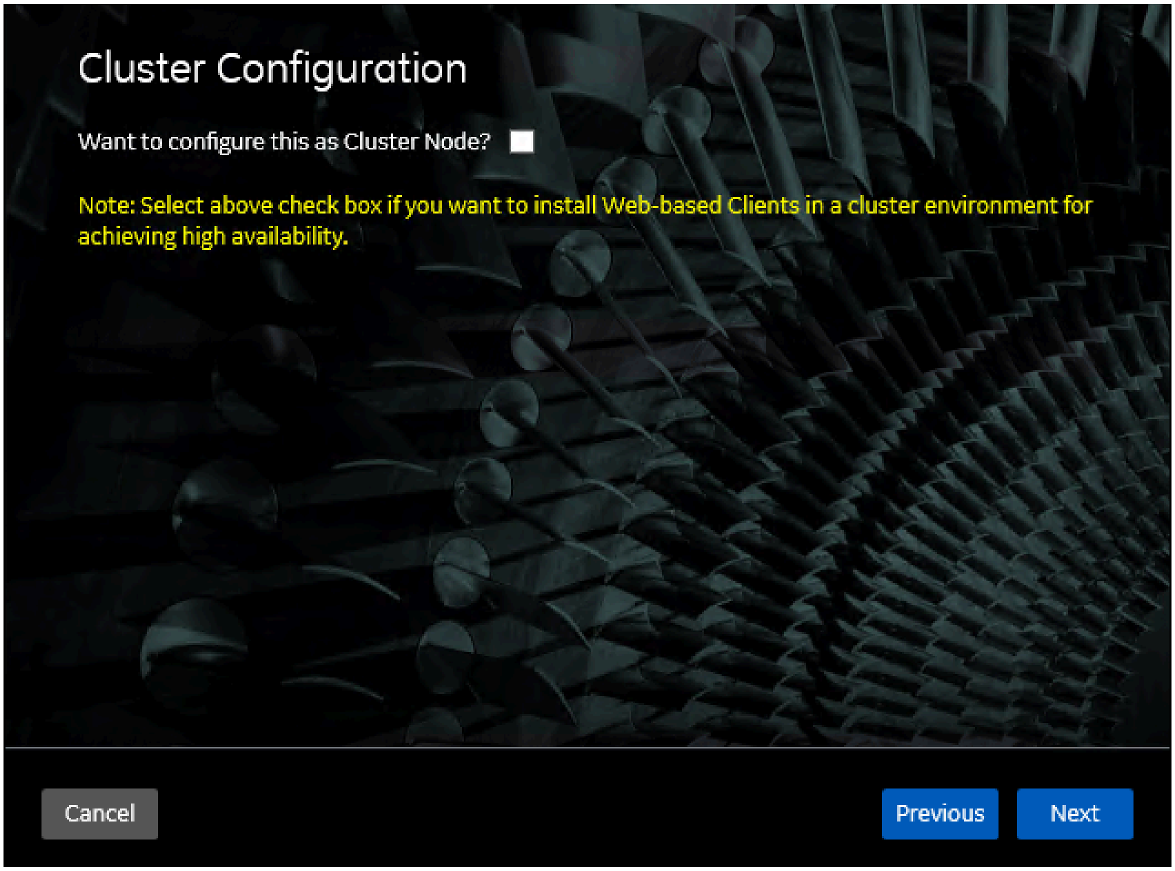
Fully Qualified Domain Name(s).

To allow users to access Historian web applications remotely using fully qualified domain names (FQDN), please list one or more here, separated by commas. Otherwise, you may leave this blank.

FQDNs:

6. In the **FDQNs** field, enter the fully qualified domain names, and then select **Next**.

   The **Cluster Configuration** page appears.



Cluster Configuration

Want to configure this as Cluster Node? □

Note: Select above check box if you want to install Web-based Clients in a cluster environment for achieving high availability.

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

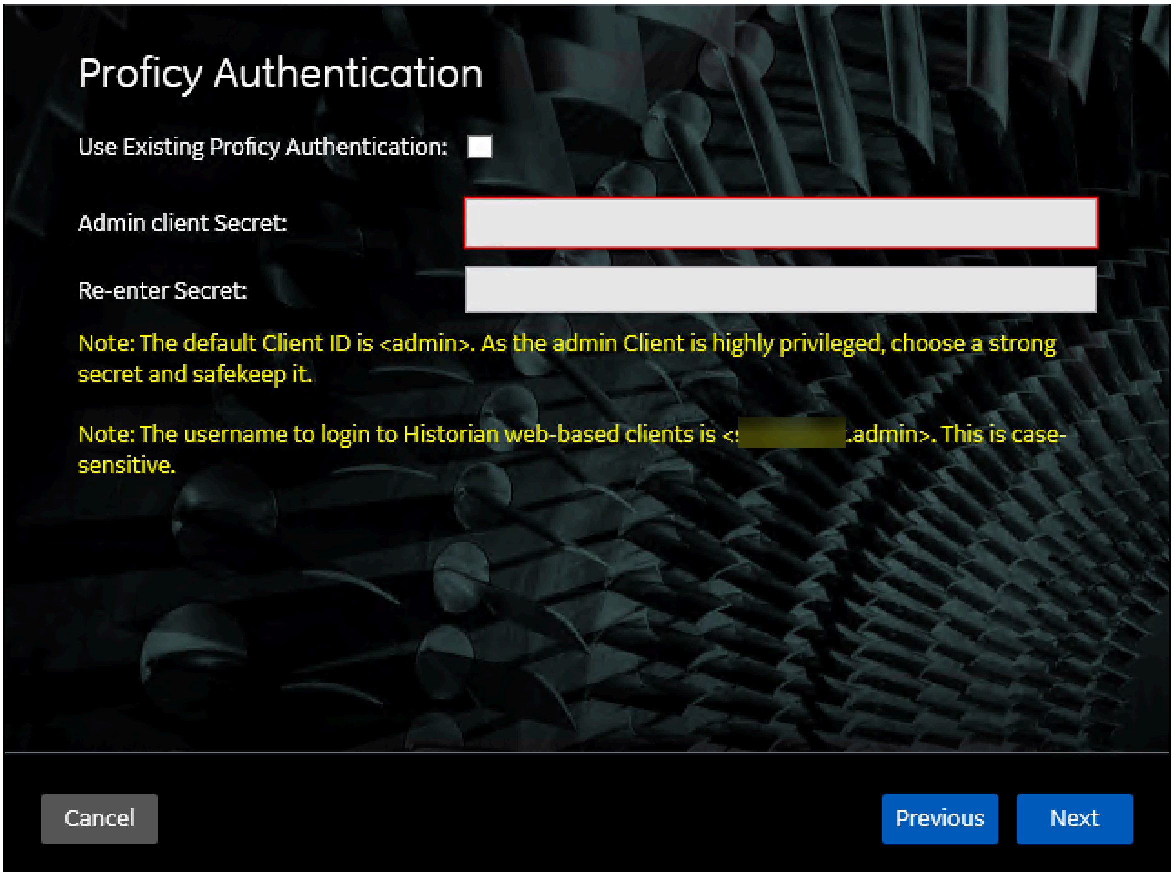
7. If you want high availability of Web-based Clients, select the **Cluster Node** check box, and enter values as described in the following table.

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)*.
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.



Proficy Authentication

Use Existing Proficy Authentication: □

Admin client Secret:

Re-enter Secret:

Note: The default Client ID is <admin>. As the admin Client is highly privileged, choose a strong secret and safekeep it.

Note: The username to login to Historian web-based clients is <[redacted].admin>. This is case-sensitive.

- 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**.

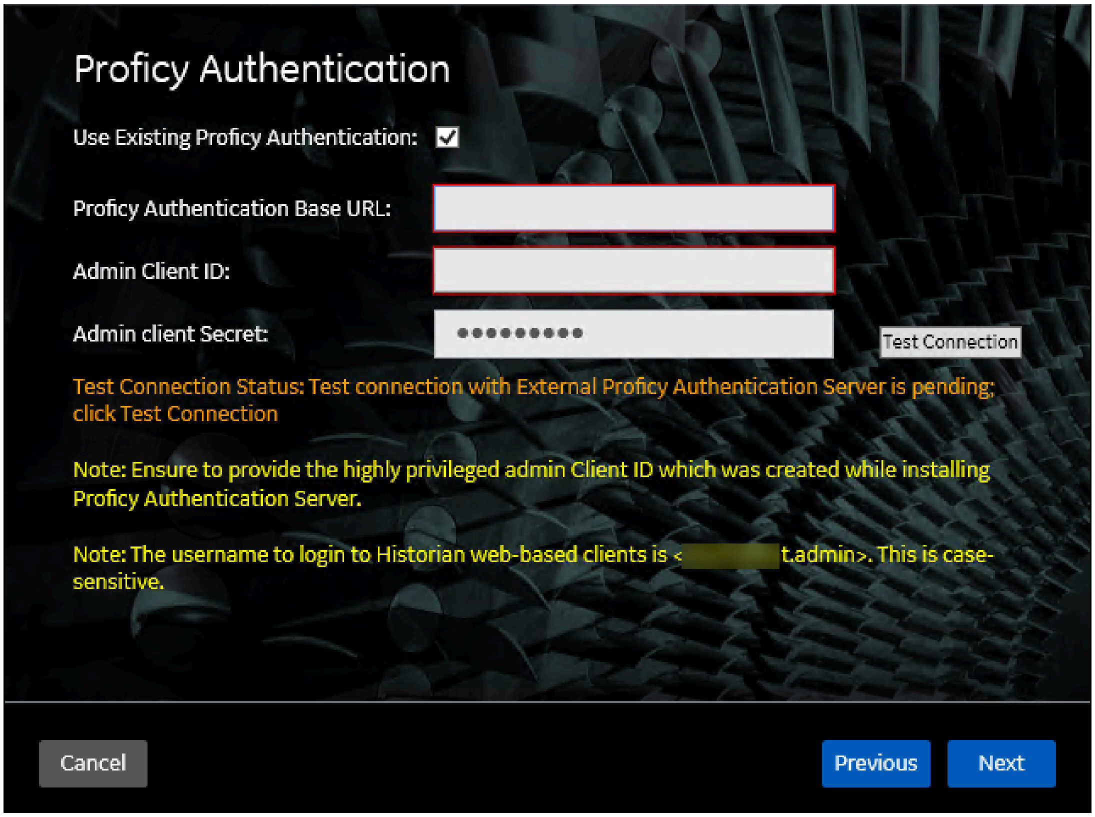
   The admin client secret must satisfy the following conditions:

   - 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.



Proficy Authentication

Use Existing Proficy Authentication: ☑

Proficy Authentication Base URL:

Admin Client ID:

Admin client Secret: ••••••••••

Test Connection Status: Test connection with External Proficy Authentication Server is pending; click Test Connection

Note: Ensure to provide the highly privileged admin Client ID which was created while installing Proficy Authentication Server.

Note: The username to login to Historian web-based clients is <[redacted]t.admin>. This is case-sensitive.

b. Enter values as described in the following table.

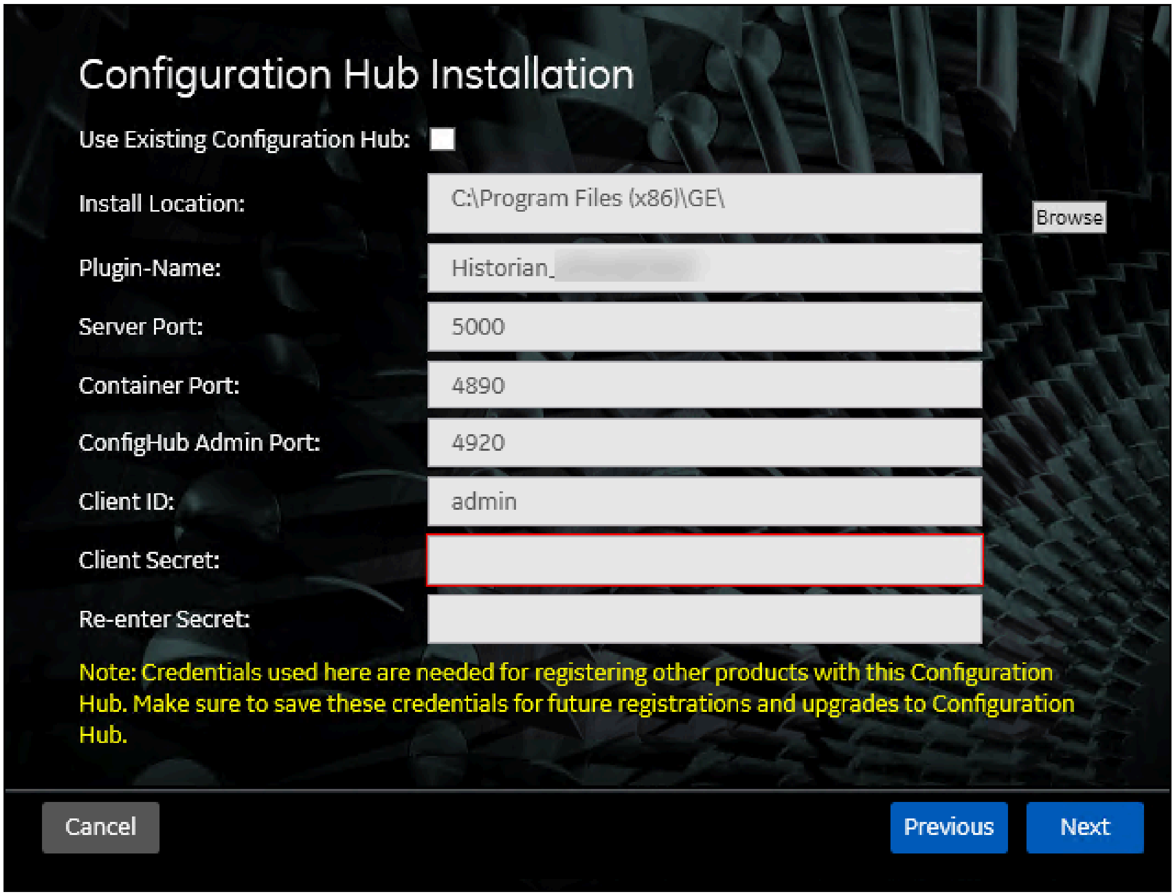
Field	Description
Proficy Authentication Base URL	Enter the URL of the external Proficy Authentication server in the following format: https://<*Proficy Authentication server name*>:<*port number*>, where <*Proficy Authentication server name*> is the FQDN or hostname of the machine on which Proficy Authentication is installed. By default, the port number is 443. **Note:** 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 Installation

Use Existing Configuration Hub: □

Install Location: C:\Program Files (x86)\GE\

Plugin-Name: Historian_[redacted]

Server Port: 5000

Container Port: 4890

ConfigHub Admin Port: 4920

Client ID: admin

Client Secret:

Re-enter Secret:

Note: Credentials used here are needed for registering other products with this Configuration Hub. Make sure to save these credentials for future registrations and upgrades to 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.

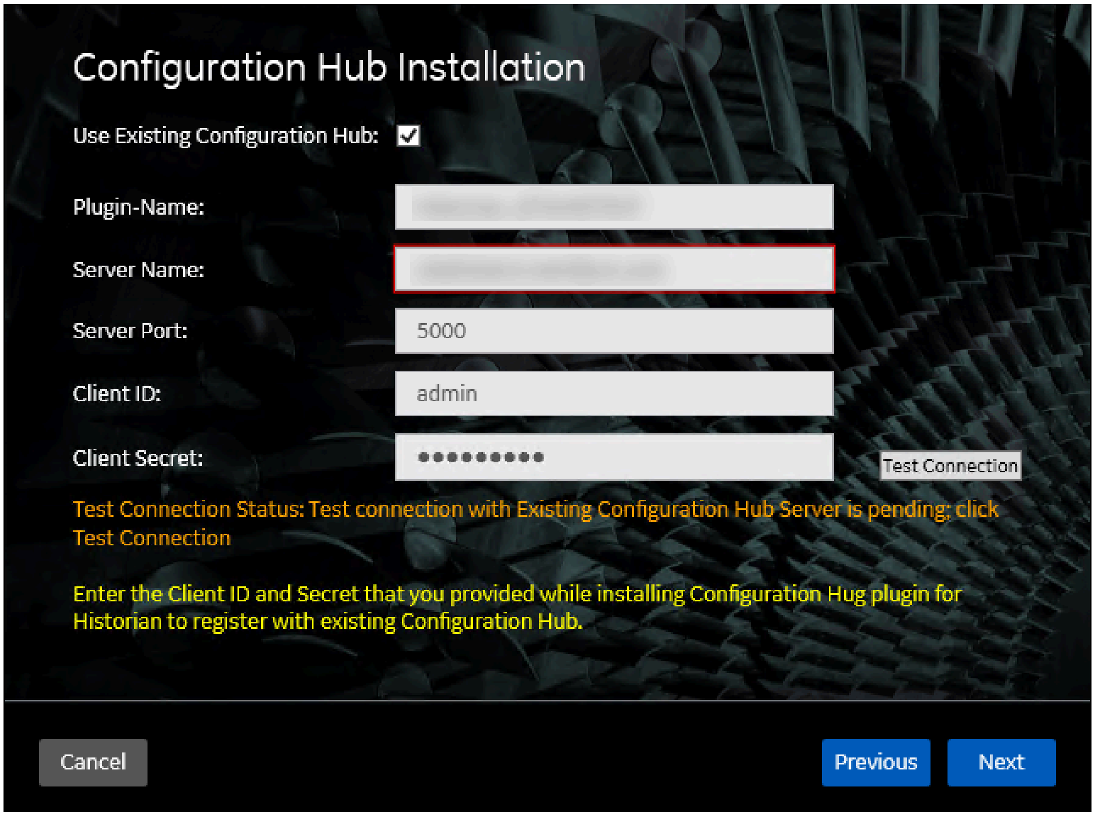
12. If you want to install Configuration Hub, ensure that the **Use Existing Configuration Hub** check box is cleared, and then provide values as described in the following table.

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



Configuration Hub Installation

Use Existing Configuration Hub: ☑

Plugin-Name:

Server Name:

Server Port: 5000

Client ID: admin

Client Secret: ●●●●●●●●●●

Test Connection Status: Test connection with Existing Configuration Hub Server is pending; click Test Connection

Enter the Client ID and Secret that you provided while installing Configuration Hug plugin for Historian to register with existing Configuration Hub.

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.

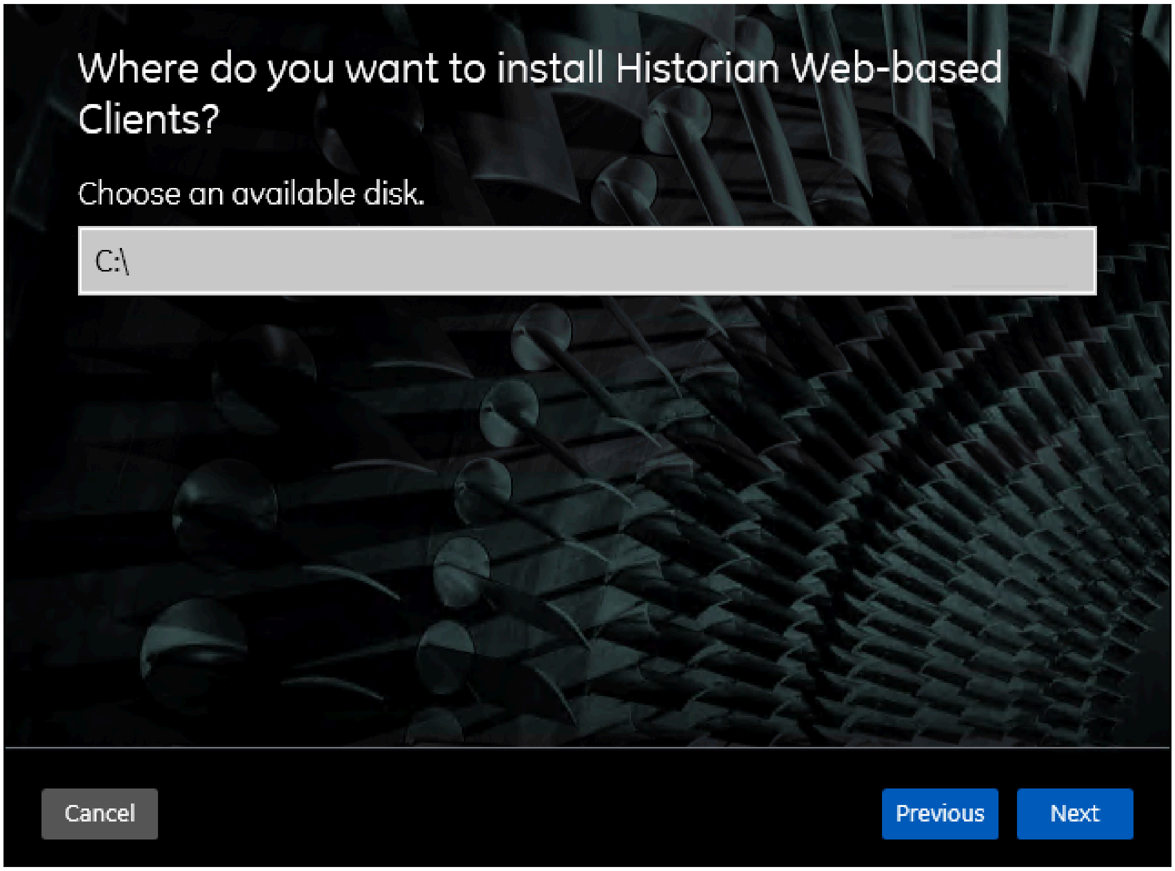
Field	Description
Client ID	If needed, modify the username to connect to Configuration Hub. The default value is admin.
Client Secret	Enter the password to connect to Configuration Hub.

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.



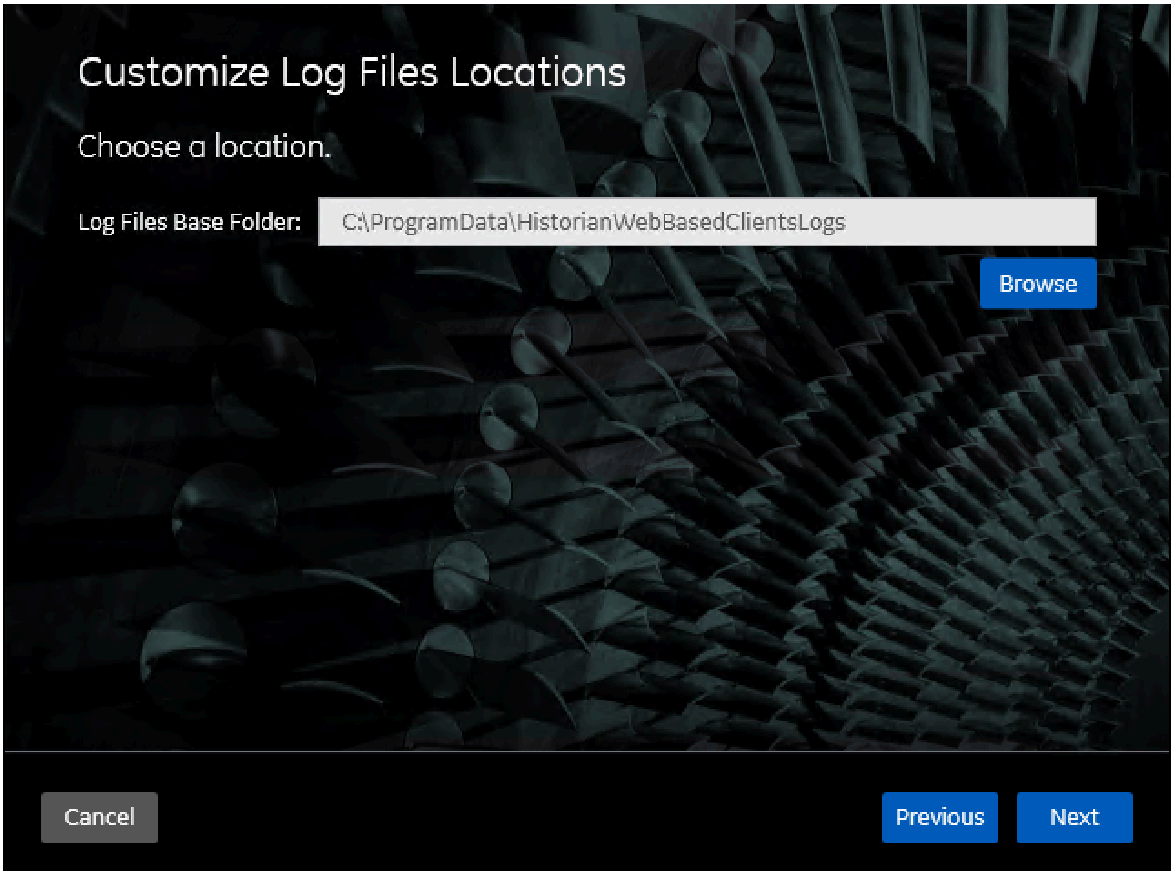
Where do you want to install Historian Web-based Clients?

Choose an available disk.

C:\

15. If needed, change the installation drive for Web-based Clients, and then select **Next**.

The log files location page appears.



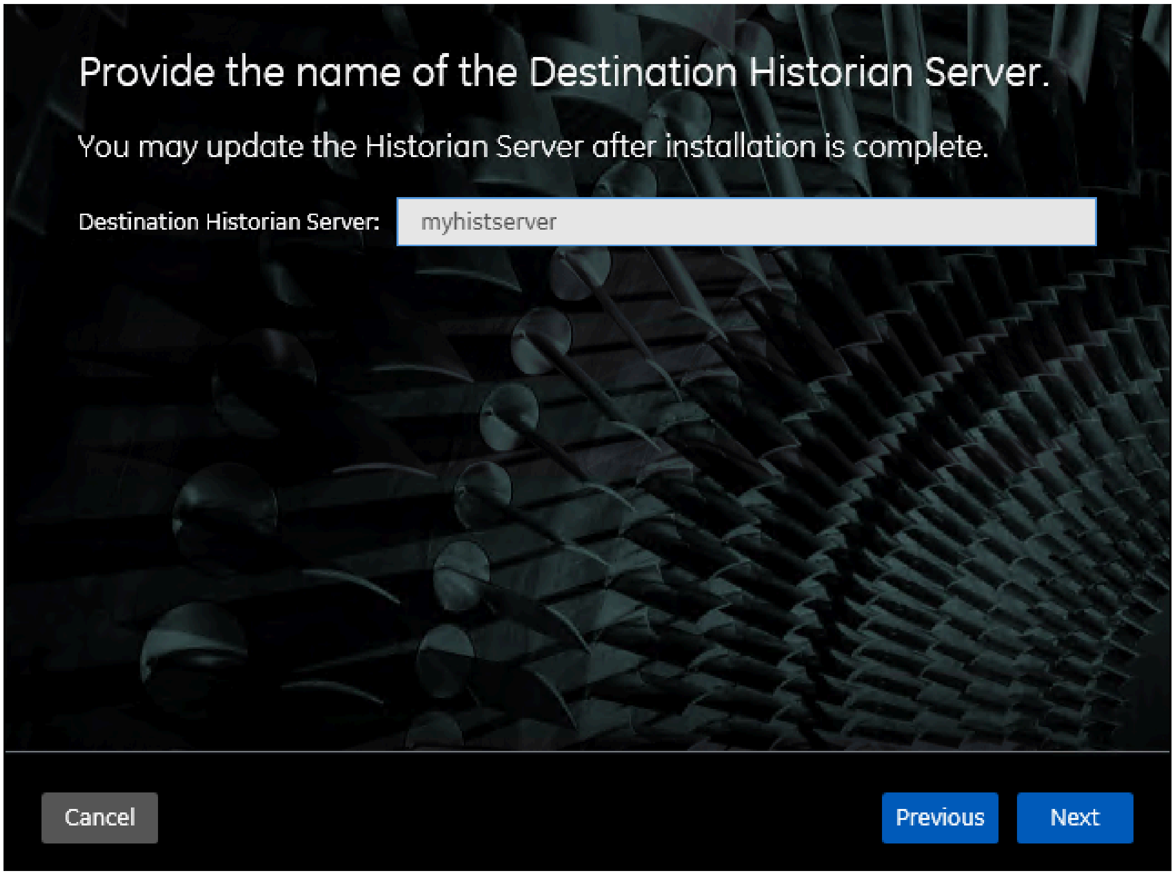
Customize Log Files Locations

Choose a location.

Log Files Base Folder: C:\ProgramData\HistorianWebBasedClientsLogs

16. If needed, change the location for log files, and then select **Next**.

    The destination Historian server page appears.



Provide the name of the Destination Historian Server.

You may update the Historian Server after installation is complete.

Destination Historian Server: myhistserver

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

- Deploy Proficy Historian for AWS *(on page 43)*.
- If you want to upgrade your collectors from any previous versions of the Cloud collectors, first uninstall the collectors and then install the Historian 2024 version collectors from the Historian 2024 install media (.ISO).

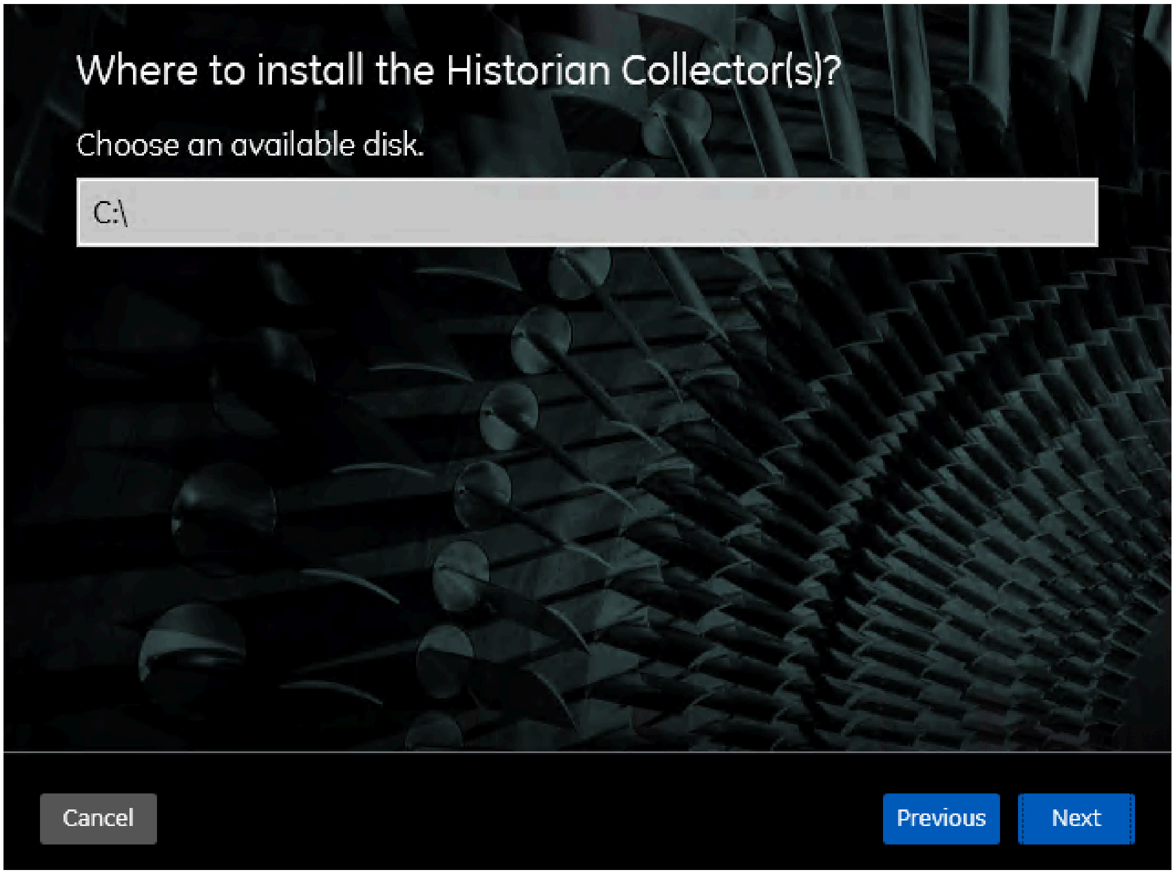
This topic describes how to install collectors using an installer. You can also install them at a command prompt *(on page)*. If you have already installed collectors in a previous version of Proficy Historian for AWS, you can upgrade them by installing them again.

1. Run the Windows Installer using 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.



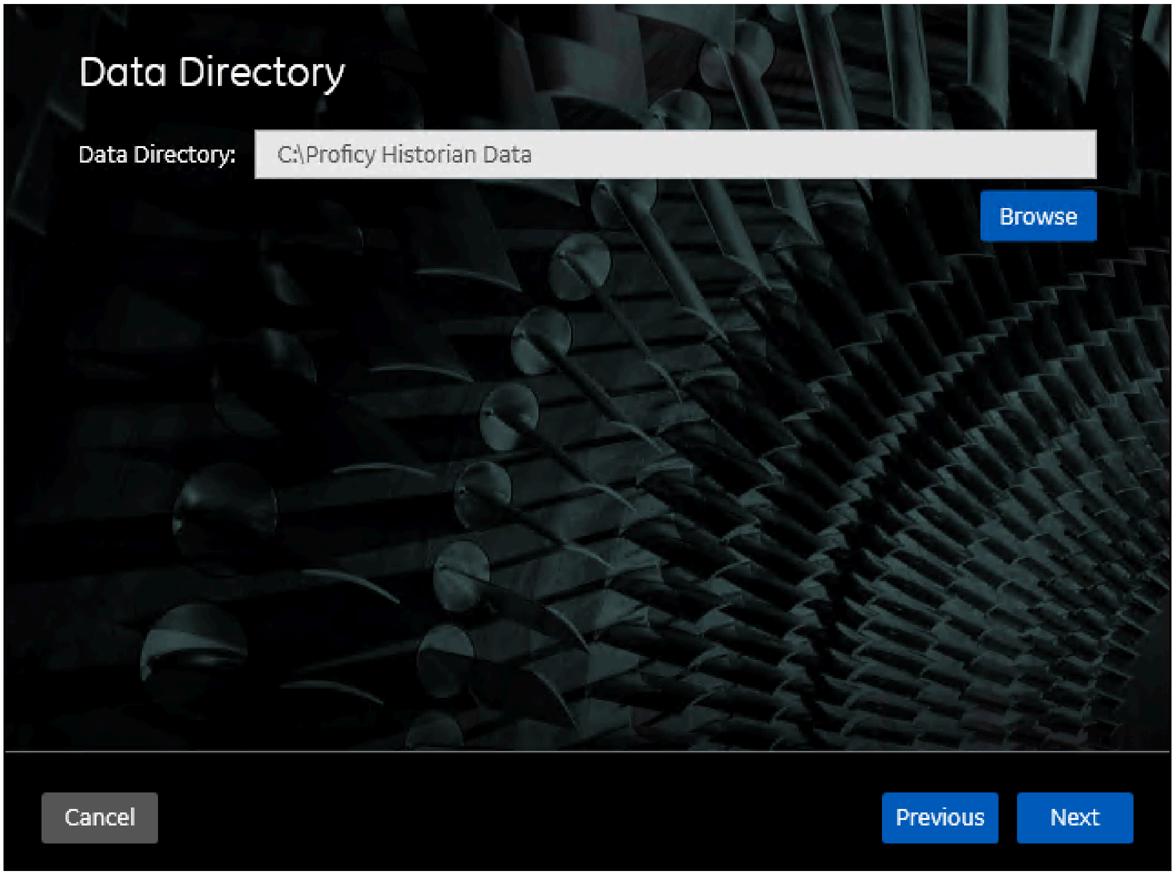
Where to install the Historian Collector(s)?

Choose an available disk.

C:\

5. If needed, modify the installation drive, and then select **Next**.

   The data directory page appears.

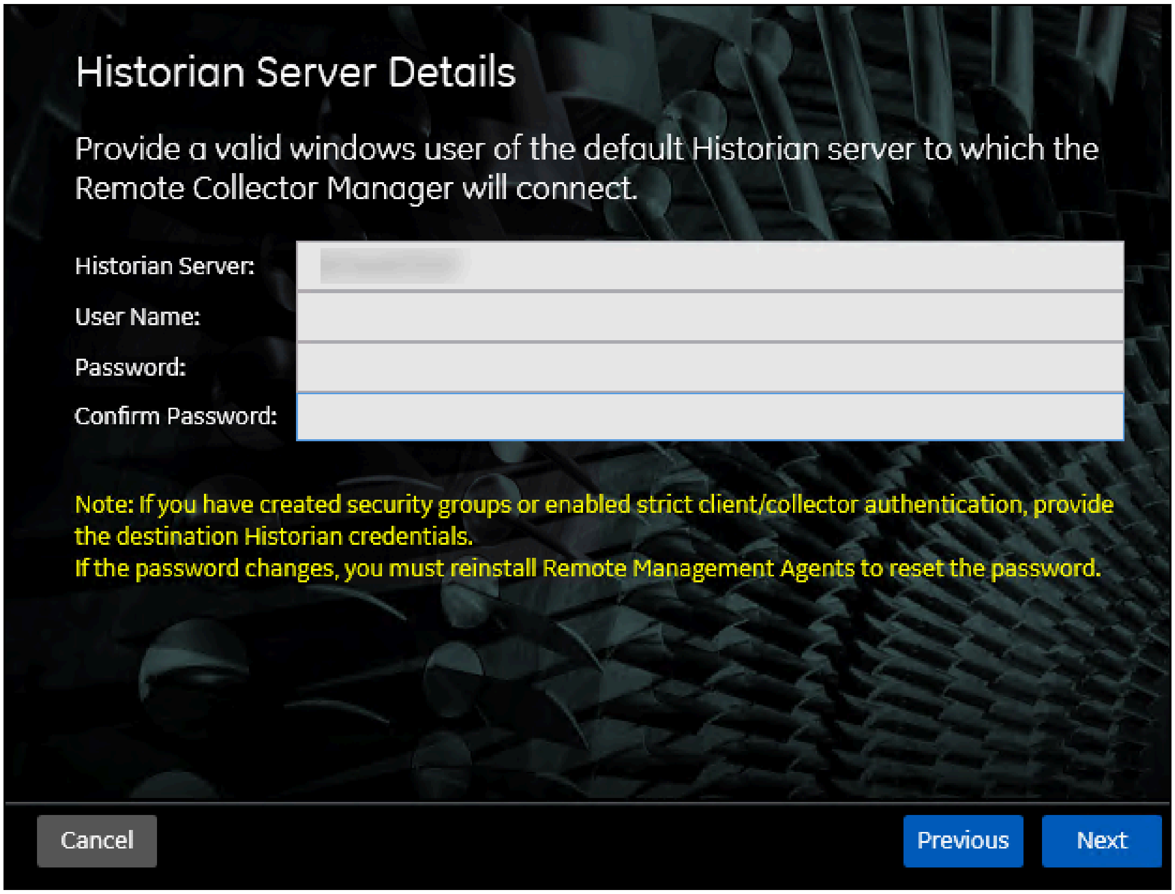


Data Directory

Data Directory: C:\Proficy Historian Data

6. If needed, change the folder for storing the collector log files, and then select **Next**.

   The destination Historian server page appears.



Historian Server Details

Provide a valid windows user of the default Historian server to which the Remote Collector Manager will connect.

Historian Server:

User Name:

Password:

Confirm Password:

Note: If you have created security groups or enabled strict client/collector authentication, provide the destination Historian credentials.
If the password changes, you must reinstall Remote Management Agents to reset the password.

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

Field	Description
Historian Server	Enter the Amazon Network Load Balancer (NLB) DNS. **Tip:** To find the NLB DNS: a. Access the EKS cluster on which you have deployed Proficy Historian for AWS. b. Access the EC2 instance. c. In the navigation pane, under **Load Balancing**, select **Load Balancers**. d. Select the load balancer for which you want to find the DNS. e. In the **Description** section, copy the DNS name.

Field	Description
User Name	Enter the username to connect to Proficy Historian for AWS.
Password	Enter the password to connect to Proficy Historian for AWS. **Tip:** For the default user, ihCloudHistAdmin, this is the value you entered in the **Password** field under **Proficy Authentication Configuration** when you created the stack.
Confirm Password	Re-enter the password.

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>`

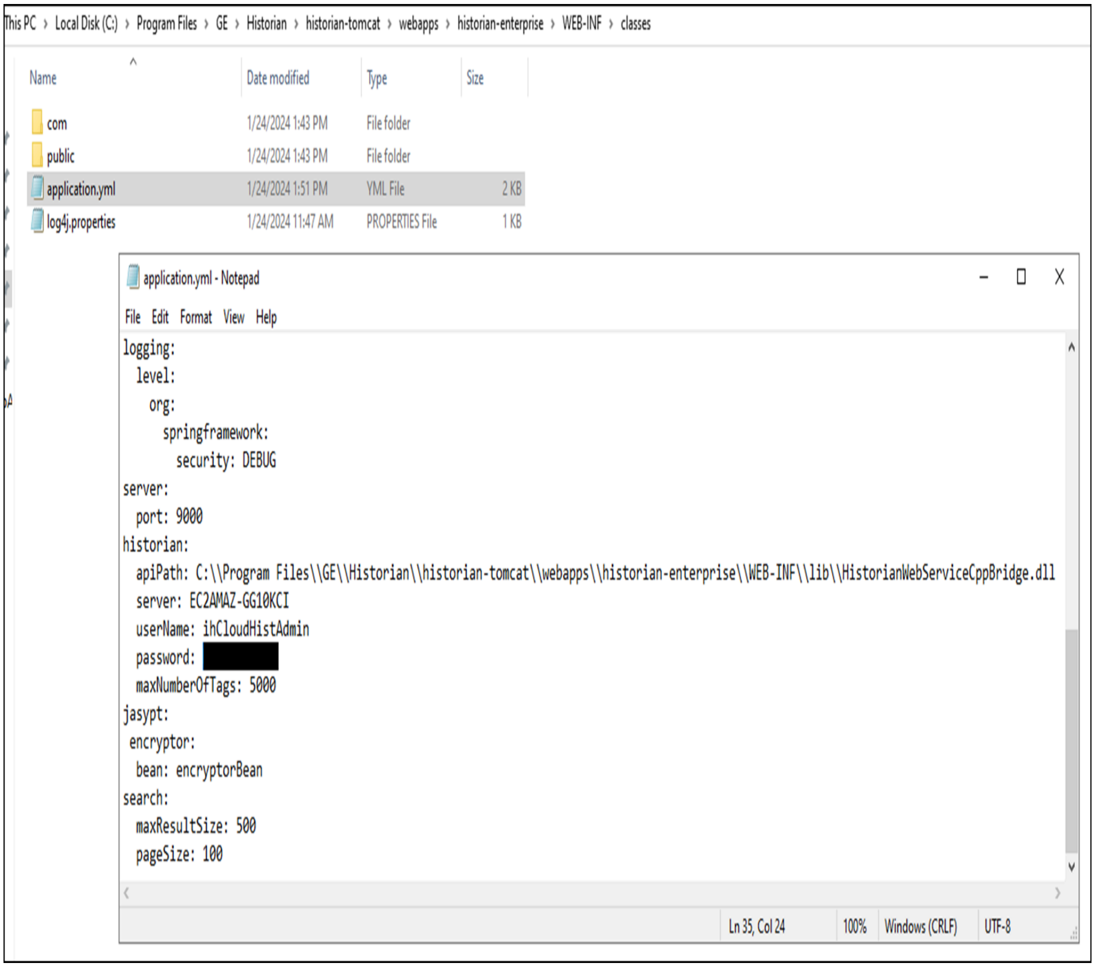
In addition, if iFIX and/or CIMPLICITY are installed on the same machine as the collectors, instances of the following collectors are created:

- 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. Enable trust for a client certificate for Configuration Hub.
3. Enable trust for a self-signed certificate on Chrome.
4. Import an issuer certificate.
5. If you are using Windows Configuration Hub to configure Cloud Historian, you need to:

a. Update the application.yml file located in the C:\Program Files\GE\Historian\historian-tomcat\webapps\historian-enterprise\WEB-INF\classes\application.yml folder with the Username and Password to connect to Cloud Historian. For example:



This PC › Local Disk (C:) › Program Files › GE › Historian › historian-tomcat › webapps › historian-enterprise › WEB-INF › classes

Name	Date modified	Type	Size
com	1/24/2024 1:43 PM	File folder	
public	1/24/2024 1:43 PM	File folder	
application.yml	1/24/2024 1:51 PM	YML File	2 KB
log4j.properties	1/24/2024 11:47 AM	PROPERTIES File	1 KB

```
logging:
  level:
    org:
      springframework:
        security: DEBUG
server:
  port: 9000
historian:
  apiPath: C:\\Program Files\\GE\\Historian\\historian-tomcat\\webapps\\historian-enterprise\\WEB-INF\\lib\\HistorianWebServiceCppBridge.dll
  server: EC2AMAZ-GG10KCI
  userName: ihCloudHistAdmin
  password: [redacted]
  maxNumberOfTags: 5000
jasypt:
 encryptor:
  bean: encryptorBean
search:
  maxResultSize: 500
  pageSize: 100
```

b. Restart the Historian Tomcat service.

c. Add the Cloud Historian system to the Configuration Hub by providing the Historian server name as NLB DNS.

You are now ready to use Configuration Hub.

Access Configuration Hub *(on page 97)*.

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. These steps only apply if using Windows-based Configuration Hub.

1. Uninstall Configuration Hub.
2. Set up Configuration Hub *(on page 75)*.

Access Configuration Hub

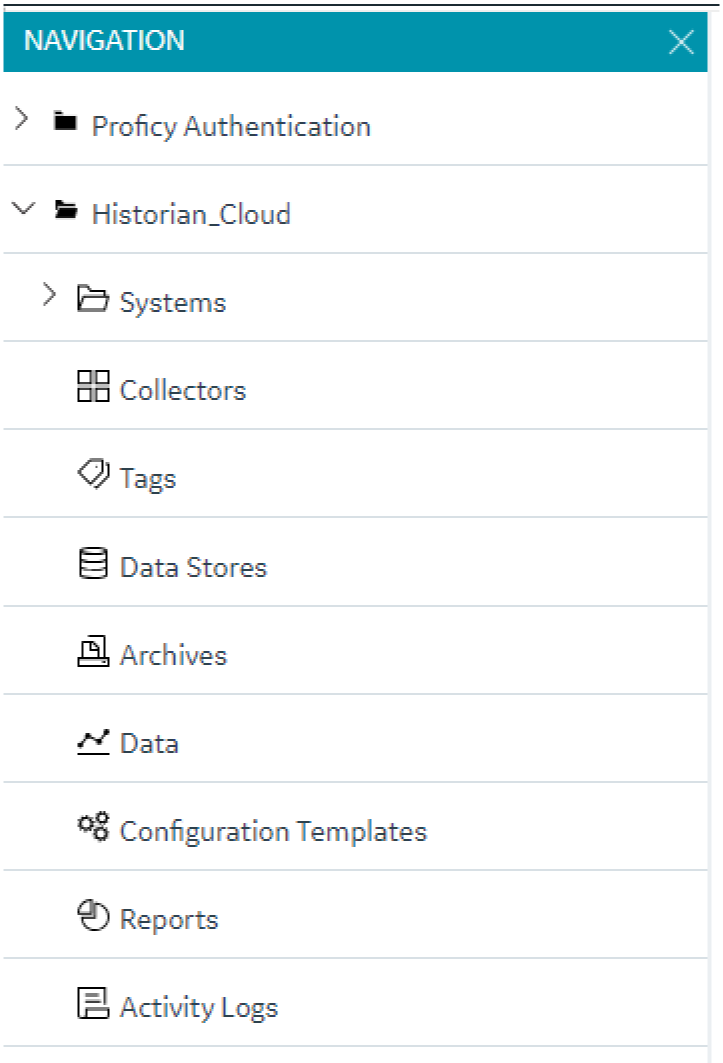
Perform the tasks outlined in About Setting up Configuration Hub *(on page 75)*.

1. When using Cloud Historian with Windows Configuration Hub (with web-based clients), or when using containerized Configuration Hub (deployed with Cloud Historian) access thru URL.
   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.

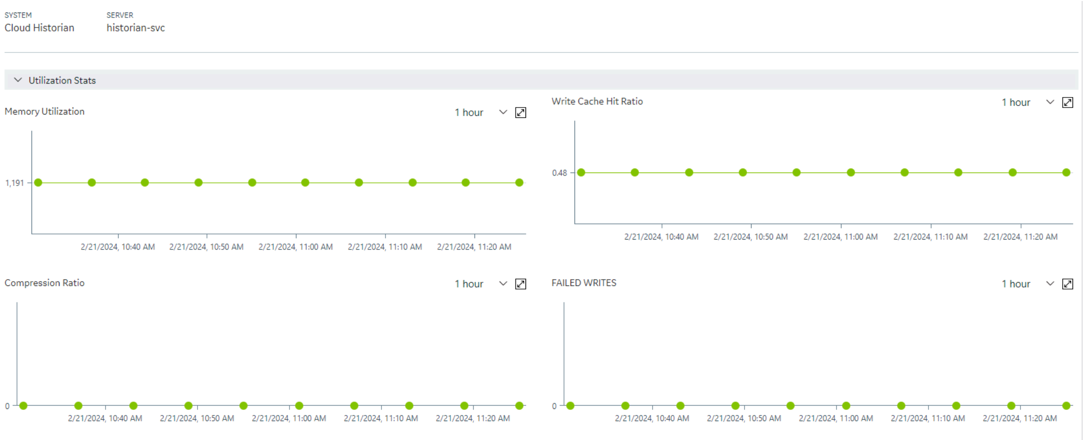
   > **Note:**
   > For instructions on setting up authentication, refer to https://www.ge.com/digital/documentation/confighub/version2024/t_authentication_setup.html

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. The default user name for Configuration Hub is: ihCloudHistAdmin (user name is case sensitive); the password was defined during deployment.
   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 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.

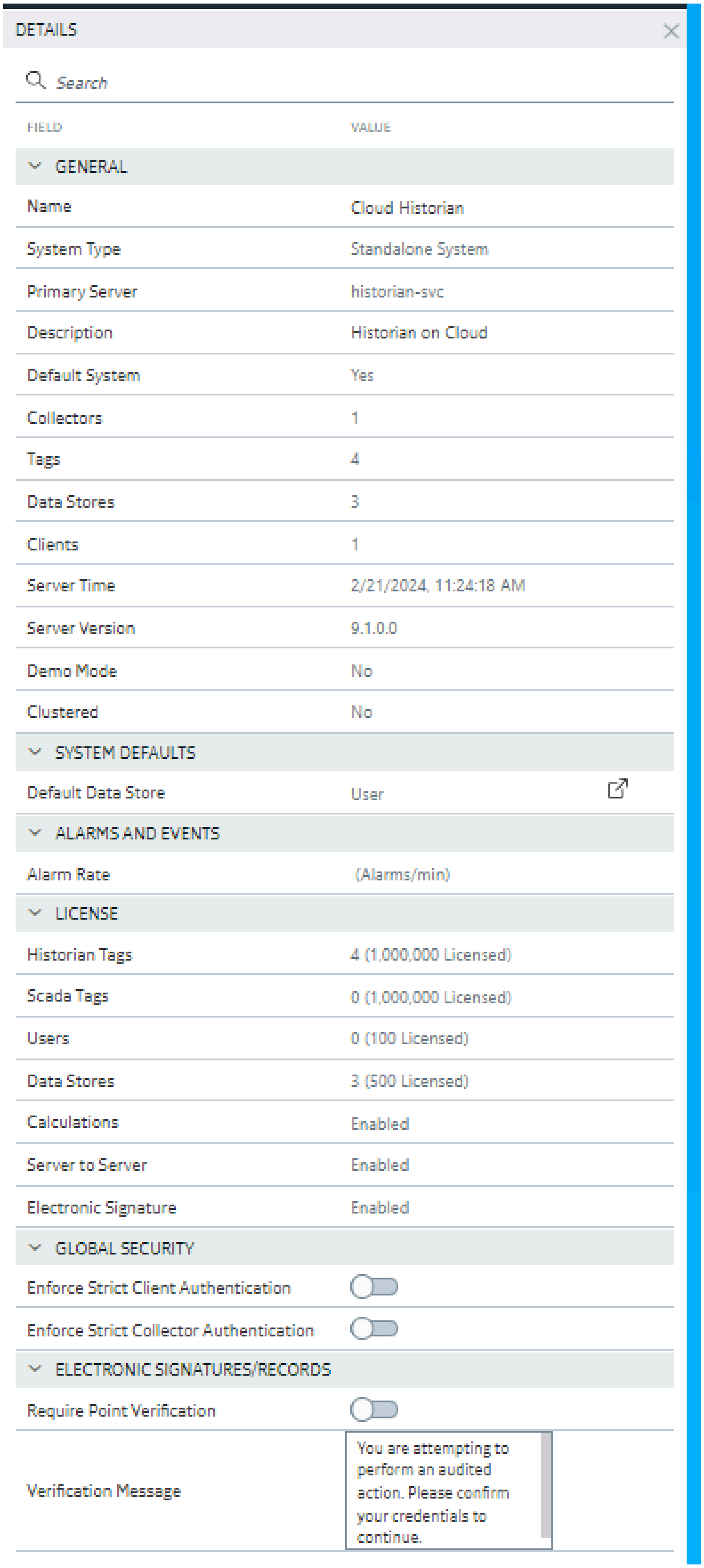


SYSTEM
Cloud Historian

SERVER
historian-svc

Utilization Stats

- **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 using the **Details** section.



DETAILS

Search

FIELD	VALUE
GENERAL	
Name	Cloud Historian
System Type	Standalone System
Primary Server	historian-svc
Description	Historian on Cloud
Default System	Yes
Collectors	1
Tags	4
Data Stores	3
Clients	1
Server Time	2/21/2024, 11:24:18 AM
Server Version	9.1.0.0
Demo Mode	No
Clustered	No
SYSTEM DEFAULTS	
Default Data Store	User
ALARMS AND EVENTS	
Alarm Rate	(Alarms/min)
LICENSE	
Historian Tags	4 (1,000,000 Licensed)
Scada Tags	0 (1,000,000 Licensed)
Users	0 (100 Licensed)
Data Stores	3 (500 Licensed)
Calculations	Enabled
Server to Server	Enabled
Electronic Signature	Enabled
GLOBAL SECURITY	
Enforce Strict Client Authentication	○
Enforce Strict Collector Authentication	○
ELECTRONIC SIGNATURES/RECORDS	
Require Point Verification	○
Verification Message	You are attempting to perform an audited action. Please confirm your credentials to continue.

Set up a stand-alone system.

Historian Plugin Management in Configuration Hub

About Historian Plugin Management in Configuration Hub

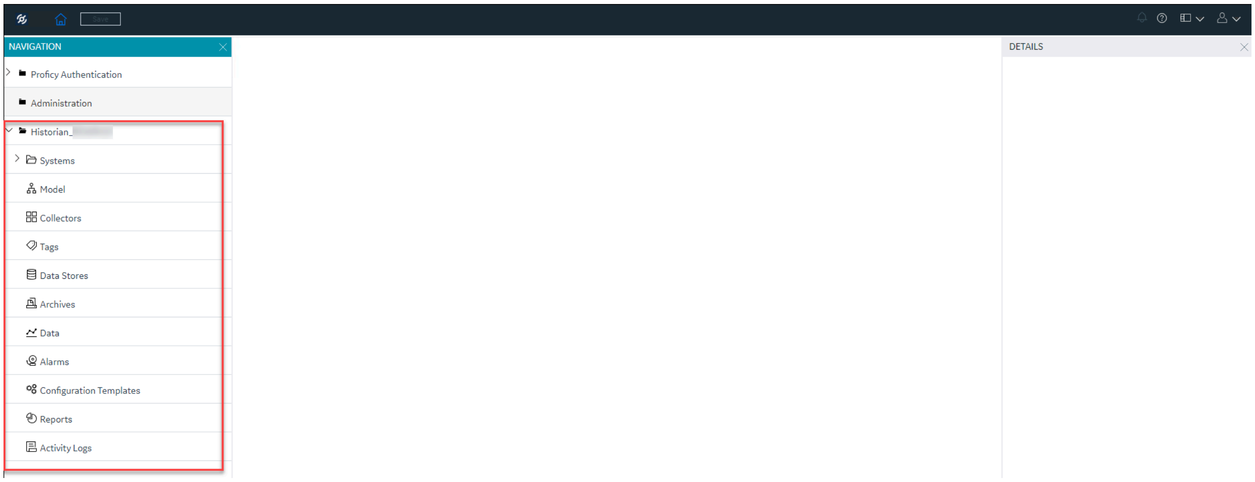
The Historian plugin in Configuration Hub enables you to perform several tasks like monitor, supervise, retrieve, and control gathering functions from a server, client, or one or more remote nodes. However, it operates as a web-based application.

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 105)* 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 108)*, unregister a plugin *(on page 111)*, and if you want to register it at a later stage, you can register the plugin *(on page 109)* again.

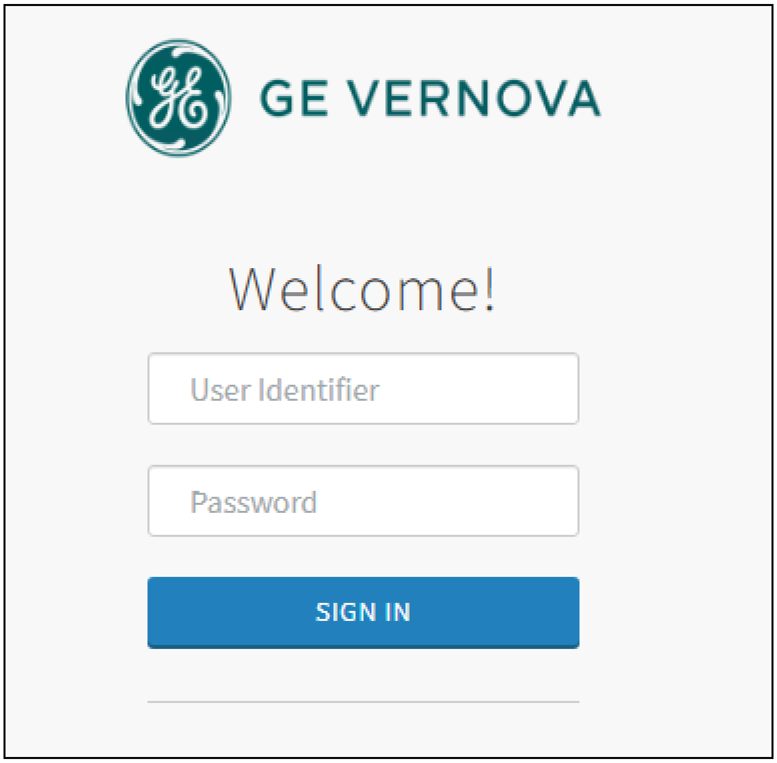


View Historian Node Details

After you added a Historian node, you can view the Historian node-specific certificate and license details.

1. Double-click the Configuration Hub icon on your desktop ( ).

   The Configuration Hub login page appears.

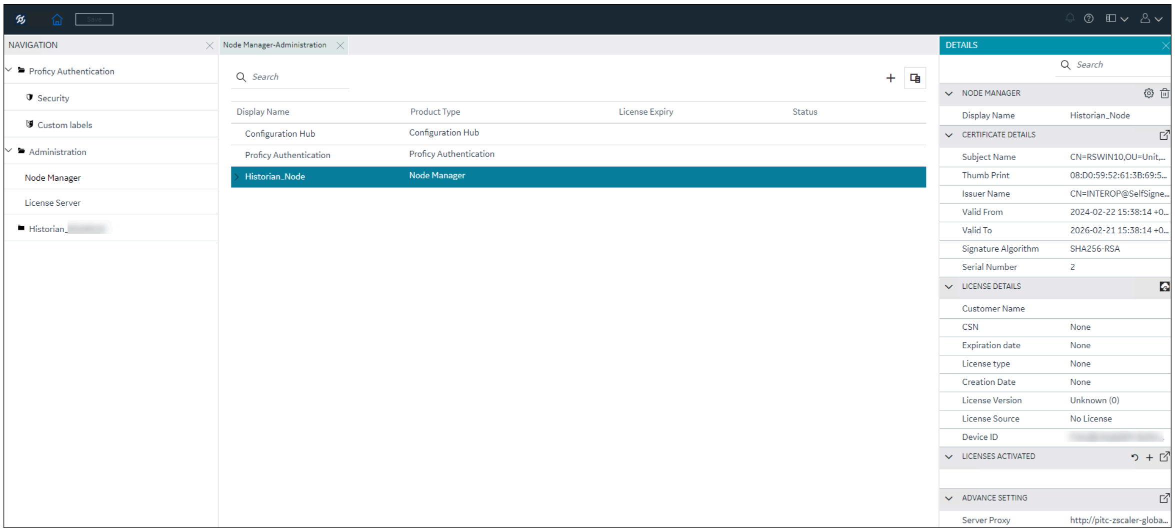
   

2. Login with the default user credentials. That is, <hostname>.admin.

   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**.

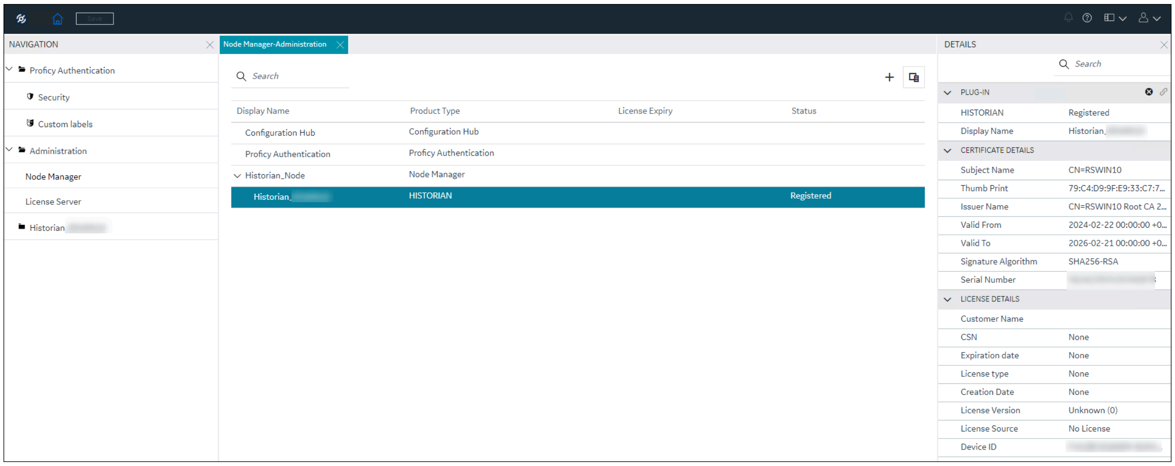
   The Node Manager administration page appears, listing the available Historian node and the plugin.
4. To view the Historian node-specific details, select the node. The node-specific details appear in the right-side section.



Display Name	Product Type	License Expiry	Status
Configuration Hub	Configuration Hub		
Proficy Authentication	Proficy Authentication		
Historian_Node	Node Manager		

NODE MANAGER	
Display Name	Historian_Node
CERTIFICATE DETAILS	
Subject Name	CN=RSWIN10,OU=Unit,...
Thumb Print	08:D0:59:52:61:3B:69:5...
Issuer Name	CN=INTEROP@SelfSigne...
Valid From	2024-02-22 15:38:14 +0...
Valid To	2026-02-21 15:38:14 +0...
Signature Algorithm	SHA256-RSA
Serial Number	2
LICENSE DETAILS	
Customer Name	
CSN	None
Expiration date	None
License type	None
Creation Date	None
License Version	Unknown (0)
License Source	No License
Device ID	[redacted]
LICENSES ACTIVATED	
ADVANCE SETTING	
Server Proxy	http://pitc-zscaler-globa...

5. To view the plugin-specific details, expand the node and select the plugin. The plugin-specific details appear in the right-side section.



Display Name	Product Type	License Expiry	Status
Configuration Hub	Configuration Hub		
Proficy Authentication	Proficy Authentication		
Historian_Node	Node Manager		
Historian_[redacted]	HISTORIAN		Registered

PLUG-IN	
HISTORIAN	Registered
Display Name	Historian_[redacted]
CERTIFICATE DETAILS	
Subject Name	CN=RSWIN10
Thumb Print	79:C4:D9:9F:E9:33:C7:7...
Issuer Name	CN=RSWIN10 Root CA 2...
Valid From	2024-02-22 00:00:00 +0...
Valid To	2026-02-21 00:00:00 +0...
Signature Algorithm	SHA256-RSA
Serial Number	[redacted]
LICENSE DETAILS	
Customer Name	
CSN	None
Expiration date	None
License type	None
Creation Date	None
License Version	Unknown (0)
License Source	No License
Device ID	[redacted]

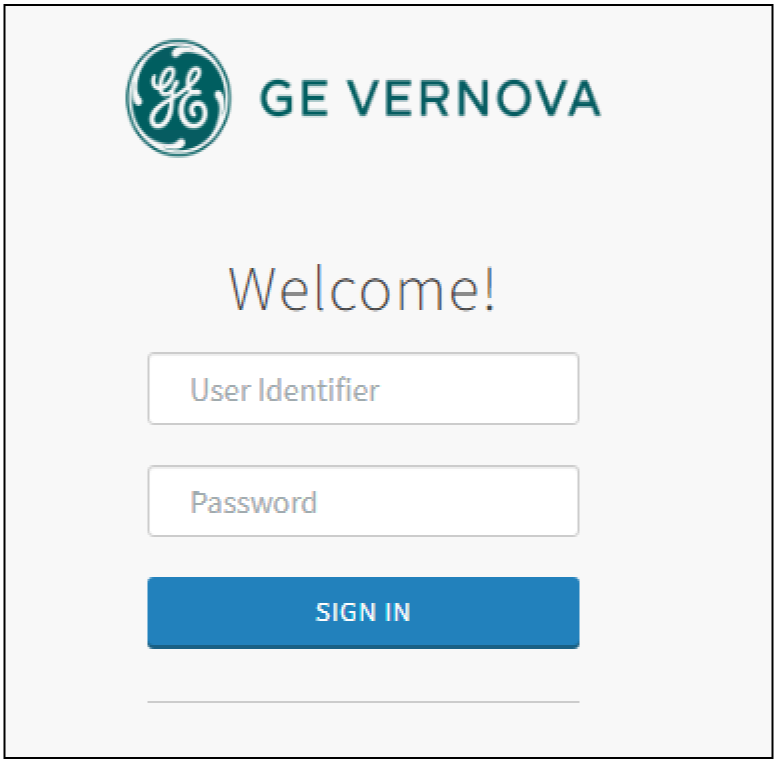
If you have the needed permission, you can perform additional certificate and node management tasks. For more information, refer to **Administration Plugin** in the Configuration Hub help. For more latest updates, refer to the online version of the Historian 2024 help.

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.

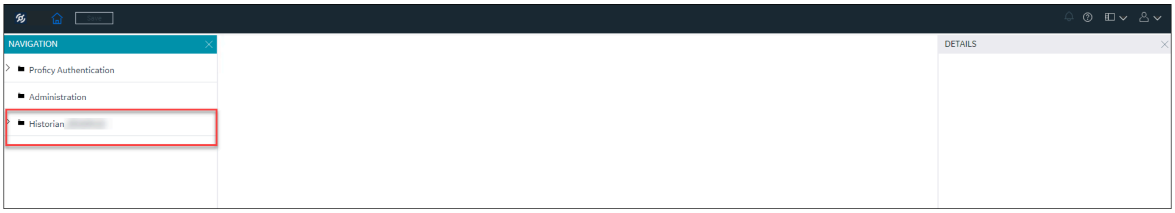
1. Double-click the Configuration Hub icon on your desktop ().

   The Configuration Hub login page appears.

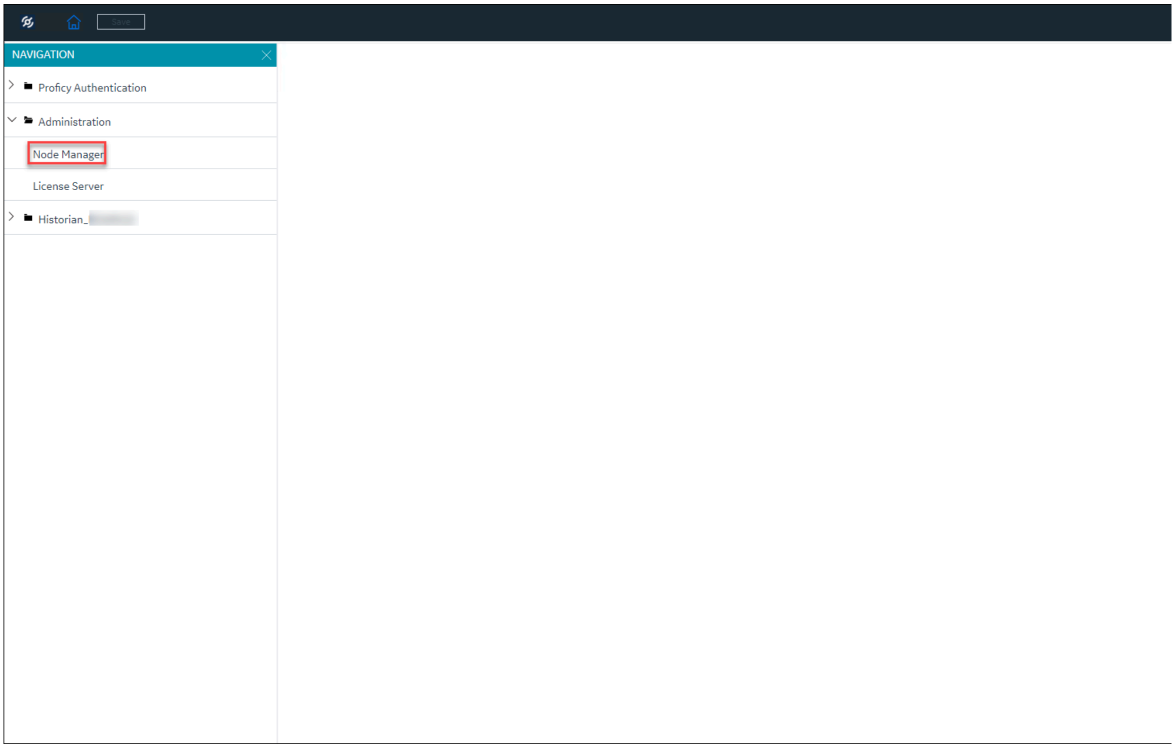
   

2. Login with the default user credentials. That is <hostname>.admin.

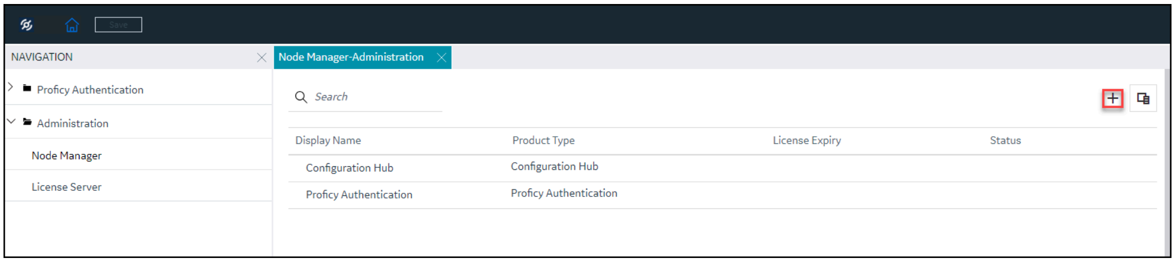
   The configuration hub application appears, listing the Historian plugin in the **NAVIGATION** pane.

3. In the **NAVIGATION** pane, select and expand **Administration**, and then select **Node Manager**.



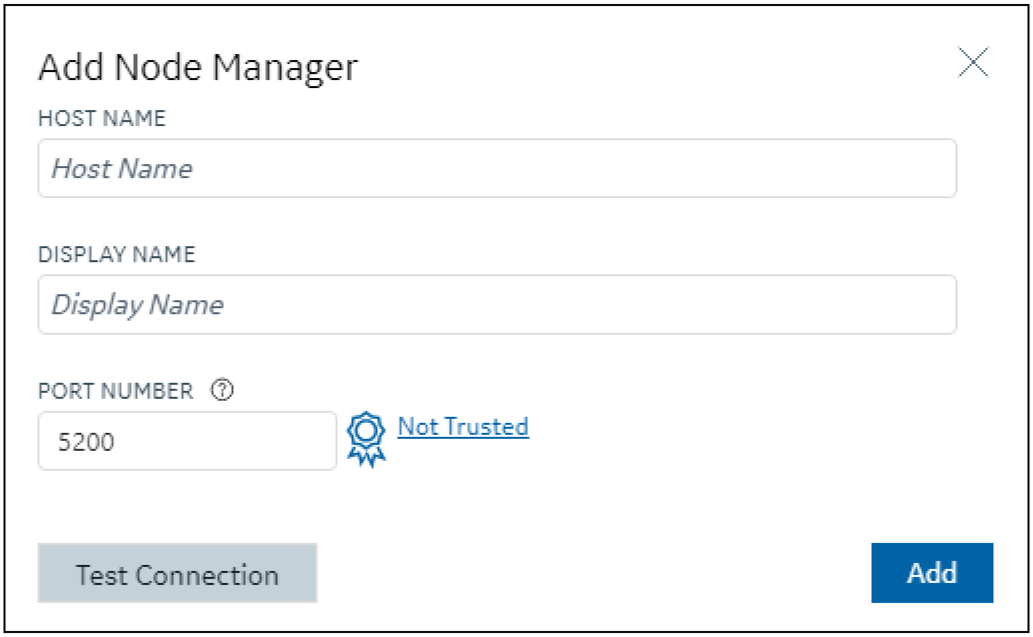
The node manager administration page appears.



Display Name	Product Type	License Expiry	Status
Configuration Hub	Configuration Hub		
Proficy Authentication	Proficy Authentication		

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**.

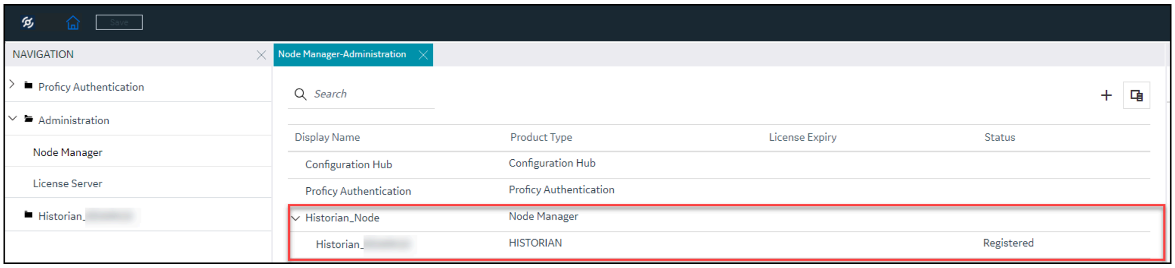
   The **Certificate Details** window appears, listing the certificate information.
9. Read the certificate details and if you trust, select **Trust**.

   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.



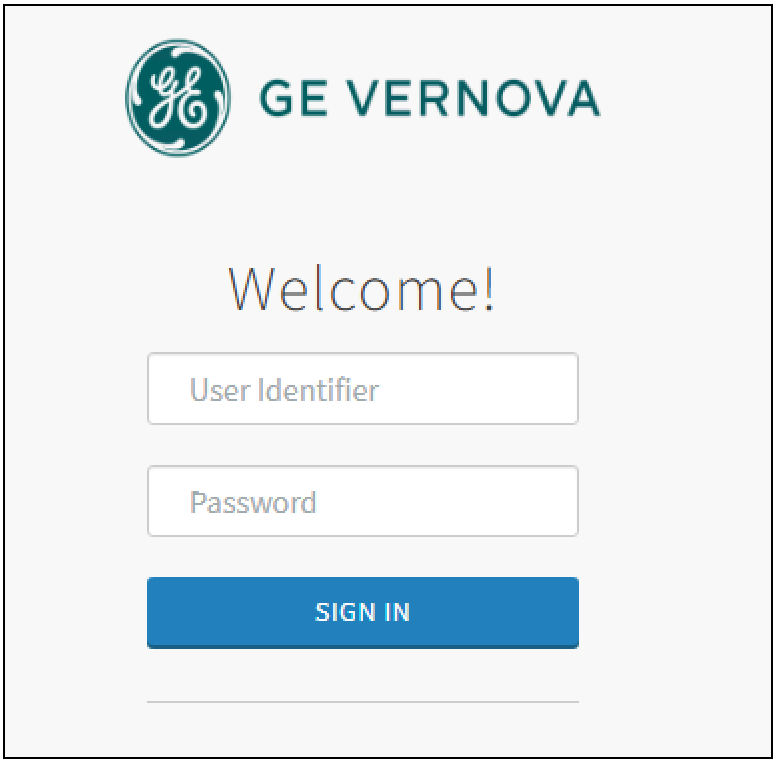
Display Name	Product Type	License Expiry	Status
Configuration Hub	Configuration Hub		
Proficy Authentication	Proficy Authentication		
Historian_Node	Node Manager		
Historian_[redacted]	HISTORIAN		Registered

In rare cases, for some reason, if the plugin is not registered, you can register the plugin *(on page 109)*.

Modify a Historian Plugin Display Name

1. Double-click the Configuration Hub icon on your desktop ( ).

   The Configuration Hub login page appears.

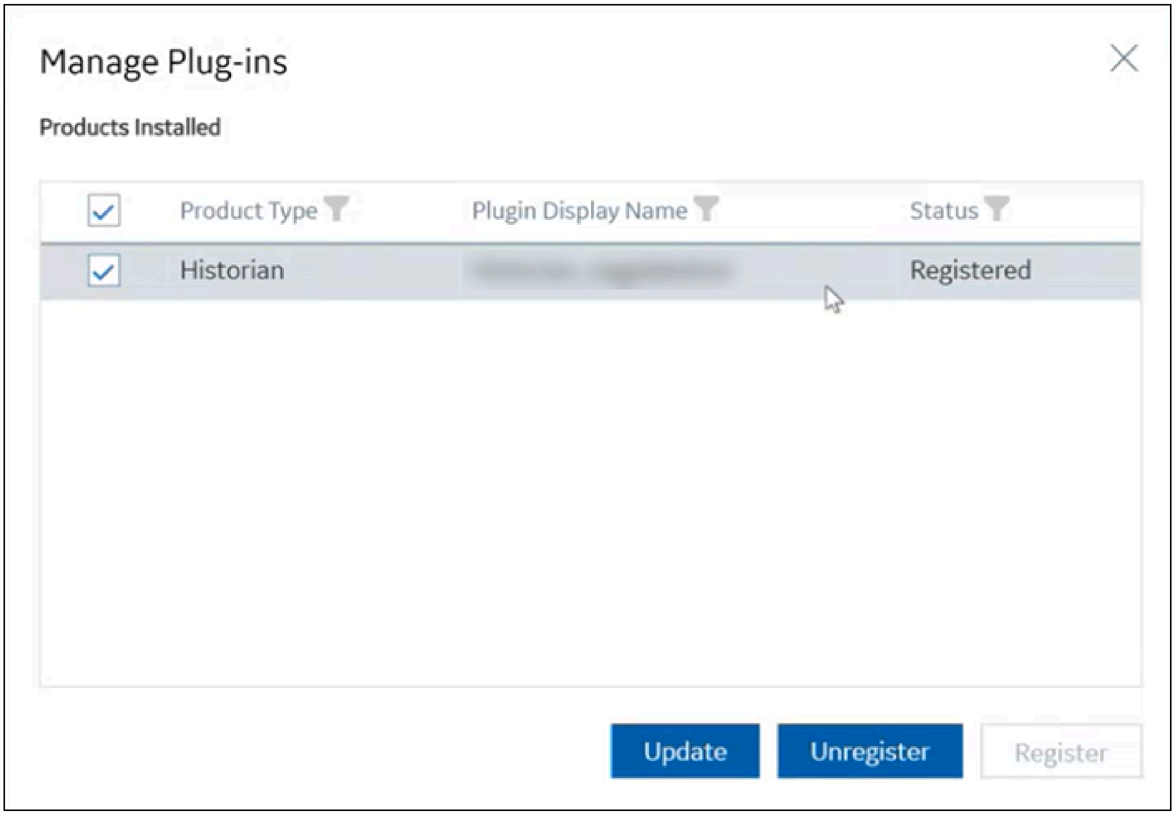
   

2. Login with the default user credentials. That is <hostname>.admin.

   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.



Manage Plug-ins

Products Installed

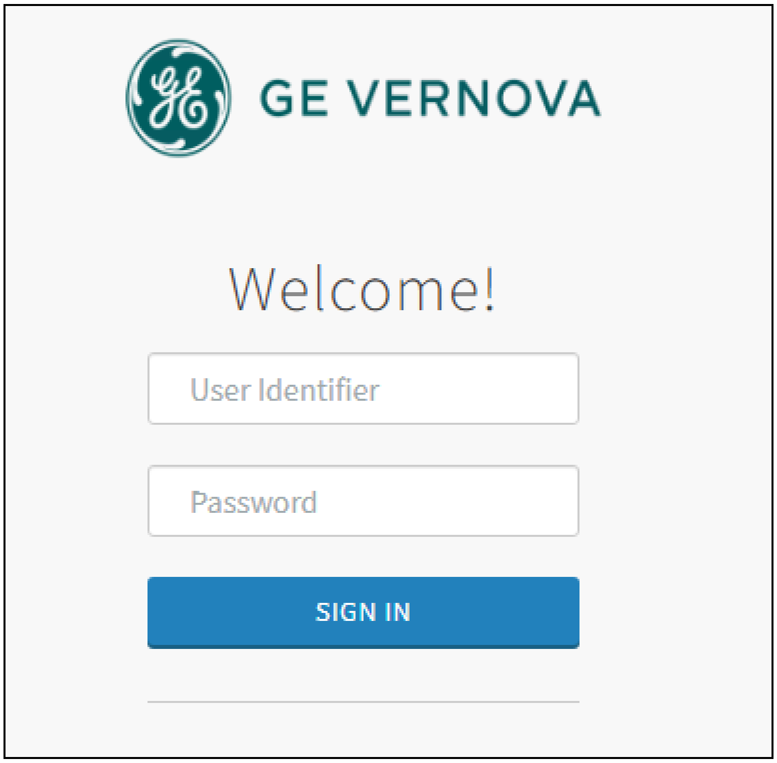
☑	Product Type	Plugin Display Name	Status
☑	Historian	[redacted]	Registered

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 105)* 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.

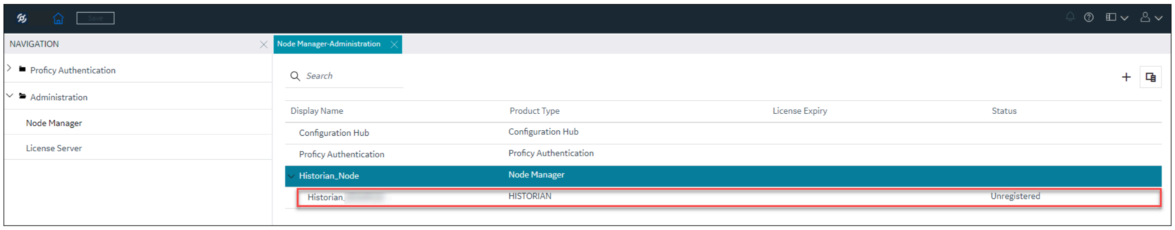


2. Login with the default user credentials. That is <hostname>.admin.

   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.



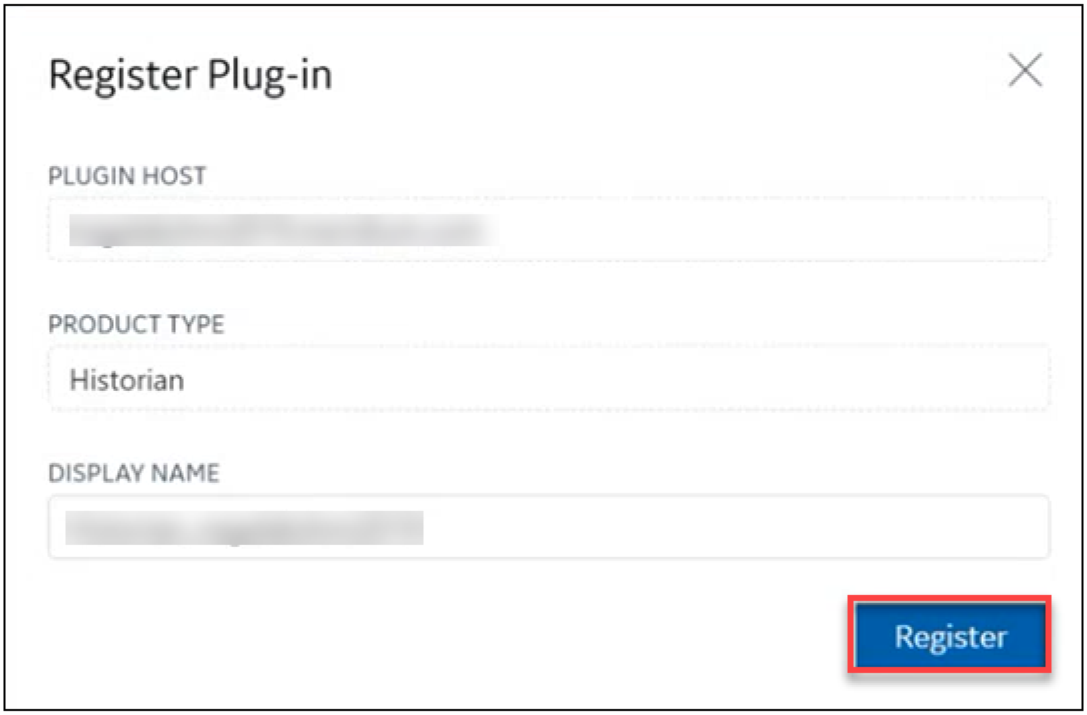
Display Name	Product Type	License Expiry	Status
Configuration Hub	Configuration Hub		
Proficy Authentication	Proficy Authentication		
Historian_Node	Node Manager		
Historian_[redacted]	HISTORIAN		Unregistered

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.



Register Plug-in

PLUGIN HOST

PRODUCT TYPE

Historian

DISPLAY NAME

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**.

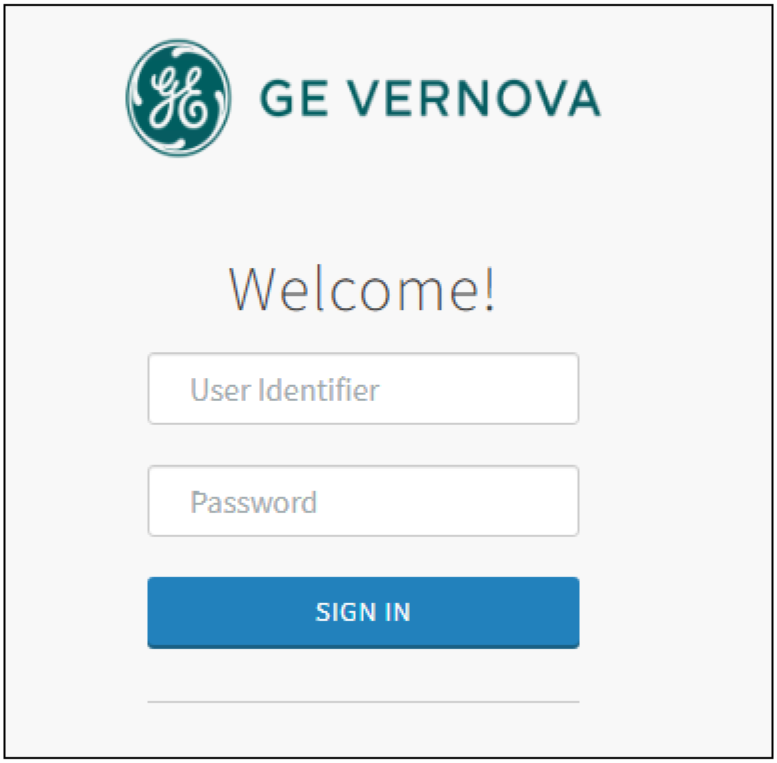
   The plugin gets registered and added in the **NAVIGATION** pane.
7. For the plugin to work, you must refresh the browser or log out and log in again to restart Configuration Hub.
8. Now try accessing the plugin.

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.

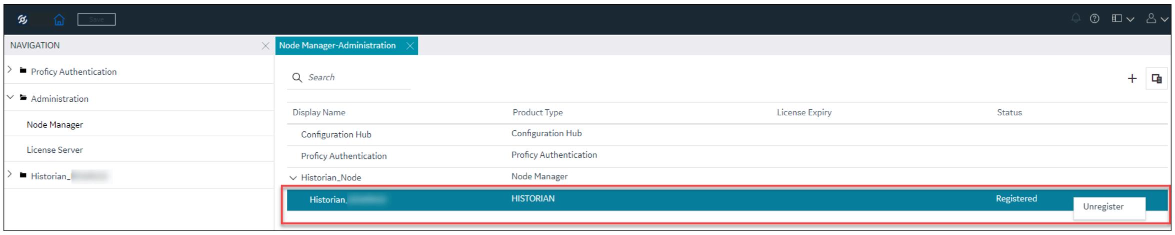


2. Login with the default user credentials. That is <hostname>.admin.

   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.

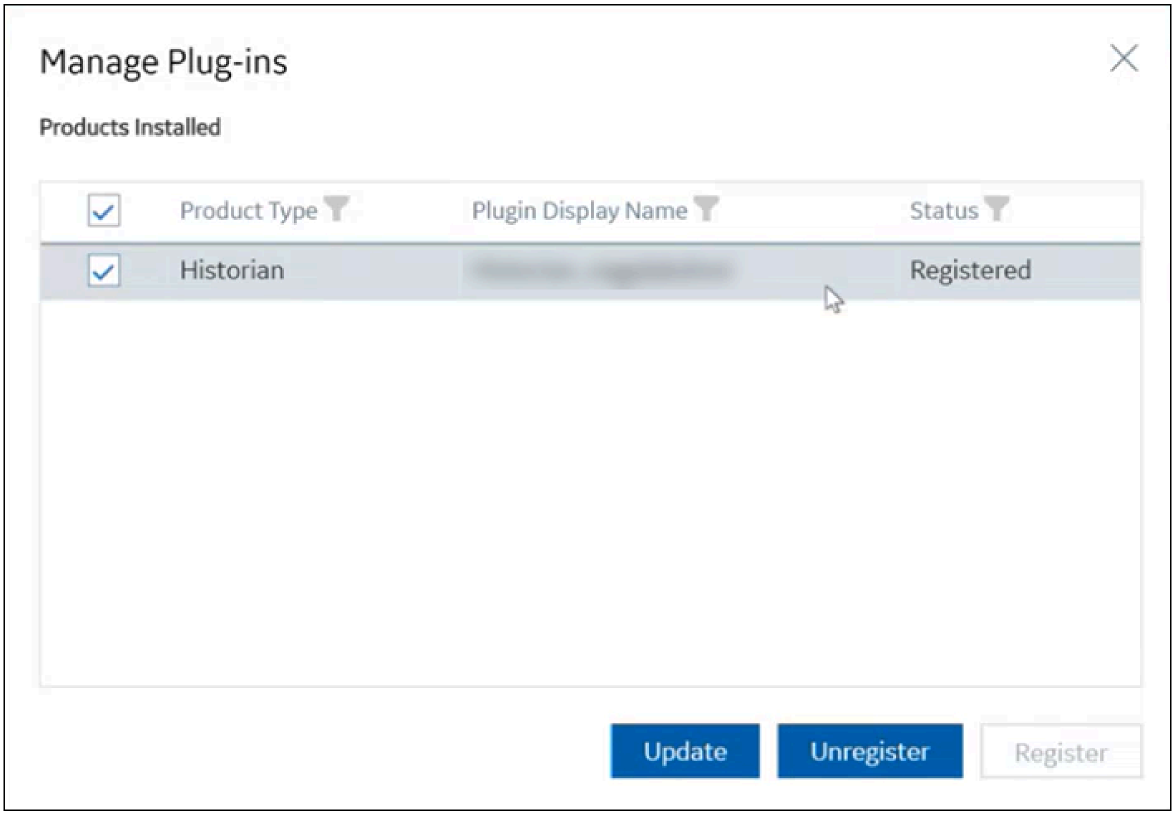


Display Name	Product Type	License Expiry	Status
Configuration Hub	Configuration Hub		
Proficy Authentication	Proficy Authentication		
Historian_Node	Node Manager		
Historian_	HISTORIAN		Registered

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.



Manage Plug-ins

Products Installed

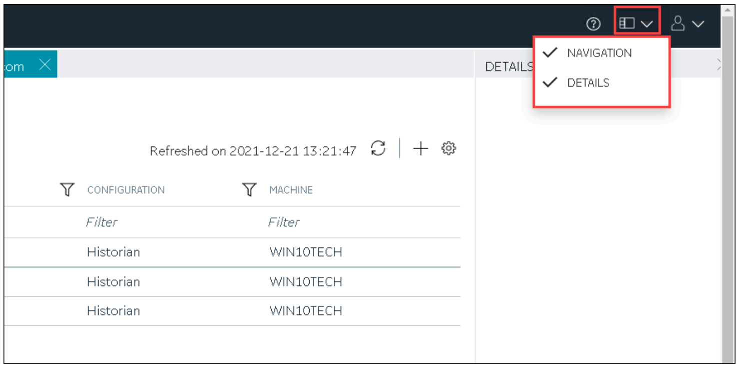
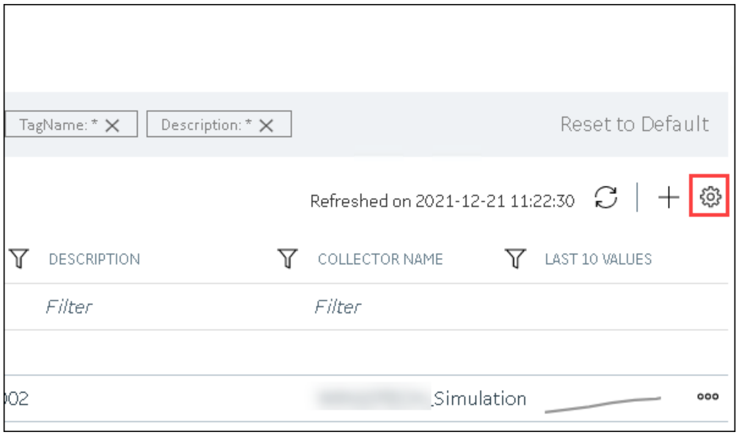
☑	Product Type	Plugin Display Name	Status
☑	Historian	[redacted]	Registered

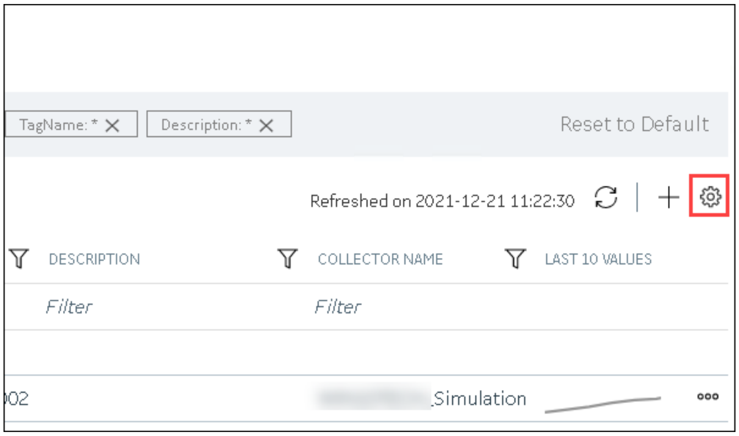
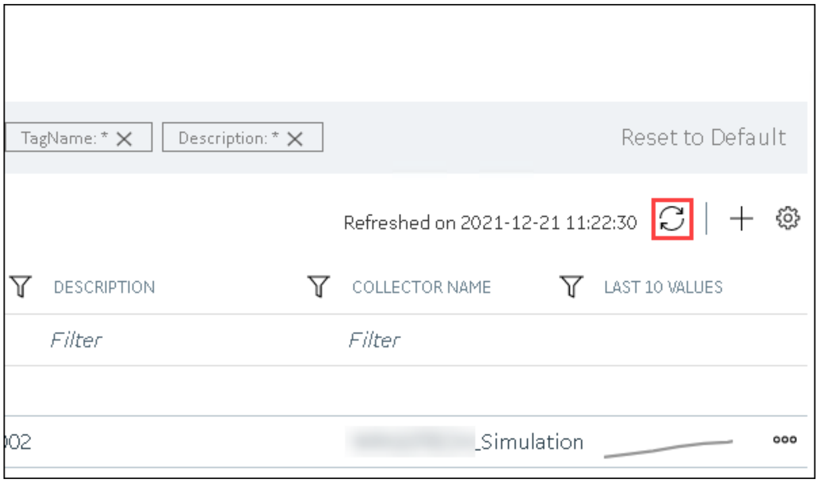
5. Select the plugin as needed.
6. Select **Unregister**.
   The plugin gets unregistered.

   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

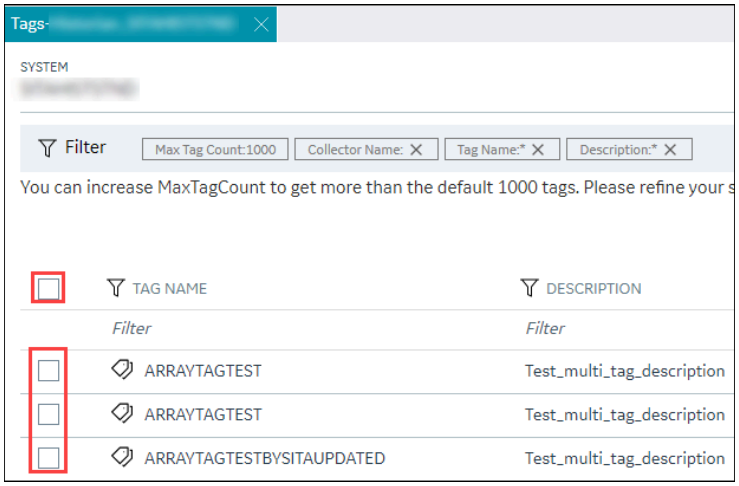
Task	Procedure
Show or hide the **Navigation** or the **Details** section.	1. In the upper-right corner of the page, select ◧. 2. Select the check boxes for the sections that you want to show.  NAVIGATION DETAILS DETAILS Refreshed on 2021-12-21 13:21:47 CONFIGURATION MACHINE Filter Filter Historian WIN10TECH Historian WIN10TECH Historian WIN10TECH
Show or hide columns in a table. **Note:** You cannot hide some of the columns (for example, the **COLLECTOR NAME** column).	1. In the upper-right corner of the table, select ⚙.  TagName: * ✕ Description: * ✕ Reset to Default Refreshed on 2021-12-21 11:22:30 DESCRIPTION COLLECTOR NAME LAST 10 VALUES Filter Filter 02 Simulation The **Table Settings** window appears. 2. Select the check boxes in the **SHOW COLUMN** column, and then select **Apply**.

Task	Procedure
Reorder columns in a table. **Note:** You cannot reorder some of the columns.	1. In the upper-right corner of the table, select ⚙.  TagName: * ✕ Description: * ✕ Reset to Default Refreshed on 2021-12-21 11:22:30 DESCRIPTION COLLECTOR NAME LAST 10 VALUES Filter Filter 02 _Simulation The **Table Settings** window appears. 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 ⟳.  TagName: * ✕ Description: * ✕ Reset to Default Refreshed on 2021-12-21 11:22:30 DESCRIPTION COLLECTOR NAME LAST 10 VALUES Filter Filter 02 _Simulation

<table>
<tr><th>Task</th><th>Procedure</th></tr>
<tr><td>Select multiple tags or clear the selection.</td><td>

- 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 of table header.



Tags-[redacted]

SYSTEM
[redacted]

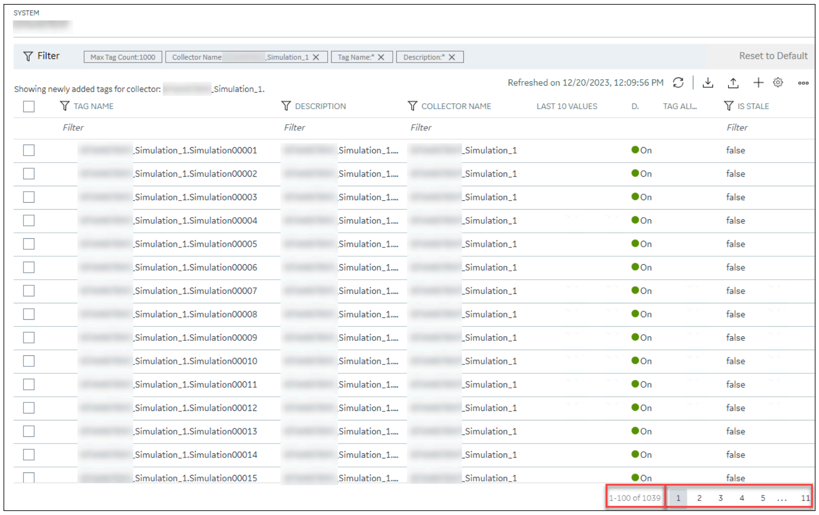
Filter Max Tag Count:1000 Collector Name: ✕ Tag Name:* ✕ Description:* ✕

You can increase MaxTagCount to get more than the default 1000 tags. Please refine your s

<table>
<tr><th>□</th><th>TAG NAME</th><th>DESCRIPTION</th></tr>
<tr><td></td><td>Filter</td><td>Filter</td></tr>
<tr><td>□</td><td>ARRAYTAGTEST</td><td>Test_multi_tag_description</td></tr>
<tr><td>□</td><td>ARRAYTAGTEST</td><td>Test_multi_tag_description</td></tr>
<tr><td>□</td><td>ARRAYTAGTESTBYSITAUPDATED</td><td>Test_multi_tag_description</td></tr>
</table>

</td></tr>
<tr><td>Navigate through grids in a page</td><td>

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.



SYSTEM
[redacted]

Filter Max Tag Count:1000 Collector Name[redacted]_Simulation_1 ✕ Tag Name:* ✕ Description:* ✕ Reset to Default

Showing newly added tags for collector: [redacted]_Simulation_1.

Refreshed on 12/20/2023, 12:09:56 PM

<table>
<tr><th>□</th><th>TAG NAME</th><th>DESCRIPTION</th><th>COLLECTOR NAME</th><th>LAST 10 VALUES</th><th>D.</th><th>TAG ALI...</th><th>IS STALE</th></tr>
<tr><td></td><td>Filter</td><td>Filter</td><td>Filter</td><td></td><td></td><td></td><td>Filter</td></tr>
<tr><td>□</td><td>[redacted]_Simulation_1.Simulation00001</td><td>[redacted]Simulation_1....</td><td>[redacted]_Simulation_1</td><td></td><td>● On</td><td></td><td>false</td></tr>
<tr><td>□</td><td>[redacted]_Simulation_1.Simulation00002</td><td>[redacted]Simulation_1....</td><td>[redacted]_Simulation_1</td><td></td><td>● On</td><td></td><td>false</td></tr>
<tr><td>□</td><td>[redacted]_Simulation_1.Simulation00003</td><td>[redacted]Simulation_1....</td><td>[redacted]_Simulation_1</td><td></td><td>● On</td><td></td><td>false</td></tr>
<tr><td>□</td><td>[redacted]_Simulation_1.Simulation00004</td><td>[redacted]Simulation_1....</td><td>[redacted]_Simulation_1</td><td></td><td>● On</td><td></td><td>false</td></tr>
<tr><td>□</td><td>[redacted]_Simulation_1.Simulation00005</td><td>[redacted]Simulation_1....</td><td>[redacted]_Simulation_1</td><td></td><td>● On</td><td></td><td>false</td></tr>
<tr><td>□</td><td>[redacted]_Simulation_1.Simulation00006</td><td>[redacted]Simulation_1....</td><td>[redacted]_Simulation_1</td><td></td><td>● On</td><td></td><td>false</td></tr>
<tr><td>□</td><td>[redacted]_Simulation_1.Simulation00007</td><td>[redacted]Simulation_1....</td><td>[redacted]_Simulation_1</td><td></td><td>● On</td><td></td><td>false</td></tr>
<tr><td>□</td><td>[redacted]_Simulation_1.Simulation00008</td><td>[redacted]Simulation_1....</td><td>[redacted]_Simulation_1</td><td></td><td>● On</td><td></td><td>false</td></tr>
<tr><td>□</td><td>[redacted]_Simulation_1.Simulation00009</td><td>[redacted]Simulation_1....</td><td>[redacted]_Simulation_1</td><td></td><td>● On</td><td></td><td>false</td></tr>
<tr><td>□</td><td>[redacted]_Simulation_1.Simulation00010</td><td>[redacted]Simulation_1....</td><td>[redacted]_Simulation_1</td><td></td><td>● On</td><td></td><td>false</td></tr>
<tr><td>□</td><td>[redacted]_Simulation_1.Simulation00011</td><td>[redacted]Simulation_1....</td><td>[redacted]_Simulation_1</td><td></td><td>● On</td><td></td><td>false</td></tr>
<tr><td>□</td><td>[redacted]_Simulation_1.Simulation00012</td><td>[redacted]Simulation_1....</td><td>[redacted]_Simulation_1</td><td></td><td>● On</td><td></td><td>false</td></tr>
<tr><td>□</td><td>[redacted]_Simulation_1.Simulation00013</td><td>[redacted]Simulation_1....</td><td>[redacted]_Simulation_1</td><td></td><td>● On</td><td></td><td>false</td></tr>
<tr><td>□</td><td>[redacted]_Simulation_1.Simulation00014</td><td>[redacted]Simulation_1....</td><td>[redacted]_Simulation_1</td><td></td><td>● On</td><td></td><td>false</td></tr>
<tr><td>□</td><td>[redacted]_Simulation_1.Simulation00015</td><td>[redacted]Simulation_1....</td><td>[redacted]_Simulation_1</td><td></td><td>● On</td><td></td><td>false</td></tr>
</table>

1-100 of 1039 1 2 3 4 5 ... 11

In addition to this, you can also see a grid's count (both total and selected).

</td></tr>
</table>

Installing 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® 2021 (32 & 64 bit)
- Microsoft® Excel® 2019 (32 & 64 bit)
- Microsoft® Excel® 2016 (32 & 64 bit)
- Microsoft® Excel® 2013 (32 & 64 bit)

This topic describes how to install Excel Add-In using the installer. You can also install it at a command prompt *(on page 117)*.

1. Run the `InstallLauncher.exe` file. Contact the AWS support team for the installer.
2. Select **Historian Excel Add-in**.
   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 993)*.

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
2. Install Excel Add-in using the installer *(on page 117)* 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 during the installation. You can then use this template to install Excel Add-in at a command prompt on other machines.

This topic describes how to install the Excel Addin for Historian at a command prompt. You can also install it using the installer *(on page 117)*.

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 993)*.

Implementing Security

Default Security Groups

This topic provides a list of the default security groups created in Historian, along with the default user, ihCloudHistAdmin, for the ih_security_admins group. The password for this user is the one you enter in the **Proficy Authentication Configuration** field while deploying Proficy Historian for AWS.

ih_security_admins

Historian power security users. Security administrators have rights to all Historian functions. By default, a user named ihCloudHistAdmin is added in this group.

ih_collector_admins

Allowed to add collector instances and change their destination.

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_archive_admins

Allowed to create, modify, and remove archives.

ih_unaudited_writers

Allowed to write data without creating any messages.

ih_unaudited_logins

Allowed to connect to Data Archiver 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.

The following table provides the types of user groups you must create based on your requirement.

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
Manage tags	X						X	
Create archive	X					X		
Read data	X				X			
Write data (unaudited)	X	X	X					
Write data (audited)	X			X				
Modify data	X	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
Update tag security	X							
Migrate	X							
Login connection messages	X	X		X	X	X	X	X
Recalculate data	X		X	X				X

Note:
Regardless of the security group to which a user belongs, the user has full privileges to the Web Admin console.

For instructions on creating and managing users, refer to Managing Users and Groups *(on page 120)*.

Managing Users and Groups

Historian provides default security groups *(on page 118)* and a user, ihCloudHistAdmin, for the ih_security_admins group. This topic describes how to create more users and add them to groups. You can also delete a user or remove the user from a group.

1. Access the folder containing the `uaa_config_tool.exe` file. It is provided with the collectors installer. After you install collectors, it will be available in the `C:\Program Files\GE Digital \Historian Cloud Config` folder by default.
2. To create a user, run the following command:

```
uaa_config_tool.exe add_user

-u <username>

-p <password>

-s <admin password>

-t https://<NLB DNS>:9090
```

For *<admin password>*, enter the password that you provided in the **Proficy Authentication Configuration** field while deploying Proficy Historian for AWS.

For *<password>*, enter a value that contains:

- Minimum eight characters
- At least one each of uppercase and lowercase letters
- At least one number
- At least one special character

> **Tip:**
>
> To find the NLB DNS:
>
> a. Access the EKS cluster on which you have deployed Proficy Historian for AWS.
> b. Access the EC2 instance.
> c. In the navigation pane, under **Load Balancing**, select **Load Balancers**.
> d. Select the load balancer for which you want to find the DNS.
> e. In the **Description** section, copy the DNS name.

3. To add a user to a group, run the following command:

```
uaa_config_tool.exe add_user_to_group

-g <group name>

-u <username>

-p <password>

-s <admin password>

-t https://<NLB DNS>:9090
```

where *<admin password>* is the password that you provided in the **Proficy Authentication Configuration** field while deploying Proficy Historian for AWS.

For *<password>*, enter a value that contains:

- Minimum eight characters
- At least one each of uppercase and lowercase letters
- At least one number
- At least one special character

> **Tip:**
>
> For a list of default security groups, refer to Default Security Groups *(on page 118)*.

4. To remove a user from a group, run the following command:

```
uaa_config_tool.exe remove_user_to_group

-g <group name>

-u <username>

-t https://<NLB DNS>:9090
```

5. To delete a user, run the following command:

```
uaa_config_tool.exe remove_user

-g <group name>

-u <username>

-s <admin password>

-t https://<NLB DNS>:9090
```

Chapter 5. Migrate Data

Migrate Data from On-Premises Historian to Proficy Historian for AWS

1. Deploy Proficy Historian for AWS *(on page 43)*.
2. Install collectors *(on page)*.

This topic describes how to migrate data from an on-premise Historian server to Proficy Historian for AWS.

If your network bandwidth is high and the amount of data to migrate is high or low, you can use AWS Transfer family.

If your network bandwidth is low and/or the amount of data to migrate is high, you can choose any of the following options to send data using a physical device:

- AWS Snowcone
- AWS Snowball
- AWS Snowmobile

The data is then migrated into an S3 bucket. You must then migrate it to EFS. For information on which option to choose, refer to https://aws.amazon.com/snow/.

Use the links below for more information:

1. Migrate Data Using an AWS Transfer Family *(on page 123)*.
2. Migrate Data to AWS Using S3 EFS *(on page 128)*.

Migrate Data Using an AWS Transfer Family

1. Deploy Proficy Historian for AWS *(on page 43)*.
2. Install collectors *(on page)*.

This topic describes how to migrate data from an on-premise Historian server to Proficy Historian for AWS using a Transfer Family.

1. Create an SFTP-enabled server.
2. After the server is running (online), you need to add a user. To create a user, generate an ssh key-pair (using powershell or ubuntu):

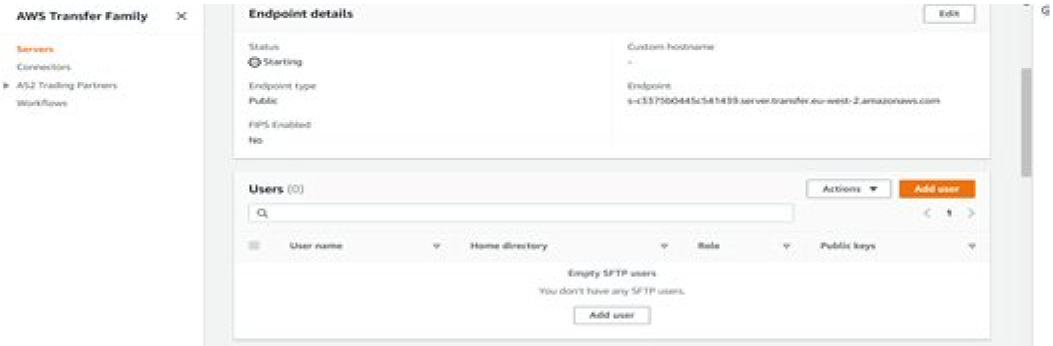
For both the private and public key:

```
ssh-keygen -t rsa
```

To copy the id_rsa.pub (using the public key):

```
cat ~/.ssh/id_rsa.pub
```

3. Add a user.



Status: Starting

Endpoint type: Public

FIPS Enabled: No

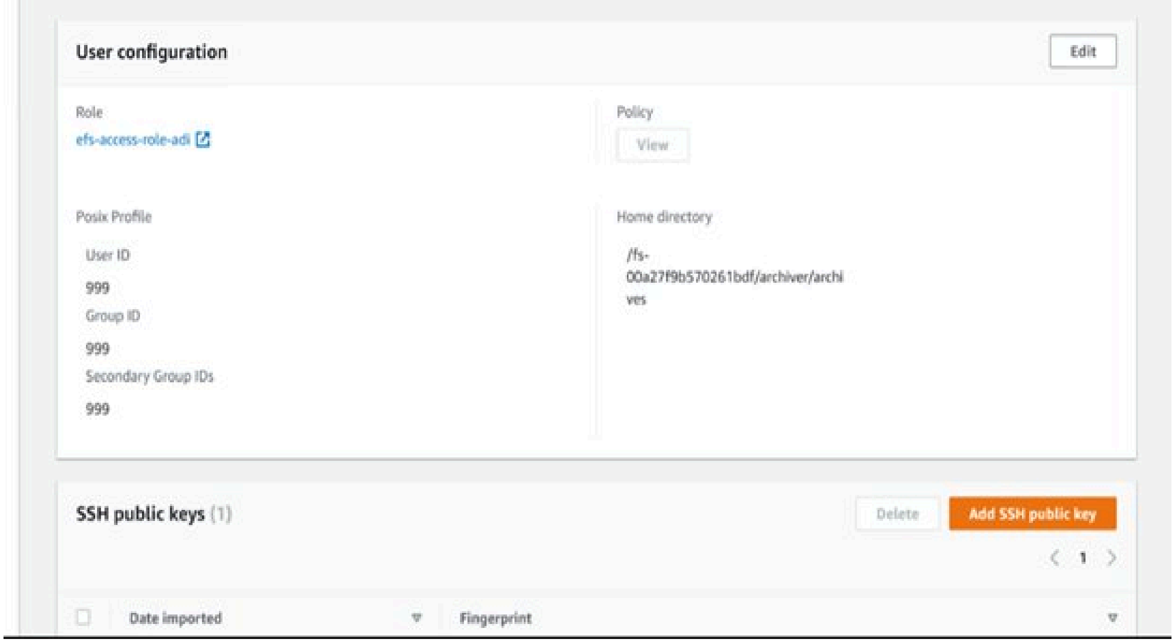
Custom hostname: -

Endpoint: s-c3375b0445c541459.server.transfer.eu-west-2.amazonaws.com

Empty SFTP users

You don't have any SFTP users.

4. While adding user we need to provide User ID, Group ID, Secondary Group IDs as **999**.
5. Specify the Home directory path as: <File system ID(EFS)>/archiver/archives:



Role: efs-access-role-adi

Policy

Posix Profile

User ID: 999

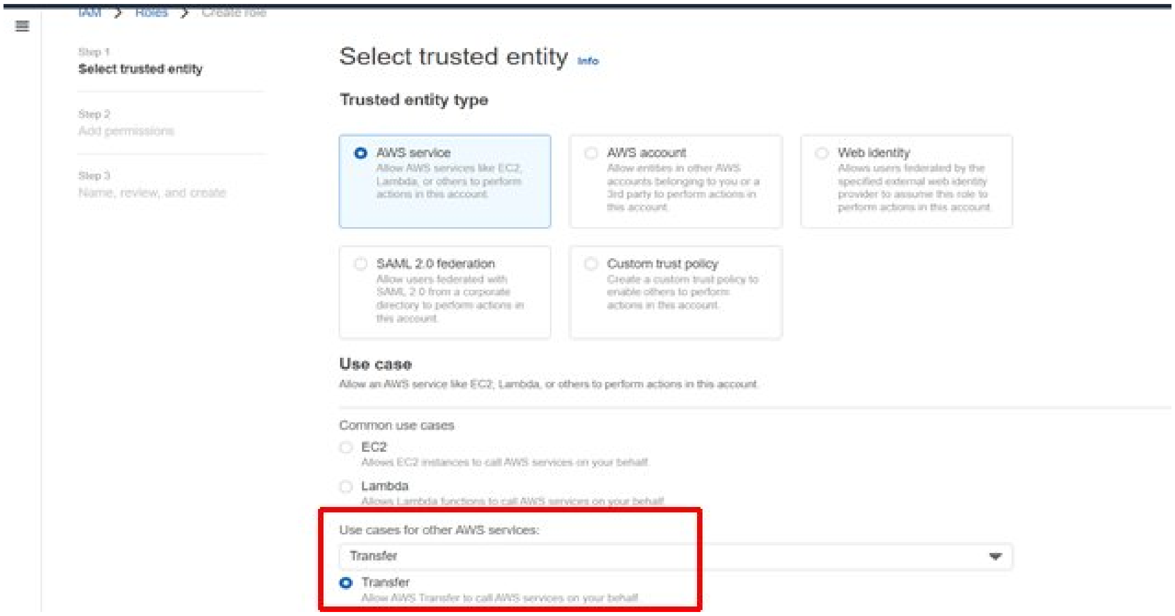
Group ID: 999

Secondary Group IDs: 999

Home directory: /fs-00a27f9b570261bdf/archiver/archives

When creating a user, it is required that IAM role with specific permissions be applied.

6. To create an IAM role, from the **IAM** screen, click **Roles** and then **Create Role.** The following screen appears.



IAM > Roles > Create role

Step 1
Select trusted entity

Step 2
Add permissions

Step 3
Name, review, and create

Select trusted entity Info

Trusted entity type

● AWS service
Allow AWS services like EC2, Lambda, or others to perform actions in this account.

○ AWS account
Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.

○ Web identity
Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.

○ SAML 2.0 federation
Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.

○ Custom trust policy
Create a custom trust policy to enable others to perform actions in this account.

Use case
Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Common use cases

○ EC2
Allows EC2 instances to call AWS services on your behalf.

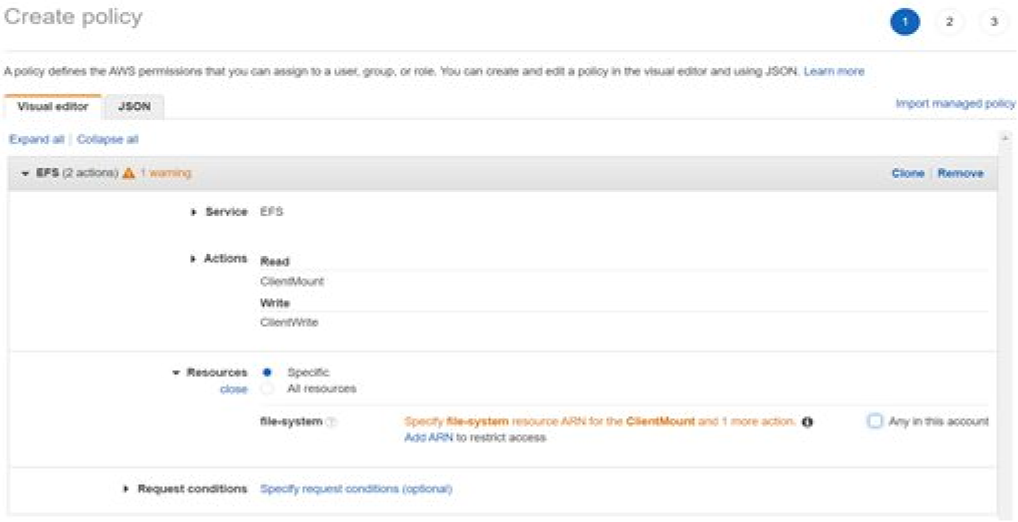
○ Lambda
Allows Lambda functions to call AWS services on your behalf.

Use cases for other AWS services:

Transfer

● Transfer
Allow AWS Transfer to call AWS services on your behalf.

7. In the Use Cases for Other AWS Services, select **Transfer** and then click **Next**.
8. To create the IMA policy, select **Service** as EFS, **Actions** as ClientMount and ClientWrite, **Resources** as All Resources, and leave the rest of the settings with the defaults.



Create policy

1 2 3

A policy defines the AWS permissions that you can assign to a user, group, or role. You can create and edit a policy in the visual editor and using JSON. Learn more

Visual editor | JSON | Import managed policy

Expand all | Collapse all

EFS (2 actions) 1 warning | Clone | Remove

Service: EFS

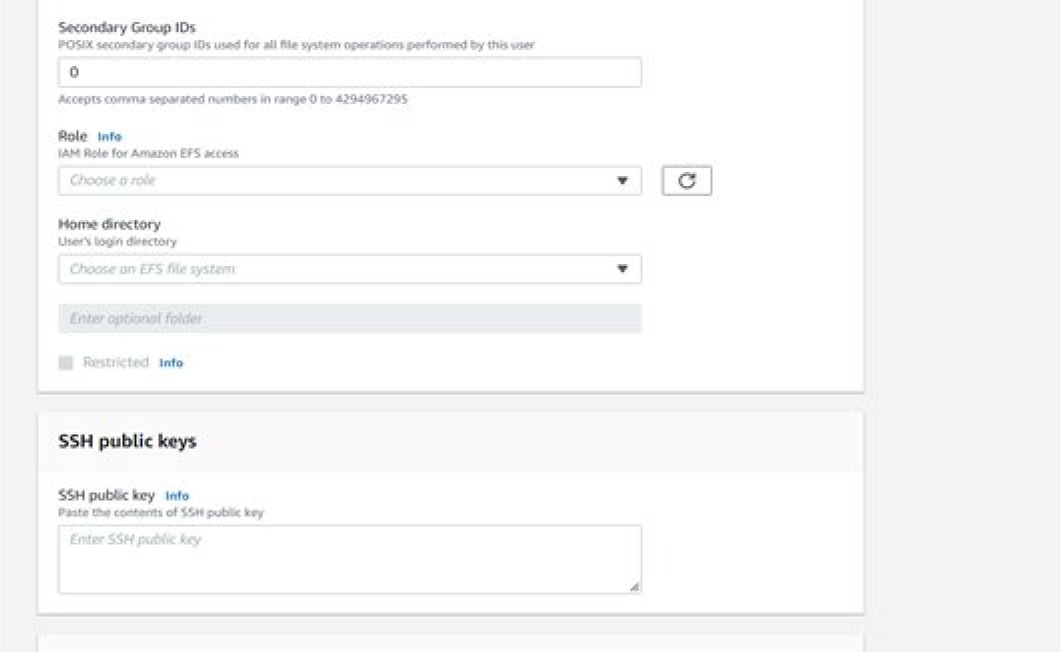
Actions:
Read
ClientMount
Write
ClientWrite

Resources (close):
● Specific
○ All resources

file-system: Specify file-system resource ARN for the ClientMount and 1 more action. Add ARN to restrict access. □ Any in this account

Request conditions: Specify request conditions (optional)

The IMA role and policy is created:



Secondary Group IDs

POSIX secondary group IDs used for all file system operations performed by this user

0

Accepts comma separated numbers in range 0 to 4294967295

Role Info

IAM Role for Amazon EFS access

Choose a role

Home directory

User's login directory

Choose an EFS file system

Enter optional folder

□ Restricted Info

SSH public keys

SSH public key Info

Paste the contents of SSH public key

Enter SSH public key

9. Provide the home directory and the VPC and public key using an application.
10. To transfer data we can use an application like sftp or WinSCP.

Using sftp:

a. Login to the sftp to the server.

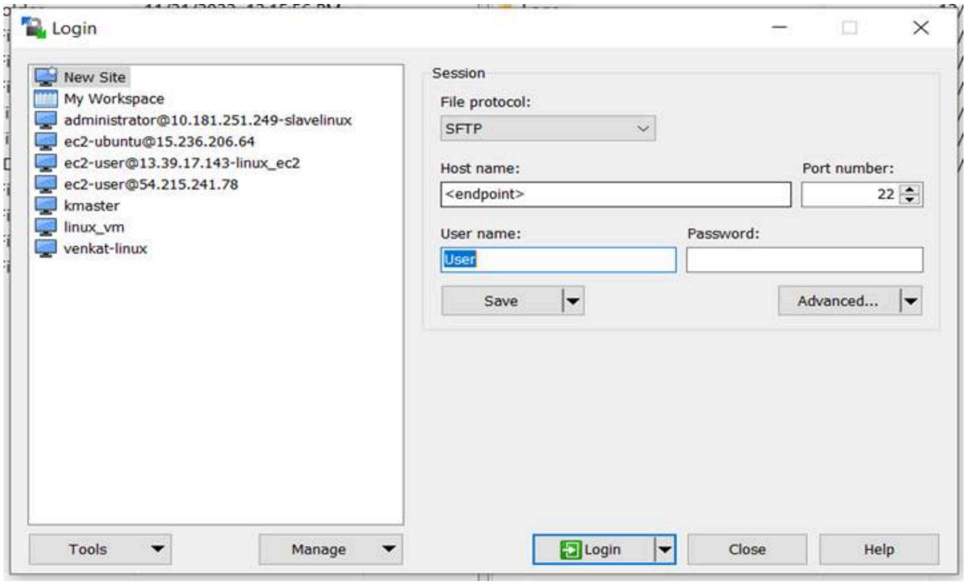
```
sftp -i <pravate_key_along with path> <name of the user created>@<endpoint generated in transfer
 family>
```

b. After you successfully login, transfer the file using the put command:

```
put <source file> <destination file>
```

Using WinSCP:

a. Open WinSCP.



File protocol: SFTP

Host name: <endpoint>

Port number: 22

User name: User

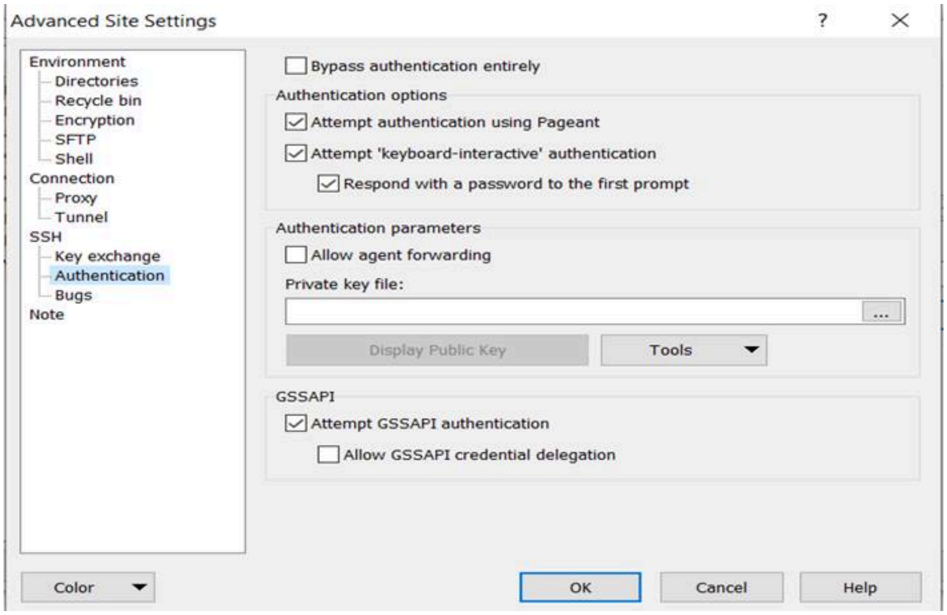
Password:

b. In the **Host Name** field, enter the hostname as Endpoint.

c. In the **User Name** field, enter the User from the AWS transfer family server.

d. Click **Advance** to add the private key.

e. Under **Authentication**, add the private key that you generated using ssh key-pair and click OK.



☐ Bypass authentication entirely

Authentication options

☑ Attempt authentication using Pageant

☑ Attempt 'keyboard-interactive' authentication

☑ Respond with a password to the first prompt

Authentication parameters

☐ Allow agent forwarding

Private key file:

GSSAPI

☑ Attempt GSSAPI authentication

☐ Allow GSSAPI credential delegation

Now WinSCP will be connected with the server and you can transfer files to EFS.

11. While connected to EFS, go to the historian/archiver/archives folder and drag-and-drop the .iha and .ihc files.

> **Note:**
>
> Rename the local .ihc file as **historian-archiver-sts-0_Config.ihc**.

12. Restart the Data Archiver.

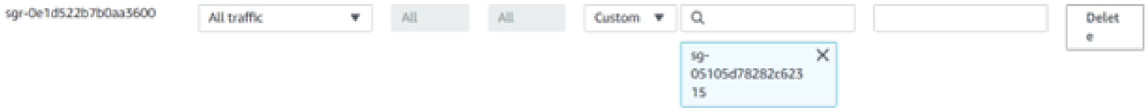
Migrate Data to AWS Using S3 EFS

1. Deploy Proficy Historian for AWS *(on page 43)*.
2. Install collectors *(on page)*.

This topic describes how to migrate data from an on-premise Historian server to Proficy Historian for AWS using S3 EFS.

1. Rename the local .ihc file as **historian-archiver-sts-0_Config.ihc**.
2. Migrate data from an on-premise Historian server to an S3 bucket. For the S3 bucket, we recommend that you use the same location in which you have deployed Proficy Historian for AWS. This step is required only if you are using any of the snow family options.
3. Migrate data from the S3 bucket to EFS. Provide the same subnet and security group details that you have used while deploying Proficy Historian for AWS. Provide the source as **s3** and destination as **efs**. The path should be `/archiver/archives`.

   An execution task is created in AWS DataSync.
4. In the security group, add all traffic port (use the same security group which is used while creating the DataSync task). If the port already exists, try to delete and recreate the port again.

   sgr-0e1d522b7b0aa3600 | All traffic ▼ | All | All | Custom ▼ | sg-05105d78282c62315 ✕ | Delete

5. Start the DataSync task.

   The task is executed, and the data is migrated to EFS.
6. Restart the Data Archiver.

Chapter 6. Sending Data to Amazon S3

About Sending Data to Amazon S3

You can send tag data stored in Data Archiver to Amazon S3 at a scheduled interval. These files are stored as Parquet and/or CSV files in Amazon S3. You can specify the following parameters while exporting tag data to Amazon S3:

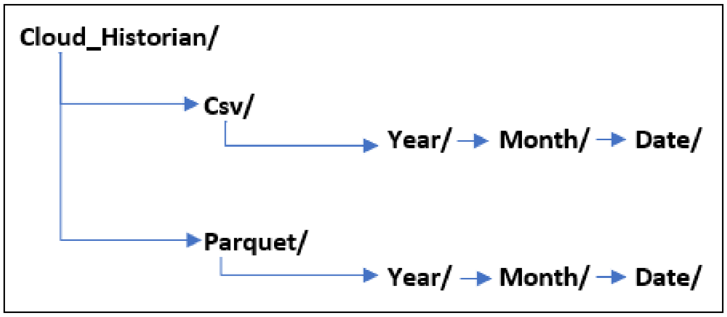
- The list of tags whose data you want to export
- The export schedule, which you can configure at a granular level
- The format in which you want to store the tag data (CSV and/or Parquet)

Advantages of storing data in the Parquet file format:

- Data is stored in Parquet files is efficient and optimizes space.
- Since Parquet files are compatible with most applications, you can analyze this data in analysis tools such as Amazon Athena, Amazon Redshift, Amazon Kinesis, Amazon QuickSight, Microsoft Power BI, and Tableau.

How it works:

1. You will create an S3 bucket in which you want to store tag data.
2. You will deploy the scheduled export. This will be automatically deployed on the same VPC on which you have deployed Proficy Historian for Cloud.
3. You will configure the settings for the scheduled export. To do so, you will edit and upload the `ScheduledExportConfiguration.xml` file to the S3 bucket that you have created. This file contains settings related to tags, export schedule, and the file format.
4. Service Manager tasks are triggered based on the schedule you have configured.
5. The Scheduled Export service reads tag data from Data Archiver based on the settings you have configured.
6. CSV and/or Parquet files are exported into the S3 bucket that you have created. Depending on the file format and the date of export, the following folder structure is created, and files are stored in the respective folders.



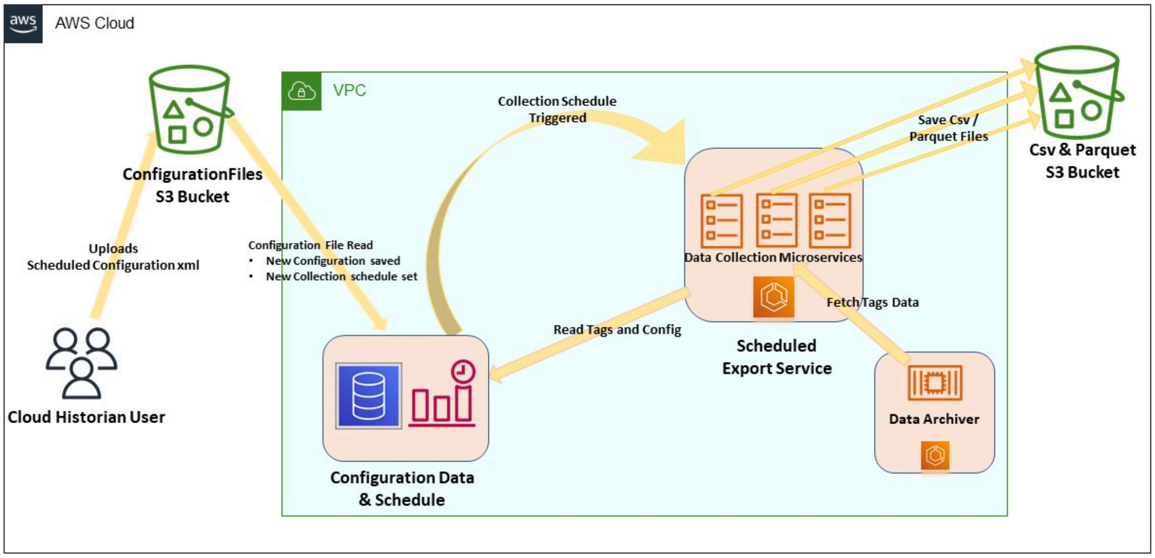
7. You can use the tag data stored in the CSV and/or Parquet files in analysis tools such as AWS Athena.

Workflow:

Step Number	Description	Notes
1	Deploy Proficy Historian for AWS *(on page 43)*.	This step is required to install the Historian server in an EC2 instance in a VPC.
2	Install collectors *(on page)* and create a collector instance.	This step is required if you want to collect data using collectors.
3	Deploy Scheduled Export to Parquet/CSV Service *(on page 131)*.	This step is required to install the components that will export tag data from Data Archiver to Amazon S3 in the Parquet and/or CSV file format.
4	Configure the export settings *(on page 134)*.	This step is required. It involves providing an XML file with the list of tags, export schedule, and the file format for the export.

After you perform these steps, tag data will be exported into S3 based on the specified schedule. This data will be available in Parquet and/or CSV files. You can then analyze this data using analysis tools.

Scheduled Export User Flow

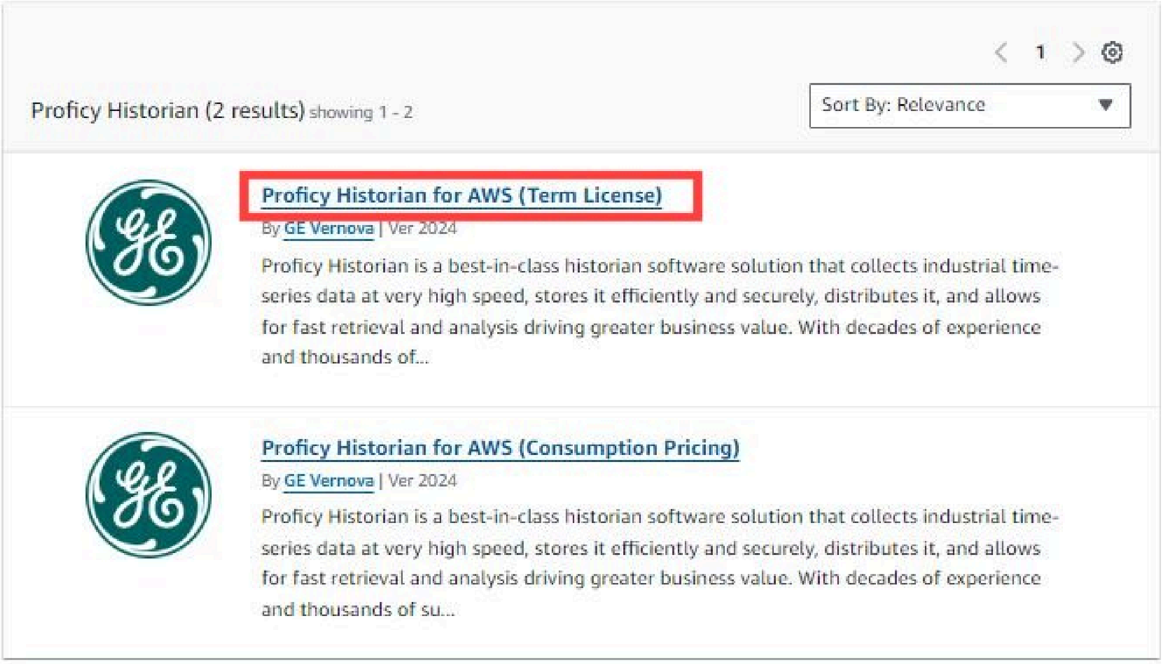

AWS Cloud
VPC
ConfigurationFiles S3 Bucket
Uploads Scheduled Configuration xml
Cloud Historian User
Configuration File Read
• New Configuration saved
• New Collection schedule set
Collection Schedule Triggered
Configuration Data & Schedule
Read Tags and Config
Data Collection Microservices
Scheduled Export Service
Fetch Tags Data
Data Archiver
Save Csv / Parquet Files
Csv & Parquet S3 Bucket

Deploy Scheduled Export

1. Deploy Proficy Historian for Cloud *(on page 43)*.
2. Ensure that you have an S3 bucket to store the exported files and the configuration files, and you must have the permission to store files in the S3 bucket. In addition, the S3 bucket must be in the same region as Proficy Historian for Cloud.

> **Note:**
> You can change the S3 bucket to a different one *(on page 136)* even after deploying scheduled export.

1. Log in to the AWS marketplace.
2. Search for Proficy Historian.
3. In the list of products that appear, if you have the Historian for Linux license, select **Proficy Historian for AWS**.



Proficy Historian (2 results) showing 1 - 2

Sort By: Relevance

Proficy Historian for AWS (Term License)

By GE Vernova | Ver 2024

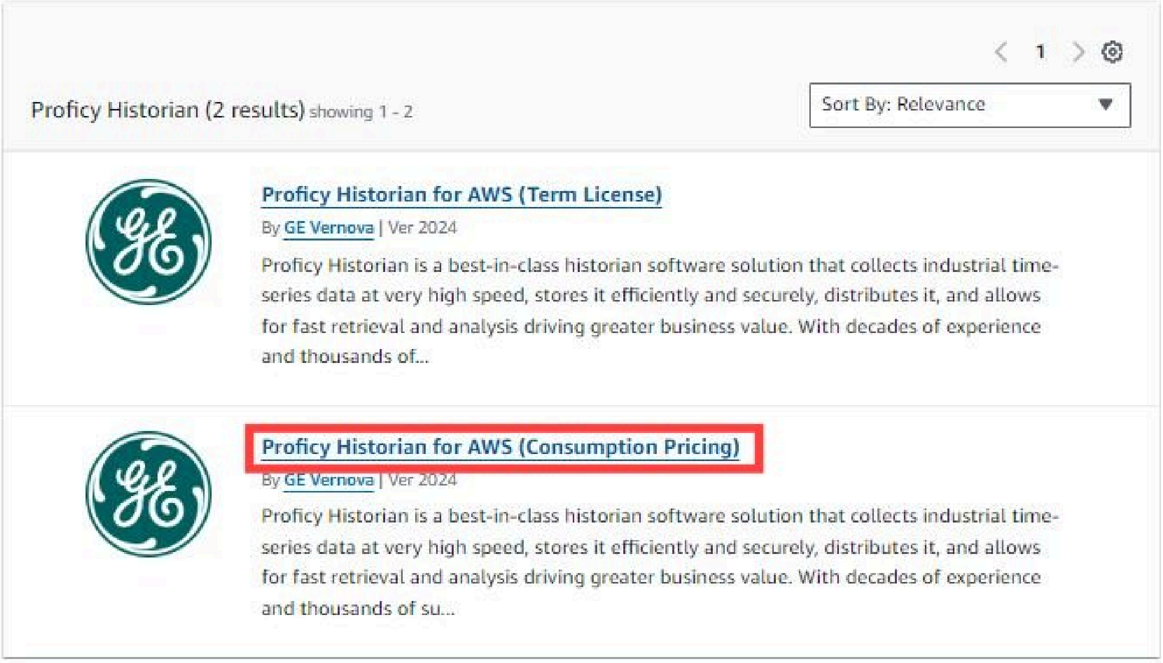
Proficy Historian is a best-in-class historian software solution that collects industrial time-series data at very high speed, stores it efficiently and securely, distributes it, and allows for fast retrieval and analysis driving greater business value. With decades of experience and thousands of...

Proficy Historian for AWS (Consumption Pricing)

By GE Vernova | Ver 2024

Proficy Historian is a best-in-class historian software solution that collects industrial time-series data at very high speed, stores it efficiently and securely, distributes it, and allows for fast retrieval and analysis driving greater business value. With decades of experience and thousands of su...

Or, if you want to use the consumption model, select **Proficy Historian for AWS (Consumption Pricing)**.



Proficy Historian (2 results) showing 1 - 2

Sort By: Relevance

Proficy Historian for AWS (Term License)

By GE Vernova | Ver 2024

Proficy Historian is a best-in-class historian software solution that collects industrial time-series data at very high speed, stores it efficiently and securely, distributes it, and allows for fast retrieval and analysis driving greater business value. With decades of experience and thousands of...

Proficy Historian for AWS (Consumption Pricing)

By GE Vernova | Ver 2024

Proficy Historian is a best-in-class historian software solution that collects industrial time-series data at very high speed, stores it efficiently and securely, distributes it, and allows for fast retrieval and analysis driving greater business value. With decades of experience and thousands of su...

4. Select **Continue to Subscribe**.

   The terms and conditions appear.
5. Select **Continue to Configuration**.

   The **Configure this software** page appears.

6. Enter values as described in the following table.

Field	Description
Fulfillment option	Select **CloudFormation Template**.
Software version	Select **2024**.

7. Select **Continue to Launch**.

   The **Launch this software** page appears.

8. Under **Deployment template**, select **Deploy Scheduled Export to Parquet/CSV Service**.

   The **Quick create stack** page appears.

9. Enter values as described in the following table.

Field	Description
Stack name	Enter a name for the stack. A value is required and must be unique. The value can include all alphanumeric characters and dashes. It must begin with an alphabetic character and cannot exceed 128 characters.
S3 Bucket Name	Enter the name of the S3 bucket in which you want in which you want to store the exported files and the configuration files. A value is required. The S3 bucket must be in the same region as Proficy Historian for Cloud. **Note:** You can change the S3 bucket to a different one *(on page 136)* even after deploying scheduled export.
EKS ClusterName	Enter the name of the EKS cluster in which you have deployed Proficy Historian for Cloud.
Username	Enter the username of a user who can read data from Data Archiver.
Password	Enter the password of the user.

   The **Configure stack options** page appears.

10. As needed, modify values in the available fields, and then select **Next**.

    The **Review <stack name>** page appears.

11. Select the check box to acknowledge the capabilities required by CloudFormation, and then select **Submit**.
    The stack is created. Scheduled export is now deployed on the same VPC as the one on which you have deployed Proficy Historian for Cloud. A folder named `ConfigurationFiles` is created in the S3 bucket.

Configure the export settings *(on page 134)*.

Configure Export Settings

Deploy the scheduled export *(on page 131)*.

This topic describes how to specify the following parameters while exporting tag data to Amazon S3:

- The list of tags whose data you want to export
- The export schedule, which you can configure at a granular level
- The format in which you want to store the tag data (CSV and/or Parquet)

1. Access the `ScheduledExportConfiguration.xml` file located at the following path: https://proficy-historian.s3.us-east-2.amazonaws.com/Scheduled+Export/ScheduledExportConfiguration.xml.
2. Under Taglist, provide a list of tags whose data you want to collect. A value is required.
   For example, to collect data for tags Tag1 and Tag2, enter:

   ```
   <Taglist>
       <Tag Name="Tag1"></Tag>
       <Tag Name="Tag2"></Tag>
   </Taglist>
   ```

   You can use the * and ? wildcard characters. For example, to collect data for all the tags that begin with Tank or Column, enter:

   ```
   <Taglist>
       <Tag Name="Tank*"></Tag>
       <Tag Name="Column*"></Tag>
   </Taglist>
   ```

3. To schedule data collection at a specific interval, enter the interval in minutes under Interval.

Note:
The Minutes should be a minimum of 30 minutes, as supported in the ScheduledExportConfiguration.xml.

For example, if you want to schedule data collection for every 6 hours, enter:

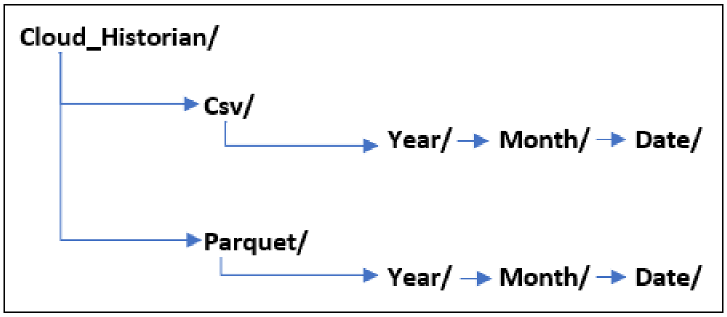
```
<Interval>

    <Minutes>360</Minutes>

</Interval>
```

Tip:
For more information on using cron expressions, refer to https://docs.aws.amazon.com/AmazonCloudWatch/latest/events/ScheduledEvents.html

4. To schedule data collection at specific timings, days of a week, months, or years, enter the values under Cron. A value is required either in the Interval or Cron section.
   For example, if you want to schedule data collection at 0:10:00, 6:10:00, 10:10:00, and 18:10:00 from Monday to Friday for November and December 2022, enter:<Cron> <Minutes>10</Minutes> <Hours>0,6,10,18</Hours> <DayOfMonth>?</DayOfMonth> <Month>11,12</Month> <DayOfWeek>mon-fri</DatOfWeek> <Year>2022</Year> </Cron>
5. Under S3, specify whether you want to store tag data in Parquet and/or CSV format. You must enter `Yes` at least for one of the formats.

```
<S3>

    <SaveCsv>Yes</SaveCsv>

    <SaveParquet>Yes</SaveParquet>

</S3>
```

6. Save the file, and upload it to the `ConfigurationFiles` folder in the S3 bucket that you have specified while deploying the scheduled export. The `ConfigurationFiles` folder is created automatically after you deploy the scheduled export.
   Tag data is now exported for the list of tags and the schedule that you have specified. Depending on the file format you have specified, these files are stored in the S3 bucket in the following folder structure:



The date is stored in the UTC time format.

Change the S3 Bucket Used for Scheduled Export

1. Access AWS CloudFormation, and then select **Stacks**.
2. Select the stack in which you have deployed the scheduled export, and then select **Update**.
3. Under **Prepare template**, select **Use current template**, and then select **Next**.
4. In the **S3 Bucket Name** field, enter the name of the new S3 bucket that you want to use, and then select **Next**.

   The **Configure stack options** page appears.
5. As needed, modify values in the available fields, and then select **Next**.

   The **Review <stack name>** page appears.
6. Select the check box to acknowledge the capabilities required by CloudFormation, and then select **Submit**.

   The stack is updated. A folder named `ConfigurationFiles` is created in the new S3 bucket.

   > **Note:**
   > Files in the old S3 bucket will not be available in the new one.

Disable Scheduled Export

1. Access CloudFormation, and then select the stack in which you have deployed scheduled export.
2. Select **Delete**.

   A message appears, asking you to confirm that you want to delete the stack.
3. Select **Delete stack**.

   The stack is deleted, and the scheduled export is disabled.

Chapter 7. Proficy Authentication

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)*.

Workflow

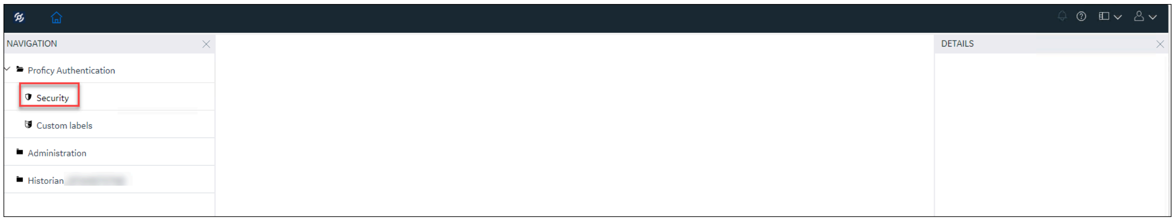
1. Provide the details of the remote Proficy Authentication service while installing Web-based Clients *(on page)*.
2. Connect to the remote Proficy Authentication service *(on page)*.
3. Map the Proficy Authentication groups with that of the Historian Proficy Authentication instance. You can map the 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
Proficy Authentication	**REG_SZ**	**https://hostname.domain:433/uaa/oauth/token** For example, https://testmachine.ge.com:443/uaa/oauth/token.
UseProficy Authentication-Authentication	**REG_DWORD**	**0x00000001**.

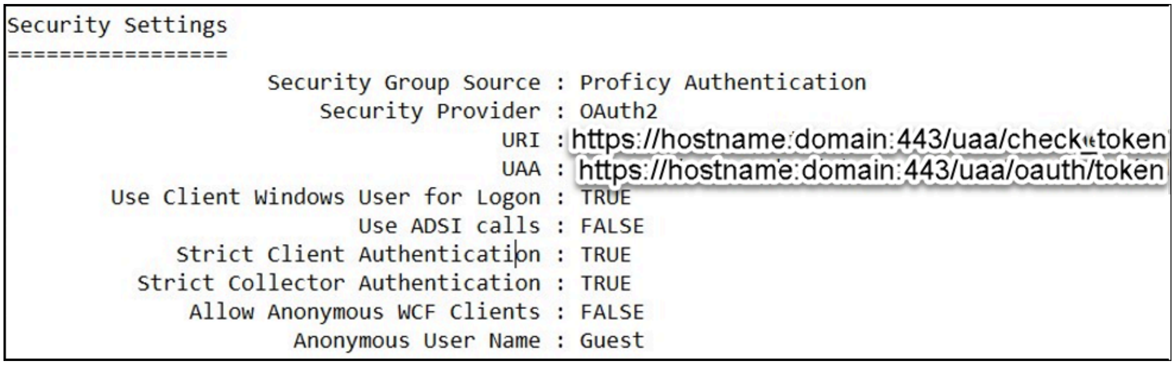
   b. Save the registries.
6. Now, you can Access Configuration Hub *(on page 97)* using the predefined username, that is, <hostname>.admin and password.
7. In the **NAVIGATION** pane, in the Configuration Hub plugin for Historian, access **Systems**, and then select the system as needed.

   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\Proficy Authentication-tomcat\webapps\Proficy Authentication\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\Proficy Authentication-tomcat\bin` folder, rename the `tomcat8w.exe` file `Proficy AuthenticationTomcat.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 \Proficy Authentication-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.

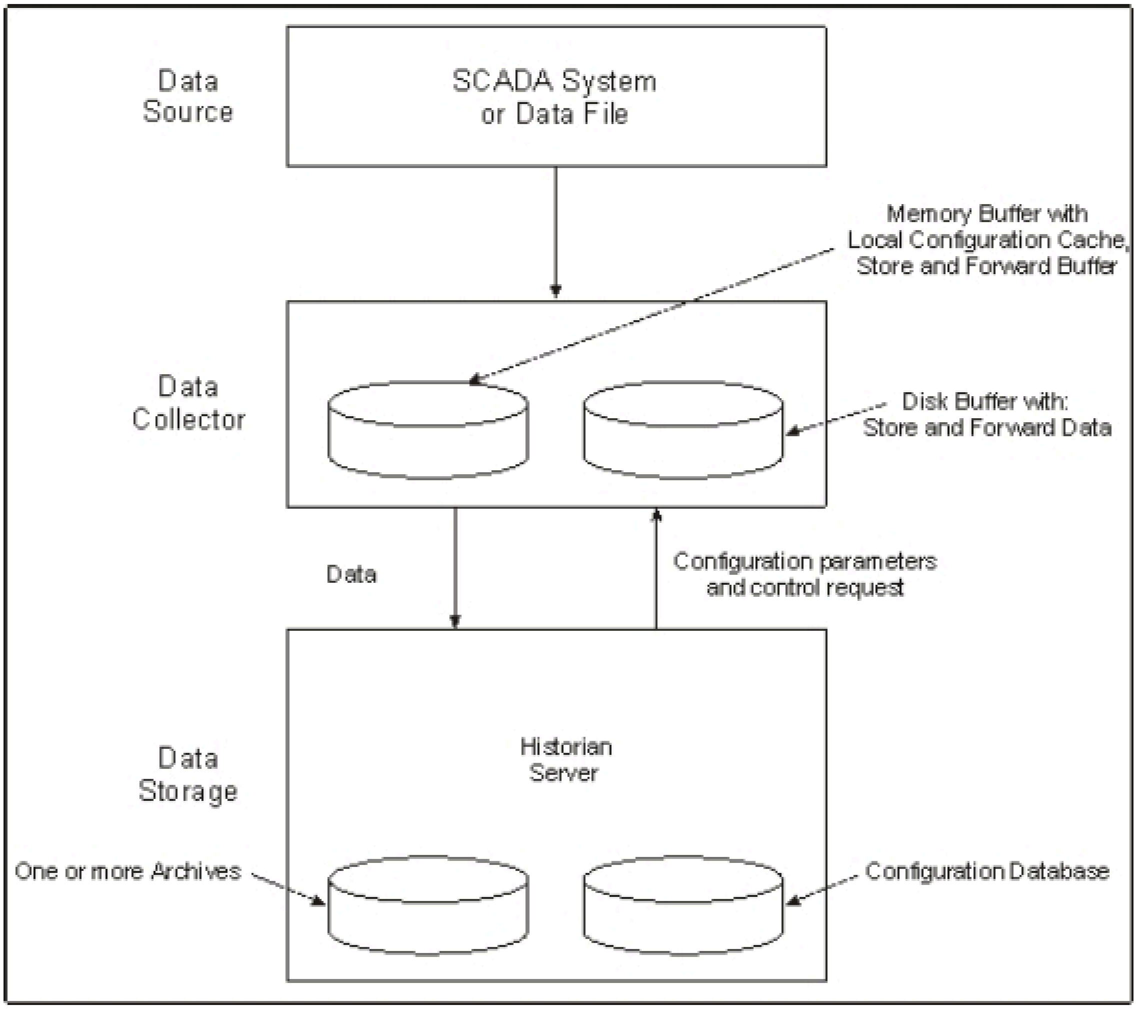
Chapter 8. Data Collectors

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.


Data Source
SCADA System or Data File
Memory Buffer with Local Configuration Cache, Store and Forward Buffer
Data Collector
Disk Buffer with: Store and Forward Data
Data
Configuration parameters and control request
Data Storage
Historian Server
One or more Archives
Configuration Database

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 231)*	Performs data calculations on values stored in the archiver.	No	Yes
The iFIX Alarms and Events collector collector *(on page 299)*	Collects alarms and events data from iFIX.	No	No
The iFIX collector *(on page 299)*	Collects tag data from iFIX.	No	No
The MQTT collector *(on page 346)*	Collects data published to a topic using an MQTT broker.	Yes	Yes
The ODBC collector *(on page 359)*	Collects data from an application based on an ODBC driver.	Yes	Yes
The OPC Classic DA collector *(on page 374)*	Collects data from an OPC Classic Data Access (DA) server (such as CIMPLICITY).		
The OPC Classic HDA collector *(on page 391)*	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 405)*	Collects data from an OPC UA DA server (such as CIMPLICITY).	Yes	Yes
The OSI PI collector *(on page 421)*	Collects data from an OSI PI server.	No	No
The Python collector *(on page 440)*	Run Python scripts on tag values and stores them in Historian	No	No

Collector Name	Description	Is Toolkit-Based?	Consumes a CAL?
The Server-to-Server collector *(on page 475)*	Collects data from a Historian server and sends it to another Historian server.	No	Yes
The Server-to-Server distributor *(on page 530)*	Collects data from a smaller Historian server and sends it to a larger, centralized Historian server or a cloud destination.	No	Yes
The Simulation collector *(on page 534)*	Generates random numbers and string patterns for testing/demonstration purposes.	No	No
The Wonderware collector *(on page 537)*	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 149)*. 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-premise 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-premise 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.

<table>
<tr><th>Functionality</th><th>Destination - Historian</th><th>Destination - Cloud</th></tr>
<tr><td>HISTORIANNODENAME registry key</td><td>Contains the destination Historian Server’s name/ IP Address.</td><td>Contains the cloud destination settings as well as proxy historian server name or IP if applicable (configServer).
Cloud destination format:
<code>CloudDestinationAddress|configServer|IdentityIssu</code>
<code>er|ClientID|</code>
<code>ClientSecret|ZoneID|Proxy|proxyUser|proxyPassword</code></td></tr>
<tr><td>Mapping Source Tags
with Destination Tags
(Add Tags)</td><td>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.</td><td>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.
If the user selects Historian, then tags will be created in the Cloud which in turn may have</td></tr>
</table>

Functionality	Destination - Historian	Destination - Cloud
		been mapped through one of the Admin tools (VB Admin/Web Admin). If Offline Configuration File is selected, the user must provide an XML configuration file containing tag configurations that need to be created 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

Collector Name	32-bit	64-bit
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)*.

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

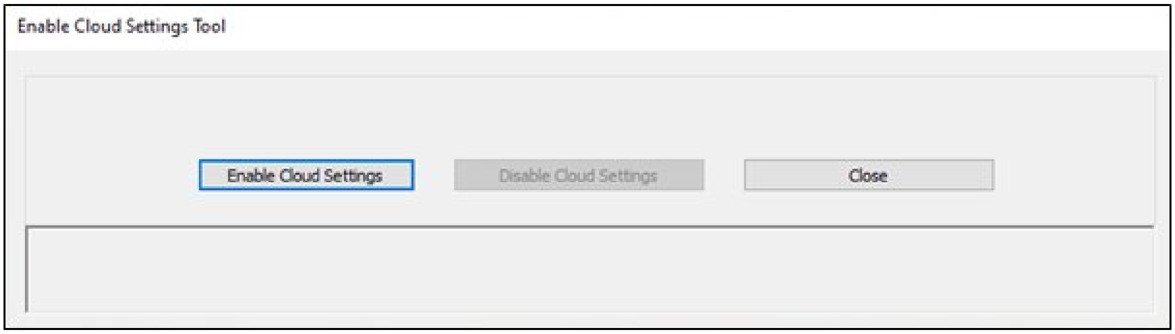
Here are some best practices that enable the collectors collect and store the most accurate data:

- Synchronize the 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.

Enable TLS encryption for Collectors Connecting to Cloud Historian

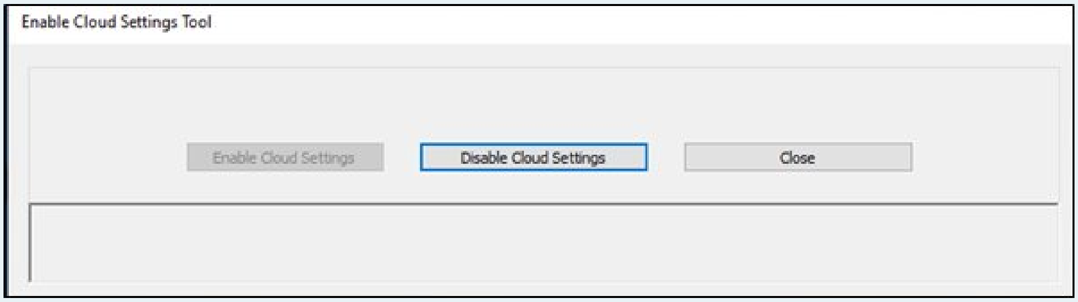
Be aware that we are using TLS version 1.2. The encryption method being used is dynamic between the client and server. During the handshake, these applications exchange encryption methods and agree on one of them to communicate.

1. Launch the InstallCloudDependencies_x64.exe file from CloudDependenciesTool folder available in Historian install media (.ISO). The following screen appears.



2. Select the **Enable Cloud Settings** button.

Note:
To disable TLS encryption for Collectors Connecting to Cloud Historian, from this same screen, select the **Disable Cloud Settings** button, as shown in the following figure.



Upgrade Collectors

- Direct upgrades from a previous version of Cloud Collectors to Historian 2024 Collectors is not supported. It is recommended that you uninstall the older version of collectors and re-install the Historian 2024 Collectors from the Historian 2024 install media (.ISO).
- 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 them up, uninstall the collectors, and then install the latest version.

Install the collectors *(on page 90)*.
The collectors are upgraded to the latest version.

Sending Data to Cloud

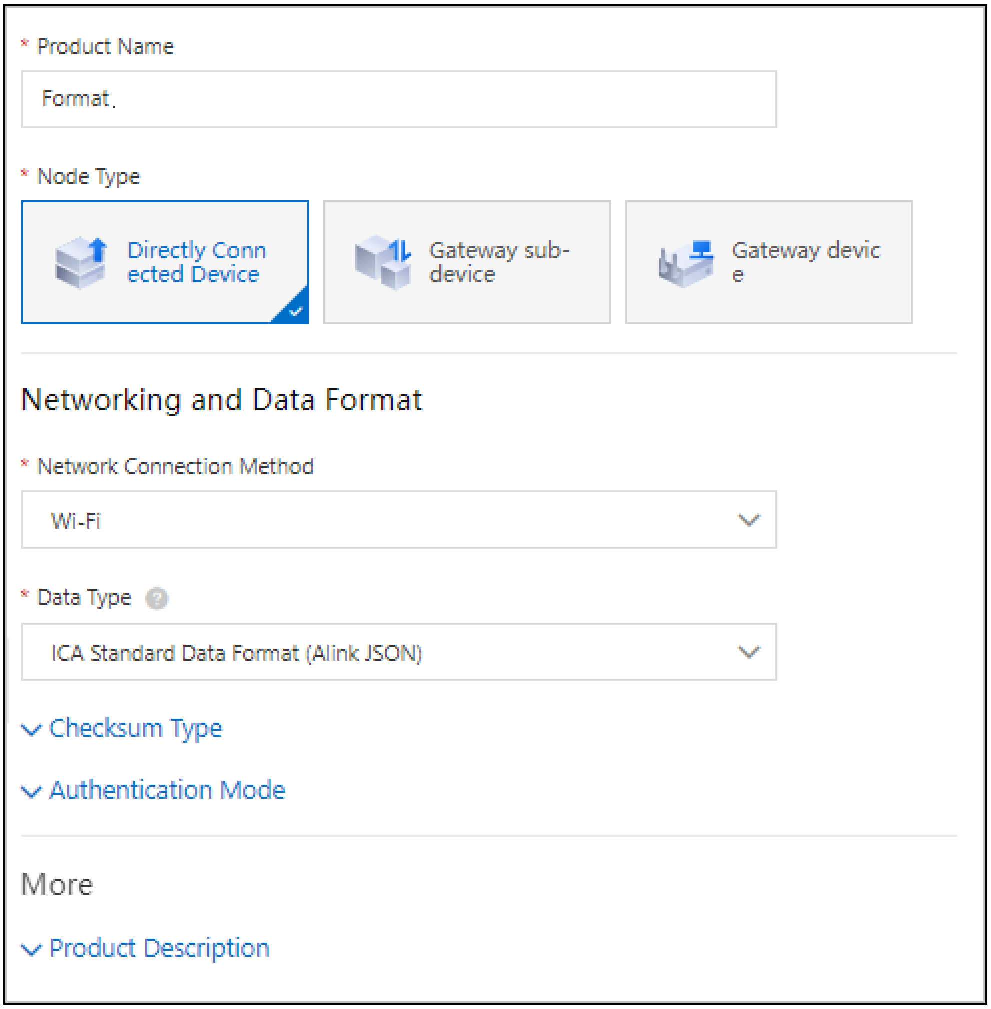
Send Data to Alibaba Cloud

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**.



* Product Name

Format.

* Node Type

Networking and Data Format

* Network Connection Method

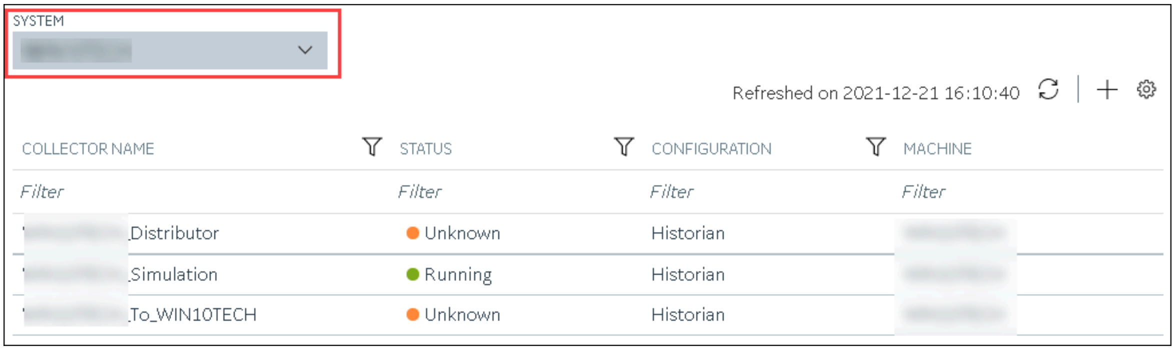
Wi-Fi

* Data Type

ICA Standard Data Format (Alink JSON)

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 97)*.
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.

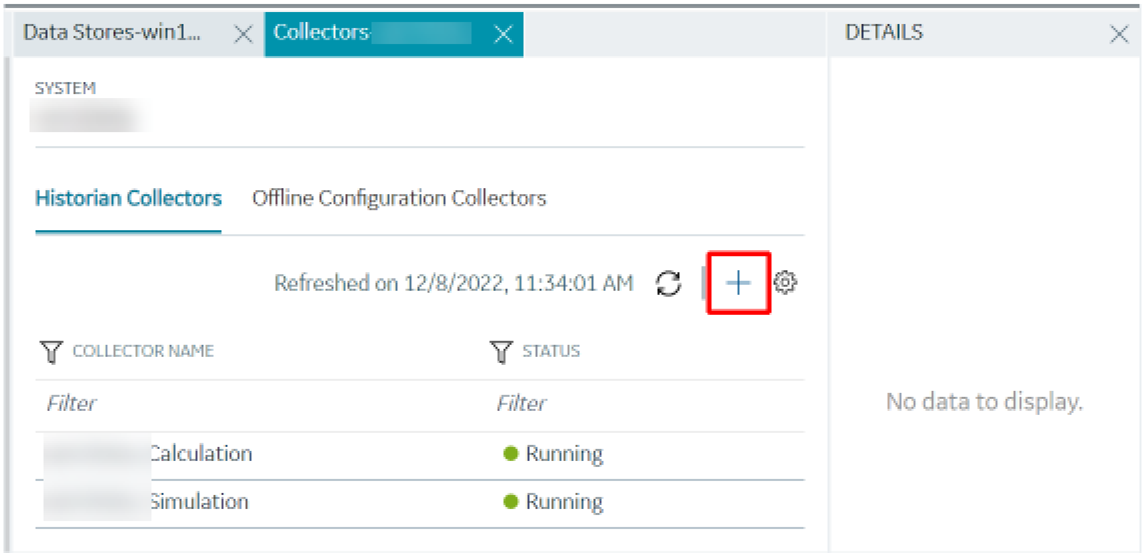


SYSTEM [redacted]

Refreshed on 2021-12-21 16:10:40

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE
Filter	*Filter*	*Filter*	*Filter*
[redacted]Distributor	● Unknown	Historian	[redacted]
[redacted]Simulation	● Running	Historian	[redacted]
[redacted]To_WIN10TECH	● Unknown	Historian	[redacted]

9. In the upper-right corner of the main section, select +.



SYSTEM [redacted]

Refreshed on 12/8/2022, 11:34:01 AM

COLLECTOR NAME	STATUS
Filter	*Filter*
[redacted]Calculation	● Running
[redacted]Simulation	● Running

No data to display.

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**.

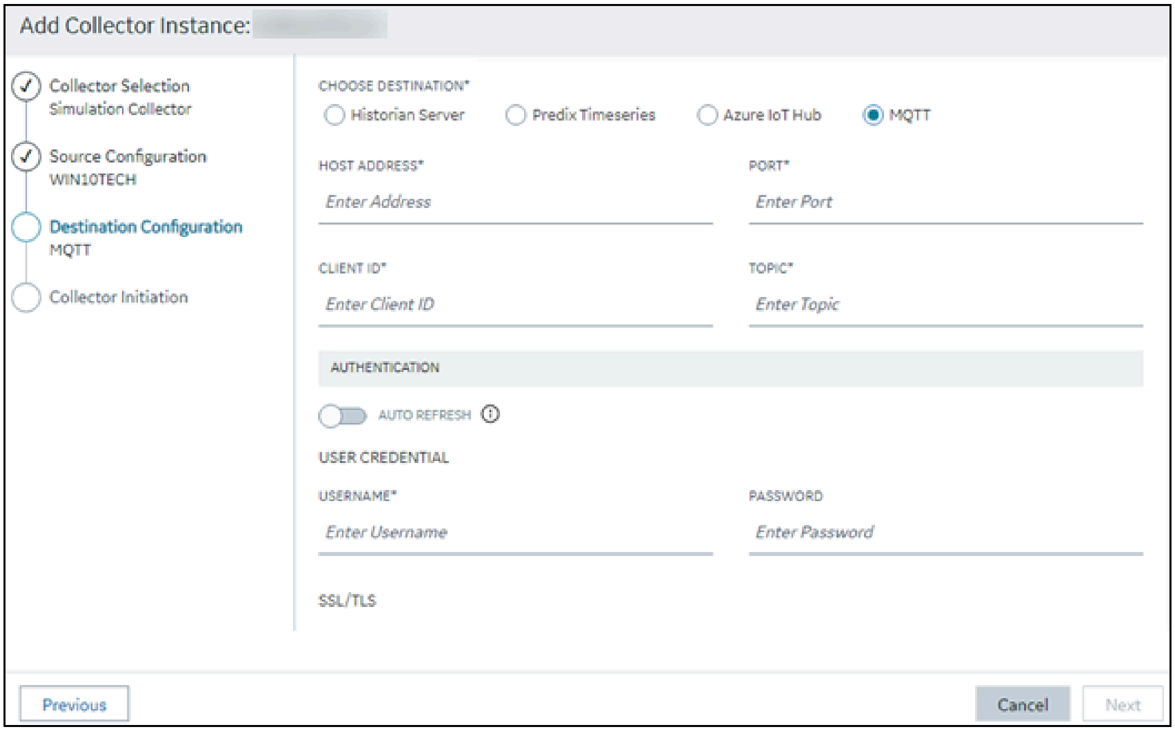
    The **INSTALLATION DRIVE** and **DATA DIRECTORY** fields are disabled and populated.

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.



Add Collector Instance: [redacted]

- ✓ Collector Selection
  Simulation Collector
- ✓ Source Configuration
  WIN10TECH
- ○ Destination Configuration
  MQTT
- ○ Collector Initiation

CHOOSE DESTINATION*
○ Historian Server ○ Predix Timeseries ○ Azure IoT Hub ● MQTT

HOST ADDRESS*
Enter Address

PORT*
Enter Port

CLIENT ID*
Enter Client ID

TOPIC*
Enter Topic

AUTHENTICATION

AUTO REFRESH

USER CREDENTIAL

USERNAME*
Enter Username

PASSWORD
Enter Password

SSL/TLS

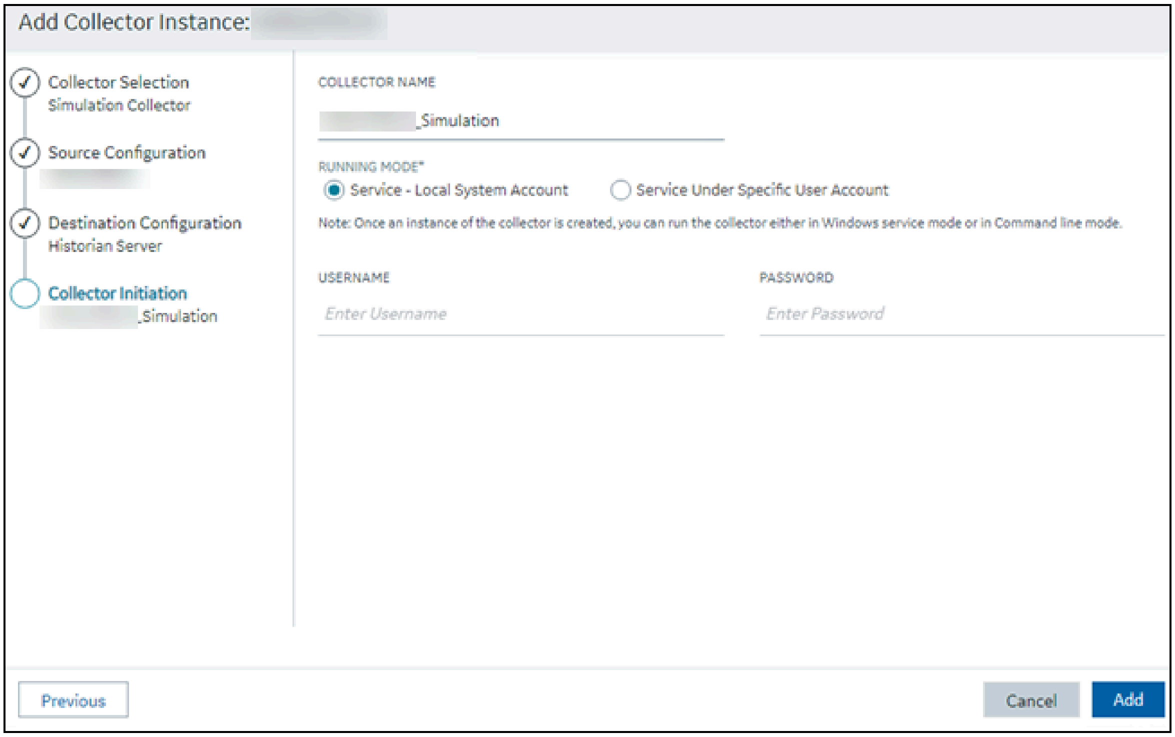
13. In the **CHOOSE DESTINATION** field, select **MQTT**, and then provide values as described in the following table.

Field	Description
HOST ADDRESS	Enter a value in the following format: `<product name>.iot-as-mqtt.<region ID>.aliyuncs.com`. A value is required. 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. ◦ For securemode, enter `2` for direct TLS connection, or enter `3` for direct TCP connection. ◦ 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. 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. 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. 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: ◦ **Historian Configuration**: Select this option if you want to add the tags manually using Historian Administrator *(on page 741)*. If you select this option, the **CONFIGURATION HISTORIAN SERVER** field appears. ◦ **Offline Configuration**: Select this option if you want to provide the tag names using the offline configuration *(on page)* 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.



Add Collector Instance: [redacted]

COLLECTOR NAME: [redacted]_Simulation

RUNNING MODE*

● Service - Local System Account ○ Service Under Specific User Account

Note: Once an instance of the collector is created, you can run the collector either in Windows service mode or in Command line mode.

USERNAME: Enter Username

PASSWORD: Enter Password

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 384)*.
    - If you have selected **Offline Configuration** in the **CHOOSE CONFIGURATION** field, specify the tags using the offline configuration file *(on page)*.

    The collector begins sending Historian data to the device that you have created.

Send Data to AWS IoT Core

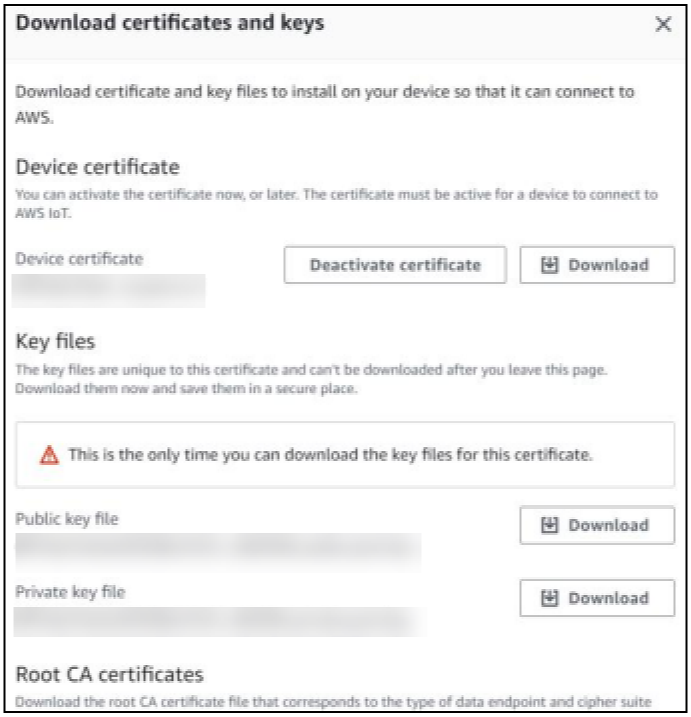
To send data to an AWS IoT Core, you can choose any of the following collectors:

- 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

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.



Download certificates and keys

Download certificate and key files to install on your device so that it can connect to AWS.

Device certificate

You can activate the certificate now, or later. The certificate must be active for a device to connect to AWS IoT.

Device certificate

Key files

The key files are unique to this certificate and can't be downloaded after you leave this page. Download them now and save them in a secure place.

This is the only time you can download the key files for this certificate.

Public key file

Private key file

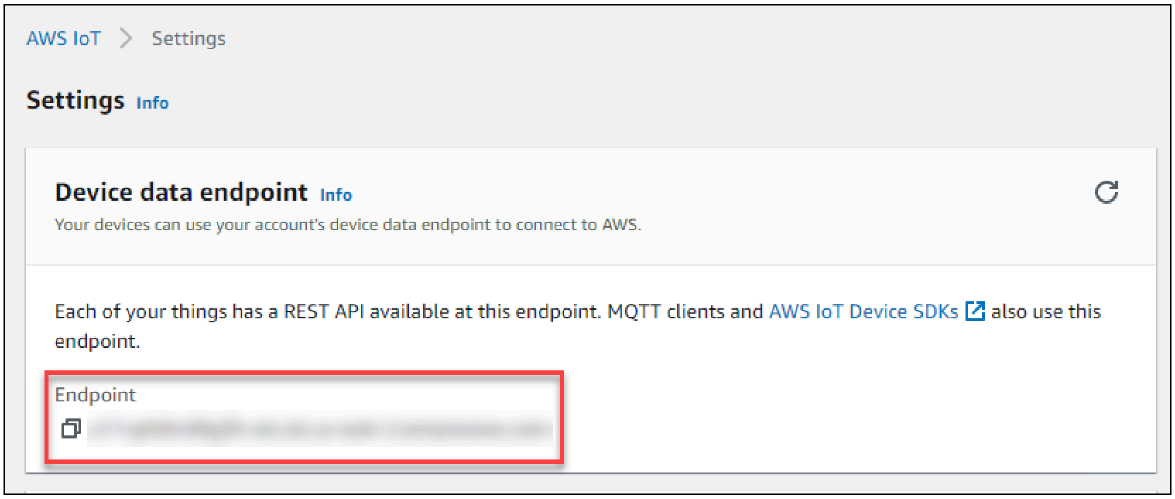
Root CA certificates

Download the root CA certificate file that corresponds to the type of data endpoint and cipher suite

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.



Settings Info

Device data endpoint Info

Your devices can use your account's device data endpoint to connect to AWS.

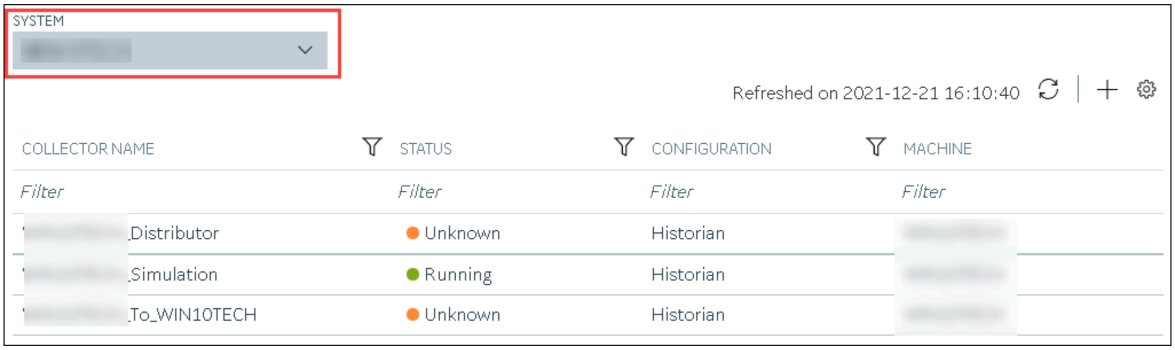
Each of your things has a REST API available at this endpoint. MQTT clients and AWS IoT Device SDKs also use this endpoint.

Endpoint

8. Access Configuration Hub *(on page 97)*.
9. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.

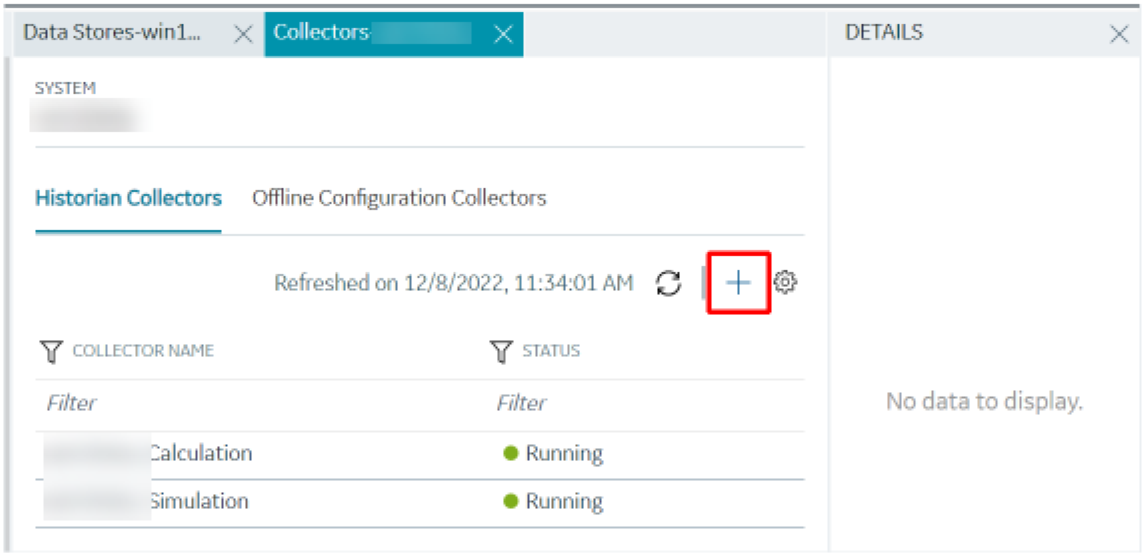
   A list of collectors in the system appears.

10. If needed, select the system in which you want to add a collector instance.



COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE
Filter	Filter	Filter	Filter
[redacted]Distributor	● Unknown	Historian	[redacted]
[redacted]Simulation	● Running	Historian	[redacted]
[redacted]To_WIN10TECH	● Unknown	Historian	[redacted]

11. In the upper-right corner of the main section, select ＋.



COLLECTOR NAME	STATUS
Filter	Filter
[redacted]Calculation	● Running
[redacted]Simulation	● Running

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**.

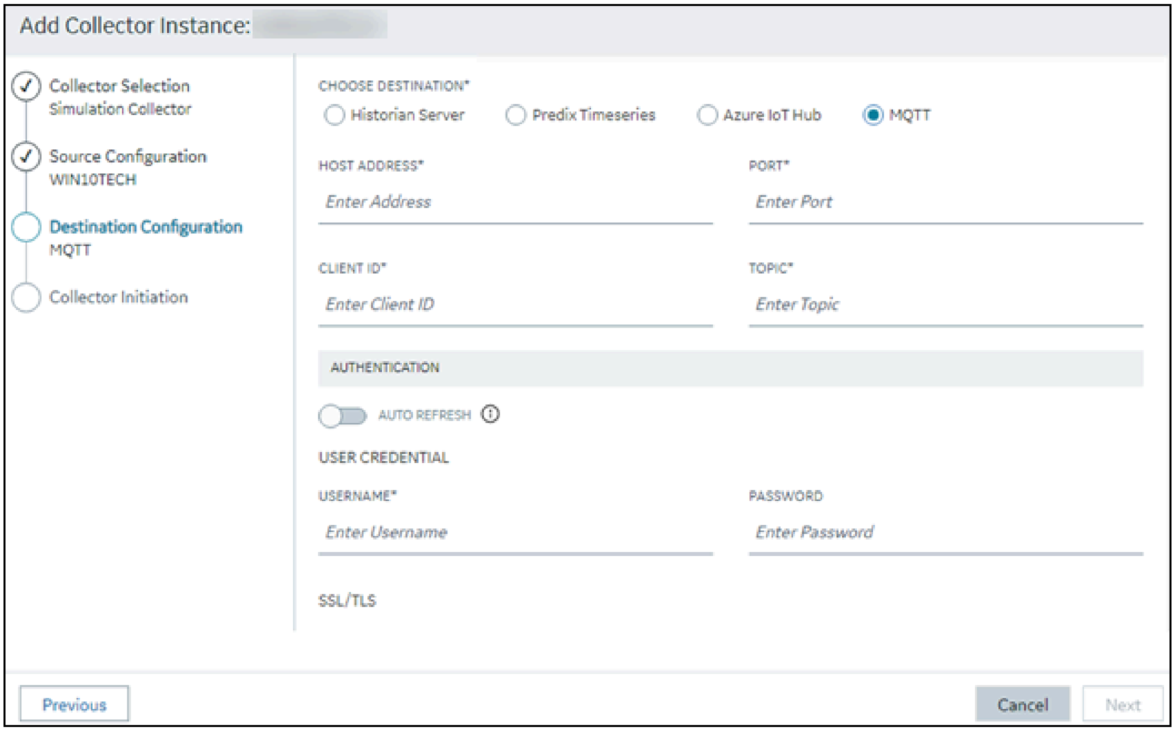
    The **INSTALLATION DRIVE** and **DATA DIRECTORY** fields are disabled and populated.

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.



Add Collector Instance: [redacted]

✓ Collector Selection
Simulation Collector

✓ Source Configuration
WIN10TECH

○ Destination Configuration
MQTT

○ Collector Initiation

CHOOSE DESTINATION*
○ Historian Server ○ Predix Timeseries ○ Azure IoT Hub ● MQTT

HOST ADDRESS*: Enter Address
PORT*: Enter Port
CLIENT ID*: Enter Client ID
TOPIC*: Enter Topic

AUTHENTICATION

AUTO REFRESH ⓘ

USER CREDENTIAL

USERNAME*: Enter Username
PASSWORD: Enter Password

SSL/TLS

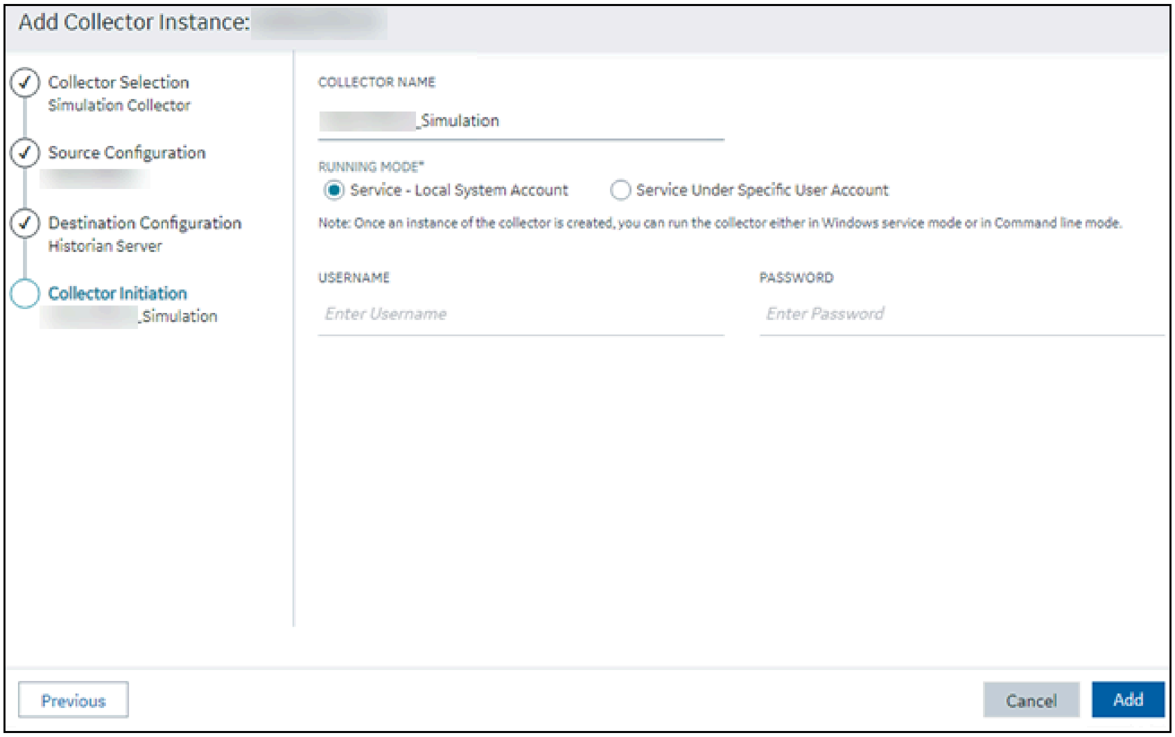
15. In the **CHOOSE DESTINATION** field, select **MQTT**, and then provide values as described in the following table.

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: ◦ **Historian Configuration**: Select this option if you want to add the tags manually using Historian Administrator *(on page 741)*. If you select this option, the **CONFIGURATION HISTORIAN SERVER** field appears. ◦ **Offline Configuration**: Select this option if you want to provide the tag names using the offline configuration *(on page)* 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.



Add Collector Instance: [redacted]

✓ Collector Selection
Simulation Collector

✓ Source Configuration
[redacted]

✓ Destination Configuration
Historian Server

○ Collector Initiation
[redacted]_Simulation

COLLECTOR NAME
[redacted]_Simulation

RUNNING MODE*
● Service - Local System Account ○ Service Under Specific User Account

Note: Once an instance of the collector is created, you can run the collector either in Windows service mode or in Command line mode.

USERNAME
Enter Username

PASSWORD
Enter Password

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**.

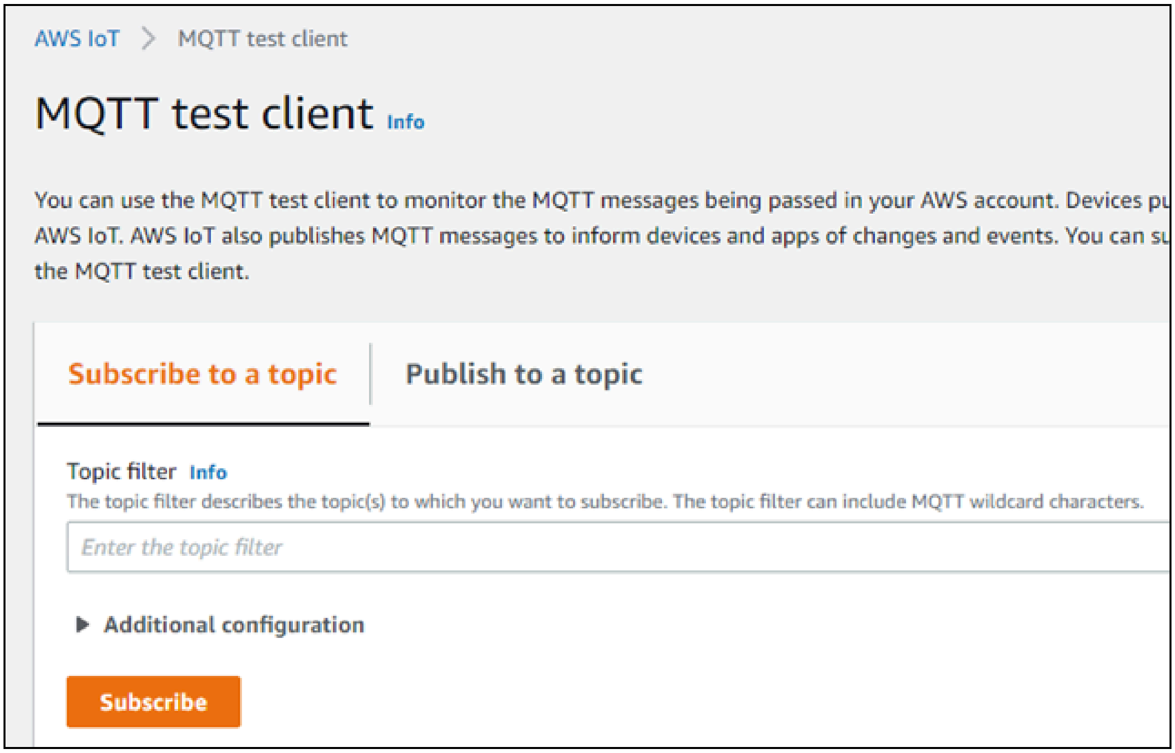
    The collector instance is created.

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 384)*.
    - If you have selected **Offline Configuration** in the **CHOOSE CONFIGURATION** field, specify the tags using the offline configuration file *(on page)*.

    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.



You can use the MQTT test client to monitor the MQTT messages being passed in your AWS account. Devices pu
AWS IoT. AWS IoT also publishes MQTT messages to inform devices and apps of changes and events. You can su
the MQTT test client.

The topic filter describes the topic(s) to which you want to subscribe. The topic filter can include MQTT wildcard characters.

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 722)* 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`

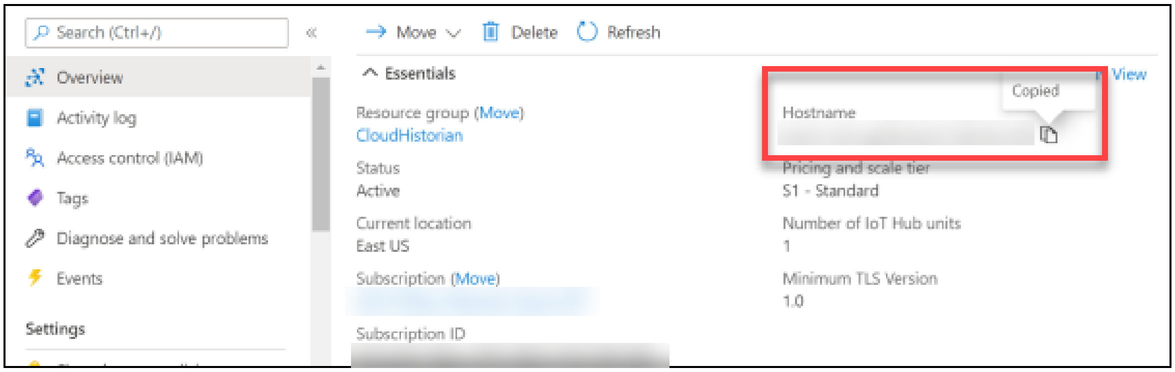
You can also send data in the KairosDB format *(on page 698)*.

> **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:



Resource group (Move)
CloudHistorian

Status
Active

Current location
East US

Subscription (Move)
[redacted]

Subscription ID
[redacted]

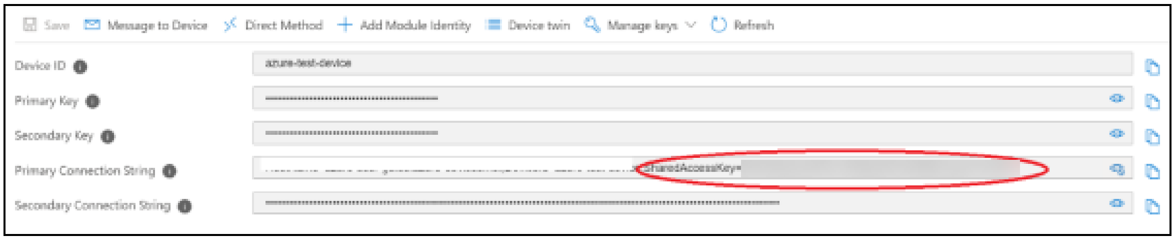
Hostname
[redacted]

Pricing and scale tier
S1 - Standard

Number of IoT Hub units
1

Minimum TLS Version
1.0

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.



Device ID: azure-test-device

Primary Key: ••

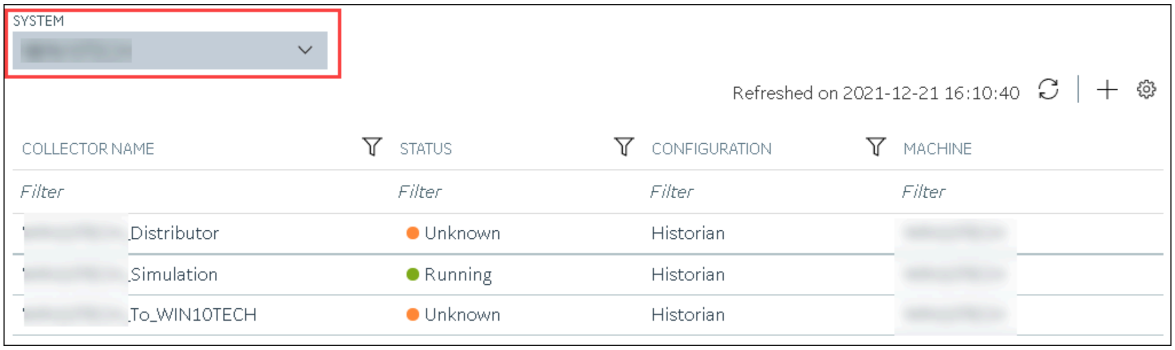
Secondary Key: ••

Primary Connection String: [redacted] SharedAccessKey=[redacted]

Secondary Connection String: ••

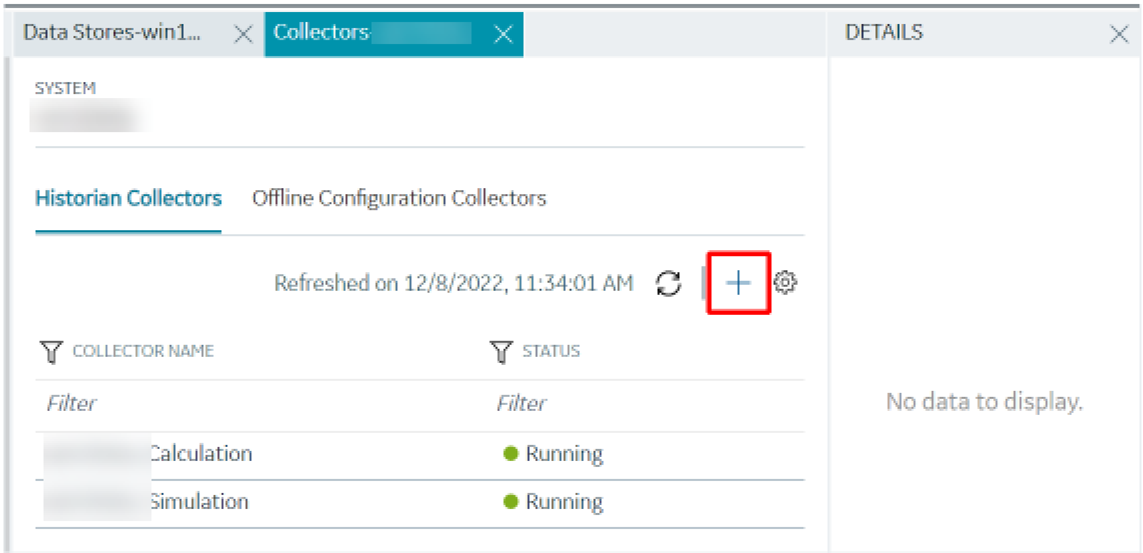
5. Access Configuration Hub *(on page 97)*.
6. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.

   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.



COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE
Filter	Filter	Filter	Filter
[redacted]_Distributor	● Unknown	Historian	[redacted]
[redacted]_Simulation	● Running	Historian	[redacted]
[redacted]_To_WIN10TECH	● Unknown	Historian	[redacted]

9. In the upper-right corner of the main section, select +.



COLLECTOR NAME	STATUS
Filter	Filter
[redacted]Calculation	● Running
[redacted]Simulation	● Running

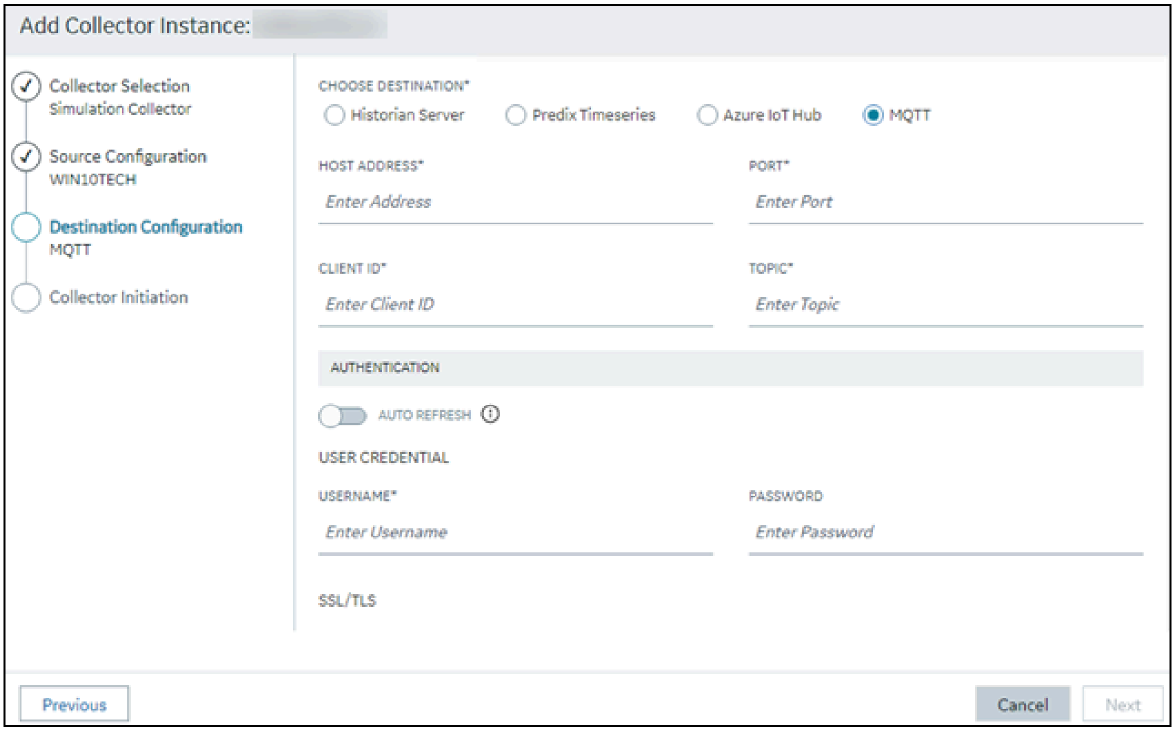
The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.

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.

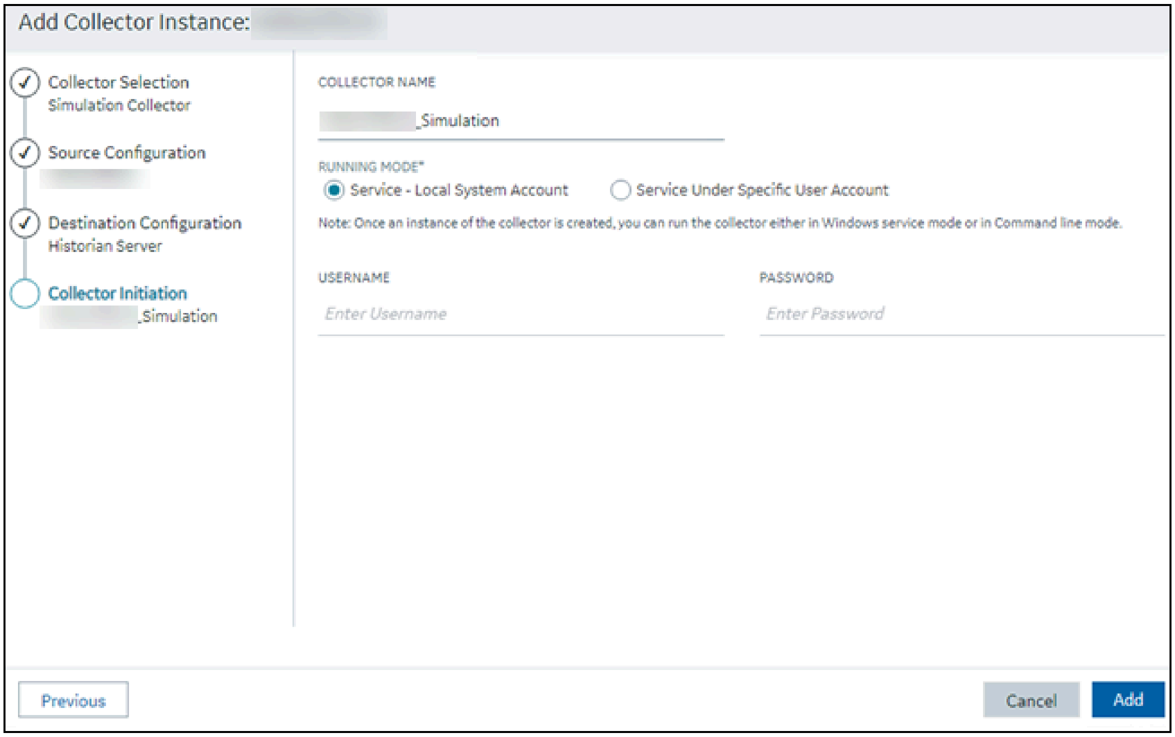
Field	Description
	◦ If you switch the toggle off, you must manually provide the token as soon as it expires. ◦ If you switch the toggle on, you must provide the shared access key that you have noted down in step 4. And, you can leave the **PASSWORD** field blank. This is applicable only if you have 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. 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. ◦ **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: ◦ **X.509 Self-Signed**: If you have selected this option, you can generate the key file using OpenSSL. ◦ **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: ◦ **X.509 Self-Signed**: If you have selected this option, you can generate the key file using OpenSSL. ◦ **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: ◦ **Historian Configuration**: Select this option if you want to add the tags manually *(on page 384)*. If you select this option, the **CONFIGURATION HISTORIAN SERVER** field appears. ◦ **Offline Configuration**: Select this option if you want to provide the tag names using the offline configuration *(on page*

Field	Description
	) 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.



Add Collector Instance: [redacted]

✓ Collector Selection
Simulation Collector

✓ Source Configuration
[redacted]

✓ Destination Configuration
Historian Server

○ Collector Initiation
[redacted]_Simulation

COLLECTOR NAME
[redacted]_Simulation

RUNNING MODE*
● Service - Local System Account ○ Service Under Specific User Account

Note: Once an instance of the collector is created, you can run the collector either in Windows service mode or in Command line mode.

USERNAME
Enter Username

PASSWORD
Enter Password

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 384)*.
    - If you have selected **Offline Configuration** in the **CHOOSE CONFIGURATION** field, specify the tags using the offline configuration file *(on page)*.

    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.

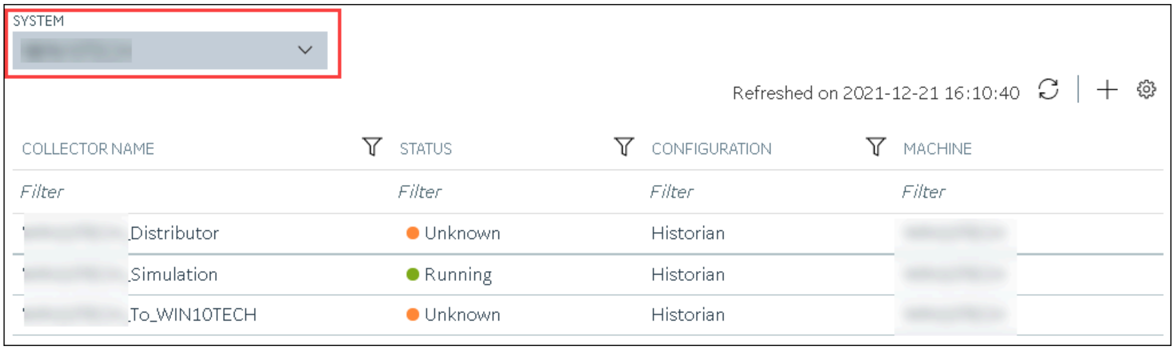
   Note down the registry ID and the region values.
4. Add a device to the registry.

   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 97)*.
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.



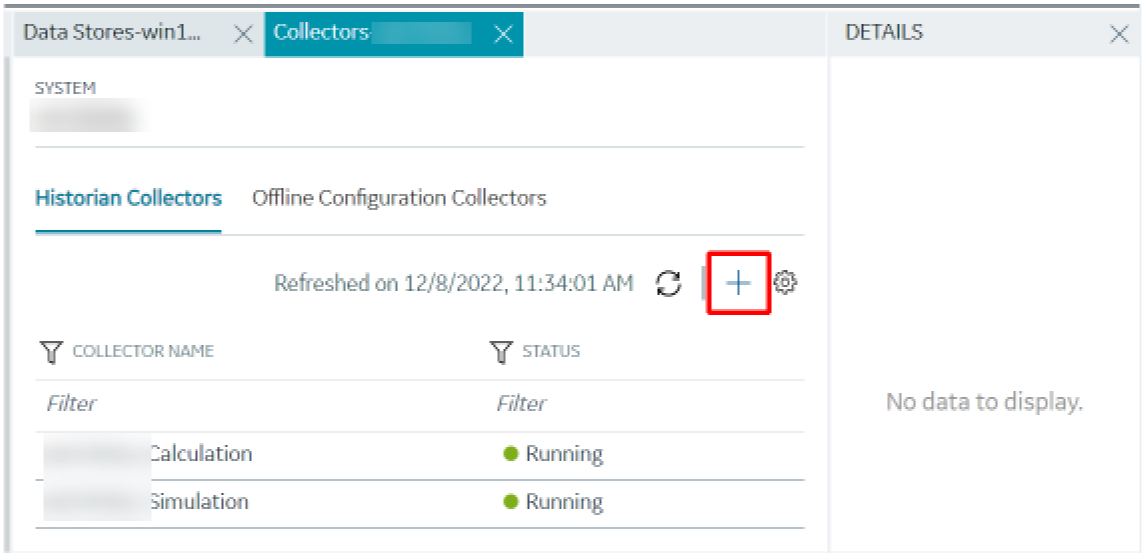
SYSTEM

[redacted]

Refreshed on 2021-12-21 16:10:40

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE
Filter	Filter	Filter	Filter
[redacted]Distributor	● Unknown	Historian	[redacted]
[redacted]Simulation	● Running	Historian	[redacted]
[redacted]To_WIN10TECH	● Unknown	Historian	[redacted]

9. In the upper-right corner of the main section, select +.



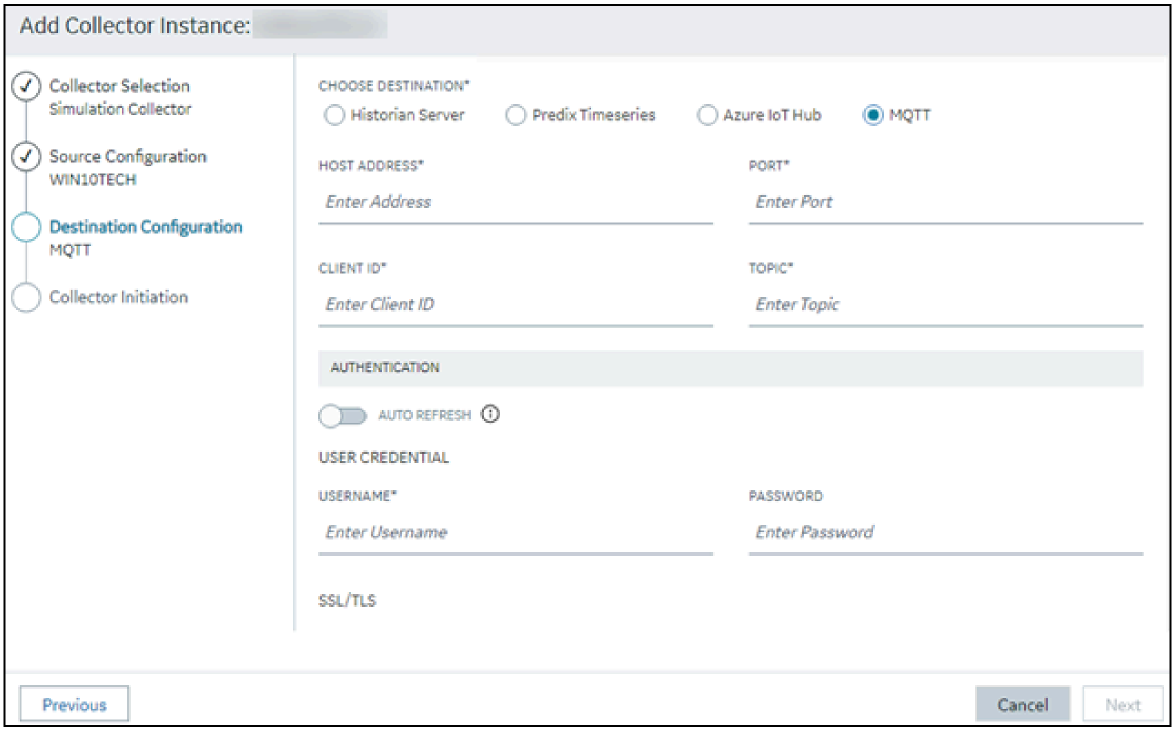
The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.

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.



Add Collector Instance: [redacted]

✓ Collector Selection
Simulation Collector

✓ Source Configuration
WIN10TECH

○ Destination Configuration
MQTT

○ Collector Initiation

CHOOSE DESTINATION*
○ Historian Server ○ Predix Timeseries ○ Azure IoT Hub ● MQTT

HOST ADDRESS*: Enter Address
PORT*: Enter Port
CLIENT ID*: Enter Client ID
TOPIC*: Enter Topic

AUTHENTICATION

AUTO REFRESH

USER CREDENTIAL

USERNAME*: Enter Username
PASSWORD: Enter Password

SSL/TLS

14. Select **MQTT**, and provide values as described in the following table.

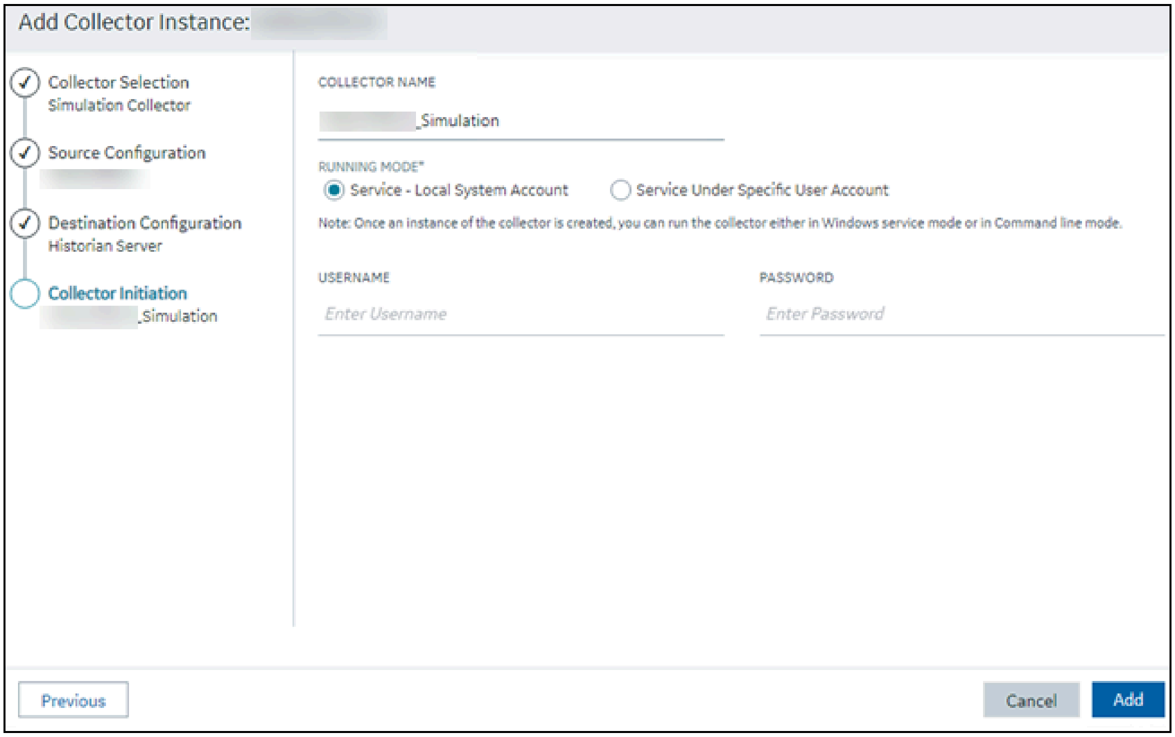
Field	Description
HOST ADDRESS	Enter `mqtt.googleapis.com`. A value is required.
PORT	Enter `8883` or `443`.
CLIENT ID	Enter the ID of the device that you created in the following format: `projects/<project ID>/locations/<cloud region>/registries/<registry ID>/devices/<device ID>`. For example: `projects/mygcpproject/locations/asia-east1/registries/testmqttgcpiot/devices/gcptesting` 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: ◦ **Historian Configuration**: Select this option if you want to add the tags manually *(on page 384)*. If you select this option, the **CONFIGURATION HISTORIAN SERVER** field appears. ◦ **Offline Configuration**: Select this option if you want to provide the tag names using the offline configuration *(on page)* 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.



Add Collector Instance: [redacted]

COLLECTOR NAME

[redacted]_Simulation

RUNNING MODE*

● Service - Local System Account ○ Service Under Specific User Account

Note: Once an instance of the collector is created, you can run the collector either in Windows service mode or in Command line mode.

USERNAME

Enter Username

PASSWORD

Enter Password

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.

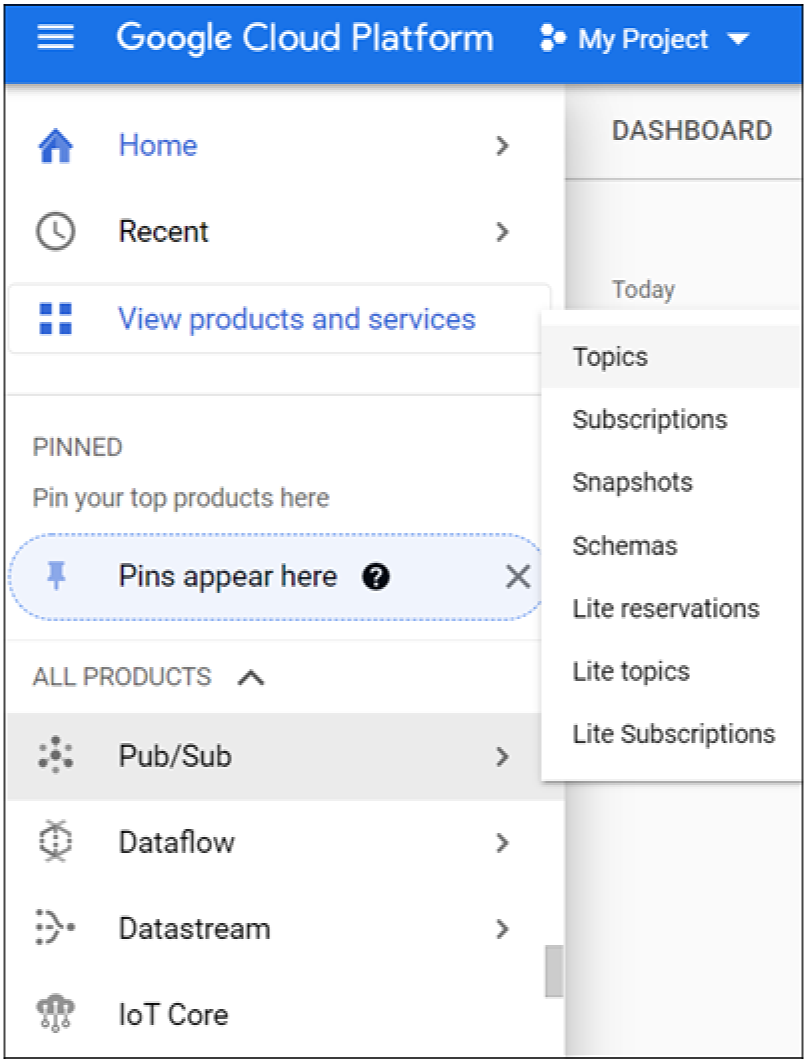
17. In the **RUNNING MODE** field, select one of the following options.

- **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**.

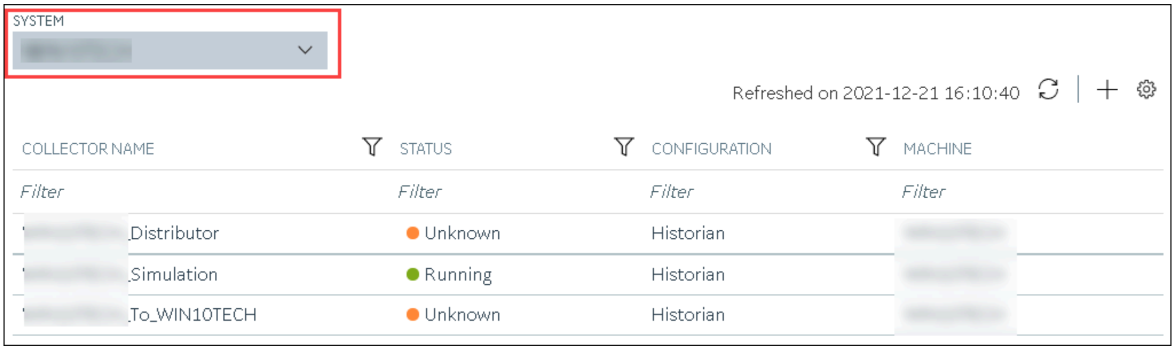
Send Data to Predix Cloud

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 Proficy Authentication 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 97)*.
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.

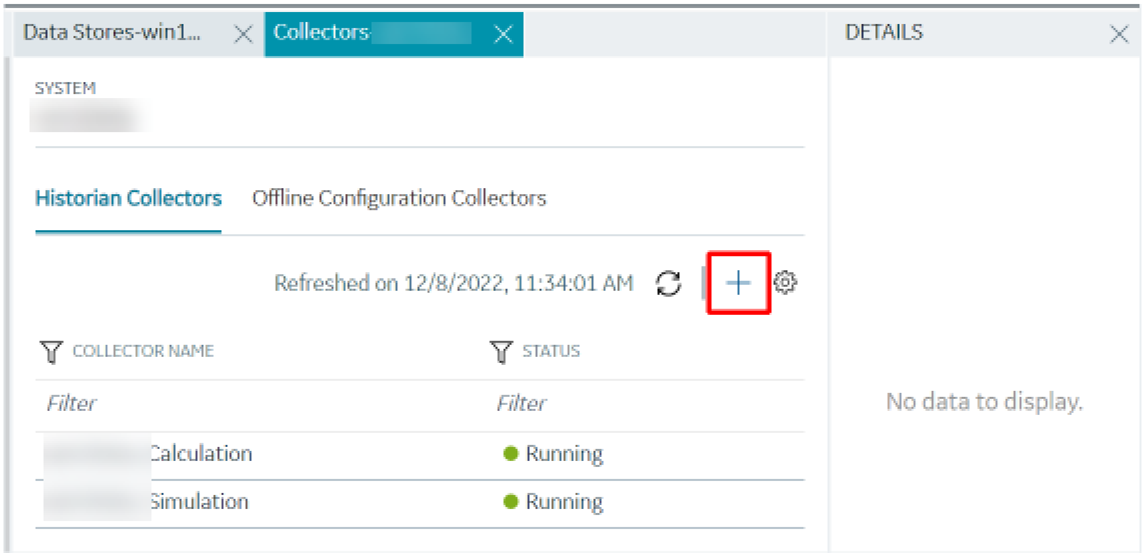


SYSTEM [redacted]

Refreshed on 2021-12-21 16:10:40

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE
Filter	*Filter*	*Filter*	*Filter*
[redacted]_Distributor	● Unknown	Historian	[redacted]
[redacted]_Simulation	● Running	Historian	[redacted]
[redacted]_To_WIN10TECH	● Unknown	Historian	[redacted]

5. In the upper-right corner of the main section, select +.



SYSTEM [redacted]

Refreshed on 12/8/2022, 11:34:01 AM

COLLECTOR NAME	STATUS
Filter	*Filter*
[redacted]Calculation	● Running
[redacted]Simulation	● Running

No data to display.

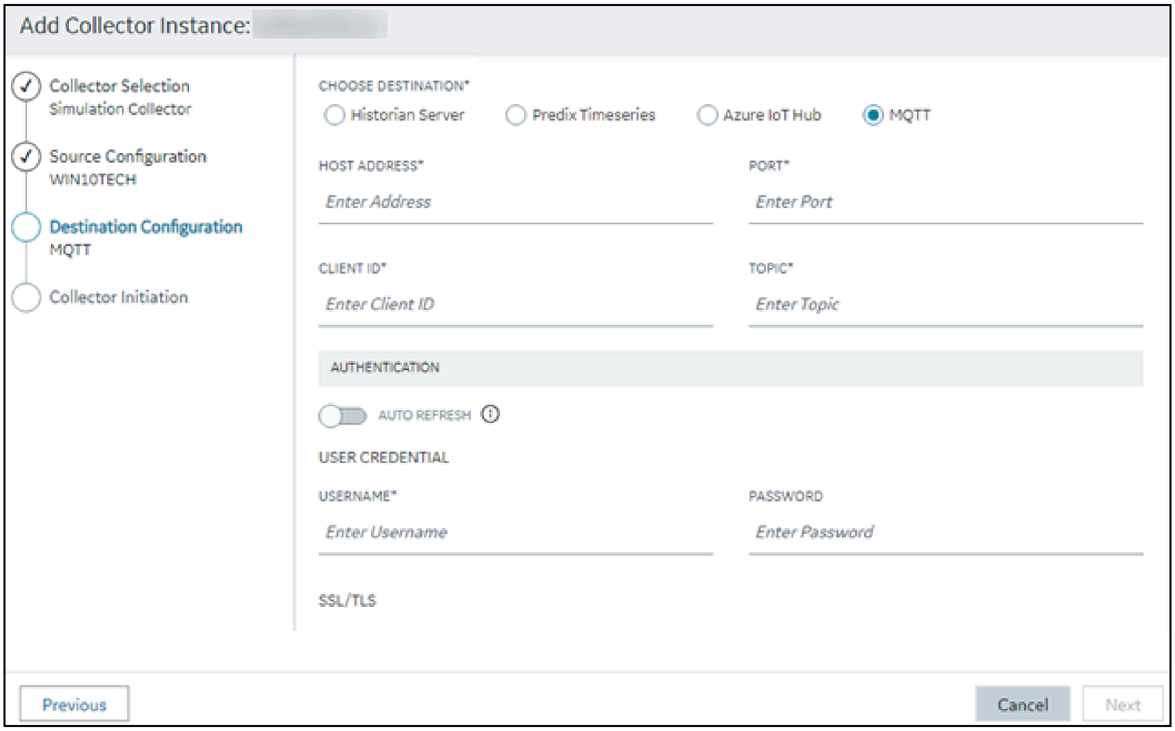
The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.

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.

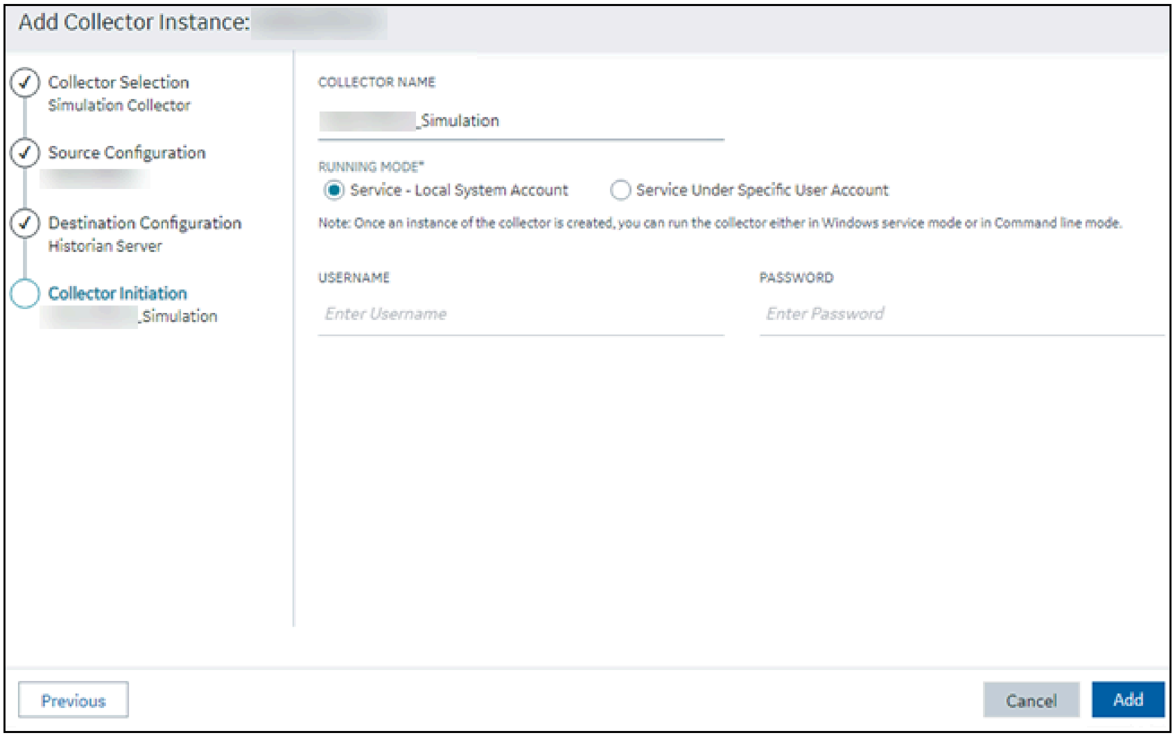
Field	Description
CLOUD DESTINATION ADDRESS	The URL of a data streaming endpoint exposed by the Predix Time Series instance to which you want to send data. Typically, it starts with “wss://”. This value is used as part of the interface name and default tag prefix of the collector. Your Predix Time Series 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: ◦ **Historian Configuration**: Select this option if you want to add the tags manually *(on page 384)*. If you select this option, the **CONFIGURATION HISTORIAN SERVER** field appears. ◦ **Offline Configuration**: Select this option if you want to provide the tag names using the offline configuration *(on page)* 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.



Add Collector Instance: [redacted]

COLLECTOR NAME
[redacted]_Simulation

RUNNING MODE*
● Service - Local System Account ○ Service Under Specific User Account

Note: Once an instance of the collector is created, you can run the collector either in Windows service mode or in Command line mode.

USERNAME
Enter Username

PASSWORD
Enter Password

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 384)*.
    - If you have selected **Offline Configuration** in the **CHOOSE CONFIGURATION** field, specify the tags using the offline configuration file *(on page)*.

    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

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.

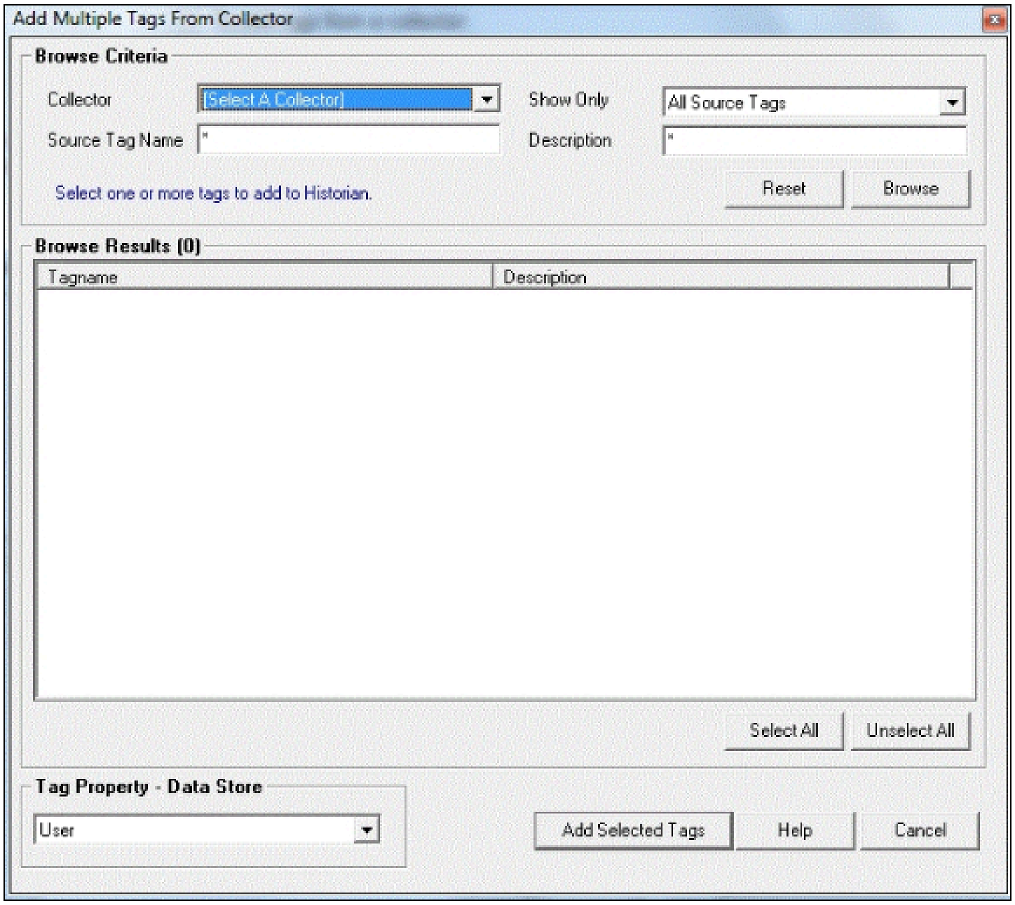
Adding Tags from a Collector

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 190)*.

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.



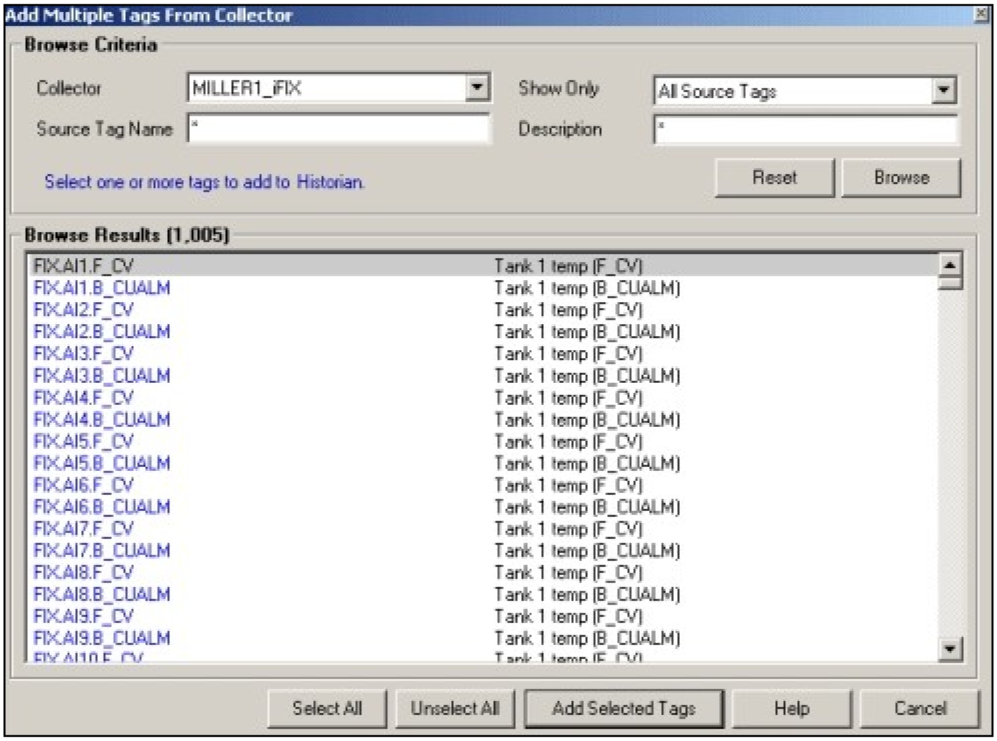
2. Select a collector from the **Collector** drop-down list.

> **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.



Collector: MILLER1_iFIX
Show Only: All Source Tags
Source Tag Name: *
Description: *

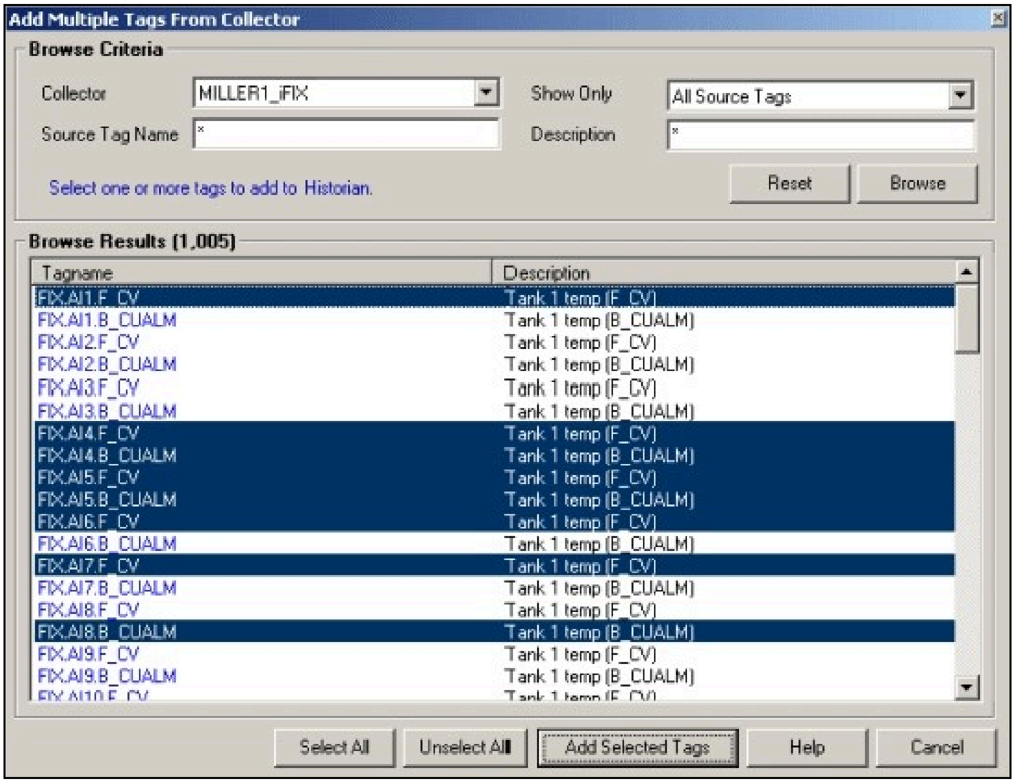
Select one or more tags to add to Historian.

FIX.AI1.F_CV	Tank 1 temp (F_CV)
FIX.AI1.B_CUALM	Tank 1 temp (B_CUALM)
FIX.AI2.F_CV	Tank 1 temp (F_CV)
FIX.AI2.B_CUALM	Tank 1 temp (B_CUALM)
FIX.AI3.F_CV	Tank 1 temp (F_CV)
FIX.AI3.B_CUALM	Tank 1 temp (B_CUALM)
FIX.AI4.F_CV	Tank 1 temp (F_CV)
FIX.AI4.B_CUALM	Tank 1 temp (B_CUALM)
FIX.AI5.F_CV	Tank 1 temp (F_CV)
FIX.AI5.B_CUALM	Tank 1 temp (B_CUALM)
FIX.AI6.F_CV	Tank 1 temp (F_CV)
FIX.AI6.B_CUALM	Tank 1 temp (B_CUALM)
FIX.AI7.F_CV	Tank 1 temp (F_CV)
FIX.AI7.B_CUALM	Tank 1 temp (B_CUALM)
FIX.AI8.F_CV	Tank 1 temp (F_CV)
FIX.AI8.B_CUALM	Tank 1 temp (B_CUALM)
FIX.AI9.F_CV	Tank 1 temp (F_CV)
FIX.AI9.B_CUALM	Tank 1 temp (B_CUALM)
FIX.AI10.F_CV	Tank 1 temp (F_CV)

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**.



Collector: MILLER1_iFIX

Show Only: All Source Tags

Source Tag Name: *

Description: *

Select one or more tags to add to Historian.

Tagname	Description
FIX.AI1.F_CV	Tank 1 temp (F_CV)
FIX.AI1.B_CUALM	Tank 1 temp (B_CUALM)
FIX.AI2.F_CV	Tank 1 temp (F_CV)
FIX.AI2.B_CUALM	Tank 1 temp (B_CUALM)
FIX.AI3.F_CV	Tank 1 temp (F_CV)
FIX.AI3.B_CUALM	Tank 1 temp (B_CUALM)
FIX.AI4.F_CV	Tank 1 temp (F_CV)
FIX.AI4.B_CUALM	Tank 1 temp (B_CUALM)
FIX.AI5.F_CV	Tank 1 temp (F_CV)
FIX.AI5.B_CUALM	Tank 1 temp (B_CUALM)
FIX.AI6.F_CV	Tank 1 temp (F_CV)
FIX.AI6.B_CUALM	Tank 1 temp (B_CUALM)
FIX.AI7.F_CV	Tank 1 temp (F_CV)
FIX.AI7.B_CUALM	Tank 1 temp (B_CUALM)
FIX.AI8.F_CV	Tank 1 temp (F_CV)
FIX.AI8.B_CUALM	Tank 1 temp (B_CUALM)
FIX.AI9.F_CV	Tank 1 temp (F_CV)
FIX.AI9.B_CUALM	Tank 1 temp (B_CUALM)
FIX.AI10.F_CV	Tank 1 temp (F_CV)

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 187)*.

> **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.

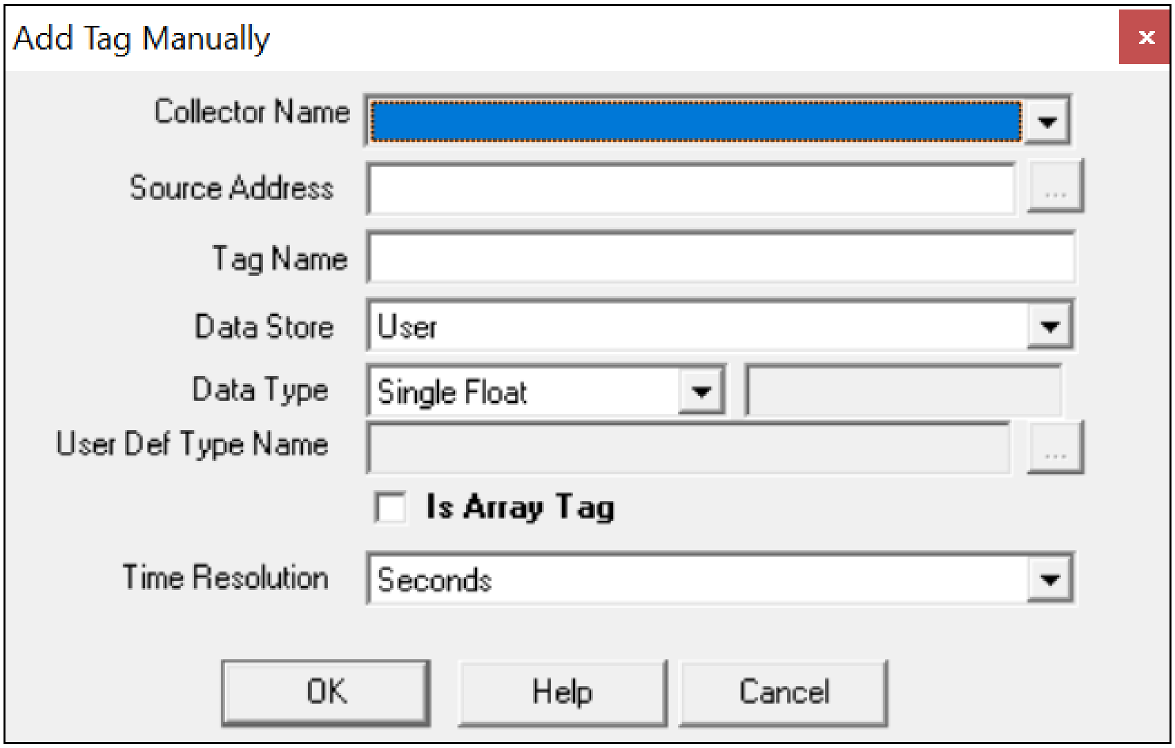
For example, if you are currently not connected to a collector, the browse is slow, or is not supported, and if you want to configure tags associated with that collector, you can add them manually.

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 556)* that you want to use to collect the tag data.
2. Create a data store *(on page)* 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.



Add Tag Manually

Collector Name

Source Address

Tag Name

Data Store User

Data Type Single Float

User Def Type Name

□ Is Array Tag

Time Resolution Seconds

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.

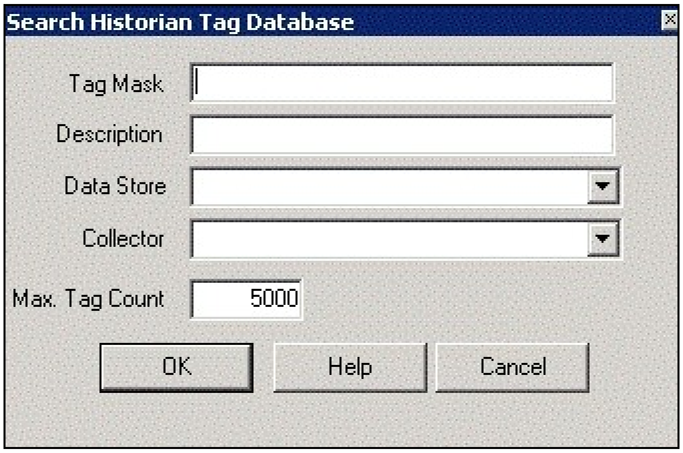
   c. Type a new tag name.

   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 191)* to speed browse sessions.

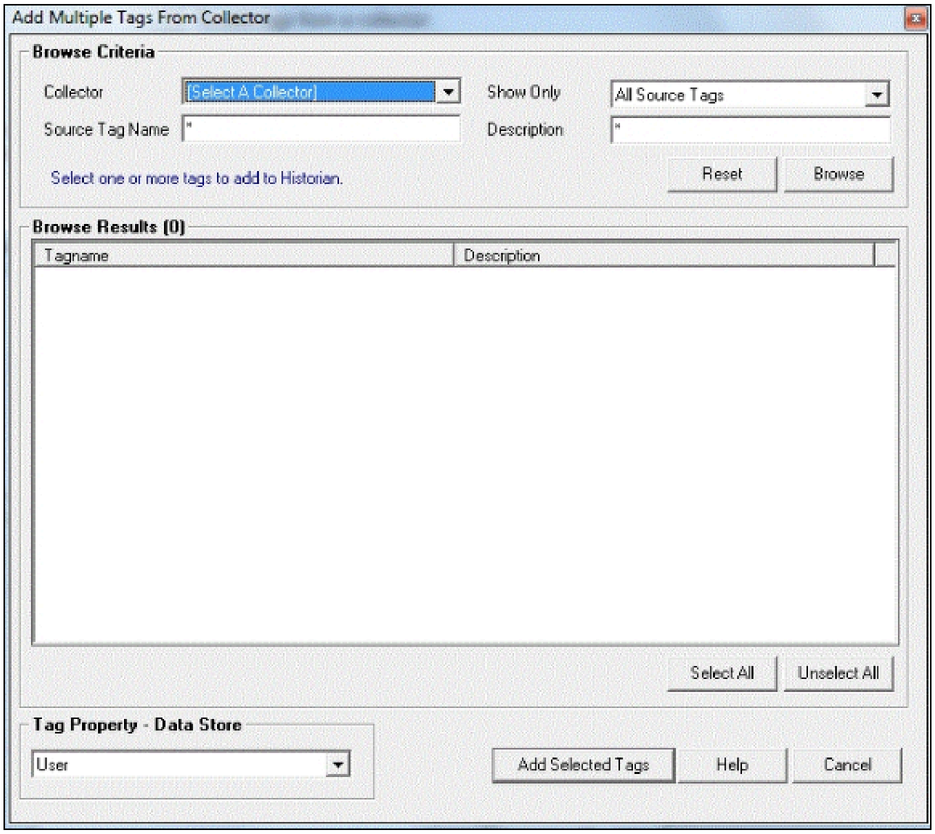
1. To browse for new tags using Historian Administrator:

   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.



Add Multiple Tags From Collector

Browse Criteria

Collector: [Select A Collector]

Show Only: All Source Tags

Source Tag Name: *

Description: *

Select one or more tags to add to Historian.

Browse Results (0)

Tag Property - Data Store: User

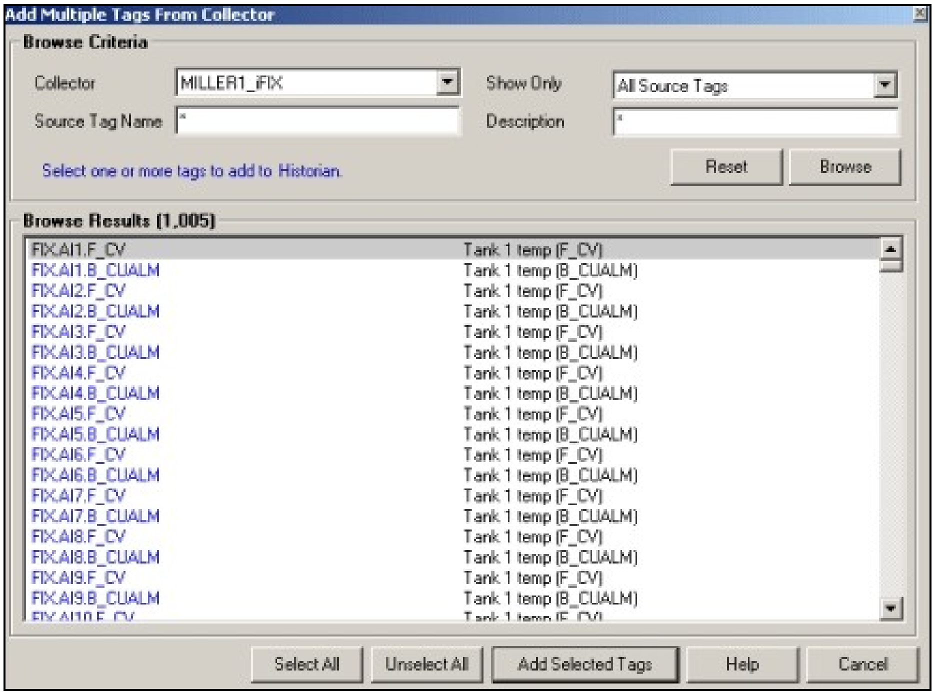
e. In the **Show Only** field, select either **All Source Tags** or **Source Tags Not Collected**. If you select the second option, the browse returns only the tags that are not currently included for collection.

   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.



Collector: MILLER1_iFIX

Show Only: All Source Tags

Source Tag Name: *

Description: *

Select one or more tags to add to Historian.

Browse Results (1,005)

FIX.AI1.F_CV	Tank 1 temp (F_CV)
FIX.AI1.B_CUALM	Tank 1 temp (B_CUALM)
FIX.AI2.F_CV	Tank 1 temp (F_CV)
FIX.AI2.B_CUALM	Tank 1 temp (B_CUALM)
FIX.AI3.F_CV	Tank 1 temp (F_CV)
FIX.AI3.B_CUALM	Tank 1 temp (B_CUALM)
FIX.AI4.F_CV	Tank 1 temp (F_CV)
FIX.AI4.B_CUALM	Tank 1 temp (B_CUALM)
FIX.AI5.F_CV	Tank 1 temp (F_CV)
FIX.AI5.B_CUALM	Tank 1 temp (B_CUALM)
FIX.AI6.F_CV	Tank 1 temp (F_CV)
FIX.AI6.B_CUALM	Tank 1 temp (B_CUALM)
FIX.AI7.F_CV	Tank 1 temp (F_CV)
FIX.AI7.B_CUALM	Tank 1 temp (B_CUALM)
FIX.AI8.F_CV	Tank 1 temp (F_CV)
FIX.AI8.B_CUALM	Tank 1 temp (B_CUALM)
FIX.AI9.F_CV	Tank 1 temp (F_CV)
FIX.AI9.B_CUALM	Tank 1 temp (B_CUALM)
FIX.AI10.F_CV	Tank 1 temp (F_CV)

See also Add Multiple Tags from a Collector Using Historian Administrator *(on page 187)*.

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 190)*.

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*.

  .

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.

Troubleshooting Tag Configuration

To troubleshoot the tag configuration:

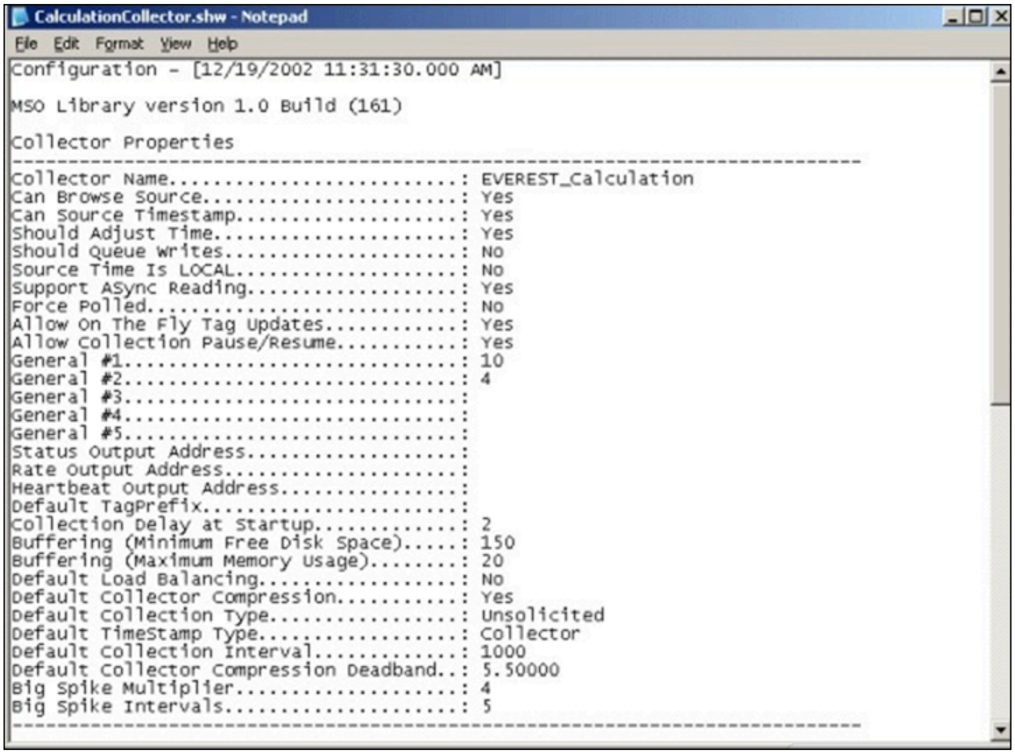
- View the Collector SHOW (`DataCollector.shw`) file in `C:\Historian Data\LogFiles \xxxCollector.shw` to review the current Data Collector configuration.
- View the `DataCollector.LOG` file in `C:\Proficy Historian Data\LogFiles \xxxCollector-01.log` to detect operational errors with Tags.
- If you 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:


CalculationCollector.shw - Notepad
File Edit Format View Help

```
Configuration - [12/19/2002 11:31:30.000 AM]

MSO Library version 1.0 Build (161)

Collector Properties
-----------------------------------------------------------------------------
Collector Name..........................: EVEREST_Calculation
Can Browse Source.......................: Yes
Can Source Timestamp....................: Yes
Should Adjust Time......................: Yes
Should Queue Writes.....................: No
Source Time Is LOCAL....................: No
Support ASync Reading...................: Yes
Force Polled............................: No
Allow On The Fly Tag Updates............: Yes
Allow Collection Pause/Resume...........: Yes
General #1..............................: 10
General #2..............................: 4
General #3..............................:
General #4..............................:
General #5..............................:
Status Output Address...................:
Rate Output Address.....................:
Heartbeat Output Address................:
Default TagPrefix.......................:
Collection Delay at Startup.............: 2
Buffering (Minimum Free Disk Space).....: 150
Buffering (Maximum Memory Usage)........: 20
Default Load Balancing..................: No
Default Collector Compression...........: Yes
Default Collection Type.................: Unsolicited
Default TimeStamp Type..................: Collector
Default Collection Interval.............: 1000
Default Collector Compression Deadband..: 5.50000
Big Spike Multiplier....................: 4
Big Spike Intervals.....................: 5
-----------------------------------------------------------------------------
```

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 `uom` (unit of measure conversion module)	Unit of Measure Python Module.pdf (This document can also be found in the help directory for your Historian installation.)	Yes

Constructing and Adding Python Expression Tags

To configure and use the Python Expression Tag in Historian:

1. Construct a Python Expression. *(on page 214)*
2. Construct the JSON configuration. *(on page 214)*
3. Add the Python Expression Tag to Historian. *(on page 216)*

   For examples, see the Python Expression Tag Examples *(on page 218)*.

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 216)*.

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 213)*.

After `"script"`, enter the python expression that you created in Constructing a Python Expression *(on page 214)*.

Provide values for the following parameters:

Parameter	Description
name	The variable name you gave to the object representing the raw data which will be preprocessed 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 212)* 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. The datatype options are: `SingleFloat, DoubleFloat, SingleInteger, DoubleInteger, QuadInteger, UnsignedSingleInteger, UnsignedDoubleInteger, UnsignedQuadInteger, 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 213)* 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 214)*.

```
"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 212)* 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 213)* 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 212)* 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 212)* 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 212)* 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 213)* 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 213)* 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 739)* or Remote Collector Management *(on page 1355)*. 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)*.

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.

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.

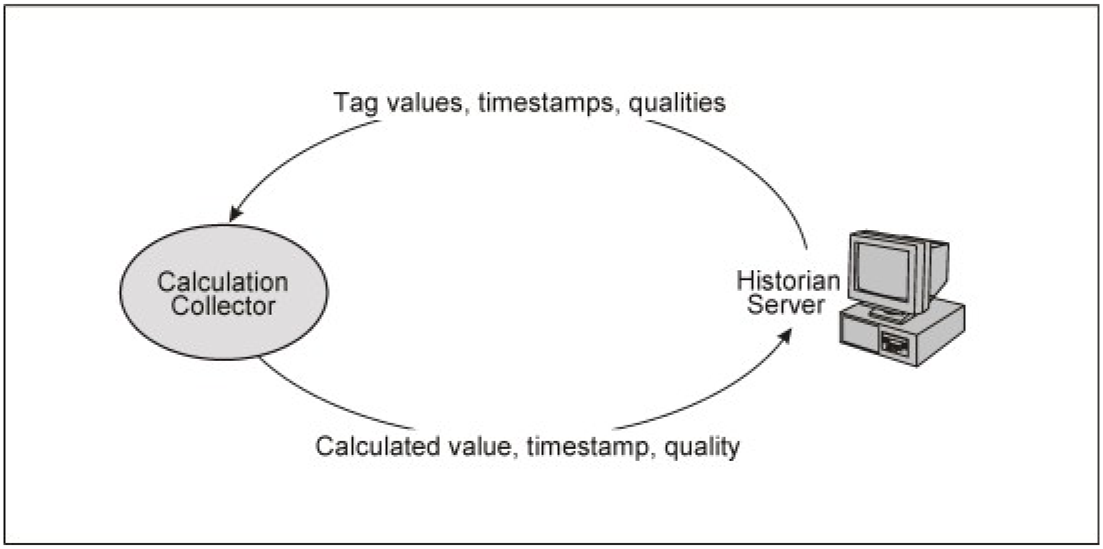
Features of the Calculation Collector

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.


Tag values, timestamps, qualities
Calculation Collector
Historian Server
Calculated value, timestamp, quality

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

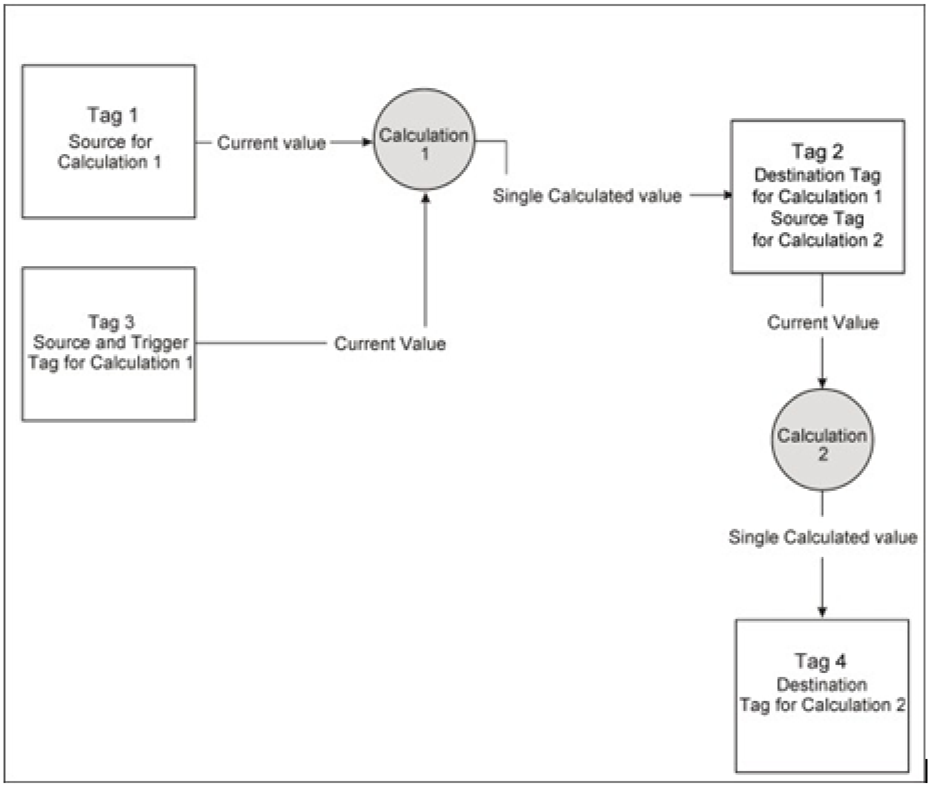
- Python Expression Tags are not supported for use together with the Calculation collector. The behavior of the Calculation collector is different from that of Python Expression Tags, which are used in cases where you do not want to store a raw data value, but wish to store only derived values. For more information, refer to Python Expression Tag Examples *(on page 218)*.
- 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 245)* or unsolicited (event-based) *(on page 247)*.
- **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)*. 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 234)*.	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 232)*. To do so, you can add a tag manually using Historian Administrator *(on page 192)* or the Web Admin console *(on page 194)*. Or, you can copy a tag *(on page 195)*. **Note:** 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 247)*.	This step is required only if the tag is unsolicited (that is, event-based).
4	Define the calculation formula *(on page 251)*.	This step is required.

Configure the Calculation Collector

1. Install the Historian server *(on page)* and collectors *(on page 90)*.
2. Create an instance of the collector using Configuration Hub *(on page 595)* or the RemoteCollectorConfigurator utility *(on page 1335)*.

1. To configure the collector using Configuration Hub:

    a. Access Configuration Hub *(on page 97)*.

    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. If you want to disable automatic calculation of the tag, set the value of this field to 0.

    d. As needed, enter values in the other sections common to all collectors *(on page 669)*.

    e. Restart the collector.

2. To configure the collector using Historian Administrator:

    a. Access Historian Administrator *(on page)*.

    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 calcula-

Field	Description
	tion 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. 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

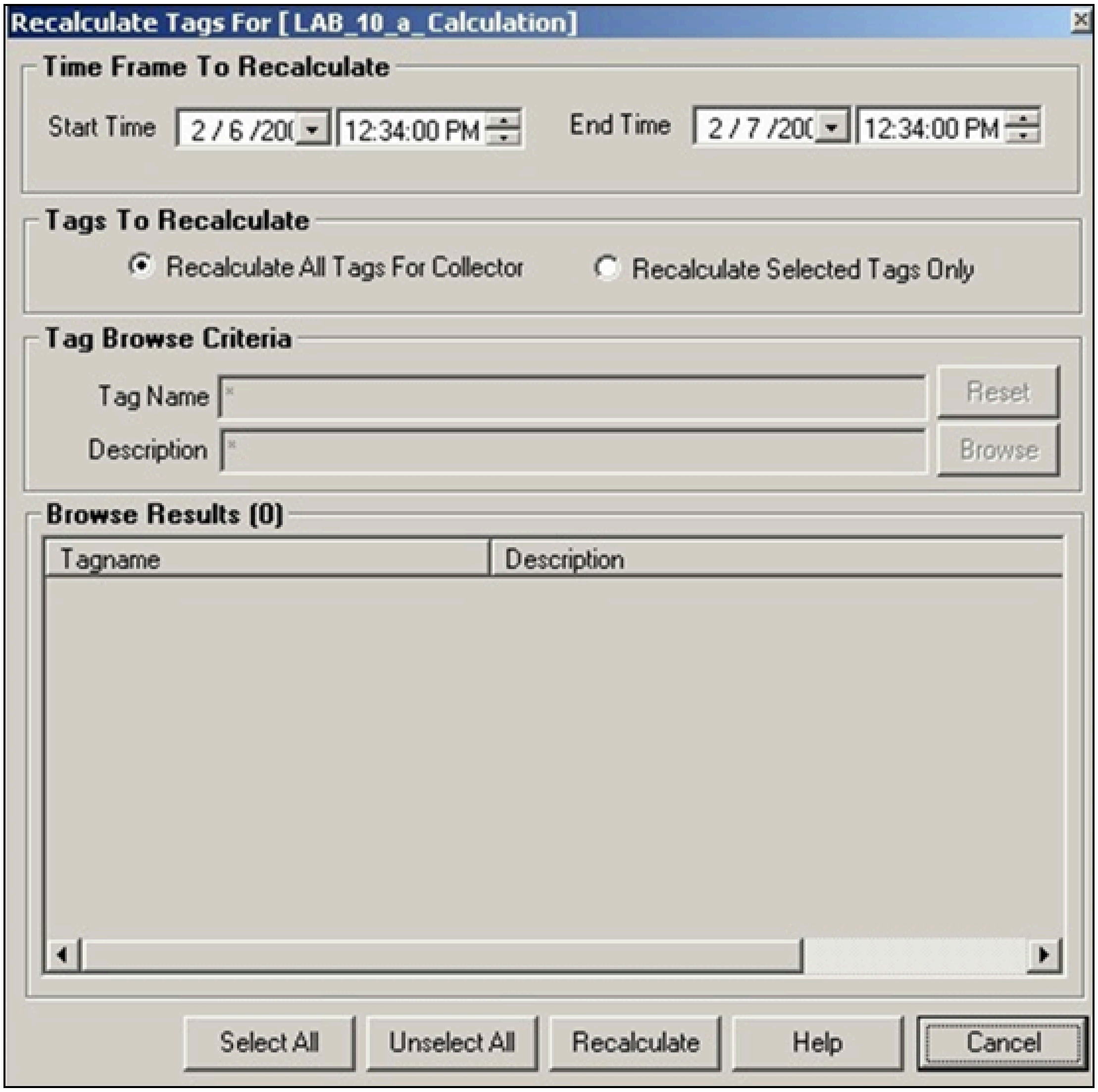
When the connection between the collector and the Historian server is lost, the collector buffers data. When the connection is lost, the buffered data is sent to the Historian server. When the buffered data arrives, the timestamps are earlier than the most recent calculation timestamp. Since the timestamp 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)*.
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.



Recalculate Tags For [LAB_10_a_Calculation]

Time Frame To Recalculate

Start Time 2 / 6 /20(12:34:00 PM

End Time 2 / 7 /20(12:34:00 PM

Tags To Recalculate

● Recalculate All Tags For Collector

○ Recalculate Selected Tags Only

Tag Browse Criteria

Tag Name *

Description *

Browse Results (0)

Tagname	Description

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 240)*. For information on the status codes and their descriptions, refer to Status Codes of the AddData Function *(on page 242)*.

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_S-PACE`	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 245)*.
- **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 247)*.

Create a Polled Trigger

1. To create a poller trigger using Configuration Hub:

    a. Access Configuration Hub *(on page 97)*.

    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)*.
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 246)*.
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 97)*.

    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)*.

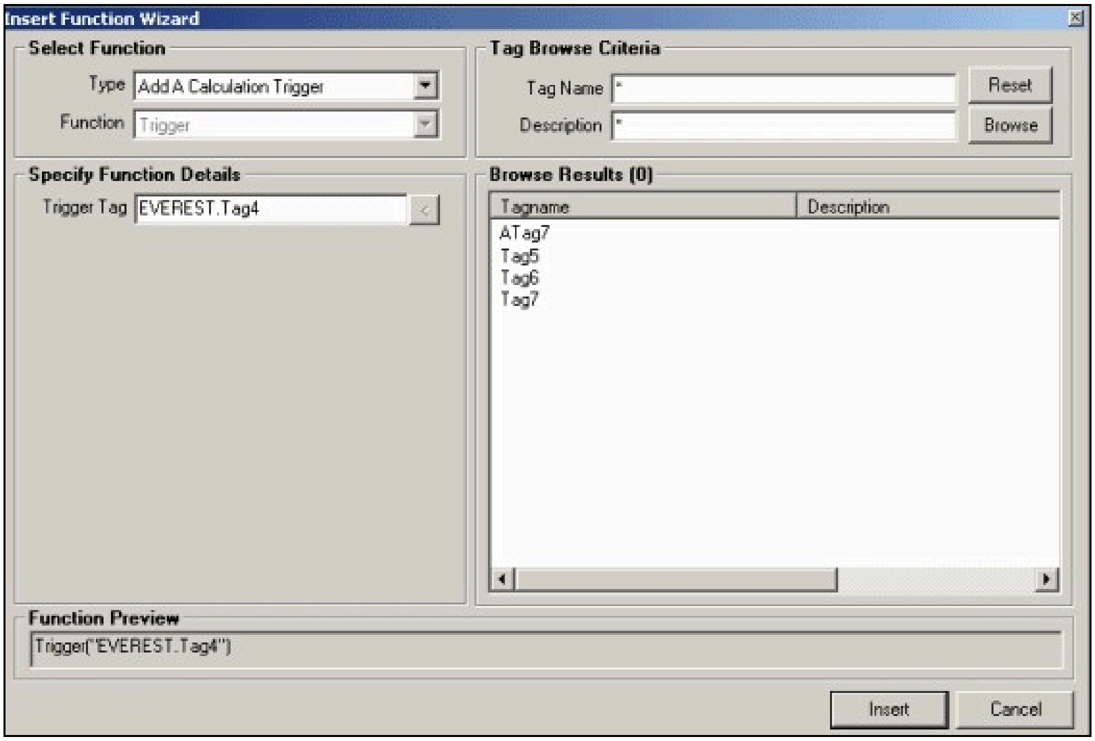
   b. Select **Tags**, and then select the tag to which you want to apply the trigger.

   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:



Type: Add A Calculation Trigger

Function: Trigger

Tag Name: *

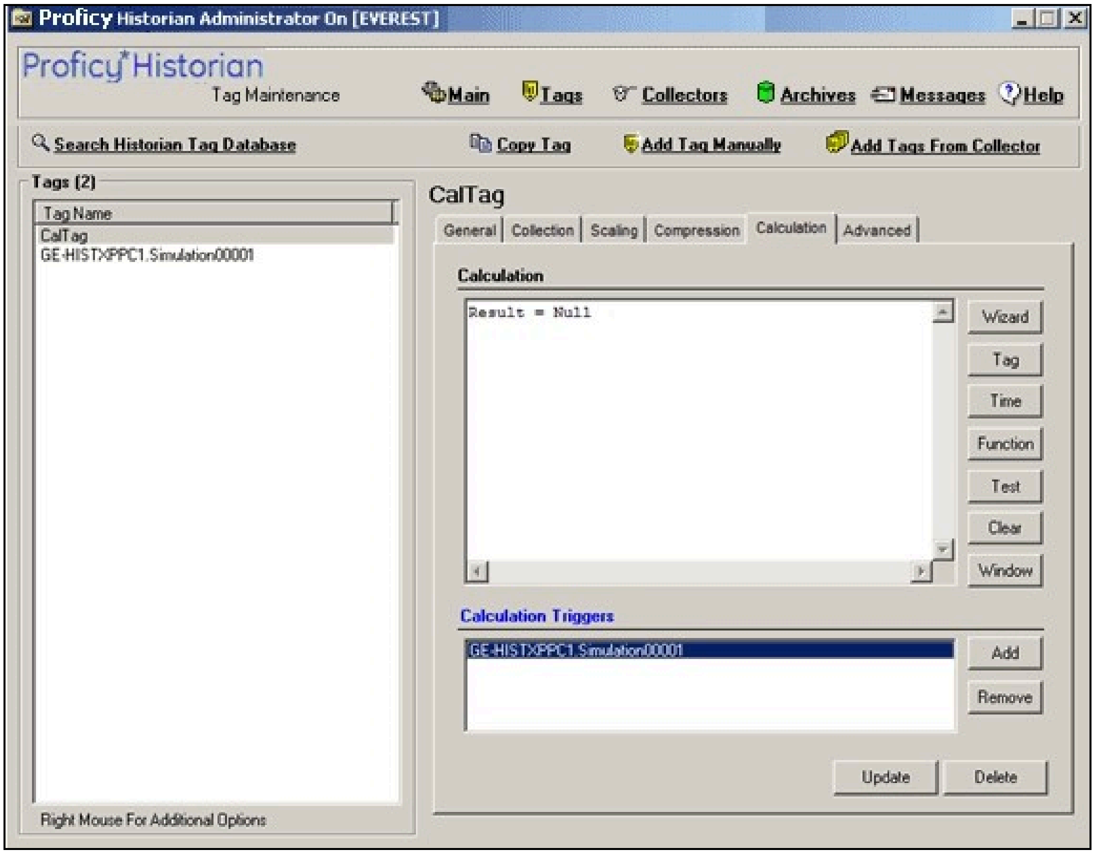
Description: *

Trigger Tag: EVEREST.Tag4

Trigger("EVEREST.Tag4")

g. Select **Insert**.

The trigger is added to the **Calculation Triggers** section.



Tags (2): CalTag, GE-HISTXPPC1.Simulation00001

CalTag — Calculation:

```
Result = Null
```

Calculation Triggers: GE-HISTXPPC1.Simulation00001

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 256)*, which helps you use any of the built-in functions *(on page 260)* orcreate your own function *(on page 258)*.
- Entering the syntax of the formula directly in the form of a VBScript code *(on page 254)*.

Before you create calculation formulas, refer to the general guidelines *(on page 252)*.

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)*.

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 271)*.

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 256)*.

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 246)* and Examples of Scheduling Unsolicited Triggers *(on page 250)*.

Create the tag that you want to use to store the calculation results. You can create the tag manually using Configuration Hub *(on page 589)*, Historian Administrator *(on page 192)* or the Web Admin console *(on page 194)*. Or, you can copy a tag *(on page 195)*.

1. To create a calculation formula using Configuration Hub:
    1. Access Configuration Hub *(on page 97)*.
    2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
       A list of tags appears.
    3. Select a Calculation tag, right-click or select more options, and then select **Calculation**.
       The calculation editor appears.
    4. Enter the VB script in the editor.
    5. To test the function, select ✓.
       A message appears, stating whether the syntax is correct.
    6. In the upper-left corner of the page, select **Save**.
       The calculation formula is created.
2. To create a calculation formula using Historian Administrator:
    1. Access Historian Administrator *(on page)*.
    2. Select **Tags**, select the tag for which you want to create a calculation formula, and then select **Calculation**.
    3. In the **Calculation Triggers** section, select **Add**.
       The **Insert Function Wizard** window appears.
    4. Under **Select Function**, in the **Type** field, select **Add A Calculation Trigger**.
    5. Under **Tag Browse Criteria**, enter the search criteria to find the tag.
       The search results appear in the **Browse Results** section.
    6. Select the tag that you want to add, and then select **Insert**.
    7. In the **Calculation** field, enter the calculation formula using the VBScript syntax.
    8. 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 254)*.

For information on a list of the available types and associated functions, refer to Types of Functions Supported *(on page 268)*. For information on each pre-defined function, refer to Built-In Functions *(on page 260)*. In addition to the built-in functions, you can create your own customized functions *(on page 269)*.

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 589)*, Historian Administrator *(on page 192)* or the Web Admin console *(on page 194)*. Or, you can copy a tag *(on page 195)*.
2. Access the advanced options of the collector, and then disable the **On-line Tag Configuration Changes** option using Historian Administrator *(on page 192)*. In Configuration Hub *(on page 589)* 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 97)*.

    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)*.

   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 269)*. You can also use any of the built-in functions *(on page 260)*.

Create the tag that you want to use to store the calculation results. You can create the tag manually using Configuration Hub *(on page 589)*, Historian Administrator *(on page 192)* or the Web Admin console *(on page 194)*. Or, you can copy a tag *(on page 195)*.

1. To create a user-defined function using Configuration Hub:

    a. Access Configuration Hub *(on page 97)*.

    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 269)*.

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 269)*.

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 271)* 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. 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: When a calculation is performed, the timestamp of the result is the time that the calculation has begun, not the time that it completed. 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.
`PreviousValue(<tag name>, Time)`	The tag value of the raw sample prior to the current time.
`PreviousQuality(<tag name>, Time)`	The quality of the tag (0 for bad quality and 100 for good quality) prior to the current time.
`PreviousGoodValue(<tag name>, Time)`	The latest good value of the raw sample prior to the current time.
`PreviousGoodQuality(<tag name>, Time)`	The good quality of the raw sample prior to the current time.
`PreviousTime(<tag name>, Time)`	The timestamp of the raw sample prior to the current time.
`PreviousGoodTime(<tag name>, Time)`	The timestamp of the latest good quality of the raw sample prior to the current time.
`NextValue(<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 1406)*.
`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 1406)*.
`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. 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. 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. 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. 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 • Previous value • Next value • Interpolated value

Type of Action	Available Functions for the Action
Insert a calculation	• Unfiltered calculation • Filtered calculation
Insert a timestamp	• Time shortcut • Previous value timestamp • Next value timestamp • Current time
Check data quality	• Current value quality • Previous value quality • Next value quality
Set data quality	• Set Quality Good • Set Quality Bad
Add data value	Value
Insert a tag name	Tagname
Insert an alarm calculation	• Previous Alarm • Next Alarm • Get Alarm Property • Set Alarm Property • Add Event • New Alarm • Update Alarm • Return to Normal
Insert a multifield operation	• GetMultiFieldValue • GetMultiFieldQuality • SetMultiFIeldValue

User-defined Functions

In addition to the built-in functions *(on page 260)*, 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 1. 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:
	`dim timetag1`
	`dim timetag2`
	`dim tag1`
	`dim tag2`
	`Dim x`
	`tag1 = "T20.di-1.F_CV"`
	`tag2 = "t20.T20.DI-1.F_CV"`
	`x = DateAdd("s", 1,CurrentTime) ' add 1 second to calc time`
	`' Get the timestamp of the newest raw sample for tag1:`
	`timetag1 = previousTime(tag1, x)`
	`' Get the timestamp of the newest raw sample for tag2:`
	`timetag2 = previousTime(tag2, x)`
	`if timetag1 > timetag2 then`
	`' If tag1 triggered me, then:`
	`result = PreviousValue(tag1,CurrentTime)`
	`else`

Tag	Description

	`' If tag2 triggered me, then:`
	`result = PreviousValue(Tag1, CurrentTime)`
	`end if`

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 285)* and Manual Recalculation *(on page 286)*.

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>
```

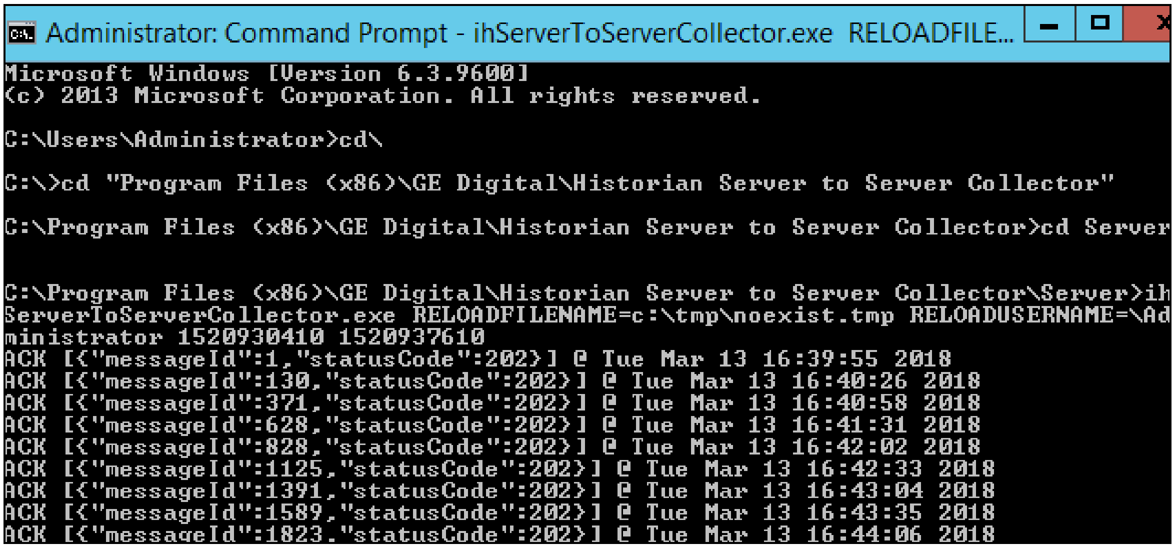
Example: `C:\Program Files\Proficy\Proficy Historian\x86\Server>ihServerToServerCollector.exe RELOADFILENAME=c:\taglist.txt RELOADUSERNAME=\Administrator 1516875659 1516875785`

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/



Administrator: Command Prompt - ihServerToServerCollector.exe RELOADFILE...

```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>cd\

C:\>cd "Program Files (x86)\GE Digital\Historian Server to Server Collector"

C:\Program Files (x86)\GE Digital\Historian Server to Server Collector>cd Server


C:\Program Files (x86)\GE Digital\Historian Server to Server Collector\Server>ih
ServerToServerCollector.exe RELOADFILENAME=c:\tmp\noexist.tmp RELOADUSERNAME=\Ad
ministrator 1520930410 1520937610
ACK [{"messageId":1,"statusCode":202}] @ Tue Mar 13 16:39:55 2018
ACK [{"messageId":130,"statusCode":202}] @ Tue Mar 13 16:40:26 2018
ACK [{"messageId":371,"statusCode":202}] @ Tue Mar 13 16:40:58 2018
ACK [{"messageId":628,"statusCode":202}] @ Tue Mar 13 16:41:31 2018
ACK [{"messageId":828,"statusCode":202}] @ Tue Mar 13 16:42:02 2018
ACK [{"messageId":1125,"statusCode":202}] @ Tue Mar 13 16:42:33 2018
ACK [{"messageId":1391,"statusCode":202}] @ Tue Mar 13 16:43:04 2018
ACK [{"messageId":1589,"statusCode":202}] @ Tue Mar 13 16:43:35 2018
ACK [{"messageId":1823,"statusCode":202}] @ Tue Mar 13 16:44:06 2018
```

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 252)*.

> **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 298)* 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 480)*.

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 556)* 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. 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. • 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. • 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

iFIX Running Mode	Recommended Configuration for the iFIX Collector	Collector Behaviour After You Add the Collector Instance
		Collector Configuration section in Historian Administrator.
iFIX is running as a service and is not secured. 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. • 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. • 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 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.

iFIX Running Mode	Recommended Configuration for the iFIX Collector	Collector Behaviour After You Add the Collector Instance
		• 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. • 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. • 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 service, an error message appears while adding a collector instance. However, the instance is configured successfully. • 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.

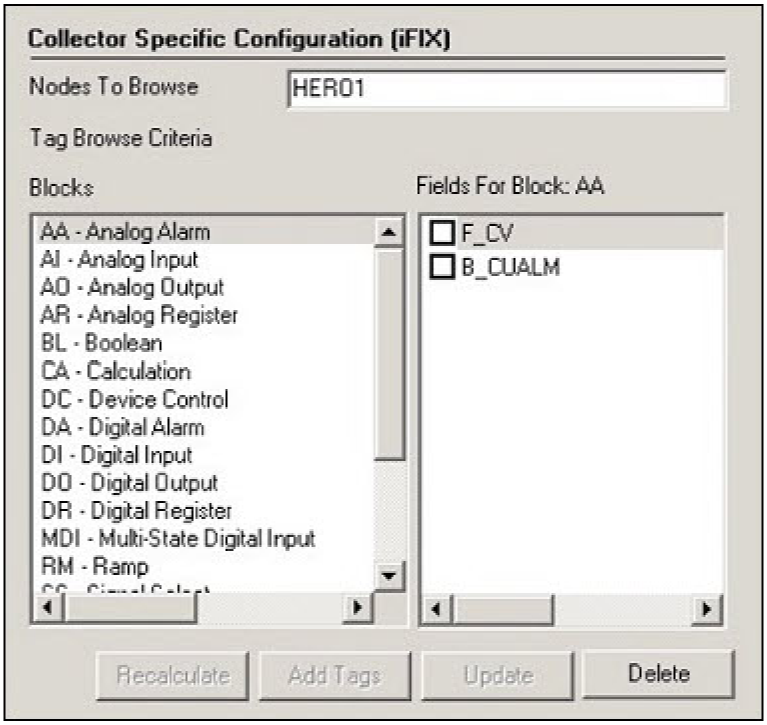
iFIX Running Mode	Recommended Configuration for the iFIX Collector	Collector Behaviour After You Add the Collector Instance
		• You must start the collector manually for the first time using the shortcut. It will then connect to the Historian server. • Once connected to server, you can start/stop it 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. • 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. • After you start iFIX, you must start the collector manually for the first time using the shortcut. It will

iFIX Running Mode	Recommended Configuration for the iFIX Collector	Collector Behaviour After You Add the Collector Instance
		then connect to the Historian server. • 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.

   Collector Specific Configuration (iFIX)

   Nodes To Browse: HERO1

   Tag Browse Criteria

   Blocks

   AA - Analog Alarm
   AI - Analog Input
   AO - Analog Output
   AR - Analog Register
   BL - Boolean
   CA - Calculation
   DC - Device Control
   DA - Digital Alarm
   DI - Digital Input
   DO - Digital Output
   DR - Digital Register
   MDI - Multi-State Digital Input
   RM - Ramp

   Fields For Block: AA

   ☐ F_CV
   ☐ B_CUALM

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.

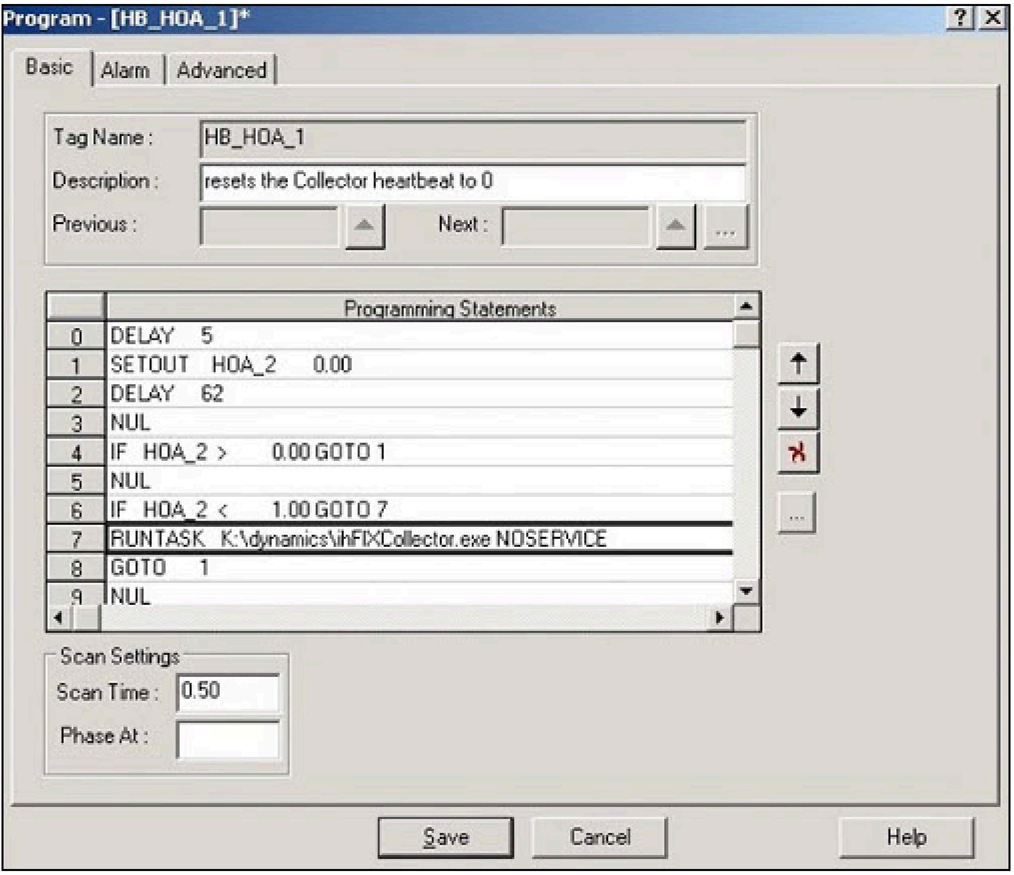
In this example, an iFIX collector runs on NODE1 and the heartbeat output address is `NODE1.HOA_2.F_CV`. We create an Analog Output Block is named HOA_2 and the Program block is named `HB_HOA_1` in a new iFIX database. There are two parts to this example: configuration in iFIX and then Historian Administrator.

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.



Tag Name : HB_HOA_1

Description : resets the Collector heartbeat to 0

Previous :

Next :

	Programming Statements
0	DELAY 5
1	SETOUT HOA_2 0.00
2	DELAY 62
3	NUL
4	IF HOA_2 > 0.00 GOTO 1
5	NUL
6	IF HOA_2 < 1.00 GOTO 7
7	RUNTASK K:\dynamics\ihFIXCollector.exe NOSERVICE
8	GOTO 1
9	NUL

Scan Settings

Scan Time : 0.50

Phase At :

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.

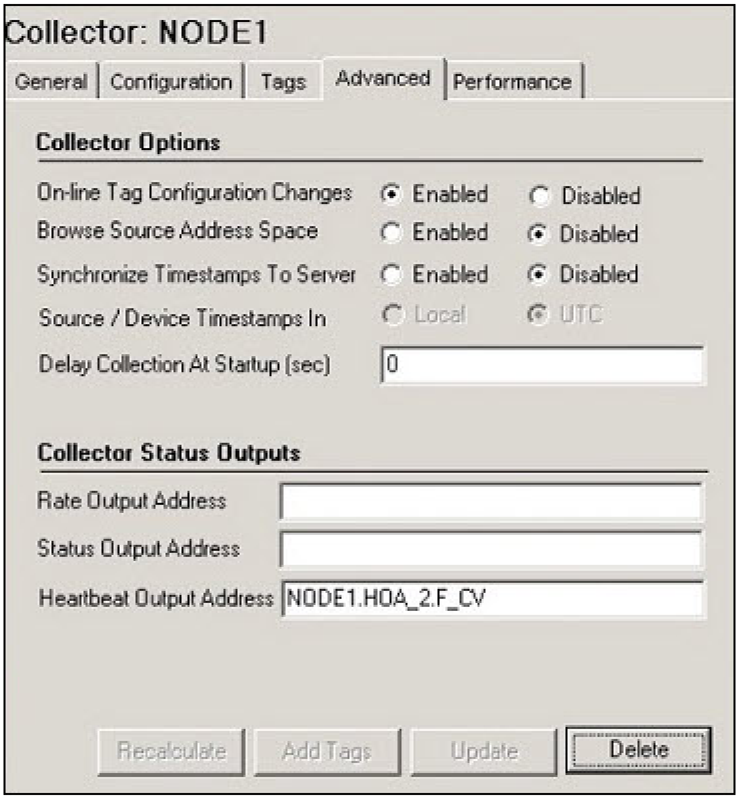
In about a minute, you should notice the iFIX collector start. The status of the Collector should change on the Collector Maintenance page of Historian Administrator.

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.



Collector: NODE1

Collector Options

On-line Tag Configuration Changes ● Enabled ○ Disabled

Browse Source Address Space ○ Enabled ● Disabled

Synchronize Timestamps To Server ○ Enabled ● Disabled

Source / Device Timestamps In ○ Local ● UTC

Delay Collection At Startup (sec) 0

Collector Status Outputs

Rate Output Address

Status Output Address

Heartbeat Output Address NODE1.HOA_2.F_CV

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.

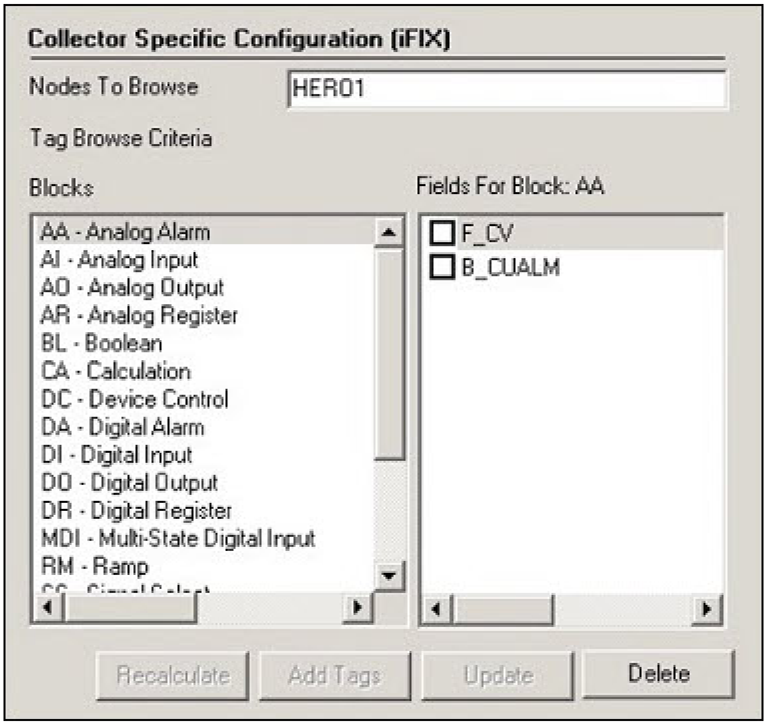
When you upgrade the iFIX collector and the iFIX Alarms and Events collectors, services are created for instances created by the installer in the previous version.

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 1358)*.

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 313)* topic describes the information that appears in this section.



Collector Specific Configuration (iFIX)

Nodes To Browse: HERO1

Tag Browse Criteria

Blocks

- AA - Analog Alarm
- AI - Analog Input
- AO - Analog Output
- AR - Analog Register
- BL - Boolean
- CA - Calculation
- DC - Device Control
- DA - Digital Alarm
- DI - Digital Input
- DO - Digital Output
- DR - Digital Register
- MDI - Multi-State Digital Input
- RM - Ramp

Fields For Block: AA

- □ F_CV
- □ B_CUALM

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. 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. **Note:** If you have modified iFIX node name, then you must also update the Nodes to Browse list before browsing tags from the iFIX collector.

Field	Description
	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 304)* for more information. **Note:** 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 305)* 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. Use an iFIX tag for the output address. Enter the address as `NODE.TAG.FIELD` (for example, `MyNode.MySIM_AO.F_CV`). 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 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

Field	Description
	Use an iFIX tag for the output address. Enter the address as `NODE.TAG.FIELD` (for example, `MyNode.MyCollector_TX.A_CV`). 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. Use an iFIX tag for the output address. Enter the address as `NODE.TAG.FIELD` (for example, `MyNode.MyCollector_AO.F_CV`) The iFIX Data Collector writes a value of 1 to this location every 60 seconds while it is running. 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.

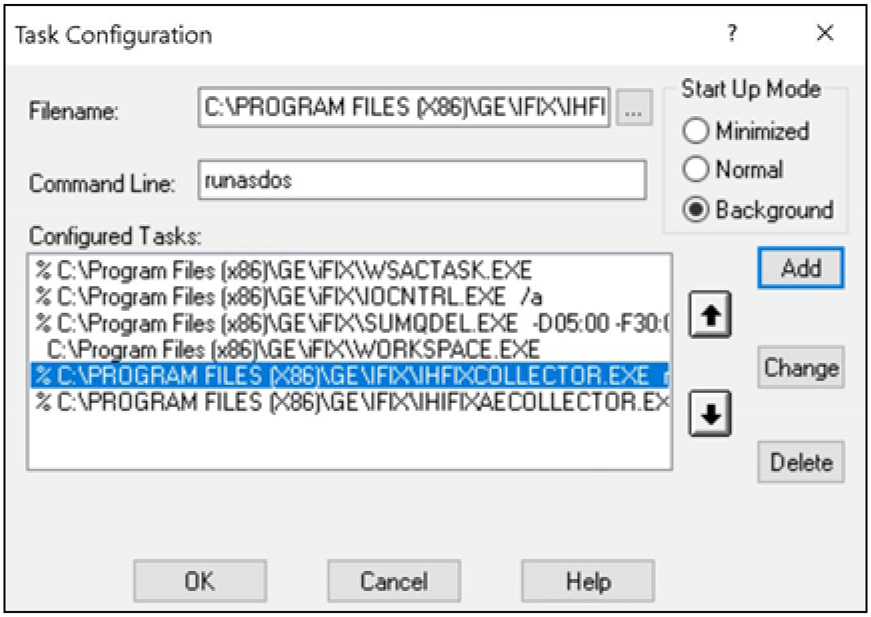
Note:

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.



Task Configuration

Filename: C:\PROGRAM FILES (X86)\GE\IFIX\IHFI

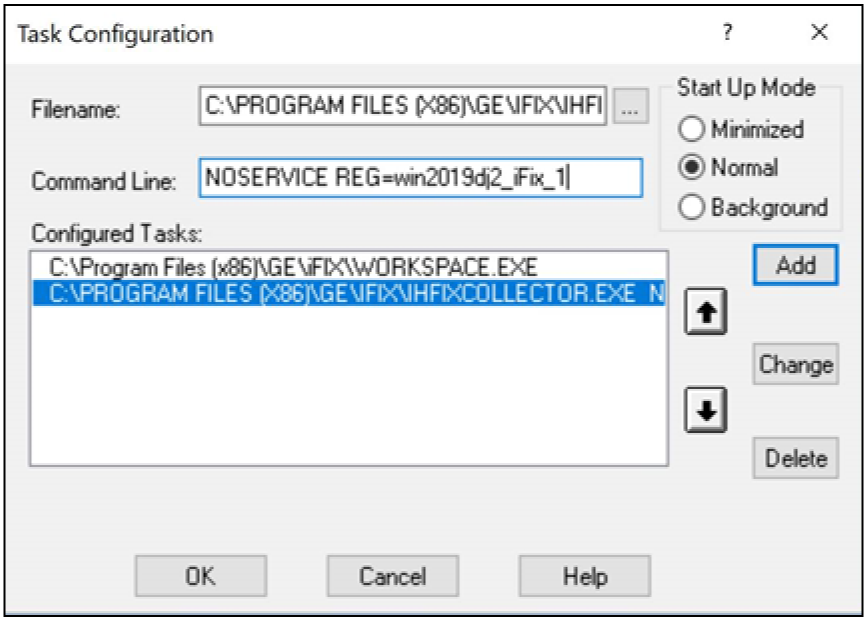
Command Line: runasdos

Start Up Mode: ○ Minimized ○ Normal ● Background

Configured Tasks:

```
% C:\Program Files (x86)\GE\iFIX\WSACTASK.EXE
% C:\Program Files (x86)\GE\iFIX\IOCNTRL.EXE  /a
% C:\Program Files (x86)\GE\iFIX\SUMQDEL.EXE  -D05:00 -F30:(
  C:\Program Files (x86)\GE\iFIX\WORKSPACE.EXE
% C:\PROGRAM FILES (X86)\GE\IFIX\IHFIXCOLLECTOR.EXE  r
% C:\PROGRAM FILES (X86)\GE\IFIX\IHIFIXAECOLLECTOR.EX
```

    - 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.



Task Configuration

Filename: C:\PROGRAM FILES (X86)\GE\IFIX\IHFI

Command Line: NOSERVICE REG=win2019dj2_iFix_1

Start Up Mode: ○ Minimized ● Normal ○ Background

Configured Tasks:

```
C:\Program Files (x86)\GE\iFIX\WORKSPACE.EXE
C:\PROGRAM FILES (X86)\GE\IFIX\IHFIXCOLLECTOR.EXE  N
```

> **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 725)* and manage them *(on page 717)* 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 300)*.

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.

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 323)*.

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)* 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 324)*.
- 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 321)*
Should I compress my data?	Planning Compression *(on page 321)*
In what order should I migrate my data?	Recommended Migration Order *(on page 321)*
Do I need any security rights to migrate my data?	Implementing Security During Migration *(on page 323)*
What if my data is being migrated into an already active Historian system?	Planning Migrations with Online Systems *(on page 322)*
Should I account for Daylight Savings Time or Time Zone differences?	Planning Daylight Savings Time *(on page 323)*
How long will my migration take?	Estimating Migration Time *(on page 324)*

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 324)*
Is there anything I should do after migrating?	After Migrating Your Data *(on page 325)*

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 322)*.

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 337)*.

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)*.

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 319)*?
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 319)*?
3. Migrate all collection groups. How *(on page 336)*?

   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 327)*?
6. Migrate Advanced Historical data or, How *(on page 333)*?
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 320)* 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 325)*?
2. Configure Classic Historian Migration Options. How *(on page 329)*?
3. Migrate existing groups. How *(on page 327)*?
4. Migrate your Classic Historian Data. How *(on page 327)*?

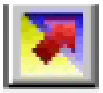
Migrating Classic Historian Data to Your Historian Database

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 322)*.

> **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

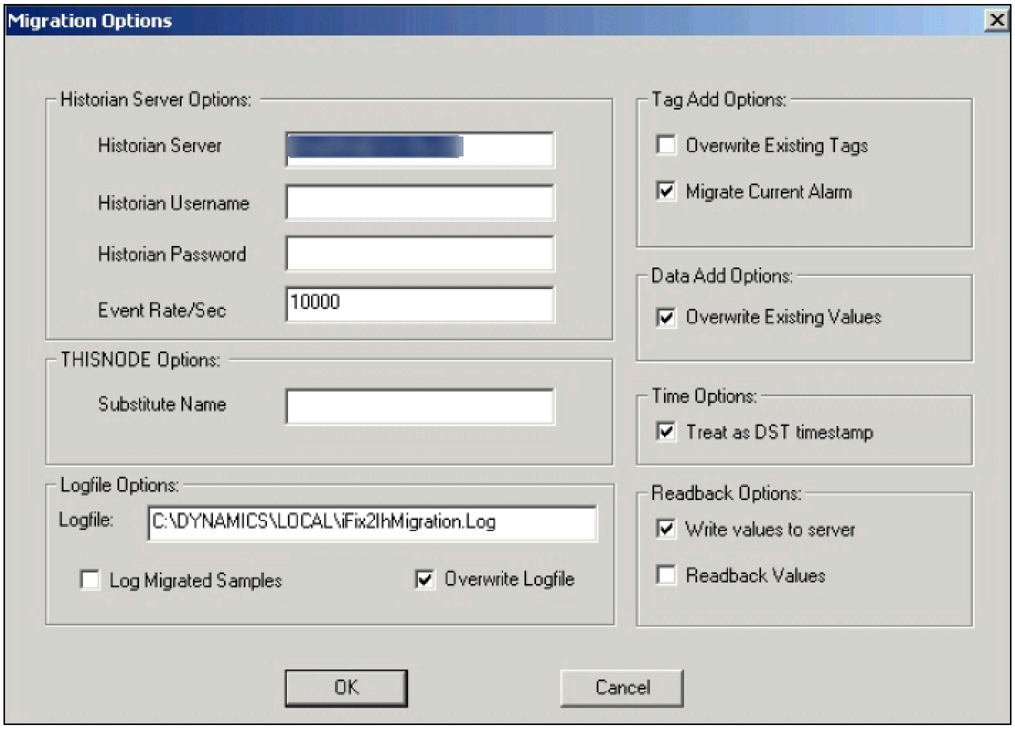
If you compare plots of average, minimum, and maximum data in Classic Historian with similar plots in Historian, you may notice the following differences:

- 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 1. Classic Historian Migration Options window

   Historian Server Options:

   Historian Server [redacted]

   Historian Username

   Historian Password

   Event Rate/Sec 10000

   THISNODE Options:

   Substitute Name

   Logfile Options:

   Logfile: C:\DYNAMICS\LOCAL\iFix2IhMigration.Log

   □ Log Migrated Samples ☑ Overwrite Logfile

   Tag Add Options:

   □ Overwrite Existing Tags

   ☑ Migrate Current Alarm

   Data Add Options:

   ☑ Overwrite Existing Values

   Time Options:

   ☑ Treat as DST timestamp

   Readback Options:

   ☑ Write values to server

   □ Readback Values

   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 329)*
   - THISNODE Options *(on page 330)*
   - Tag Add Options *(on page 330)*
   - Logfile Options *(on page 331)*
   - Readback Options *(on page 332)*

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)* 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 322)*.

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:
	If you wish to migrate alarms and events data from iFIX to Historian, use the iFIX Alarms and Events collector Migration Tool.
	For example:
	The Classic Historian Tag:
	`FIX.TAG1.F_CV`
	will convert to the Historian Tags:
	`FIX.TAG1.F_CV`,`FIX.TAG1.B_CUALM`
	Note:
	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 **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 **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 323)*.

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: 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. **Important:** Selecting the **Readback Values** option puts additional load on the archiver and can slow the migration process. 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. 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 320)* for more information and establish your migration options.

1. Add the Historian Toolbar to the iFIX v2.5 or later WorkSpace. How *(on page 325)*?
2. Explore the Advanced Migration Utility Options. How *(on page 333)*?
3. Use the Classic Historian Migration Utility to migrate historical tag groups. How *(on page 336)*?

> **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 333)*?
5. Migrate Advanced Historian Remote Data. How *(on page 337)*?

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 334)* Displays information for the Advanced Historian machine you are migrating your historical data from.
- Historian Information *(on page 334)* Displays the information for the Historian Server that you want to migrate your historical data to.
- Migration Options *(on page 334)* Allows you to select all or specific tag masks and set the time range for migration.
- Migration Status *(on page 335)* Displays the current status of the migration.
- Migration History *(on page 335)* 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)*.

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. 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 329)* 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 333)* 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 324)* 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 322)* 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 334)*.
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 339)* 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 333)* 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 324)* 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 322)* 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 334)*

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 342)* options:

   a. From the **Options** menu, choose **Alarm Source**. The **Alarm Source Options** window appears.

   b. Configure the ODBC Login *(on page 342)* 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 342)* .

   d. Configure Severity Mapping *(on page 342)*.

3. Select **OK**.
4. Set up the Alarm Destination *(on page 344)* options:
   a. From the **Options** menu, choose **Alarm Destination**. The **Alarm Destination Options** window appears.
   b. Configure the Historian Server Options *(on page 344)*.
   c. Configure the Logfile *(on page 344)*.
   d. Configure the Time Options *(on page 344)*.
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: • Access • Oracle • SQL Server • Sybase **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. **Note:** 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. Refer to Manage Daylight Savings Time *(on page 323)* 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**.

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.

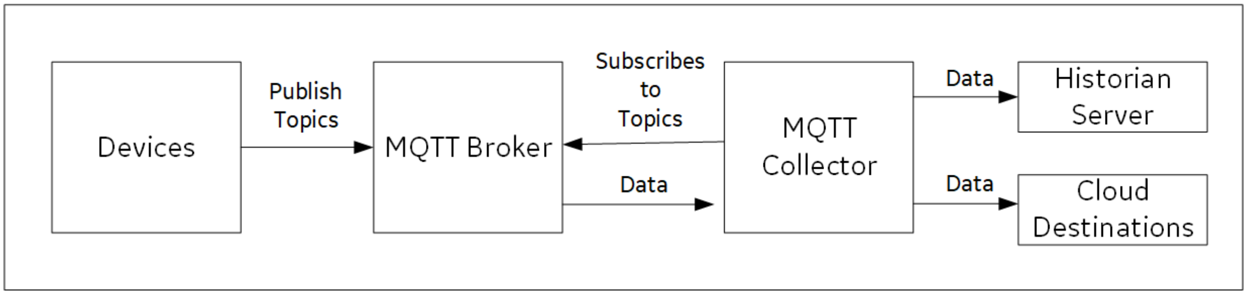
Features:

- 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. Deploy Proficy Historian for AWS. *(on page 43)*
2. Install collectors *(on page 90)*. You can install them on-premises or on a VPC (which can be different from the one on which Proficy Historian for AWS is deployed).
3. Enable TLS encryption for Collectors Connecting to Cloud Historian *(on page 150)*.
4. 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.

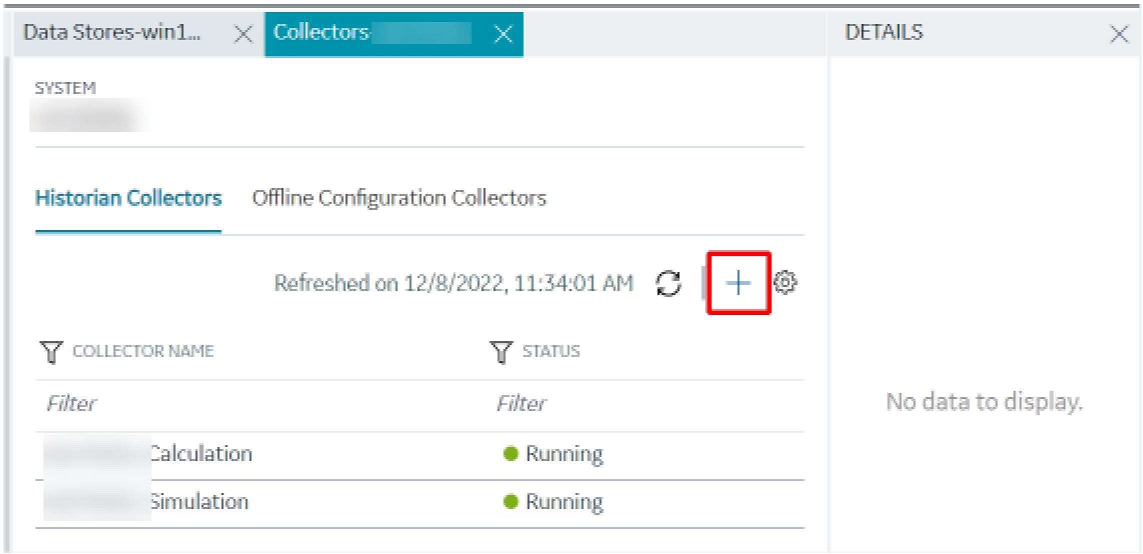
5. If you want to use username/password-based authentication or certificate-based authentication to connect the MQTT broker and the MQTT collector, configure the authentication in the MQTT broker.
6. If you want to use certificate-based authentication, ensure that the following files are available on 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 346)*.

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 1335)*, which does not require you to install Web-based Clients.

1. Access Configuration Hub *(on page 97)*.
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 +.



SYSTEM

[redacted]

Refreshed on 12/8/2022, 11:34:01 AM

COLLECTOR NAME	STATUS
Filter	*Filter*
[redacted] Calculation	● Running
[redacted] Simulation	● Running

No data to display.

   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.

Field	Description
	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` where: ◦ `namespace` is the Sparkplug version. Enter `spBv1.0`. ◦ `group_id` is the ID of the group of nodes from which you want to collect data. ◦ `message_type` is the message type from which you want to collect data. The collector processes data only from NDATA and DDATA message types. ◦ `edge_node_id` is used to identify the MQTT EoN node within the infrastructure. ◦ `device_id` a device attached to the MQTT EoN node either physically or logically. 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.
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.

Field	Description
REQUESTED QUALITY OF SERVICE (QOS) LEVEL	Select the quality of service that you want to use while collecting data from an MQTT broker. ◦ **QoS 0**: Indicates that the message is delivered at most once or it is not delivered at all. ◦ **QoS 1**: Indicates that the message is always delivered at least once. ◦ **QoS 2**: Indicates that the message is delivered once. 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: ◦ **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. ◦ **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. 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. 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:

Field	Description
	◦ **JSON**: Select this option if you want to use the KairosDB format. ◦ **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-premise 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 179)*.
      - **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 698)*.
      - **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 151)*.
        - AWS cloud. For more information, refer to AWS Cloud *(on page 158)*.
        - Google cloud. For more information, refer to Google Cloud *(on page 172)*.

   b. If you need to send data to an on-premise/Cloud Historian server, select **Historian Server**.

      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.

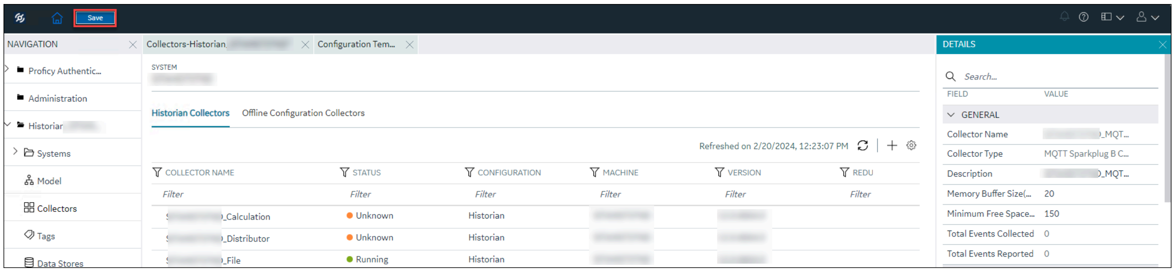
    INSTANCE CONFIGURATION

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. 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` where:

Field	Description
	◦ `namespace` is the Sparkplug version. Enter `spBv1.0`. ◦ `group_id` is the ID of the group of nodes from which you want to collect data. ◦ `message_type` is the message type from which you want to collect data. The collector processes data only from NDATA and DDATA message types. ◦ `edge_node_id` is used to identify the MQTT EoN node within the infrastructure. ◦ `device_id` a device attached to the MQTT EoN node either physically or logically. 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.

Field	Description
	◦ **QoS 0**: Indicates that the message is delivered at most once or it is not delivered at all. ◦ **QoS 1**: Indicates that the message is always delivered at least once. ◦ **QoS 2**: Indicates that the message is delivered once. 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: ◦ **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. ◦ **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. 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. 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: ◦ **JSON**: Select this option if you want to use the KairosDB format. ◦ **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 669)*.
16. In the upper-left corner of the page, select **Save**.



SYSTEM [redacted]

Refreshed on 2/20/2024, 12:23:07 PM

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSION	REDU
Filter	Filter	Filter	Filter	Filter	Filter
[redacted]_Calculation	● Unknown	Historian	[redacted]	[redacted]	
[redacted]_Distributor	● Unknown	Historian	[redacted]	[redacted]	
[redacted]_File	● Running	Historian	[redacted]	[redacted]	

FIELD	VALUE
GENERAL	
Collector Name	[redacted]_MQT...
Collector Type	MQTT Sparkplug B C...
Description	[redacted]_MQT...
Memory Buffer Size(...	20
Minimum Free Space...	150
Total Events Collected	0
Total Events Reported	0

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,

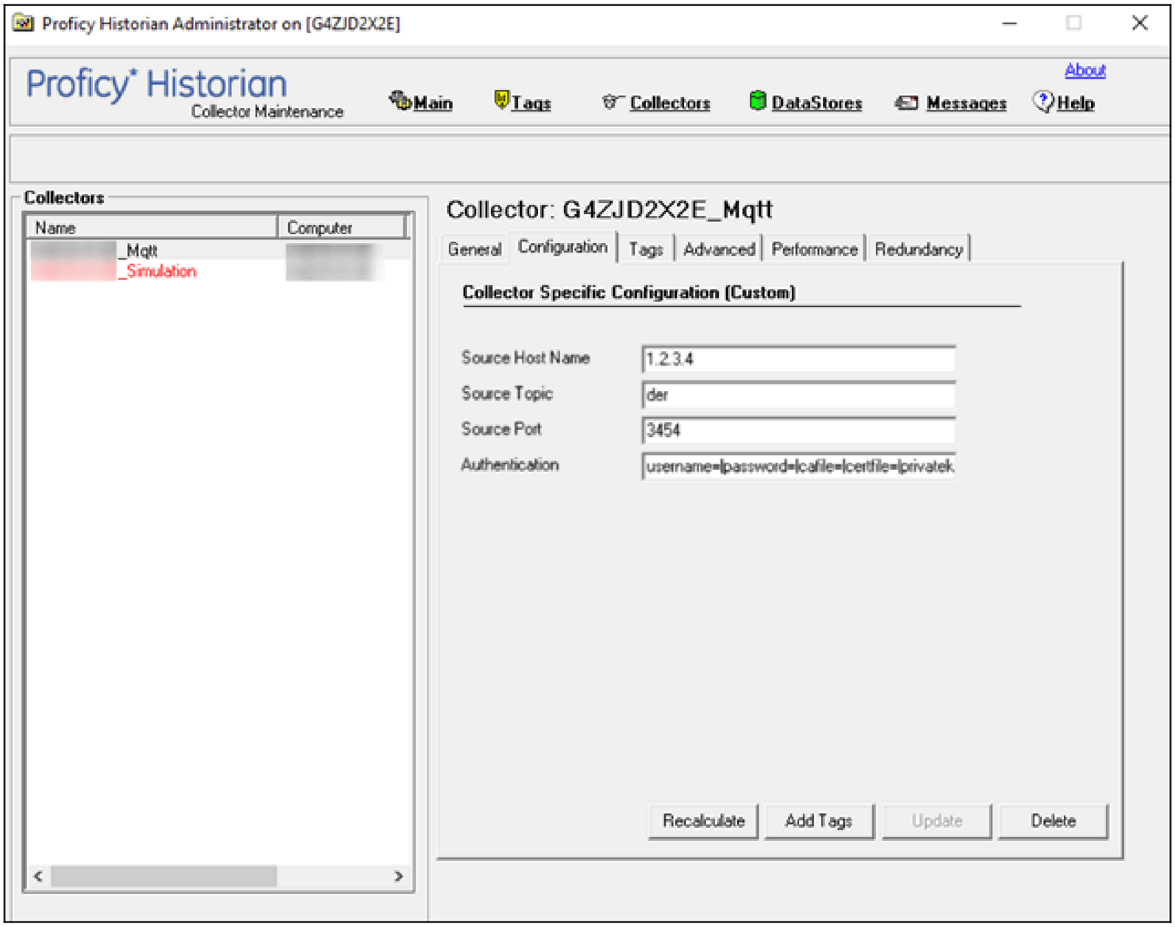
- If you have selected **Historian Configuration**, specify the tags using Configuration Hub *(on page 384)*.

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.



Collector: G4ZJD2X2E_Mqtt

Collector Specific Configuration (Custom)

Source Host Name: 1.2.3.4

Source Topic: der

Source Port: 3454

Authentication: username=|password=|cafile=|certfile=|privatek

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. 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` 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>` **Note:** 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.

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-premise 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.

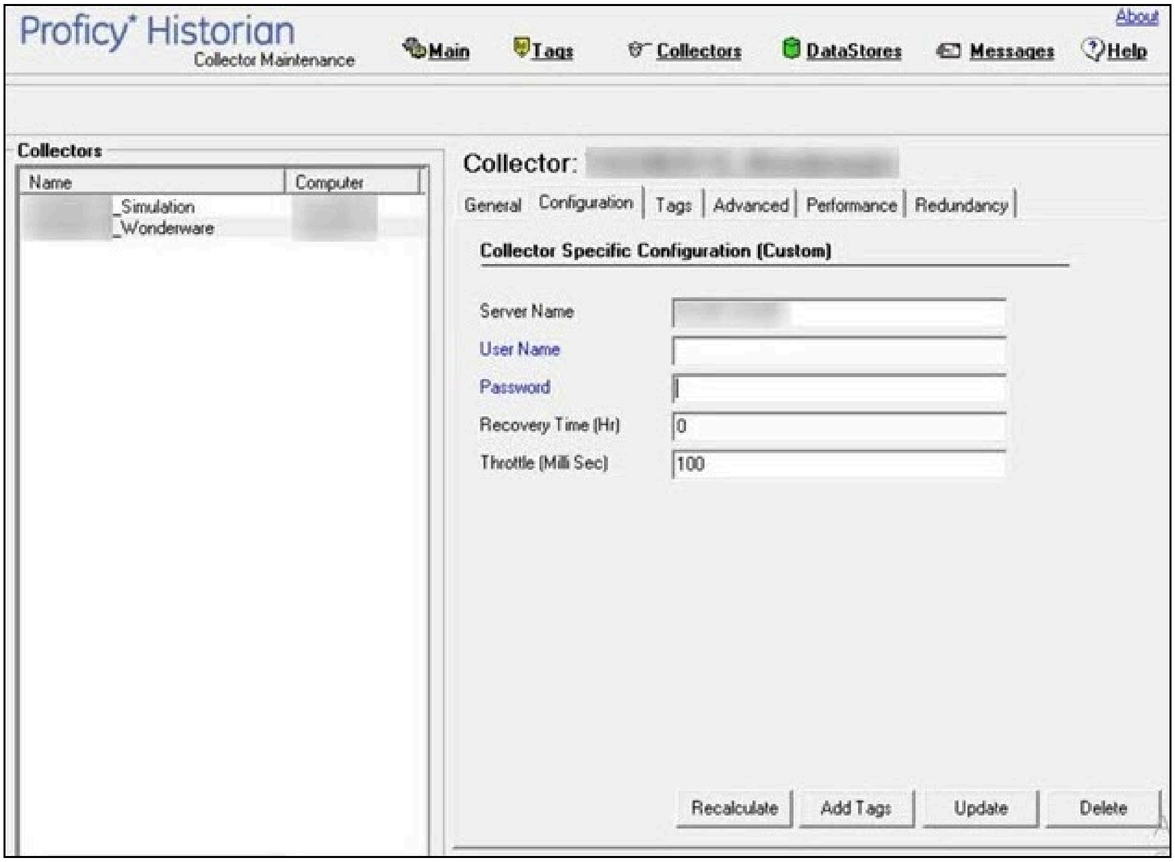
Supported Data Attributes:

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
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.



Collectors

Name	Computer
[redacted]_Simulation [redacted]_Wonderware	[redacted]

Collector: [redacted]

Collector Specific Configuration (Custom)

Server Name [redacted]
User Name
Password
Recovery Time (Hr) 0
Throttle (Milli Sec) 100

Configuration

Add and Configure an ODBC Collector Using Configuration Hub

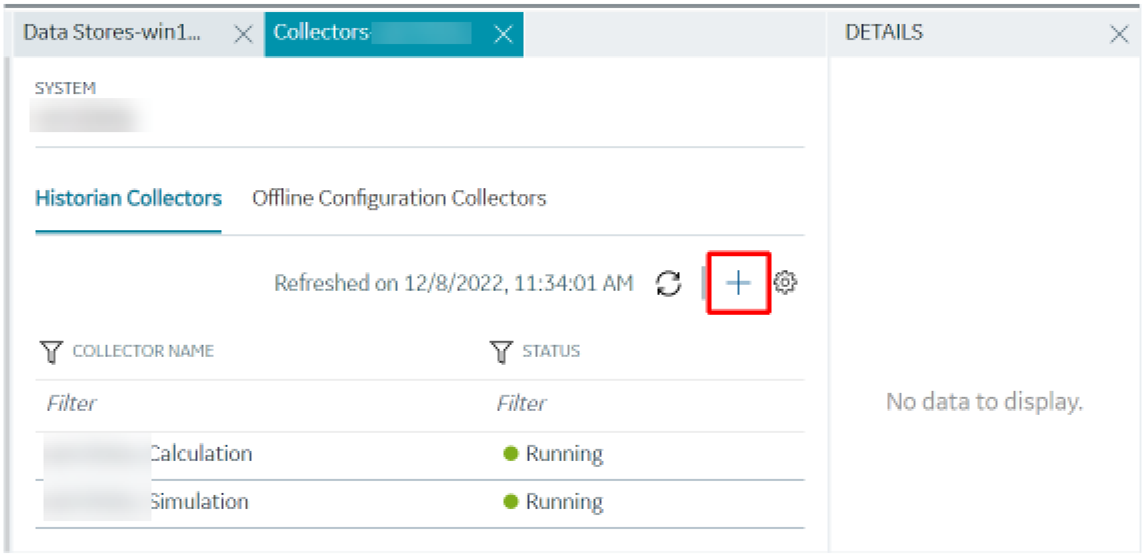
1. Deploy Proficy Historian for AWS. *(on page 43)*
2. Install collectors *(on page 90)*. You can install them on-premises or on a VPC (which can be different from the one on which Proficy Historian for AWS is deployed).
3. Enable TLS encryption for Collectors Connecting to Cloud Historian *(on page 150)*.

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 359)*.

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 1335)*, which does not require you to install Web-based Clients.

1. Access Configuration Hub *(on page 97)*.
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 +.

COLLECTOR NAME	STATUS
Filter	*Filter*
[redacted] Calculation	● Running
[redacted] Simulation	● Running

   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 179)*.
      - **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 698)*.
      - **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 151)*.
        - AWS cloud. For more information, refer to AWS Cloud *(on page 158)*.
        - Google cloud. For more information, refer to Google Cloud *(on page 172)*.

   b. If you need to send data to an on-premise/Cloud Historian server, select **Historian Server**.
      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

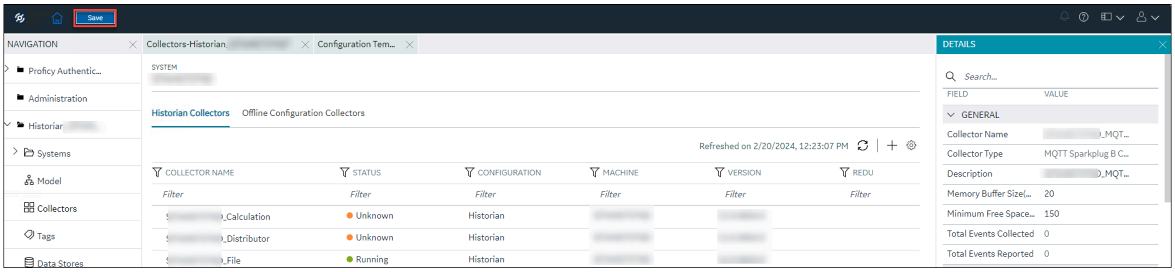
You can also configure the collector to start automatically when you start the computer.

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. Continuous data collection is resumed only after the previous data has been recovered. 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. **Note:** If this field is blank, enter the required minimum value of 100 milliseconds.

15. As needed, enter values in the other sections common to all collectors *(on page 669)*.
16. In the upper-left corner of the page, select **Save**.



SYSTEM [redacted]

Historian Collectors | Offline Configuration Collectors

Refreshed on 2/20/2024, 12:23:07 PM

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSION	REDU
Filter	Filter	Filter	Filter	Filter	Filter
[redacted]_Calculation	● Unknown	Historian	[redacted]	[redacted]	
[redacted]_Distributor	● Unknown	Historian	[redacted]	[redacted]	
[redacted]_File	● Running	Historian	[redacted]	[redacted]	

FIELD	VALUE
GENERAL	
Collector Name	[redacted]_MQT...
Collector Type	MQTT Sparkplug B C...
Description	[redacted]_MQT...
Memory Buffer Size(...	20
Minimum Free Space...	150
Total Events Collected	0
Total Events Reported	0

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 384)*.

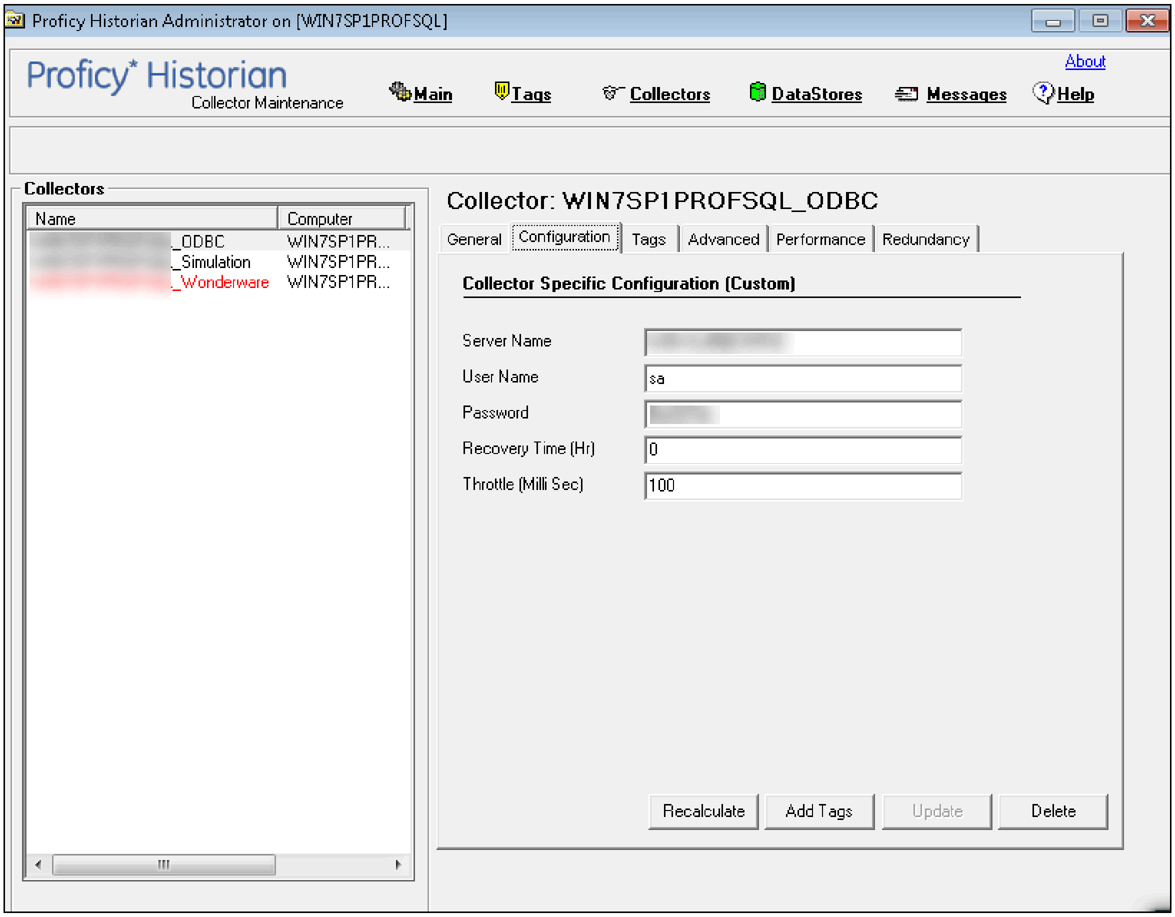
Configure the ODBC Collector Using Historian Administrator

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.



Name	Computer
[redacted]_ODBC	WIN7SP1PR...
[redacted]_Simulation	WIN7SP1PR...
[redacted]_Wonderware	WIN7SP1PR...

Server Name [redacted]
User Name sa
Password [redacted]
Recovery Time (Hr) 0
Throttle (Milli Sec) 100

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.

Field	Description
	This time is calculated as the duration between the current time and the last known write time. Continuous data collection is resumed only after the previous data has been recovered. 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. **Note:** 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 367)*.
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 365)*.
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 367)*.
4. Restart the collector.

If this workaround is not successful, restart the collector instance.

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

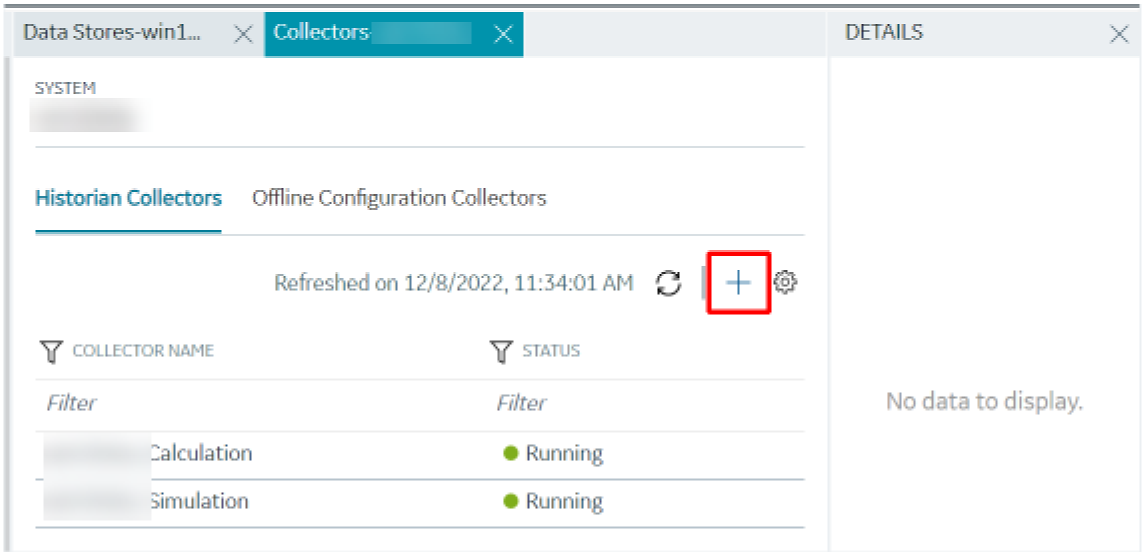
Add and Configure an OPC Classic Data Access Collector

1. Deploy Proficy Historian for AWS. *(on page 43)*
2. Install collectors *(on page 90)*. You can install them on-premises or on a VPC (which can be different from the one on which Proficy Historian for AWS is deployed).
3. Enable TLS encryption for Collectors Connecting to Cloud Historian *(on page 150)*.

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 374)*.

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 1335)*, which does not require you to install Web-based Clients.

1. Access Configuration Hub *(on page 97)*.
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 +.

COLLECTOR NAME	STATUS
Filter	*Filter*
[redacted]Calculation	● Running
[redacted]Simulation	● Running

   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.
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 179)*.
      - **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 698)*.
      - **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 151)*.
        - AWS cloud. For more information, refer to AWS Cloud *(on page 158)*.
        - Google cloud. For more information, refer to Google Cloud *(on page 172)*.

   b. If you need to send data to an on-premise/Cloud Historian server, select **Historian Server**.
      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.

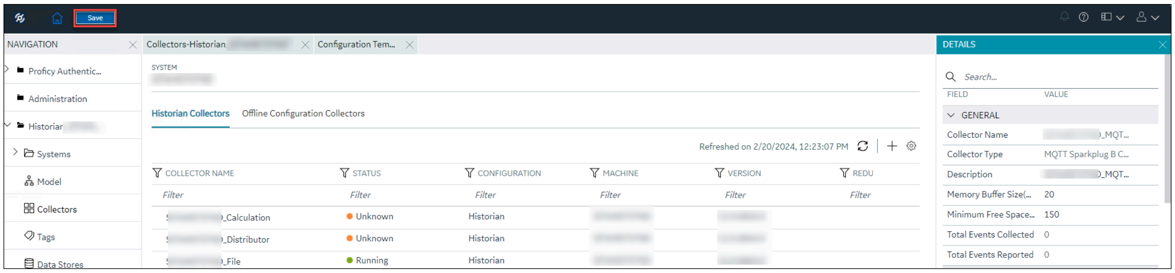
13. Select **Add**.
    The collector instance is added. The fields specific to the collector section appear in the **DETAILS** section.
14. Under **COLLECTOR SPECIFIC CONFIGURATION**, configure values as described in the following table.

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.

Field	Description
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. 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: ◦ USGB014 -and- ◦ 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. ◦ **Multithreaded**: Select this option for better performance. We recommend that you configure your collector to use the default multi-threading model. ◦ **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. 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.

Field	Description
	◦ **Made On-Line**: Select this option to process any configuration changes immediately (after 30 seconds) after you select the **Update** button. **Note:** ▪ Some OPC servers cannot handle processing configuration changes online. If you experience any instability with changes made online, use the next option. ◦ **Made After Collector Restart**: Select this option to hold all configuration changes until you manually restart the collector.

15. As needed, enter values in the other sections common to all collectors *(on page 669)*.
16. In the upper-left corner of the page, select **Save**.



COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSION	REDU
Filter	Filter	Filter	Filter	Filter	Filter
[redacted]_Calculation	● Unknown	Historian	[redacted]	[redacted]	
[redacted]_Distributor	● Unknown	Historian	[redacted]	[redacted]	
[redacted]_File	● Running	Historian	[redacted]	[redacted]	

FIELD	VALUE
GENERAL	
Collector Name	[redacted]_MQT...
Collector Type	MQTT Sparkplug B C...
Description	[redacted]_MQT...
Memory Buffer Size(...	20
Minimum Free Space...	150
Total Events Collected	0
Total Events Reported	0

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,

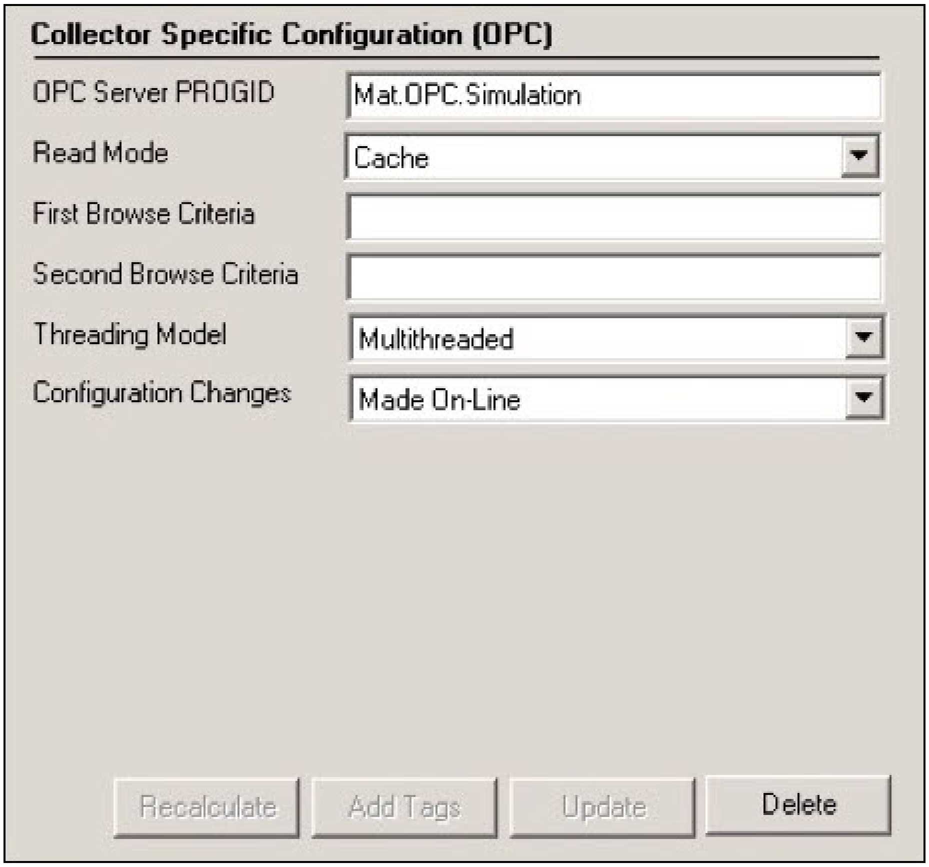
- If you have selected **Historian Configuration**, specify the tags for data collection *(on page 384)*.

Configure the OPC Classic DA Collector Using Historian Administrator

1. Access Historian Administrator *(on page)*.
2. Select **Collectors**, and then select the OPC Classic DA collector instance that you want to configure.

3. Select **Configuration**.

   The **Configuration** section appears.



Collector Specific Configuration (OPC)

OPC Server PROGID: Mat.OPC.Simulation

Read Mode: Cache

First Browse Criteria:

Second Browse Criteria:

Threading Model: Multithreaded

Configuration Changes: Made On-Line

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. 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 -and- ◦ 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. ◦ **Multithreaded**: Select this option for better performance. We recommend that you configure your collector to use the default multi-threading model. ◦ **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. 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. ◦ **Made On-Line**: Select this option to process any configuration changes immediately (after 30 seconds) after you select the **Update** button. **Note:** ▪ Some OPC servers cannot handle processing configuration changes online. If you experience any instability with changes made online, use the next option. ◦ **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. 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

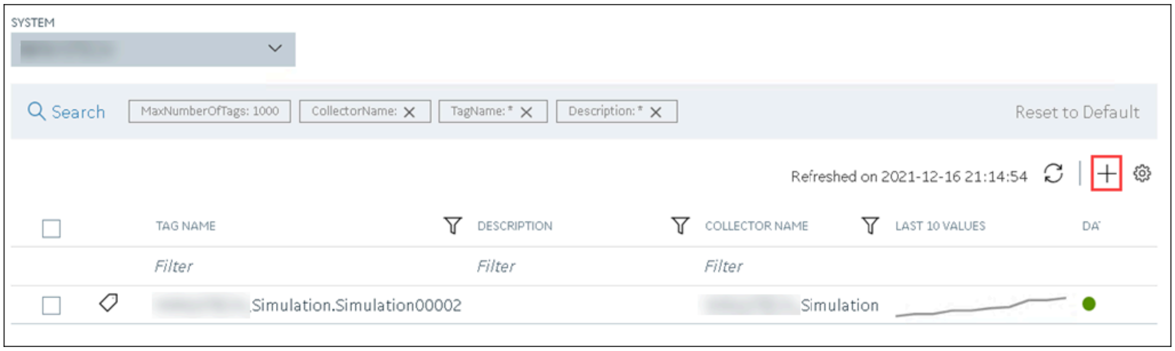
Add Tags for the Data Store Using Configuration Hub

- Add the collector instance *(on page 556)* 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 575)*.

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 589)*.

1. Access Configuration Hub *(on page 97)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
3. Select +.



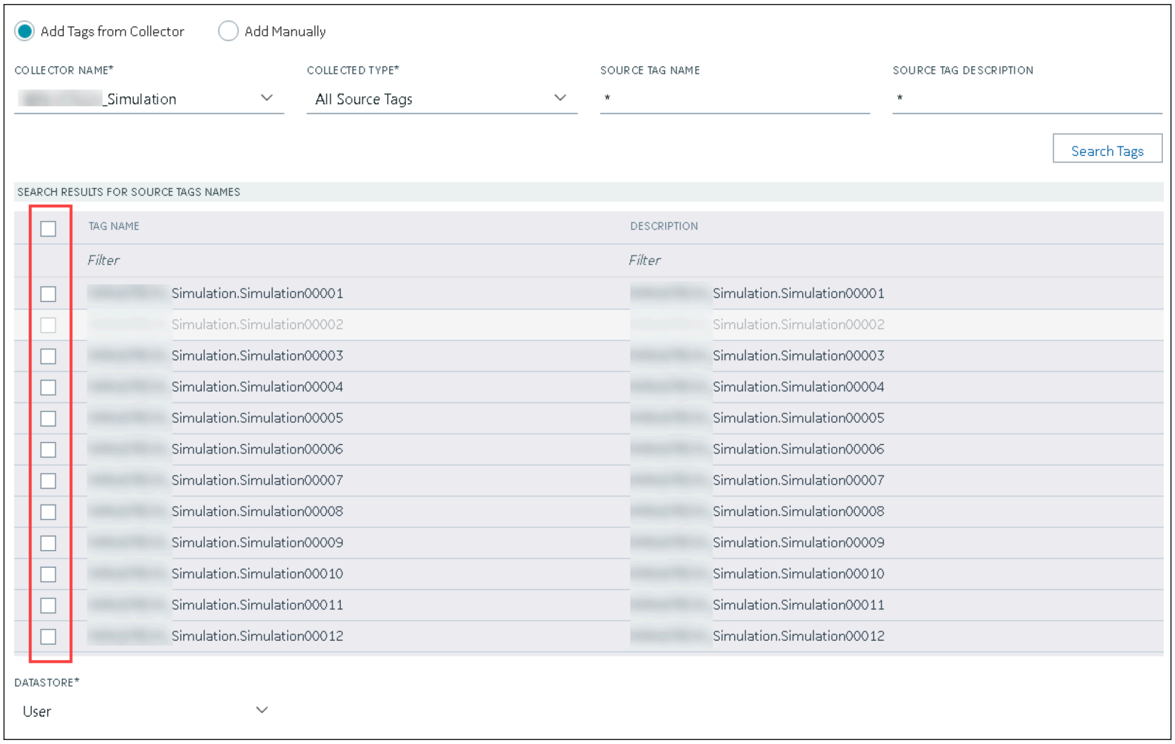
The **Add Tag-<system name>** page appears. The **Add Tags from Collector** option is selected by default.

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.



● Add Tags from Collector ○ Add Manually

COLLECTOR NAME*	COLLECTED TYPE*	SOURCE TAG NAME	SOURCE TAG DESCRIPTION
[redacted]_Simulation	All Source Tags	*	*

SEARCH RESULTS FOR SOURCE TAGS NAMES

□	TAG NAME	DESCRIPTION
	Filter	*Filter*
□	[redacted]Simulation.Simulation00001	[redacted]Simulation.Simulation00001
□	[redacted]Simulation.Simulation00002	[redacted]Simulation.Simulation00002
□	[redacted]Simulation.Simulation00003	[redacted]Simulation.Simulation00003
□	[redacted]Simulation.Simulation00004	[redacted]Simulation.Simulation00004
□	[redacted]Simulation.Simulation00005	[redacted]Simulation.Simulation00005
□	[redacted]Simulation.Simulation00006	[redacted]Simulation.Simulation00006
□	[redacted]Simulation.Simulation00007	[redacted]Simulation.Simulation00007
□	[redacted]Simulation.Simulation00008	[redacted]Simulation.Simulation00008
□	[redacted]Simulation.Simulation00009	[redacted]Simulation.Simulation00009
□	[redacted]Simulation.Simulation00010	[redacted]Simulation.Simulation00010
□	[redacted]Simulation.Simulation00011	[redacted]Simulation.Simulation00011
□	[redacted]Simulation.Simulation00012	[redacted]Simulation.Simulation00012

DATASTORE*

User

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.

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

The OPC Data Type	Recommended Data Type in Historian
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

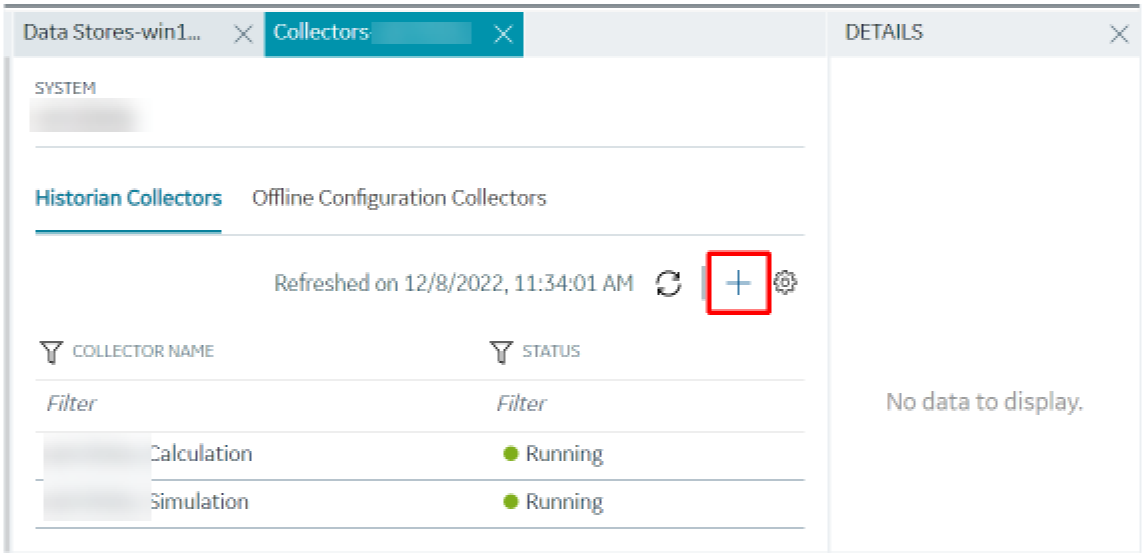
Add and Configure an OPC Classic HDA Collector

1. Deploy Proficy Historian for AWS. *(on page 43)*
2. Install collectors *(on page 90)*. You can install them on-premises or on a VPC (which can be different from the one on which Proficy Historian for AWS is deployed).
3. Enable TLS encryption for Collectors Connecting to Cloud Historian *(on page 150)*.

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 397)*.

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 1335)*, which does not require you to install Web-based Clients.

1. Access Configuration Hub *(on page 97)*.
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 ＋.



COLLECTOR NAME	STATUS
Filter	*Filter*
[redacted]Calculation	● Running
[redacted]Simulation	● Running

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-premise 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 179)*.
      - **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 698)*.
      - **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 151)*.
         - AWS cloud. For more information, refer to AWS Cloud *(on page 158)*.
         - Google cloud. For more information, refer to Google Cloud *(on page 172)*.

   b. If you need to send data to an on-premise/Cloud Historian server, select **Historian Server**.

      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

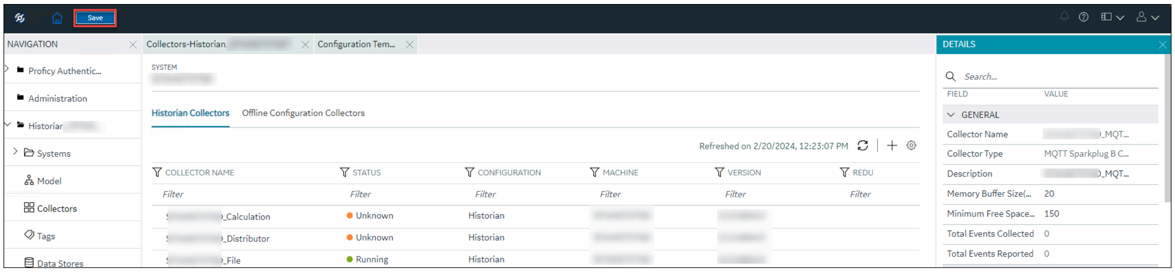
You can also configure the collector to start automatically when you start the computer.

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. Continuous data collection is resumed only after the previous data has been recovered. You can enter a value between 1 and 150.

15. As needed, enter values in the other sections common to all collectors *(on page 669)*.
16. In the upper-left corner of the page, select **Save**.



COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSION	REDU
Filter	Filter	Filter	Filter	Filter	Filter
[redacted]_Calculation	Unknown	Historian	[redacted]	[redacted]	
[redacted]_Distributor	Unknown	Historian	[redacted]	[redacted]	
[redacted]_File	Running	Historian	[redacted]	[redacted]	

FIELD	VALUE
GENERAL	
Collector Name	[redacted]_MQT...
Collector Type	MQTT Sparkplug B C...
Description	[redacted]_MQT...
Memory Buffer Size(...	20
Minimum Free Space...	150
Total Events Collected	0
Total Events Reported	0

    The changes to the collector instance are saved.
17. If needed, restart the collector.

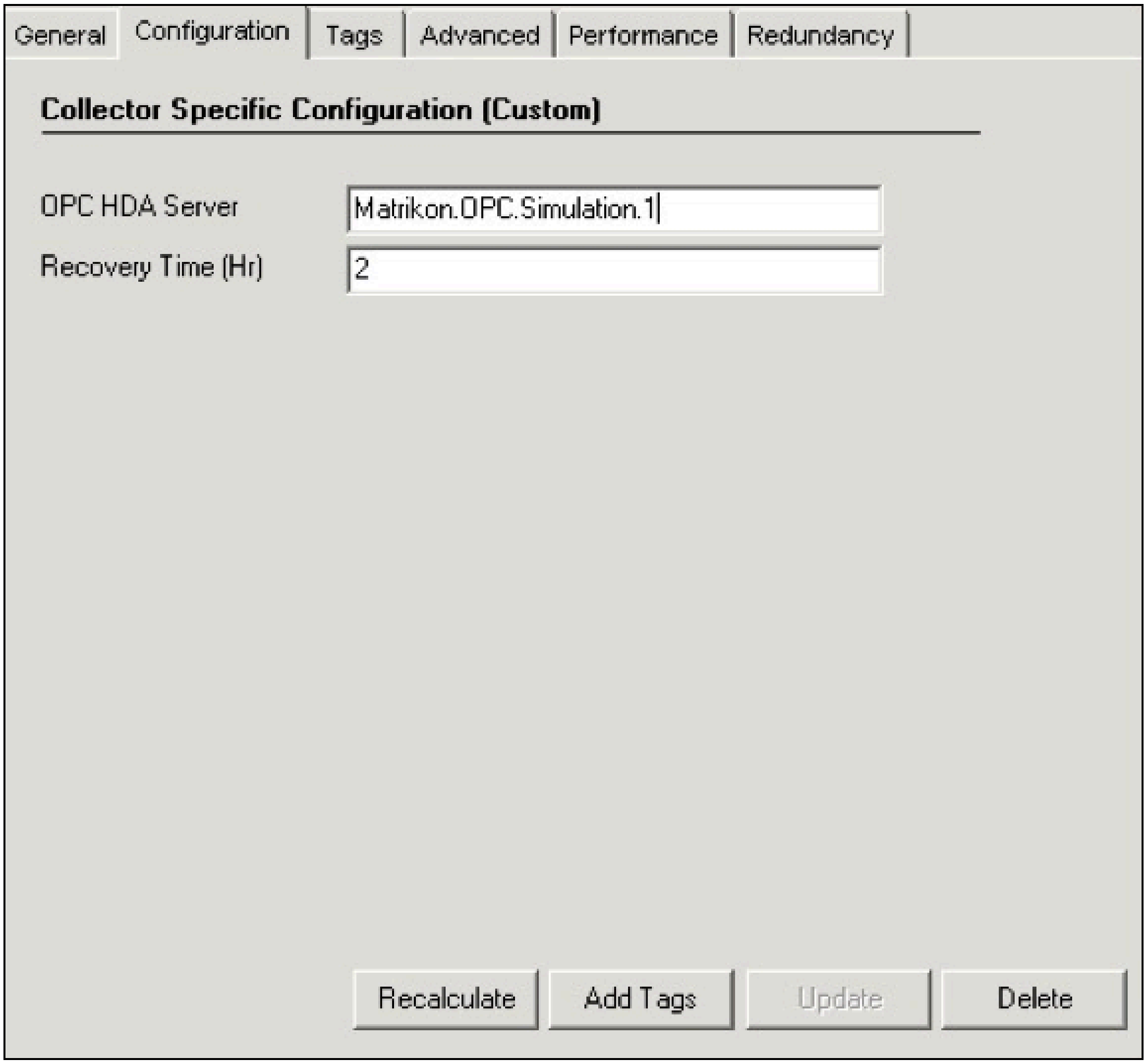
Specify the tags whose data you want to collect using the collector. In the **CHOOSE CONFIGURATION** field,

- If you have selected **Historian Configuration**, specify the tags using Configuration Hub *(on page 384)*.

Configure the OPC Classic HDA Collector Using Historian Administrator

1. Access Historian Administrator *(on page)*.
2. Select **Collectors**, and then select the OPC Classic HDA collector instance that you want to configure.
3. Select **Configuration**.

   The **Configuration** section appears.



Collector Specific Configuration (Custom)

OPC HDA Server: Matrikon.OPC.Simulation.1

Recovery Time (Hr): 2

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.

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 OPC server is re-established. This time is calculated as the duration between the current time and the last known write time. Continuous data collection is resumed only after the previous data has been recovered. 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

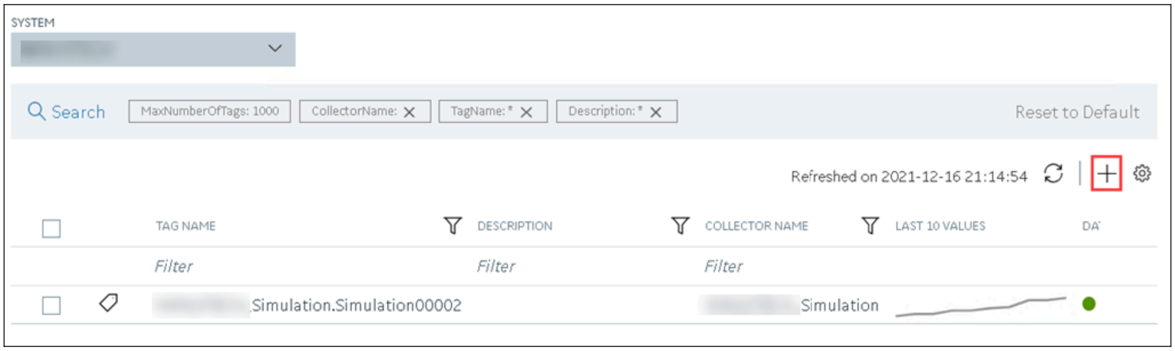
Add Tags for the Data Store Using Configuration Hub

- Add the collector instance *(on page 556)* 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 575)*.

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 589)*.

1. Access Configuration Hub *(on page 97)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
3. Select +.

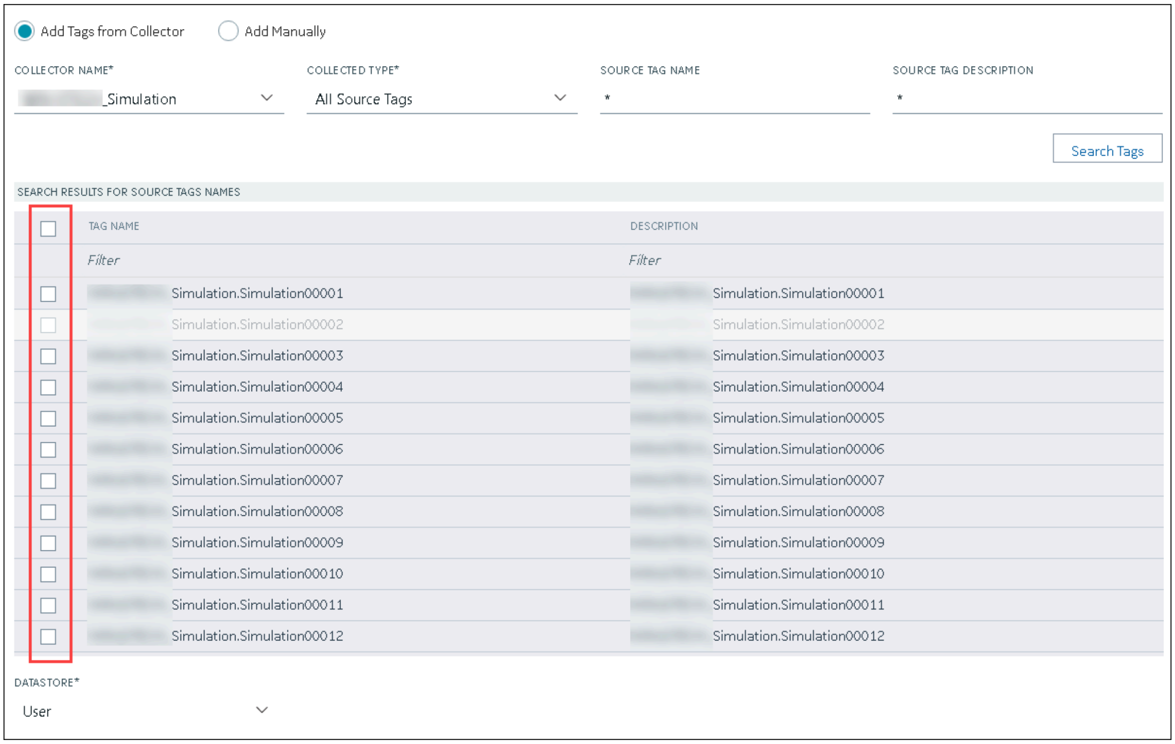


The **Add Tag-<system name>** page appears. The **Add Tags from Collector** option is selected by default.

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.



● Add Tags from Collector ○ Add Manually

COLLECTOR NAME*: [redacted]_Simulation

COLLECTED TYPE*: All Source Tags

SOURCE TAG NAME: *

SOURCE TAG DESCRIPTION: *

SEARCH RESULTS FOR SOURCE TAGS NAMES

□	TAG NAME	DESCRIPTION
	Filter	Filter
□	[redacted]Simulation.Simulation00001	[redacted]Simulation.Simulation00001
□	[redacted]Simulation.Simulation00002	[redacted]Simulation.Simulation00002
□	[redacted]Simulation.Simulation00003	[redacted]Simulation.Simulation00003
□	[redacted]Simulation.Simulation00004	[redacted]Simulation.Simulation00004
□	[redacted]Simulation.Simulation00005	[redacted]Simulation.Simulation00005
□	[redacted]Simulation.Simulation00006	[redacted]Simulation.Simulation00006
□	[redacted]Simulation.Simulation00007	[redacted]Simulation.Simulation00007
□	[redacted]Simulation.Simulation00008	[redacted]Simulation.Simulation00008
□	[redacted]Simulation.Simulation00009	[redacted]Simulation.Simulation00009
□	[redacted]Simulation.Simulation00010	[redacted]Simulation.Simulation00010
□	[redacted]Simulation.Simulation00011	[redacted]Simulation.Simulation00011
□	[redacted]Simulation.Simulation00012	[redacted]Simulation.Simulation00012

DATASTORE*: User

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.

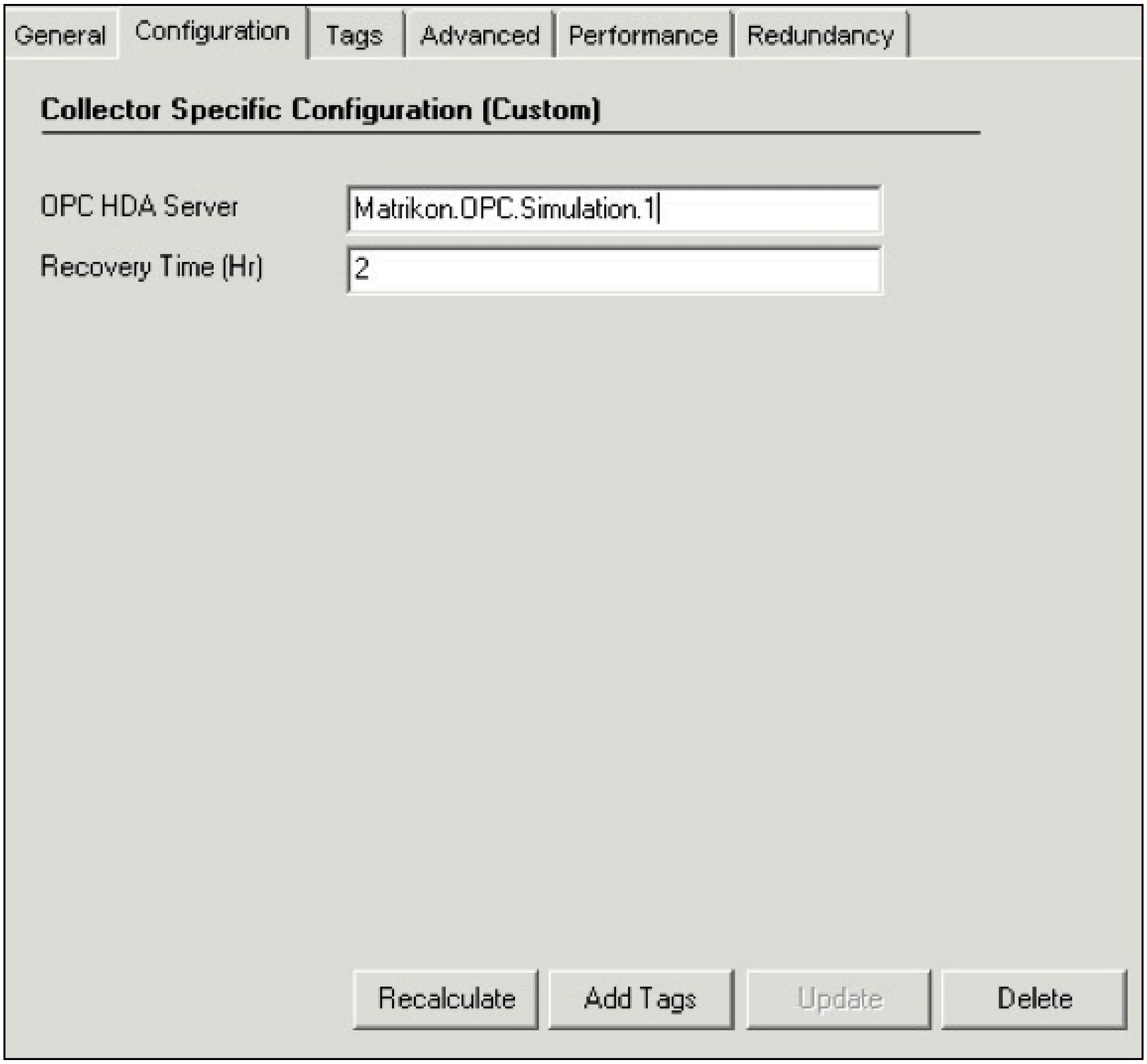
As needed, configure each tag by providing values for the tag properties. For information on the delta query modes, refer to Counter Delta Queries.

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.



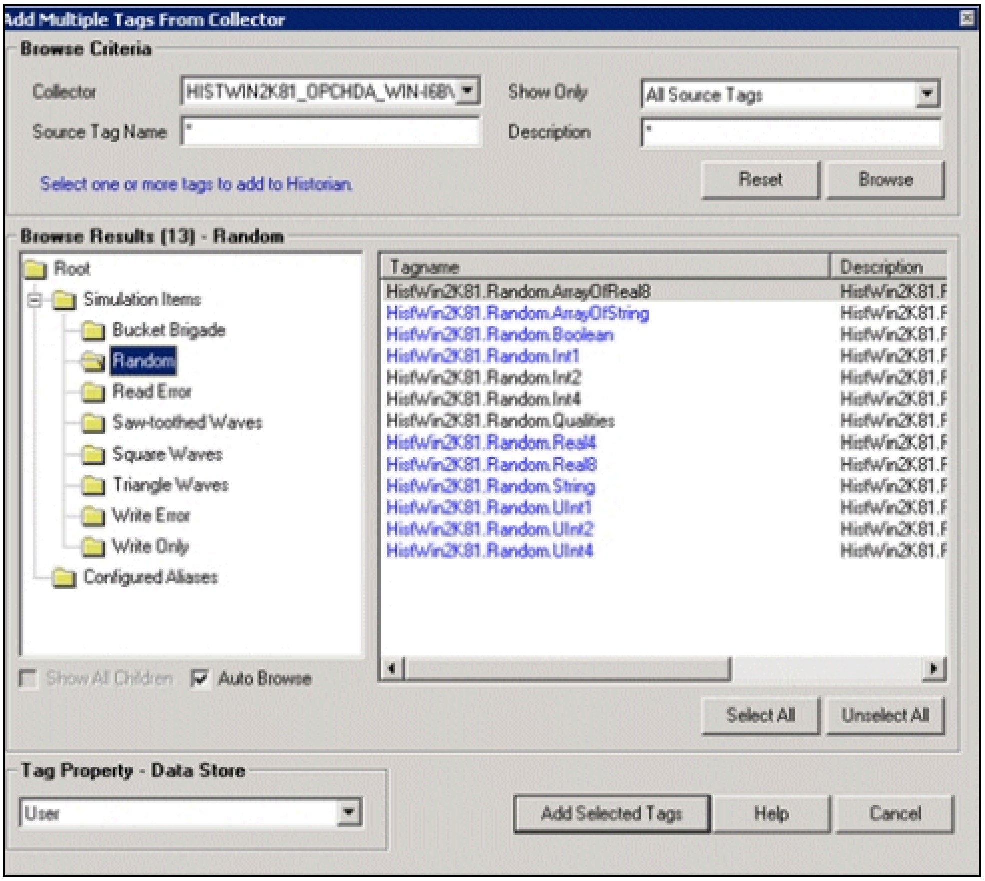
Collector Specific Configuration (Custom)

OPC HDA Server: Matrikon.OPC.Simulation.1

Recovery Time (Hr): 2

4. Select **Add Tags**.

   The **Add Multiple Tags from Collector** window appears.



Collector: HISTWIN2K81_OPCHDA_WIN-I68\

Show Only: All Source Tags

Source Tag Name: *

Description: *

Select one or more tags to add to Historian.

Tagname	Description
HistWin2K81.Random.ArrayOfReal8	HistWin2K81.F
HistWin2K81.Random.ArrayOfString	HistWin2K81.F
HistWin2K81.Random.Boolean	HistWin2K81.F
HistWin2K81.Random.Int1	HistWin2K81.F
HistWin2K81.Random.Int2	HistWin2K81.F
HistWin2K81.Random.Int4	HistWin2K81.F
HistWin2K81.Random.Qualities	HistWin2K81.F
HistWin2K81.Random.Real4	HistWin2K81.F
HistWin2K81.Random.Real8	HistWin2K81.F
HistWin2K81.Random.String	HistWin2K81.F
HistWin2K81.Random.UInt1	HistWin2K81.F
HistWin2K81.Random.UInt2	HistWin2K81.F
HistWin2K81.Random.UInt4	HistWin2K81.F

☐ Show All Children ☑ Auto Browse

Tag Property - Data Store: User

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 365)*.
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.

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`

OPC UA DA Collector Data Type	Recommended Data Type in Historian
`OpcUaType_Int16`	`ihTKInteger`
`OpcUaType_UInt16`	`ihTKUInt16`
`OpcUaType_Int32`	`ihTKDoubleInteger`
`OpcUaType_UInt32`	`ihTKUInt32`
`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

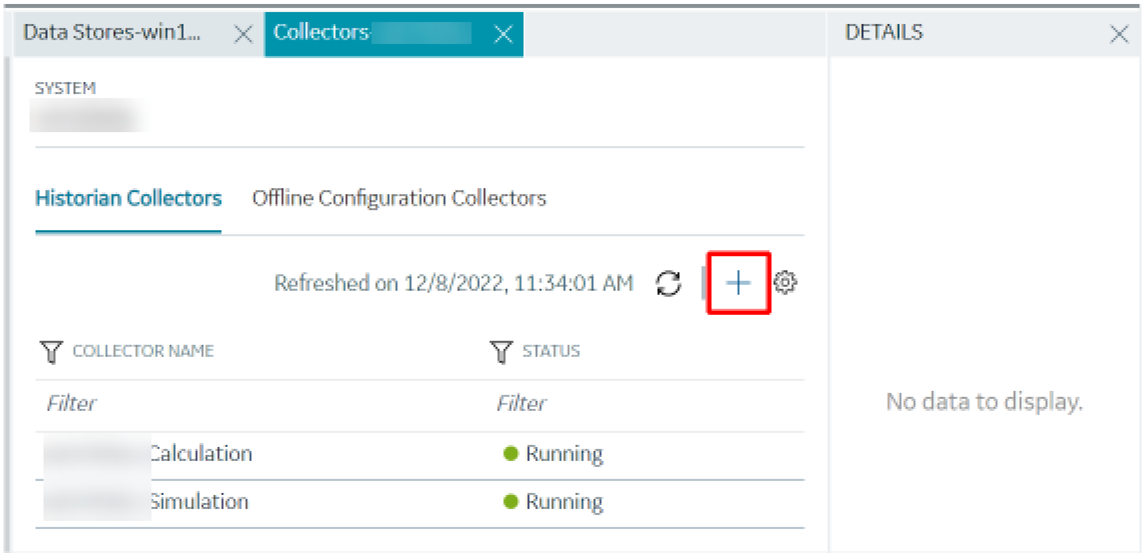
Add and Configure an OPC UA Data Access Collector

1. Deploy Proficy Historian for AWS. *(on page 43)*
2. Install collectors *(on page 90)*. You can install them on-premises or on a VPC (which can be different from the one on which Proficy Historian for AWS is deployed).
3. Enable TLS encryption for Collectors Connecting to Cloud Historian *(on page 150)*.

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 411)*.

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 1335)*, which does not require you to install Web-based Clients.

1. Access Configuration Hub *(on page 97)*.
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 +.



COLLECTOR NAME	STATUS
Filter	*Filter*
Calculation	● Running
Simulation	● Running

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-premise/Cloud 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 179)*.
   - **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 698)*.
   - **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 151)*.
     - AWS cloud. For more information, refer to AWS Cloud *(on page 158)*.
     - Google cloud. For more information, refer to Google Cloud *(on page 172)*.

b. If you need to send data to an on-premise/Cloud Historian server, select **Historian Server**.

   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**.

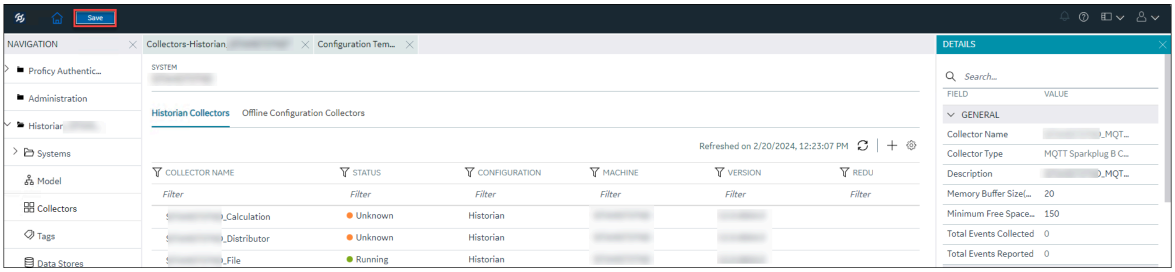
    The collector instance is added. The fields specific to the collector section appear in the **DETAILS** section.

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. You can establish a secured connectivity in one of the following ways: ◦ **Using certificates:** To use certificates, switch off the **User Security** toggle. ◦ **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 413)*.
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.

Field	Description
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.

15. As needed, enter values in the other sections common to all collectors *(on page 669)*.
16. In the upper-left corner of the page, select **Save**.



SYSTEM [redacted]

Refreshed on 2/20/2024, 12:23:07 PM

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSION	REDU
Filter	Filter	Filter	Filter	Filter	Filter
[redacted]_Calculation	● Unknown	Historian	[redacted]	[redacted]	
[redacted]_Distributor	● Unknown	Historian	[redacted]	[redacted]	
[redacted]_File	● Running	Historian	[redacted]	[redacted]	

FIELD	VALUE
GENERAL	
Collector Name	[redacted]_MQT...
Collector Type	MQTT Sparkplug B C...
Description	[redacted]_MQT...
Memory Buffer Size(...	20
Minimum Free Space...	150
Total Events Collected	0
Total Events Reported	0

The changes to the collector instance are saved.

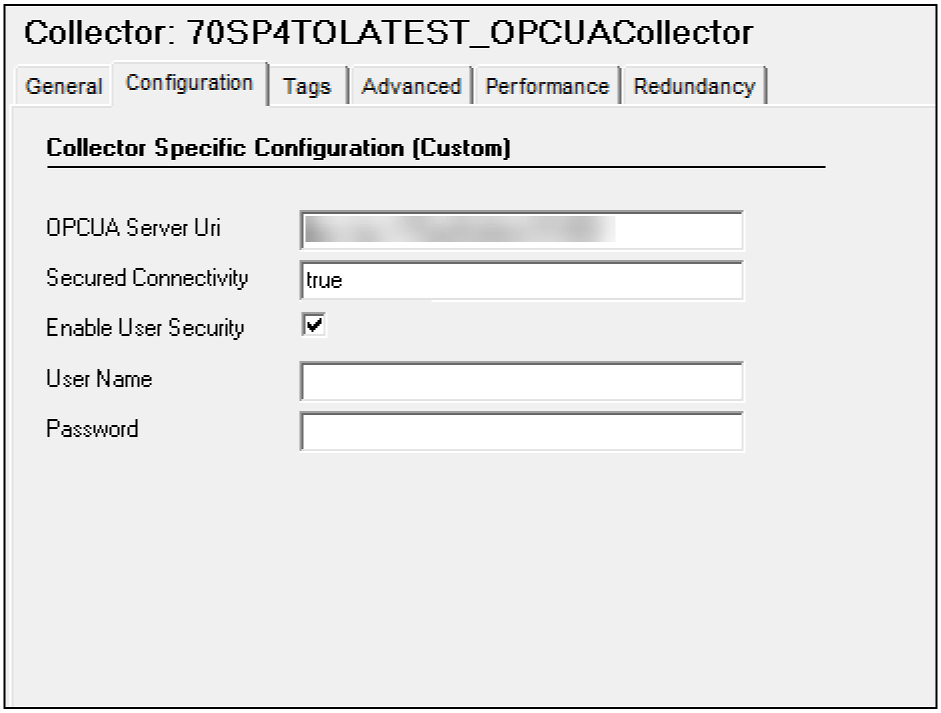
17. If needed, restart the collector.

If you have enabled secured connection, establish a secured connection between the OPC server and the collector *(on page 413)*.

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.



Collector: 70SP4TOLATEST_OPCUACollector

Collector Specific Configuration (Custom)

OPCUA Server Uri [redacted]

Secured Connectivity true

Enable User Security ☑

User Name

Password

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. **Note:** 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. You can establish a secured connectivity in one of the following ways: ◦ **Using certificates:** To use certificates, clear the **Enable User Security** check box. ◦ **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 413)*.

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 411)*, 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: ◦ `0`: Enter this value if want no trust. ◦ `1`: Enter this value if you want to trust temporarily. ◦ 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: ◦ `0`: Does not use a security policy. ◦ `1`: Uses the Basic128Rsa15 policy. This policy has theoretical problems and is not recommended. ◦ `2`: Uses the Basic 256 policy. This policy has known vulnerabilities and must not be used unless absolutely necessary. ◦ `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. ◦ `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. ◦ `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: ◦ `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. ◦ `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. ◦ `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

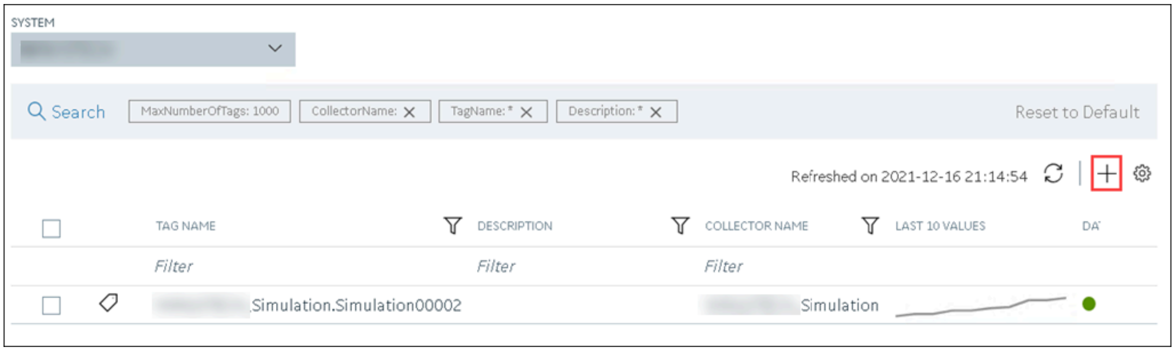
Add Tags for the Data Store Using Configuration Hub

- Add the collector instance *(on page 556)* 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 575)*.

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 589)*.

1. Access Configuration Hub *(on page 97)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
3. Select +.



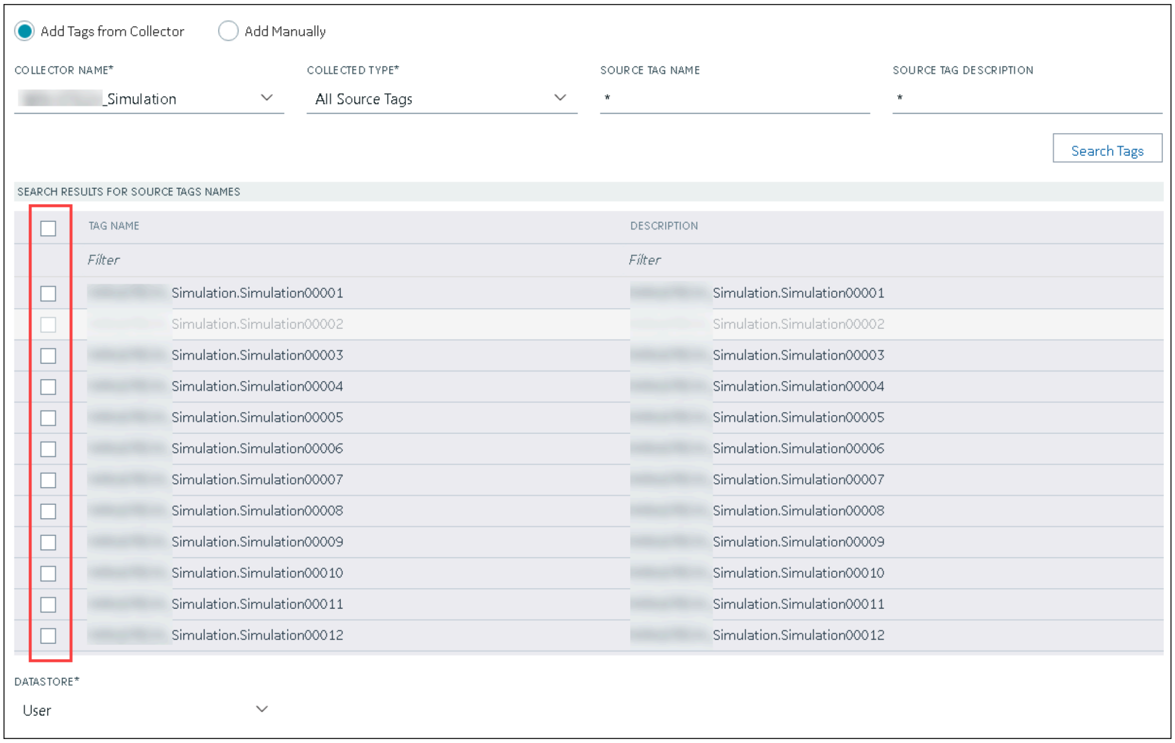
The **Add Tag-<system name>** page appears. The **Add Tags from Collector** option is selected by default.
4. Enter values as described in the following table.

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.



● Add Tags from Collector ○ Add Manually

COLLECTOR NAME*: [redacted]_Simulation

COLLECTED TYPE*: All Source Tags

SOURCE TAG NAME: *

SOURCE TAG DESCRIPTION: *

Search Tags

SEARCH RESULTS FOR SOURCE TAGS NAMES

□	TAG NAME	DESCRIPTION
	Filter	Filter
□	[redacted]Simulation.Simulation00001	[redacted]Simulation.Simulation00001
□	[redacted]Simulation.Simulation00002	[redacted]Simulation.Simulation00002
□	[redacted]Simulation.Simulation00003	[redacted]Simulation.Simulation00003
□	[redacted]Simulation.Simulation00004	[redacted]Simulation.Simulation00004
□	[redacted]Simulation.Simulation00005	[redacted]Simulation.Simulation00005
□	[redacted]Simulation.Simulation00006	[redacted]Simulation.Simulation00006
□	[redacted]Simulation.Simulation00007	[redacted]Simulation.Simulation00007
□	[redacted]Simulation.Simulation00008	[redacted]Simulation.Simulation00008
□	[redacted]Simulation.Simulation00009	[redacted]Simulation.Simulation00009
□	[redacted]Simulation.Simulation00010	[redacted]Simulation.Simulation00010
□	[redacted]Simulation.Simulation00011	[redacted]Simulation.Simulation00011
□	[redacted]Simulation.Simulation00012	[redacted]Simulation.Simulation00012

DATASTORE*: User

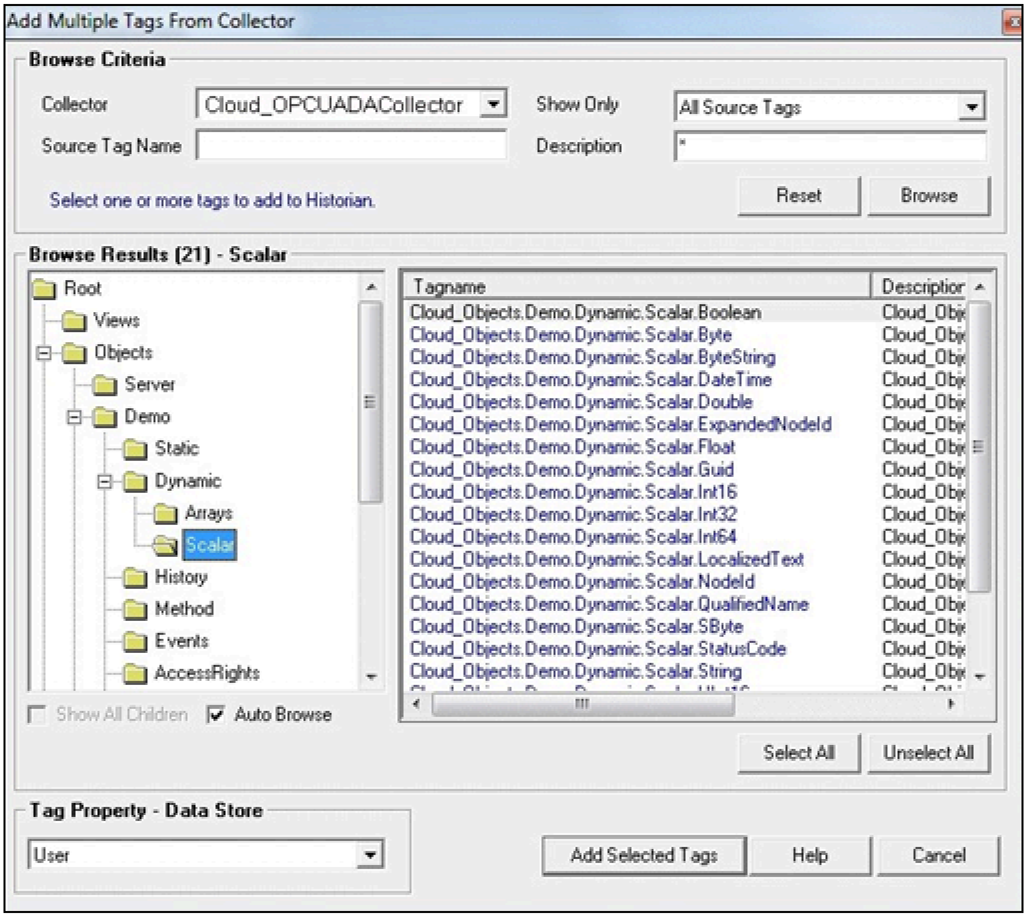
7. In the **DATA STORE** field, if you want to store the data in a different data store than the user data store, select the same.
8. Select **Add Tag**.

   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.



Add Multiple Tags From Collector

Browse Criteria

Collector: Cloud_OPCUADACollector

Show Only: All Source Tags

Source Tag Name:

Description: *

Select one or more tags to add to Historian.

Browse Results (21) - Scalar

Tagname	Description
Cloud_Objects.Demo.Dynamic.Scalar.Boolean	Cloud_Obj
Cloud_Objects.Demo.Dynamic.Scalar.Byte	Cloud_Obj
Cloud_Objects.Demo.Dynamic.Scalar.ByteString	Cloud_Obj
Cloud_Objects.Demo.Dynamic.Scalar.DateTime	Cloud_Obj
Cloud_Objects.Demo.Dynamic.Scalar.Double	Cloud_Obj
Cloud_Objects.Demo.Dynamic.Scalar.ExpandedNodeId	Cloud_Obj
Cloud_Objects.Demo.Dynamic.Scalar.Float	Cloud_Obj
Cloud_Objects.Demo.Dynamic.Scalar.Guid	Cloud_Obj
Cloud_Objects.Demo.Dynamic.Scalar.Int16	Cloud_Obj
Cloud_Objects.Demo.Dynamic.Scalar.Int32	Cloud_Obj
Cloud_Objects.Demo.Dynamic.Scalar.Int64	Cloud_Obj
Cloud_Objects.Demo.Dynamic.Scalar.LocalizedText	Cloud_Obj
Cloud_Objects.Demo.Dynamic.Scalar.NodeId	Cloud_Obj
Cloud_Objects.Demo.Dynamic.Scalar.QualifiedName	Cloud_Obj
Cloud_Objects.Demo.Dynamic.Scalar.SByte	Cloud_Obj
Cloud_Objects.Demo.Dynamic.Scalar.StatusCode	Cloud_Obj
Cloud_Objects.Demo.Dynamic.Scalar.String	Cloud_Obj

□ Show All Children ☑ Auto Browse

Tag Property - Data Store: User

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 413)*.
- Ensure that you have provided a valid username and password *(on page 411)*.
- 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.

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 431)*.

  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 432)*/delete *(on page 432)* 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:

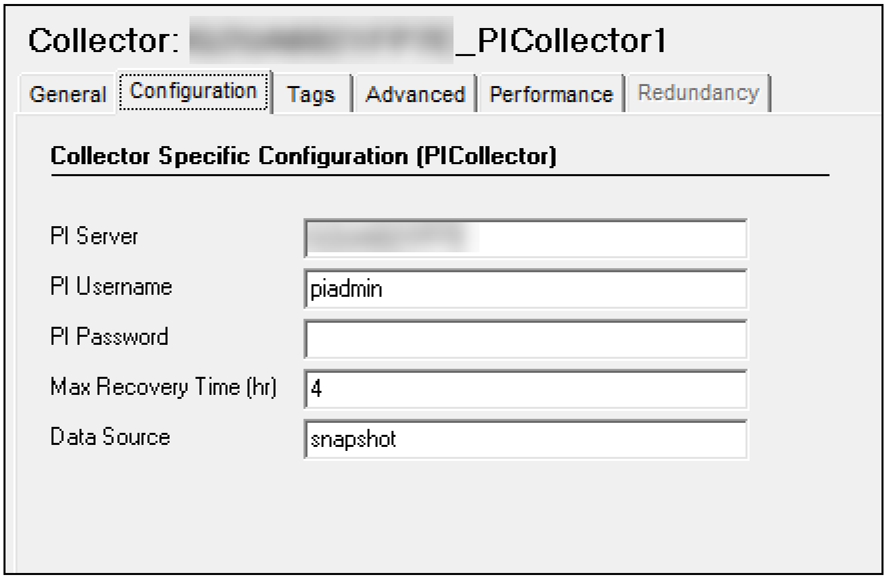
1. Run the Registry Editor (`regedit.exe`).
2. Navigate to `HKEY_LOCAL_MACHINE\SOFTWARE\Intellution, Inc.\iHistorian\Services\PICollector`.
3. In the case of a 64-bit system this may be found at `HKEY_LOCAL_ MACHINE\SOFTWARE\Wow6432Node \Intellution, Inc.\iHistorian\Services\PICollector`
4. If you have 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 433)*, 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.



Collector: [redacted]_PICollector1

Collector Specific Configuration (PICollector)

PI Server [redacted]
PI Username piadmin
PI Password
Max Recovery Time (hr) 4
Data Source 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 433)*.

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**.

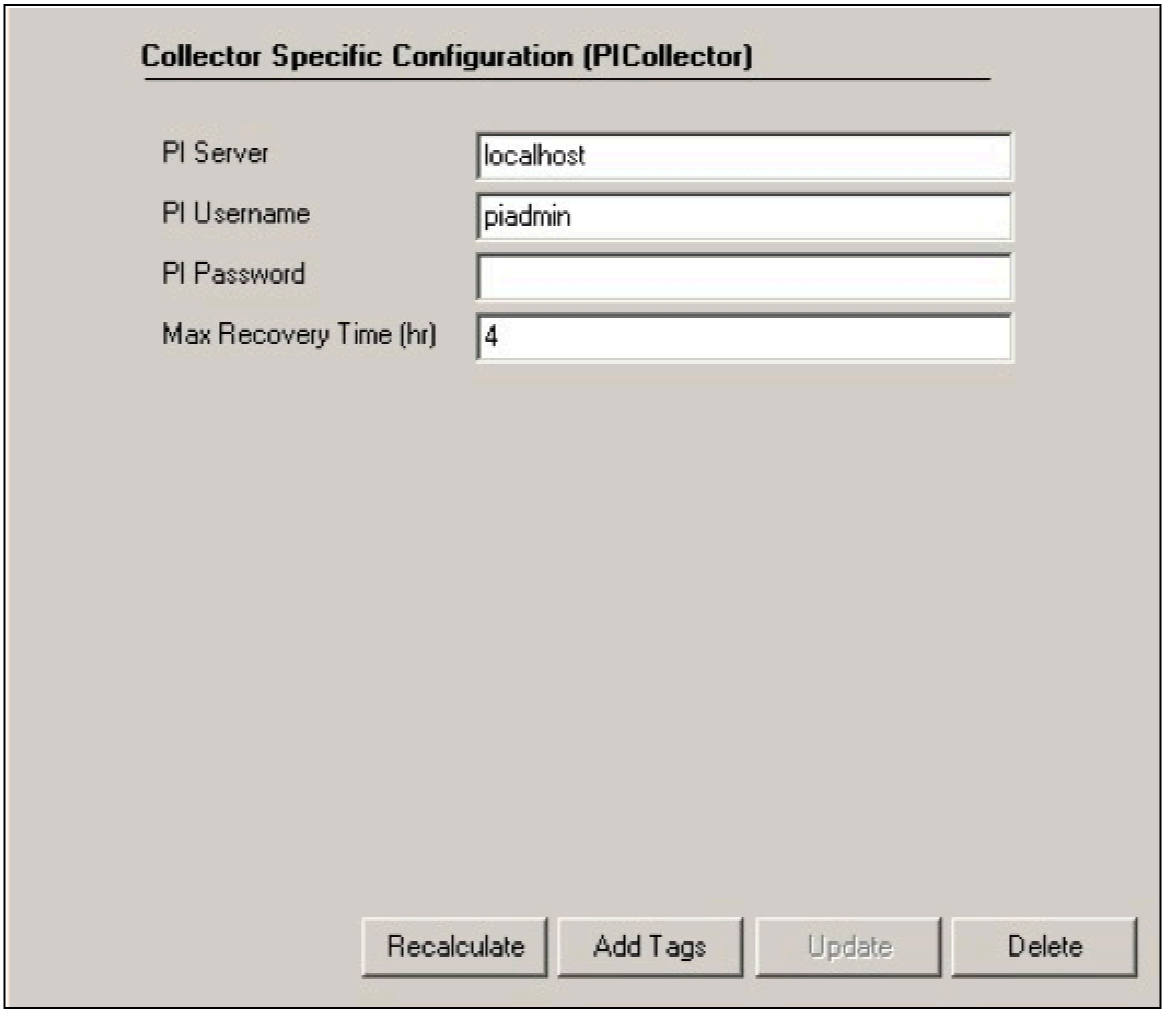
    b. Enter a prefix to add to OSI PI Tags in Historian, for example: `PI_`.

    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.



Collector Specific Configuration (PICollector)

PI Server: localhost
PI Username: piadmin
PI Password:
Max Recovery Time (hr): 4

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.

This field...	Indicates...
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 429)*.

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` 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 431)*.

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.

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
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 435)* and About Configuring OSI PI Data Archiver for OSI PI Distributor *(on page 436)* 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 429)*.

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`
`DIGITAL`	`Single Integer` 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.

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 441)*.
- 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)* and collectors *(on page)*.
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 442)* or RemoteCollectorConfigurator *(on page 444)*.

Adding a Python Collector Instance

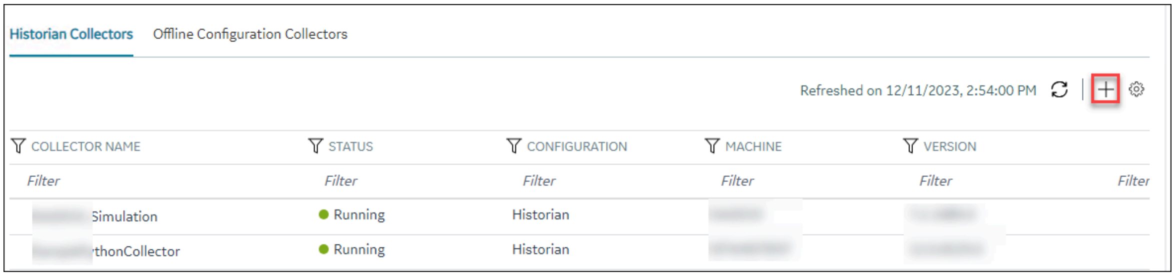
Add a Python Collector Instance using Configuration Hub

- Install Python Collector *(on page 441)*.

This topic describes how to add a Python collector instance using Configuration hub.

1. Access Configuration Hub *(on page 97)*.
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 +.



Refreshed on 12/11/2023, 2:54:00 PM

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSION	
Filter	Filter	Filter	Filter	Filter	Filter
[redacted]Simulation	● Running	Historian	[redacted]	[redacted]	
[redacted]thonCollector	● Running	Historian	[redacted]	[redacted]	

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.Collectors can send data to an on-premise or Cloud Historian server as well as cloud destinations.

   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 179)*.
      - **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 698)*.
      - **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 151)*.
        - AWS cloud. For more information, refer to AWS Cloud *(on page 158)*.
        - Google cloud. For more information, refer to Google Cloud *(on page 172)*.

   b. If you need to send data to an on-premise/Cloud 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-premise/Cloud 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 384)* whose data you want to collect using the collector.

Add a Python Collector Instance using RemoteCollectorConfigurator

- Install Python Collector *(on page 441)*.
- 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 1357)*.

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 1334)*.

    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 1334)*.

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 1339)*.

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 384)* 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).*

Configuring the Python Collector

Configure the Python Collector using Configuration Hub

Before you begin, install the Python collector *(on page 441)* and add an instance *(on page 442)* of the collector.

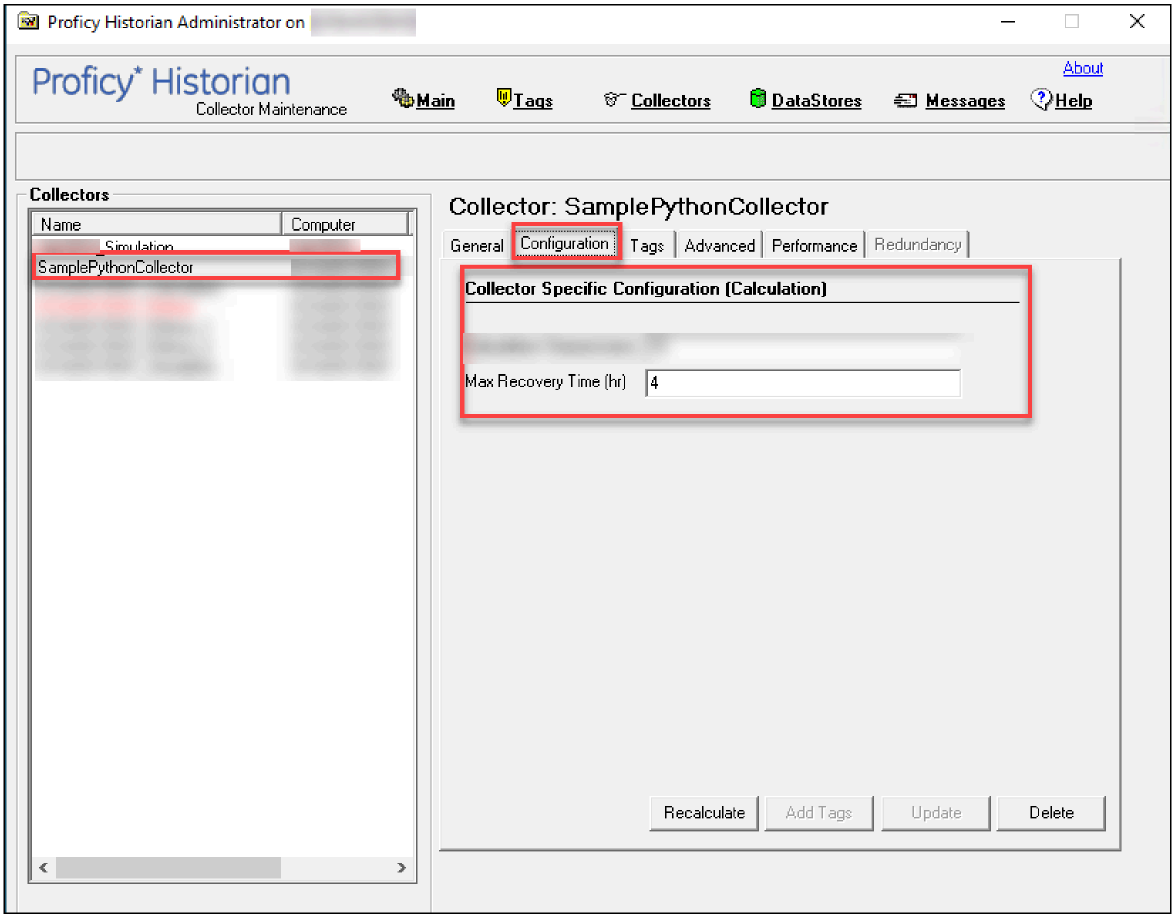
1. Access Configuration Hub *(on page 97)*.
2. Select **Collectors**, and then select the Python Collector instance that you want to configure.
   The fields specific to the collector instance appear in the **DETAILS** section.
3. In the **COLLECTOR SPECIFIC CONFIGURATION** section, enter values as specified 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. If you want to disable automatic calculation of the tag, set the value of this field to 0.

4. As needed, enter values in the other sections common to all collectors *(on page 669)*.
5. Restart the collector.
   The collector instance is configured.

Configure the Python Collector using Historian Administrator

1. Access Historian Administrator *(on page).*
2. Select **Collectors**, and then select the Python Collector instance that you want to configure.



Collector: SamplePythonCollector

Collector Specific Configuration (Calculation)

Max Recovery Time (hr) 4

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.

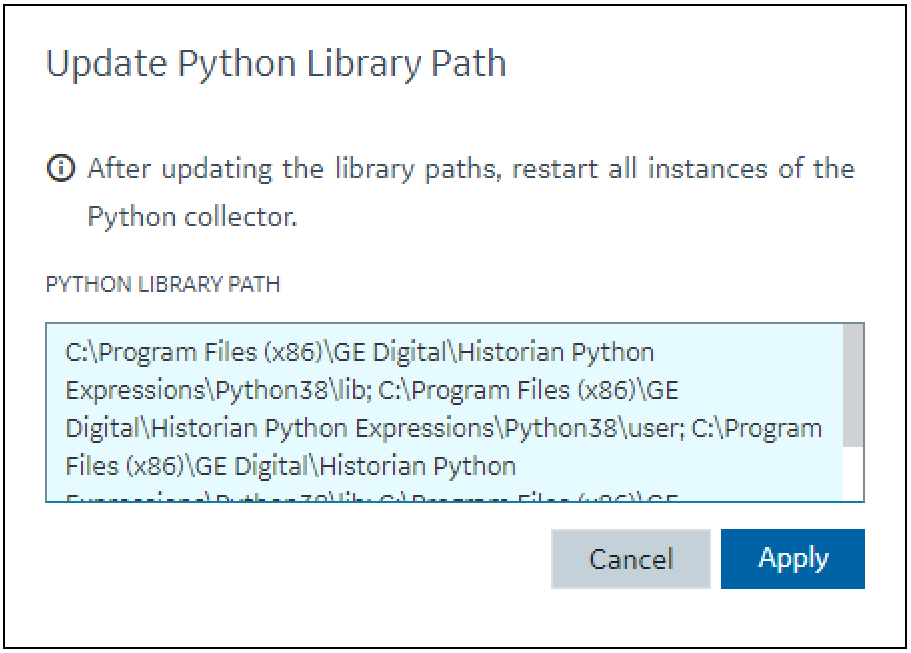
Field	Description
	If you want to disable automatic calculation of the tag, set the value of this field to 0.

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 97)*.
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.



Update Python Library Path

ⓘ After updating the library paths, restart all instances of the Python collector.

PYTHON LIBRARY PATH

C:\Program Files (x86)\GE Digital\Historian Python Expressions\Python38\lib; C:\Program Files (x86)\GE Digital\Historian Python Expressions\Python38\user; C:\Program Files (x86)\GE Digital\Historian Python

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

Parameter	Description
	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 453)*.

- **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.

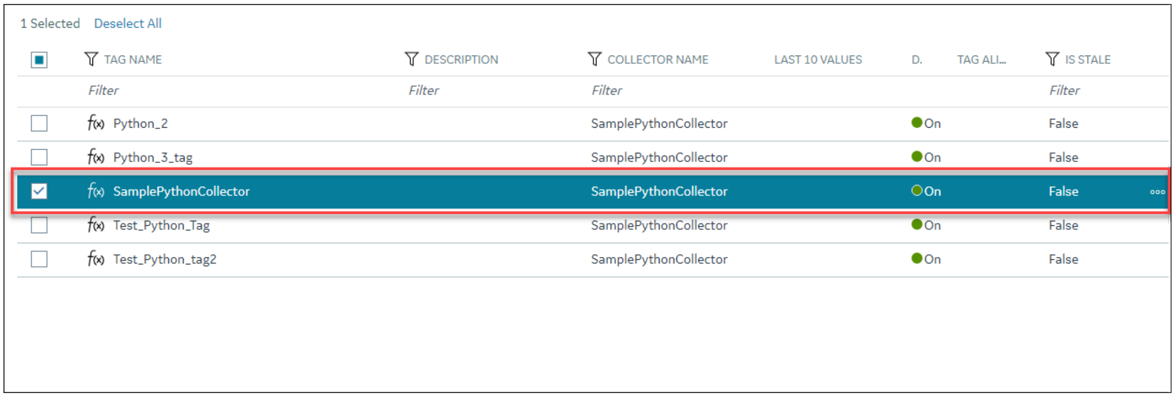
  Event-based triggers have a collection interval. The Python Collector notifies the archiver not to send notification of changes to trigger tags any faster than the collection interval setting.

  For information on creating an unsolicited trigger, refer to create an unsolicited trigger *(on page 453)*.

Create a Polled Trigger

1. Access Configuration Hub *(on page 97)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.

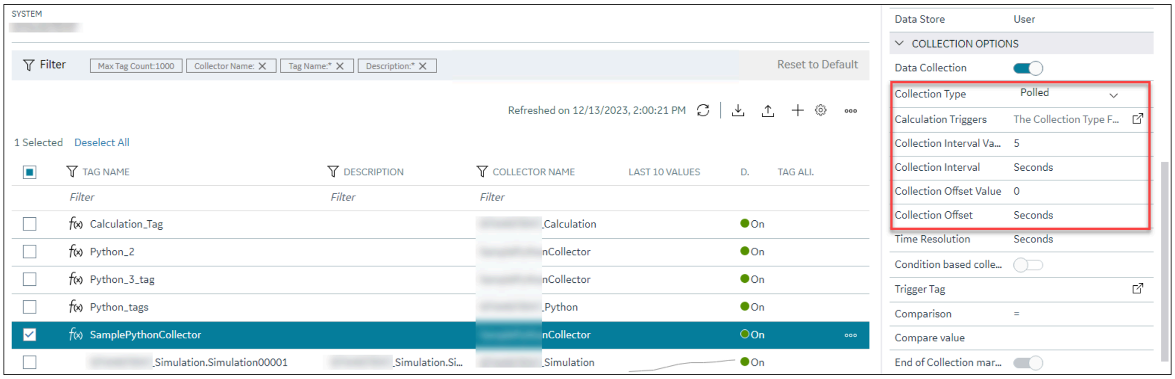
   A list of tags appears.
3. From the list of tags, select the Python collector tag (say, SamplePythonCollector).



1 Selected Deselect All

■	TAG NAME	DESCRIPTION	COLLECTOR NAME	LAST 10 VALUES	D.	TAG ALI...	IS STALE
	Filter	Filter	Filter				Filter
□	f(x) Python_2		SamplePythonCollector		● On		False
□	f(x) Python_3_tag		SamplePythonCollector		● On		False
☑	f(x) SamplePythonCollector		SamplePythonCollector		● On		False
□	f(x) Test_Python_Tag		SamplePythonCollector		● On		False
□	f(x) Test_Python_tag2		SamplePythonCollector		● On		False

4. In **DETAILS**, in the **Collection Options** section, select **Polled** from **Collection Type**.



SYSTEM

Filter Max Tag Count:1000 Collector Name: ✕ Tag Name:* ✕ Description:* ✕ Reset to Default

Refreshed on 12/13/2023, 2:00:21 PM

1 Selected Deselect All

■	TAG NAME	DESCRIPTION	COLLECTOR NAME	LAST 10 VALUES	D.	TAG ALI.
	Filter	Filter	Filter			
□	f(x) Calculation_Tag		_Calculation		● On	
□	f(x) Python_2		nCollector		● On	
□	f(x) Python_3_tag		nCollector		● On	
□	f(x) Python_tags		_Python		● On	
☑	f(x) SamplePythonCollector		nCollector		● On	
□	_Simulation.Simulation00001	_Simulation.Si...	_Simulation		● On	

Data Store	User
COLLECTION OPTIONS	
Data Collection	On
Collection Type	Polled
Calculation Triggers	The Collection Type F...
Collection Interval Va...	5
Collection Interval	Seconds
Collection Offset Value	0
Collection Offset	Seconds
Time Resolution	Seconds
Condition based colle...	Off
Trigger Tag	
Comparison	=
Compare value	
End of Collection mar...	Off

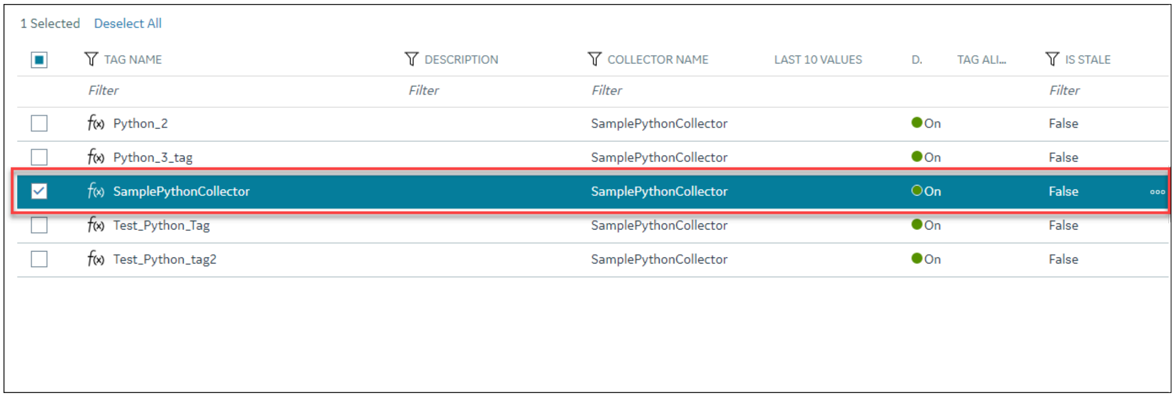
5. Set the **Collection Interval Value** and **Collection Offset** values. For example, if you want to set a trigger every day, set these values to 24 hours and 8 hours, respectively.
6. In the upper left corner of the page select **Save**.

   The triggers are created.

Create an Unsolicited Trigger

1. Access Configuration Hub *(on page 97)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.

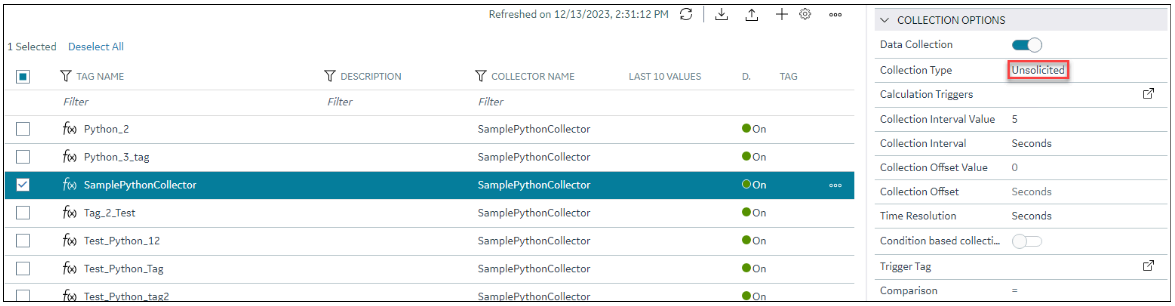
   A list of tags appears.
3. From the list of tags, select the Python collector tag (say, SamplePythonCollector).



1 Selected Deselect All

■	TAG NAME	DESCRIPTION	COLLECTOR NAME	LAST 10 VALUES	D.	TAG ALI...	IS STALE
	Filter	Filter	Filter				Filter
□	f(x) Python_2		SamplePythonCollector		● On		False
□	f(x) Python_3_tag		SamplePythonCollector		● On		False
☑	f(x) SamplePythonCollector		SamplePythonCollector		○ On		False
□	f(x) Test_Python_Tag		SamplePythonCollector		● On		False
□	f(x) Test_Python_tag2		SamplePythonCollector		● On		False

4. In **DETAILS**, in the **Collection Options** section, select **Unsolicited** from **Collection Type**.



Refreshed on 12/13/2023, 2:31:12 PM

1 Selected Deselect All

■	TAG NAME	DESCRIPTION	COLLECTOR NAME	LAST 10 VALUES	D.	TAG
	Filter	Filter	Filter			
□	f(x) Python_2		SamplePythonCollector		● On	
□	f(x) Python_3_tag		SamplePythonCollector		● On	
☑	f(x) SamplePythonCollector		SamplePythonCollector		○ On	
□	f(x) Tag_2_Test		SamplePythonCollector		● On	
□	f(x) Test_Python_12		SamplePythonCollector		● On	
□	f(x) Test_Python_Tag		SamplePythonCollector		● On	
□	f(x) Test_Python_tag2		SamplePythonCollector		● On	

COLLECTION OPTIONS

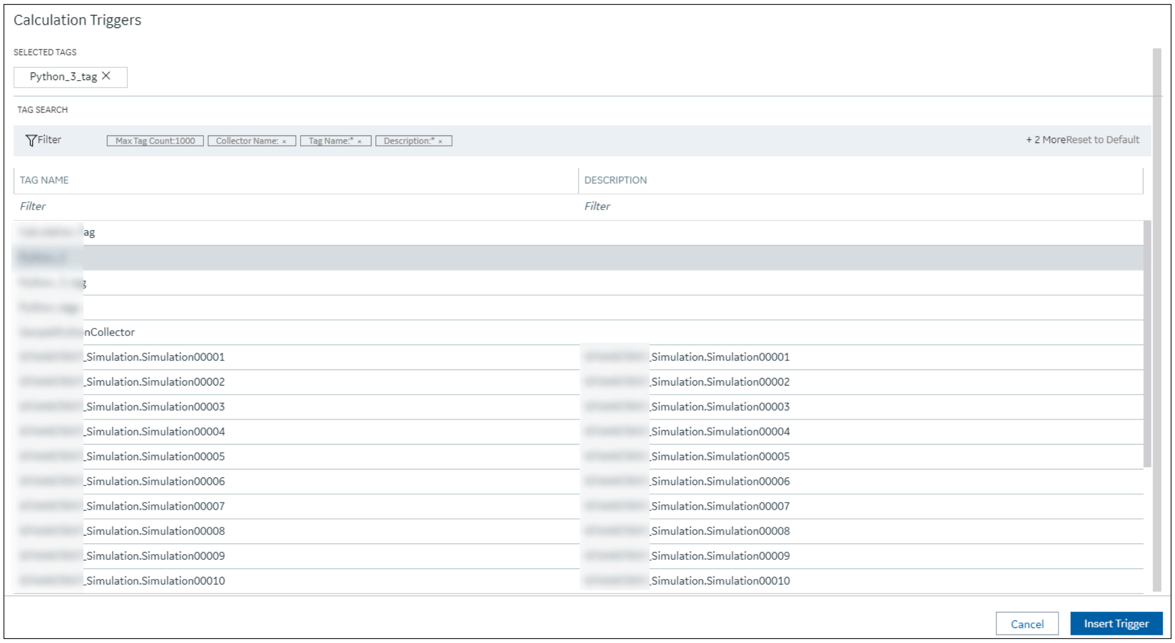
Data Collection	●
Collection Type	Unsolicited
Calculation Triggers	↗
Collection Interval Value	5
Collection Interval	Seconds
Collection Offset Value	0
Collection Offset	Seconds
Time Resolution	Seconds
Condition based collecti...	○
Trigger Tag	↗
Comparison	=

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.



Calculation Triggers

SELECTED TAGS

Python_3_tag ✕

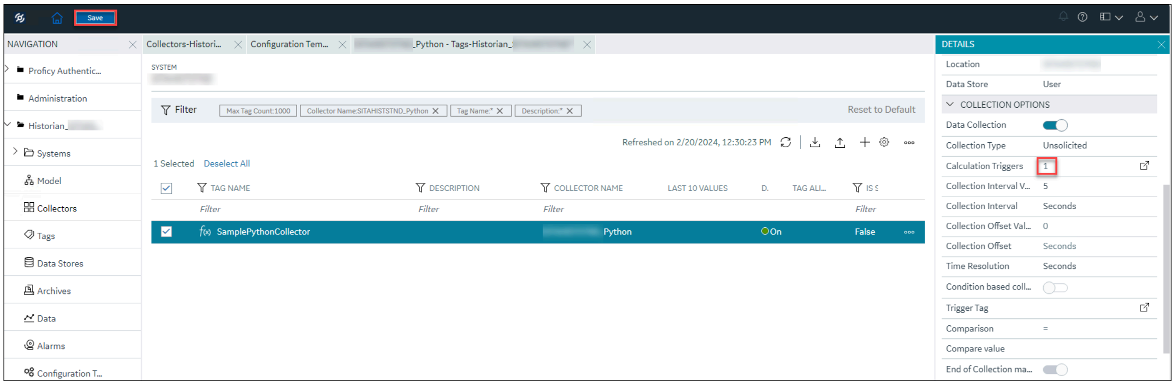
TAG SEARCH

Filter Max Tag Count:1000 Collector Name: × Tag Name:* × Description:* × + 2 MoreReset to Default

TAG NAME	DESCRIPTION
Filter	Filter
[redacted]ag	
[redacted]	
[redacted]g	
[redacted]	
[redacted]nCollector	
[redacted].Simulation.Simulation00001	[redacted].Simulation.Simulation00001
[redacted].Simulation.Simulation00002	[redacted].Simulation.Simulation00002
[redacted].Simulation.Simulation00003	[redacted].Simulation.Simulation00003
[redacted].Simulation.Simulation00004	[redacted].Simulation.Simulation00004
[redacted].Simulation.Simulation00005	[redacted].Simulation.Simulation00005
[redacted].Simulation.Simulation00006	[redacted].Simulation.Simulation00006
[redacted].Simulation.Simulation00007	[redacted].Simulation.Simulation00007
[redacted].Simulation.Simulation00008	[redacted].Simulation.Simulation00008
[redacted].Simulation.Simulation00009	[redacted].Simulation.Simulation00009
[redacted].Simulation.Simulation00010	[redacted].Simulation.Simulation00010

8. Select **Insert Trigger**.

   The trigger tag count will be displayed.



TAG NAME	DESCRIPTION	COLLECTOR NAME	LAST 10 VALUES	D.	TAG ALI...	IS S
☑ SamplePythonCollector		[redacted]Python		● On		False

Location	[redacted]
Data Store	User
Data Collection	● On
Collection Type	Unsolicited
Calculation Triggers	1
Collection Interval V...	5
Collection Interval	Seconds
Collection Offset Val...	0
Collection Offset	Seconds
Time Resolution	Seconds
Condition based coll...	○ Off
Trigger Tag	
Comparison	=
Compare value	
End of Collection ma...	

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.

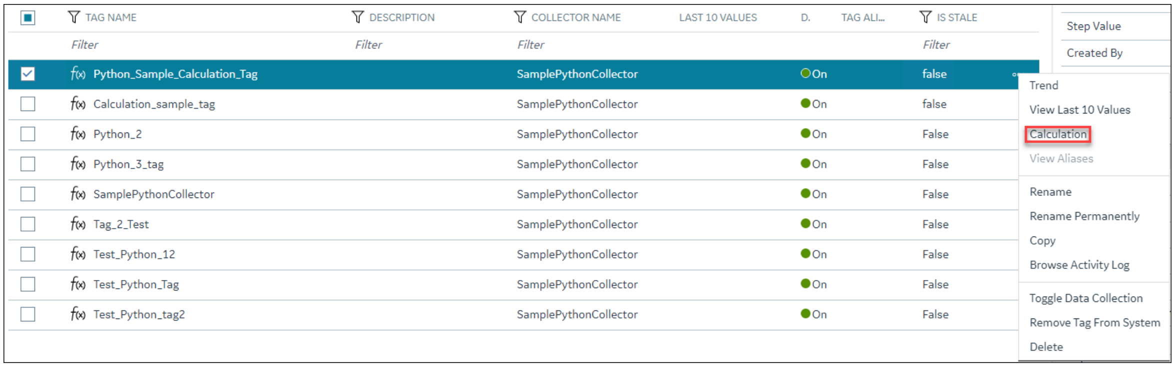
- Using pre-built functions *(on page 455)*.
- Using third-party or custom Python modules *(on page 458)*.

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 459)*.

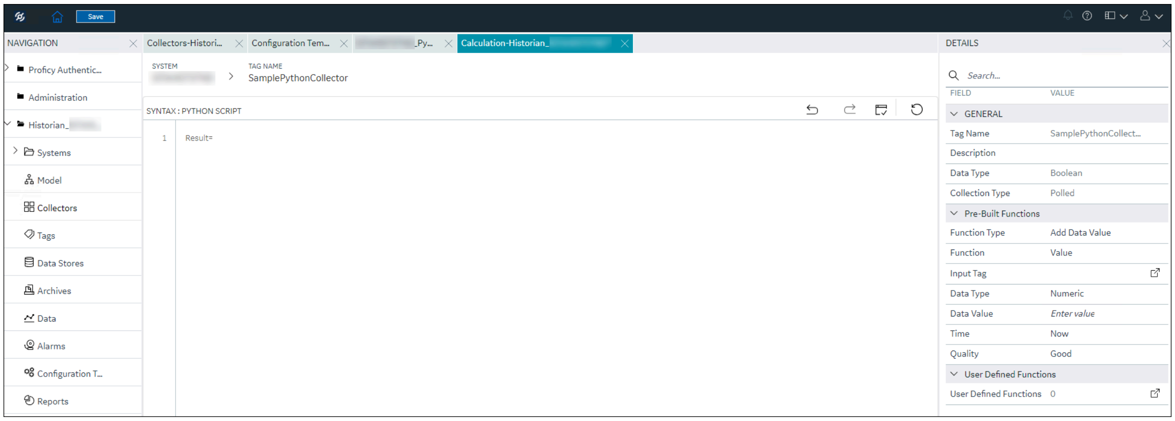
1. Access Configuration Hub *(on page 97)*
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.

   A list of tags appears.
3. Select a Python collector tag, right-click or select more options, and then select **Calculation**.



■	TAG NAME	DESCRIPTION	COLLECTOR NAME	LAST 10 VALUES	D.	TAG ALI...	IS STALE
	Filter	Filter	Filter				Filter
☑	f(x) Python_Sample_Calculation_Tag		SamplePythonCollector		● On		false
☐	f(x) Calculation_sample_tag		SamplePythonCollector		● On		false
☐	f(x) Python_2		SamplePythonCollector		● On		False
☐	f(x) Python_3_tag		SamplePythonCollector		● On		False
☐	f(x) SamplePythonCollector		SamplePythonCollector		● On		False
☐	f(x) Tag_2_Test		SamplePythonCollector		● On		False
☐	f(x) Test_Python_12		SamplePythonCollector		● On		False
☐	f(x) Test_Python_Tag		SamplePythonCollector		● On		False
☐	f(x) Test_Python_tag2		SamplePythonCollector		● On		False

The **PYTHON SCRIPT** editor appears.



SYSTEM [redacted] > TAG NAME SamplePythonCollector

SYNTAX : PYTHON SCRIPT

```
Result=
```

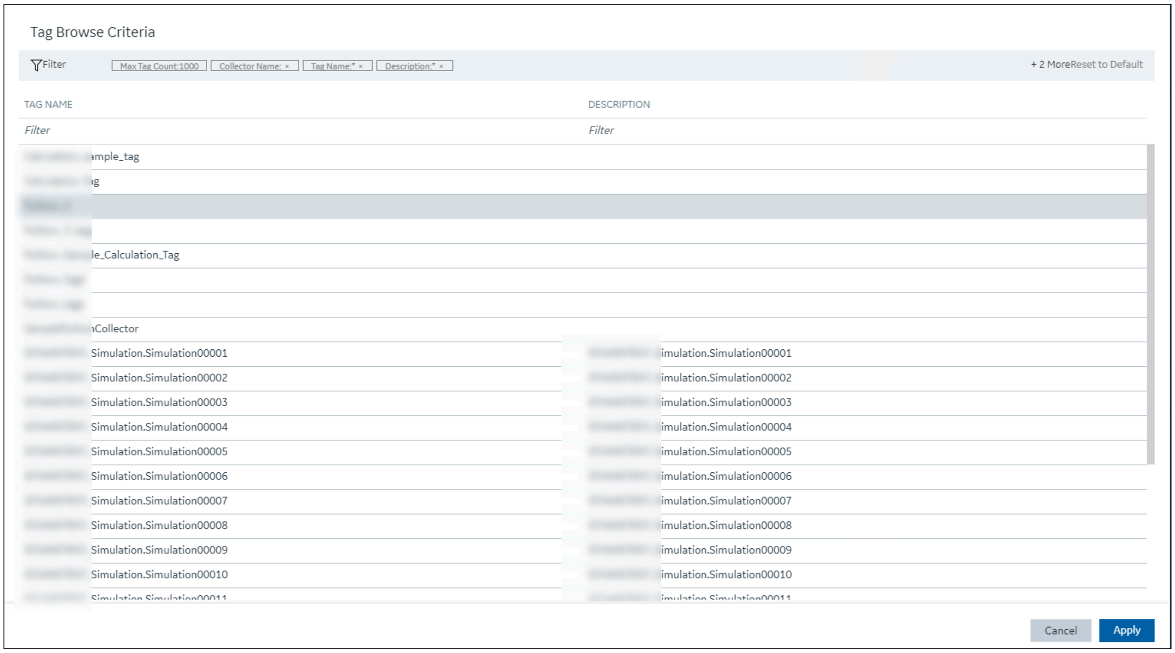
FIELD	VALUE
GENERAL	
Tag Name	SamplePythonCollect...
Description	
Data Type	Boolean
Collection Type	Polled
Pre-Built Functions	
Function Type	Add Data Value
Function	Value
Input Tag	
Data Type	Numeric
Data Value	Enter value
Time	Now
Quality	Good
User Defined Functions	
User Defined Functions	0

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.



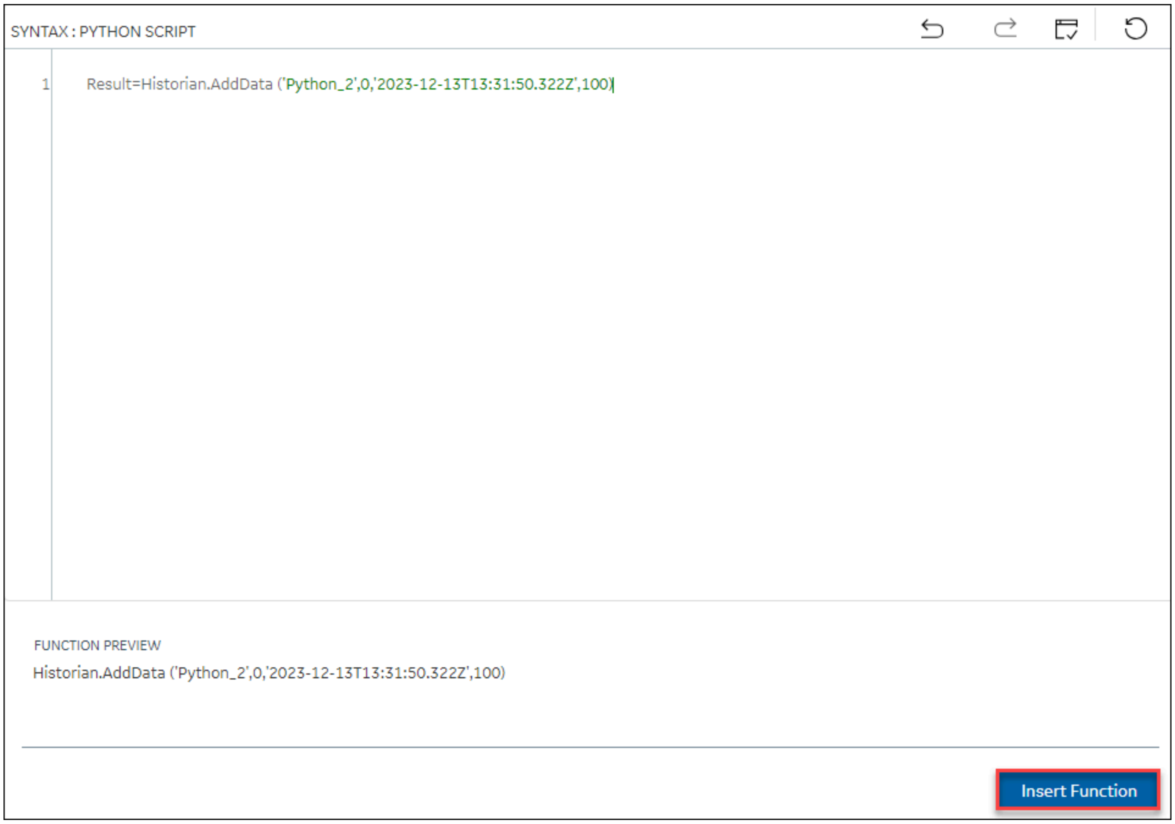
Tag Browse Criteria

TAG NAME	DESCRIPTION
Filter	*Filter*
[redacted]ample_tag	
[redacted]g	
[redacted]	
[redacted]	
[redacted]le_Calculation_Tag	
[redacted]	
[redacted]	
[redacted]Collector	
[redacted]Simulation.Simulation00001	[redacted]imulation.Simulation00001
[redacted]Simulation.Simulation00002	[redacted]imulation.Simulation00002
[redacted]Simulation.Simulation00003	[redacted]imulation.Simulation00003
[redacted]Simulation.Simulation00004	[redacted]imulation.Simulation00004
[redacted]Simulation.Simulation00005	[redacted]imulation.Simulation00005
[redacted]Simulation.Simulation00006	[redacted]imulation.Simulation00006
[redacted]Simulation.Simulation00007	[redacted]imulation.Simulation00007
[redacted]Simulation.Simulation00008	[redacted]imulation.Simulation00008
[redacted]Simulation.Simulation00009	[redacted]imulation.Simulation00009
[redacted]Simulation.Simulation00010	[redacted]imulation.Simulation00010
[redacted]Simulation.Simulation00011	[redacted]imulation.Simulation00011

9. Select **Apply**.

   The selected tag is added as the input tag.

10. Select **Insert Function**.



SYNTAX : PYTHON SCRIPT

```
Result=Historian.AddData ('Python_2',0,'2023-12-13T13:31:50.322Z',100)
```

FUNCTION PREVIEW

Historian.AddData ('Python_2',0,'2023-12-13T13:31:50.322Z',100)

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
    ```

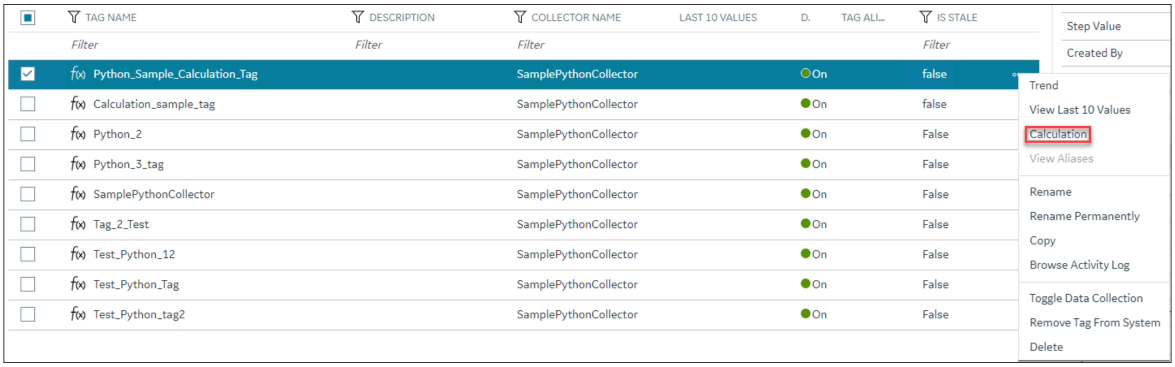
    ```
    C:\Program Files (x86)\GE Digital\Historian Python Expressions\Python38\user
    ```

    ```
    C:\Users\Administrator\AppData\Local\Programs\Python\Python38-32\Lib
    ```

    ```
    C:\Users\Administrator\AppData\Local\Programs\Python\Python38-32\Lib\site-packages
    ```

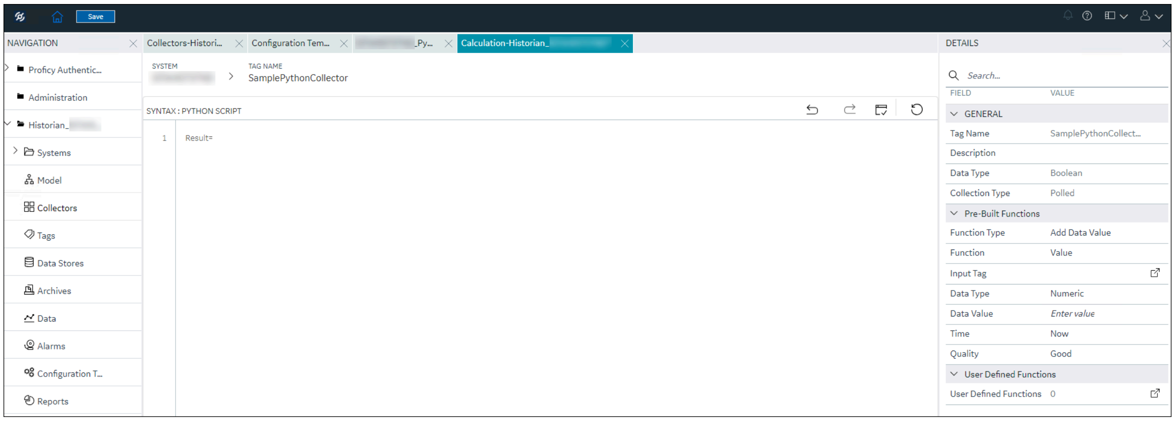
    You can also configure Python library path using Configuration Hub. For more information, refer to Configure Python Library path using Configuration Hub.
3. Access Configuration Hub *(on page 97)*
4. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.

    A list of tags appears.
5. Select a Python collector tag, right-click or select more options, and then select **Calculation**.



■	TAG NAME	DESCRIPTION	COLLECTOR NAME	LAST 10 VALUES	D.	TAG ALI...	IS STALE
	Filter	Filter	Filter				Filter
☑	f(x) Python_Sample_Calculation_Tag		SamplePythonCollector		● On		false
□	f(x) Calculation_sample_tag		SamplePythonCollector		● On		false
□	f(x) Python_2		SamplePythonCollector		● On		False
□	f(x) Python_3_tag		SamplePythonCollector		● On		False
□	f(x) SamplePythonCollector		SamplePythonCollector		● On		False
□	f(x) Tag_2_Test		SamplePythonCollector		● On		False
□	f(x) Test_Python_12		SamplePythonCollector		● On		False
□	f(x) Test_Python_Tag		SamplePythonCollector		● On		False
□	f(x) Test_Python_tag2		SamplePythonCollector		● On		False

The **PYTHON SCRIPT** editor appears.



SYSTEM [redacted] > TAG NAME SamplePythonCollector

SYNTAX : PYTHON SCRIPT

```
Result=
```

FIELD	VALUE
GENERAL	
Tag Name	SamplePythonCollect...
Description	
Data Type	Boolean
Collection Type	Polled
Pre-Built Functions	
Function Type	Add Data Value
Function	Value
Input Tag	
Data Type	Numeric
Data Value	*Enter value*
Time	Now
Quality	Good
User Defined Functions	
User Defined Functions	0

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 458)*.

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 as string in standard ISO 8601 format.	`Result=Historian.NextGoodQuality('Tag1','2023-12-13T16:34:19.154Z')`
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	`datetime_str =`
			` Historian.CurrentTime()`

Function Type	Function	Description	Example

		the time specified. Time is provided as string in standard ISO 8601 format.	`datetime_object = datetime.strptime(datetime_str, '%m/%d/%y %H:%M:%S.%f')`
			`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 1406)*. 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=10)`
			`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)`

Function Type	Function	Description	Example
			`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 1406)*. 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')`
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.	`The result is returned as a string.`
			`Format of timestamp is: "%m/%d/%y %H:%M:%S.%f"`
			`Example: “04/11/23 22:09:40.000”`

Function Type	Function	Description	Example

		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.
		Note: When a calculation is performed, the timestamp of the result is the time that the calculation has begun, not the time that it completed.
		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.
			`#Example to convert string timestamp to Python datetime object:`
			
			`from datetime import datetime`
			`datetime_str = Historian.CurrentTime()`
			`datetime_object = datetime.strptime(datetime_str, '%m/%d/%y %H:%M:%S.%f')`

Insert A Timestamp	`Historian.NextGoodTime(‘tag name’, ‘time’)`	The timestamp of the good raw sample after the time. Time is provid-	`Result=Historian.NextGoodTime('Tag',2023-12-13T17:54:44.223Z)`

Function Type	Function	Description	Example
		ed as string in standard ISO 8601 format.	
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')`
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	`Result=Historian.InterpolatedValue('Tag', '2023-12-13T18:05:29.367Z')'`

Function Type	Function	Description	Example
		as string in standard ISO 8601 format.	
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`
			`from time import time`
			
			` # Getting today's date and time`
			`todays_Date = datetime.now()`
			
			`# subtract 1 minute from current time`
			`timeToQuery = todays_Date - timedelta(hours=0,minutes=1,seconds=0)`
			` 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.	`datetime_str = Historian.CurrentTime()`

Function Type	Function	Description	Example

		Time is provided as string in standard ISO 8601 format.	`datetime_object =`
			`datetime.strptime(datetime_str, '%m/%d/%y %H:%M:%S.%f')`
			`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', ‘CriteriaString’, 'StartTime', 'EndTime', 'FilterTagname', 'FilterMode', 'FilterComparison', 'FilterValue')`	Filtered calculated data query with criteria string that returns a single value. All times are provided as string in standard ISO 8601 format.	`Result=Historian.AdvancedCalculationFilter('Tag','Average','','2023-12-13T18:16:29.333Z','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', 'Calcula-`	Filtered calculated data query with criteria string that returns the qual-	`Result=Historian.AdvancedCalculationFilterQual-`

Function Type	Function	Description	Example
	`tionMode', ‘Criteria-String’, 'StartTime', 'EndTime', 'FilterTagname', 'FilterMode', 'FilterComparison', 'FilterValue')`	ity of value. All times are provided as string in standard ISO 8601 format.	`ity('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
```

Server-to-Server Collector

Overview

Overview of the Server-to-Server Collector

Note:
When instantiating Server-to-Server and Server-to-Distributor from Configuration Hub, the Historian Node Name is set to "historian-svc". Provide the NLB DNS in the Destination Historian server field during collector instantiation.

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.

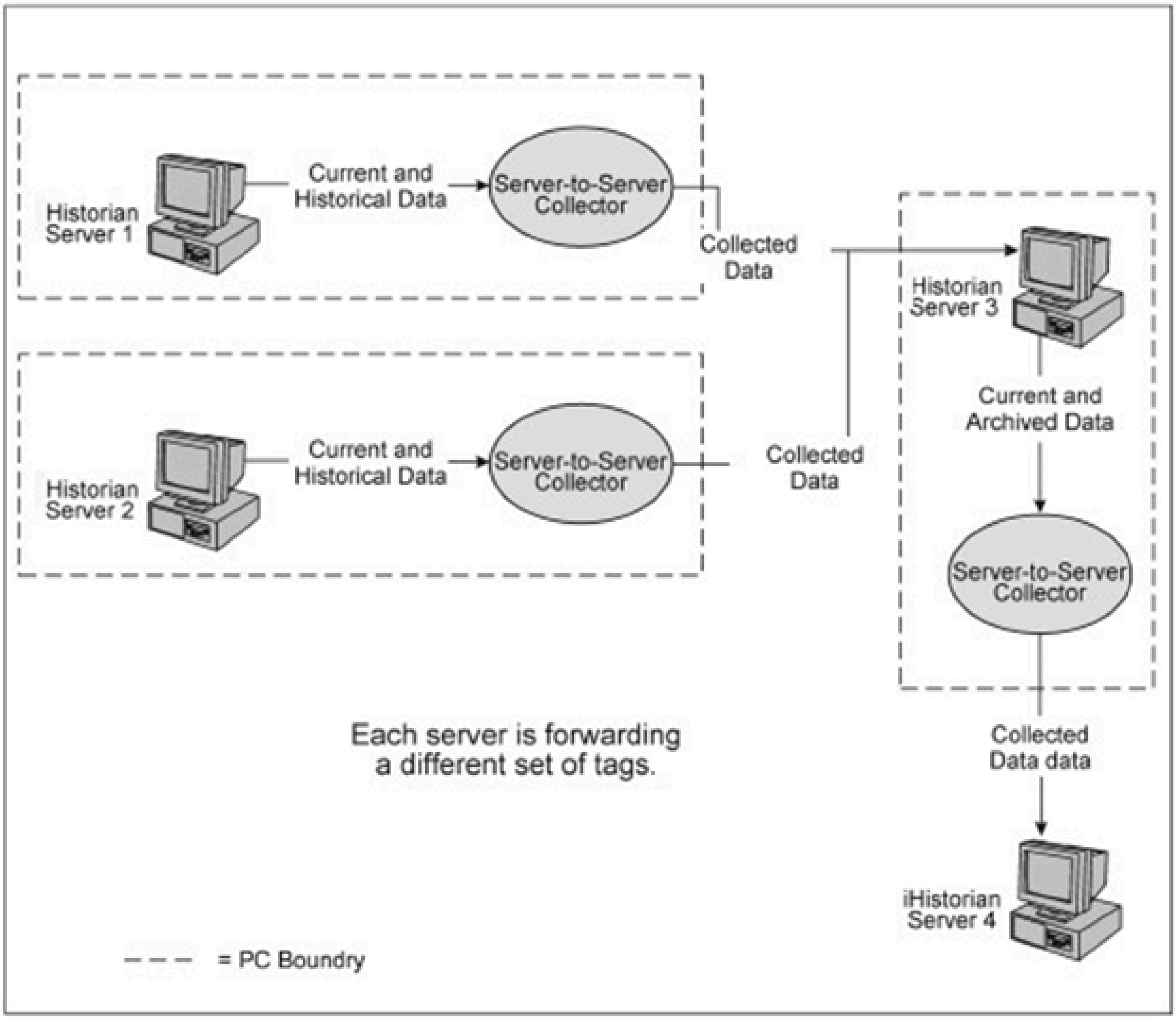
Message replication, if enabled, is event-based. Messages and alerts are sent to the destination server as they happen.

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 485)*.
- 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.

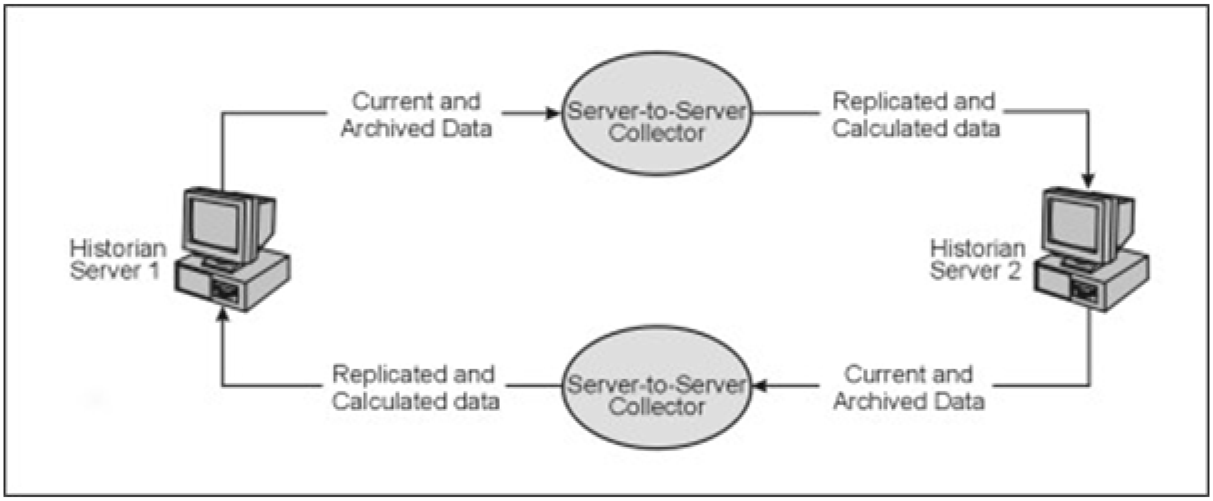
Data Flow in Multiple Server-to-Server Collectors

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.


Historian Server 1
Current and Historical Data
Server-to-Server Collector
Collected Data
Historian Server 2
Current and Historical Data
Server-to-Server Collector
Collected Data
Historian Server 3
Current and Archived Data
Server-to-Server Collector
Collected Data data
iHistorian Server 4
Each server is forwarding a different set of tags.
- - - = PC Boundry

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.


Current and Archived Data
Server-to-Server Collector
Replicated and Calculated data
Historian Server 1
Historian Server 2
Replicated and Calculated data
Server-to-Server Collector
Current and Archived Data

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 483)* 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

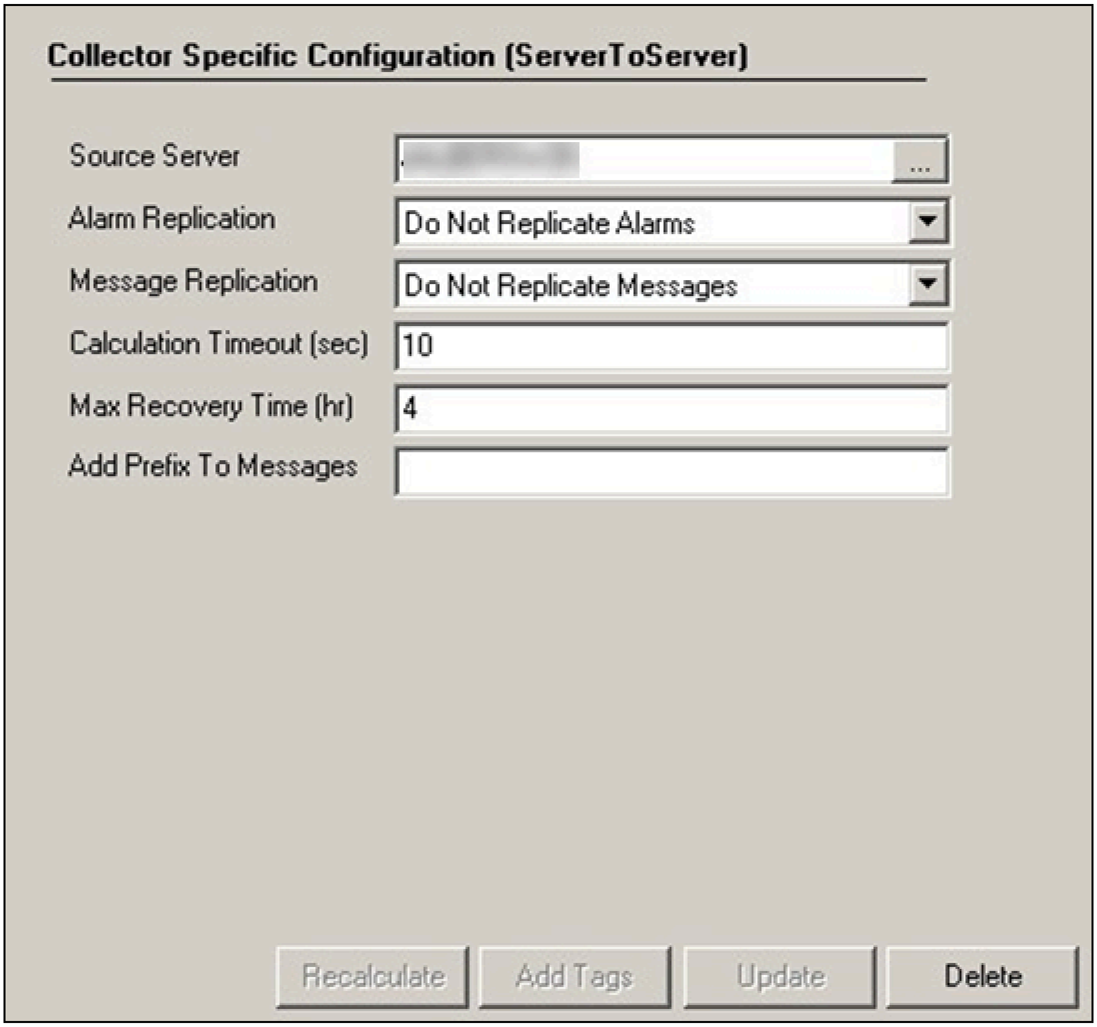
To use the Server-to-Server collector, you must perform the following tasks:

Number	Task	Notes
1	Install the collectors *(on page 90)* 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 1328)*.	This step is required to manage collectors installed on a remote machine.
3	Add an instance *(on page 556)* of the Server-to-Server collector using Configuration Hub.	This step is required.
4	Configure the Server-to-Server collector *(on page 483)* 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 199)*, adding it manually *(on page 191)*, or copying a tag *(on page 195)*.	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 245)* and Create an Unsolicited Trigger *(on page 247)*.
7	Create a formula similar to creating a formula for the Calculation collector. For instructions, refer to About Calculation Formulas *(on page 251)*.	This step is optional. 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. 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.



Collector Specific Configuration (ServerToServer)

Source Server: [redacted]

Alarm Replication: Do Not Replicate Alarms

Message Replication: Do Not Replicate Messages

Calculation Timeout (sec): 10

Max Recovery Time (hr): 4

Add Prefix To Messages:

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. Alarms and events data will automatically have a prefix added to it with the following syntax: `MachineName_Datasource` 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 EGUDescription
Collection	Data Type DataLength
Scaling0	HiEGU LoEGU InputScaling HiScale LoScale
Compression	ArchiveCompression ArchiveDeadband(%)

Examples of Data Collection

Raw Samples Collection Example

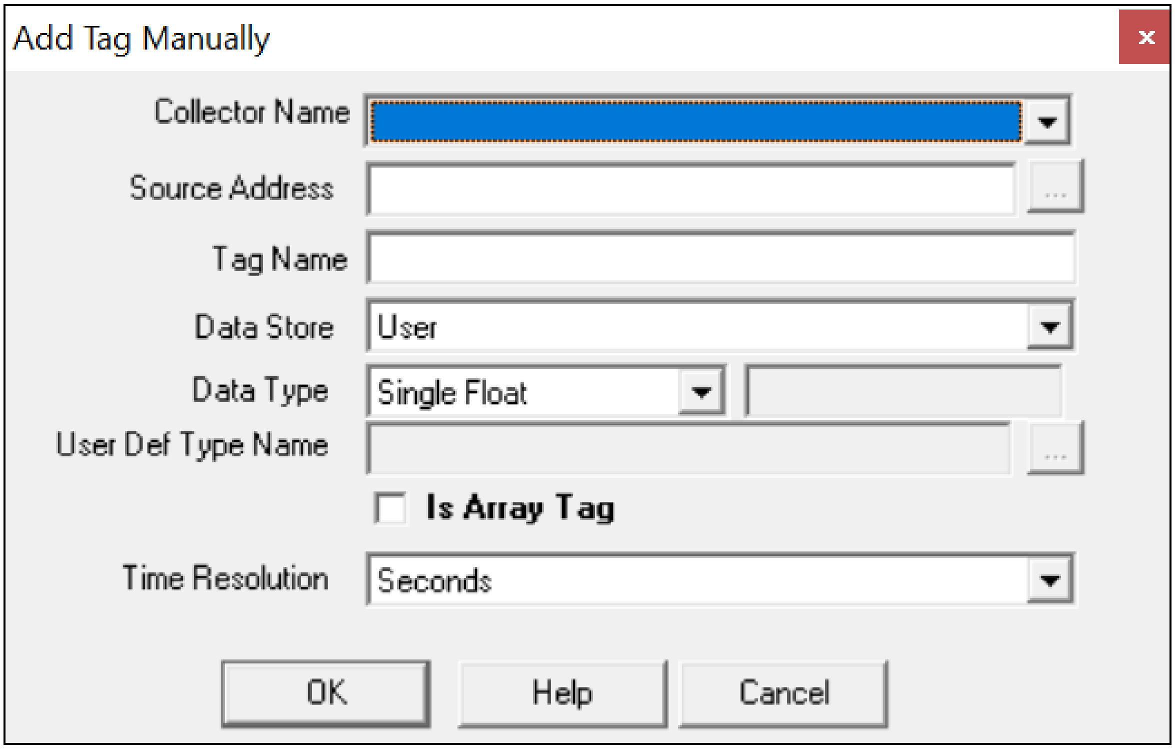
This topic describes how to collect raw samples using the Server-to-Server collector. The tagnames are the same on the source and destination. In this example, you add a tag manually, and give it a tagname on the destination server (representing the meaning of the calculated value).

1. Using Historian Administrator, browse the Server-to-Server collector for tags *(on page 199)* 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:



Add Tag Manually

Collector Name

Source Address

Tag Name

Data Store: User

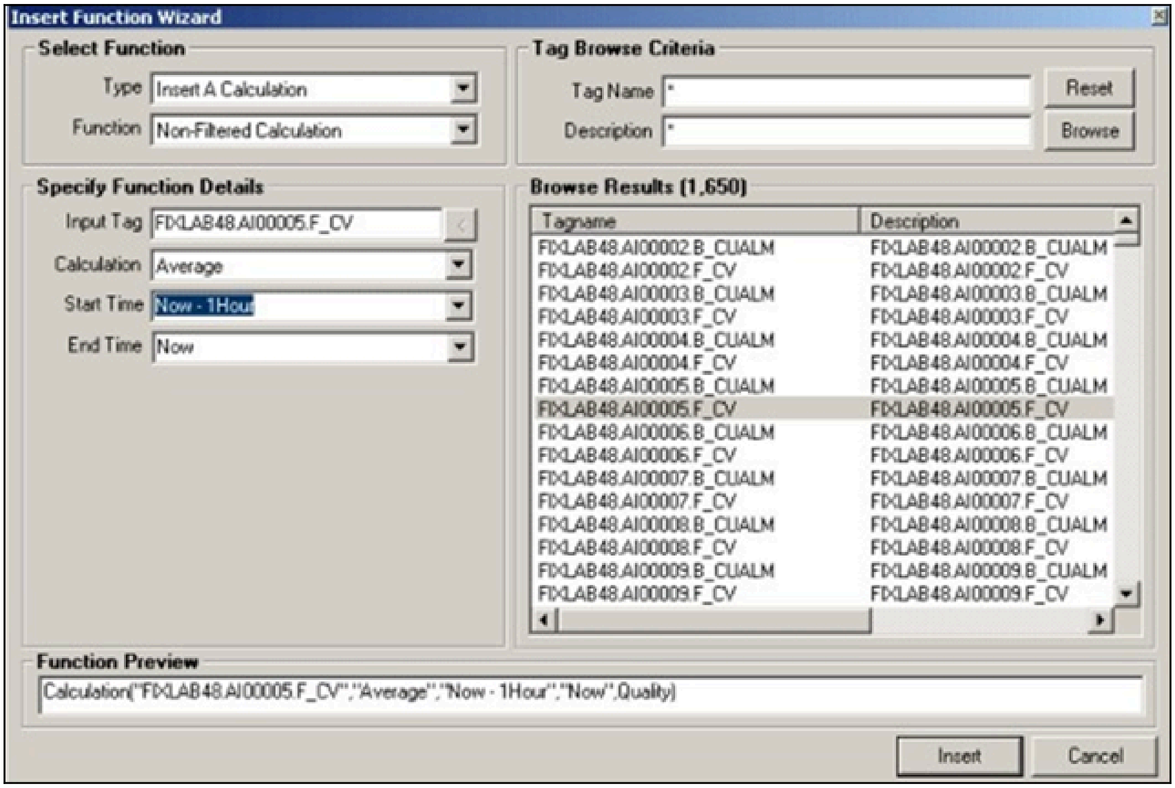
Data Type: Single Float

User Def Type Name

□ Is Array Tag

Time Resolution: Seconds

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**.



Type: Insert A Calculation

Function: Non-Filtered Calculation

Tag Name: *

Description: *

Input Tag: FIXLAB48.AI00005.F_CV

Calculation: Average

Start Time: Now - 1Hour

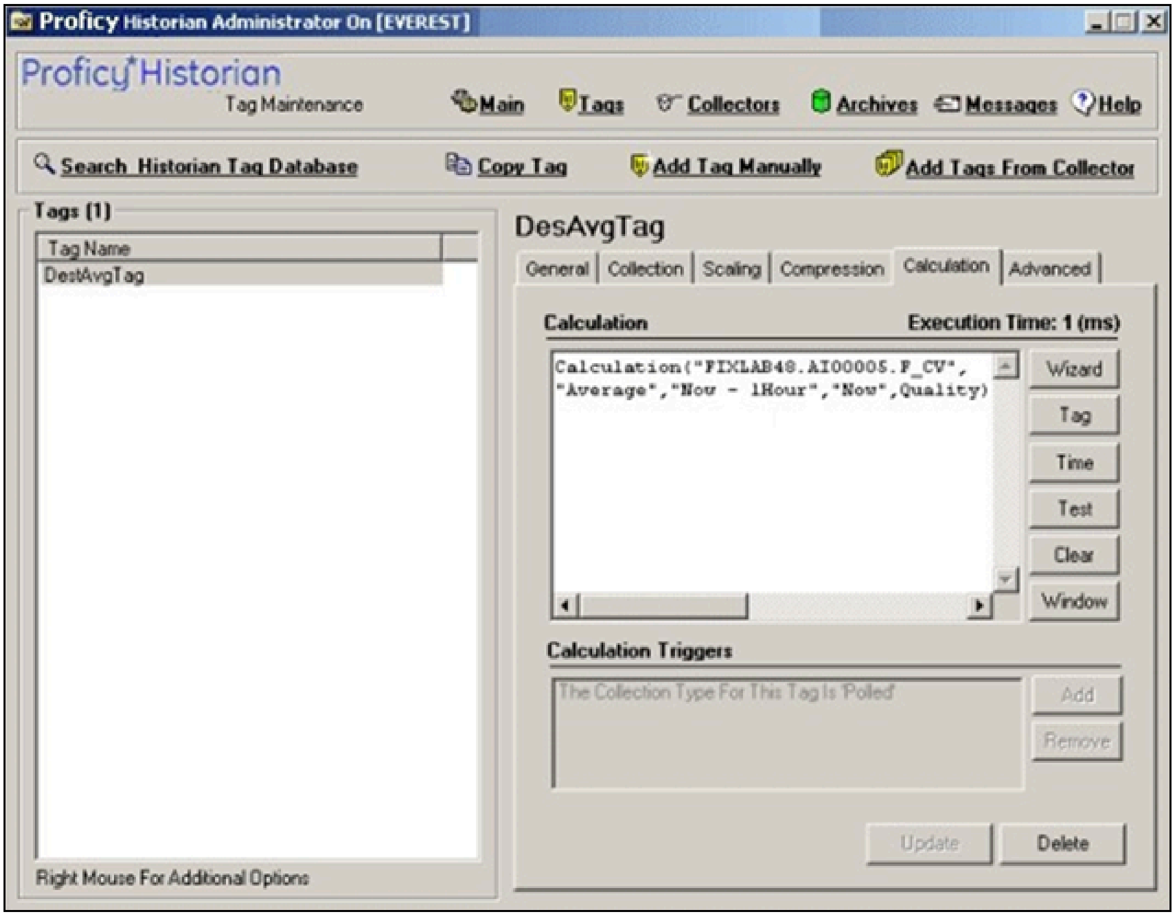
End Time: Now

Tagname	Description
FIXLAB48.AI00002.B_CUALM	FIXLAB48.AI00002.B_CUALM
FIXLAB48.AI00002.F_CV	FIXLAB48.AI00002.F_CV
FIXLAB48.AI00003.B_CUALM	FIXLAB48.AI00003.B_CUALM
FIXLAB48.AI00003.F_CV	FIXLAB48.AI00003.F_CV
FIXLAB48.AI00004.B_CUALM	FIXLAB48.AI00004.B_CUALM
FIXLAB48.AI00004.F_CV	FIXLAB48.AI00004.F_CV
FIXLAB48.AI00005.B_CUALM	FIXLAB48.AI00005.B_CUALM
FIXLAB48.AI00005.F_CV	FIXLAB48.AI00005.F_CV
FIXLAB48.AI00006.B_CUALM	FIXLAB48.AI00006.B_CUALM
FIXLAB48.AI00006.F_CV	FIXLAB48.AI00006.F_CV
FIXLAB48.AI00007.B_CUALM	FIXLAB48.AI00007.B_CUALM
FIXLAB48.AI00007.F_CV	FIXLAB48.AI00007.F_CV
FIXLAB48.AI00008.B_CUALM	FIXLAB48.AI00008.B_CUALM
FIXLAB48.AI00008.F_CV	FIXLAB48.AI00008.F_CV
FIXLAB48.AI00009.B_CUALM	FIXLAB48.AI00009.B_CUALM
FIXLAB48.AI00009.F_CV	FIXLAB48.AI00009.F_CV

```
Calculation("FIXLAB48.AI00005.F_CV","Average","Now - 1Hour","Now",Quality)
```

7. Select **Insert**.

   The **Tag Maintenance** page appears, showing the formula in the **Calculation** section.



```
Calculation("FIXLAB48.AI00005.F_CV",
"Average","Now - 1Hour","Now",Quality)
```

The Collection Type For This Tag Is 'Polled'

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 256)*, which helps you use any of the built-in functions *(on page 260)* orcreate your own function *(on page 258)*.
- Entering the syntax of the formula directly in the form of a VBScript code *(on page 254)*.

Before you create calculation formulas, refer to the general guidelines *(on page 252)*.

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)*.

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 271)*.

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 256)*.

> **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 246)* and Examples of Scheduling Unsolicited Triggers *(on page 250)*.

Create the tag that you want to use to store the calculation results. You can create the tag manually using Configuration Hub *(on page 589)*, Historian Administrator *(on page 192)* or the Web Admin console *(on page 194)*. Or, you can copy a tag *(on page 195)*.

1. To create a calculation formula using Configuration Hub:
    a. Access Configuration Hub *(on page 97)*.
    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)*.
    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 269)*.

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 271)* 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. **Note:** When a calculation is performed, the timestamp of the result is the time that the calculation has begun, not the time that it completed. 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.
`PreviousValue(<tag name>, Time)`	The tag value of the raw sample prior to the current time.
`PreviousQuality(<tag name>, Time)`	The quality of the tag (0 for bad quality and 100 for good quality) prior to the current time.
`PreviousGoodValue(<tag name>, Time)`	The latest good value of the raw sample prior to the current time.
`PreviousGoodQuality(<tag name>, Time)`	The good quality of the raw sample prior to the current time.
`PreviousTime(<tag name>, Time)`	The timestamp of the raw sample prior to the current time.
`PreviousGoodTime(<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 1406)*.
`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 1406)*.
`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. **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. 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. 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. 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. 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 260)*, 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 269)*. You can also use any of the built-in functions *(on page 260)*.

Create the tag that you want to use to store the calculation results. You can create the tag manually using Configuration Hub *(on page 589)*, Historian Administrator *(on page 192)* or the Web Admin console *(on page 194)*. Or, you can copy a tag *(on page 195)*.

1. To create a user-defined function using Configuration Hub:

    a. Access Configuration Hub *(on page 97)*.

    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 269)*.

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 2. 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 254)*.

For information on a list of the available types and associated functions, refer to Types of Functions Supported *(on page 268)*. For information on each pre-defined function, refer to Built-In Functions *(on page 260)*. In addition to the built-in functions, you can create your own customized functions *(on page 269)*.

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 589)*, Historian Administrator *(on page 192)* or the Web Admin console *(on page 194)*. Or, you can copy a tag *(on page 195)*.
2. Access the advanced options of the collector, and then disable the **On-line Tag Configuration Changes** option using Historian Administrator *(on page 192)*. In Configuration Hub *(on page 589)* 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 97)*.

    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)*.

   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 • Previous value • Next value • Interpolated value

Type of Action	Available Functions for the Action
Insert a calculation	• Unfiltered calculation • Filtered calculation
Insert a timestamp	• Time shortcut • Previous value timestamp • Next value timestamp • Current time
Check data quality	• Current value quality • Previous value quality • Next value quality
Set data quality	• Set Quality Good • Set Quality Bad
Add data value	Value
Insert a tag name	Tagname
Insert an alarm calculation	• Previous Alarm • Next Alarm • Get Alarm Property • Set Alarm Property • Add Event • New Alarm • Update Alarm • Return to Normal
Insert a multifield operation	• GetMultiFieldValue • GetMultiFieldQuality • 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 285)* and Manual Recalculation *(on page 286)*.

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`

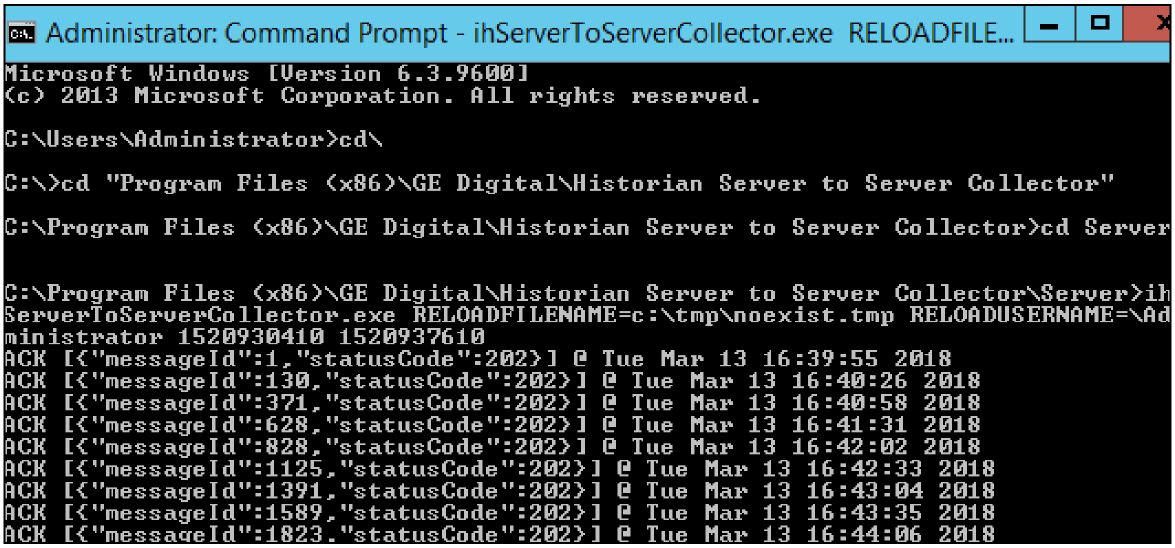
For multi-instance support, the command requires the interface name as shown in the following example:

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/


Administrator: Command Prompt - ihServerToServerCollector.exe RELOADFILE...

```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>cd\

C:\>cd "Program Files (x86)\GE Digital\Historian Server to Server Collector"

C:\Program Files (x86)\GE Digital\Historian Server to Server Collector>cd Server


C:\Program Files (x86)\GE Digital\Historian Server to Server Collector\Server>ih
ServerToServerCollector.exe RELOADFILENAME=c:\tmp\noexist.tmp RELOADUSERNAME=\Ad
ministrator 1520930410 1520937610
ACK [{"messageId":1,"statusCode":202}] @ Tue Mar 13 16:39:55 2018
ACK [{"messageId":130,"statusCode":202}] @ Tue Mar 13 16:40:26 2018
ACK [{"messageId":371,"statusCode":202}] @ Tue Mar 13 16:40:58 2018
ACK [{"messageId":628,"statusCode":202}] @ Tue Mar 13 16:41:31 2018
ACK [{"messageId":828,"statusCode":202}] @ Tue Mar 13 16:42:02 2018
ACK [{"messageId":1125,"statusCode":202}] @ Tue Mar 13 16:42:33 2018
ACK [{"messageId":1391,"statusCode":202}] @ Tue Mar 13 16:43:04 2018
ACK [{"messageId":1589,"statusCode":202}] @ Tue Mar 13 16:43:35 2018
ACK [{"messageId":1823,"statusCode":202}] @ Tue Mar 13 16:44:06 2018
```

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:
	`dim timetag1`
	`dim timetag2`
	`dim tag1`
	`dim tag2`
	`Dim x`

Tag	Description

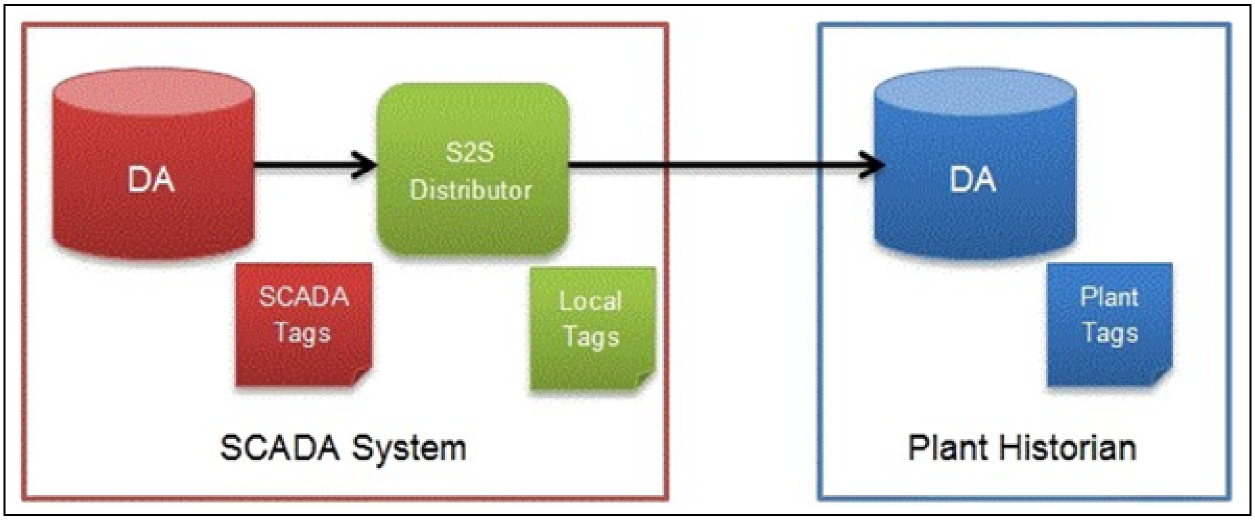
	tag1 = "T20.di-1.F_CV"
	tag2 = "t20.T20.DI-1.F_CV"
	x = DateAdd("s", 1,CurrentTime) ' add 1 second to calc time
	' Get the timestamp of the newest raw sample for tag1:
	timetag1 = previousTime(tag1, x)
	' Get the timestamp of the newest raw sample for tag2:
	timetag2 = previousTime(tag2, x)
	if timetag1 > timetag2 then
	' If tag1 triggered me, then:
	result = PreviousValue(tag1,CurrentTime)
	else
	' If tag2 triggered me, then:
	result = PreviousValue(Tag1, CurrentTime)
	end if

The Server-to-Server Distributor

Overview of the Server-to-Server Distributor

Note:
When instantiating Server-to-Server and Server-to-Distributor from Configuration Hub, the Historian Node Name is set to "historian-svc". Provide the NLB DNS in the Destination Historian server field during collector instantiation.

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

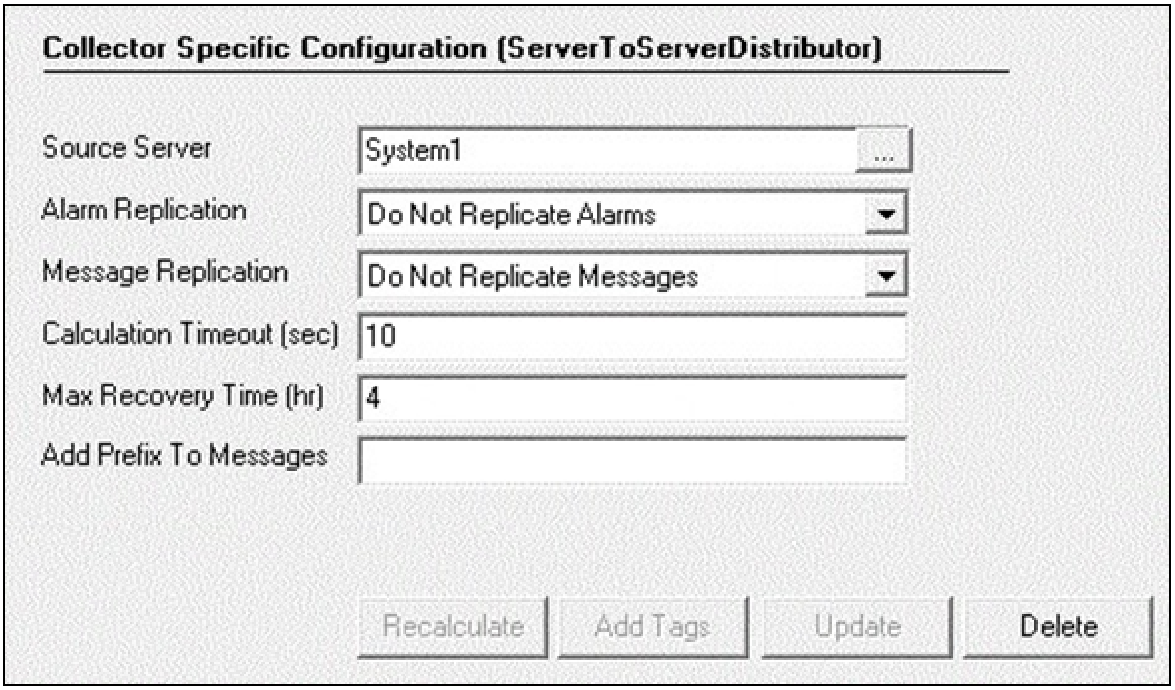
To use the Server-to-Server distributor, you must perform the following tasks:

Number	Task	Notes
1	Install the collectors *(on page 90)* 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 658)* on the source Historian server.	This step is required.
3	Start the Server-to-Server distributor *(on page 725)*.	This step is required.
4	Configure the Server-to-Server distributor *(on page 658)*.	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 754)*, adding it manually *(on page 589)*, or copying a tag *(on page 781)*.	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)* 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.



Collector Specific Configuration (ServerToServerDistributor)

Source Server: System1

Alarm Replication: Do Not Replicate Alarms

Message Replication: Do Not Replicate Messages

Calculation Timeout (sec): 10

Max Recovery Time (hr): 4

Add Prefix To Messages:

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. 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.

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 536)*.
- 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)*, and create an instance of the collector *(on page 662)*.

1. Access Configuration Hub *(on page 97)*.
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.

4. As needed, enter values in the other sections common to all collectors *(on page 669)*.
5. Restart the collector.

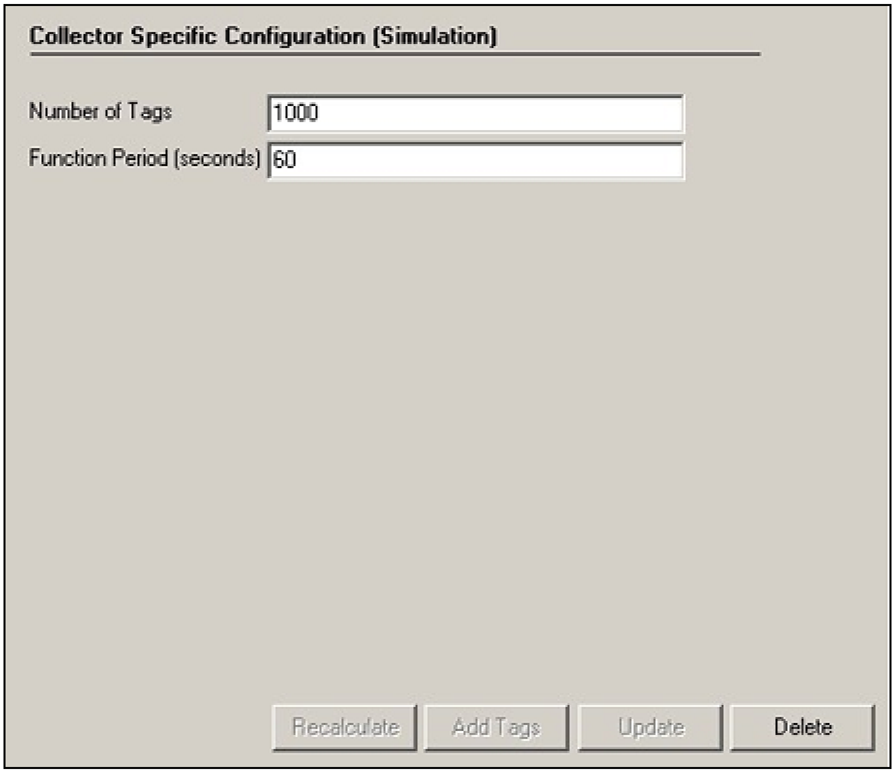
   The collector instance is configured.

Configure the Simulation Collector Using Historian Administrator

Install collectors *(on page)*, and create an instance of the collector *(on page 662)*.

1. Access Historian Administrator *(on page)*.
2. Select **Collectors**, and then select the Simulation collector instance that you want to configure.

   The fields specific to the collector instance appear.



Collector Specific Configuration (Simulation)

Number of Tags: 1000

Function Period (seconds): 60

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 669)*.
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.

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.

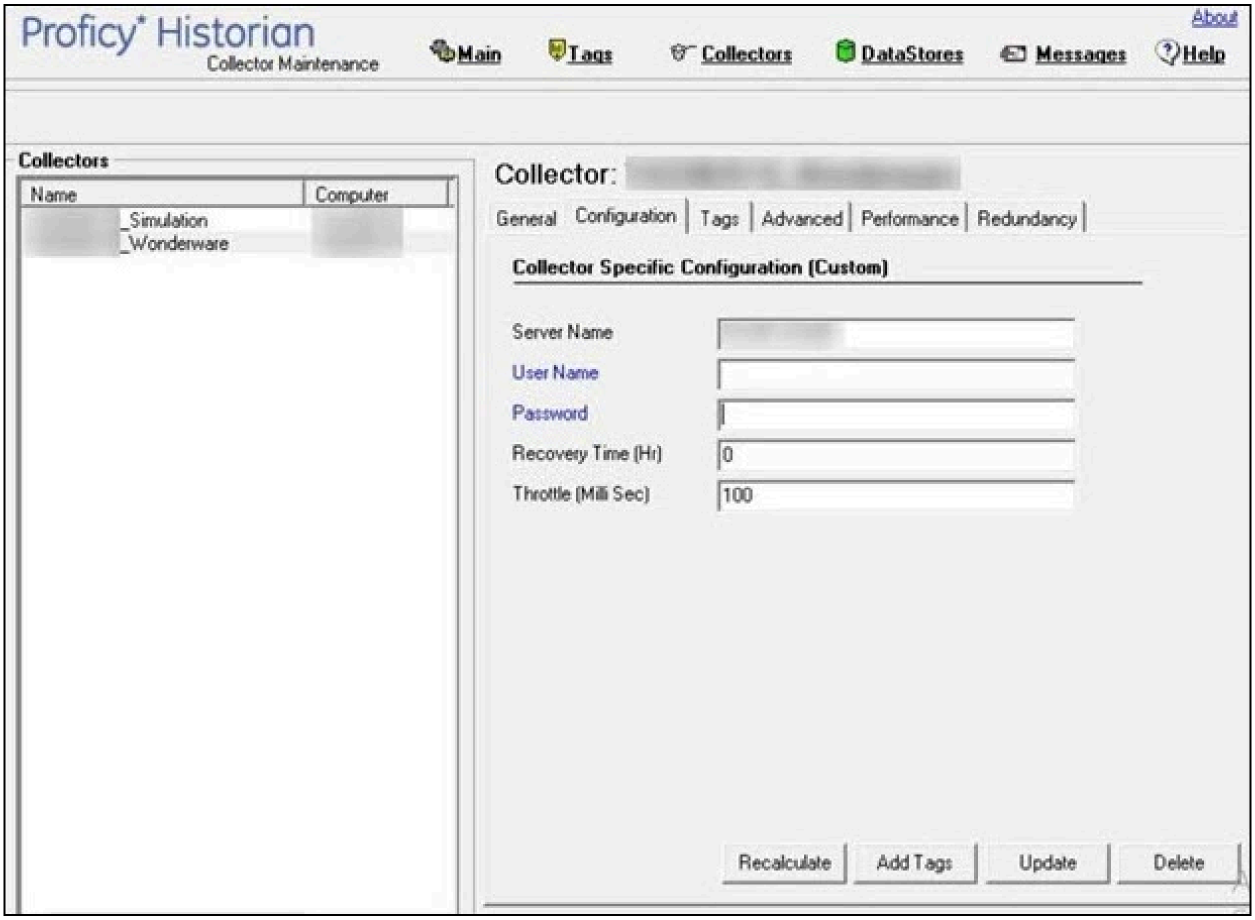
This collector supports collecting of analog, digital and string types of data from the Wonderware Historian Server. This collector supports a distributed model, where the Wonderware Historian Server, the Historian Data Collector, and GE Historian software are installed on different machines. Typically, 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.



Collectors

Name	Computer
[redacted]_Simulation	[redacted]
[redacted]_Wonderware	

Collector: [redacted]

Collector Specific Configuration (Custom)

Server Name: [redacted]
User Name:
Password:
Recovery Time (Hr): 0
Throttle (Milli Sec): 100

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

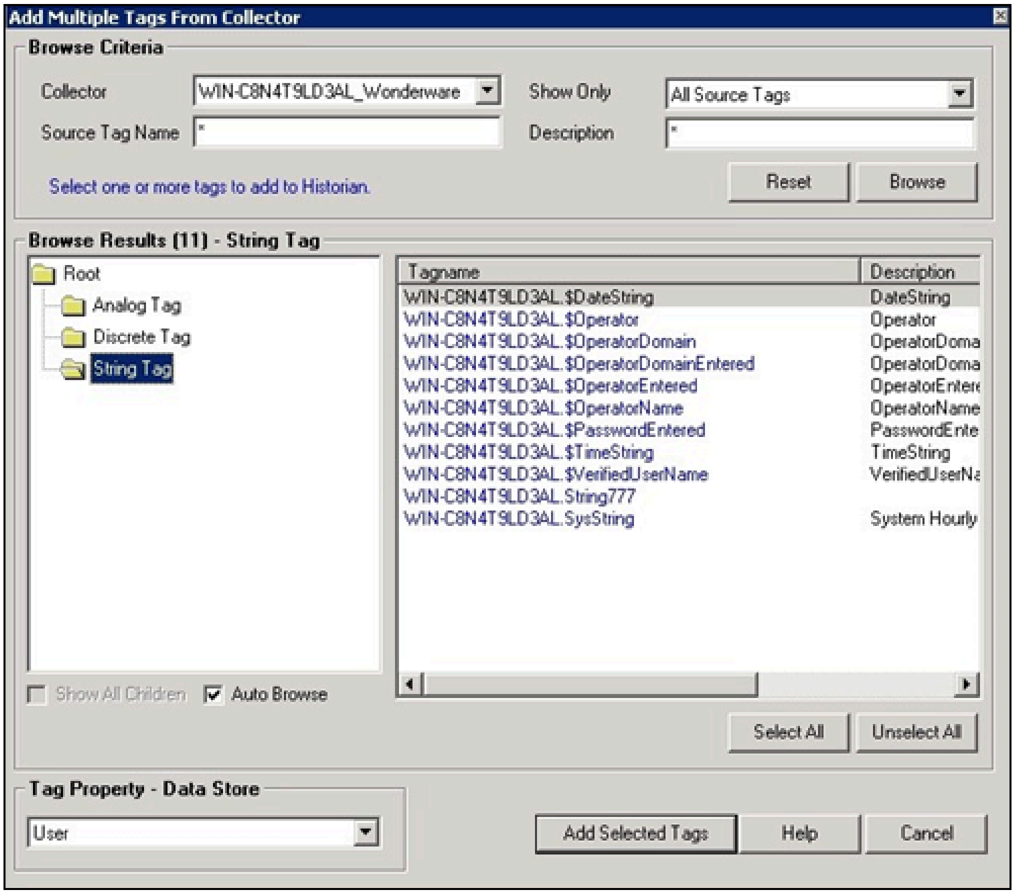
Feature	Capability
Minimum Poll Interval	100 ms (milliseconds)
Unsolicited Collection	Yes
Time stamp Resolution	1 ms
Floating Point Data	Yes
Integer Data	Yes
String Data	Yes
Python Expression Tags	No
Time Assigned	**Note:** You must set this field to Source as the Wonderware Co only supports unsolicited 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.



Collector: WIN-C8N4T9LD3AL_Wonderware

Show Only: All Source Tags

Source Tag Name: *

Description: *

Select one or more tags to add to Historian.

Tagname	Description
WIN-C8N4T9LD3AL.$DateString	DateString
WIN-C8N4T9LD3AL.$Operator	Operator
WIN-C8N4T9LD3AL.$OperatorDomain	OperatorDoma
WIN-C8N4T9LD3AL.$OperatorDomainEntered	OperatorDoma
WIN-C8N4T9LD3AL.$OperatorEntered	OperatorEnter
WIN-C8N4T9LD3AL.$OperatorName	OperatorName
WIN-C8N4T9LD3AL.$PasswordEntered	PasswordEnte
WIN-C8N4T9LD3AL.$TimeString	TimeString
WIN-C8N4T9LD3AL.$VerifiedUserName	VerifiedUserNa
WIN-C8N4T9LD3AL.String777	
WIN-C8N4T9LD3AL.SysString	System Hourly

□ Show All Children ☑ Auto Browse

User

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.

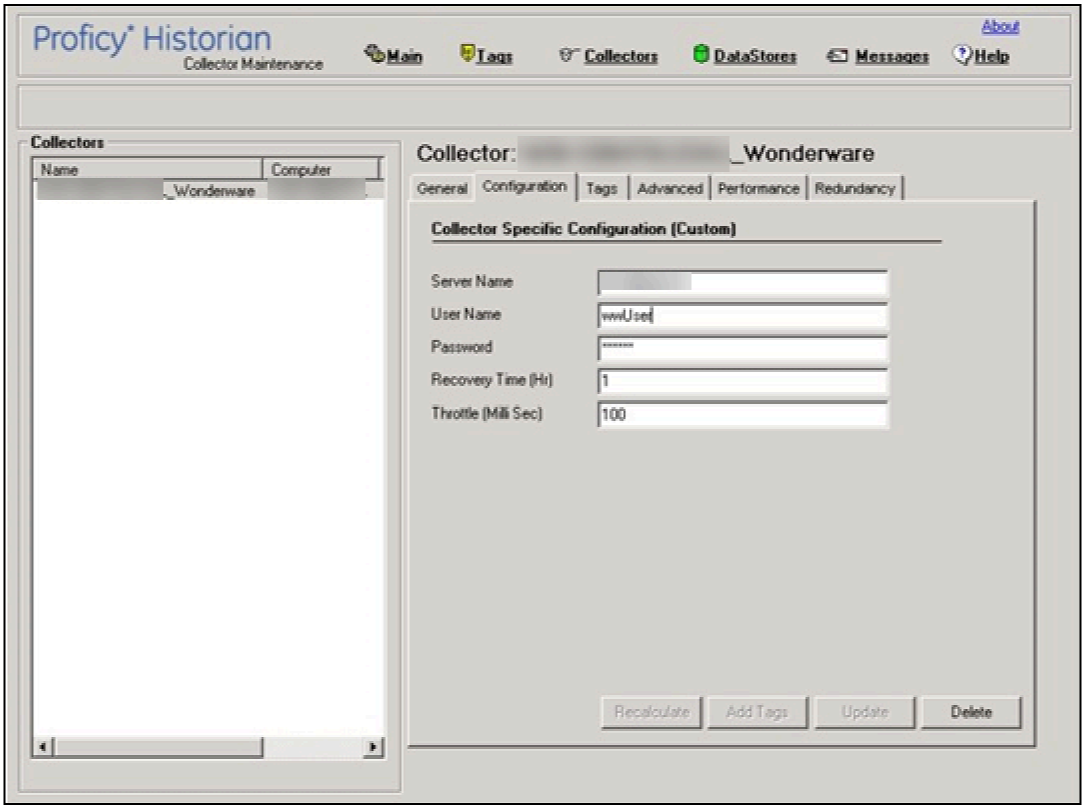
Supported Data Types for Wonderware Collector

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:



[redacted]_Wonderware [redacted]

Collector Specific Configuration (Custom)

Server Name: [redacted]
User Name: wwUser
Password: ******
Recovery Time (Hr): 1
Throttle (Milli Sec): 100

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. 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. **Note:** The default recovery time is 0 hours.
Throttle (Milliseconds)	Frequency of Wonderware data polling. 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. **Note:** 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 365)*.
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.

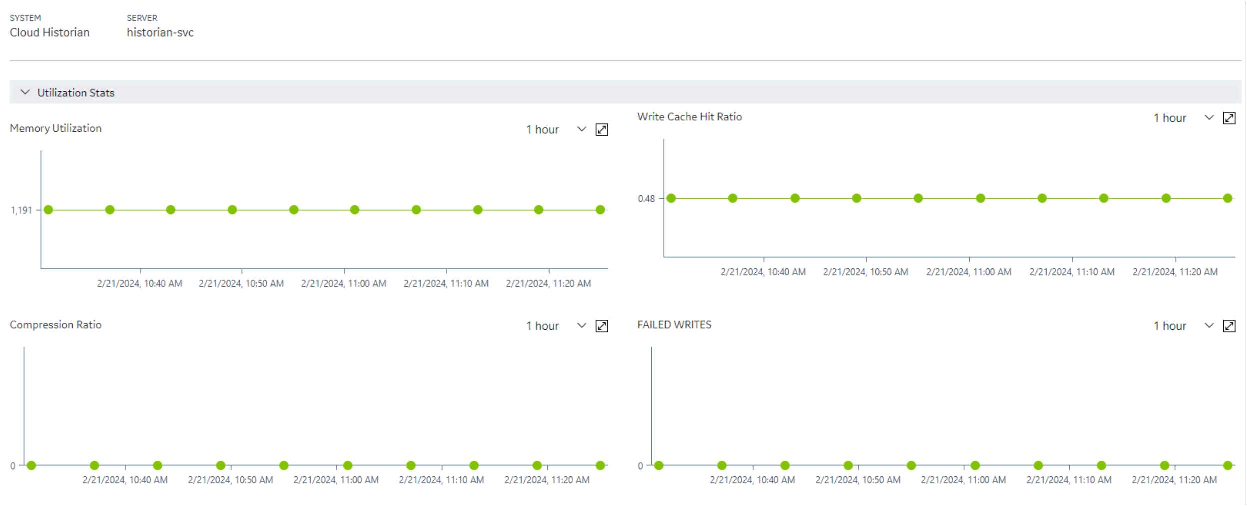
Chapter 9. Using Configuration Hub

Overview

About Configuration Hub

The Configuration Hub application allows you to manage the Historian systems, and their components.

> **Note:**
> You can only add Cloud Historian to Windows Configuration Hub when Windows Configuration Hub and Cloud Historian are pointed to same Proficy Authentication instance.



SYSTEM
Cloud Historian

SERVER
historian-svc

Advantages of Using Configuration Hub:

- **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.
- **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.

Limitations

- 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**.
- You cannot add Cloud Historian to Windows Configuration Hub when Windows Configuration Hub and Cloud Historian are pointed to different Proficy Authentication instances.

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)*.
3. Set up Historian on a Cloud system. 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. Specify the tags for data collection *(on page 384)*.

After you perform these initial steps, data is collected and stored in the Historian server. You can then retrieve and analyze the data.

Access Configuration Hub

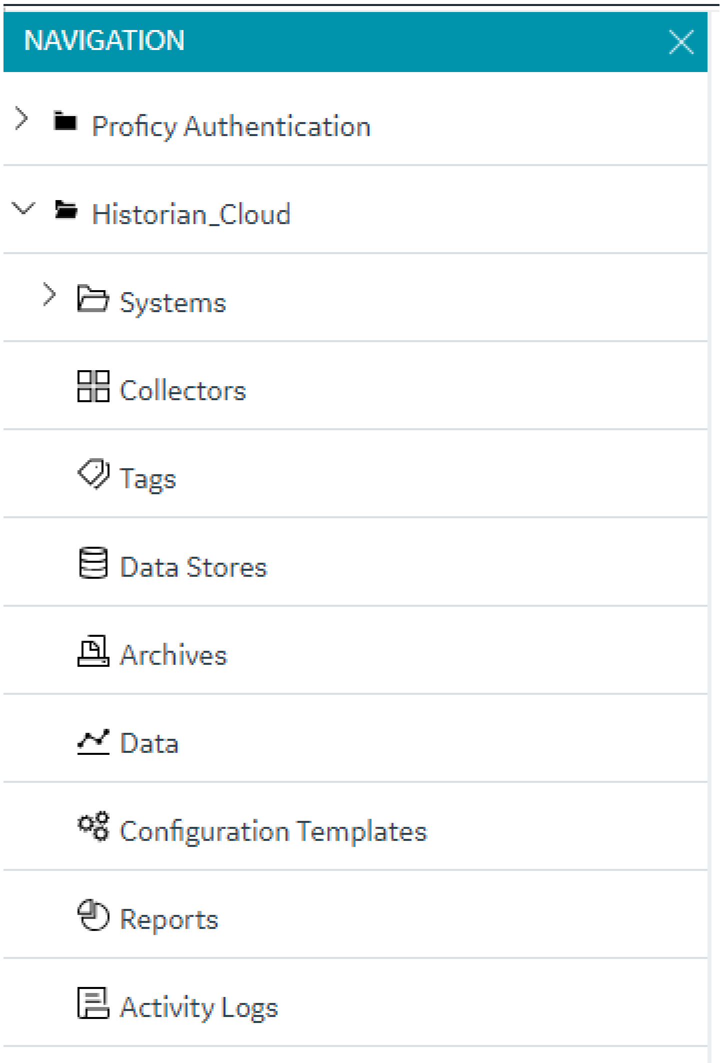
Perform the tasks outlined in About Setting up Configuration Hub *(on page 75)*.

1. When using Cloud Historian with Windows Configuration Hub (with web-based clients), or when using containerized Configuration Hub (deployed with Cloud Historian) access thru URL.
   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.

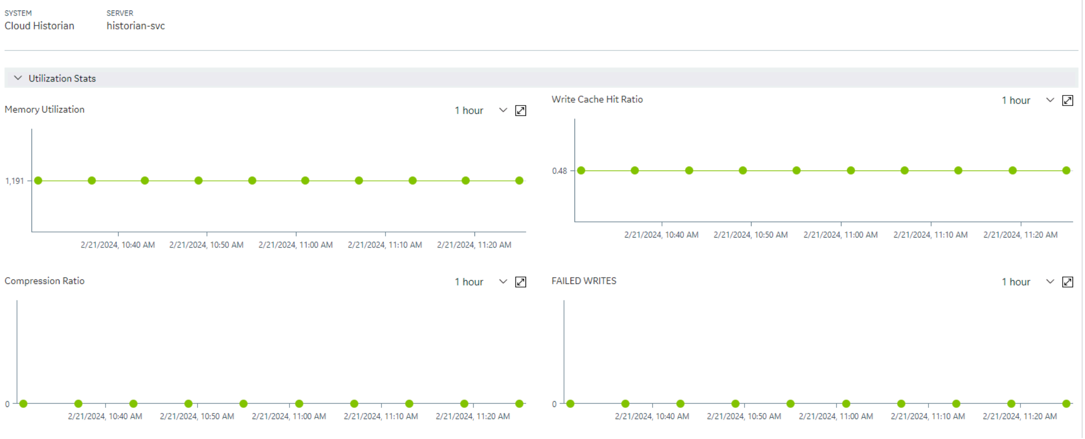
Note:
For instructions on setting up authentication, refer to https://www.ge.com/digital/documentation/confighub/version2024/t_authentication_setup.html

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. The default user name for Configuration Hub is: ihCloudHistAdmin (user name is case sensitive); the password was defined during deployment.
   The Configuration Hub application appears, displaying the following sections:

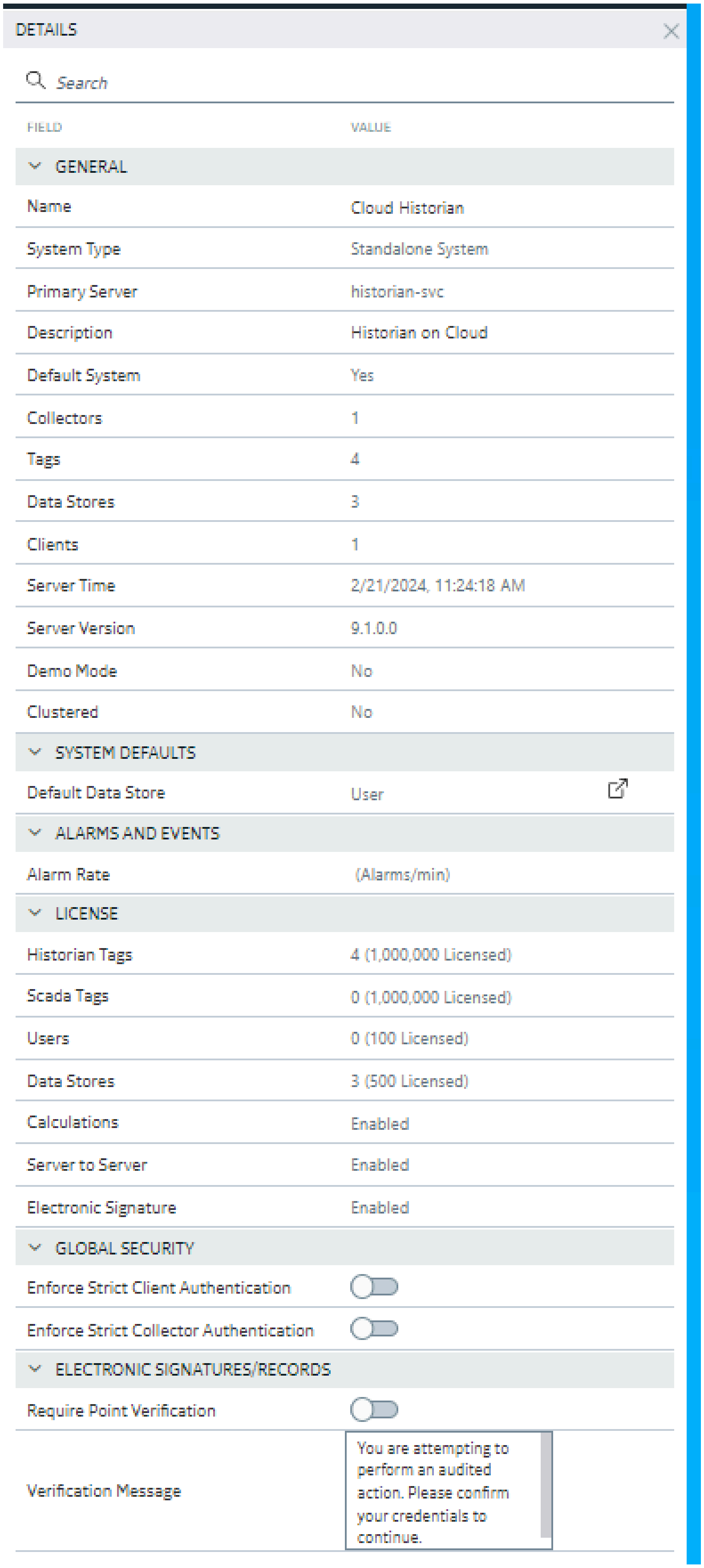
- **The Navigation section:** Contains a list of systems that you have added. In addition, it helps you navigate to the Collectors and Tags sections. You can also access Proficy Authentication to create users and groups.



- **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.


SYSTEM
Cloud Historian
SERVER
historian-svc
Utilization Stats
Memory Utilization
1 hour
1,191
2/21/2024, 10:40 AM
2/21/2024, 10:50 AM
2/21/2024, 11:00 AM
2/21/2024, 11:10 AM
2/21/2024, 11:20 AM
Write Cache Hit Ratio
1 hour
0.48
Compression Ratio
1 hour
0
FAILED WRITES
1 hour
0

- **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 using the **Details** section.



FIELD	VALUE
GENERAL	
Name	Cloud Historian
System Type	Standalone System
Primary Server	historian-svc
Description	Historian on Cloud
Default System	Yes
Collectors	1
Tags	4
Data Stores	3
Clients	1
Server Time	2/21/2024, 11:24:18 AM
Server Version	9.1.0.0
Demo Mode	No
Clustered	No
SYSTEM DEFAULTS	
Default Data Store	User
ALARMS AND EVENTS	
Alarm Rate	(Alarms/min)
LICENSE	
Historian Tags	4 (1,000,000 Licensed)
Scada Tags	0 (1,000,000 Licensed)
Users	0 (100 Licensed)
Data Stores	3 (500 Licensed)
Calculations	Enabled
Server to Server	Enabled
Electronic Signature	Enabled
GLOBAL SECURITY	
Enforce Strict Client Authentication	○
Enforce Strict Collector Authentication	○
ELECTRONIC SIGNATURES/RECORDS	
Require Point Verification	○
Verification Message	You are attempting to perform an audited action. Please confirm your credentials to continue.

Set up a stand-alone system.

Setting up a Cloud Historian System

About Setting up a Cloud Historian System

In a Cloud Historian Historian system, there is only one Historian server. This type of system is suitable for a small-scale Historian setup.

To set up a Cloud Historian system, you must first set up Configuration Hub *(on page 75)*.

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)*.

- **The Historian server:** You must deploy a Cloud Historian Server, and apply the license *(on page)*.
- **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-premise Historian server or a cloud destination such as Predix Time Series and Azure IoT hub.

  You must add a collector instance *(on page 556)* 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 384)* 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 595)*
- The iFIX collector *(on page 602)*
- The MQTT collector *(on page 349)*
- The ODBC collector *(on page 361)*
- The OPC Classic DA collector *(on page 375)*
- The OPC Classic HDA collector *(on page 393)*
- The OPC UA DA collector *(on page 407)*
- The OSI PI collector *(on page 644)*
- The OSI PI distributor *(on page 648)*
- The Python Collector *(on page 442)*
- The Server-to-Server collector *(on page 654)*
- The Server-to-Server distributor *(on page 658)*
- The Simulation collector *(on page 662)*
- The Wonderware collector *(on page 665)*

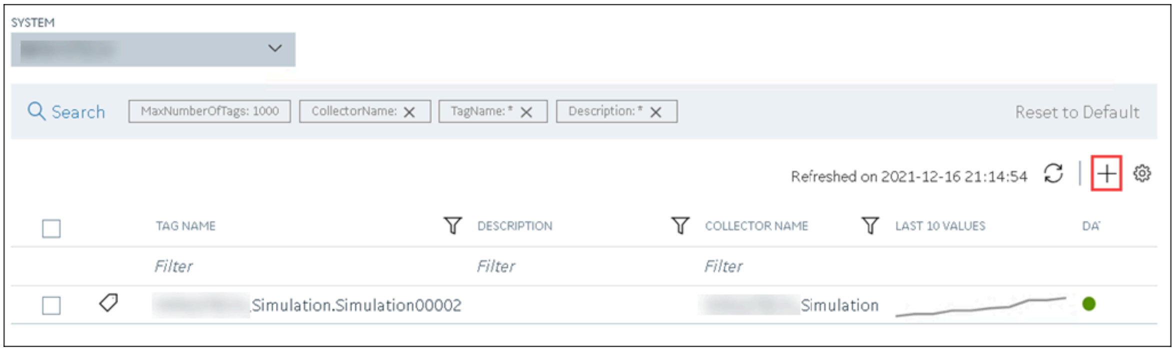
Add Tags for the Data Store Using Configuration Hub

- Add the collector instance *(on page 556)* 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 575)*.

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 589)*.

1. Access Configuration Hub *(on page 97)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
3. Select ＋.

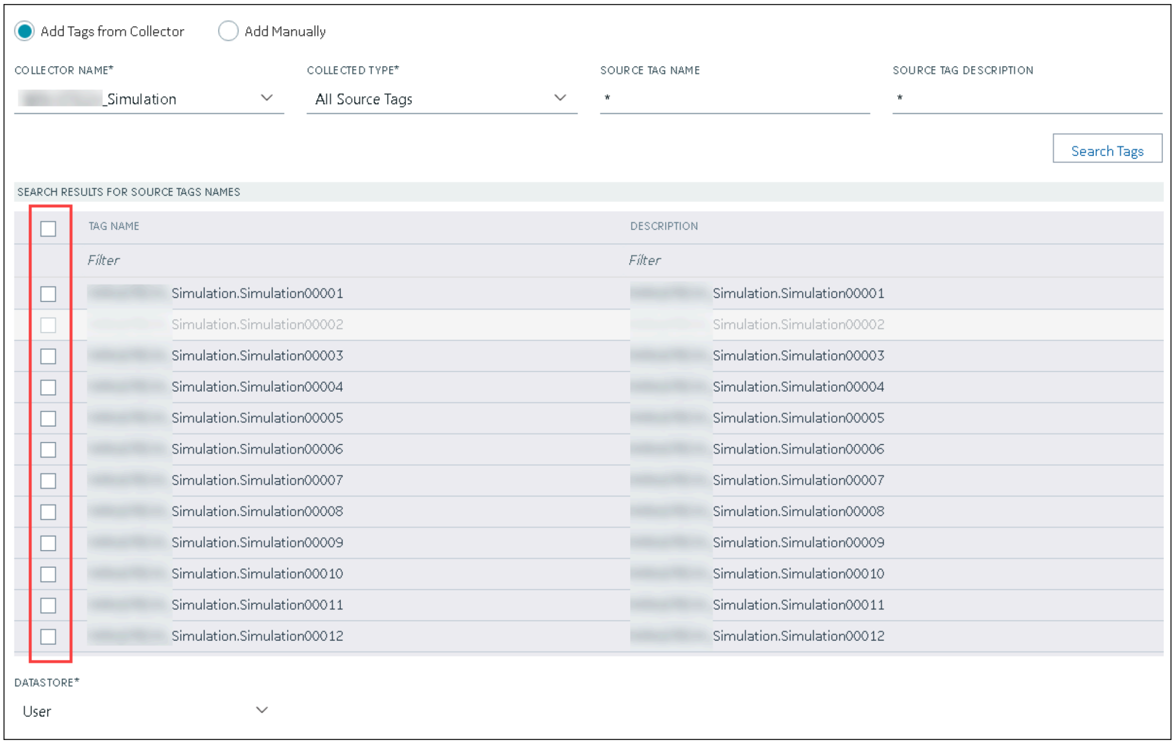


□		TAG NAME	DESCRIPTION	COLLECTOR NAME	LAST 10 VALUES	DA
		Filter	Filter	Filter		
□		[redacted] Simulation.Simulation00002		[redacted] Simulation		●

   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.



● Add Tags from Collector ○ Add Manually

COLLECTOR NAME*: [redacted]_Simulation

COLLECTED TYPE*: All Source Tags

SOURCE TAG NAME: *

SOURCE TAG DESCRIPTION: *

Search Tags

SEARCH RESULTS FOR SOURCE TAGS NAMES

□	TAG NAME	DESCRIPTION
	Filter	Filter
□	[redacted]Simulation.Simulation00001	[redacted]Simulation.Simulation00001
□	[redacted]Simulation.Simulation00002	[redacted]Simulation.Simulation00002
□	[redacted]Simulation.Simulation00003	[redacted]Simulation.Simulation00003
□	[redacted]Simulation.Simulation00004	[redacted]Simulation.Simulation00004
□	[redacted]Simulation.Simulation00005	[redacted]Simulation.Simulation00005
□	[redacted]Simulation.Simulation00006	[redacted]Simulation.Simulation00006
□	[redacted]Simulation.Simulation00007	[redacted]Simulation.Simulation00007
□	[redacted]Simulation.Simulation00008	[redacted]Simulation.Simulation00008
□	[redacted]Simulation.Simulation00009	[redacted]Simulation.Simulation00009
□	[redacted]Simulation.Simulation00010	[redacted]Simulation.Simulation00010
□	[redacted]Simulation.Simulation00011	[redacted]Simulation.Simulation00011
□	[redacted]Simulation.Simulation00012	[redacted]Simulation.Simulation00012

DATASTORE*: User

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.

Managing Historian Systems

Access a System

1. Access Configuration Hub *(on page 97)*.
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 3. 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: ◦ You can have only one default system in Configuration Hub. ◦ You cannot delete a default system. For instructions of setting a default system, refer to Set a Default System *(on page 573)*.
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. **Tip:** 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.

Table 4. 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

Field	Description
	adding a tag. For instructions on setting a data store as default, refer to Set a Default Data Store *(on page 584)*.

Table 5. The License Section

Field	Description
Historian Tags	The number of tags in the system (out of the total number of licensed tags). **Note:** 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). 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. 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.

Field	Description
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.
Electronic Signature	Indicates whether electronic signature is licensed on the software key.

Table 6. The Global Security Section

Field	Description
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)*.

Table 7. 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. **Note:**

Field	Description
	◦ The audit features are not dependent on this feature being enabled. Historian audits all user actions regardless of whether this option is enabled. 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. **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. ◦ **USERNAME**: This is populated with the user that is logged in to Configuration Hub and disabled. ◦ **PASSWORD**: The logged in user's password. ◦ **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. 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 dead-

Field	Description
	bands on individual tags in the **Tags** section to activate compression. 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. 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.
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. 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.

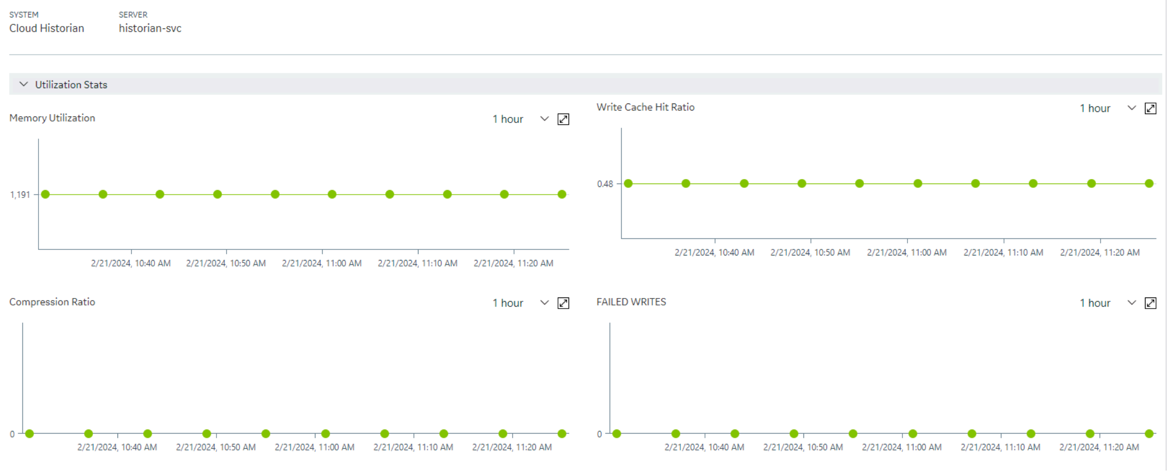
Field	Description
WRITE THREAD USAGE	The percentage of the write threads currently in use by the system. At the system level, this value is calculated as the average of the corresponding values of individual servers in the system.
OUT OF ORDER WRITE RATE	The number of out-of-order events per minute. 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. 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. 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.
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.

Field	Description
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 114)*.

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 97)*.
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: ◦ **Started** ◦ **Stopped** ◦ **Running** ◦ **Paused** ◦ **Unknown**
CONFIGURATION	The source of the tag configuration for the collector. Contains one of the following values: ◦ **HISTORIAN**: Indicates that tags are configured using Historian Administrator. ◦ **OFFLINE**: Indicates that tags are configured using an offline configuration *(on page)* 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.
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.

Column	Description
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 114)*.

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 721)*.

1. Access Configuration Hub *(on page 97)*.
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.

Column	Description
TAG ALIAS	Indicates whether the tag contains aliases, which are created when you rename the tag using an alias *(on page 779)*.

> **Tip:**
> You can show/hide/reorder columns in the table. For instructions, refer to Common Tasks in Configuration Hub *(on page 114)*.

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

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 97)*
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.
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 575)*.
- 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)* and then add them to the system *(on page)*.

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 97)*.
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 97)*.
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
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.
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.

Field	Description
NUMBER OF WRITE THREADS	Enter the number of write threads to use parallel writing of data.

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: ◦ **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.
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 1370)*.
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 97)*.
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 97)*.
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 97)*.
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 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 97)*.
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.
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 97)*.
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 8. 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: ◦ Running: Indicates that the data store is actively storing data. ◦ Stopped: Indicates that the data store is not storing data. This field is read-only.

Field	Description
Storage Type	The storage type of the data store, which can be one of the following values: ◦ **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. ◦ **SCADA Buffer:** Tags stored under SCADA buffer data store will store data for a specific duration of time based on license. 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 384)* and Add a Tag Manually *(on page 589)*.

Table 9. 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. Select one of the following options: ◦ **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**). ◦ **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**

Field	Description
	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 791)* 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. 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.

Table 10. 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.

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 11. Security

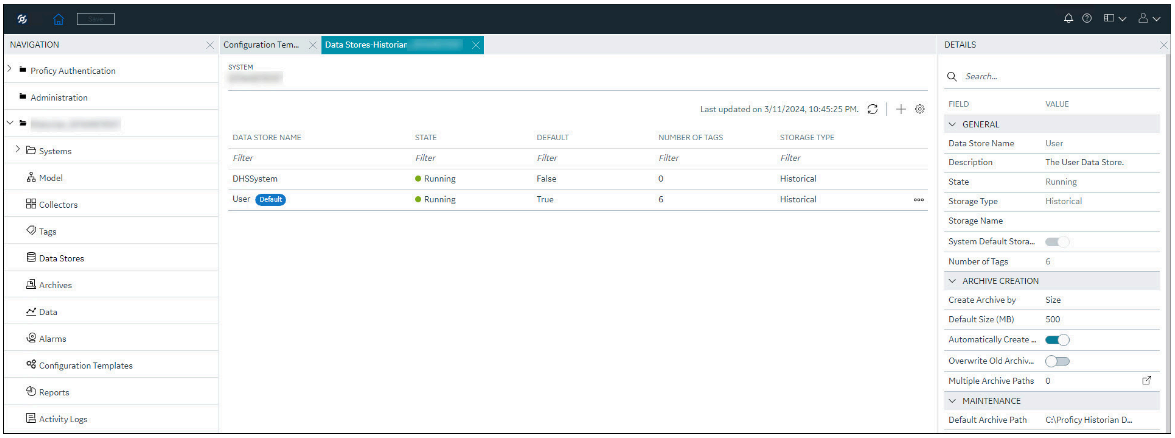
Field	Description
Data is Read-Only After (Hours)	The number of hours for data to be stored in a read/write archive. After the time lapses, that portion of the archive file is automatically made read-only. Incoming data values with timestamps prior to this time are rejected. A single archive file, therefore, 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.
Read Group	The Windows security group that can retrieve the tag data and plot it in a trend chart for the selected data store. 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.

Field	Description
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)*.

> **Note:**
> 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**.



Last updated on 3/11/2024, 10:45:25 PM.

DATA STORE NAME	STATE	DEFAULT	NUMBER OF TAGS	STORAGE TYPE
Filter	*Filter*	*Filter*	*Filter*	*Filter*
DHSSystem	● Running	False	0	Historical
User Default	● Running	True	6	Historical

FIELD	VALUE
GENERAL	
Data Store Name	User
Description	The User Data Store.
State	Running
Storage Type	Historical
Storage Name	
System Default Stora...	
Number of Tags	6
ARCHIVE CREATION	
Create Archive by	Size
Default Size (MB)	500
Automatically Create ...	
Overwrite Old Archiv...	
Multiple Archive Paths	0
MAINTENANCE	
Default Archive Path	C:\Proficy Historian D...

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 97)*.
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 12. 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.
Storage Type	The storage type of the data store, which can be one of the following values: ◦ **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. ◦ **SCADA Buffer:** Tags stored under SCADA buffer data store will store data for a specific duration of time based on license. 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 384)* and Add a Tag Manually *(on page 589)*.

Table 13. Archive Creation

Field	Description
Automatically Create Archives	Indicates whether you want to create a new archive *(on page 788)* 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.

Field	Description
Overwrite Old Archives	Indicates whether you want to overwrite an old archive file when a new one is created. 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.
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. Select one of the following options: ◦ **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**). ◦ **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.

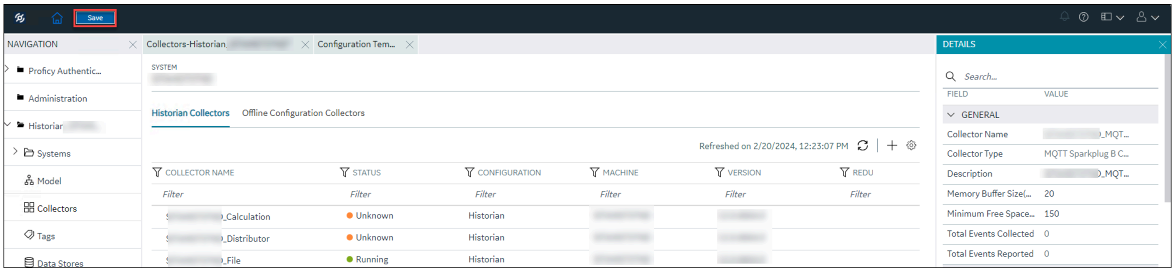
Table 14. Maintenance

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. 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 15. 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**.



Refreshed on 2/20/2024, 12:23:07 PM

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSION	REDU
Filter	Filter	Filter	Filter	Filter	Filter
[redacted]_Calculation	● Unknown	Historian	[redacted]	[redacted]	
[redacted]_Distributor	● Unknown	Historian	[redacted]	[redacted]	
[redacted]_File	● Running	Historian	[redacted]	[redacted]	

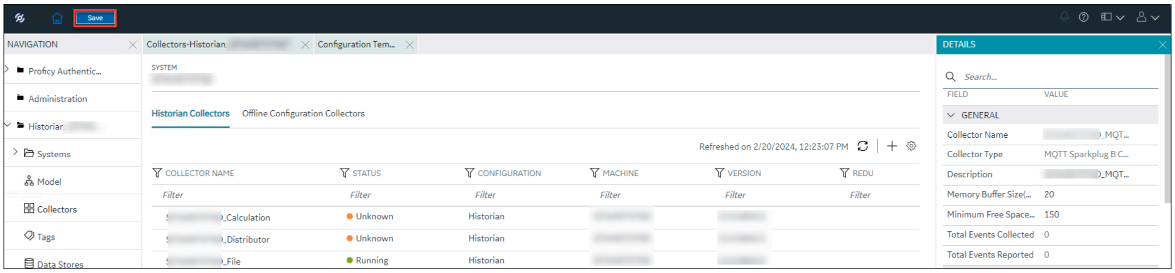
FIELD	VALUE
GENERAL	
Collector Name	[redacted]_MQT...
Collector Type	MQTT Sparkplug B C...
Description	[redacted]_MQT...
Memory Buffer Size(...	20
Minimum Free Space...	150
Total Events Collected	0
Total Events Reported	0

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 97)*.
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**.



Refreshed on 2/20/2024, 12:23:07 PM

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSION	REDU
Filter	Filter	Filter	Filter	Filter	Filter
[redacted]_Calculation	● Unknown	Historian	[redacted]	[redacted]	
[redacted]_Distributor	● Unknown	Historian	[redacted]	[redacted]	
[redacted]_File	● Running	Historian	[redacted]	[redacted]	

FIELD	VALUE
GENERAL	
Collector Name	[redacted]_MQT...
Collector Type	MQTT Sparkplug B C...
Description	[redacted]_MQT...
Memory Buffer Size(...	20
Minimum Free Space...	150
Total Events Collected	0
Total Events Reported	0

The data store is set as default.

Access the Archives in a Data Store

1. Access Configuration Hub *(on page 97)*.
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.

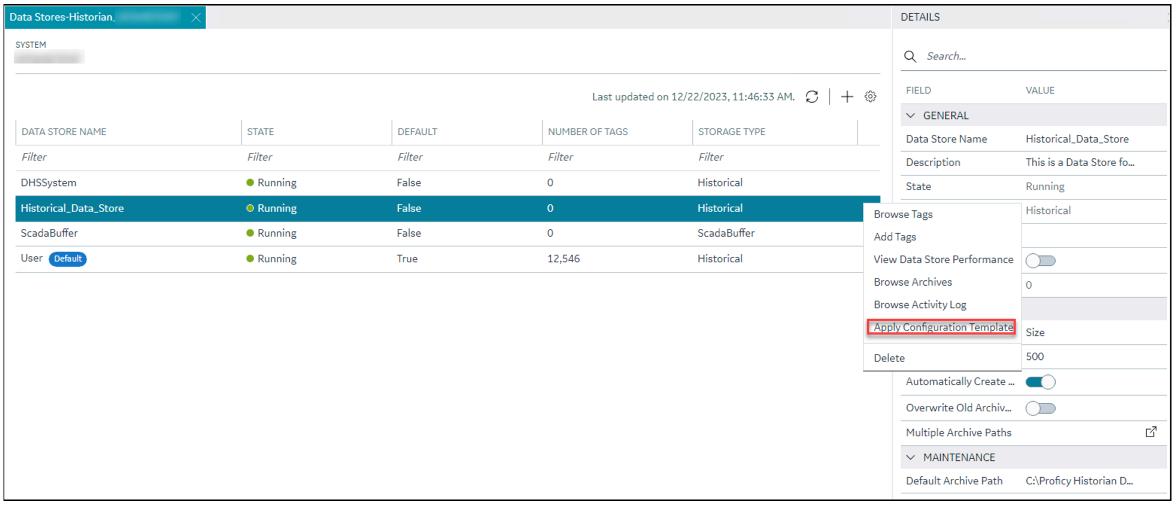
Apply the Configuration Template to a Data Store

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 575)*.
- Ensure that you have a configuration template for data stores *(on page 823)*.

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 97)*.
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**.



Data Stores-Historian_

SYSTEM

Last updated on 12/22/2023, 11:46:33 AM.

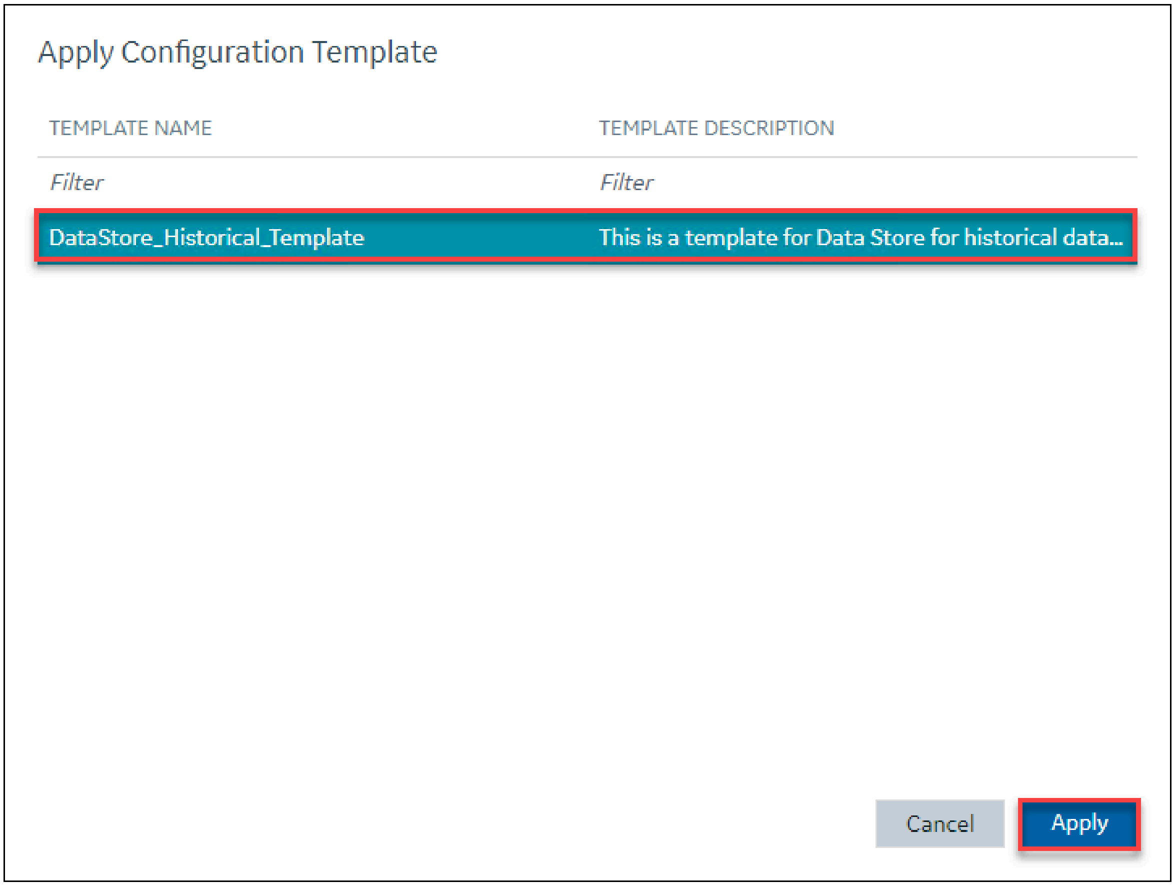
DATA STORE NAME	STATE	DEFAULT	NUMBER OF TAGS	STORAGE TYPE
Filter	*Filter*	*Filter*	*Filter*	*Filter*
DHSSystem	● Running	False	0	Historical
Historical_Data_Store	● Running	False	0	Historical
ScadaBuffer	● Running	False	0	ScadaBuffer
User Default	● Running	True	12,546	Historical

DETAILS

Search...

FIELD	VALUE
GENERAL	
Data Store Name	Historical_Data_Store
Description	This is a Data Store fo...
State	Running
	Historical
	○
	0
	Size
	500
Automatically Create ...	●
Overwrite Old Archiv...	○
Multiple Archive Paths	
MAINTENANCE	
Default Archive Path	C:\Proficy Historian D...

The **Apply Configuration Template** window appears, listing the available templates.



Apply Configuration Template

TEMPLATE NAME	TEMPLATE DESCRIPTION
Filter	*Filter*
DataStore_Historical_Template	This is a template for Data Store for historical data...

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.

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 97)*.
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 97)*.
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.

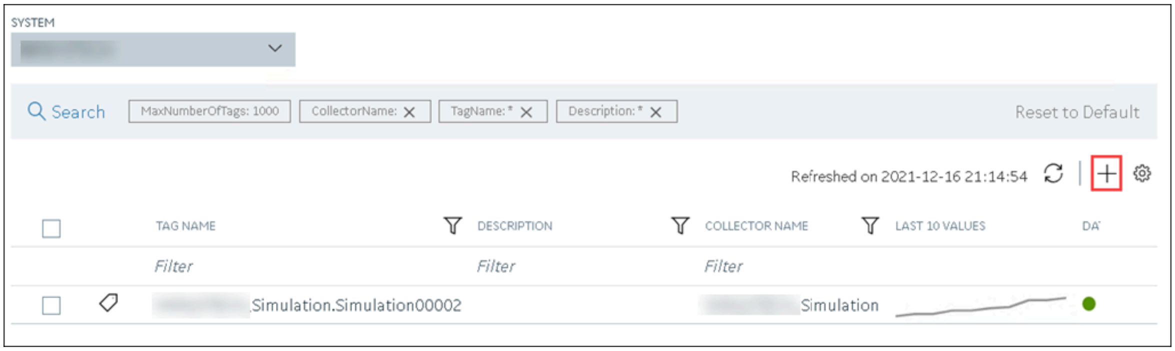
Add Tags for the Data Store Using Configuration Hub

- Add the collector instance *(on page 556)* 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 575)*.

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 589)*.

1. Access Configuration Hub *(on page 97)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
3. Select +.

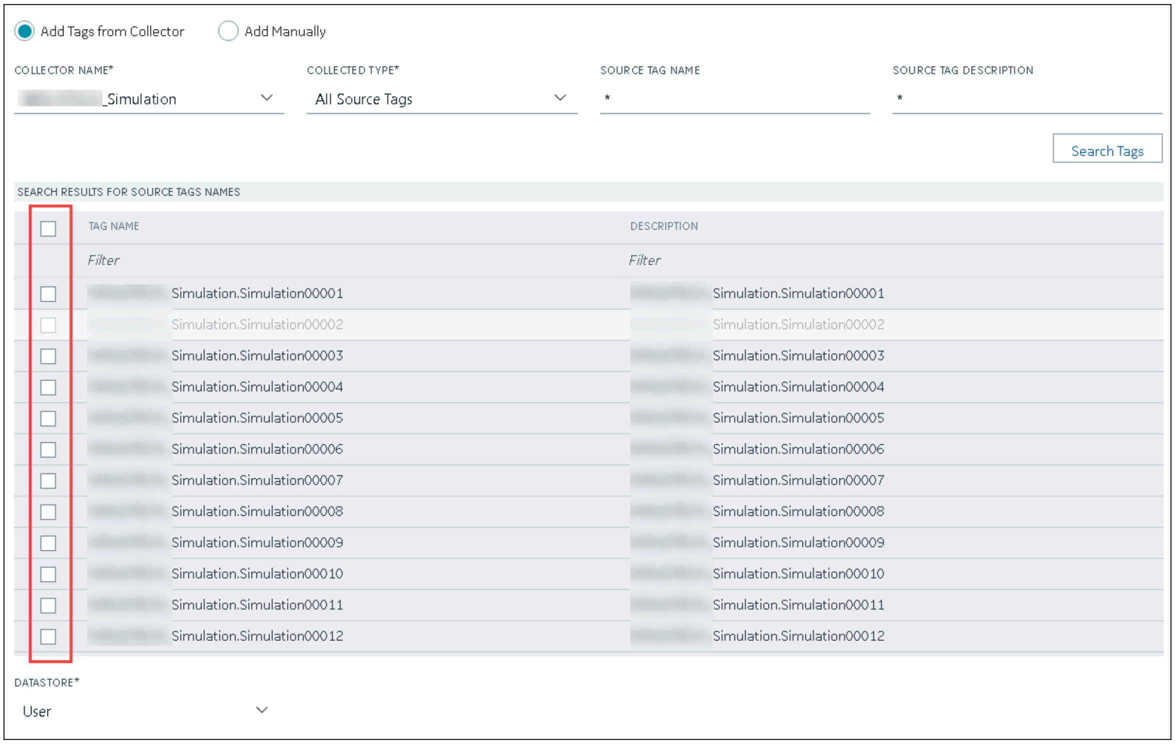


The **Add Tag-<system name>** page appears. The **Add Tags from Collector** option is selected by default.

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.



● Add Tags from Collector ○ Add Manually

COLLECTOR NAME*: [redacted]_Simulation

COLLECTED TYPE*: All Source Tags

SOURCE TAG NAME: *

SOURCE TAG DESCRIPTION: *

Search Tags

SEARCH RESULTS FOR SOURCE TAGS NAMES

□	TAG NAME	DESCRIPTION
	Filter	Filter
□	[redacted]Simulation.Simulation00001	[redacted]Simulation.Simulation00001
□	[redacted]Simulation.Simulation00002	[redacted]Simulation.Simulation00002
□	[redacted]Simulation.Simulation00003	[redacted]Simulation.Simulation00003
□	[redacted]Simulation.Simulation00004	[redacted]Simulation.Simulation00004
□	[redacted]Simulation.Simulation00005	[redacted]Simulation.Simulation00005
□	[redacted]Simulation.Simulation00006	[redacted]Simulation.Simulation00006
□	[redacted]Simulation.Simulation00007	[redacted]Simulation.Simulation00007
□	[redacted]Simulation.Simulation00008	[redacted]Simulation.Simulation00008
□	[redacted]Simulation.Simulation00009	[redacted]Simulation.Simulation00009
□	[redacted]Simulation.Simulation00010	[redacted]Simulation.Simulation00010
□	[redacted]Simulation.Simulation00011	[redacted]Simulation.Simulation00011
□	[redacted]Simulation.Simulation00012	[redacted]Simulation.Simulation00012

DATASTORE*: User

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.

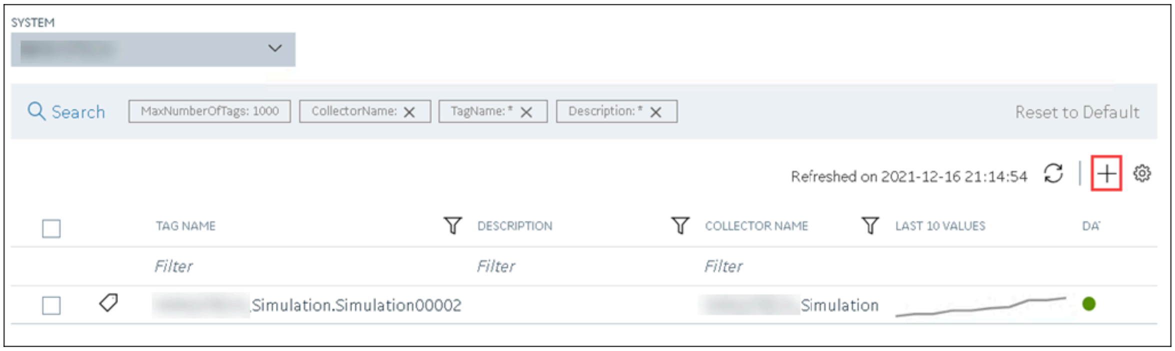
As needed, configure each tag by providing values for the tag properties. For information on the delta query modes, refer to Counter Delta Queries.

Add a Tag Manually

- Add the collector instance *(on page 556)* 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 575)*.

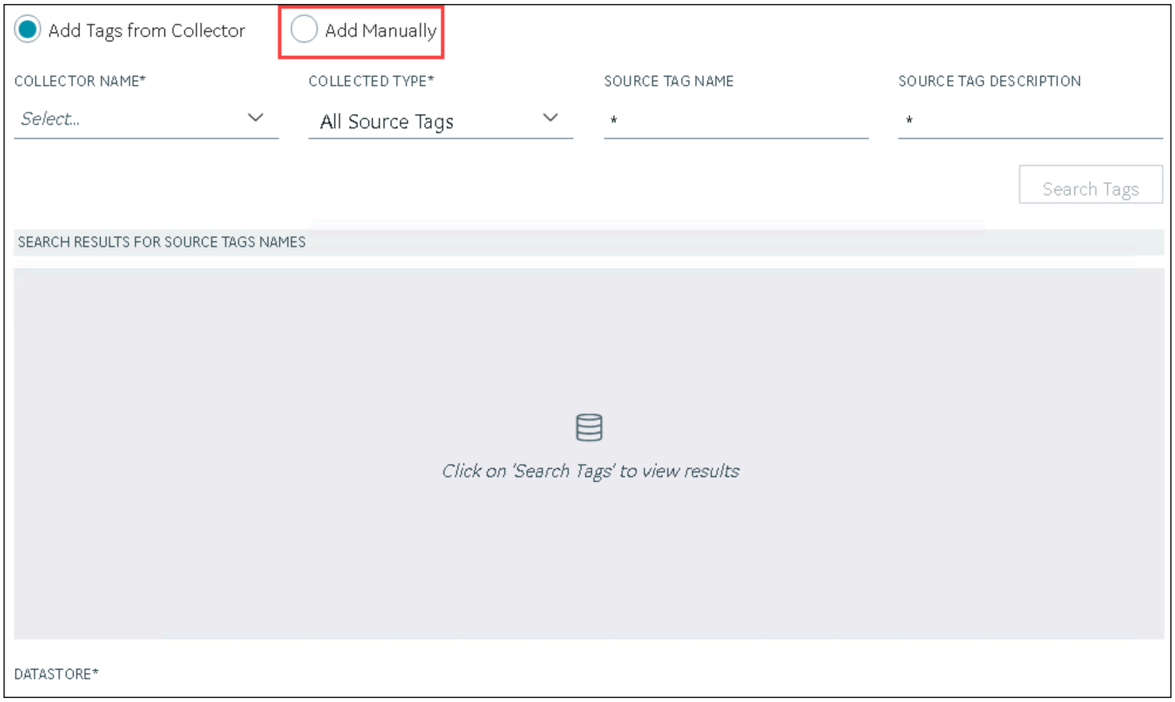
After you create a collector instance, you specify which tags from the source must be used for data collection *(on page 384)*. In addition, if you want to use the same tag twice (say, with a different collection interval or collector compression settings), you can add the tag manually. You can also create a calculation tag or a tag to store the values imported using the Excel Add-in.

1. Access Configuration Hub *(on page 97)*.
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. The value that you enter: ◦ Must begin with a letter or a number. ◦ Can contain up to 256 characters. ◦ Can include any of the following special characters: /!|#{}%$-_ ◦ 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. **Important:** If you select an unsupported data type, you may receive incorrect data or even lose data. If you select **Multi-Field**, the **USER-DEFINED TYPE NAME** field appears, and the **ENUMERATED SET** and **ARRAY TAG** fields are disabled. 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. 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)* 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)* 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**. For information on array tags, refer to About Array Tags *(on page 741)*.
TIME RESOLUTION	Select the time resolution for the tag. A value is required. 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 97)*.
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. 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	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 cal-

Field	Description
	culated 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). 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. You may also want to enable the **Overwrite Old Archives** option if you have limited disk capacity. Enabling this option, 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.
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. 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.

Field	Description
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 97)*.
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

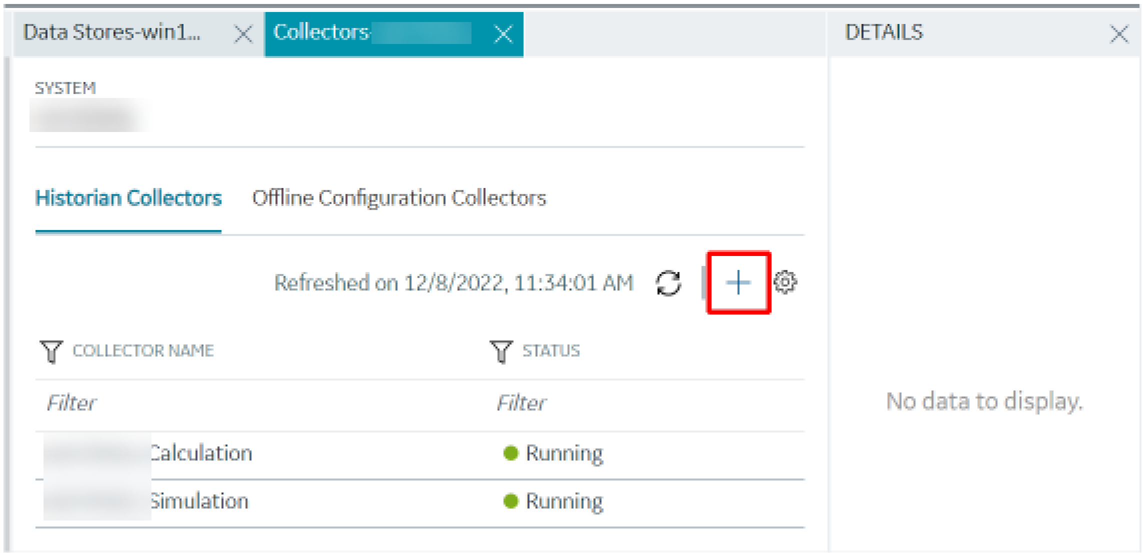
1. Deploy Proficy Historian for AWS. *(on page 43)*
2. Install collectors *(on page 90)*. You can install them on-premises or on a VPC (which can be different from the one on which Proficy Historian for AWS is deployed).
3. Enable TLS encryption for Collectors Connecting to Cloud Historian *(on page 150)*.

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-premise 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 1335)*, which does not require you to install Web-based Clients.

1. Access Configuration Hub *(on page 97)*.
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. Select **Next**. The **Collector Initiation** section appears.

9. If needed, modify the value in the **COLLECTOR NAME** field.

   The value that you enter:

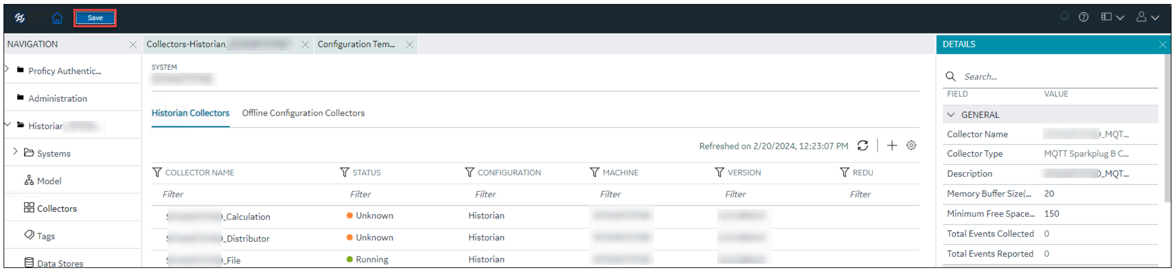
   - Must be unique.
   - Must not exceed 15 characters.
   - Must not contain a space.
   - Must not contain special characters except a hyphen, period, and an underscore.
10. In the **RUNNING MODE** field, select one of the following options.
    - **Service - Local System Account**: Select this option 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.
12. 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. If you want to disable automatic calculation of the tag, set the value of this field to 0.

13. If needed, enter values in the other sections *(on page 669)*.
14. In the upper-left corner of the page, select **Save**.



SYSTEM [redacted]

Historian Collectors | Offline Configuration Collectors

Refreshed on 2/20/2024, 12:23:07 PM

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSION	REDU
Filter	Filter	Filter	Filter	Filter	Filter
[redacted]_Calculation	● Unknown	Historian	[redacted]	[redacted]	
[redacted]_Distributor	● Unknown	Historian	[redacted]	[redacted]	
[redacted]_File	● Running	Historian	[redacted]	[redacted]	

FIELD	VALUE
GENERAL	
Collector Name	[redacted]_MQT...
Collector Type	MQTT Sparkplug B C...
Description	[redacted]_MQT...
Memory Buffer Size(...	20
Minimum Free Space...	150
Total Events Collected	0
Total Events Reported	0

The changes to the collector instance are saved.

15. If needed, restart the collector.

Specify the tags *(on page 384)* whose data you want to collect using the collector.

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. 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. • 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. • 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 running as a service and is not secured. 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. • 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. • 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 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.

iFIX Running Mode	Recommended Configuration for the iFIX Collector	Collector Behaviour After You Add the Collector Instance
		• 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. • 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. • 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 service, an error message appears while adding a collector instance. However, the instance is configured successfully. • 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.

iFIX Running Mode	Recommended Configuration for the iFIX Collector	Collector Behaviour After You Add the Collector Instance
		• You must start the collector manually for the first time using the shortcut. It will then connect to the Historian server. • 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. • 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. • After you start iFIX, you must start the collector manually for the first time using the shortcut. It will

iFIX Running Mode	Recommended Configuration for the iFIX Collector	Collector Behaviour After You Add the Collector Instance
		then connect to the Historian server. • Once connected to server, you can start/stop it at a command prompt.

Add and Configure an iFIX Collector

1. Deploy Proficy Historian for AWS. *(on page 43)*
2. Install collectors *(on page 90)*. You can install them on-premises or on a VPC (which can be different from the one on which Proficy Historian for AWS is deployed).
3. Enable TLS encryption for Collectors Connecting to Cloud Historian *(on page 150)*.
4. Ensure that iFIX is running in a Windows-service mode. For more information, refer to About Adding an iFIX Collector Instance *(on page 300)*.

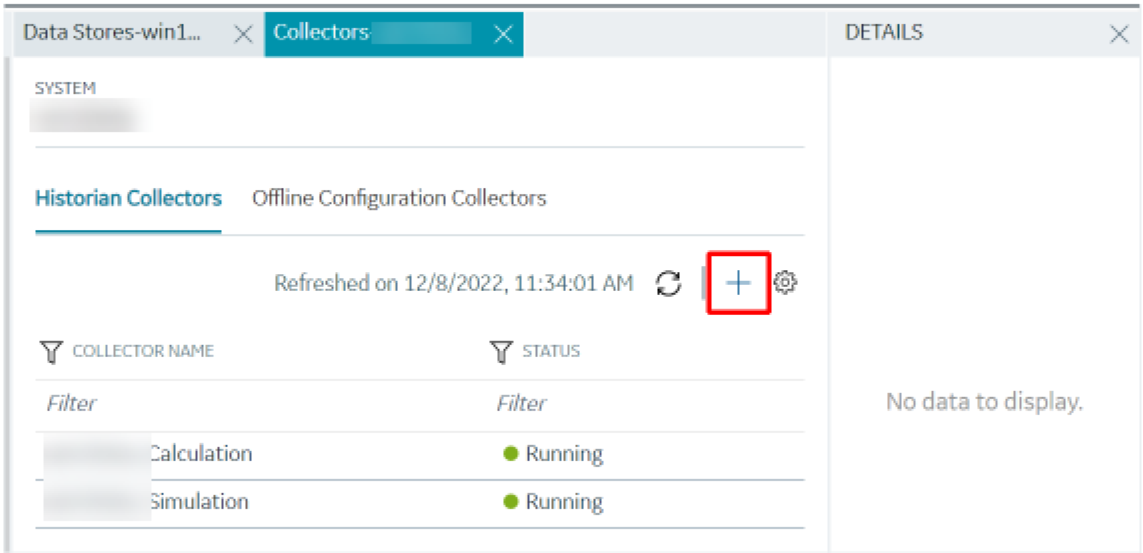
The iFIX collectors collect data from iFIX and store it in the Historian server. They include:

- 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 1335)*, which does not require you to install Web-based Clients.

1. Access Configuration Hub *(on page 97)*.
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 179)*.
   - **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 698)*.
   - **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 151)*.
     - AWS cloud. For more information, refer to AWS Cloud *(on page 158)*.
     - Google cloud. For more information, refer to Google Cloud *(on page 172)*.

b. If you need to send data to an on-premise/Cloud Historian server, select **Historian Server**.
   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.

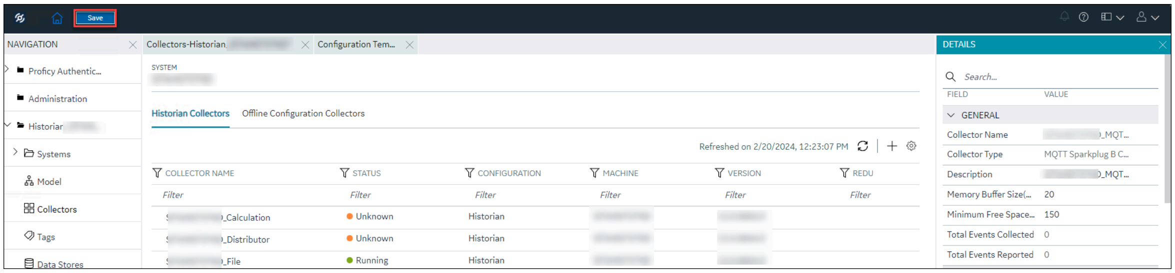
      If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:
        - iH Security Admins
        - iH Collector Admins
        - iH Tag Admins

    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 300)*.
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. 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 informa-

Field	Description
	tion, refer to the iFIX product documentation on iFIX networking. **Note:** 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. 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 384)*. **Note:** 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 305)* for more information.

16. As needed, enter values in the other sections common o all collectors *(on page 669)*.
17. In the upper-left corner of the page, select **Save**.



COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSION	REDU
Filter	Filter	Filter	Filter	Filter	Filter
[redacted]_Calculation	● Unknown	Historian	[redacted]	[redacted]	
[redacted]_Distributor	● Unknown	Historian	[redacted]	[redacted]	
[redacted]_File	● Running	Historian	[redacted]	[redacted]	

FIELD	VALUE
GENERAL	
Collector Name	[redacted]_MQT...
Collector Type	MQTT Sparkplug B C...
Description	[redacted]_MQT...
Memory Buffer Size(...	20
Minimum Free Space...	150
Total Events Collected	0
Total Events Reported	0

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 384)*.
- 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)*. 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. Deploy Proficy Historian for AWS. *(on page 43)*
2. Install collectors *(on page 90)*. You can install them on-premises or on a VPC (which can be different from the one on which Proficy Historian for AWS is deployed).
3. Enable TLS encryption for Collectors Connecting to Cloud Historian *(on page 150)*.
4. Ensure that you have an MQTT broker.

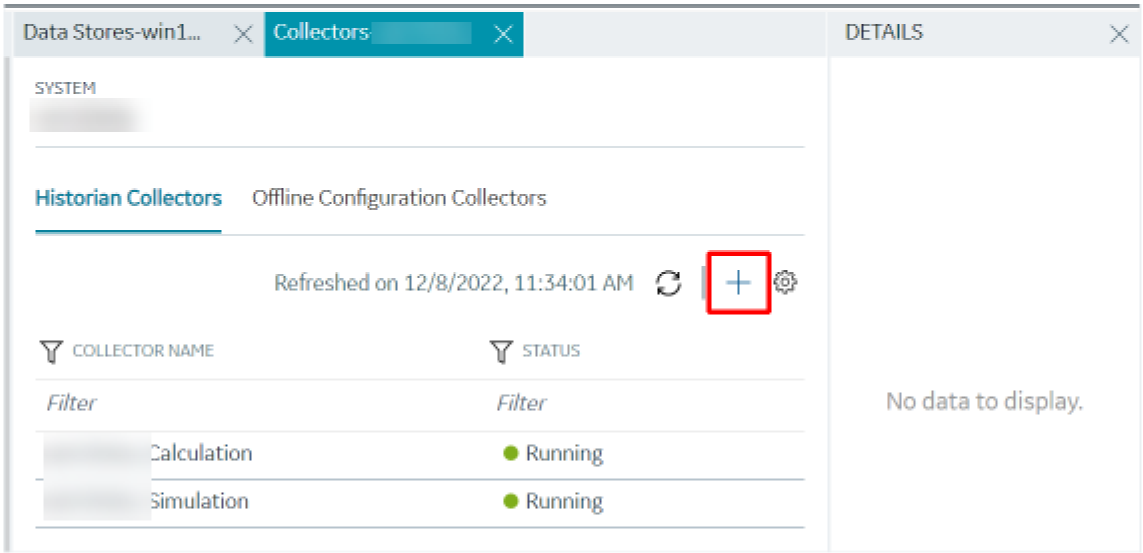
> **Note:**
> We have tested with the MQTT brokers Mosquitto 2.0.15 and HiveMQ-4.2.1. You can, however, use other MQTT brokers as well.

5. 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.
6. 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 346)*.

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 1335)*, which does not require you to install Web-based Clients.

1. Access Configuration Hub *(on page 97)*.
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 +.

COLLECTOR NAME	STATUS
Filter	*Filter*
[redacted] Calculation	● Running
[redacted] Simulation	● Running

   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.

Field	Description
TOPIC	Enter the MQTT topic from which you want to collect data. A value is required. You can enter multiple topics separated by commas. 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` where: ◦ `namespace` is the Sparkplug version. Enter `spBv1.0`. ◦ `group_id` is the ID of the group of nodes from which you want to collect data. ◦ `message_type` is the message type from which you want to collect data. The collector processes data only from NDATA and DDATA message types. ◦ `edge_node_id` is used to identify the MQTT EoN node within the infrastructure. ◦ `device_id` a device attached to the MQTT EoN node either physically or logically. 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.
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.

Field	Description
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. ◦ **QoS 0**: Indicates that the message is delivered at most once or it is not delivered at all. ◦ **QoS 1**: Indicates that the message is always delivered at least once. ◦ **QoS 2**: Indicates that the message is delivered once. 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: ◦ **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. ◦ **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. 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. This field is applicable only for MQTT V5 and only if you set the **CLEAN SESSION** field to **False**.

Field	Description
CONTENT TYPE	Select the format that you want to use for the payload: ◦ **JSON**: Select this option if you want to use the KairosDB format. ◦ **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-premise 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 179)*.
      - **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 698)*.
      - **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 151)*.
        - AWS cloud. For more information, refer to AWS Cloud *(on page 158)*.
        - Google cloud. For more information, refer to Google Cloud *(on page 172)*.

   b. If you need to send data to an on-premise/Cloud Historian server, select **Historian Server**.

      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.

    INSTANCE CONFIGURATION

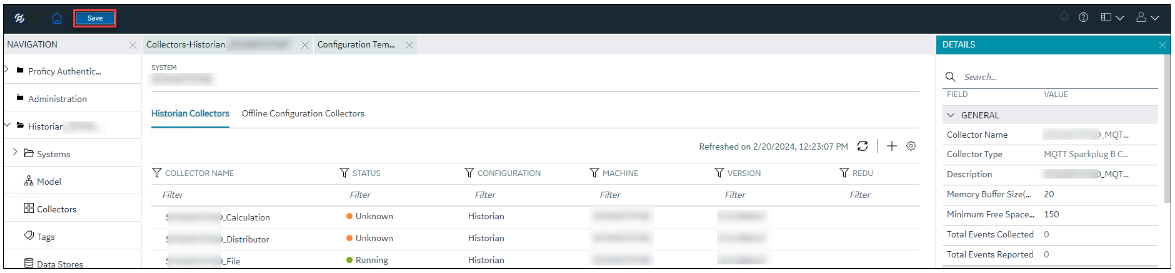
Field	Description
MQTT Broker Address	Enter the host name of the MQTT broker using which you want to collect data. A value is required.
MQTT Broker Topic	Enter the MQTT topic from which you want to collect data. A value is required. You can enter multiple topics separated by commas.

Field	Description
	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` where: ◦ `namespace` is the Sparkplug version. Enter `spBv1.0`. ◦ `group_id` is the ID of the group of nodes from which you want to collect data. ◦ `message_type` is the message type from which you want to collect data. The collector processes data only from NDATA and DDATA message types. ◦ `edge_node_id` is used to identify the MQTT EoN node within the infrastructure. ◦ `device_id` a device attached to the MQTT EoN node either physically or logically. 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.

Field	Description
Requested Quality Of Service (QoS) Level	Select the quality of service that you want to use while collecting data from an MQTT broker. ◦ **QoS 0**: Indicates that the message is delivered at most once or it is not delivered at all. ◦ **QoS 1**: Indicates that the message is always delivered at least once. ◦ **QoS 2**: Indicates that the message is delivered once. 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: ◦ **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. ◦ **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. 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. 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:

Field	Description
	◦ **JSON**: Select this option if you want to use the KairosDB format. ◦ **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 669)*.
16. In the upper-left corner of the page, select **Save**.



Refreshed on 2/20/2024, 12:23:07 PM

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSION	REDU
Filter	Filter	Filter	Filter	Filter	Filter
[redacted]_Calculation	● Unknown	Historian	[redacted]	[redacted]	
[redacted]_Distributor	● Unknown	Historian	[redacted]	[redacted]	
[redacted]_File	● Running	Historian	[redacted]	[redacted]	

FIELD	VALUE
GENERAL	
Collector Name	[redacted]_MQT...
Collector Type	MQTT Sparkplug B C...
Description	[redacted]_MQT...
Memory Buffer Size(...	20
Minimum Free Space...	150
Total Events Collected	0
Total Events Reported	0

The changes to the collector instance are saved.

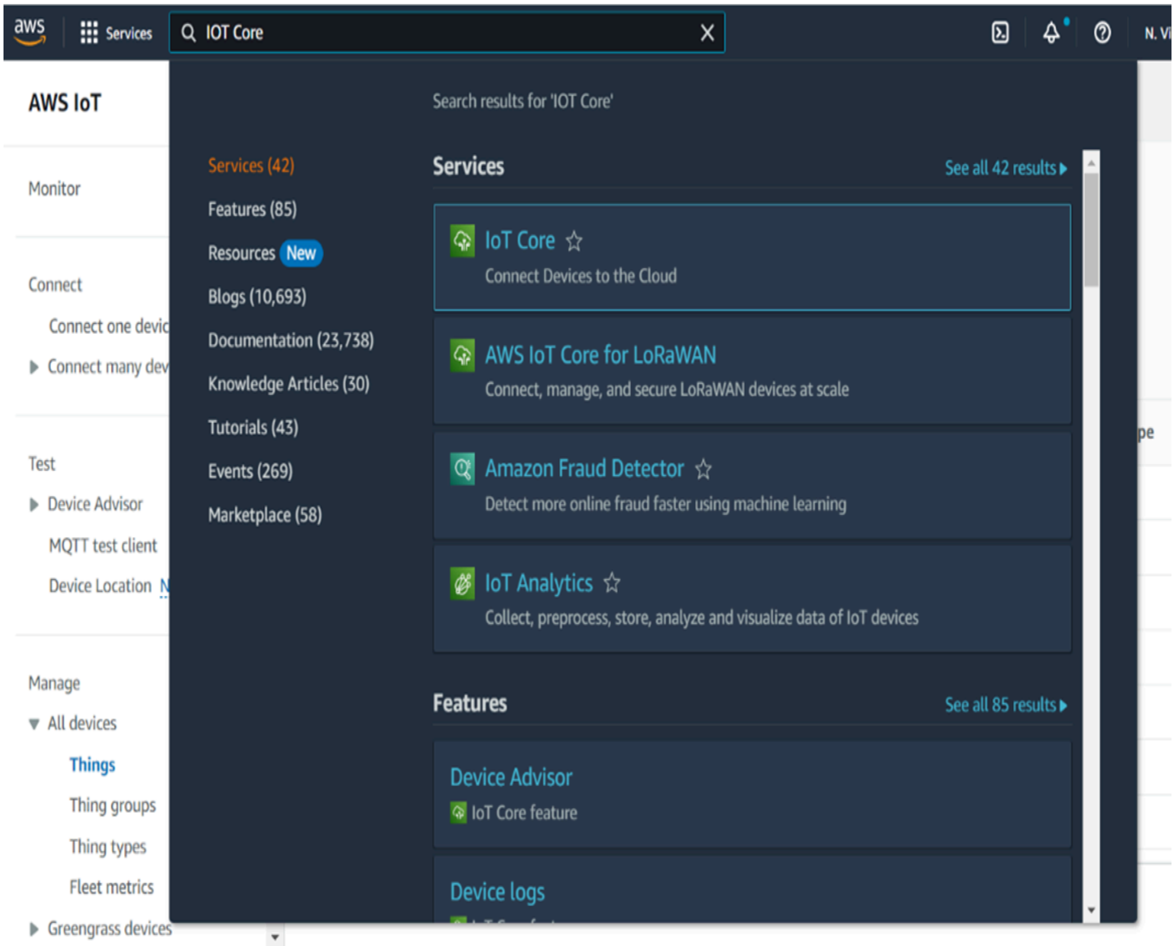
17. If needed, restart the collector.

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 384)*.

Connecting the MQTT Collector to the IOT Core

1. In the AWS console, search for the **IOT core** as shown in the following figure.



Search results for 'IOT Core'

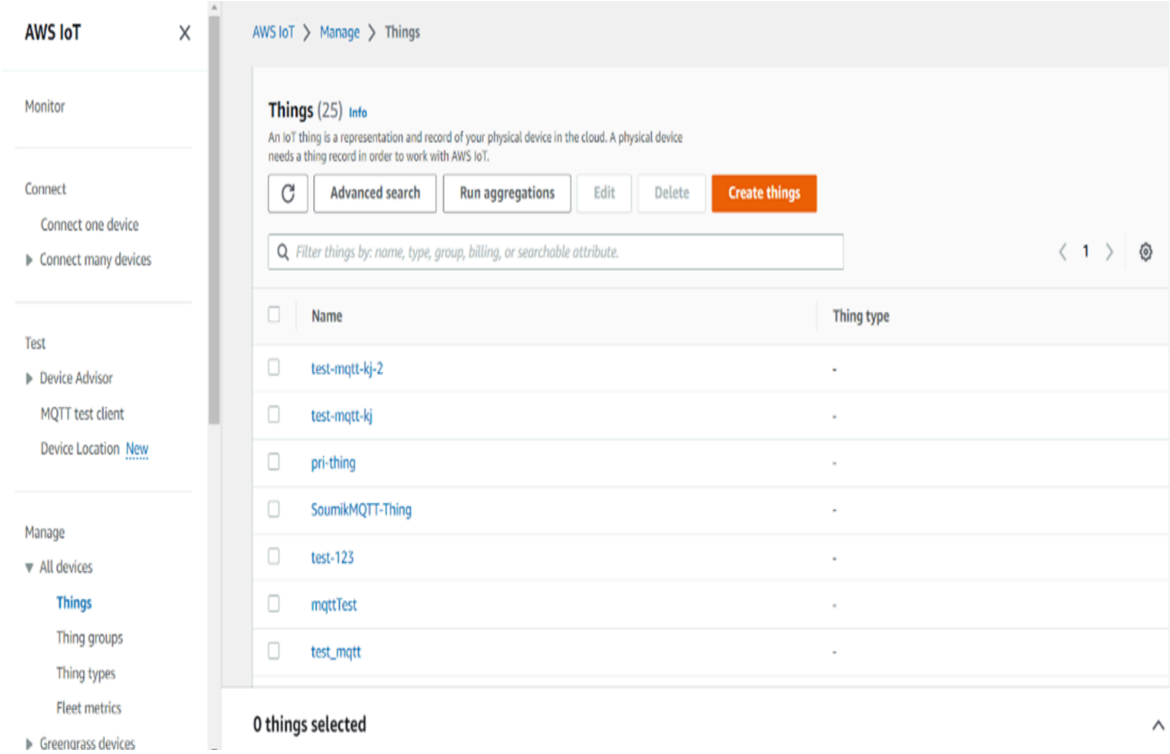
Connect Devices to the Cloud

Connect, manage, and secure LoRaWAN devices at scale

Detect more online fraud faster using machine learning

Collect, preprocess, store, analyze and visualize data of IoT devices

2. In the navigation pane, select **All Devices** and then **Things**.



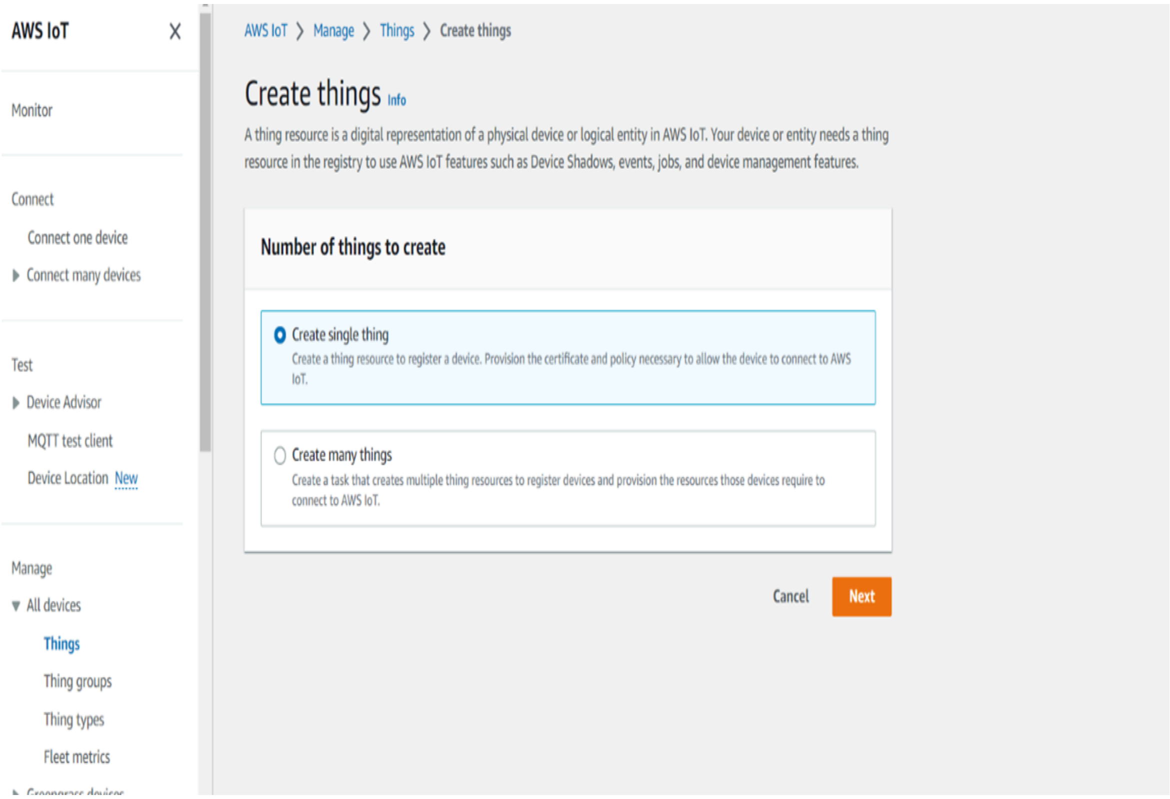
Things (25) Info

An IoT thing is a representation and record of your physical device in the cloud. A physical device needs a thing record in order to work with AWS IoT.

□	Name	Thing type
□	test-mqtt-kj-2	-
□	test-mqtt-kj	-
□	pri-thing	-
□	SoumikMQTT-Thing	-
□	test-123	-
□	mqttTest	-
□	test_mqtt	-

0 things selected

3. Select **Create things**, and then select **Create single thing** and click **Next**.



Create things Info

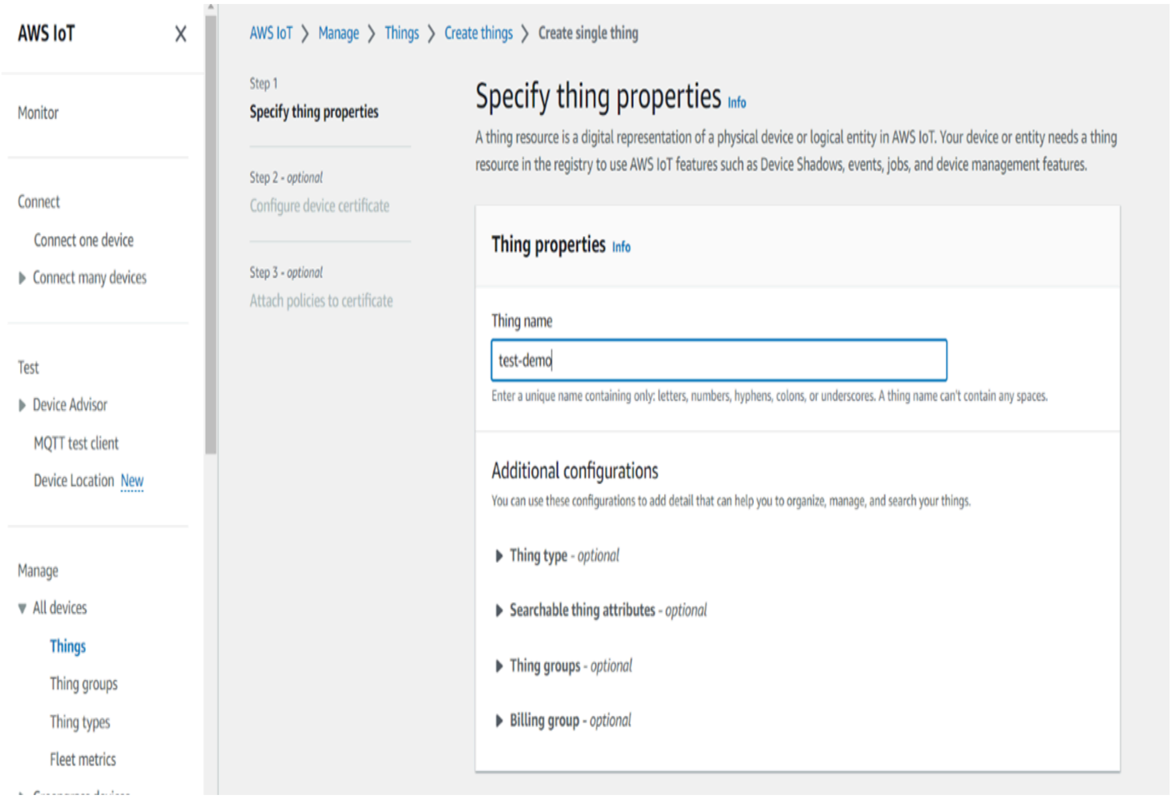
A thing resource is a digital representation of a physical device or logical entity in AWS IoT. Your device or entity needs a thing resource in the registry to use AWS IoT features such as Device Shadows, events, jobs, and device management features.

Number of things to create

● Create single thing
Create a thing resource to register a device. Provision the certificate and policy necessary to allow the device to connect to AWS IoT.

○ Create many things
Create a task that creates multiple thing resources to register devices and provision the resources those devices require to connect to AWS IoT.

4. Provide the thing name and keep the default configurations as is, and then click on **Next**.



Specify thing properties

A thing resource is a digital representation of a physical device or logical entity in AWS IoT. Your device or entity needs a thing resource in the registry to use AWS IoT features such as Device Shadows, events, jobs, and device management features.

Thing properties

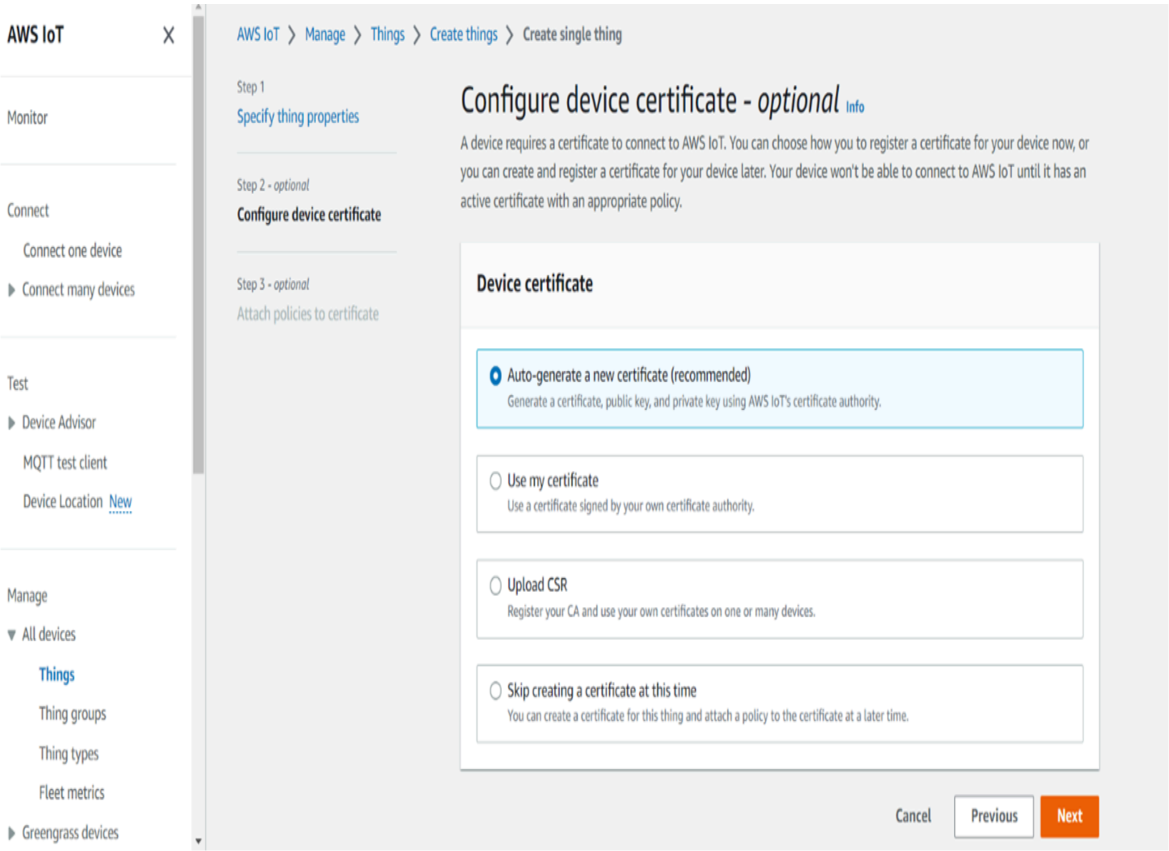
Thing name: test-demo

Enter a unique name containing only: letters, numbers, hyphens, colons, or underscores. A thing name can't contain any spaces.

Additional configurations

You can use these configurations to add detail that can help you to organize, manage, and search your things.

5. In Configure Device certificate screen, use the default options and leave **Auto-generate a new certificate** selected, and then click **Next**.



Configure device certificate - *optional* Info

A device requires a certificate to connect to AWS IoT. You can choose how you to register a certificate for your device now, or you can create and register a certificate for your device later. Your device won't be able to connect to AWS IoT until it has an active certificate with an appropriate policy.

Device certificate

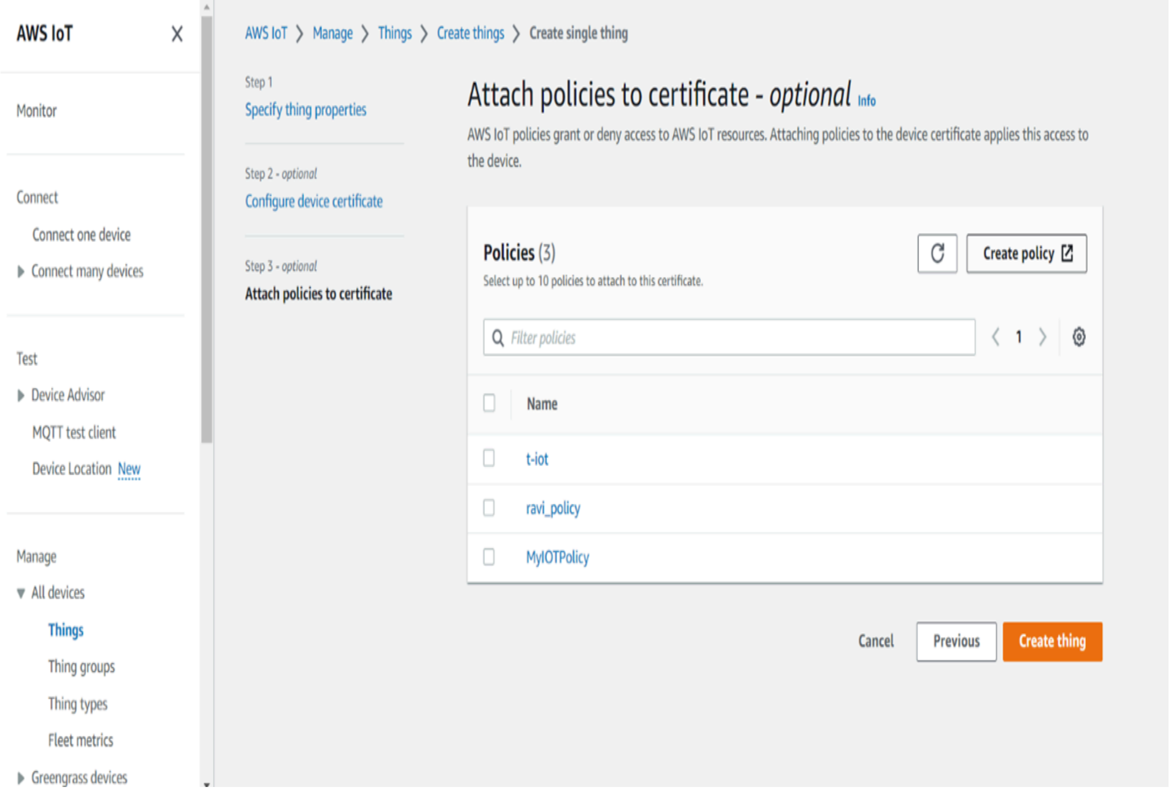
● Auto-generate a new certificate (recommended)
Generate a certificate, public key, and private key using AWS IoT's certificate authority.

○ Use my certificate
Use a certificate signed by your own certificate authority.

○ Upload CSR
Register your CA and use your own certificates on one or many devices.

○ Skip creating a certificate at this time
You can create a certificate for this thing and attach a policy to the certificate at a later time.

6. Attach the policy to the things if you have existing ones, or else create a new policy.



Attach policies to certificate - *optional* Info

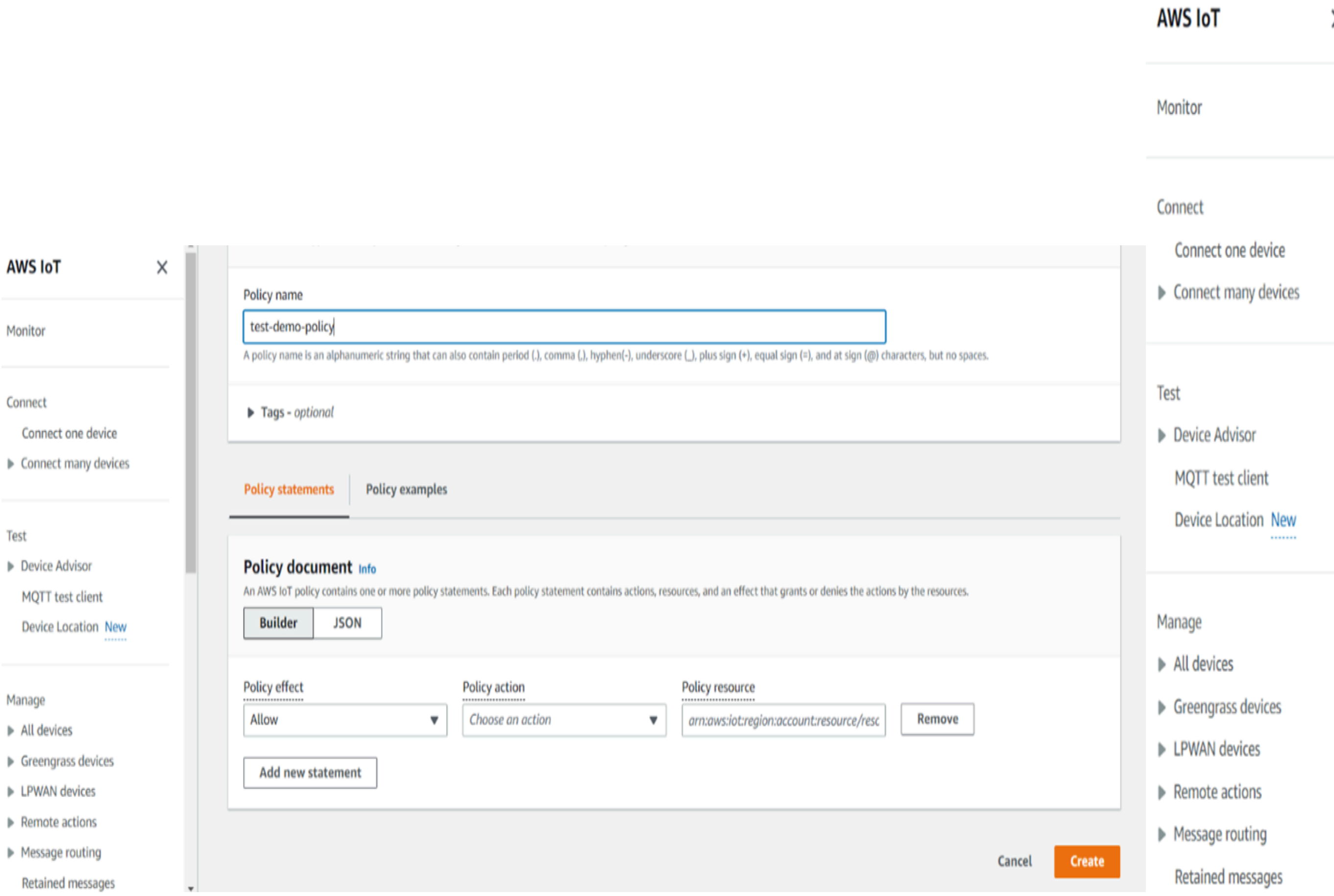
AWS IoT policies grant or deny access to AWS IoT resources. Attaching policies to the device certificate applies this access to the device.

Policies (3)

Select up to 10 policies to attach to this certificate.

	Name
□	t-iot
□	ravi_policy
□	MyIOTPolicy

7. Provide the **Policy name** and attach the same permissions as shown in the following figure and select **Create.**



Policy name

test-demo-policy

A policy name is an alphanumeric string that can also contain period (.), comma (,), hyphen(-), underscore (_), plus sign (+), equal sign (=), and at sign (@) characters, but no spaces.

▶ Tags - *optional*

Policy statements | Policy examples

Policy document Info

An AWS IoT policy contains one or more policy statements. Each policy statement contains actions, resources, and an effect that grants or denies the actions by the resources.

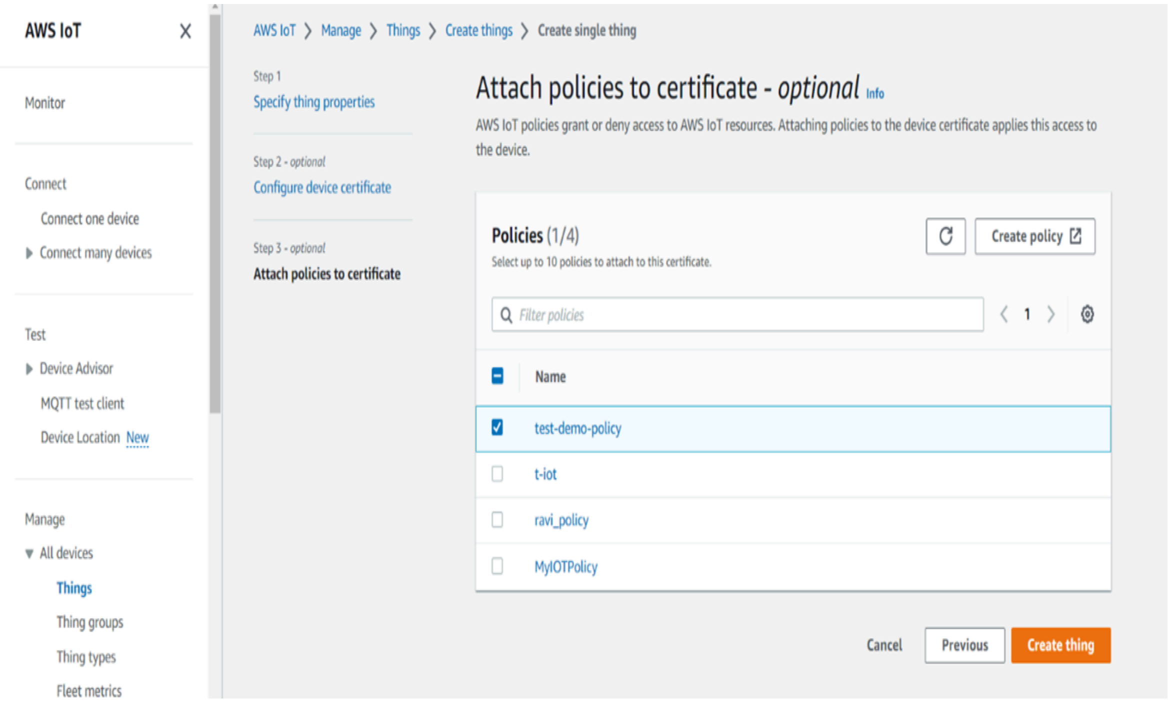
Builder | JSON

Policy effect: Allow

Policy action: *Choose an action*

Policy resource: *arn:aws:iot:region:account:resource/resc*

8. Select the policy you created, and then click **Create thing**.



Attach policies to certificate - *optional* Info

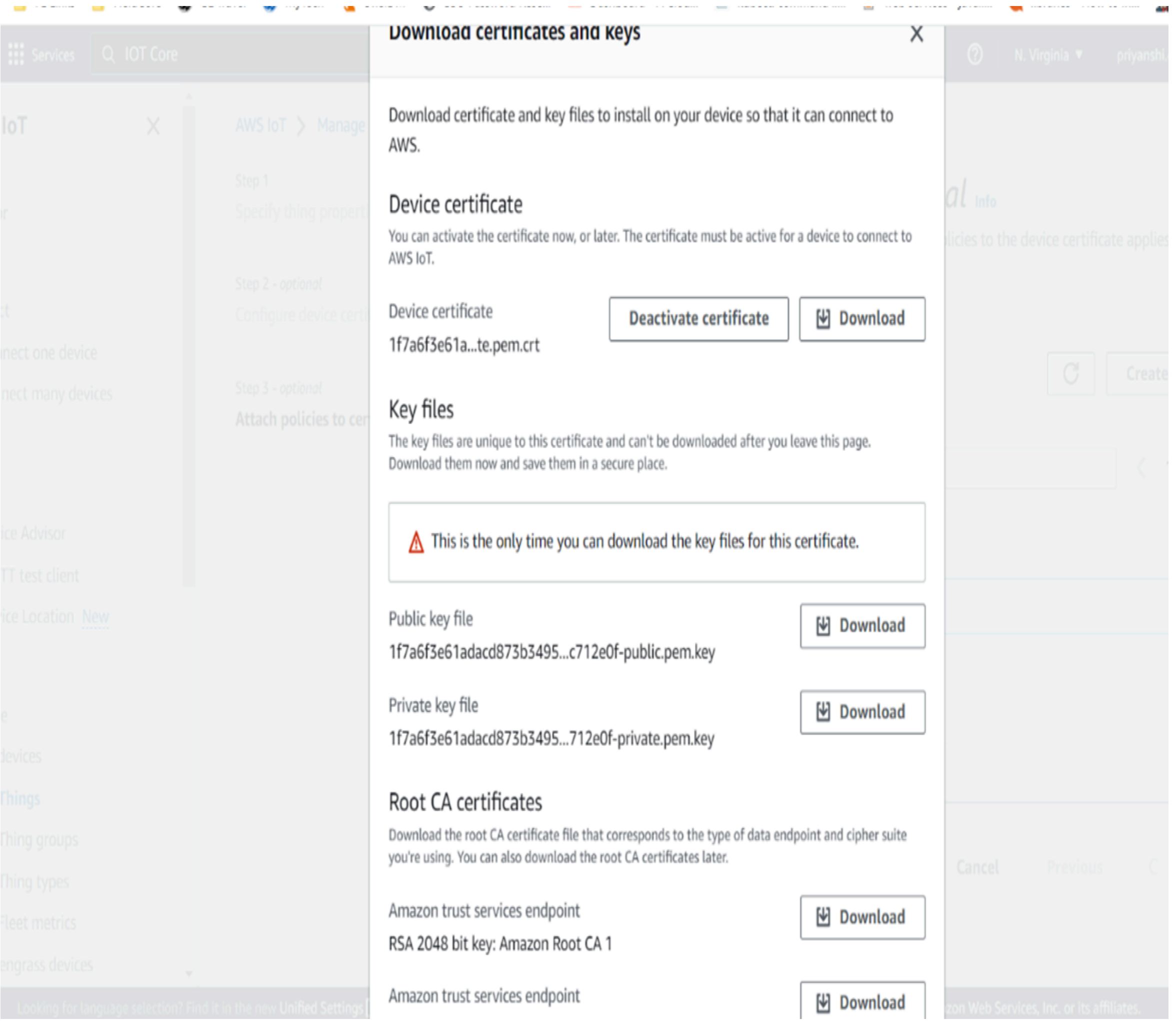
AWS IoT policies grant or deny access to AWS IoT resources. Attaching policies to the device certificate applies this access to the device.

Policies (1/4)

Select up to 10 policies to attach to this certificate.

	Name
☑	test-demo-policy
□	t-iot
□	ravi_policy
□	MyIOTPolicy

9. Download all the certificates and key files that have been created.



Download certificates and keys

Download certificate and key files to install on your device so that it can connect to AWS.

Device certificate

You can activate the certificate now, or later. The certificate must be active for a device to connect to AWS IoT.

Device certificate
1f7a6f3e61a...te.pem.crt

Key files

The key files are unique to this certificate and can't be downloaded after you leave this page. Download them now and save them in a secure place.

⚠ This is the only time you can download the key files for this certificate.

Public key file
1f7a6f3e61adacd873b3495...c712e0f-public.pem.key

Private key file
1f7a6f3e61adacd873b3495...712e0f-private.pem.key

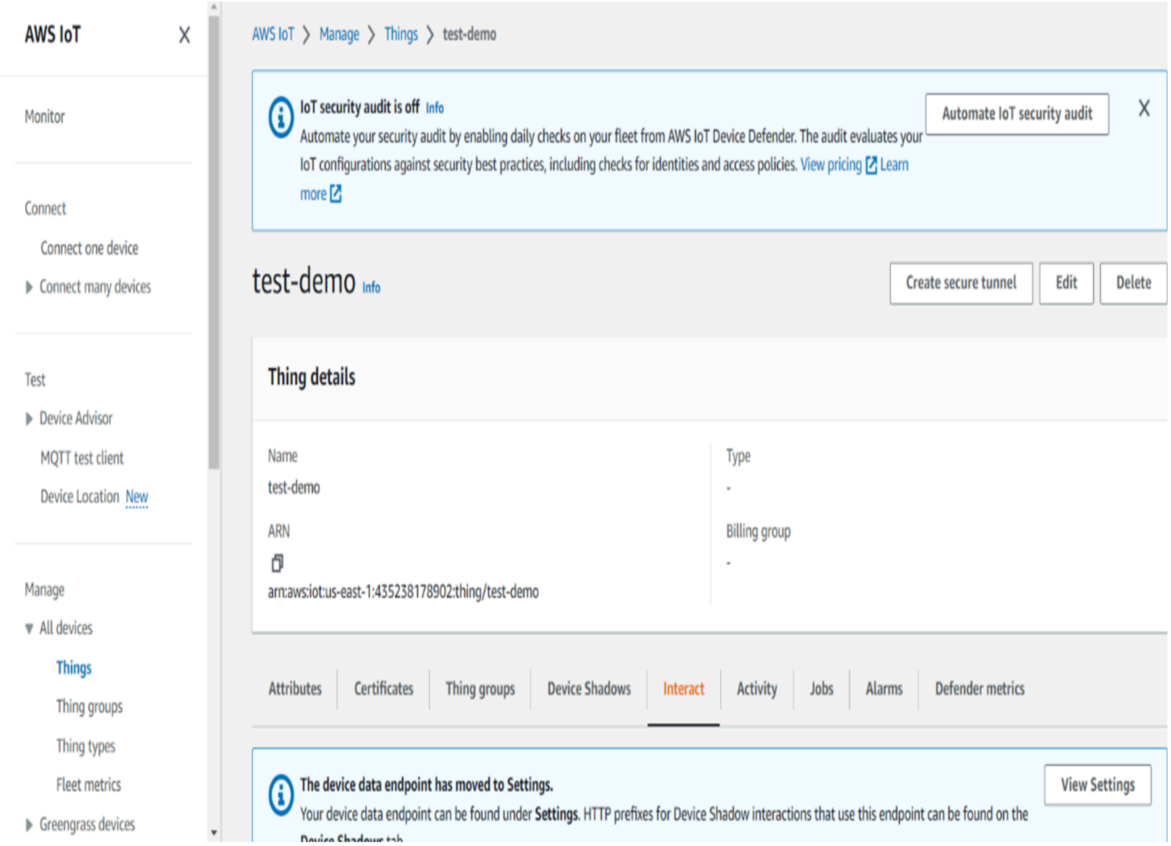
Root CA certificates

Download the root CA certificate file that corresponds to the type of data endpoint and cipher suite you're using. You can also download the root CA certificates later.

Amazon trust services endpoint
RSA 2048 bit key: Amazon Root CA 1

Amazon trust services endpoint

10. After the things are created, select the things and click the Interact tab.



IoT security audit is off Info

Automate your security audit by enabling daily checks on your fleet from AWS IoT Device Defender. The audit evaluates your IoT configurations against security best practices, including checks for identities and access policies. View pricing Learn more

test-demo Info

Thing details

Name
test-demo

ARN
arn:aws:iot:us-east-1:435238178902:thing/test-demo

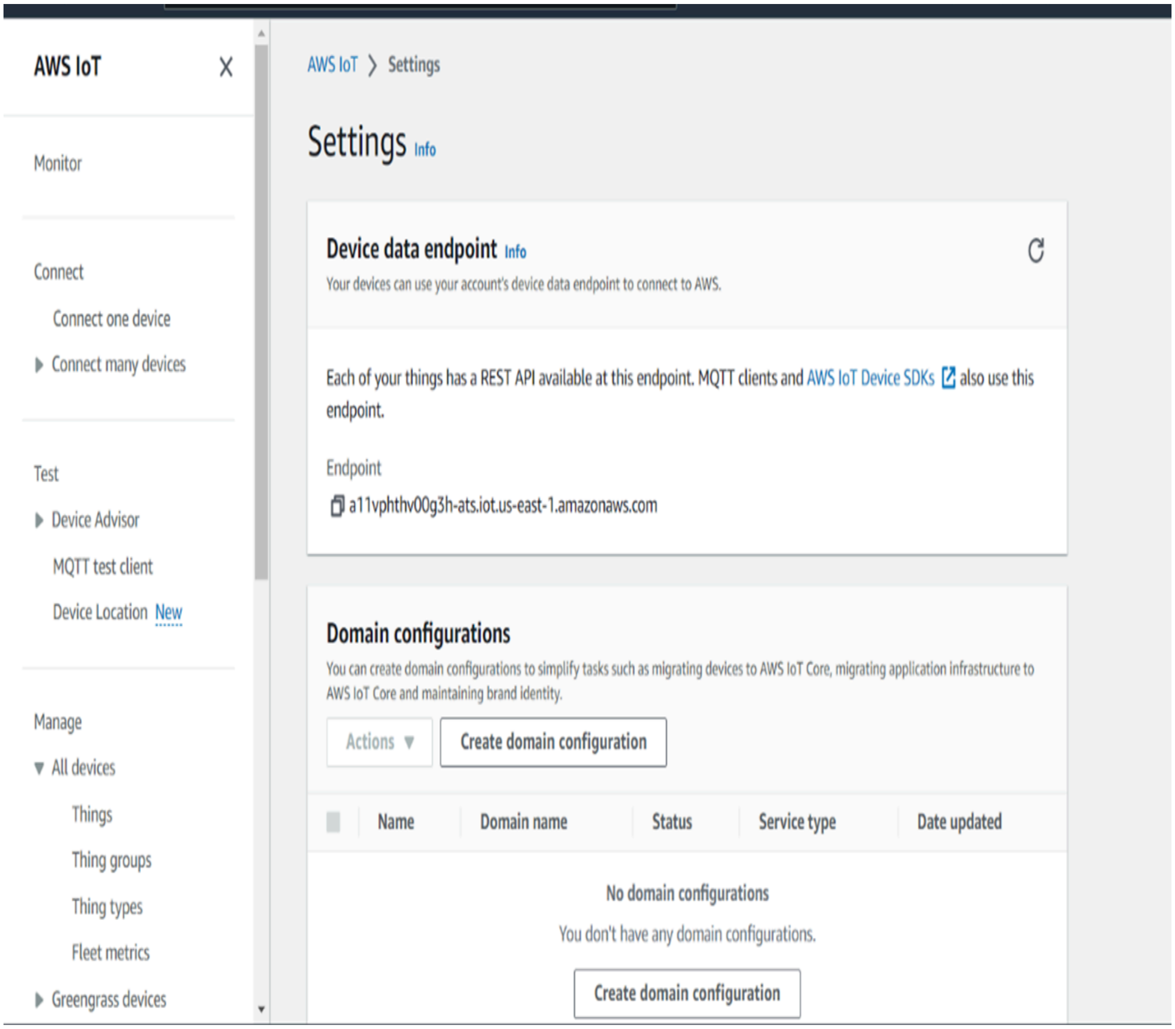
Type
-

Billing group
-

The device data endpoint has moved to Settings.

Your device data endpoint can be found under **Settings**. HTTP prefixes for Device Shadow interactions that use this endpoint can be found on the Device Shadows tab

11. Copy the **Endpoint** that will be used for creating the MQTT collector.



Settings

Device data endpoint

Your devices can use your account's device data endpoint to connect to AWS.

Each of your things has a REST API available at this endpoint. MQTT clients and AWS IoT Device SDKs also use this endpoint.

Endpoint
a11vphthv00g3h-ats.iot.us-east-1.amazonaws.com

Domain configurations

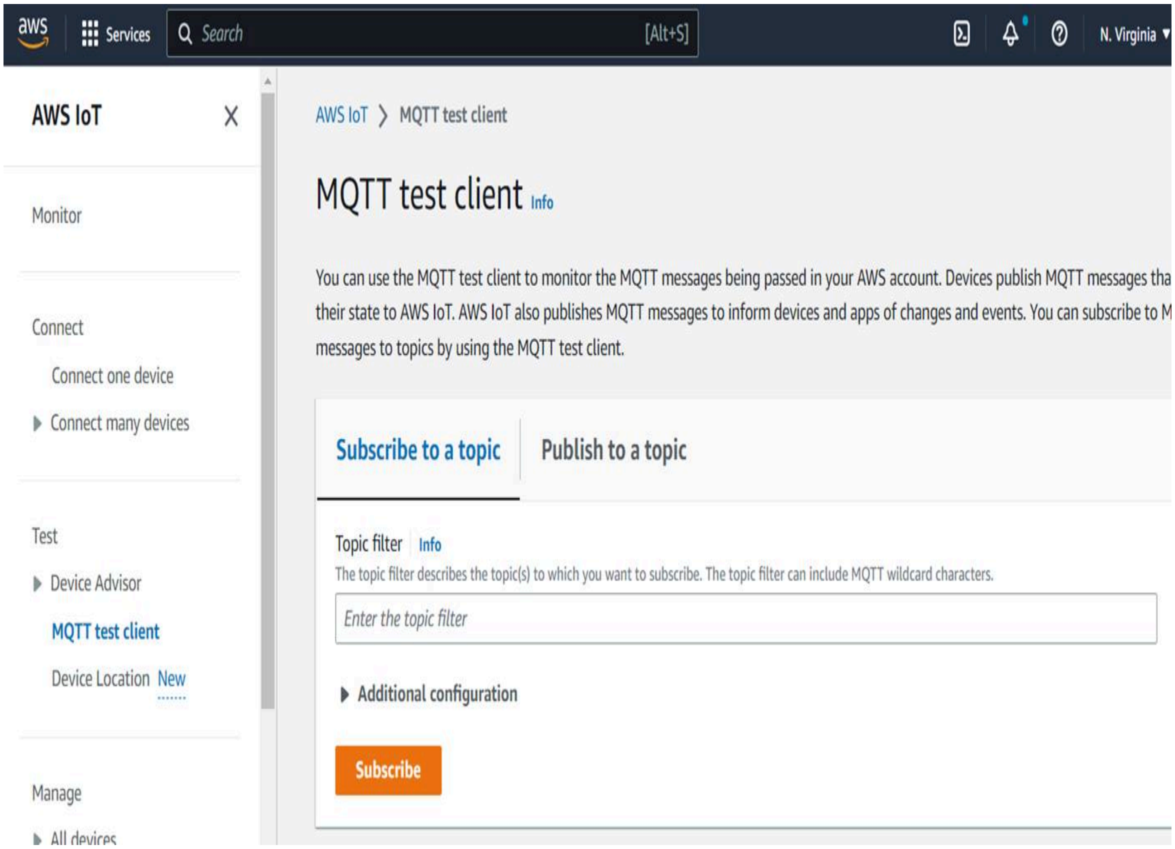
You can create domain configurations to simplify tasks such as migrating devices to AWS IoT Core, migrating application infrastructure to AWS IoT Core and maintaining brand identity.

Name	Domain name	Status	Service type	Date updated

No domain configurations

You don't have any domain configurations.

12. On the left side of the Test tab, click on the MQTT test client.



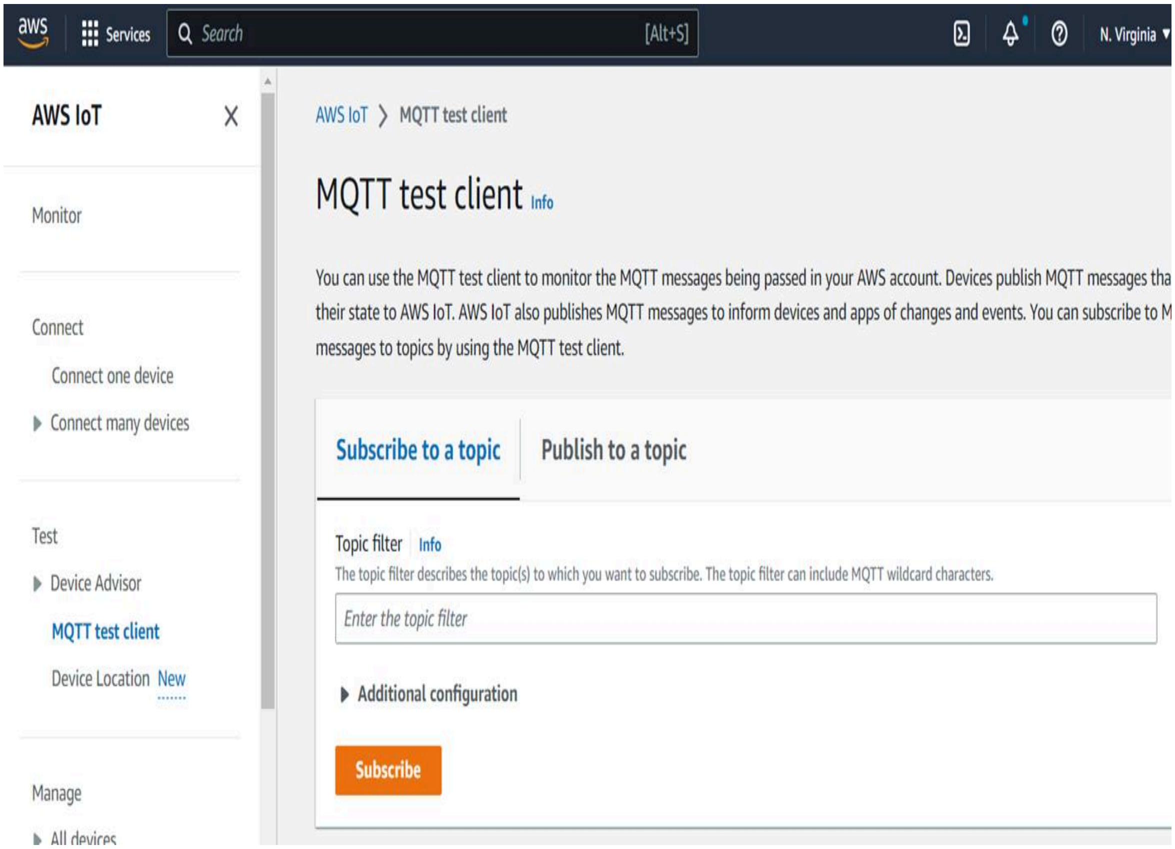
MQTT test client Info

You can use the MQTT test client to monitor the MQTT messages being passed in your AWS account. Devices publish MQTT messages tha their state to AWS IoT. AWS IoT also publishes MQTT messages to inform devices and apps of changes and events. You can subscribe to M messages to topics by using the MQTT test client.

Topic filter Info

The topic filter describes the topic(s) to which you want to subscribe. The topic filter can include MQTT wildcard characters.

13. Create a topic of your choice and then **Subscribe** to it.



You can use the MQTT test client to monitor the MQTT messages being passed in your AWS account. Devices publish MQTT messages tha their state to AWS IoT. AWS IoT also publishes MQTT messages to inform devices and apps of changes and events. You can subscribe to M messages to topics by using the MQTT test client.

The topic filter describes the topic(s) to which you want to subscribe. The topic filter can include MQTT wildcard characters.

14. Restart the MQTT collector instance.

> **Note:**
> The name of the collector interface should be name that you have provide while creating things. Our thing's name was "test-demo," as shown in the following figure.

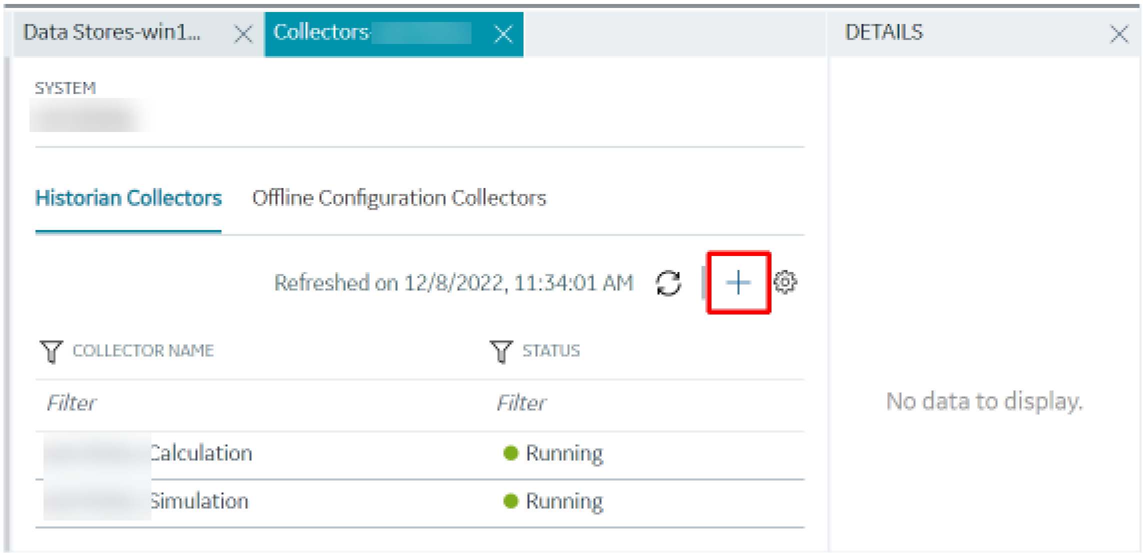
Add and Configure an ODBC Collector Using Configuration Hub

1. Deploy Proficy Historian for AWS. *(on page 43)*
2. Install collectors *(on page 90)*. You can install them on-premises or on a VPC (which can be different from the one on which Proficy Historian for AWS is deployed).
3. Enable TLS encryption for Collectors Connecting to Cloud Historian *(on page 150)*.

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 359)*.

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 1335)*, which does not require you to install Web-based Clients.

1. Access Configuration Hub *(on page 97)*.
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 +.



COLLECTOR NAME	STATUS
Filter	*Filter*
Calculation	● Running
Simulation	● Running

   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.

Field	Description
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 179)*.
      - **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 698)*.
      - **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 151)*.
        - AWS cloud. For more information, refer to AWS Cloud *(on page 158)*.
        - Google cloud. For more information, refer to Google Cloud *(on page 172)*.

   b. If you need to send data to an on-premise/Cloud Historian server, select **Historian Server**. 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.

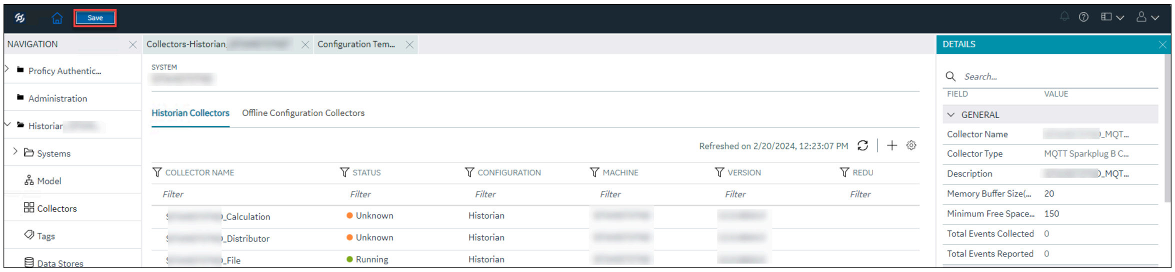
      If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:
        - iH Security Admins
        - iH Collector Admins
        - iH Tag Admins

    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. Continuous data collection is resumed only after the previous data has been recovered. By default, this value is set to 0, which means data recovery is not attempted. The maximum

Field	Description
	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. **Note:** If this field is blank, enter the required minimum value of 100 milliseconds.

15. As needed, enter values in the other sections common to all collectors *(on page 669)*.
16. In the upper-left corner of the page, select **Save**.



Refreshed on 2/20/2024, 12:23:07 PM

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSION	REDU
Filter	Filter	Filter	Filter	Filter	Filter
[redacted]_Calculation	● Unknown	Historian	[redacted]	[redacted]	
[redacted]_Distributor	● Unknown	Historian	[redacted]	[redacted]	
[redacted]_File	● Running	Historian	[redacted]	[redacted]	

FIELD	VALUE
GENERAL	
Collector Name	[redacted]_MQT...
Collector Type	MQTT Sparkplug B C...
Description	[redacted]_MQT...
Memory Buffer Size(...	20
Minimum Free Space...	150
Total Events Collected	0
Total Events Reported	0

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 384)*.

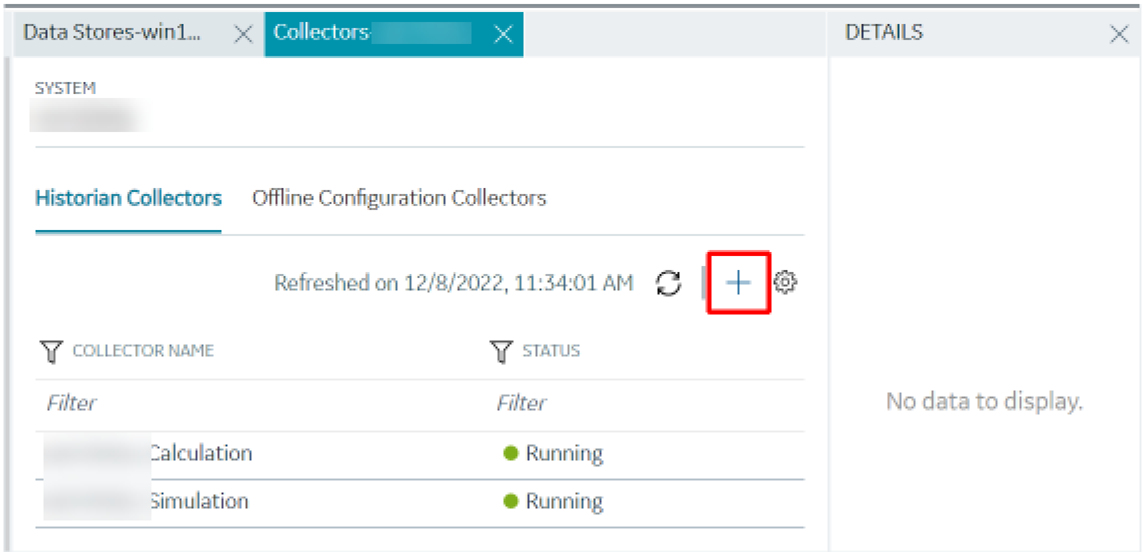
Add and Configure an OPC Classic Data Access Collector

1. Deploy Proficy Historian for AWS. *(on page 43)*
2. Install collectors *(on page 90)*. You can install them on-premises or on a VPC (which can be different from the one on which Proficy Historian for AWS is deployed).
3. Enable TLS encryption for Collectors Connecting to Cloud Historian *(on page 150)*.

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 374)*.

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 1335)*, which does not require you to install Web-based Clients.

1. Access Configuration Hub *(on page 97)*.
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 +.

   Data Stores-win1... | Collectors | DETAILS

   SYSTEM

   Historian Collectors | Offline Configuration Collectors

   Refreshed on 12/8/2022, 11:34:01 AM

COLLECTOR NAME	STATUS
Filter	*Filter*
Calculation	● Running
Simulation	● Running

   No data to display.

   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.
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 179)*.
      - **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 698)*.
      - **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 151)*.
        - AWS cloud. For more information, refer to AWS Cloud *(on page 158)*.
        - Google cloud. For more information, refer to Google Cloud *(on page 172)*.

   b. If you need to send data to an on-premise/Cloud Historian server, select **Historian Server**.
      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.

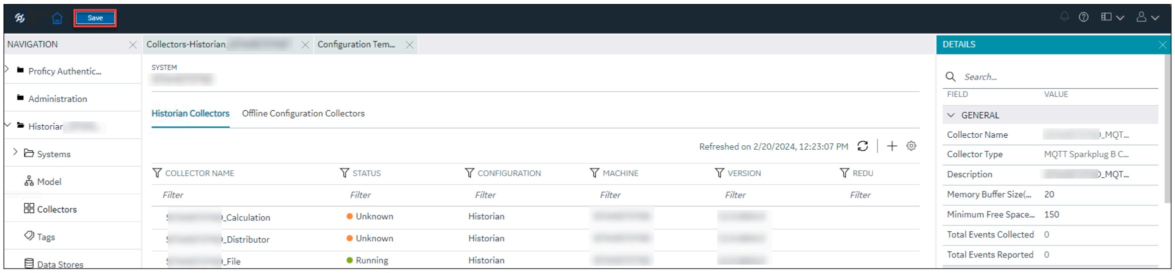
13. Select **Add**.
    The collector instance is added. The fields specific to the collector section appear in the **DETAILS** section.
14. Under **COLLECTOR SPECIFIC CONFIGURATION**, configure values as described in the following table.

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.

Field	Description
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. 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: ◦ USGB014 -and- ◦ 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. ◦ **Multithreaded**: Select this option for better performance. We recommend that you configure your collector to use the default multi-threading model. ◦ **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. 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.

Field	Description
	◦ **Made On-Line**: Select this option to process any configuration changes immediately (after 30 seconds) after you select the **Update** button. **Note:** ▪ Some OPC servers cannot handle processing configuration changes online. If you experience any instability with changes made online, use the next option. ◦ **Made After Collector Restart**: Select this option to hold all configuration changes until you manually restart the collector.

15. As needed, enter values in the other sections common to all collectors *(on page 669)*.
16. In the upper-left corner of the page, select **Save**.



SYSTEM
[redacted]

Refreshed on 2/20/2024, 12:23:07 PM

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSION	REDU
Filter	Filter	Filter	Filter	Filter	Filter
[redacted]_Calculation	● Unknown	Historian	[redacted]	[redacted]	
[redacted]_Distributor	● Unknown	Historian	[redacted]	[redacted]	
[redacted]_File	● Running	Historian	[redacted]	[redacted]	

FIELD	VALUE
GENERAL	
Collector Name	[redacted]_MQT...
Collector Type	MQTT Sparkplug B C...
Description	[redacted]_MQT...
Memory Buffer Size(...	20
Minimum Free Space...	150
Total Events Collected	0
Total Events Reported	0

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 384)*.

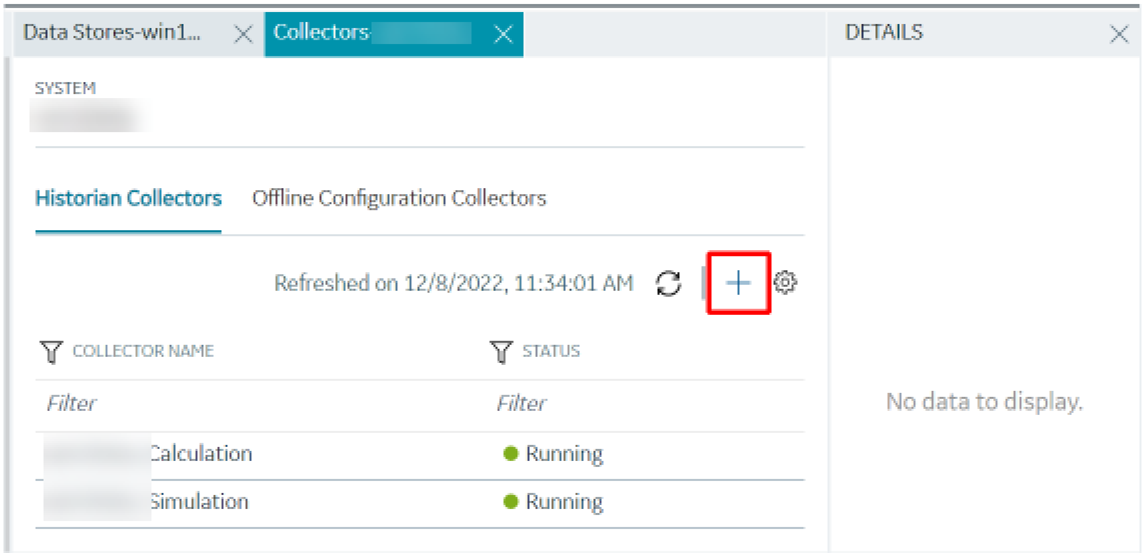
Add and Configure an OPC Classic HDA Collector

1. Deploy Proficy Historian for AWS. *(on page 43)*
2. Install collectors *(on page 90)*. You can install them on-premises or on a VPC (which can be different from the one on which Proficy Historian for AWS is deployed).
3. Enable TLS encryption for Collectors Connecting to Cloud Historian *(on page 150)*.

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 397)*.

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 1335)*, which does not require you to install Web-based Clients.

1. Access Configuration Hub *(on page 97)*.
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-premise 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 179)*.
      - **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 698)*.
      - **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 151)*.
        - AWS cloud. For more information, refer to AWS Cloud *(on page 158)*.
        - Google cloud. For more information, refer to Google Cloud *(on page 172)*.

   b. If you need to send data to an on-premise/Cloud Historian server, select **Historian Server**.
      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

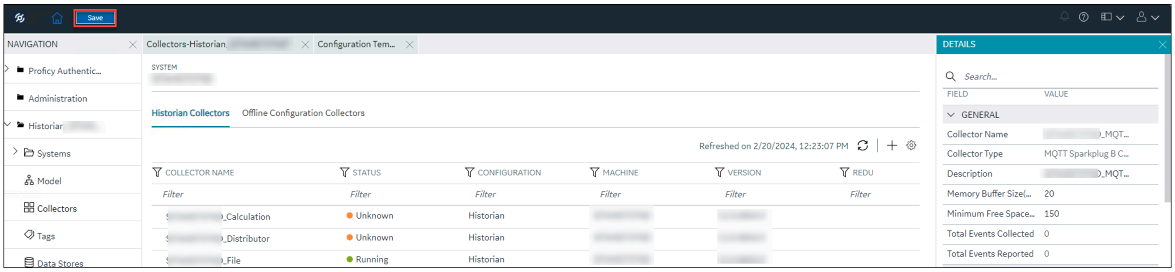
    You can also configure the collector to start automatically when you start the computer.
13. Select **Add**.

    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

Field	Description
	duration between the current time and the last known write time. Continuous data collection is resumed only after the previous data has been recovered. You can enter a value between 1 and 150.

15. As needed, enter values in the other sections common to all collectors *(on page 669)*.
16. In the upper-left corner of the page, select **Save**.



Refreshed on 2/20/2024, 12:23:07 PM

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSION	REDU
Filter	Filter	Filter	Filter	Filter	Filter
[redacted]_Calculation	● Unknown	Historian	[redacted]	[redacted]	
[redacted]_Distributor	● Unknown	Historian	[redacted]	[redacted]	
[redacted]_File	● Running	Historian	[redacted]	[redacted]	

FIELD	VALUE
Collector Name	[redacted]_MQT...
Collector Type	MQTT Sparkplug B C...
Description	[redacted]_MQT...
Memory Buffer Size(...	20
Minimum Free Space...	150
Total Events Collected	0
Total Events Reported	0

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 384)*.

Add and Configure an OPC UA Data Access Collector

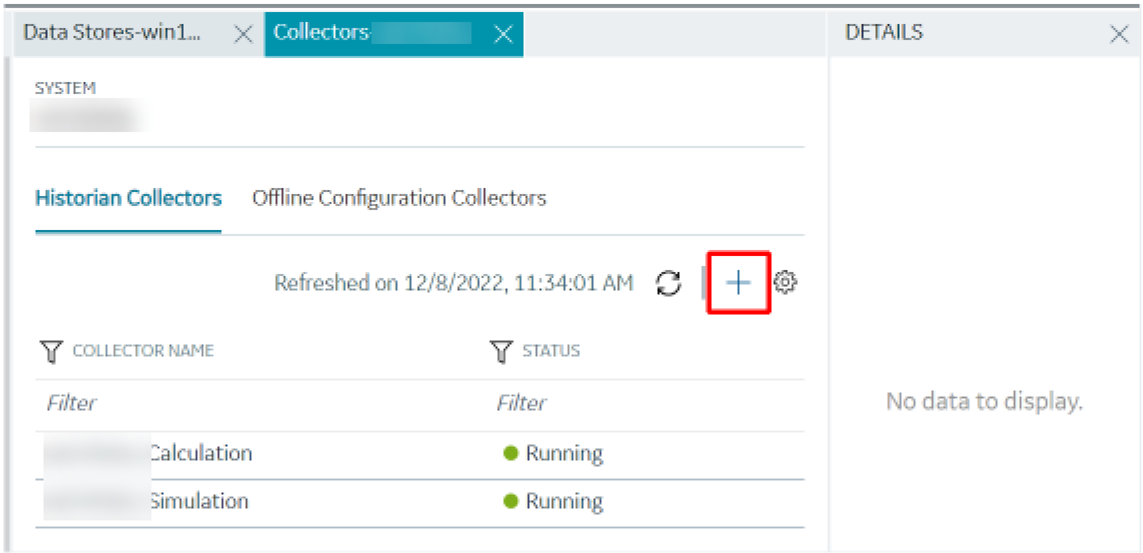
1. Deploy Proficy Historian for AWS. *(on page 43)*
2. Install collectors *(on page 90)*. You can install them on-premises or on a VPC (which can be different from the one on which Proficy Historian for AWS is deployed).
3. Enable TLS encryption for Collectors Connecting to Cloud Historian *(on page 150)*.

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 411)*.

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 1335)*, which does not require you to install Web-based Clients.

1. Access Configuration Hub *(on page 97)*.
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 +.

COLLECTOR NAME	STATUS
Filter	Filter
Calculation	● Running
Simulation	● Running

   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-premise/Cloud 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 179)*.
        - **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 698)*.
        - **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 151)*.
            - AWS cloud. For more information, refer to AWS Cloud *(on page 158)*.
            - Google cloud. For more information, refer to Google Cloud *(on page 172)*.

    b. If you need to send data to an on-premise/Cloud Historian server, select **Historian Server**. 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

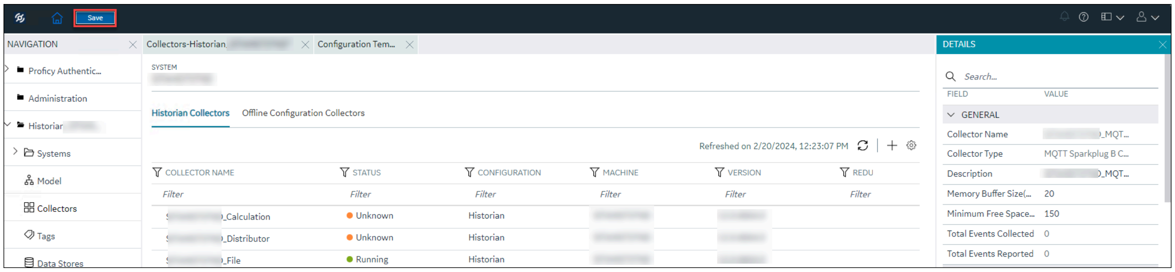
You can also configure the collector to start automatically when you start the computer.

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. You can establish a secured connectivity in one of the following ways: ◦ **Using certificates:** To use certificates, switch off the **User Security** toggle. ◦ **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 413)*.

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.

15. As needed, enter values in the other sections common to all collectors *(on page 669)*.
16. In the upper-left corner of the page, select **Save**.



COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSION	REDU
Filter	Filter	Filter	Filter	Filter	Filter
[redacted]_Calculation	● Unknown	Historian	[redacted]	[redacted]	
[redacted]_Distributor	● Unknown	Historian	[redacted]	[redacted]	
[redacted]_File	● Running	Historian	[redacted]	[redacted]	

FIELD	VALUE
GENERAL	
Collector Name	[redacted]_MQT...
Collector Type	MQTT Sparkplug B C...
Description	[redacted]_MQT...
Memory Buffer Size(...	20
Minimum Free Space...	150
Total Events Collected	0
Total Events Reported	0

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 413)*.

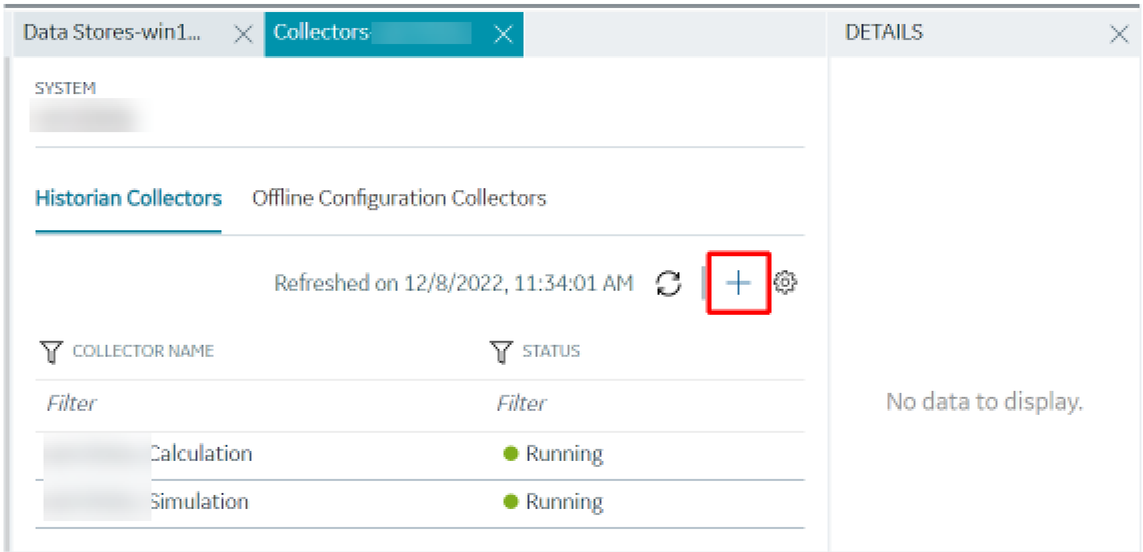
Add and Configure an OSI PI Collector

1. Deploy Proficy Historian for AWS. *(on page 43)*
2. Install collectors *(on page 90)*. You can install them on-premises or on a VPC (which can be different from the one on which Proficy Historian for AWS is deployed).
3. Enable TLS encryption for Collectors Connecting to Cloud Historian *(on page 150)*.
4. 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 421)*.

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 1335)*, which does not require you to install Web-based Clients.

1. Access Configuration Hub *(on page 97)*.
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.

Field	Description
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 179)*.
      - **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 698)*.
      - **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 151)*.
        - AWS cloud. For more information, refer to AWS Cloud *(on page 158)*.
        - Google cloud. For more information, refer to Google Cloud *(on page 172)*.

   b. If you need to send data to an on-premise/Cloud Historian server, select **Historian Server**.

      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

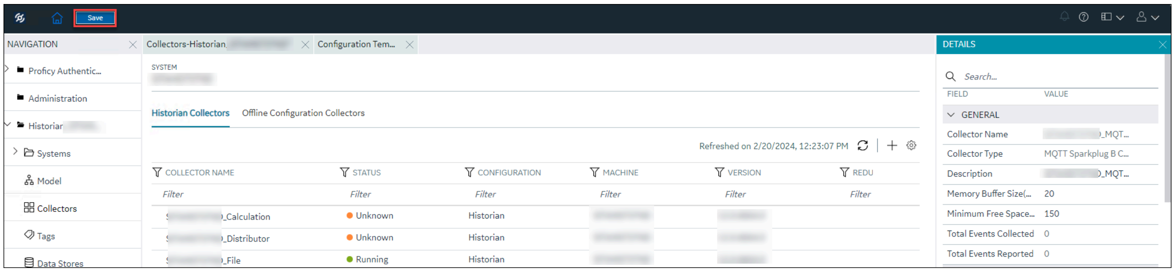
    You can also configure the collector to start automatically when you start the computer.

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 between the collector and the OSI PI server is reestablished. This time is calculated as the duration between the current time and the last known write time. Continuous data collection is resumed only after the previous data has been recovered. The default value is 4 hours.
Data Source	Specify whether you want to collect data from PI archive or PI snapshot:

Field	Description
	◦ **Archive**: Stores timeseries-based data. ◦ **Snapshot**: Stores the most recent values of tags.

15. As needed, enter values in the other sections common to all collectors *(on page 669)*.
16. In the upper-left corner of the page, select **Save**.



SYSTEM [redacted]

Refreshed on 2/20/2024, 12:23:07 PM

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSION	REDU
Filter	*Filter*	*Filter*	*Filter*	*Filter*	*Filter*
[redacted]_Calculation	● Unknown	Historian	[redacted]	[redacted]	
[redacted]_Distributor	● Unknown	Historian	[redacted]	[redacted]	
[redacted]_File	● Running	Historian	[redacted]	[redacted]	

FIELD	VALUE
GENERAL	
Collector Name	[redacted]_MQT...
Collector Type	MQTT Sparkplug B C...
Description	[redacted]_MQT...
Memory Buffer Size(...	20
Minimum Free Space...	150
Total Events Collected	0
Total Events Reported	0

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 384)*.

Add and Configure an OSI PI Distributor

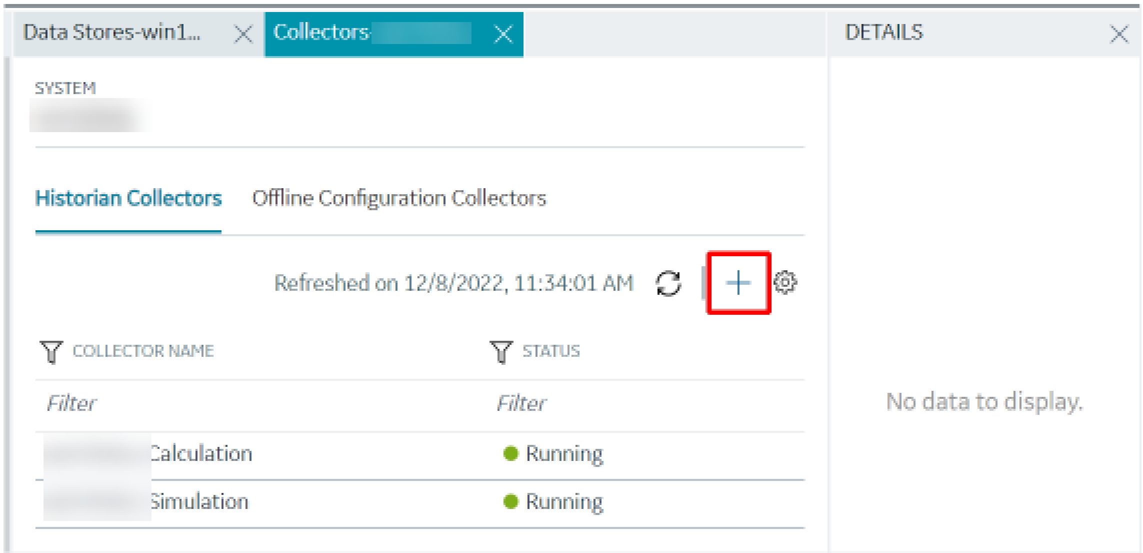
1. Deploy Proficy Historian for AWS. *(on page 43)*
2. Install collectors *(on page 90)*. You can install them on-premises or on a VPC (which can be different from the one on which Proficy Historian for AWS is deployed).
3. Enable TLS encryption for Collectors Connecting to Cloud Historian *(on page 150)*.
4. 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 434)*.

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 1335)*, which does not require you to install Web-based Clients.

1. Access Configuration Hub *(on page 97)*.
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

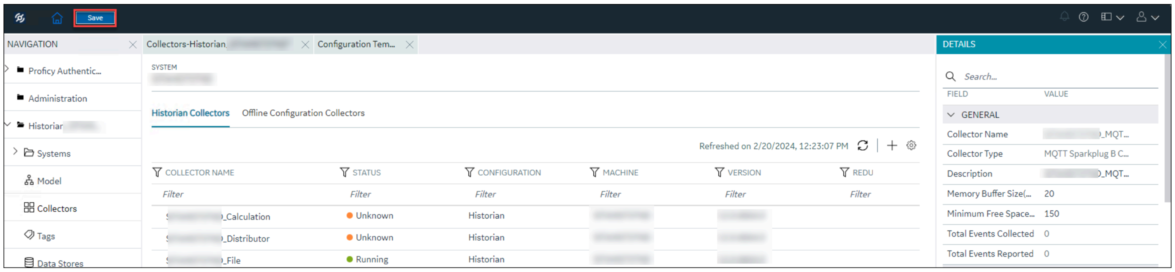
You can also configure the collector to start automatically when you start the computer.

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. Continuous data collection is resumed only after the previous data has been recovered. The default value is 4 hours.
Data Source	Specify whether you want to collect data from PI archive or PI snapshot: ◦ **Archive**: Stores timeseries-based data. ◦ **Snapshot**: Stores the most recent values of tags.

15. As needed, enter values in the other sections common to all collectors *(on page 669)*.
16. In the upper-left corner of the page, select **Save**.



SYSTEM
[redacted]

Historian Collectors | Offline Configuration Collectors

Refreshed on 2/20/2024, 12:23:07 PM

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSION	REDU
Filter	Filter	Filter	Filter	Filter	Filter
[redacted]_Calculation	● Unknown	Historian	[redacted]	[redacted]	
[redacted]_Distributor	● Unknown	Historian	[redacted]	[redacted]	
[redacted]_File	● Running	Historian	[redacted]	[redacted]	

FIELD	VALUE
GENERAL	
Collector Name	[redacted]_MQT...
Collector Type	MQTT Sparkplug B C...
Description	[redacted]_MQT...
Memory Buffer Size(...	20
Minimum Free Space...	150
Total Events Collected	0
Total Events Reported	0

The changes to the collector instance are saved.

17. If needed, restart the collector.

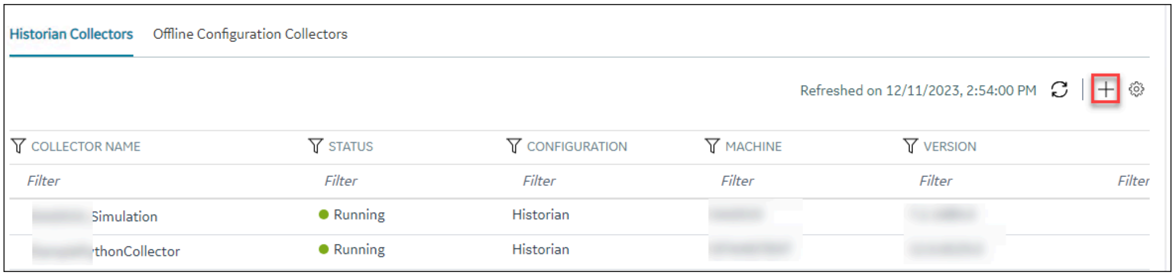
Specify the tags *(on page 384)* whose data you want to collect using the collector.

Add a Python Collector Instance using Configuration Hub

- Install Python Collector *(on page)*.

This topic describes how to add a Python collector instance using Configuration hub.

1. Access Configuration Hub *(on page)*.
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 +.

   Refreshed on 12/11/2023, 2:54:00 PM

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSION
Filter	*Filter*	*Filter*	*Filter*	*Filter*
[redacted]Simulation	● Running	Historian	[redacted]	[redacted]
[redacted]thonCollector	● Running	Historian	[redacted]	[redacted]

   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)*.
        - **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)*.
        - **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)*.
            - AWS cloud. For more information, refer to AWS Cloud *(on page)*.
            - Google cloud. For more information, refer to Google Cloud *(on page)*.

    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)* whose data you want to collect using the collector.

Add and Configure a Server-to-Server Collector

1. Deploy Proficy Historian for AWS. *(on page 43)*
2. Install collectors *(on page 90)*. You can install them on-premises or on a VPC (which can be different from the one on which Proficy Historian for AWS is deployed).
3. Enable TLS encryption for Collectors Connecting to Cloud Historian *(on page 150)*.

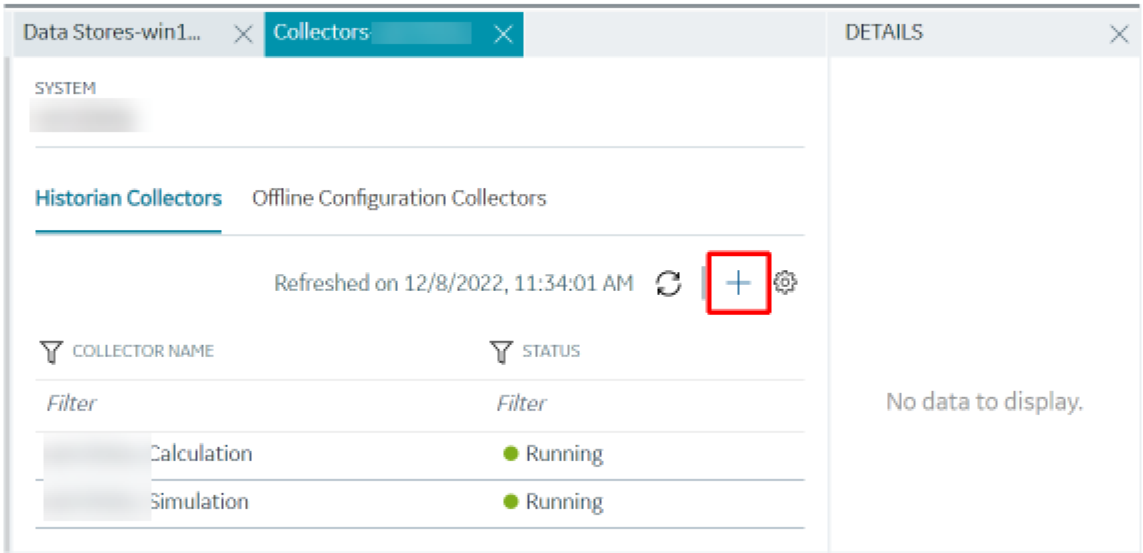
The Server-to-Server collector collects data and messages from a source Historian server to a destination Historian server or a cloud destination. For more information, refer to Overview of the Server-to-Server Collector *(on page 475)*.

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 1335)*, which does not require you to install Web-based Clients.

> **Important:**
> If you want to send data from the cloud to an on-premises Historian, you must disable TLS encryption by disabling the Cloud Settings as mentioned in the note in the Enable TLS encryption

for Collectors Connecting to Cloud Historian *(on page 150)*section before instantiating the collector. You can enable it back to instantiate another collector connecting to Cloud Historian for which you need TLS encryption to be enabled.

1. Access Configuration Hub *(on page 97)*.
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 +.

COLLECTOR NAME	STATUS
Filter	Filter
Calculation	● Running
Simulation	● Running

   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-premise/Cloud 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**.

   In case of windows Configuration Hub, 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 **NLB DNS** of Cloud Historian, you can change it in case of another Historian server. In case of Containerized Configuration Hub, 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 **historian-svc**, you need to change it to the DNS of Destination 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 179)*.
       - **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 698)*.
       - **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 151)*.
         - AWS cloud. For more information, refer to AWS Cloud *(on page 158)*.
         - Google cloud. For more information, refer to Google Cloud *(on page 172)*.

    b. If you need to send data to an on-premise/Cloud 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.

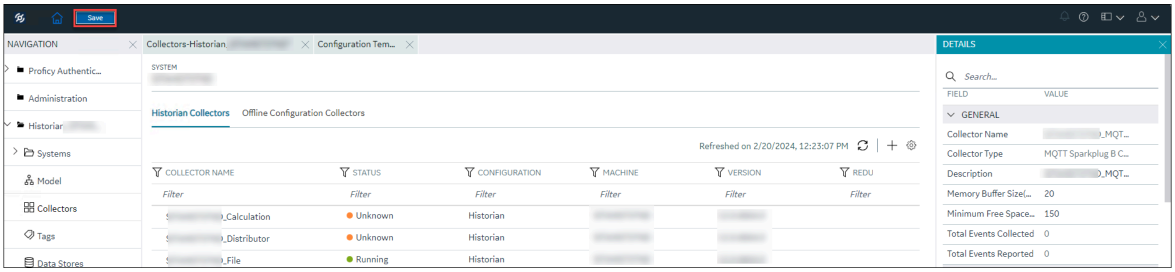
      If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:
        - iH Security Admins
        - iH Collector Admins
        - iH Tag Admins

    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.

Field	Description
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. Alarms and events data will automatically have a prefix added to it with the following syntax: `MachineName_Datasource` For example, if your alarm is forwarded from the server `Almserver12` with a data source named `OPCAE`, the prefix will be `Almserver12_OPCAE`.

16. As needed, enter values in the other sections common to all collectors *(on page 669)*.
17. In the upper-left corner of the page, select **Save**.



SYSTEM [redacted]

Refreshed on 2/20/2024, 12:23:07 PM

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSION	REDU
Filter	Filter	Filter	Filter	Filter	Filter
[redacted]_Calculation	● Unknown	Historian	[redacted]	[redacted]	
[redacted]_Distributor	● Unknown	Historian	[redacted]	[redacted]	
[redacted]_File	● Running	Historian	[redacted]	[redacted]	

FIELD	VALUE
GENERAL	
Collector Name	[redacted]_MQT...
Collector Type	MQTT Sparkplug B C...
Description	[redacted]_MQT...
Memory Buffer Size(...	20
Minimum Free Space...	150
Total Events Collected	0
Total Events Reported	0

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 384)*.

Add and Configure a Server-to-Server Distributor

1. Deploy Proficy Historian for AWS. *(on page 43)*
2. Install collectors *(on page 90)*. You can install them on-premises or on a VPC (which can be different from the one on which Proficy Historian for AWS is deployed).
3. Enable TLS encryption for Collectors Connecting to Cloud Historian *(on page 150)*.

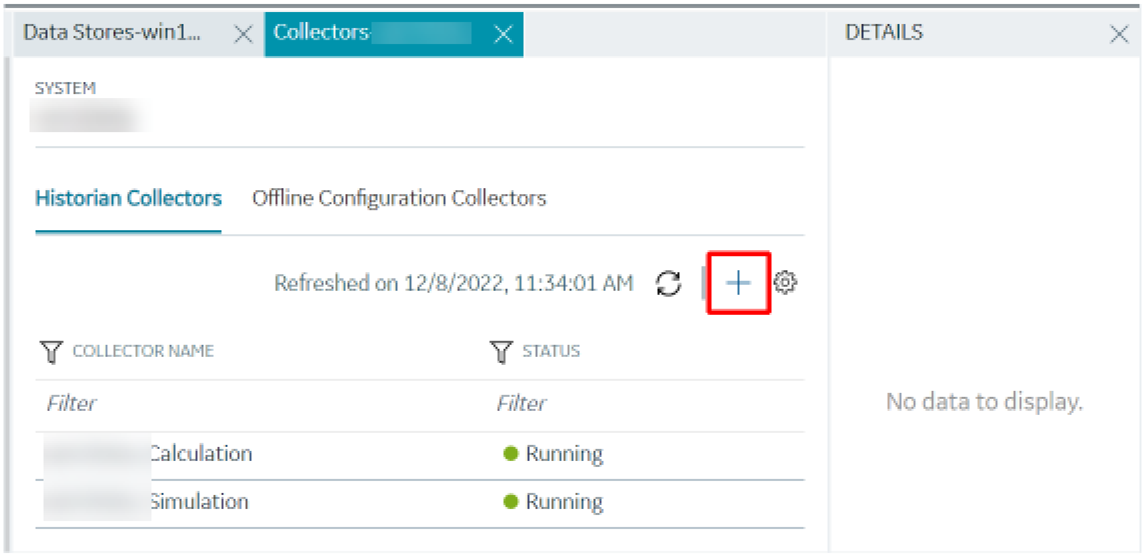
The Server-to-Server distributor is used to send data from a smaller Historian server to a larger, centralized on-premise Historian server. For more information, refer to Overview of the Server-to-Server

Distributor *(on page 530)*. If you want to send data to a cloud destination, use the Server-to-Server collector *(on page 654)* 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 1335)*, which does not require you to install Web-based Clients.

> **Important:**
> If you want to send data from the cloud to an on-premises Historian, you must disable TLS encryption by disabling the Cloud Settings as mentioned in the note in the Enable TLS encryption for Collectors Connecting to Cloud Historian *(on page 150)*section before instantiating the collector. You can enable it back to instantiate another collector connecting to Cloud Historian for which you need TLS encryption to be enabled.

1. Access Configuration Hub *(on page 97)*.
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 +.



SYSTEM
[redacted]

Refreshed on 12/8/2022, 11:34:01 AM

COLLECTOR NAME	STATUS
Filter	Filter
[redacted]Calculation	● Running
[redacted]Simulation	● Running

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-premise/Cloud 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**.

   In case of windows Configuration Hub with Cloud Historian, 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 **NLB DNS** of Cloud Historian, you can change it in case of another Historian server. In case of Containerized Configuration Hub, 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 **historian-svc**, you need to change it to the DNS of Destination 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

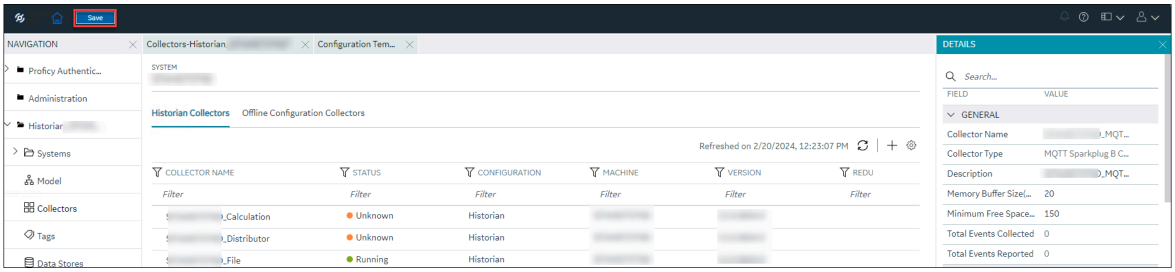
You can also configure the collector to start automatically when you start the computer.

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.

Field	Description
Add Prefix to Messages	The prefix to identify replicated messages on the destination. Alarms and events data will automatically have a prefix added to it with the following syntax: `MachineName_Datasource` For example, if your alarm is forwarded from the server `Almserver12` with a data source named `OPCAE`, the prefix will be `Almserver12_OPCAE`.

16. As needed, enter values in the other sections common to all collectors *(on page 669)*.
17. In the upper-left corner of the page, select **Save**.



Refreshed on 2/20/2024, 12:23:07 PM

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSION	REDU
Filter	Filter	Filter	Filter	Filter	Filter
[redacted]_Calculation	Unknown	Historian	[redacted]	[redacted]	
[redacted]_Distributor	Unknown	Historian	[redacted]	[redacted]	
[redacted]_File	Running	Historian	[redacted]	[redacted]	

FIELD	VALUE
GENERAL	
Collector Name	[redacted]_MQT...
Collector Type	MQTT Sparkplug B C...
Description	[redacted]_MQT...
Memory Buffer Size(...	20
Minimum Free Space...	150
Total Events Collected	0
Total Events Reported	0

The changes to the collector instance are saved.

18. If needed, restart the collector.

Specify the tags *(on page 384)* whose data you want to collect using the collector.

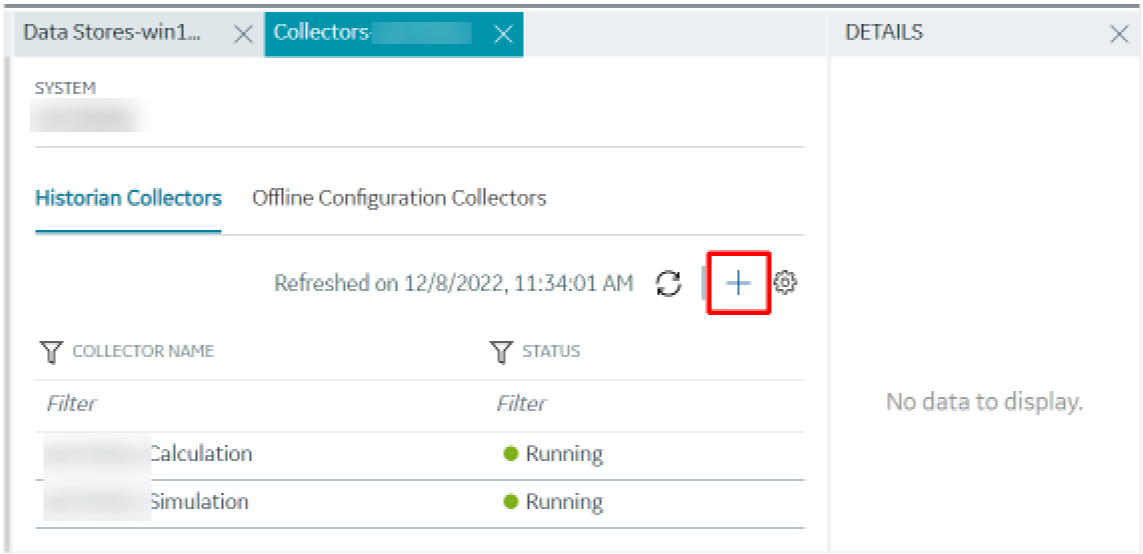
Add and Configure a Simulation Collector

1. Deploy Proficy Historian for AWS. *(on page 43)*
2. Install collectors *(on page 90)*. You can install them on-premises or on a VPC (which can be different from the one on which Proficy Historian for AWS is deployed).
3. Enable TLS encryption for Collectors Connecting to Cloud Historian *(on page 150)*.

The Simulation collector generates random numbers and string patterns for demonstration purposes. For more information, refer to Overview of the Simulation Collector *(on page 534)*.

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 1335)*, which does not require you to install Web-based Clients.

1. Access Configuration Hub *(on page 97)*.
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 +.



COLLECTOR NAME	STATUS
Filter	Filter
[redacted]Calculation	● Running
[redacted]Simulation	● Running

   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 179)*.
      - **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 698)*.
      - **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 151)*.
         - AWS cloud. For more information, refer to AWS Cloud *(on page 158)*.
         - Google cloud. For more information, refer to Google Cloud *(on page 172)*.

   b. If you need to send data to an on-premise/Cloud Historian server, select **Historian Server**.

      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

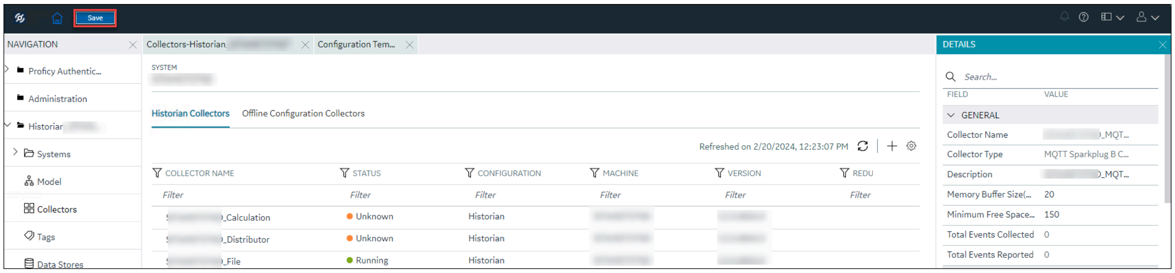
    You can also configure the collector to start automatically when you start the computer.

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.

14. As needed, enter values in the other sections common to all collectors *(on page 669)*.
15. In the upper-left corner of the page, select **Save**.



SYSTEM

[redacted]

Historian Collectors | Offline Configuration Collectors

Refreshed on 2/20/2024, 12:23:07 PM

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSION	REDU
Filter	Filter	Filter	Filter	Filter	Filter
[redacted]_Calculation	● Unknown	Historian	[redacted]	[redacted]	
[redacted]_Distributor	● Unknown	Historian	[redacted]	[redacted]	
[redacted]_File	● Running	Historian	[redacted]	[redacted]	

FIELD	VALUE
GENERAL	
Collector Name	[redacted]_MQT...
Collector Type	MQTT Sparkplug B C...
Description	[redacted]_MQT...
Memory Buffer Size(...	20
Minimum Free Space...	150
Total Events Collected	0
Total Events Reported	0

    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 384)*.

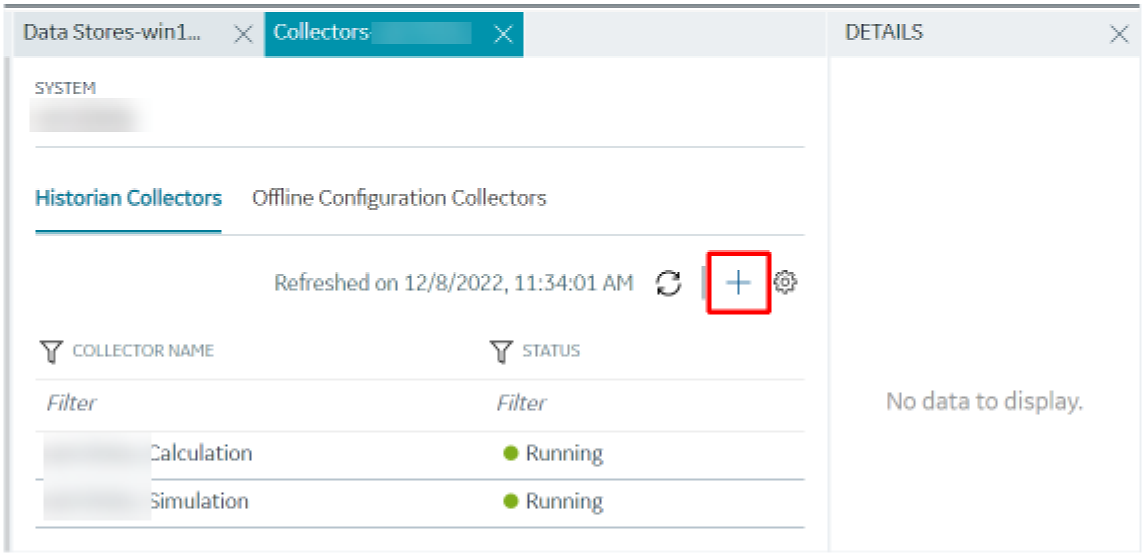
Add and Configure a Wonderware Collector

1. Deploy Proficy Historian for AWS. *(on page 43)*
2. Install collectors *(on page 90)*. You can install them on-premises or on a VPC (which can be different from the one on which Proficy Historian for AWS is deployed).
3. Enable TLS encryption for Collectors Connecting to Cloud Historian *(on page 150)*.

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 537)*.

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 1335)*, which does not require you to install Web-based Clients.

1. Access Configuration Hub *(on page 97)*.
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 +.

COLLECTOR NAME	STATUS
Filter	Filter
Calculation	● Running
Simulation	● Running

   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 179)*.
       - **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 698)*.
       - **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 151)*.
         - AWS cloud. For more information, refer to AWS Cloud *(on page 158)*.
         - Google cloud. For more information, refer to Google Cloud *(on page 172)*.

    b. If you need to send data to an on-premise/Cloud Historian server, select **Historian Server**.

       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.

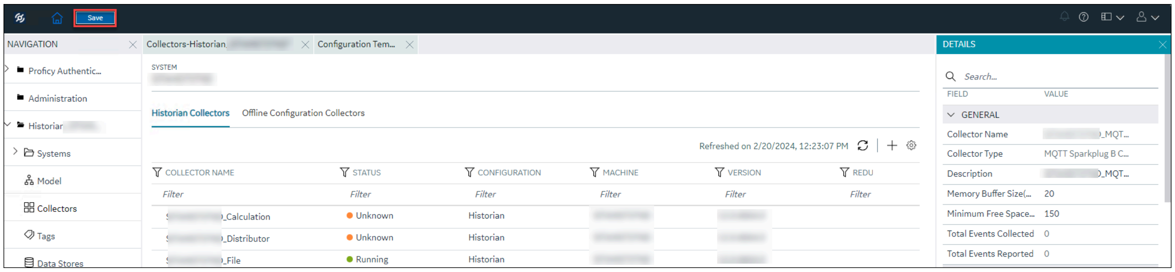
  If you have enabled the **Enforce Strict Collector Authentication** option in Historian Administrator, you must provide the credentials of a user who is added to at least one of the following security groups:
  - iH Security Admins
  - iH Collector Admins
  - iH Tag Admins

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. Continuous data collection is resumed only after the previous data has been recovered. 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.

17. As needed, enter values in the the other sections common to all collectors *(on page 669)*.
18. In the upper-left corner of the page, select **Save**.



Refreshed on 2/20/2024, 12:23:07 PM

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSION	REDU
Filter	Filter	Filter	Filter	Filter	Filter
[redacted]_Calculation	● Unknown	Historian	[redacted]	[redacted]	
[redacted]_Distributor	● Unknown	Historian	[redacted]	[redacted]	
[redacted]_File	● Running	Historian	[redacted]	[redacted]	

FIELD	VALUE
Collector Name	[redacted]_MQT...
Collector Type	MQTT Sparkplug B C...
Description	[redacted]_MQT...
Memory Buffer Size(...	20
Minimum Free Space...	150
Total Events Collected	0
Total Events Reported	0

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 384)*.

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 151)*, AWS Cloud *(on page 158)*, Azure Cloud *(on page 165)*, Google Could *(on page 172)*, and Predix Cloud *(on page 179)*. 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 16. 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.

Table 16. The General Section (continued)

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 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. 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 17. 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. 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.

Table 17. The Tags section (continued)

Field	Description
	- For polled data collection, this value represents the time required to complete a poll of tags in the collector. - 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. You can set this value for each tag as well. **Important:** 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: - **Polled:** Data is collected based on a scheduled time interval. This type of data collection is supported only for: ◦ The Calculation collector ◦ The HAB collector ◦ The iFIX collector ◦ The OPC Classic DA collector ◦ The OPC UA DA collector ◦ The Python collector ◦ The Simulation collector ◦ The Windows Performance collector - **Unsolicited:** Data is collected based on an event. This type of data collection is supported only for:

Table 17. The Tags section (continued)

Field	Description
	◦ The Calculation collector ◦ The HAB collector ◦ The MQTT collector ◦ The MQTT Sparkplug B collector ◦ The ODBC collector ◦ The OPC Classic DA collector ◦ The OPC Classic HDA collector ◦ The OPC UA DA collector ◦ The OSI PI collector ◦ The OSI PI distributor ◦ The Python collector ◦ The Server-to-Server collector ◦ The Server-to-Server distributor ◦ 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 18. 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. For more information, refer to About Collector and Archive Compression *(on page 742)*.

Table 18. The Collector Compression Section (continued)

Field	Description
Deadband	Indicates whether you want to apply a deadband based on the percentage of values or on absolute values. 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. **Note:** 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. 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. For more information, refer to About Collector and Archive Compression *(on page 742)*.
Compression Timeout	The time for one poll cycle for which collector compression is not used, thus sending all the samples to Historian.

Table 18. The Collector Compression Section (continued)

Field	Description
	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. For more information, refer to About Collector and Archive Compression *(on page 742)*.
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. For more information, refer to Enable Spike Logic *(on page)*.
Multiplier	Specifies how much larger a spike value must be than the deadband range before the spike logic is invoked. 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.

Table 18. The Collector Compression Section (continued)

Field	Description
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 19. 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 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. **Note:** • 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).

Table 19. The Collector Options Section (continued)

Field	Description
	• 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.
Delay Collection at Startup (sec)	The duration, in seconds, after which you want the data collection to begin post tag configuration.

Table 20. 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. **Note:** Leaving the debug mode enabled for a long time consumes disk space.
Message Compression	Indicates whether you want to apply message compression.

Table 21. 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.

Table 21. The Collector Status Output Section (continued)

Field	Description
	You must enter the address of a writable analog field. 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 operator (or a HMI/SCADA application) learn the current status of the collector. The value is updated only if the status of the collector changes. You must enter the address of a writable text field of at least eight characters. 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. You must enter the address of a writable analog field. 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`). 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

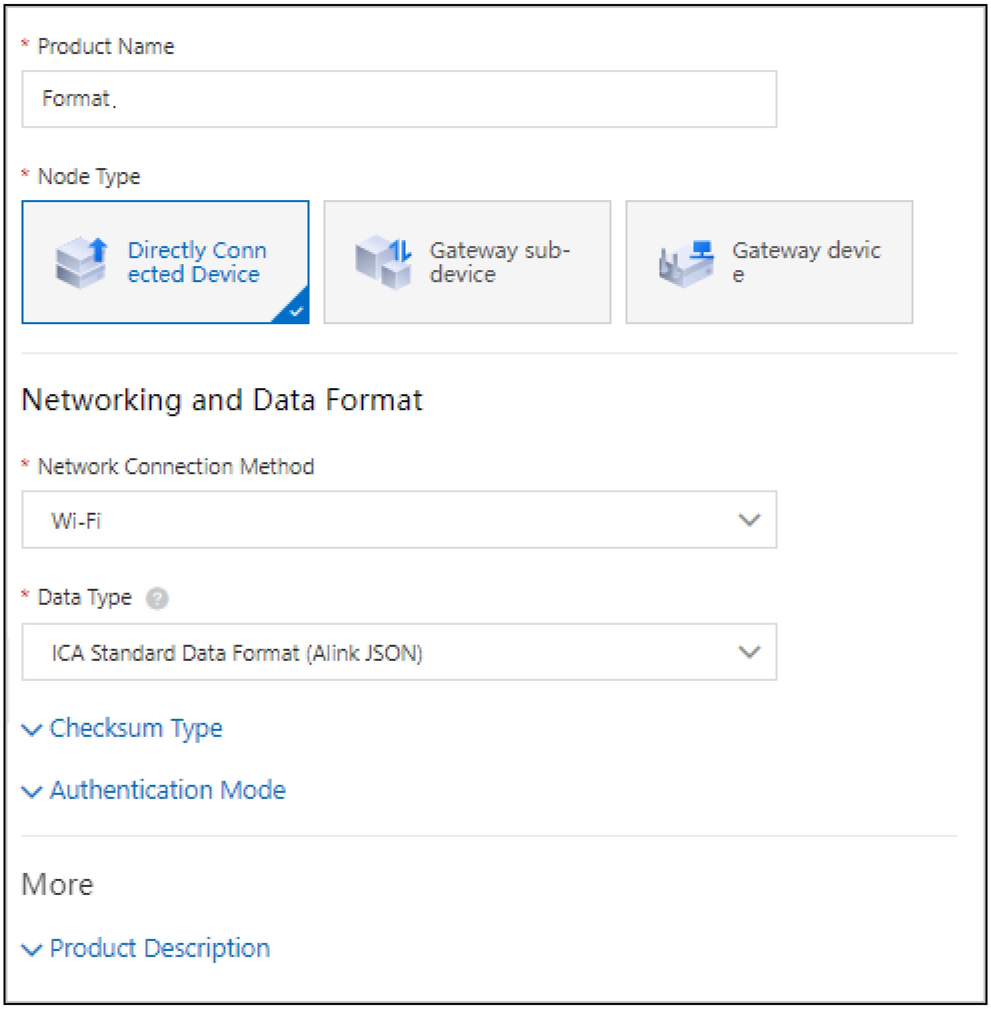
Send Data to Alibaba Cloud

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**.



* Product Name

Format

* Node Type

Networking and Data Format

* Network Connection Method

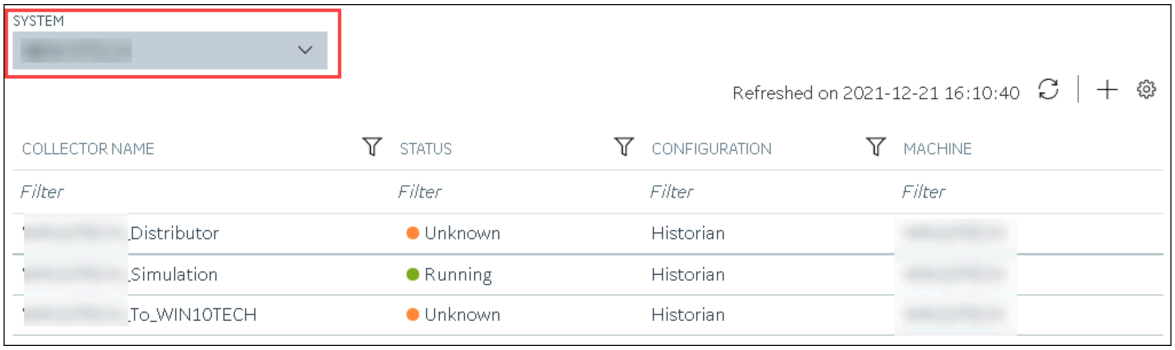
Wi-Fi

* Data Type

ICA Standard Data Format (Alink JSON)

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 97)*.
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.



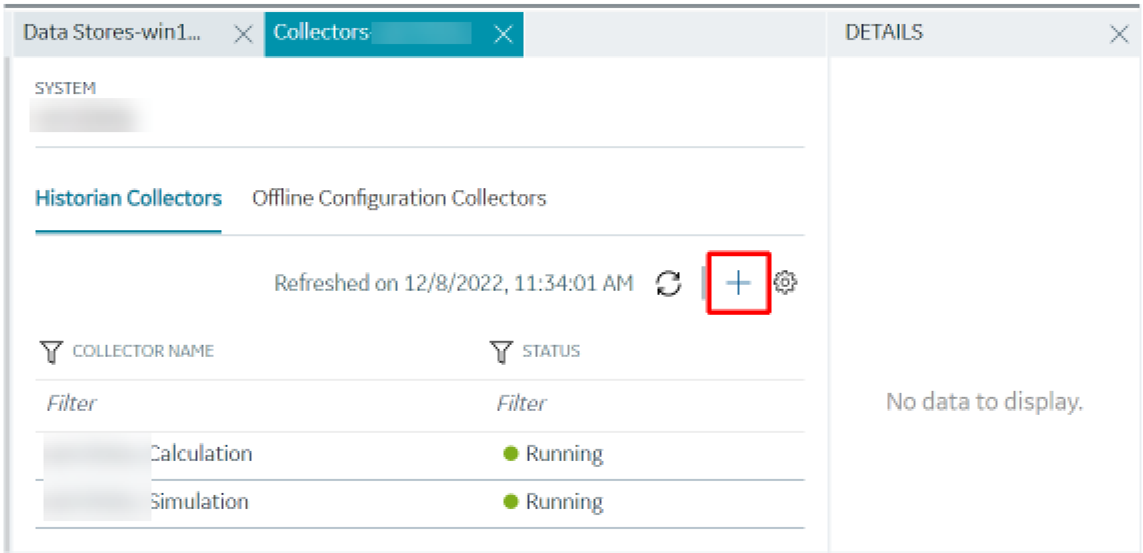
SYSTEM

[redacted]

Refreshed on 2021-12-21 16:10:40

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE
Filter	*Filter*	*Filter*	*Filter*
[redacted]Distributor	● Unknown	Historian	[redacted]
[redacted]Simulation	● Running	Historian	[redacted]
[redacted]To_WIN10TECH	● Unknown	Historian	[redacted]

9. In the upper-right corner of the main section, select +.



[redacted]

Refreshed on 12/8/2022, 11:34:01 AM

COLLECTOR NAME	STATUS
Filter	*Filter*
[redacted]Calculation	● Running
[redacted]Simulation	● Running

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**.

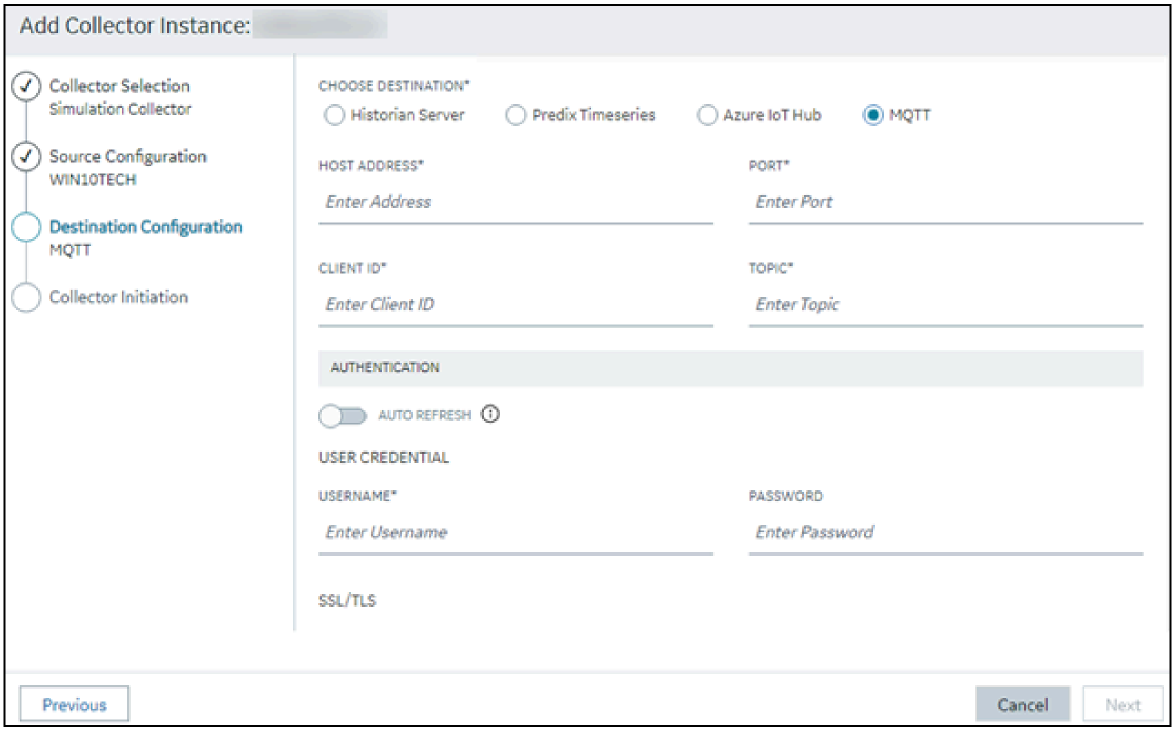
    The **INSTALLATION DRIVE** and **DATA DIRECTORY** fields are disabled and populated.

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.



Add Collector Instance:

- ✓ Collector Selection: Simulation Collector
- ✓ Source Configuration: WIN10TECH
- Destination Configuration: MQTT
- Collector Initiation

CHOOSE DESTINATION*: ○ Historian Server ○ Predix Timeseries ○ Azure IoT Hub ● MQTT

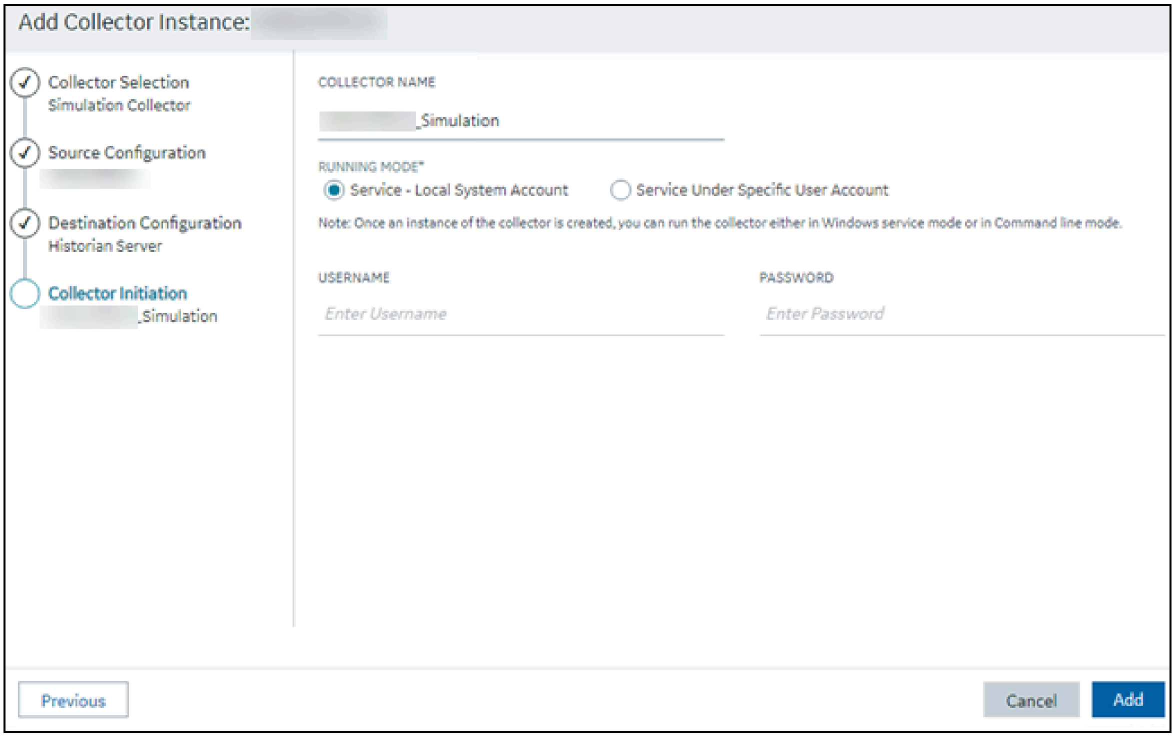
13. In the **CHOOSE DESTINATION** field, select **MQTT**, and then provide values as described in the following table.

Field	Description
HOST ADDRESS	Enter a value in the following format: `<product name>.iot-as-mqtt.<region ID>.aliyuncs.com`. A value is required. 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. ◦ For securemode, enter `2` for direct TLS connection, or enter `3` for direct TCP connection. ◦ 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. 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. 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. 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: ◦ **Historian Configuration**: Select this option if you want to add the tags manually using Historian Administrator *(on page 741)*. If you select this option, the **CONFIGURATION HISTORIAN SERVER** field appears. ◦ **Offline Configuration**: Select this option if you want to provide the tag names using the offline configuration *(on page)* 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.



Add Collector Instance: [redacted]

✓ Collector Selection
Simulation Collector

✓ Source Configuration
[redacted]

✓ Destination Configuration
Historian Server

○ Collector Initiation
[redacted]_Simulation

COLLECTOR NAME
[redacted]_Simulation

RUNNING MODE*
● Service - Local System Account ○ Service Under Specific User Account

Note: Once an instance of the collector is created, you can run the collector either in Windows service mode or in Command line mode.

USERNAME
Enter Username

PASSWORD
Enter Password

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 384)*.
    - If you have selected **Offline Configuration** in the **CHOOSE CONFIGURATION** field, specify the tags using the offline configuration file *(on page)*.

    The collector begins sending Historian data to the device that you have created.

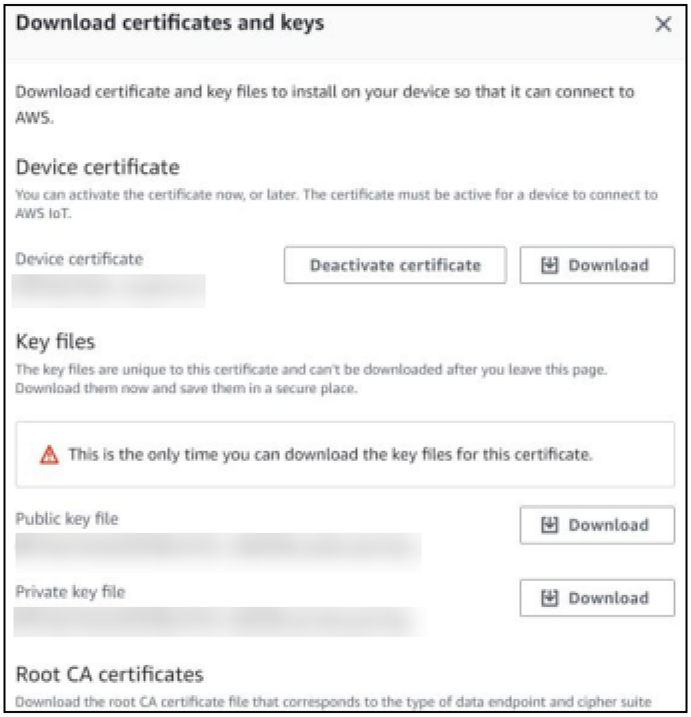
Send Data to AWS IoT Core

To send data to an AWS IoT Core, 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

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.



Download certificates and keys

Download certificate and key files to install on your device so that it can connect to AWS.

Device certificate

You can activate the certificate now, or later. The certificate must be active for a device to connect to AWS IoT.

Device certificate

Key files

The key files are unique to this certificate and can't be downloaded after you leave this page. Download them now and save them in a secure place.

This is the only time you can download the key files for this certificate.

Public key file

Private key file

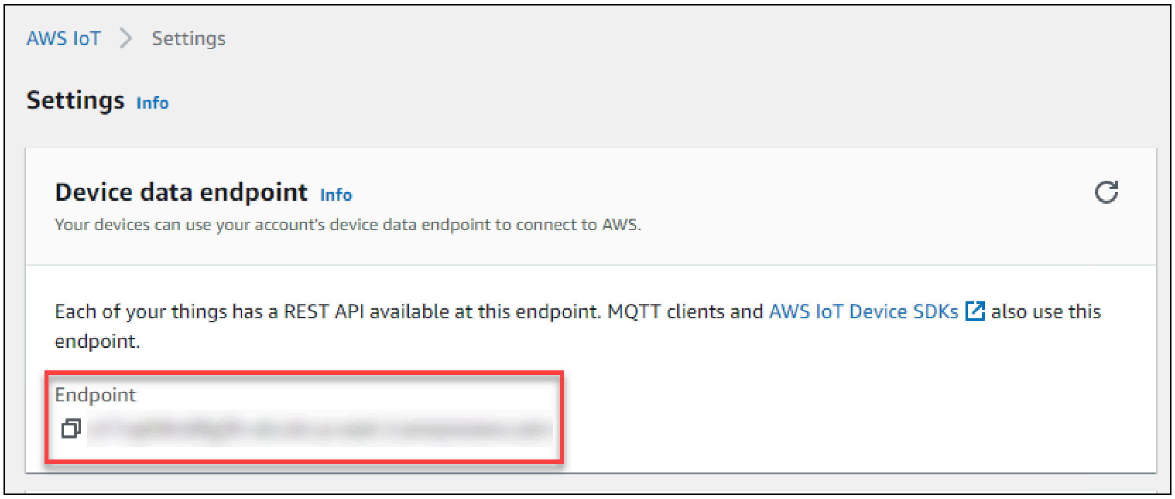
Root CA certificates

Download the root CA certificate file that corresponds to the type of data endpoint and cipher suite

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.



Settings Info

Device data endpoint Info

Your devices can use your account's device data endpoint to connect to AWS.

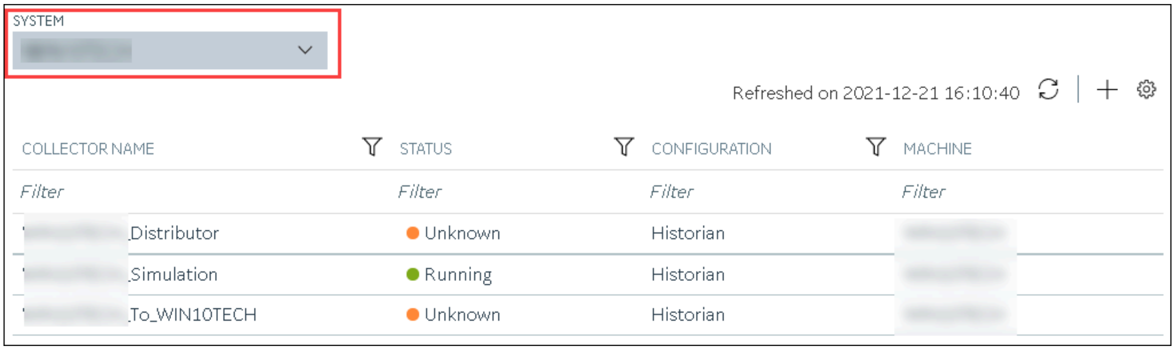
Each of your things has a REST API available at this endpoint. MQTT clients and AWS IoT Device SDKs also use this endpoint.

Endpoint

8. Access Configuration Hub *(on page 97)*.
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.

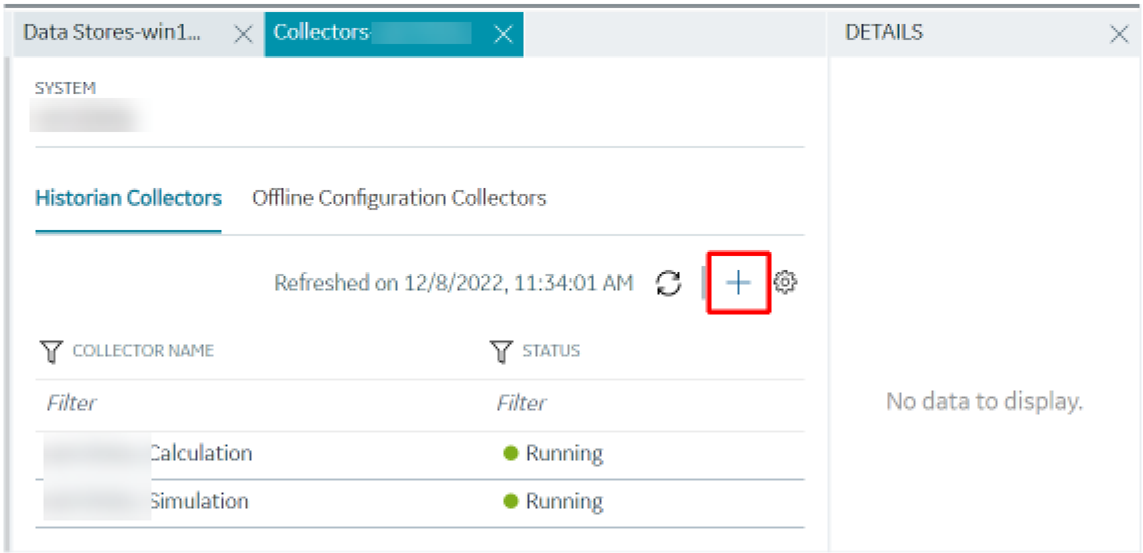


SYSTEM [redacted]

Refreshed on 2021-12-21 16:10:40

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE
Filter	*Filter*	*Filter*	*Filter*
[redacted]_Distributor	● Unknown	Historian	[redacted]
[redacted]_Simulation	● Running	Historian	[redacted]
[redacted]_To_WIN10TECH	● Unknown	Historian	[redacted]

11. In the upper-right corner of the main section, select +.



SYSTEM [redacted]

Refreshed on 12/8/2022, 11:34:01 AM

COLLECTOR NAME	STATUS
Filter	*Filter*
[redacted]Calculation	● Running
[redacted]Simulation	● Running

No data to display.

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**.

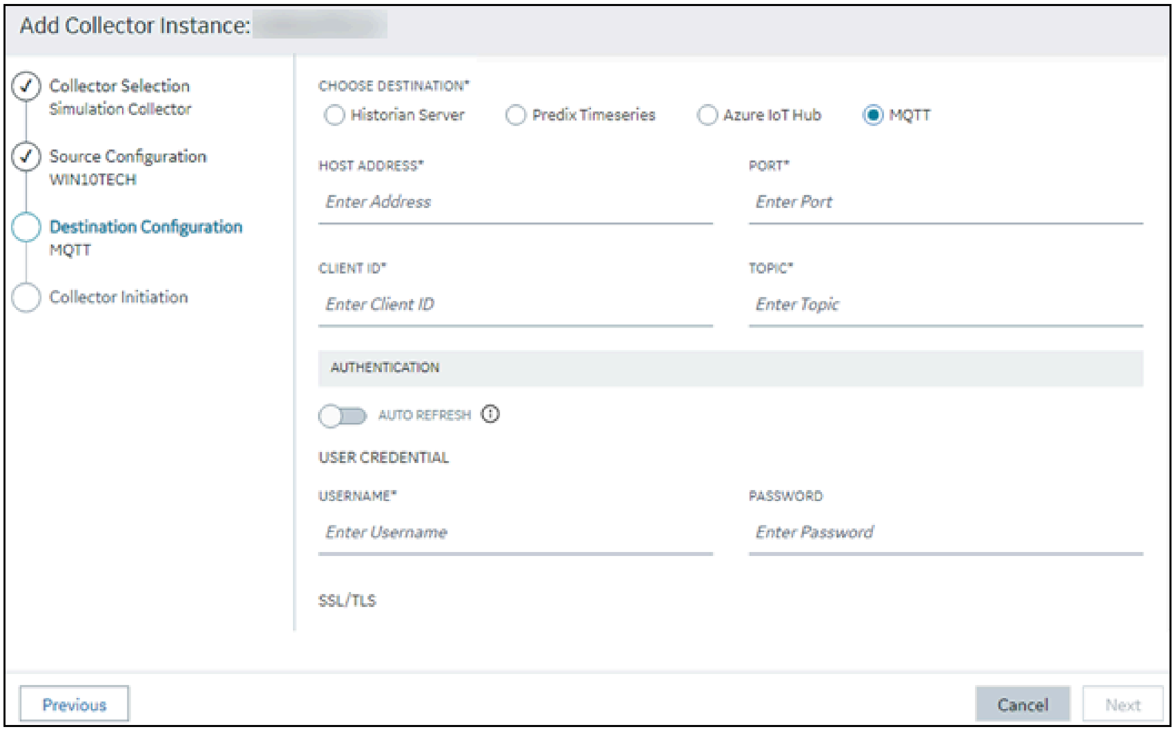
    The **INSTALLATION DRIVE** and **DATA DIRECTORY** fields are disabled and populated.

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.



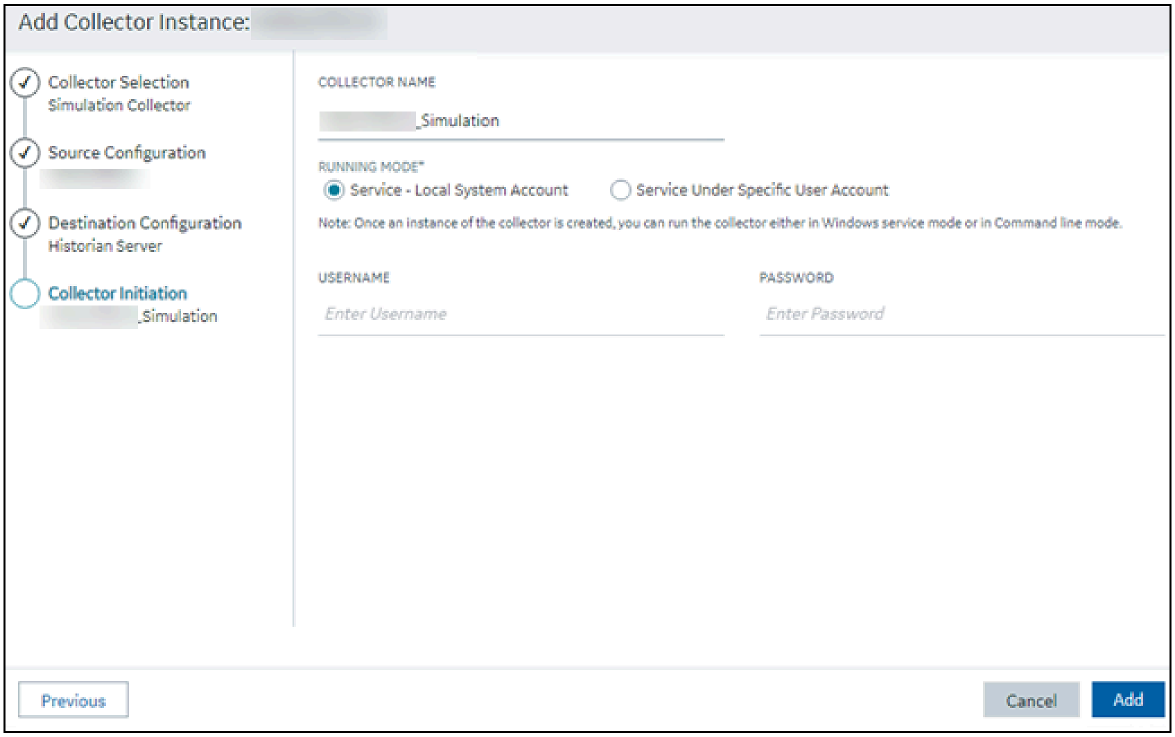
15. In the **CHOOSE DESTINATION** field, select **MQTT**, and then provide values as described in the following table.

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: ◦ **Historian Configuration**: Select this option if you want to add the tags manually using Historian Administrator *(on page 741)*. If you select this option, the **CONFIGURATION HISTORIAN SERVER** field appears. ◦ **Offline Configuration**: Select this option if you want to provide the tag names using the offline configuration *(on page)* 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.



Add Collector Instance: [redacted]

✓ Collector Selection
Simulation Collector

✓ Source Configuration
[redacted]

✓ Destination Configuration
Historian Server

○ Collector Initiation
[redacted]_Simulation

COLLECTOR NAME

[redacted]_Simulation

RUNNING MODE*

● Service - Local System Account ○ Service Under Specific User Account

Note: Once an instance of the collector is created, you can run the collector either in Windows service mode or in Command line mode.

USERNAME

Enter Username

PASSWORD

Enter Password

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**.

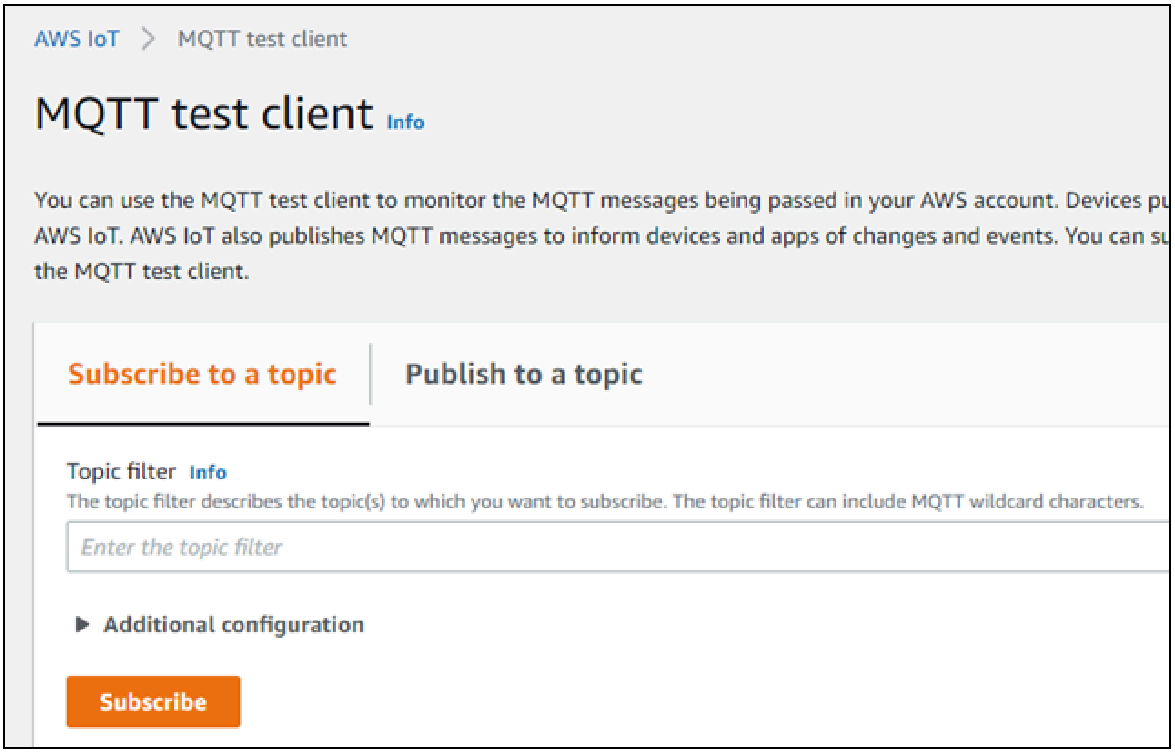
    The collector instance is created.

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 384)*.
    - If you have selected **Offline Configuration** in the **CHOOSE CONFIGURATION** field, specify the tags using the offline configuration file *(on page)*.

    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.

    AWS IoT > MQTT test client

    MQTT test client Info

    You can use the MQTT test client to monitor the MQTT messages being passed in your AWS account. Devices pu
    AWS IoT. AWS IoT also publishes MQTT messages to inform devices and apps of changes and events. You can su
    the MQTT test client.

    Topic filter Info

    The topic filter describes the topic(s) to which you want to subscribe. The topic filter can include MQTT wildcard characters.

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 722)* 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`

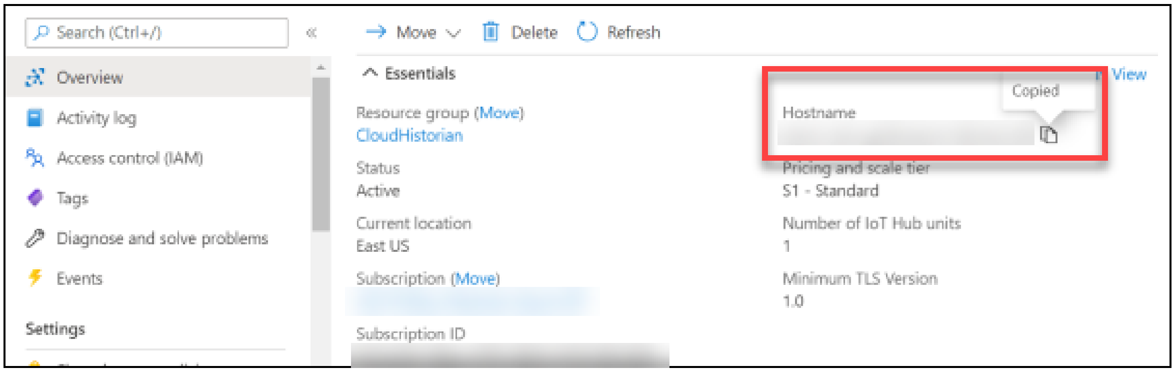
You can also send data in the KairosDB format *(on page 698)*.

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:



Resource group (Move)
CloudHistorian

Status
Active

Current location
East US

Subscription (Move)
[redacted]

Subscription ID
[redacted]

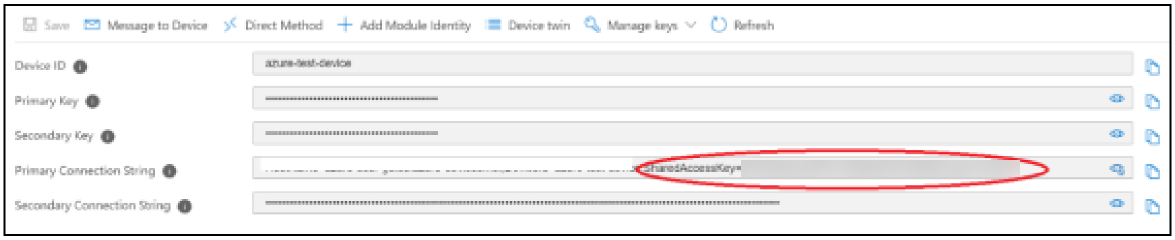
Hostname
[redacted]

Pricing and scale tier
S1 - Standard

Number of IoT Hub units
1

Minimum TLS Version
1.0

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.



Device ID: azure-test-device

Primary Key: ••

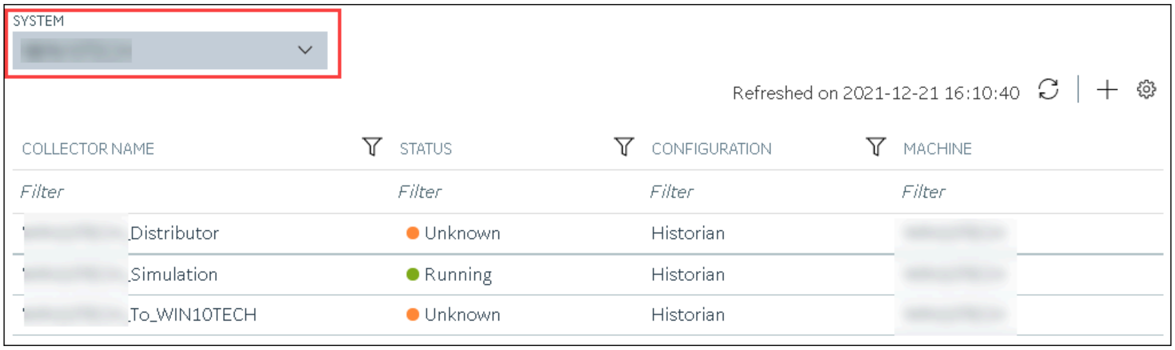
Secondary Key: ••

Primary Connection String: [redacted]SharedAccessKey=[redacted]

Secondary Connection String: ••

5. Access Configuration Hub *(on page 97)*.
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.

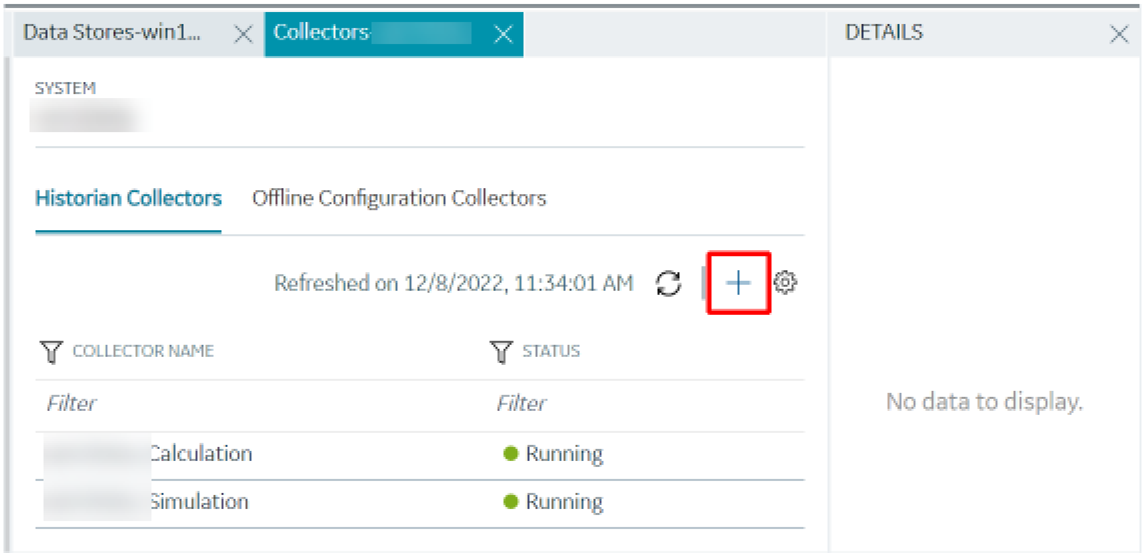


SYSTEM [redacted]

Refreshed on 2021-12-21 16:10:40

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE
Filter	*Filter*	*Filter*	*Filter*
[redacted]_Distributor	● Unknown	Historian	[redacted]
[redacted]_Simulation	● Running	Historian	[redacted]
[redacted]_To_WIN10TECH	● Unknown	Historian	[redacted]

9. In the upper-right corner of the main section, select +.



SYSTEM [redacted]

Refreshed on 12/8/2022, 11:34:01 AM

COLLECTOR NAME	STATUS
Filter	*Filter*
[redacted]Calculation	● Running
[redacted]Simulation	● Running

No data to display.

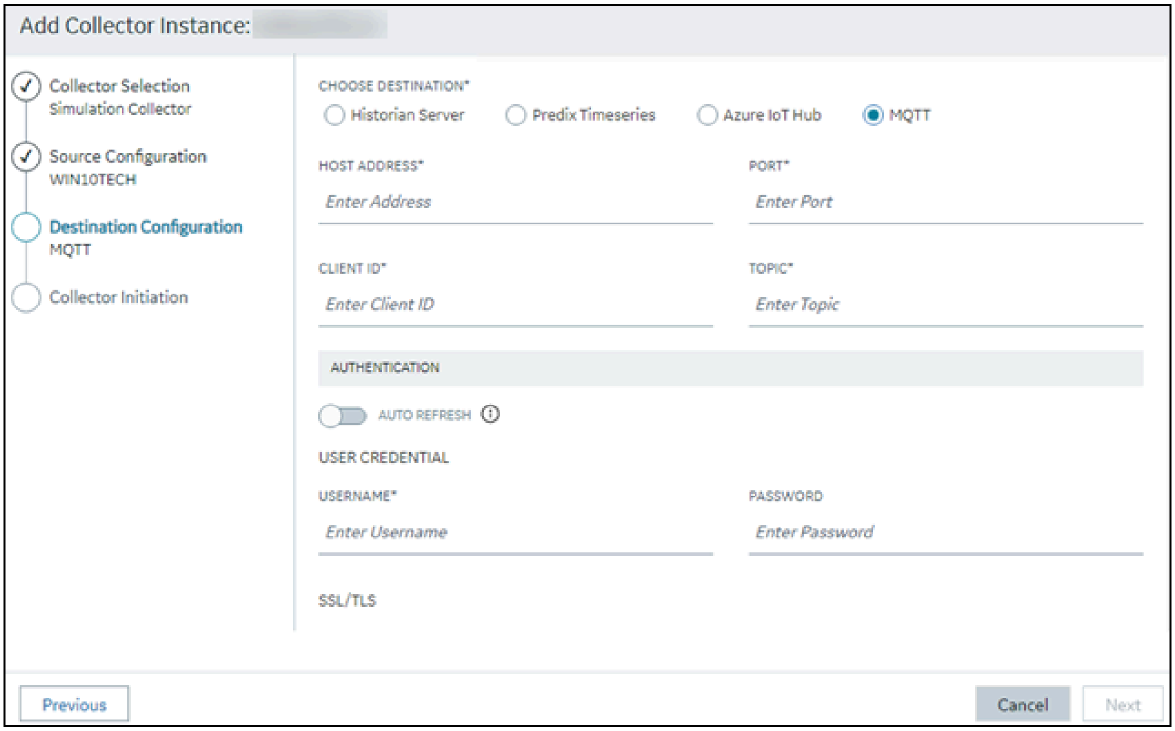
The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.

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.



Add Collector Instance: [redacted]

- ✓ Collector Selection
  Simulation Collector
- ✓ Source Configuration
  WIN10TECH
- ○ Destination Configuration
  MQTT
- ○ Collector Initiation

CHOOSE DESTINATION*
○ Historian Server ○ Predix Timeseries ○ Azure IoT Hub ● MQTT

HOST ADDRESS*: Enter Address
PORT*: Enter Port
CLIENT ID*: Enter Client ID
TOPIC*: Enter Topic

AUTHENTICATION

AUTO REFRESH ⓘ

USER CREDENTIAL

USERNAME*: Enter Username
PASSWORD: Enter Password

SSL/TLS

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.

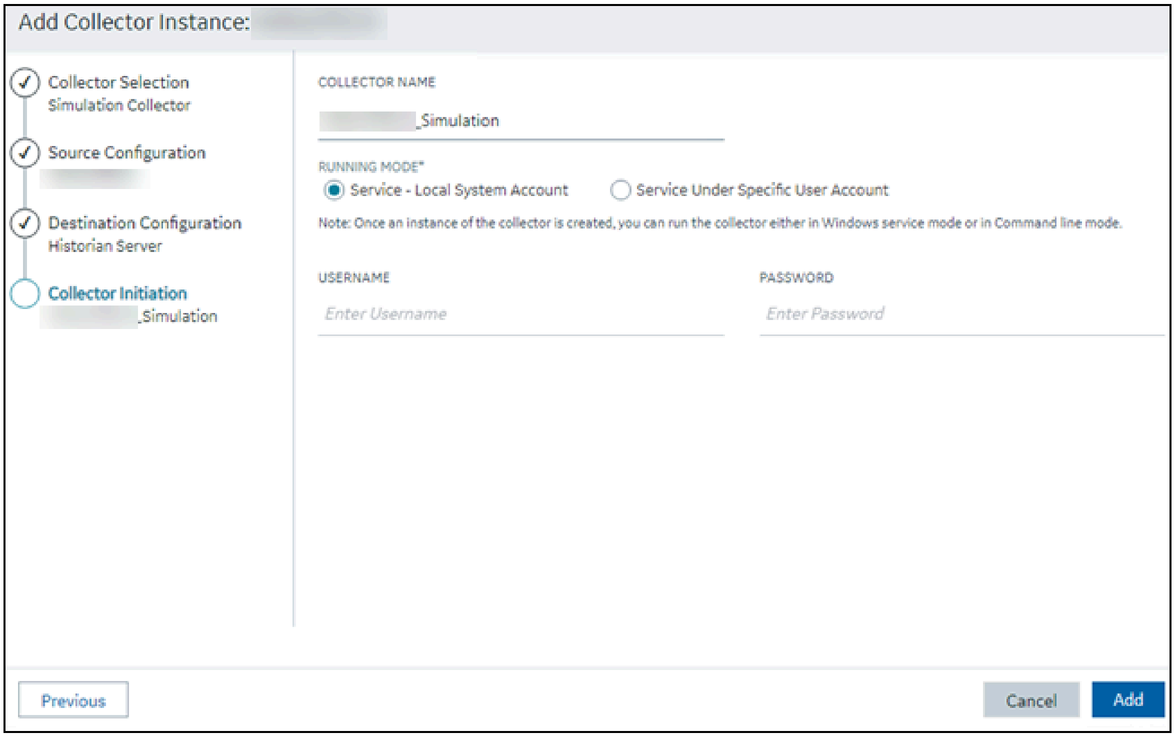
Field	Description
	◦ If you switch the toggle off, you must manually provide the token as soon as it expires. ◦ If you switch the toggle on, you must provide the shared access key that you have noted down in step 4. And, you can leave the **PASSWORD** field blank. This is applicable only if you have 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. 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. ◦ **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: ◦ **X.509 Self-Signed**: If you have selected this option, you can generate the key file using OpenSSL. ◦ **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: ◦ **X.509 Self-Signed**: If you have selected this option, you can generate the key file using OpenSSL. ◦ **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: ◦ **Historian Configuration**: Select this option if you want to add the tags manually *(on page 384)*. If you select this option, the **CONFIGURATION HISTORIAN SERVER** field appears. ◦ **Offline Configuration**: Select this option if you want to provide the tag names using the offline configuration *(on page*

Field	Description
	) 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.



Add Collector Instance: [redacted]

✓ Collector Selection
Simulation Collector

✓ Source Configuration
[redacted]

✓ Destination Configuration
Historian Server

○ Collector Initiation
[redacted]_Simulation

COLLECTOR NAME
[redacted]_Simulation

RUNNING MODE*
● Service - Local System Account ○ Service Under Specific User Account

Note: Once an instance of the collector is created, you can run the collector either in Windows service mode or in Command line mode.

USERNAME
Enter Username

PASSWORD
Enter Password

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 384)*.
    - If you have selected **Offline Configuration** in the **CHOOSE CONFIGURATION** field, specify the tags using the offline configuration file *(on page)*.

    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.

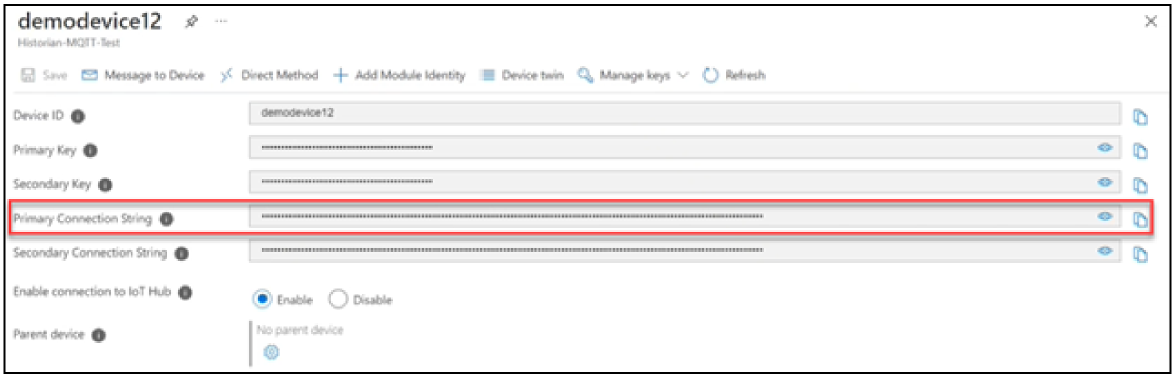
You can also send data in the key-value format *(on page 165)*.

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.



Device ID: demodevice12

Primary Key

Secondary Key

Primary Connection String

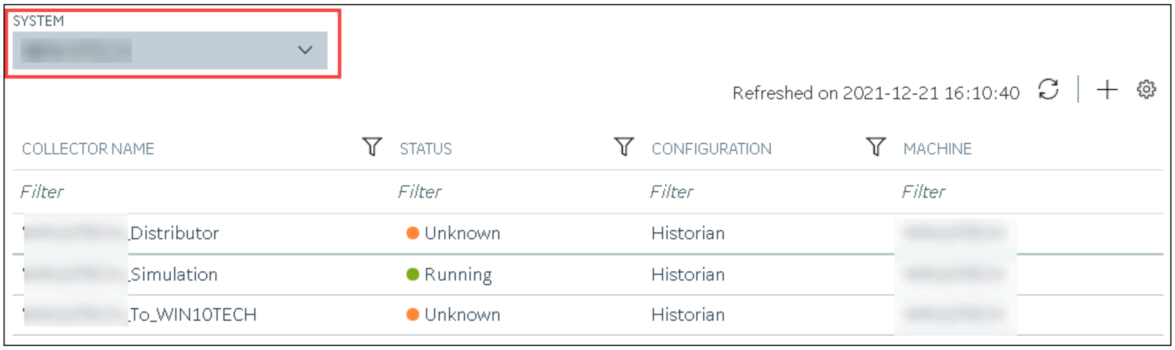
Secondary Connection String

Enable connection to IoT Hub: ● Enable ○ Disable

Parent device: No parent device

4. Access Configuration Hub *(on page 97)*.
5. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.

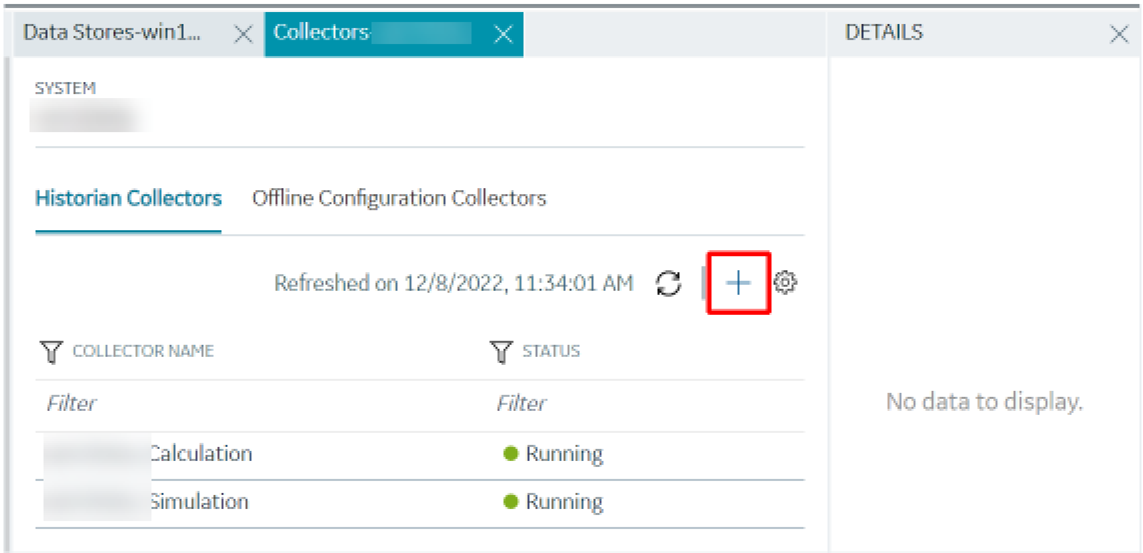
   A list of collectors in the default system appears.
6. If needed, select the system in which you want to add a collector instance.
7. If needed, select the system in which you want to add a collector instance.



Refreshed on 2021-12-21 16:10:40

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE
Filter	Filter	Filter	Filter
Distributor	● Unknown	Historian	
Simulation	● Running	Historian	
To_WIN10TECH	● Unknown	Historian	

8. In the upper-right corner of the main section, select + .



Refreshed on 12/8/2022, 11:34:01 AM

COLLECTOR NAME	STATUS
Filter	Filter
Calculation	● Running
Simulation	● Running

No data to display.

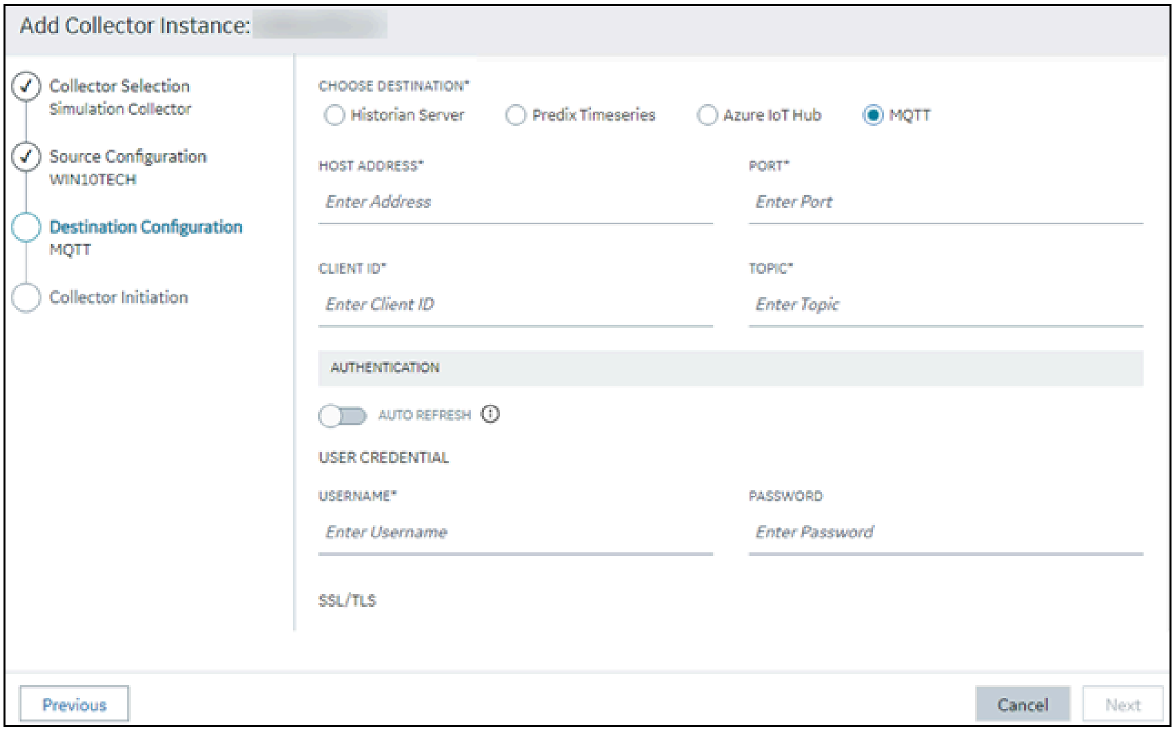
The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.

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.

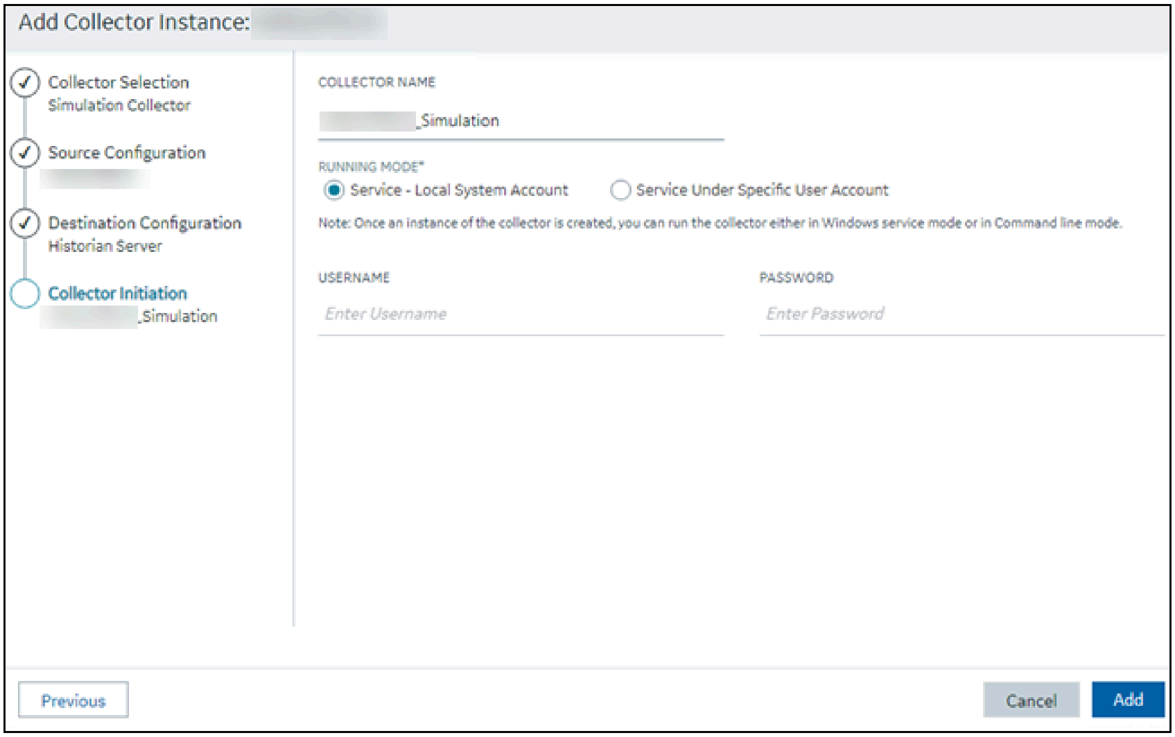
Field	Description
TRANSPORT PROTOCOL	The protocol that you want to use to send data to Azure IoT Hub. Select one of the following options: ◦ HTTP ◦ MQTT ◦ AMQP ◦ MQTT_OVER_WEBSOCKETS ◦ AMQP_OVER_WEBSOCKETS For information on which protocol to use, refer to Protocols and Port Numbers *(on page 184)*.
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: ◦ **Historian Configuration**: Select this option if you want to add the tags manually *(on page 384)*. If you select this option, the **CONFIGURATION HISTORIAN SERVER** field appears. ◦ **Offline Configuration**: Select this option if you want to provide the tag names us-

Field	Description
	ing the offline configuration *(on page)* 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.



Add Collector Instance: [redacted]

- Collector Selection: Simulation Collector
- Source Configuration: [redacted]
- Destination Configuration: Historian Server
- Collector Initiation: [redacted]_Simulation

COLLECTOR NAME: [redacted]_Simulation

RUNNING MODE*: ● Service - Local System Account ○ Service Under Specific User Account

Note: Once an instance of the collector is created, you can run the collector either in Windows service mode or in Command line mode.

USERNAME: Enter Username

PASSWORD: Enter Password

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 384)*.
    - If you have selected **Offline Configuration** in the **CHOOSE CONFIGURATION** field, specify the tags using the offline configuration file *(on page)*.

    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.

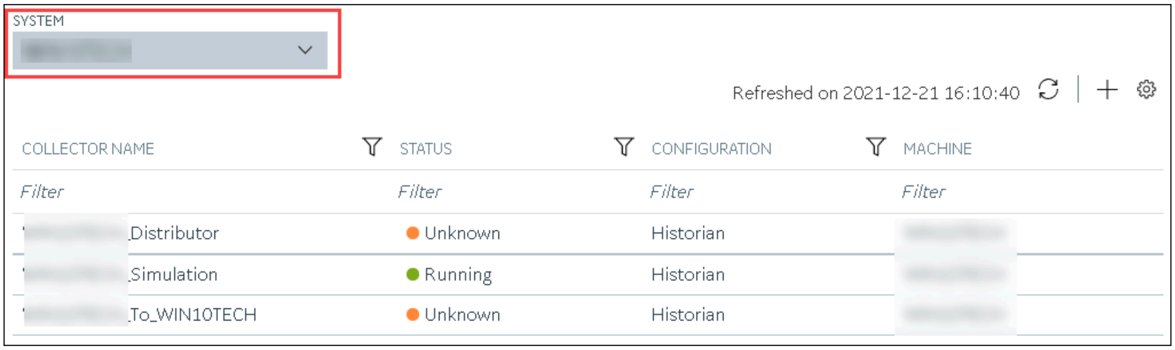
   Note down the registry ID and the region values.
4. Add a device to the registry.

   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 97)*.
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.



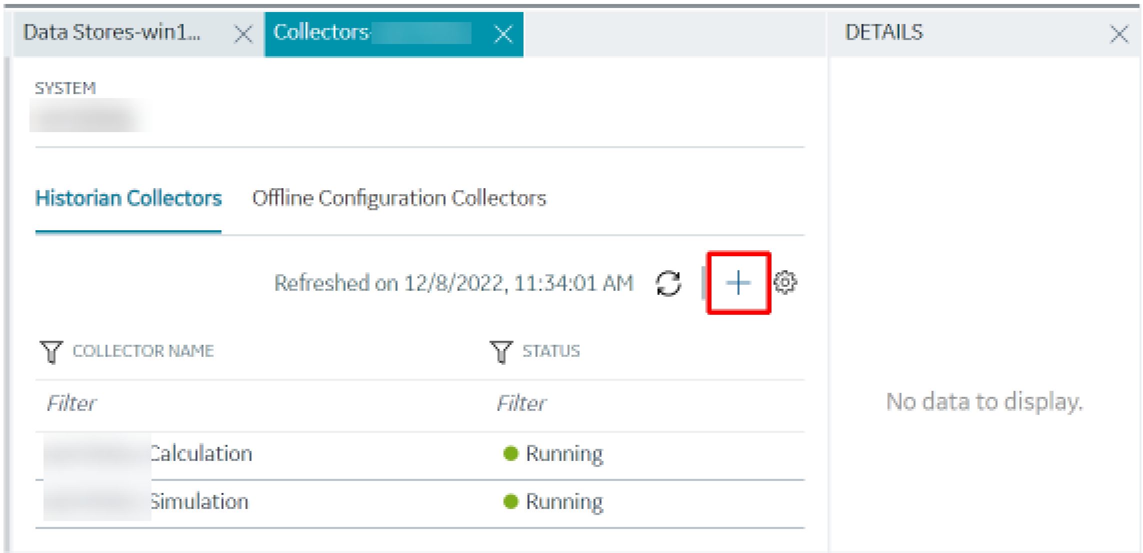
SYSTEM

[redacted]

Refreshed on 2021-12-21 16:10:40

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE
Filter	*Filter*	*Filter*	*Filter*
[redacted]_Distributor	● Unknown	Historian	[redacted]
[redacted]_Simulation	● Running	Historian	[redacted]
[redacted]_To_WIN10TECH	● Unknown	Historian	[redacted]

9. In the upper-right corner of the main section, select +.



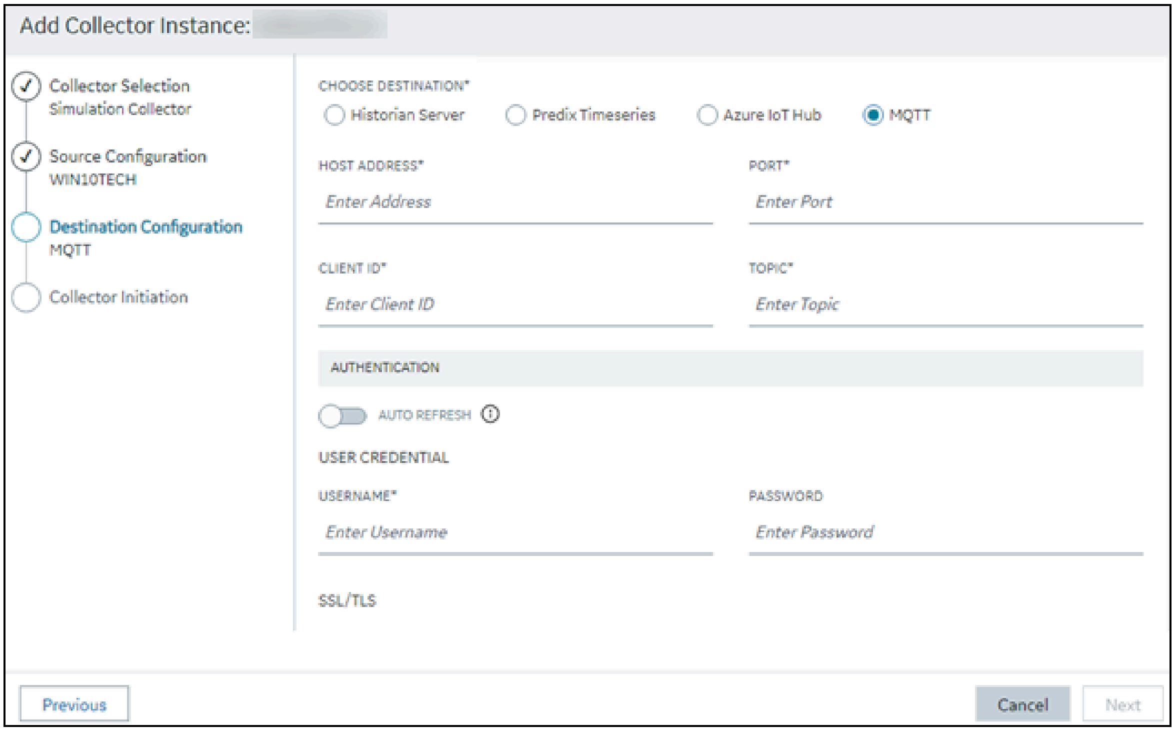
The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.

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.



Add Collector Instance: [redacted]

- ✓ Collector Selection
  Simulation Collector
- ✓ Source Configuration
  WIN10TECH
- ○ Destination Configuration
  MQTT
- ○ Collector Initiation

CHOOSE DESTINATION*
○ Historian Server ○ Predix Timeseries ○ Azure IoT Hub ● MQTT

HOST ADDRESS* Enter Address

PORT* Enter Port

CLIENT ID* Enter Client ID

TOPIC* Enter Topic

AUTHENTICATION

AUTO REFRESH ⓘ

USER CREDENTIAL

USERNAME* Enter Username

PASSWORD Enter Password

SSL/TLS

14. Select **MQTT**, and provide values as described in the following table.

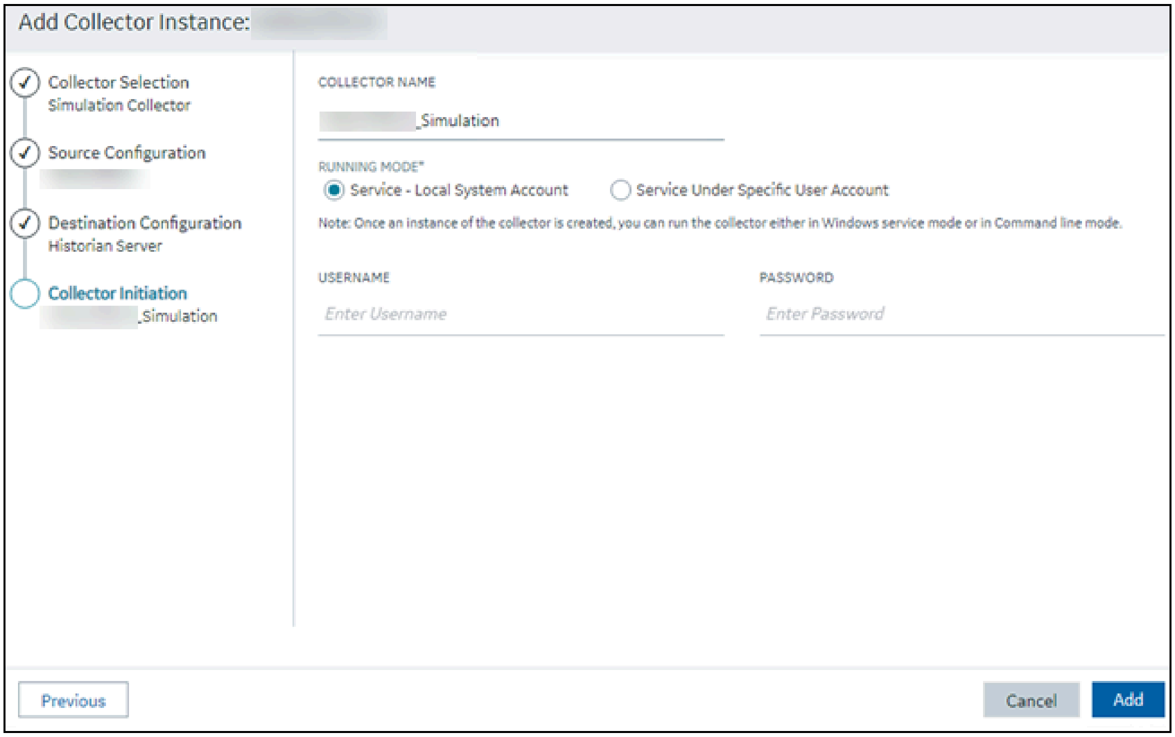
Field	Description
HOST ADDRESS	Enter `mqtt.googleapis.com`. A value is required.
PORT	Enter `8883` or `443`.
CLIENT ID	Enter the ID of the device that you created in the following format: `projects/<project ID>/locations/<cloud region>/registries/<registry ID>/devices/<device ID>`. For example: `projects/mygcpproject/locations/asia-east1/registries/testmqttgcpiot/devices/gcptesting` 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: ◦ **Historian Configuration**: Select this option if you want to add the tags manually *(on page 384)*. If you select this option, the **CONFIGURATION HISTORIAN SERVER** field appears. ◦ **Offline Configuration**: Select this option if you want to provide the tag names using the offline configuration *(on page)* 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.



Add Collector Instance: [redacted]

✓ Collector Selection
Simulation Collector

✓ Source Configuration
[redacted]

✓ Destination Configuration
Historian Server

○ Collector Initiation
[redacted]_Simulation

COLLECTOR NAME
[redacted]_Simulation

RUNNING MODE*
● Service - Local System Account ○ Service Under Specific User Account

Note: Once an instance of the collector is created, you can run the collector either in Windows service mode or in Command line mode.

USERNAME
Enter Username

PASSWORD
Enter Password

16. Enter a collector name.

    The value that you enter:

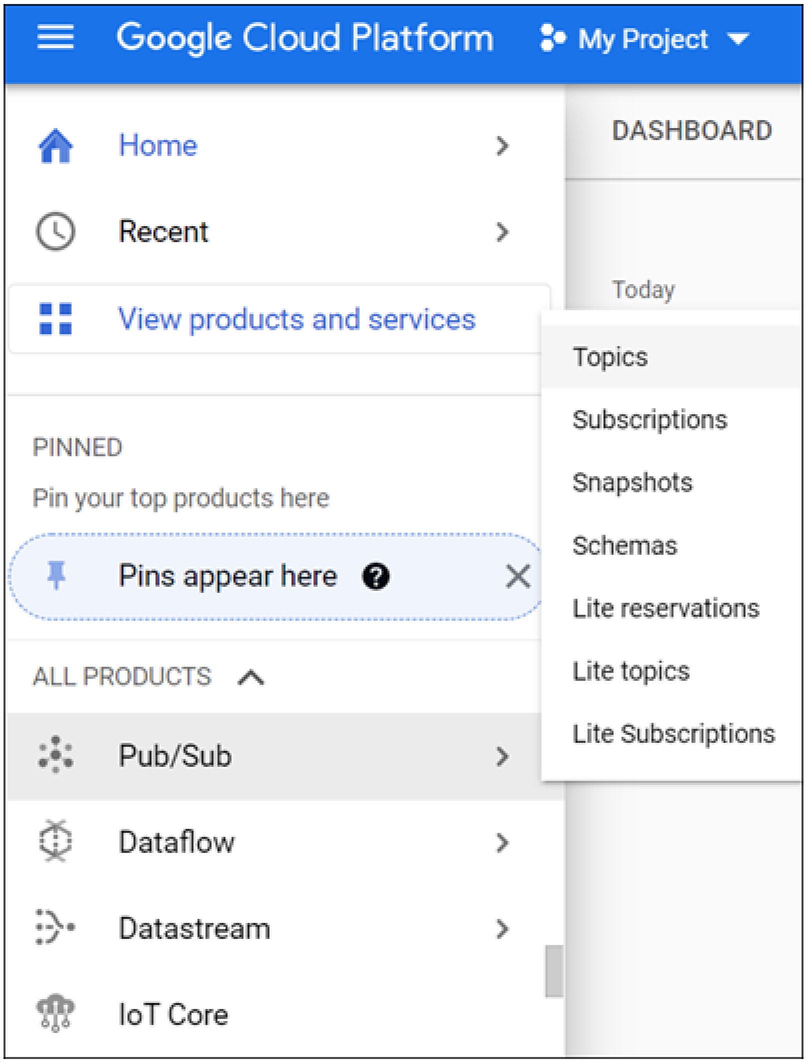
    - Must be unique.
    - Must not exceed 15 characters.
    - Must not contain a space.
    - Must not contain special characters except a hyphen, period, and an underscore.

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:

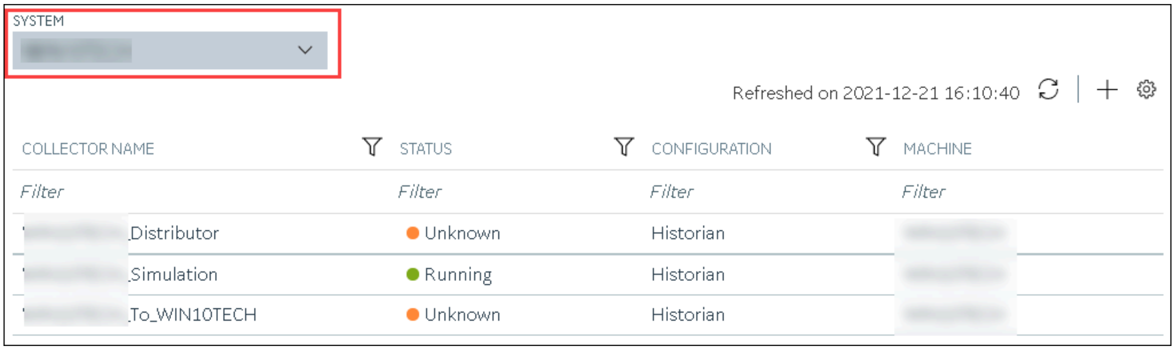
- The iFIX collector
- The MQTT collector
- The ODBC collector
- The OPC Classic DA collector
- The OPC Classic HDA collector

- 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 Proficy Authentication 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 97)*.
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.

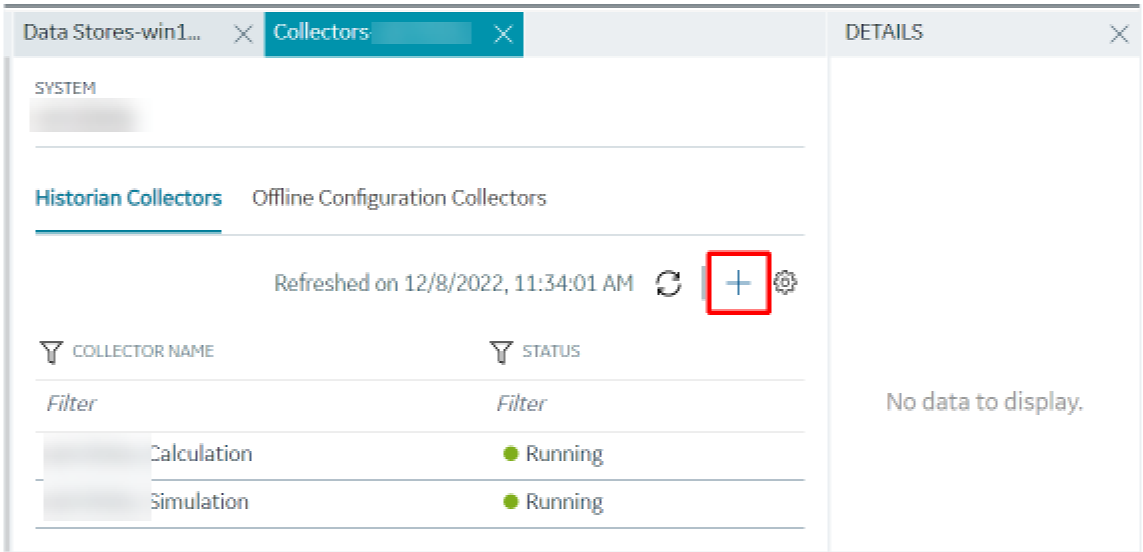


SYSTEM [redacted]

Refreshed on 2021-12-21 16:10:40

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE
Filter	*Filter*	*Filter*	*Filter*
[redacted]Distributor	● Unknown	Historian	[redacted]
[redacted]Simulation	● Running	Historian	[redacted]
[redacted]To_WIN10TECH	● Unknown	Historian	[redacted]

5. In the upper-right corner of the main section, select ＋.



SYSTEM [redacted]

Refreshed on 12/8/2022, 11:34:01 AM

COLLECTOR NAME	STATUS
Filter	*Filter*
[redacted]Calculation	● Running
[redacted]Simulation	● Running

No data to display.

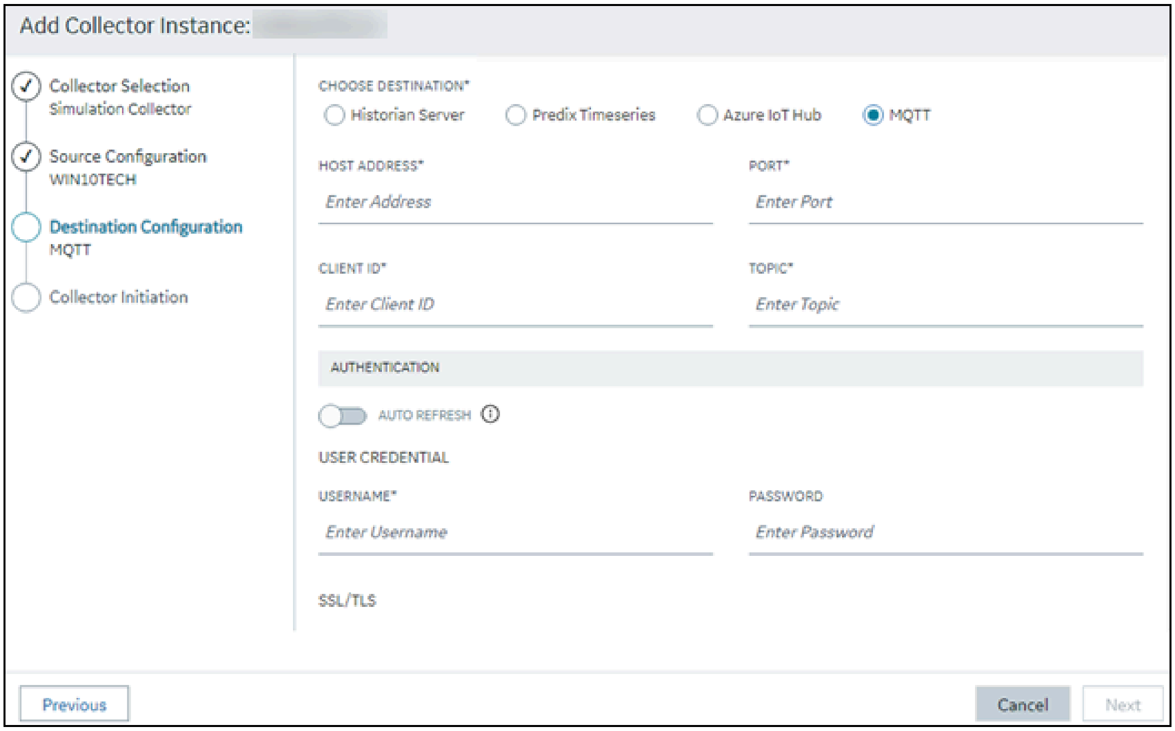
The **Add Collector Instance: <system name>** window appears, displaying the **Collector Selection** section. The **MACHINE NAME** field contains a list of machines on which you have installed collectors.

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.

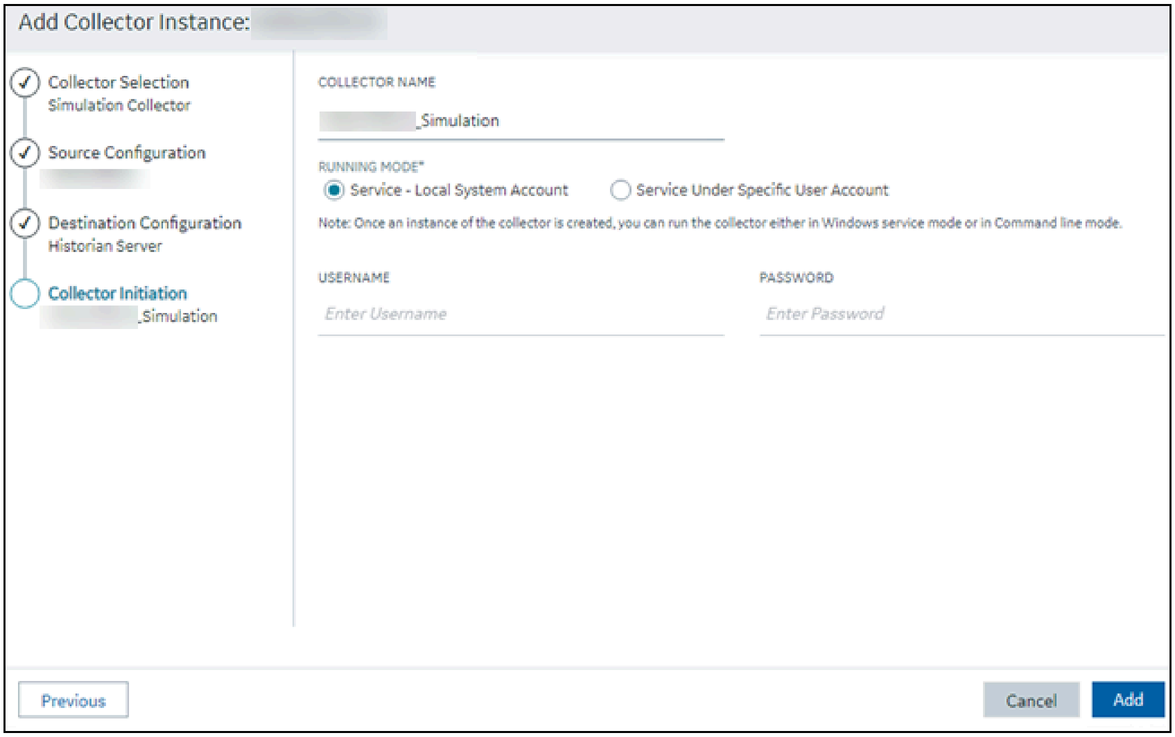
Field	Description
CLOUD DESTINATION ADDRESS	The URL of a data streaming endpoint exposed by the Predix Time Series instance to which you want to send data. Typically, it starts with “wss://”. This value is used as part of the interface name and default tag prefix of the collector. Your Predix Time Series 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: ◦ **Historian Configuration**: Select this option if you want to add the tags manually *(on page 384)*. If you select this option, the **CONFIGURATION HISTORIAN SERVER** field appears. ◦ **Offline Configuration**: Select this option if you want to provide the tag names using the offline configuration *(on page)* 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.



Add Collector Instance: [redacted]

✓ Collector Selection
Simulation Collector

✓ Source Configuration
[redacted]

✓ Destination Configuration
Historian Server

○ Collector Initiation
[redacted]_Simulation

COLLECTOR NAME
[redacted]_Simulation

RUNNING MODE*
● Service - Local System Account ○ Service Under Specific User Account

Note: Once an instance of the collector is created, you can run the collector either in Windows service mode or in Command line mode.

USERNAME
Enter Username

PASSWORD
Enter Password

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 384)*.
    - If you have selected **Offline Configuration** in the **CHOOSE CONFIGURATION** field, specify the tags using the offline configuration file *(on page)*.

    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 142)*.

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 556)*. You can also add a collector instance using the RemoteCollectorConfigurator utility *(on page 1335)*, which does not require you to install Web-based Clients.

Access a Collector Instance

1. Access Configuration Hub *(on page 97)*.
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: ◦ **Started** ◦ **Stopped** ◦ **Running** ◦ **Paused**
CONFIGURATION	The source of the tag configuration for the collector. Contains one of the following values: ◦ **HISTORIAN**: Indicates that tags are configured using Historian Administrator. ◦ **OFFLINE**: Indicates that tags are configured using an offline configuration *(on page)* 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.
TAG COUNT	The number of tags for which the collector collects data.
COMMENTS	The comments that you have entered for the collector.

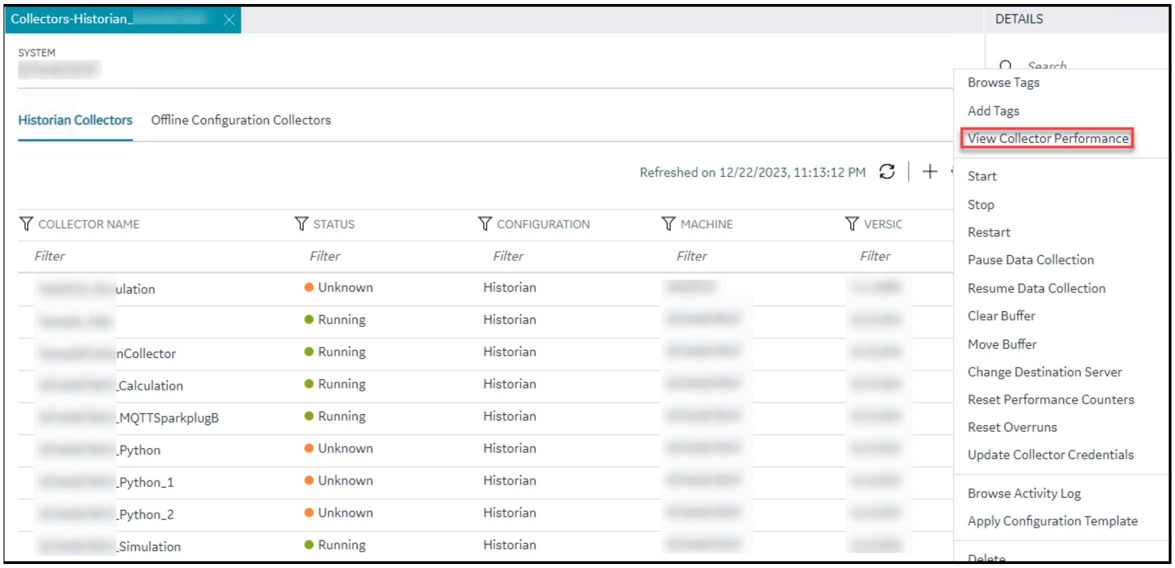
Tip:
You can also add, reorder, and remove columns from the table. For instructions, refer to Common Tasks in Configuration Hub *(on page 114)*.

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**.


Collectors-Historian_[redacted]
DETAILS
SYSTEM
[redacted]
Search
Historian Collectors
Offline Configuration Collectors
Refreshed on 12/22/2023, 11:13:12 PM
Browse Tags
Add Tags
View Collector Performance
Start
Stop
Restart
Pause Data Collection
Resume Data Collection
Clear Buffer
Move Buffer
Change Destination Server
Reset Performance Counters
Reset Overruns
Update Collector Credentials
Browse Activity Log
Apply Configuration Template
Delete

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSIC
Filter	Filter	Filter	Filter	Filter
[redacted]ulation	● Unknown	Historian	[redacted]	[redacted]
[redacted]	● Running	Historian	[redacted]	[redacted]
[redacted]nCollector	● Running	Historian	[redacted]	[redacted]
[redacted]_Calculation	● Running	Historian	[redacted]	[redacted]
[redacted]_MQTTSparkplugB	● Running	Historian	[redacted]	[redacted]
[redacted]_Python	● Unknown	Historian	[redacted]	[redacted]
[redacted]_Python_1	● Unknown	Historian	[redacted]	[redacted]
[redacted]_Python_2	● Unknown	Historian	[redacted]	[redacted]
[redacted]_Simulation	● Running	Historian	[redacted]	[redacted]

The **Collector Performance** section appears, displaying the following graphs:

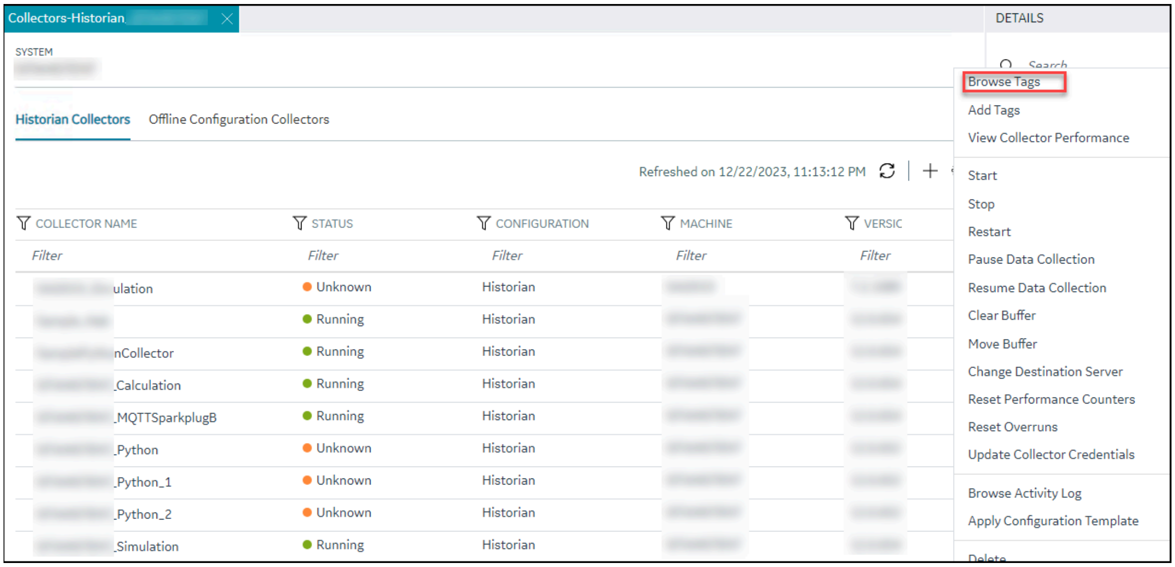
- **REPORT RATE**: The rate at which the collector collects data.
- **TOTAL EVENTS REPORTED**: The total number of events reported to the Historian archive from the collector. This number may not match the total events collected due to collector compression.
- **STATUS**: The status of the collector 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 97)*.
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**.



Collectors-Historian [redacted]
DETAILS
SYSTEM
[redacted]
Historian Collectors
Offline Configuration Collectors
Search
Browse Tags
Add Tags
View Collector Performance
Start
Stop
Restart
Pause Data Collection
Resume Data Collection
Clear Buffer
Move Buffer
Change Destination Server
Reset Performance Counters
Reset Overruns
Update Collector Credentials
Browse Activity Log
Apply Configuration Template
Delete

Refreshed on 12/22/2023, 11:13:12 PM

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSIC
Filter	Filter	Filter	Filter	Filter
[redacted]ulation	● Unknown	Historian	[redacted]	[redacted]
[redacted]	● Running	Historian	[redacted]	[redacted]
[redacted]nCollector	● Running	Historian	[redacted]	[redacted]
[redacted]_Calculation	● Running	Historian	[redacted]	[redacted]
[redacted]_MQTTSparkplugB	● Running	Historian	[redacted]	[redacted]
[redacted]_Python	● Unknown	Historian	[redacted]	[redacted]
[redacted]_Python_1	● Unknown	Historian	[redacted]	[redacted]
[redacted]_Python_2	● Unknown	Historian	[redacted]	[redacted]
[redacted]_Simulation	● Running	Historian	[redacted]	[redacted]

   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 595)*
- The iFIX collector *(on page 602)*
- The MQTT collector *(on page 349)*
- The ODBC collector *(on page 361)*
- The OPC Classic DA collector *(on page 375)*
- The OPC Classic HDA collector *(on page 393)*
- The OPC UA DA collector *(on page 407)*
- The OSI PI collector *(on page 644)*
- The OSI PI distributor *(on page 648)*
- The Python Collector *(on page 442)*
- The Server-to-Server collector *(on page 654)*
- The Server-to-Server distributor *(on page 658)*
- The Simulation collector *(on page 662)*
- The Wonderware collector *(on page 665)*

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 1337)*, which does not require you to install Web-based Clients.

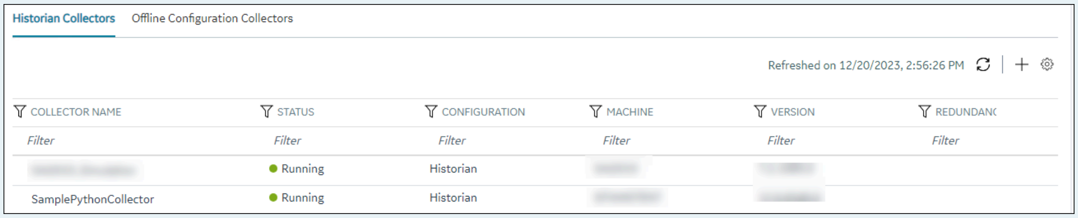
Note:

- 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 97)*.
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.



Historian Collectors | Offline Configuration Collectors

Refreshed on 12/20/2023, 2:56:26 PM

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSION	REDUNDANC
Filter	Filter	Filter	Filter	Filter	Filter
[redacted]	● Running	Historian	[redacted]	[redacted]	
SamplePythonCollector	● Running	Historian	[redacted]	[redacted]	

3. Select the collector instance that you want to modify.

   The details of the collector appear in the **DETAILS** section.

   Tip:

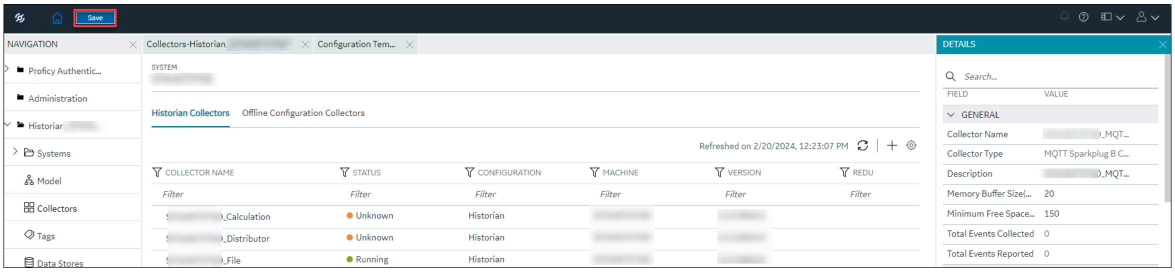
   If the **DETAILS** section does not appear, in the upper-right corner of the page, select ◧, and then select **Details**.

4. As needed, modify values in the available fields *(on page 669)*.

   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**.



COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSION	REDU
Filter	Filter	Filter	Filter	Filter	Filter
[redacted]_Calculation	● Unknown	Historian	[redacted]	[redacted]	
[redacted]_Distributor	● Unknown	Historian	[redacted]	[redacted]	
[redacted]_File	● Running	Historian	[redacted]	[redacted]	

DETAILS

FIELD	VALUE
GENERAL	
Collector Name	[redacted]_MQT...
Collector Type	MQTT Sparkplug B C...
Description	[redacted]_MQT...
Memory Buffer Size(...	20
Minimum Free Space...	150
Total Events Collected	0
Total Events Reported	0

   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.

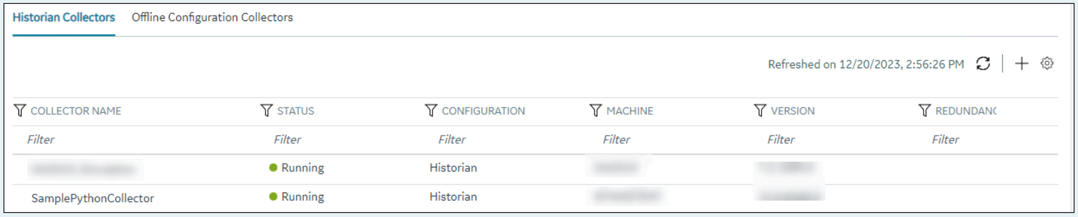
Add a Comment to a Collector Instance

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 97)*.
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.
   >
   > 
   >
   > Historian Collectors | Offline Configuration Collectors
   >
   > Refreshed on 12/20/2023, 2:56:26 PM
   >
   > | COLLECTOR NAME | STATUS | CONFIGURATION | MACHINE | VERSION | REDUNDANC |
   > |---|---|---|---|---|---|
   > | *Filter* | *Filter* | *Filter* | *Filter* | *Filter* | *Filter* |
   > | [redacted] | ● Running | Historian | [redacted] | [redacted] | |
   > | SamplePythonCollector | ● Running | Historian | [redacted] | [redacted] | |

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**.

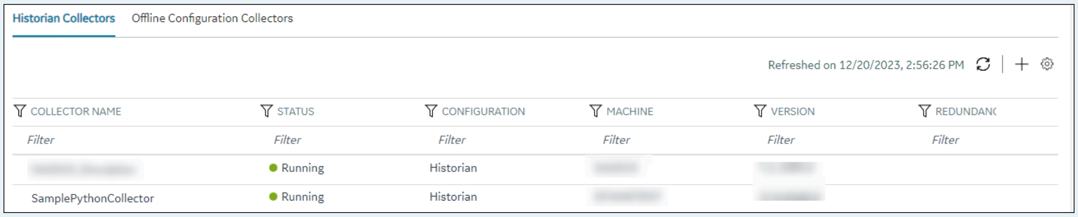
4. In the **DETAILS** section, in the text box below **Comments**, enter your comment, and then select **Add Comment**.

   The comment is added to the collector instance.

Access a Comment on a Collector Instance

1. Access Configuration Hub *(on page 97)*.
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.
   >
   > 
   >
   > Historian Collectors | Offline Configuration Collectors
   >
   > Refreshed on 12/20/2023, 2:56:26 PM
   >
   > | COLLECTOR NAME | STATUS | CONFIGURATION | MACHINE | VERSION | REDUNDANC |
   > |---|---|---|---|---|---|
   > | Filter | Filter | Filter | Filter | Filter | Filter |
   > | [redacted] | ● Running | Historian | [redacted] | [redacted] | |
   > | SamplePythonCollector | ● Running | Historian | [redacted] | [redacted] | |

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

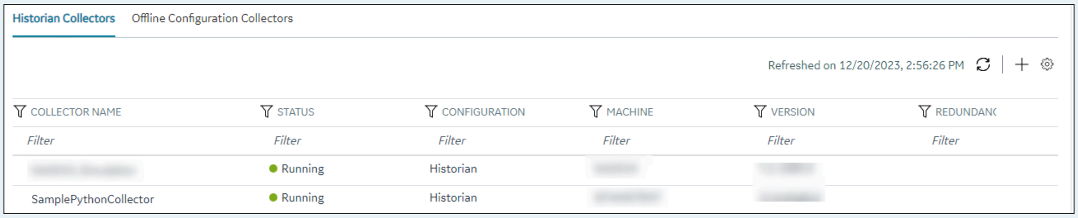
You can start a collector using one of the following options:

- **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 97)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.

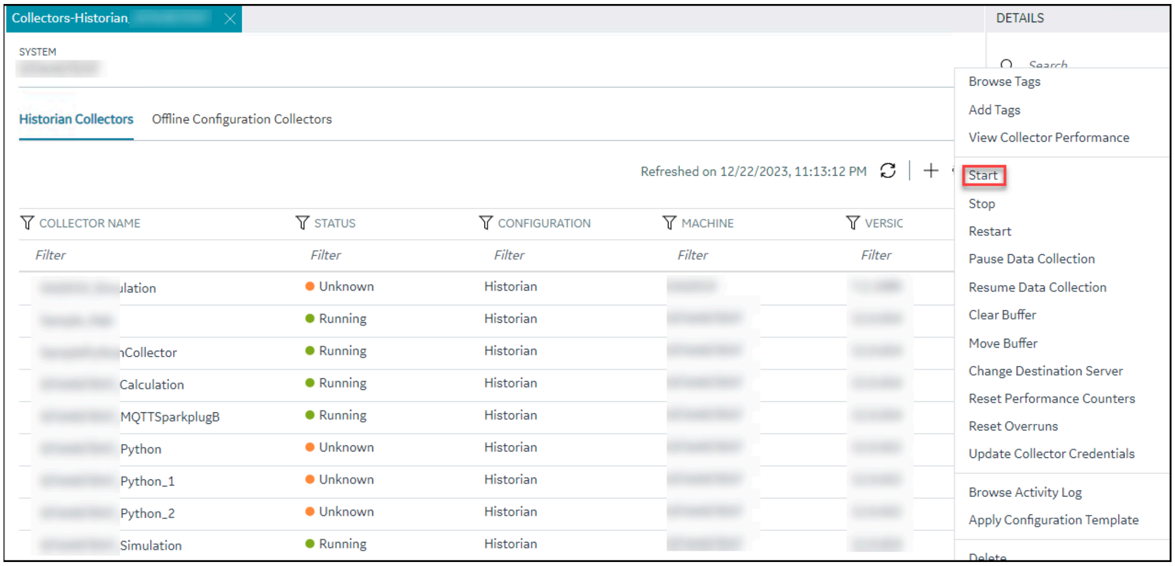
   A list of collectors appears.

> **Tip:**
>
> You can filter the collectors by the system name.


Historian Collectors
Offline Configuration Collectors
Refreshed on 12/20/2023, 2:56:26 PM

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSION	REDUNDANC
Filter	Filter	Filter	Filter	Filter	Filter
[redacted]	● Running	Historian	[redacted]	[redacted]	
SamplePythonCollector	● Running	Historian	[redacted]	[redacted]	

3. Right-click the collector instance that you want to start (or select ⋯), and then select **Start**.


Collectors-Historian [redacted]
DETAILS
SYSTEM
[redacted]
Search
Historian Collectors
Offline Configuration Collectors
Refreshed on 12/22/2023, 11:13:12 PM
Browse Tags
Add Tags
View Collector Performance
Start
Stop
Restart
Pause Data Collection
Resume Data Collection
Clear Buffer
Move Buffer
Change Destination Server
Reset Performance Counters
Reset Overruns
Update Collector Credentials
Browse Activity Log
Apply Configuration Template
Delete

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSIC
Filter	Filter	Filter	Filter	Filter
[redacted]ulation	● Unknown	Historian	[redacted]	[redacted]
[redacted]	● Running	Historian	[redacted]	[redacted]
[redacted]nCollector	● Running	Historian	[redacted]	[redacted]
[redacted] Calculation	● Running	Historian	[redacted]	[redacted]
[redacted] MQTTSparkplugB	● Running	Historian	[redacted]	[redacted]
[redacted] Python	● Unknown	Historian	[redacted]	[redacted]
[redacted] Python_1	● Unknown	Historian	[redacted]	[redacted]
[redacted] Python_2	● Unknown	Historian	[redacted]	[redacted]
[redacted] Simulation	● Running	Historian	[redacted]	[redacted]

   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.

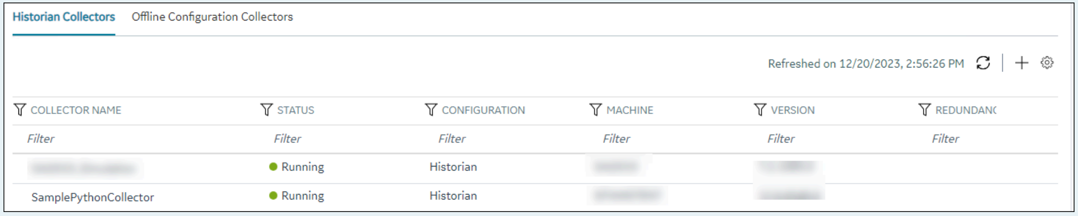
Stop a Collector

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 729)*.

1. Access Configuration Hub *(on page 97)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.

   A list of collectors appears.

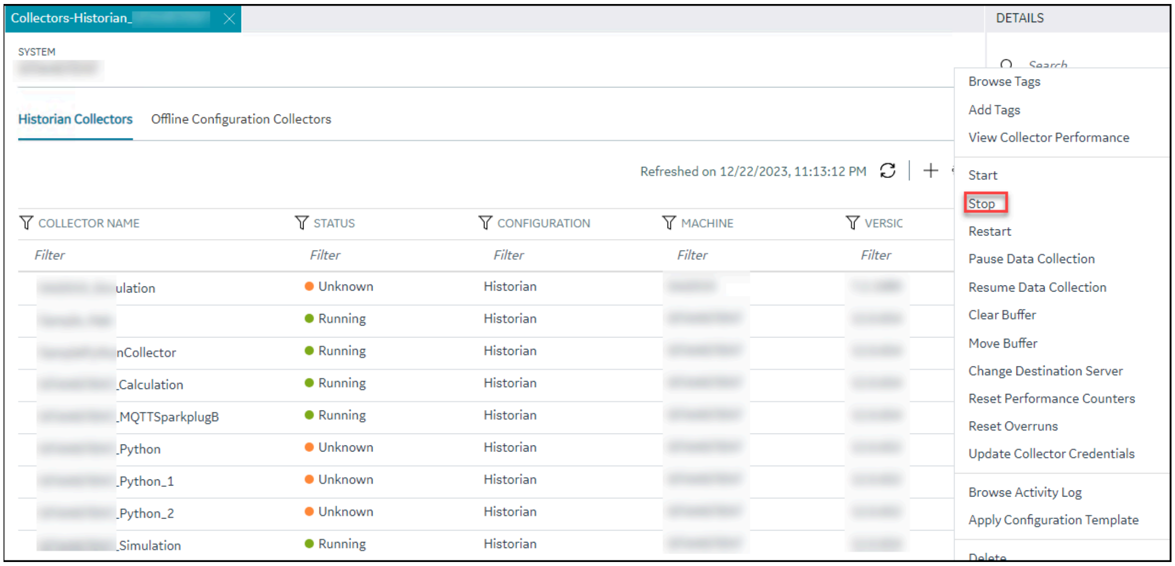
> **Tip:**
> You can filter the collectors by the system name.



Historian Collectors
Offline Configuration Collectors
Refreshed on 12/20/2023, 2:56:26 PM

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSION	REDUNDANC
Filter	Filter	Filter	Filter	Filter	Filter
[redacted]	● Running	Historian	[redacted]	[redacted]	
SamplePythonCollector	● Running	Historian	[redacted]	[redacted]	

3. Right-click the collector instance that you want to stop (or select ⋯), and then select **Stop**.


Collectors-Historian_[redacted]
DETAILS
SYSTEM
[redacted]
Search
Historian Collectors
Offline Configuration Collectors
Refreshed on 12/22/2023, 11:13:12 PM
Browse Tags
Add Tags
View Collector Performance
Start
Stop
Restart
Pause Data Collection
Resume Data Collection
Clear Buffer
Move Buffer
Change Destination Server
Reset Performance Counters
Reset Overruns
Update Collector Credentials
Browse Activity Log
Apply Configuration Template
Delete

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSIC
Filter	Filter	Filter	Filter	Filter
[redacted]ulation	● Unknown	Historian	[redacted]	[redacted]
[redacted]	● Running	Historian	[redacted]	[redacted]
[redacted]nCollector	● Running	Historian	[redacted]	[redacted]
[redacted]_Calculation	● Running	Historian	[redacted]	[redacted]
[redacted]_MQTTSparkplugB	● Running	Historian	[redacted]	[redacted]
[redacted]_Python	● Unknown	Historian	[redacted]	[redacted]
[redacted]_Python_1	● Unknown	Historian	[redacted]	[redacted]
[redacted]_Python_2	● Unknown	Historian	[redacted]	[redacted]
[redacted]_Simulation	● Running	Historian	[redacted]	[redacted]

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.

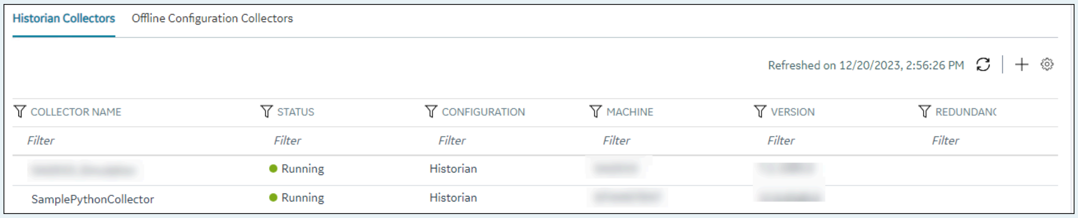
Restart a Collector

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 97)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.

   A list of collectors appears.

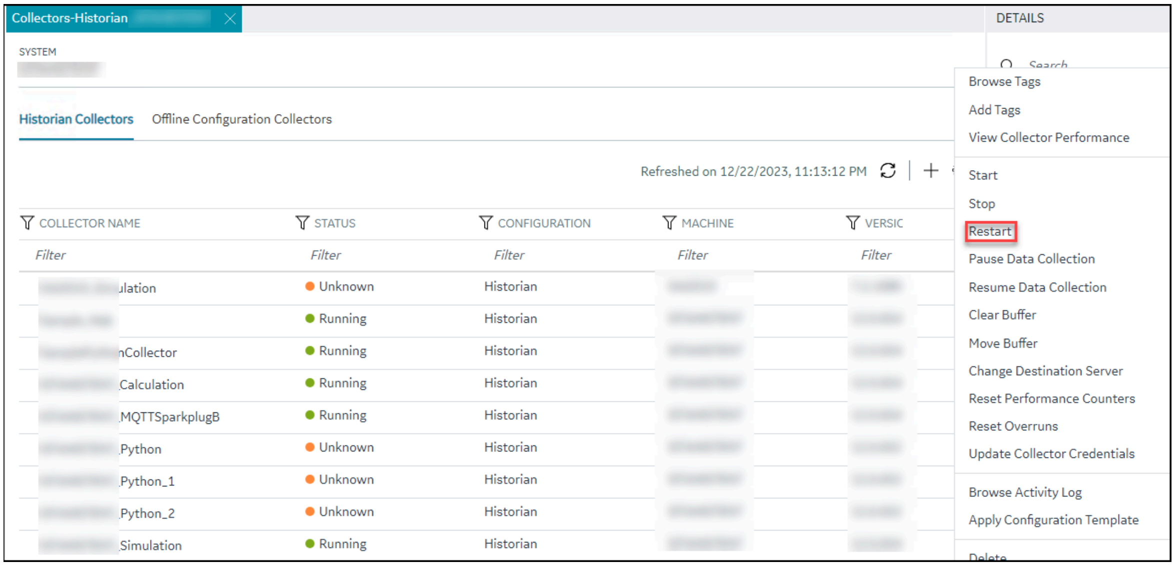
> **Tip:**
>
> You can filter the collectors by the system name.



Refreshed on 12/20/2023, 2:56:26 PM

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSION	REDUNDANC
Filter	Filter	Filter	Filter	Filter	Filter
[redacted]	● Running	Historian	[redacted]	[redacted]	
SamplePythonCollector	● Running	Historian	[redacted]	[redacted]	

3. Right-click the collector instance that you want to restart (or select •••), and then select **Restart**.



Collectors-Historian [redacted]
DETAILS
SYSTEM
Search
Historian Collectors
Offline Configuration Collectors
Browse Tags
Add Tags
View Collector Performance
Start
Stop
Restart
Pause Data Collection
Resume Data Collection
Clear Buffer
Move Buffer
Change Destination Server
Reset Performance Counters
Reset Overruns
Update Collector Credentials
Browse Activity Log
Apply Configuration Template
Delete

Refreshed on 12/22/2023, 11:13:12 PM

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSIC
Filter	Filter	Filter	Filter	Filter
[redacted]ulation	● Unknown	Historian	[redacted]	[redacted]
[redacted]	● Running	Historian	[redacted]	[redacted]
[redacted]nCollector	● Running	Historian	[redacted]	[redacted]
[redacted]Calculation	● Running	Historian	[redacted]	[redacted]
[redacted]MQTTSparkplugB	● Running	Historian	[redacted]	[redacted]
[redacted]Python	● Unknown	Historian	[redacted]	[redacted]
[redacted]Python_1	● Unknown	Historian	[redacted]	[redacted]
[redacted]Python_2	● Unknown	Historian	[redacted]	[redacted]
[redacted]Simulation	● Running	Historian	[redacted]	[redacted]

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

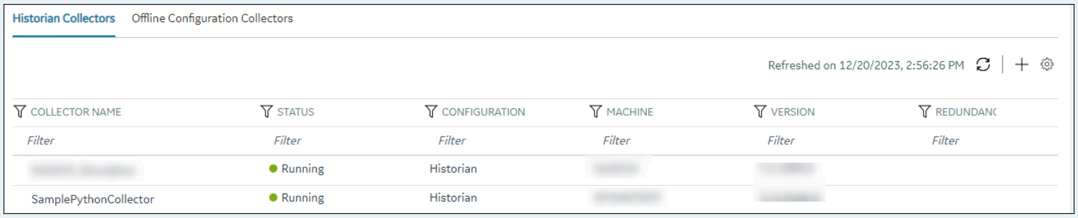
When you pause data collection, the collector stops collecting the data. However, the collector is still connected to the destination. If you want to disconnect the collector from the destination, stop the collector *(on page 727)*.

Note:
You cannot pause the data collection of an offline collector.

1. Access Configuration Hub *(on page 97)*.
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.

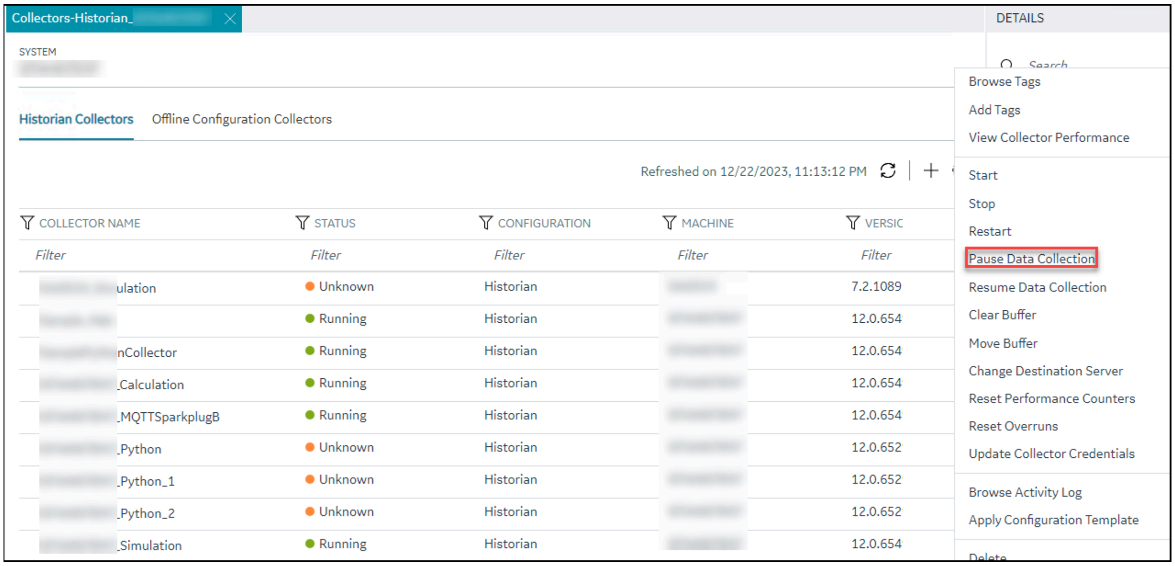


Historian Collectors | Offline Configuration Collectors

Refreshed on 12/20/2023, 2:56:26 PM

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSION	REDUNDANC
Filter	Filter	Filter	Filter	Filter	Filter
[redacted]	● Running	Historian	[redacted]	[redacted]	
SamplePythonCollector	● Running	Historian	[redacted]	[redacted]	

3. Right-click the collector instance for which you want to pause data collection (or select ⋯), and then select **Pause Data Collection**.



Collectors-Historian_[redacted]

SYSTEM [redacted]

Historian Collectors | Offline Configuration Collectors

Refreshed on 12/22/2023, 11:13:12 PM

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSIC
Filter	Filter	Filter	Filter	Filter
[redacted]ulation	● Unknown	Historian	[redacted]	7.2.1089
[redacted]	● Running	Historian	[redacted]	12.0.654
[redacted]nCollector	● Running	Historian	[redacted]	12.0.654
[redacted]_Calculation	● Running	Historian	[redacted]	12.0.654
[redacted]_MQTTSparkplugB	● Running	Historian	[redacted]	12.0.654
[redacted]_Python	● Unknown	Historian	[redacted]	12.0.652
[redacted]_Python_1	● Unknown	Historian	[redacted]	12.0.652
[redacted]_Python_2	● Unknown	Historian	[redacted]	12.0.652
[redacted]_Simulation	● Running	Historian	[redacted]	12.0.654

   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.

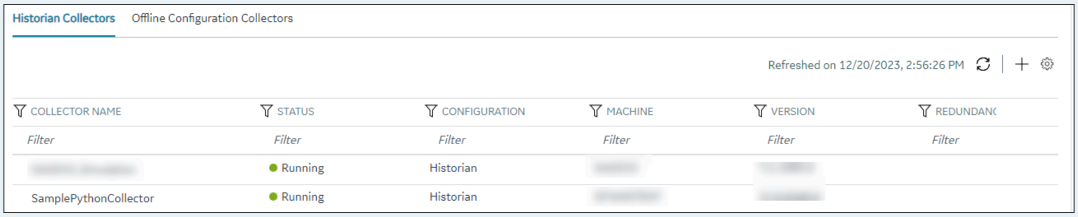
Resume the Data Collection of a Collector

1. Access Configuration Hub *(on page 97)*.
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.

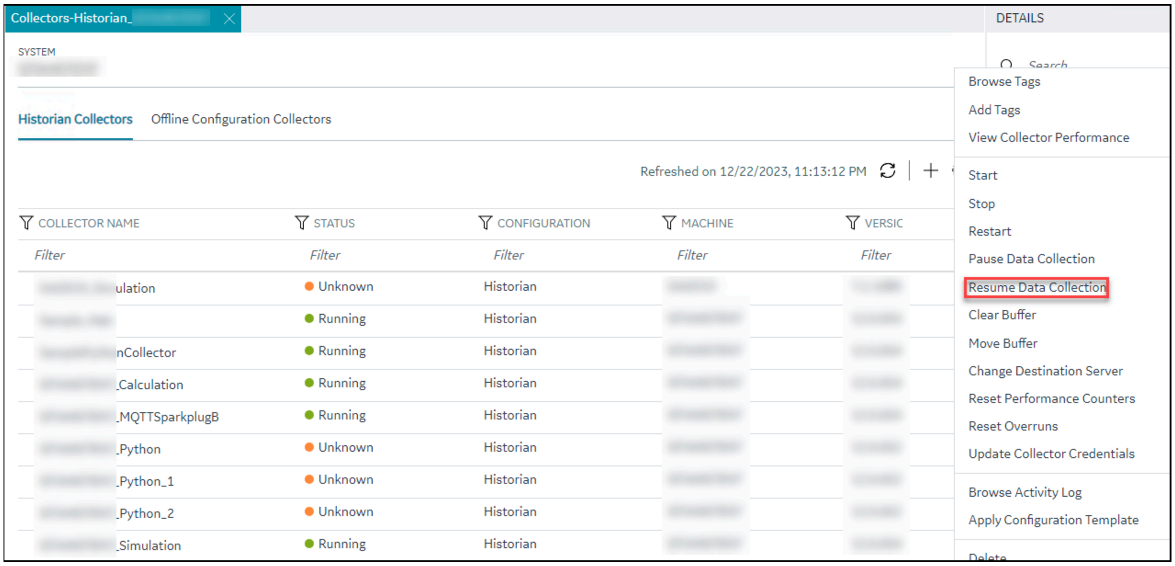


Historian Collectors | Offline Configuration Collectors

Refreshed on 12/20/2023, 2:56:26 PM

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSION	REDUNDANC
Filter	*Filter*	*Filter*	*Filter*	*Filter*	*Filter*
[redacted]	● Running	Historian	[redacted]	[redacted]	
SamplePythonCollector	● Running	Historian	[redacted]	[redacted]	

3. Right-click the collector instance for which you want to resume data collection (or select ⚬⚬⚬), and then select **Resume Data Collection**.



Collectors-Historian_[redacted]

SYSTEM

[redacted]

Historian Collectors | Offline Configuration Collectors

Refreshed on 12/22/2023, 11:13:12 PM

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSIC
Filter	*Filter*	*Filter*	*Filter*	*Filter*
[redacted]ulation	● Unknown	Historian	[redacted]	[redacted]
[redacted]	● Running	Historian	[redacted]	[redacted]
[redacted]nCollector	● Running	Historian	[redacted]	[redacted]
[redacted]_Calculation	● Running	Historian	[redacted]	[redacted]
[redacted]_MQTTSparkplugB	● Running	Historian	[redacted]	[redacted]
[redacted]_Python	● Unknown	Historian	[redacted]	[redacted]
[redacted]_Python_1	● Unknown	Historian	[redacted]	[redacted]
[redacted]_Python_2	● Unknown	Historian	[redacted]	[redacted]
[redacted]_Simulation	● Running	Historian	[redacted]	[redacted]

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

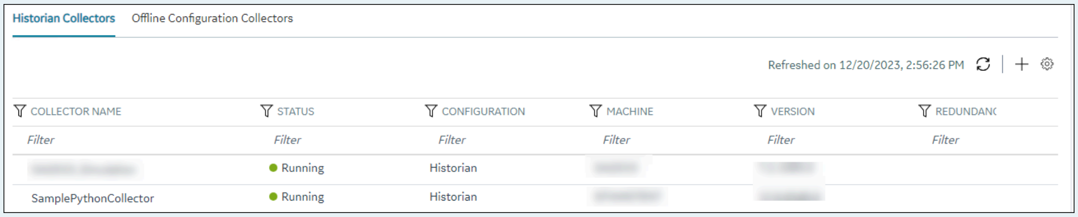
When you delete buffer files, the collector is stopped, and after the buffer files are deleted, it is restarted.

1. Access Configuration Hub *(on page 97)*.
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.

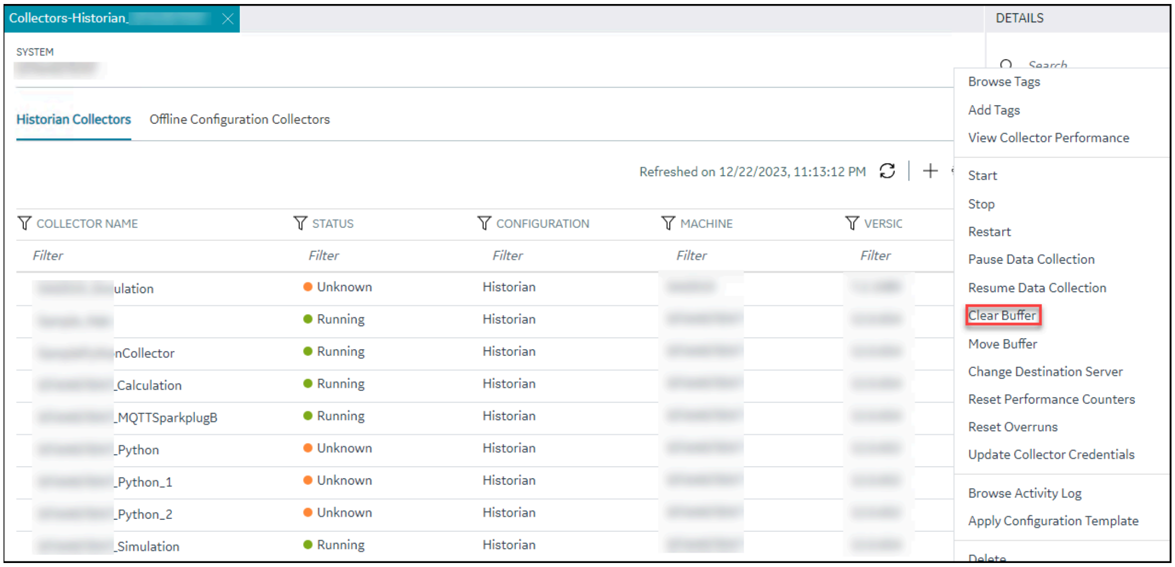


Historian Collectors | Offline Configuration Collectors

Refreshed on 12/20/2023, 2:56:26 PM

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSION	REDUNDANC
Filter	Filter	Filter	Filter	Filter	Filter
[redacted]	● Running	Historian	[redacted]	[redacted]	
SamplePythonCollector	● Running	Historian	[redacted]	[redacted]	

3. Right-click the collector instance whose buffer files you want to delete (or select ⚬⚬⚬), and then select **Clear Buffer**.



Collectors-Historian_[redacted]

SYSTEM

[redacted]

Historian Collectors | Offline Configuration Collectors

Refreshed on 12/22/2023, 11:13:12 PM

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSIC
Filter	Filter	Filter	Filter	Filter
[redacted]ulation	● Unknown	Historian	[redacted]	[redacted]
[redacted]	● Running	Historian	[redacted]	[redacted]
[redacted]nCollector	● Running	Historian	[redacted]	[redacted]
[redacted]_Calculation	● Running	Historian	[redacted]	[redacted]
[redacted]_MQTTSparkplugB	● Running	Historian	[redacted]	[redacted]
[redacted]_Python	● Unknown	Historian	[redacted]	[redacted]
[redacted]_Python_1	● Unknown	Historian	[redacted]	[redacted]
[redacted]_Python_2	● Unknown	Historian	[redacted]	[redacted]
[redacted]_Simulation	● Running	Historian	[redacted]	[redacted]

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.

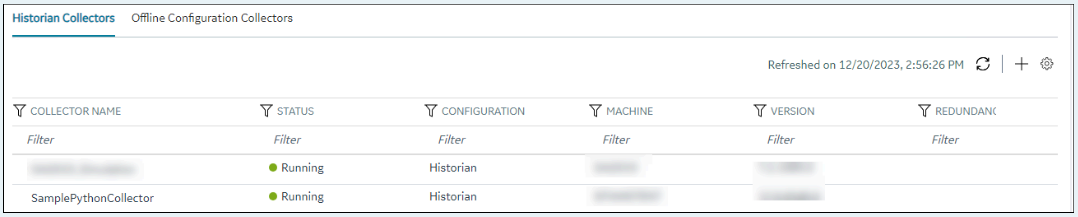
When you move buffer files, the collector is stopped, and after the buffer files are moved, it is restarted.

1. Access Configuration Hub *(on page 97)*.
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.

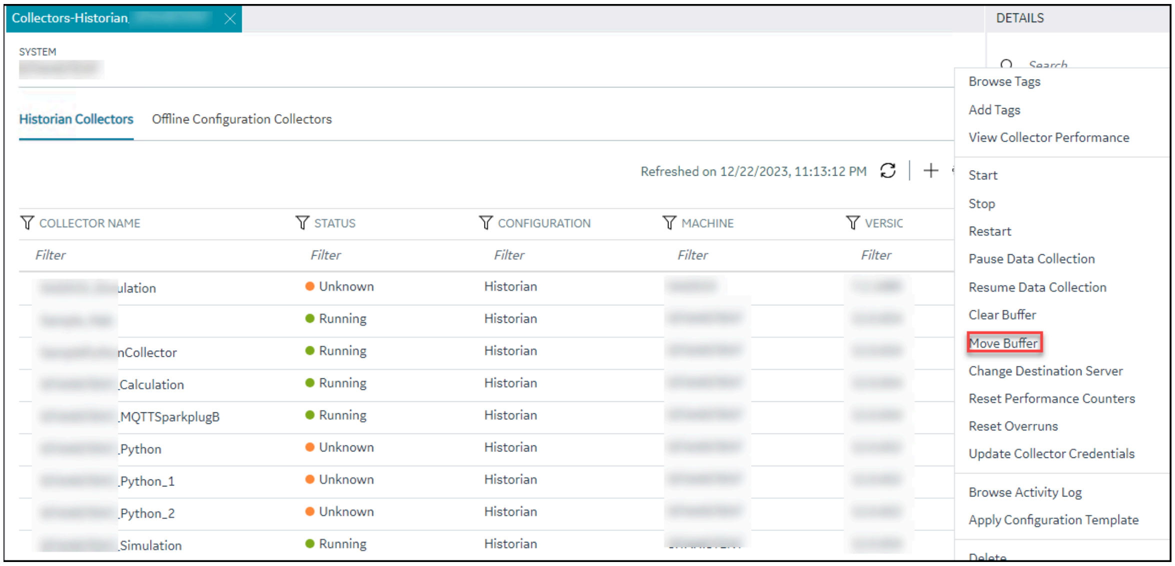
   

   Historian Collectors | Offline Configuration Collectors

   Refreshed on 12/20/2023, 2:56:26 PM

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSION	REDUNDANC
Filter	Filter	Filter	Filter	Filter	Filter
[redacted]	● Running	Historian	[redacted]	[redacted]	
SamplePythonCollector	● Running	Historian	[redacted]	[redacted]	

3. Right-click the collector instance whose buffer files you want to move (or select •••), and then select **Move Buffer**.

   Collectors-Historian [redacted]

   SYSTEM

   [redacted]

   Historian Collectors | Offline Configuration Collectors

   Refreshed on 12/22/2023, 11:13:12 PM

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSIC
Filter	Filter	Filter	Filter	Filter
[redacted]ulation	● Unknown	Historian	[redacted]	[redacted]
[redacted]	● Running	Historian	[redacted]	[redacted]
[redacted]nCollector	● Running	Historian	[redacted]	[redacted]
[redacted]Calculation	● Running	Historian	[redacted]	[redacted]
[redacted]MQTTSparkplugB	● Running	Historian	[redacted]	[redacted]
[redacted]Python	● Unknown	Historian	[redacted]	[redacted]
[redacted]Python_1	● Unknown	Historian	[redacted]	[redacted]
[redacted]Python_2	● Unknown	Historian	[redacted]	[redacted]
[redacted]Simulation	● Running	Historian	[redacted]	[redacted]

   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.

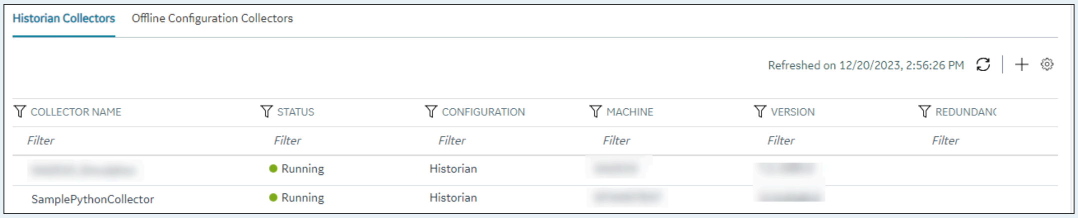
Change the Destination Server of a Collector

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 97)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.
   A list of collectors appears.

Tip:

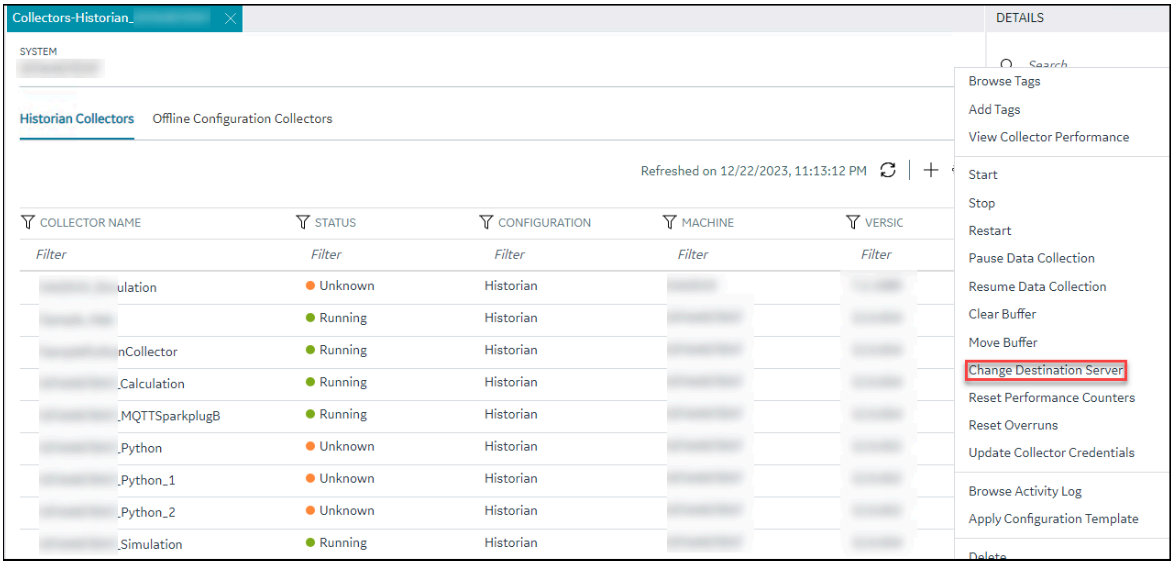
You can filter the collectors by the system name.



Refreshed on 12/20/2023, 2:56:26 PM

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSION	REDUNDANC
Filter	Filter	Filter	Filter	Filter	Filter
[redacted]	● Running	Historian	[redacted]	[redacted]	
SamplePythonCollector	● Running	Historian	[redacted]	[redacted]	

3. Right-click the collector instance whose destination server you want to change (or select ⋯), and then select **Change Destination Server**.


Collectors-Historian_[redacted]
DETAILS
SYSTEM
[redacted]
Search
Historian Collectors
Offline Configuration Collectors
Browse Tags
Add Tags
View Collector Performance
Start
Stop
Restart
Pause Data Collection
Resume Data Collection
Clear Buffer
Move Buffer
Change Destination Server
Reset Performance Counters
Reset Overruns
Update Collector Credentials
Browse Activity Log
Apply Configuration Template
Delete

Refreshed on 12/22/2023, 11:13:12 PM

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSIC
Filter	Filter	Filter	Filter	Filter
[redacted]ulation	● Unknown	Historian	[redacted]	[redacted]
[redacted]	● Running	Historian	[redacted]	[redacted]
[redacted]nCollector	● Running	Historian	[redacted]	[redacted]
[redacted]_Calculation	● Running	Historian	[redacted]	[redacted]
[redacted]_MQTTSparkplugB	● Running	Historian	[redacted]	[redacted]
[redacted]_Python	● Unknown	Historian	[redacted]	[redacted]
[redacted]_Python_1	● Unknown	Historian	[redacted]	[redacted]
[redacted]_Python_2	● Unknown	Historian	[redacted]	[redacted]
[redacted]_Simulation	● Running	Historian	[redacted]	[redacted]

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.

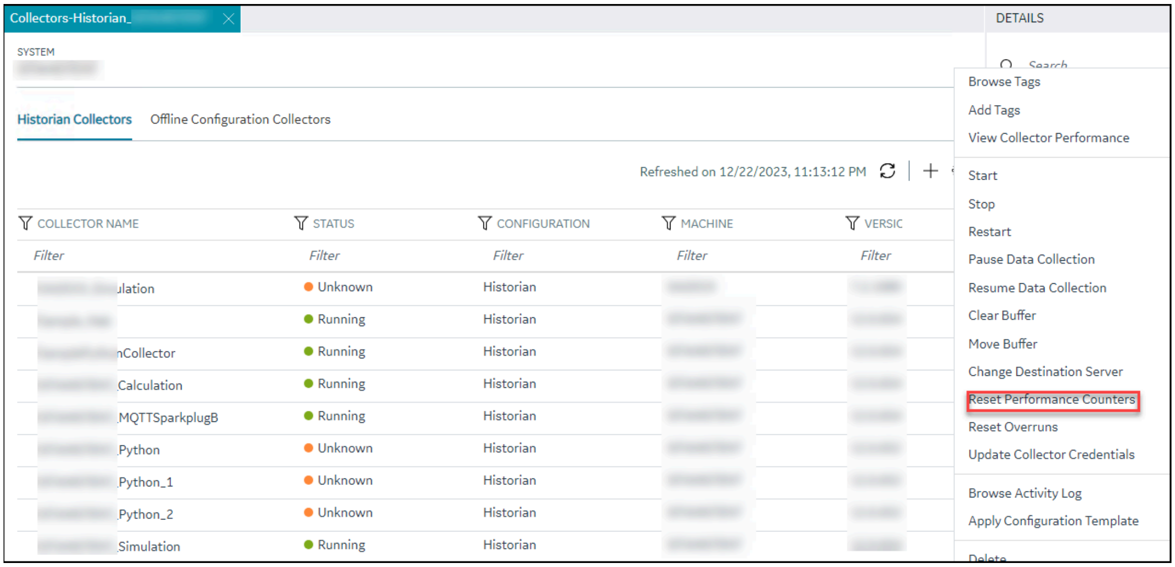
1. Update the network message compression of the collector by modifying the collector instance using Configuration Hub.
2. Reconfigure the collector properties using Historian Administrator.
3. Restart the collector *(on page 728)*.

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 97)*.
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**.

   
   Collectors-Historian_[redacted]
DETAILS
Search
Browse Tags
Add Tags
View Collector Performance
Start
Stop
Restart
Pause Data Collection
Resume Data Collection
Clear Buffer
Move Buffer
Change Destination Server
Reset Performance Counters
Reset Overruns
Update Collector Credentials
Browse Activity Log
Apply Configuration Template
Delete

   SYSTEM
   [redacted]

   Historian Collectors | Offline Configuration Collectors

   Refreshed on 12/22/2023, 11:13:12 PM

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSIC
Filter	Filter	Filter	Filter	Filter
[redacted]ulation	● Unknown	Historian	[redacted]	[redacted]
[redacted]	● Running	Historian	[redacted]	[redacted]
[redacted]nCollector	● Running	Historian	[redacted]	[redacted]
[redacted]Calculation	● Running	Historian	[redacted]	[redacted]
[redacted]MQTTSparkplugB	● Running	Historian	[redacted]	[redacted]
[redacted]Python	● Unknown	Historian	[redacted]	[redacted]
[redacted]Python_1	● Unknown	Historian	[redacted]	[redacted]
[redacted]Python_2	● Unknown	Historian	[redacted]	[redacted]
[redacted]Simulation	● Running	Historian	[redacted]	[redacted]

   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.

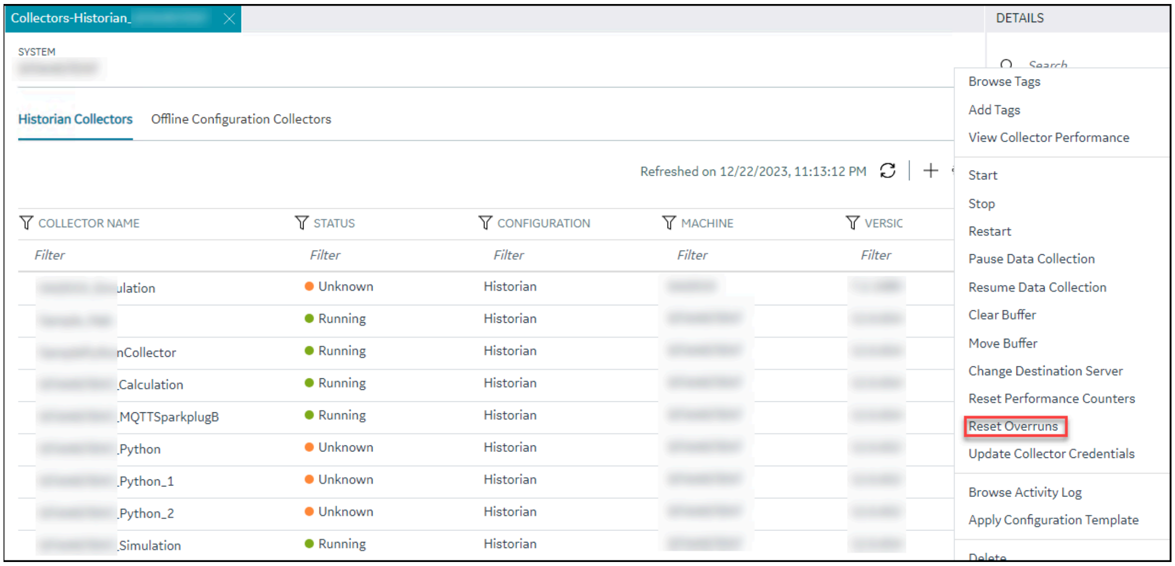
Reset Overruns

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 97)*.
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**.


Collectors-Historian_[redacted]
DETAILS
SYSTEM
[redacted]
Historian Collectors
Offline Configuration Collectors
Refreshed on 12/22/2023, 11:13:12 PM
Search
Browse Tags
Add Tags
View Collector Performance
Start
Stop
Restart
Pause Data Collection
Resume Data Collection
Clear Buffer
Move Buffer
Change Destination Server
Reset Performance Counters
Reset Overruns
Update Collector Credentials
Browse Activity Log
Apply Configuration Template
Delete

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSIC
Filter	*Filter*	*Filter*	*Filter*	*Filter*
[redacted]ulation	● Unknown	Historian	[redacted]	[redacted]
[redacted]	● Running	Historian	[redacted]	[redacted]
[redacted]nCollector	● Running	Historian	[redacted]	[redacted]
[redacted]_Calculation	● Running	Historian	[redacted]	[redacted]
[redacted]_MQTTSparkplugB	● Running	Historian	[redacted]	[redacted]
[redacted]_Python	● Unknown	Historian	[redacted]	[redacted]
[redacted]_Python_1	● Unknown	Historian	[redacted]	[redacted]
[redacted]_Python_2	● Unknown	Historian	[redacted]	[redacted]
[redacted]_Simulation	● Running	Historian	[redacted]	[redacted]

A confirmation window appears, prompting you whether or not to reset the performance counter to zero.

5. Select **Reset**.

   The Overruns are reset.

Apply Configuration Template to a Collector

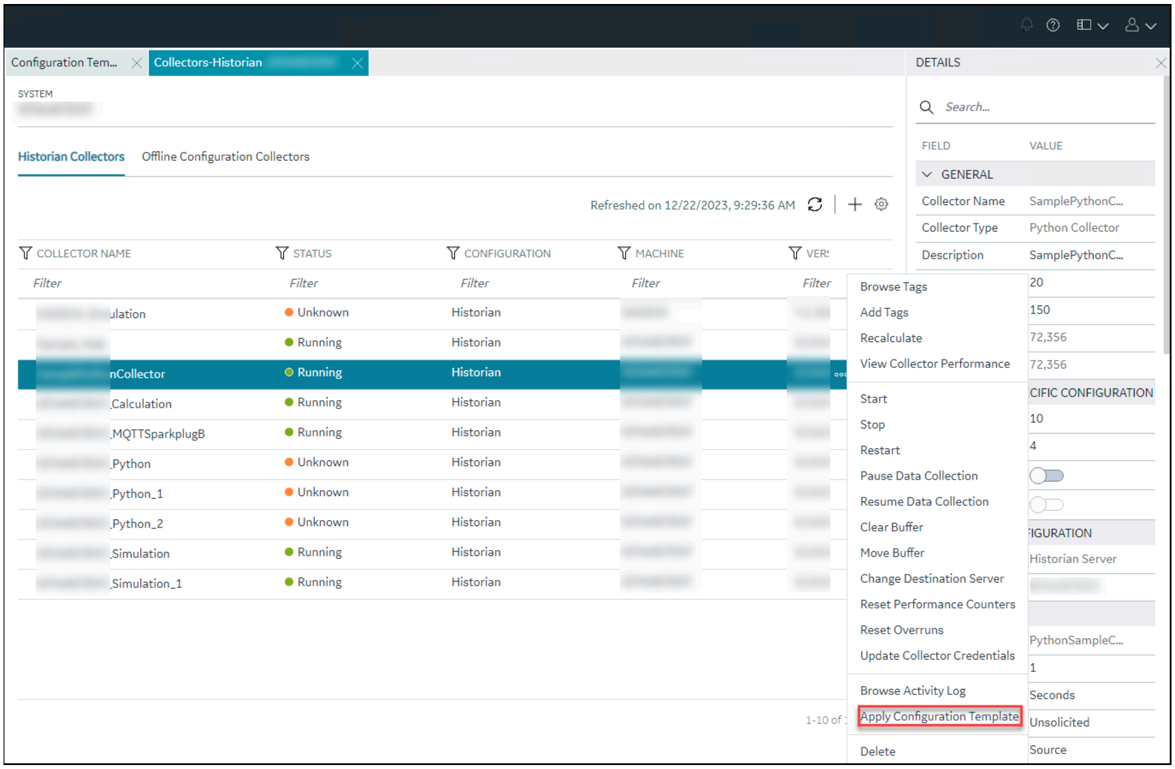
You can apply the created template to collectors as needed. You will be prompted to confirm whether you want to overwrite few of the configuration values with the values in the template.

- Ensure that you have a collector instance added *(on page 556)*.
- Ensure that you created a configuration template for collectors *(on page 813)*.

This topic describes how to apply a configuration template to a collector.

1. Access Configuration Hub *(on page 97)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.

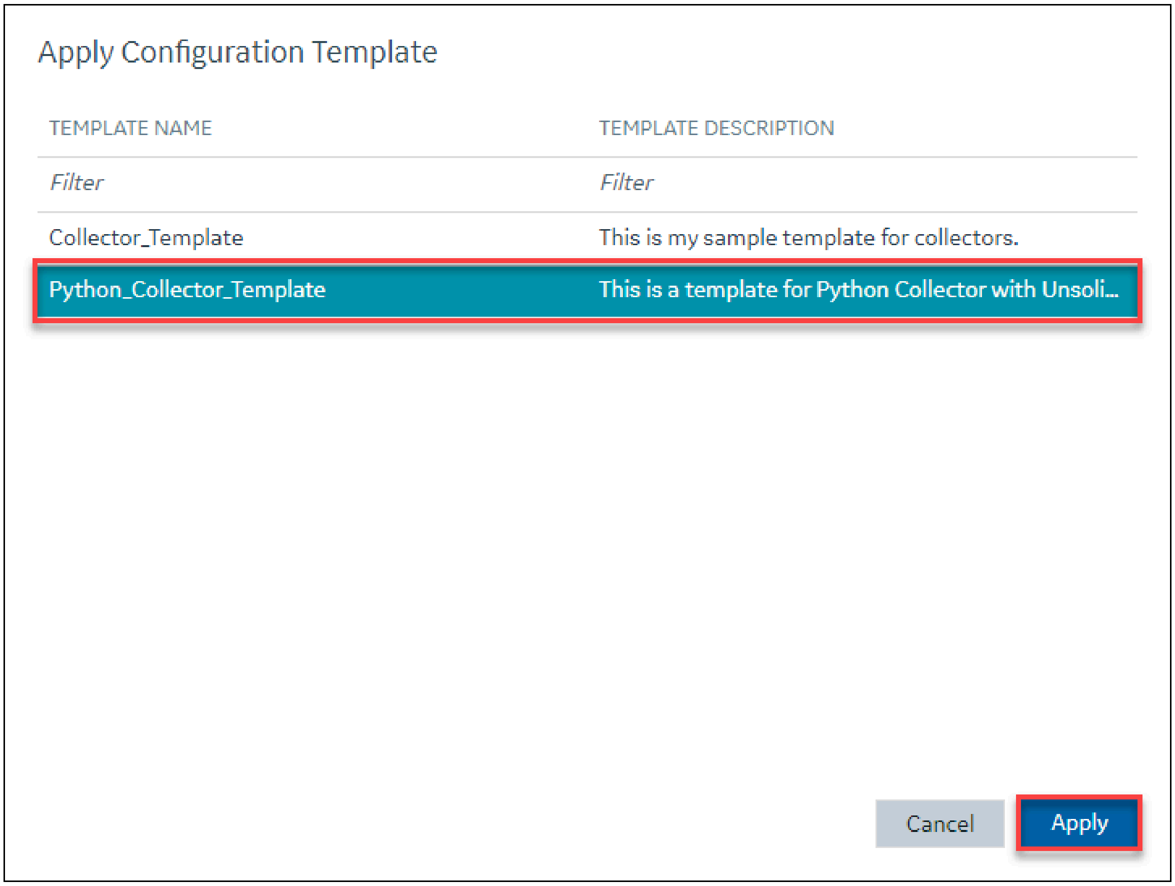
   A list of collectors appears.
3. Right-click the collector (or select ⋯), and then select **Apply Configuration Template**.



COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VER
Filter	Filter	Filter	Filter	Filter
[redacted]ulation	● Unknown	Historian	[redacted]	[redacted]
[redacted]	● Running	Historian	[redacted]	[redacted]
[redacted]nCollector	● Running	Historian	[redacted]	[redacted]
[redacted]_Calculation	● Running	Historian	[redacted]	[redacted]
[redacted]_MQTTSparkplugB	● Running	Historian	[redacted]	[redacted]
[redacted]_Python	● Unknown	Historian	[redacted]	[redacted]
[redacted]_Python_1	● Unknown	Historian	[redacted]	[redacted]
[redacted]_Python_2	● Unknown	Historian	[redacted]	[redacted]
[redacted]_Simulation	● Running	Historian	[redacted]	[redacted]
[redacted]_Simulation_1	● Running	Historian	[redacted]	[redacted]

FIELD	VALUE
GENERAL	
Collector Name	SamplePythonC...
Collector Type	Python Collector
Description	SamplePythonC...
	20
	150
	72,356
	72,356
CIFIC CONFIGURATION	
	10
	4
FIGURATION	
	Historian Server
	[redacted]
	PythonSampleC...
	1
	Seconds
	Unsolicited
	Source

The **Apply Configuration Template** window appears, listing the available templates.



Apply Configuration Template

TEMPLATE NAME	TEMPLATE DESCRIPTION
Filter	*Filter*
Collector_Template	This is my sample template for collectors.
Python_Collector_Template	This is a template for Python Collector with Unsoli...

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.

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.

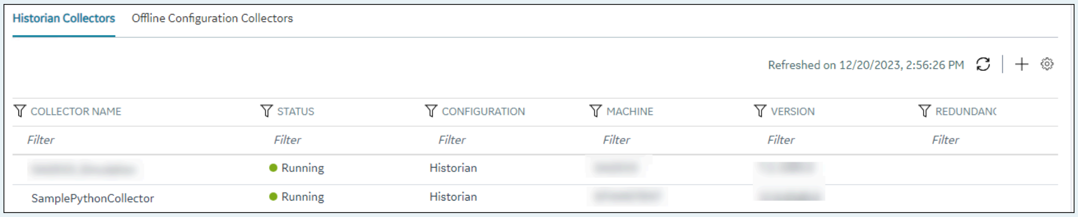
This topic describes how to delete a collector instance using Configuration Hub. You can also delete a collector instance using the RemoteCollectorConfigurator utility *(on page 1355)*, which does not require you to install Web-based Clients.

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 97)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.

   A list of collectors appears.

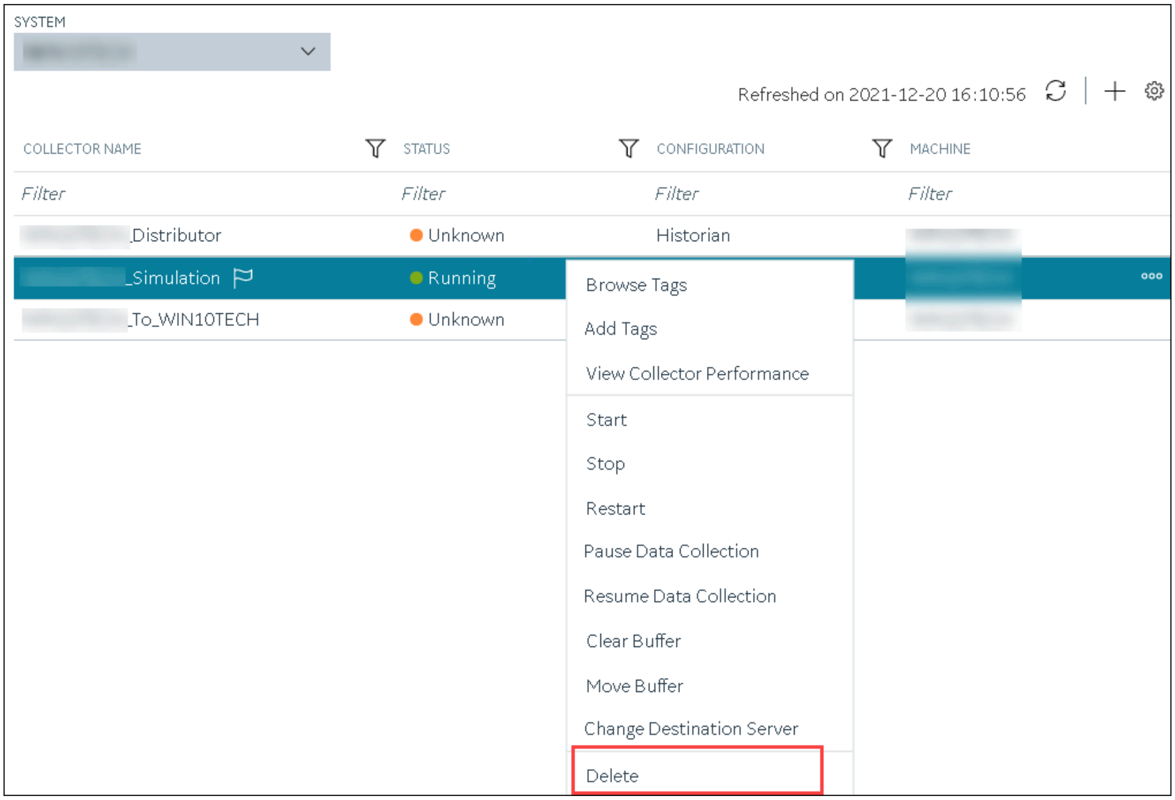
   Tip:
   You can filter the collectors by the system name.

   Refreshed on 12/20/2023, 2:56:26 PM

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSION	REDUNDANC
Filter	Filter	Filter	Filter	Filter	Filter
[redacted]	● Running	Historian	[redacted]	[redacted]	
SamplePythonCollector	● Running	Historian	[redacted]	[redacted]	

3. Right-click the collector instance that you want to delete (or select ⁰⁰⁰), and then select **Delete**.



SYSTEM [redacted]

Refreshed on 2021-12-20 16:10:56

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE
Filter	Filter	Filter	Filter
[redacted]_Distributor	● Unknown	Historian	[redacted]
[redacted]_Simulation	● Running		[redacted]
[redacted]_To_WIN10TECH	● Unknown		[redacted]

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 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)*.

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

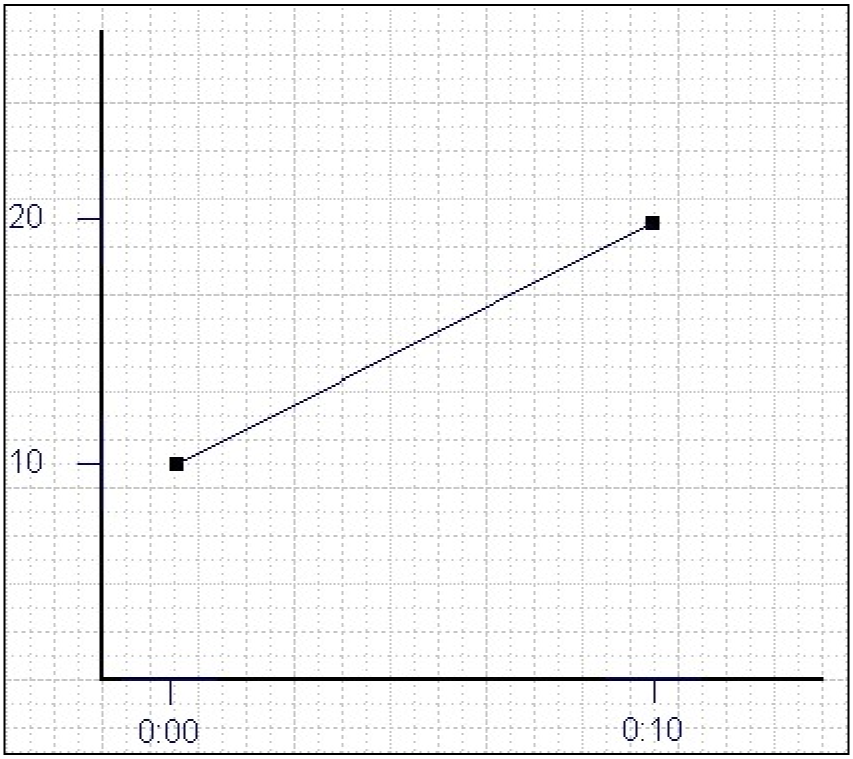
For example, a collector reads a value X once per second, with a compression deadband of 1.0. If the value of X is 10.0 for a number of seconds starting at 0:00:00 and jumps to 20.0 at 0:00:10, the data samples read would be:

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
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)

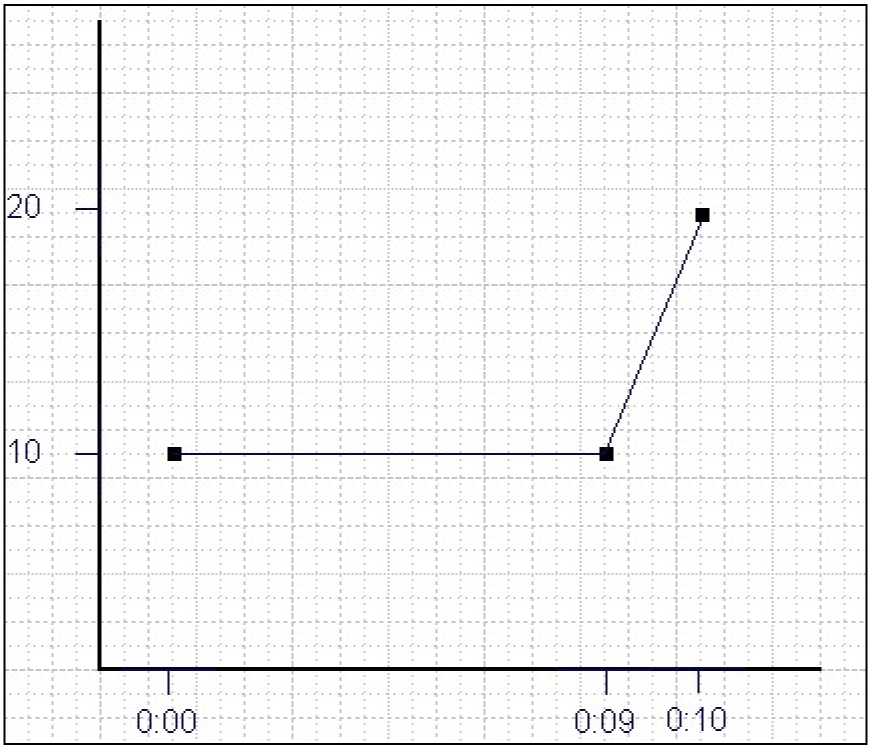
However, without the marker value, if this data were to be put into a chart, it would look like the data value **ramped** over 10 seconds from a value of 10.0 to 20.0, as shown in the following chart.



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)*.

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)*.

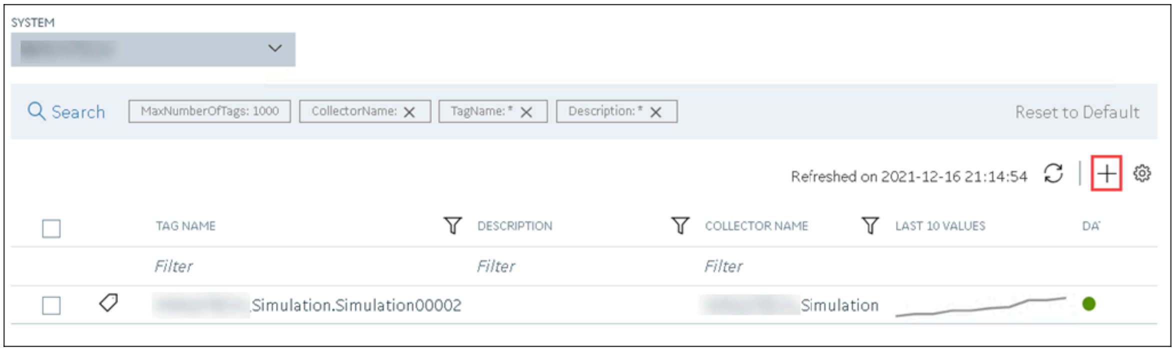
Add Tags for the Data Store Using Configuration Hub

- Add the collector instance *(on page 556)* 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 575)*.

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 589)*.

1. Access Configuration Hub *(on page 97)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
3. Select +.



Refreshed on 2021-12-16 21:14:54

□		TAG NAME	DESCRIPTION	COLLECTOR NAME	LAST 10 VALUES	DA
		Filter	Filter	Filter		
□		Simulation.Simulation00002		Simulation		●

The **Add Tag-<system name>** page appears. The **Add Tags from Collector** option is selected by default.

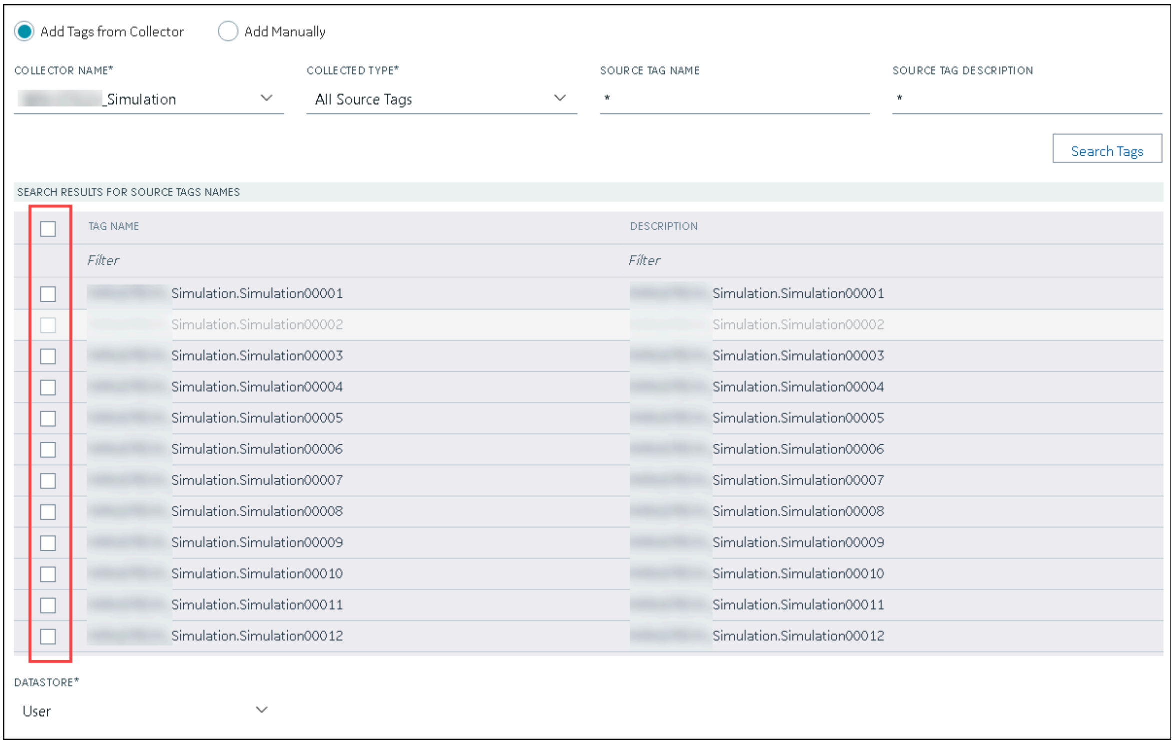
4. Enter values as described in the following table.

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.



● Add Tags from Collector ○ Add Manually

COLLECTOR NAME*: [redacted]_Simulation

COLLECTED TYPE*: All Source Tags

SOURCE TAG NAME: *

SOURCE TAG DESCRIPTION: *

SEARCH RESULTS FOR SOURCE TAGS NAMES

□	TAG NAME	DESCRIPTION
	Filter	Filter
□	[redacted]Simulation.Simulation00001	[redacted]Simulation.Simulation00001
□	[redacted]Simulation.Simulation00002	[redacted]Simulation.Simulation00002
□	[redacted]Simulation.Simulation00003	[redacted]Simulation.Simulation00003
□	[redacted]Simulation.Simulation00004	[redacted]Simulation.Simulation00004
□	[redacted]Simulation.Simulation00005	[redacted]Simulation.Simulation00005
□	[redacted]Simulation.Simulation00006	[redacted]Simulation.Simulation00006
□	[redacted]Simulation.Simulation00007	[redacted]Simulation.Simulation00007
□	[redacted]Simulation.Simulation00008	[redacted]Simulation.Simulation00008
□	[redacted]Simulation.Simulation00009	[redacted]Simulation.Simulation00009
□	[redacted]Simulation.Simulation00010	[redacted]Simulation.Simulation00010
□	[redacted]Simulation.Simulation00011	[redacted]Simulation.Simulation00011
□	[redacted]Simulation.Simulation00012	[redacted]Simulation.Simulation00012

DATASTORE*: User

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.

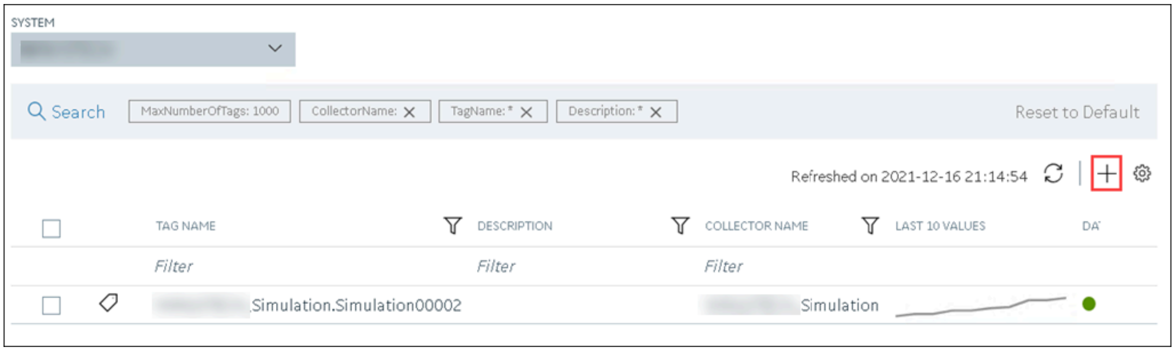
As needed, configure each tag by providing values for the tag properties. For information on the delta query modes, refer to Counter Delta Queries.

Add a Tag Manually

- Add the collector instance *(on page 556)* 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 575)*.

After you create a collector instance, you specify which tags from the source must be used for data collection *(on page 384)*. 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.

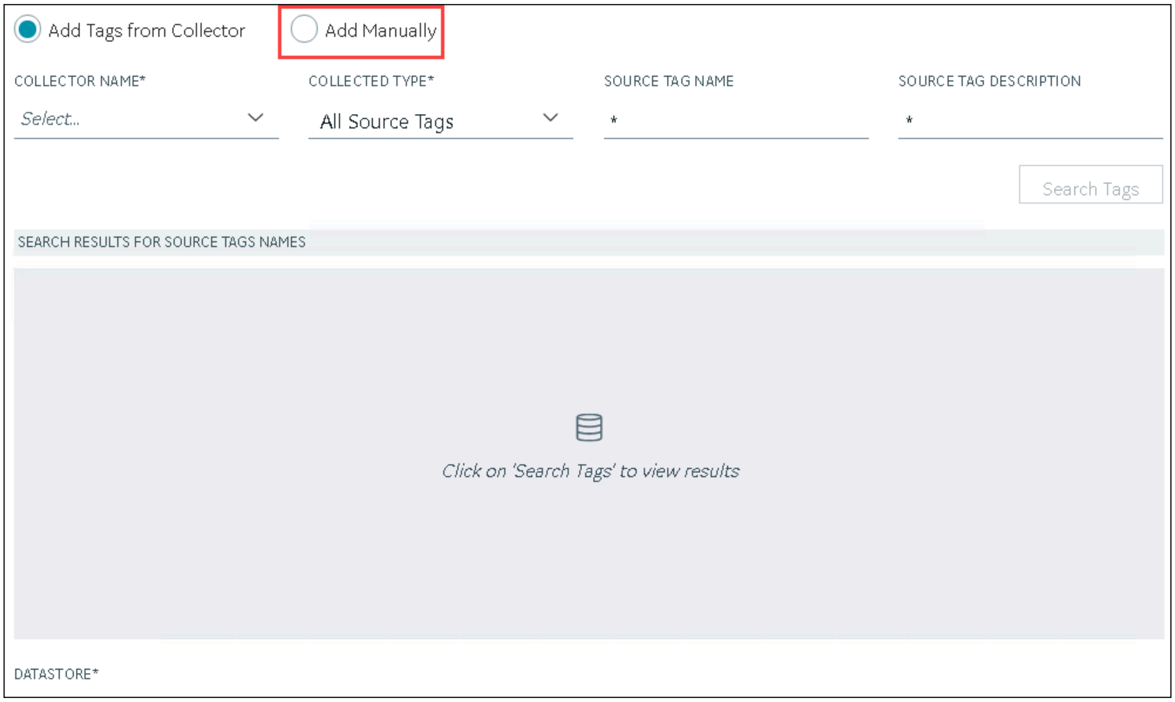
1. Access Configuration Hub *(on page 97)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
3. Select +.



Refreshed on 2021-12-16 21:14:54

□		TAG NAME	DESCRIPTION	COLLECTOR NAME	LAST 10 VALUES	DA
		Filter	Filter	Filter		
□		[redacted]Simulation.Simulation00002		[redacted]Simulation		●

   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. The value that you enter: ◦ Must begin with a letter or a number. ◦ Can contain up to 256 characters.

Field	Description
	◦ Can include any of the following special characters: /!|#{}%$-_ ◦ 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. **Important:** If you select an unsupported data type, you may receive incorrect data or even lose data. If you select **Multi-Field**, the **USER-DEFINED TYPE NAME** field appears, and the **ENUMERATED SET** and **ARRAY TAG** fields are disabled. 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. 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)* 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)* 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**. For information on array tags, refer to About Array Tags *(on page 741)*.

Field	Description
TIME RESOLUTION	Select the time resolution for the tag. A value is required. 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.

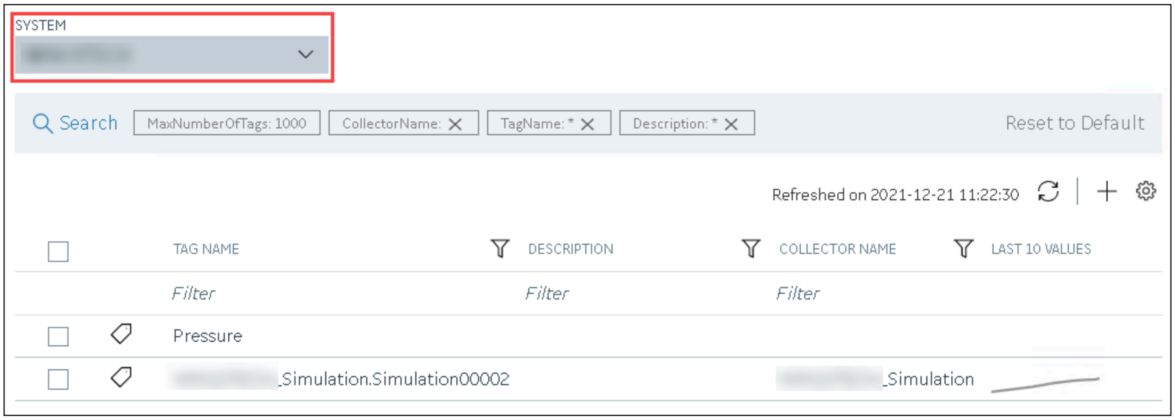
Note:
By default, maximum one million tags are retrieved. If the Historian clients are configured to retrieve more than a million tags, to retrieve all of them, add the `MaxTagsToRetrieve` registry key under `HKEY_LOCAL_MACHINE\SOFTWARE\Intellution, Inc.\iHistorian\Services\DataArchiver\`, and then set 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 97)*.
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 779)*.

> **Tip:**
> You can show/hide/reorder columns in the table. For instructions, refer to Common Tasks in Configuration Hub *(on page 114)*.

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.



SYSTEM

[redacted]

Search | MaxNumberOfTags: 1000 | CollectorName: | TagName: * | Description: * | Reset to Default

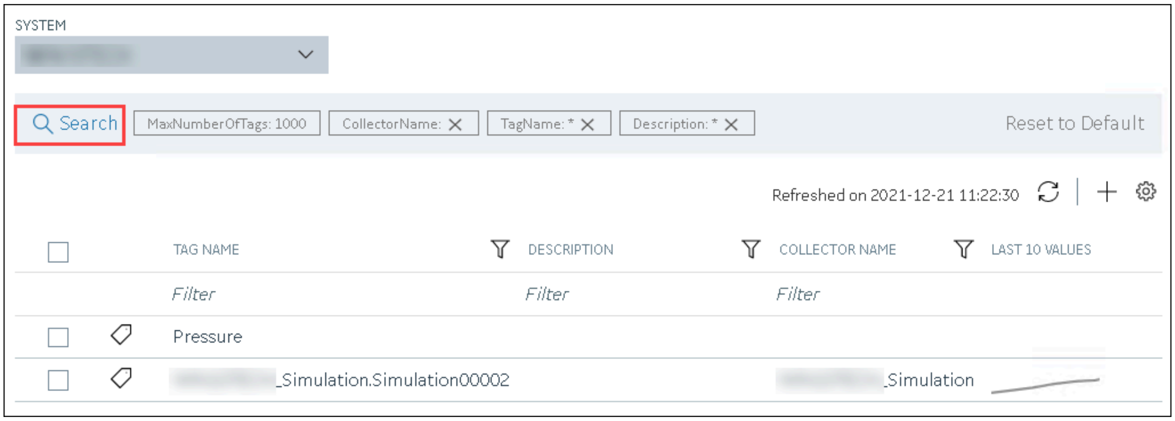
Refreshed on 2021-12-21 11:22:30

□		TAG NAME	DESCRIPTION	COLLECTOR NAME	LAST 10 VALUES
		Filter	Filter	Filter	
□		Pressure			
□		[redacted]_Simulation.Simulation00002		[redacted]_Simulation	

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**.



☐		TAG NAME	DESCRIPTION	COLLECTOR NAME	LAST 10 VALUES
		Filter	Filter	Filter	
☐		Pressure			
☐		_Simulation.Simulation00002		_Simulation	

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 22. 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.
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 23. The Collection Section

Field	Description
Collector	The name of the collector that collects data for the selected tag.

Field	Description
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. For Python Expression tags, this field contains the full applicable JSON configuration, which includes an indication of the source address. **Note:** 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. 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. **Note:** 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)* that you want to assign to the tag.
Array Tag	Indicates that the tag is an array tag *(on page 741)*.
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.

Field	Description
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. **Note:** 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 748)* 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:

Field	Description
	◦ **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. ◦ **< =**: Setting condition as trigger tag value less than or equal to the compare value. ◦ **> =** Setting condition as trigger tag value greater than or equal to the compare value. ◦ **<**: Setting condition as trigger tag value less than the compare value. ◦ **>**: Setting condition as trigger tag value greater than the compare value. ◦ **=**: Setting condition as trigger tag value equals compare value. ◦ **!=**: 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 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 24. 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.

Field	Description
Collection Type	The type of data collection used for this tag: ◦ **Polled**: The data collector requests data from the data source at the collection interval specified in the polling schedule. ◦ **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. **Note:** 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 collection	Indicates whether condition-based data collection *(on page 748)* 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: ◦ **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. ◦ **< =**: Setting condition as trigger tag value less than or equal to the compare value.

Field	Description
	◦ **> =** Setting condition as trigger tag value greater than or equal to the compare value. ◦ **<**: Setting condition as trigger tag value less than the compare value. ◦ **>**: Setting condition as trigger tag value greater than the compare value. ◦ **=**: Setting condition as trigger tag value equals compare value. ◦ **!=**: 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 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 25. 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. For example, you can enter "Temperature" or "degree Celsius" in **EGU Description** for a single tag or multiple tags associated with temperature values.

Field	Description
	When you view the last 10 values of the tag *(on page 772)*, or when you generate a query *(on page 797)*/write *(on page 801)* 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. 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.
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. 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, 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. **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:

Field	Description
	◦ **Hi Engineering Units**: Enter `0`. ◦ **Lo Engineering Units**: Enter `1`. ◦ **Hi Scaling Value**: Enter `0`. ◦ **Lo Scaling Value**: Enter `-1`. ◦ **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 26. The Collector Compression Section

Field	Description
Collector Compression	Indicates whether collector compression *(on page 742)* 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. **Note:** 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. **Example:** 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.) 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.

Field	Description
	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)*.
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. 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. 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. 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.

Field	Description
	Compression timeout is supported in all collectors except the PI collector.

Table 27. The Archive Compression Section

Field	Description
Archive Compression	Indicates whether archive compression *(on page 742)* 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.
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. The data archiver treats the incoming sample after the timeout occurs as if it exceeded compression. It then stores the pending sample.

Table 28. 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. 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. 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

Field	Description
	device does not change for 10 minutes, no new data will be added to the archive for 10 minutes. **Note:** 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). 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. **Note:** This field only applies to tags associated with the Server-to-Server collector that use a polled collection type.

Table 29. The Security Section

Field	Description
Read Group	The Windows security group that can retrieve the tag data and plot it in a trend chart. 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.
For more information, refer to implementing tag-level security *(on page)*. **Note:** 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 30. 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.

Field	Description
	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 1446)*.

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 779)*.

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.

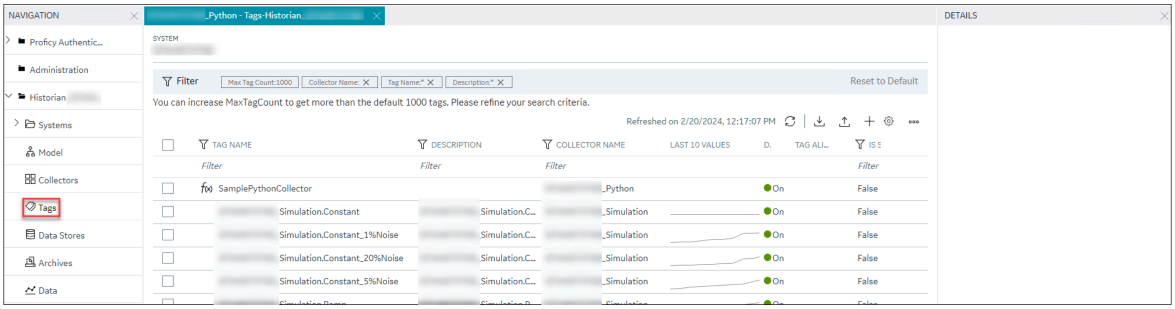
Before you begin the configuration of multiple tags, know the following:

- 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.
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. 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 97)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.



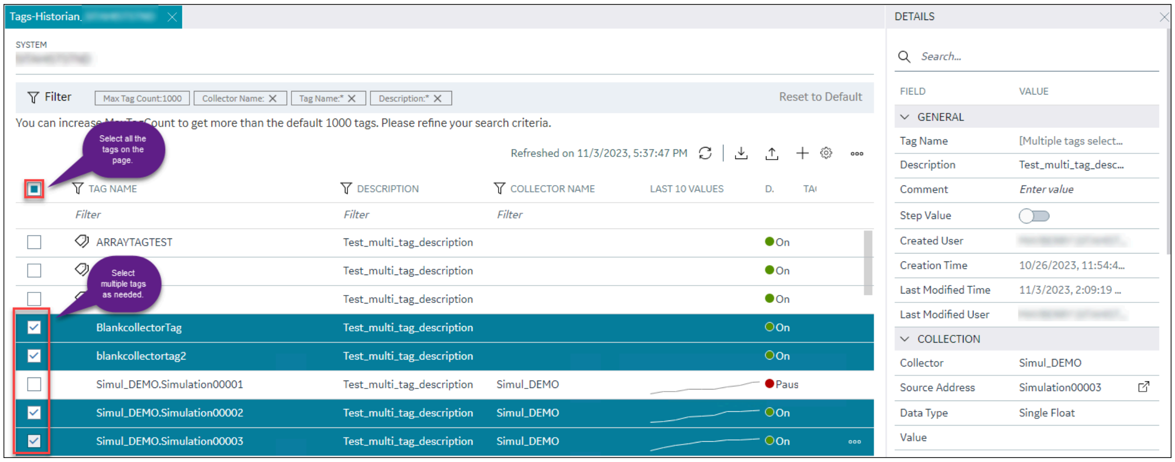
You can increase MaxTagCount to get more than the default 1000 tags. Please refine your search criteria.

Refreshed on 2/20/2024, 12:17:07 PM

	TAG NAME	DESCRIPTION	COLLECTOR NAME	LAST 10 VALUES	D.	TAG ALI...	IS S
	Filter	Filter	Filter				Filter
□	SamplePythonCollector		[redacted]_Python		● On		False
□	[redacted]Simulation.Constant	[redacted]Simulation.C...	[redacted]_Simulation		● On		False
□	[redacted]Simulation.Constant_1%Noise	[redacted]Simulation.C...	[redacted]_Simulation		● On		False
□	[redacted]Simulation.Constant_20%Noise	[redacted]Simulation.C...	[redacted]_Simulation		● On		False
□	[redacted]Simulation.Constant_5%Noise	[redacted]Simulation.C...	[redacted]_Simulation		● On		False

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.



You can increase MaxTagCount to get more than the default 1000 tags. Please refine your search criteria.

Refreshed on 11/3/2023, 5:37:47 PM

	TAG NAME	DESCRIPTION	COLLECTOR NAME	LAST 10 VALUES	D.	TA
	Filter	Filter	Filter			
□	ARRAYTAGTEST	Test_multi_tag_description			● On	
□		Test_multi_tag_description			● On	
□		Test_multi_tag_description			● On	
☑	BlankcollectorTag	Test_multi_tag_description			○ On	
☑	blankcollectortag2	Test_multi_tag_description			○ On	
□	Simul_DEMO.Simulation00001	Test_multi_tag_description	Simul_DEMO		● Paus	
☑	Simul_DEMO.Simulation00002	Test_multi_tag_description	Simul_DEMO		○ On	
☑	Simul_DEMO.Simulation00003	Test_multi_tag_description	Simul_DEMO		○ On	

FIELD	VALUE
GENERAL	
Tag Name	[Multiple tags select...
Description	Test_multi_tag_desc...
Comment	*Enter value*
Step Value	
Created User	[redacted]
Creation Time	10/26/2023, 11:54:4...
Last Modified Time	11/3/2023, 2:09:19 ...
Last Modified User	[redacted]
COLLECTION	
Collector	Simul_DEMO
Source Address	Simulation00003
Data Type	Single Float
Value	

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**.

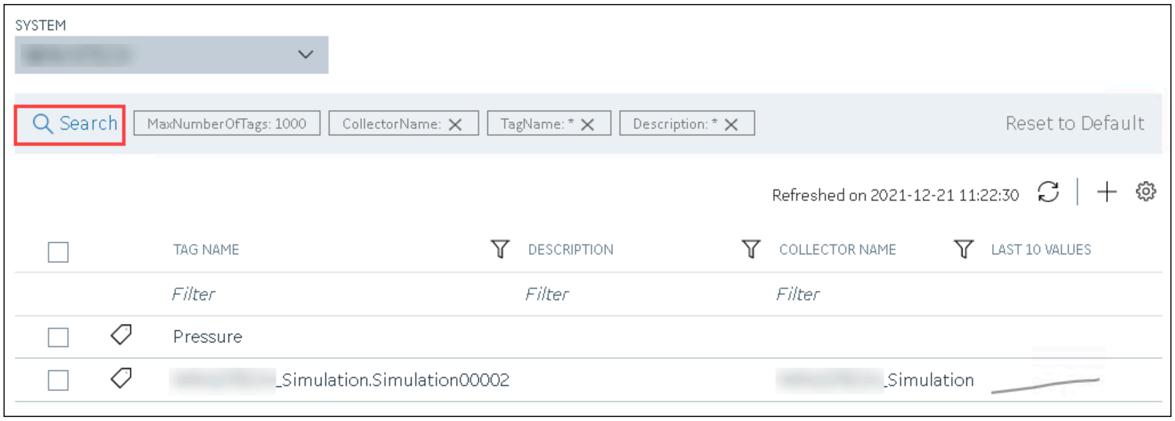
Access the Trend Chart of Tag Values

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.

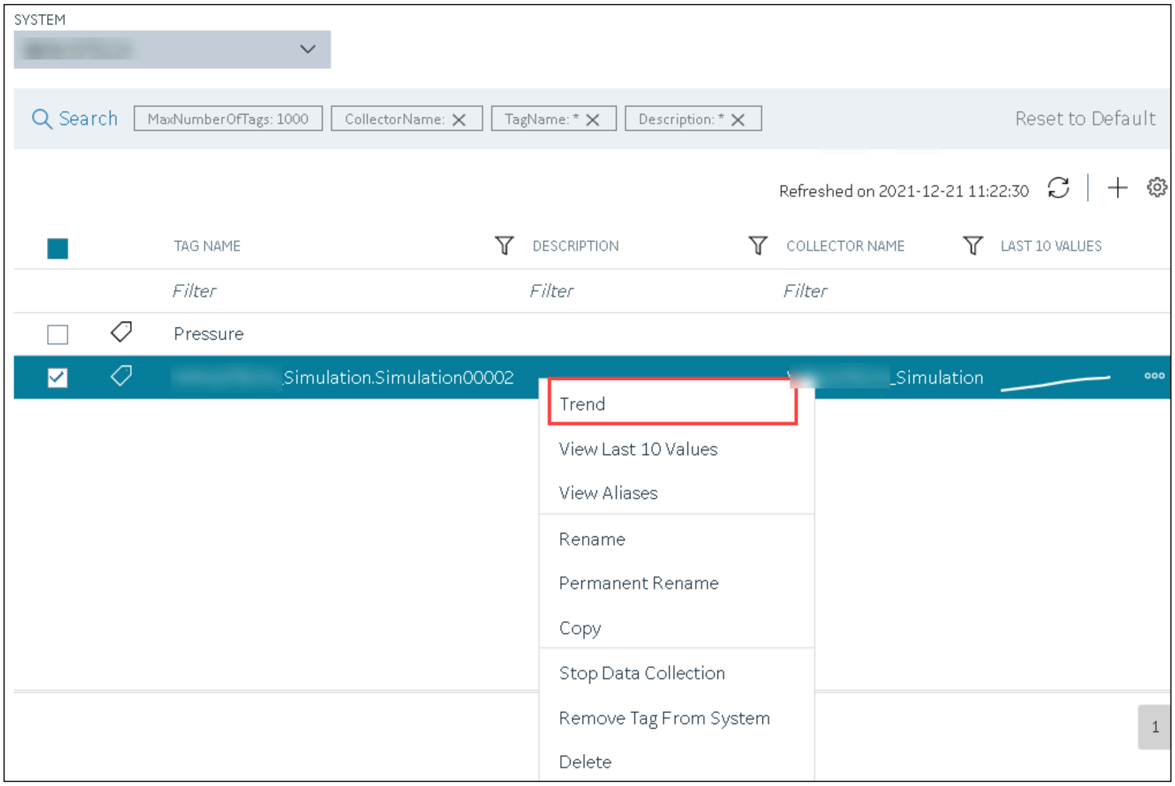
1. Access Configuration Hub *(on page 97)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.

   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.

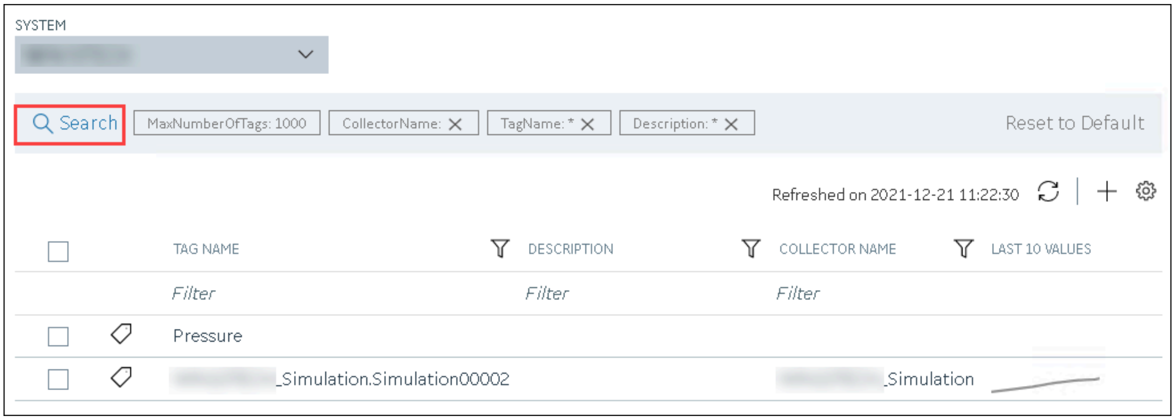
> **Tip:**
> You can also filter the chart by changing the duration, sampling type, time interval, and so on by selecting ▽.

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 97)*.
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**.

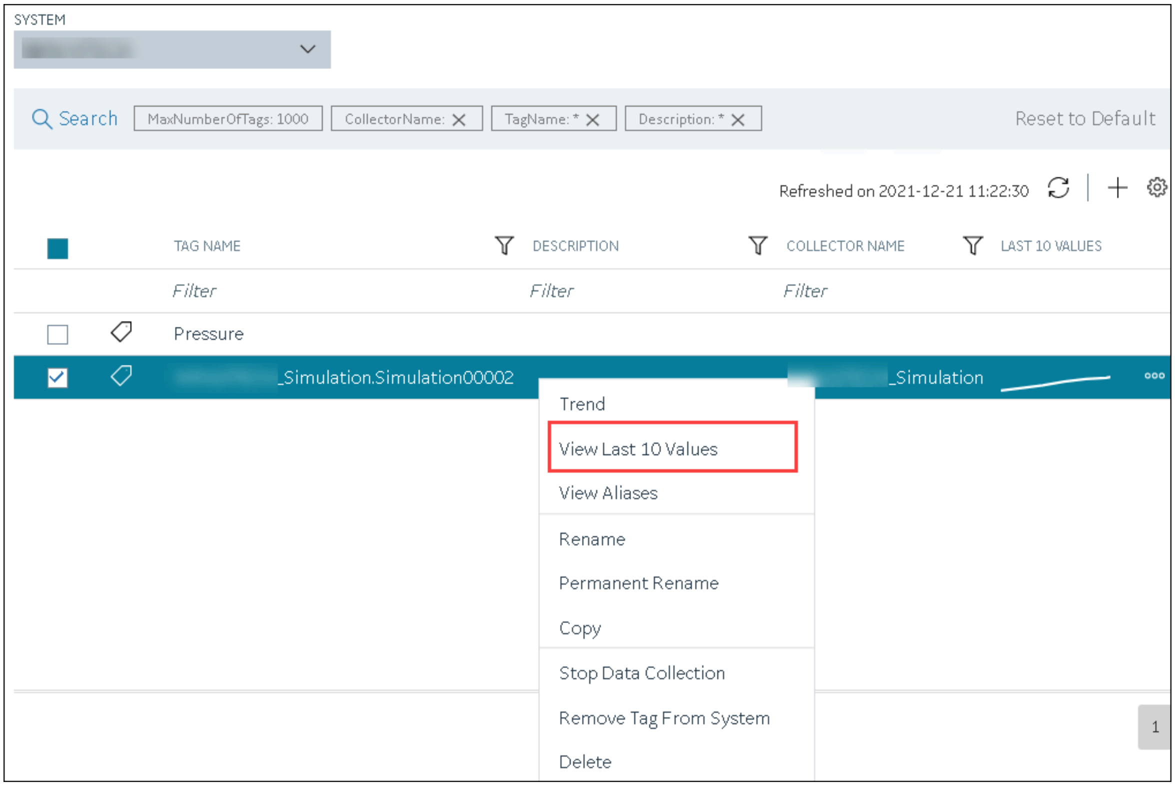
   SYSTEM
   [redacted]

   Search | MaxNumberOfTags: 1000 | CollectorName: ✕ | TagName: * ✕ | Description: * ✕ | Reset to Default

   Refreshed on 2021-12-21 11:22:30

□		TAG NAME	DESCRIPTION	COLLECTOR NAME	LAST 10 VALUES
		Filter	Filter	Filter	
□		Pressure			
□		[redacted]_Simulation.Simulation00002		[redacted]_Simulation	

   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.

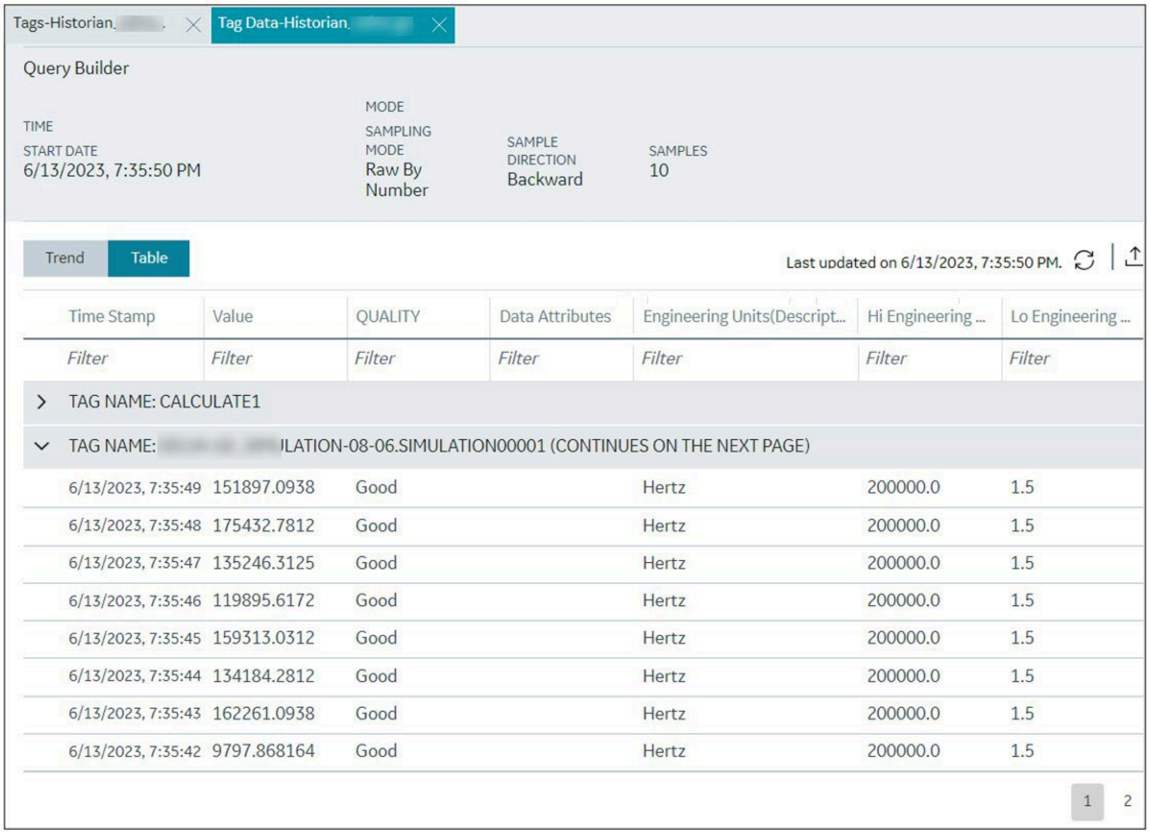
In table view you can view data attributes which are introduced to store the 128-bit subquality for every sample.

> **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**.



TIME
START DATE
6/13/2023, 7:35:50 PM

MODE
SAMPLING MODE
Raw By Number

SAMPLE DIRECTION
Backward

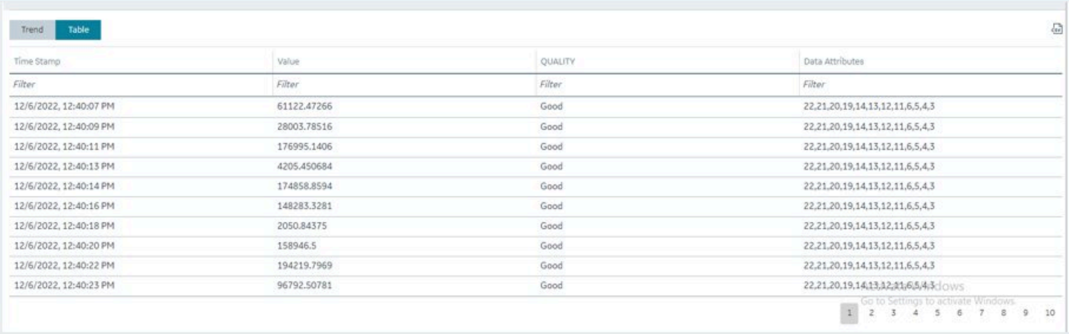
SAMPLES
10

Last updated on 6/13/2023, 7:35:50 PM.

Time Stamp	Value	QUALITY	Data Attributes	Engineering Units(Descript...	Hi Engineering ...	Lo Engineering ...
Filter	Filter	Filter	Filter	Filter	Filter	Filter
TAG NAME: CALCULATE1						
TAG NAME: [redacted]LATION-08-06.SIMULATION00001 (CONTINUES ON THE NEXT PAGE)						
6/13/2023, 7:35:49	151897.0938	Good		Hertz	200000.0	1.5
6/13/2023, 7:35:48	175432.7812	Good		Hertz	200000.0	1.5
6/13/2023, 7:35:47	135246.3125	Good		Hertz	200000.0	1.5
6/13/2023, 7:35:46	119895.6172	Good		Hertz	200000.0	1.5
6/13/2023, 7:35:45	159313.0312	Good		Hertz	200000.0	1.5
6/13/2023, 7:35:44	134184.2812	Good		Hertz	200000.0	1.5
6/13/2023, 7:35:43	162261.0938	Good		Hertz	200000.0	1.5
6/13/2023, 7:35:42	9797.868164	Good		Hertz	200000.0	1.5

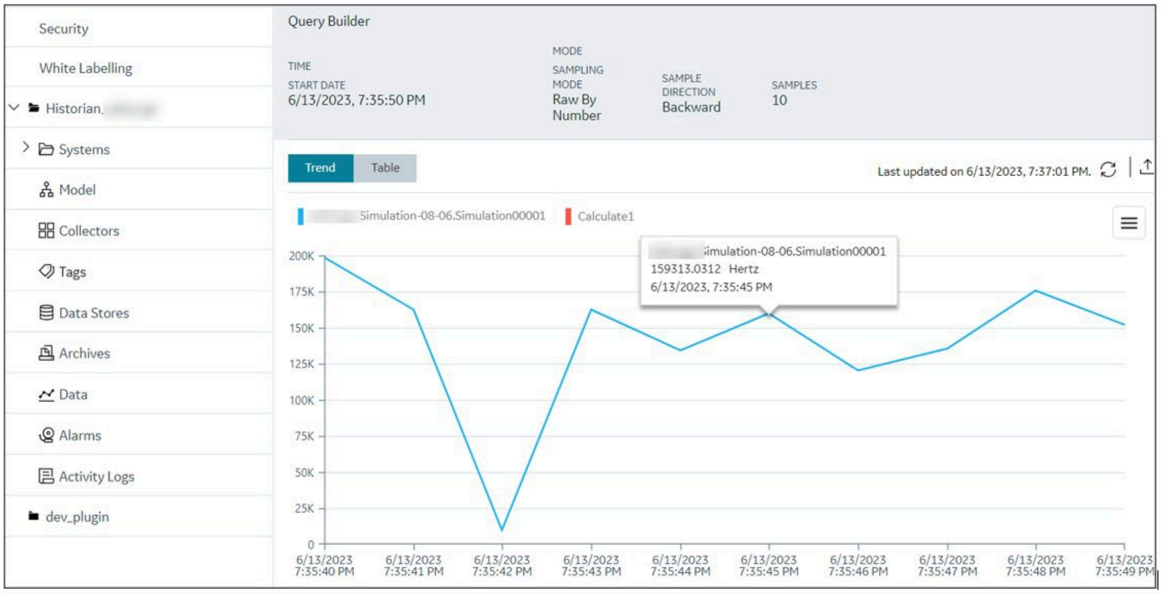
Tip:

You can also filter the values by changing the duration, sampling type, time interval, engineering units, and so on by selecting ▽.



Time Stamp	Value	QUALITY	Data Attributes
Filter	*Filter*	*Filter*	*Filter*
12/6/2022, 12:40:07 PM	61122.47266	Good	22,21,20,19,14,13,12,11,6,5,4,3
12/6/2022, 12:40:09 PM	28003.78516	Good	22,21,20,19,14,13,12,11,6,5,4,3
12/6/2022, 12:40:11 PM	176995.1406	Good	22,21,20,19,14,13,12,11,6,5,4,3
12/6/2022, 12:40:13 PM	4205.450684	Good	22,21,20,19,14,13,12,11,6,5,4,3
12/6/2022, 12:40:14 PM	174858.8594	Good	22,21,20,19,14,13,12,11,6,5,4,3
12/6/2022, 12:40:16 PM	148283.3281	Good	22,21,20,19,14,13,12,11,6,5,4,3
12/6/2022, 12:40:18 PM	2050.84375	Good	22,21,20,19,14,13,12,11,6,5,4,3
12/6/2022, 12:40:20 PM	158946.5	Good	22,21,20,19,14,13,12,11,6,5,4,3
12/6/2022, 12:40:22 PM	194219.7969	Good	22,21,20,19,14,13,12,11,6,5,4,3
12/6/2022, 12:40:23 PM	96792.50781	Good	22,21,20,19,14,13,12,11,6,5,4,3

7. For the selected tag, to view the results in a trend chart, select **Trend**.



TIME
START DATE
6/13/2023, 7:35:50 PM

MODE
SAMPLING MODE
Raw By Number

SAMPLE DIRECTION
Backward

SAMPLES
10

Last updated on 6/13/2023, 7:37:01 PM.

[redacted] Simulation-08-06.Simulation00001

Calculate1

[redacted] imulation-08-06.Simulation00001
159313.0312 Hertz
6/13/2023, 7:35:45 PM

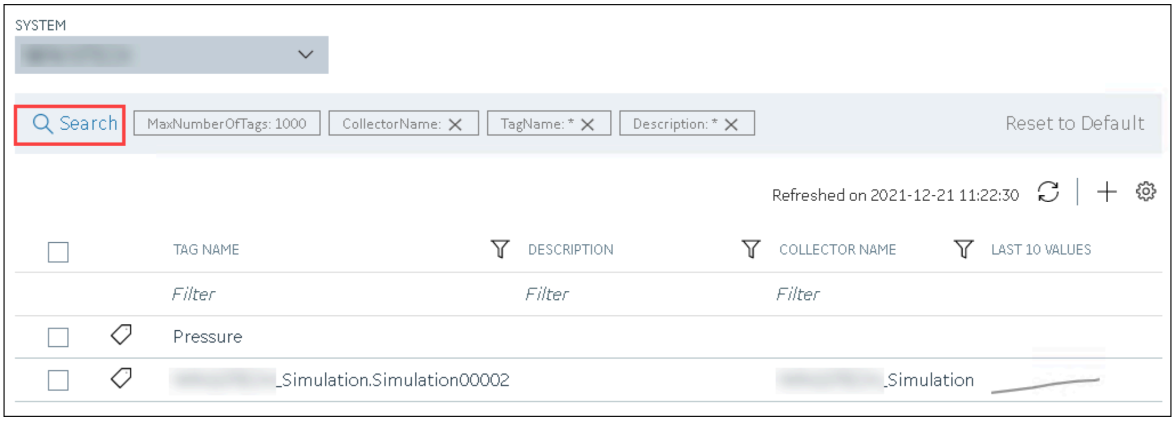
Access a Tag Alias

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 97)*.
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**.

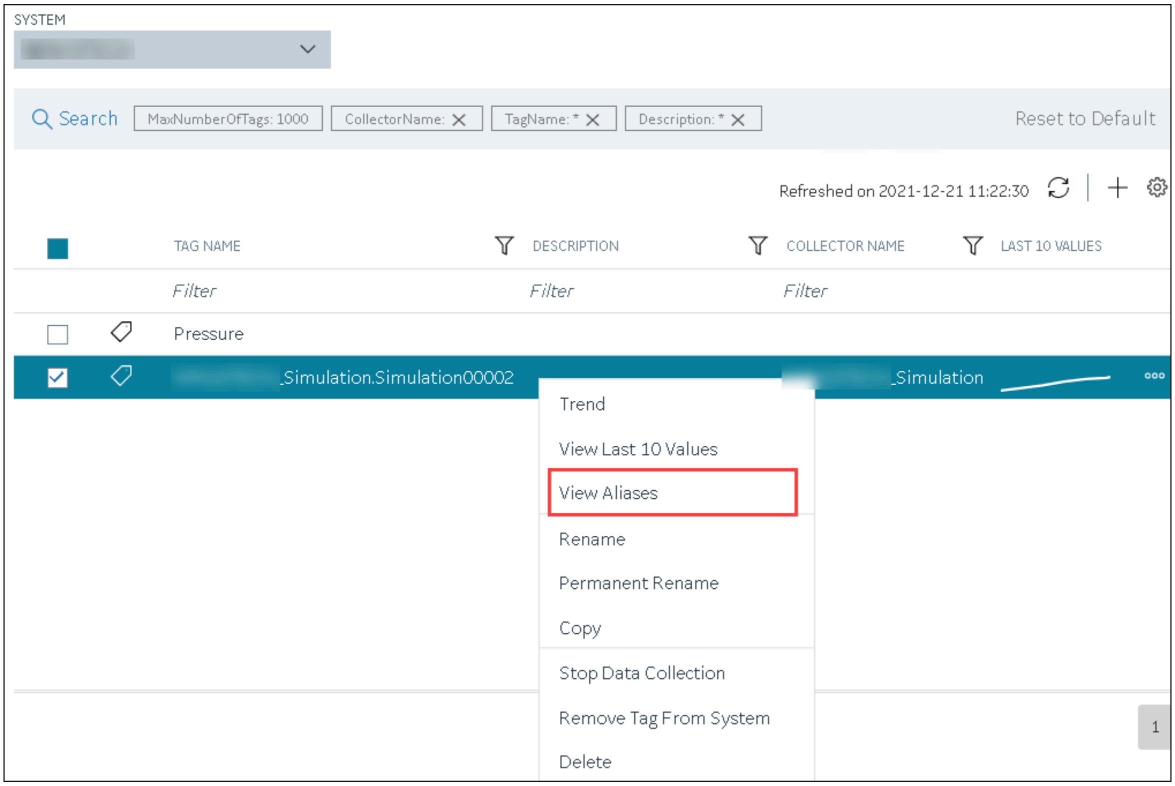


☐		TAG NAME	DESCRIPTION	COLLECTOR NAME	LAST 10 VALUES
		Filter	Filter	Filter	
☐		Pressure			
☐		[redacted]_Simulation.Simulation00002		[redacted]_Simulation	

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**.



■		TAG NAME	DESCRIPTION	COLLECTOR NAME	LAST 10 VALUES
		Filter	Filter	Filter	
☐		Pressure			
☑		[redacted]_Simulation.Simulation00002		[redacted]_Simulation	

Alternatively, you can select ✎ in the **TAG ALIAS** column.

All the tag aliases of the selected tag appear.

Export Tags as a CSV File

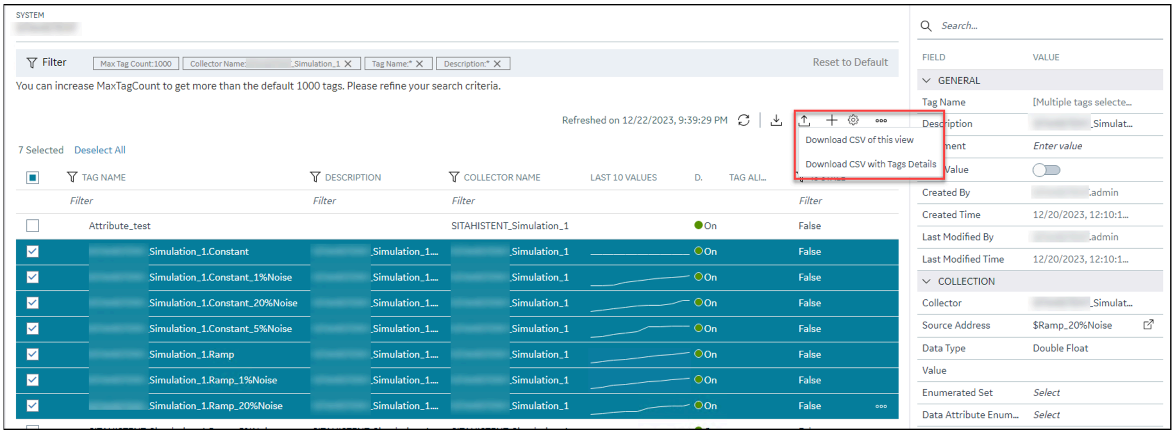
This topic describes how to export tags as a CSV file. You can export tags from Configuration Hub as a CSV file and add/modify tags in bulk and then import them. You can also import the CSV file that was exported using Excel Add-in.

> **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 97)*.
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.


SYSTEM
Filter
Max Tag Count:1000
Collector Name:[redacted]_Simulation_1
Tag Name:*
Description:*
Reset to Default
Download CSV of this view
Download CSV with Tags Details
Deselect All
Search...

You can increase MaxTagCount to get more than the default 1000 tags. Please refine your search criteria.

Refreshed on 12/22/2023, 9:39:29 PM

7 Selected

	TAG NAME	DESCRIPTION	COLLECTOR NAME	LAST 10 VALUES	D.	TAG ALI...
	Filter	Filter	Filter			Filter
☐	Attribute_test		SITAHISTENT_Simulation_1		● On	False
☑	[redacted]Simulation_1.Constant	[redacted]_Simulation_1....	[redacted]_Simulation_1		○ On	False
☑	[redacted]Simulation_1.Constant_1%Noise	[redacted]_Simulation_1....	[redacted]_Simulation_1		○ On	False
☑	[redacted]Simulation_1.Constant_20%Noise	[redacted]_Simulation_1....	[redacted]_Simulation_1		○ On	False
☑	[redacted]Simulation_1.Constant_5%Noise	[redacted]_Simulation_1....	[redacted]_Simulation_1		○ On	False
☑	[redacted]Simulation_1.Ramp	[redacted]_Simulation_1....	[redacted]_Simulation_1		○ On	False
☑	[redacted]Simulation_1.Ramp_1%Noise	[redacted]_Simulation_1....	[redacted]_Simulation_1		○ On	False
☑	[redacted]Simulation_1.Ramp_20%Noise	[redacted]_Simulation_1....	[redacted]_Simulation_1		○ On	False

FIELD	VALUE
GENERAL	
Tag Name	[Multiple tags selecte...
Description	[redacted]_Simulat...
ment	Enter value
Value	○
Created By	[redacted].admin
Created Time	12/20/2023, 12:10:1...
Last Modified By	[redacted].admin
Last Modified Time	12/20/2023, 12:10:1...
COLLECTION	
Collector	[redacted]_Simulat...
Source Address	$Ramp_20%Noise
Data Type	Double Float
Value	
Enumerated Set	Select
Data Attribute Enum...	Select

4. Enter a name for the file and save it in a location as needed.

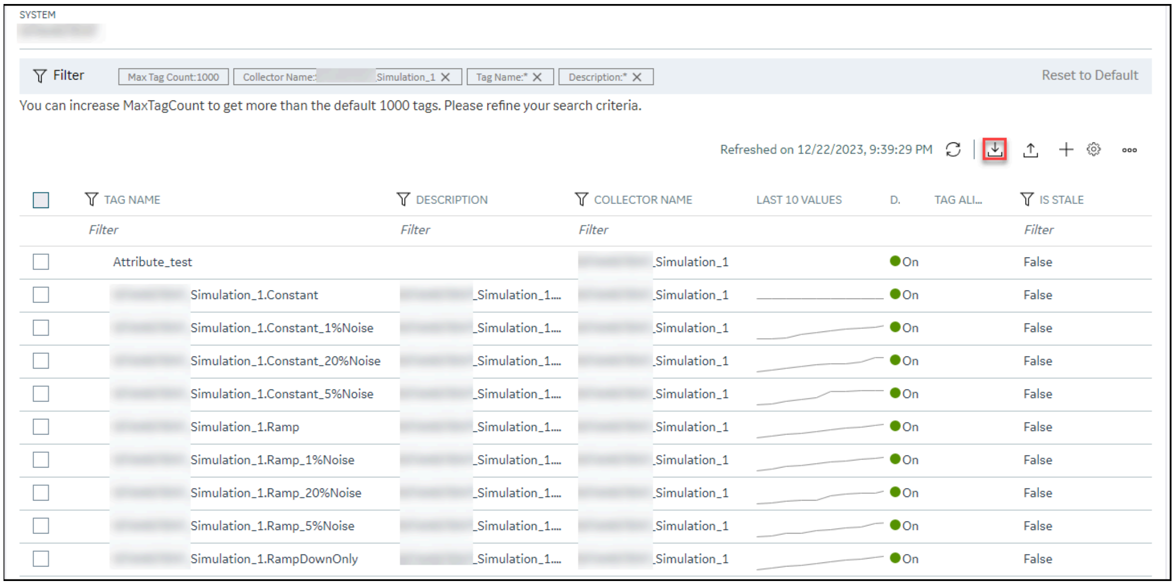
Import Tags from a CSV File

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 97)*.
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 ⤓.

   SYSTEM

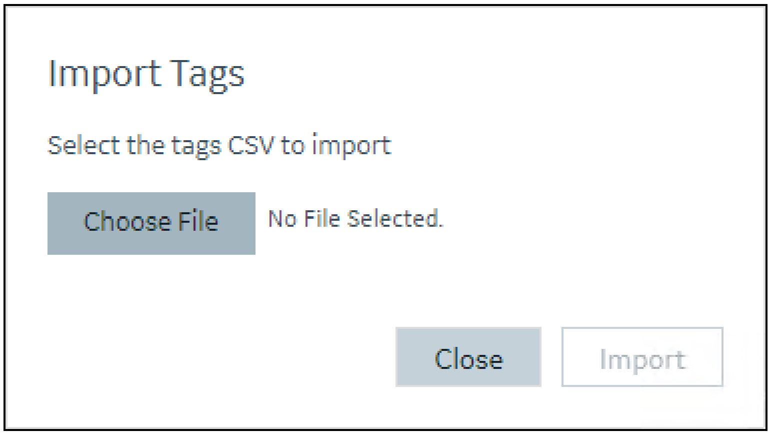
   Filter Max Tag Count:1000 Collector Name: _Simulation_1 ✕ Tag Name:* ✕ Description:* ✕ Reset to Default

   You can increase MaxTagCount to get more than the default 1000 tags. Please refine your search criteria.

   Refreshed on 12/22/2023, 9:39:29 PM

☐	TAG NAME	DESCRIPTION	COLLECTOR NAME	LAST 10 VALUES	D.	TAG ALI...	IS STALE
	Filter	Filter	Filter				Filter
☐	Attribute_test		_Simulation_1		● On		False
☐	Simulation_1.Constant	_Simulation_1....	_Simulation_1		● On		False
☐	Simulation_1.Constant_1%Noise	_Simulation_1....	_Simulation_1		● On		False
☐	Simulation_1.Constant_20%Noise	_Simulation_1....	_Simulation_1		● On		False
☐	Simulation_1.Constant_5%Noise	_Simulation_1....	_Simulation_1		● On		False
☐	Simulation_1.Ramp	_Simulation_1....	_Simulation_1		● On		False
☐	Simulation_1.Ramp_1%Noise	_Simulation_1....	_Simulation_1		● On		False
☐	Simulation_1.Ramp_20%Noise	_Simulation_1....	_Simulation_1		● On		False
☐	Simulation_1.Ramp_5%Noise	_Simulation_1....	_Simulation_1		● On		False
☐	Simulation_1.RampDownOnly	_Simulation_1....	_Simulation_1		● On		False

   The **Import Tags** page appears.



Import Tags

Select the tags CSV to import

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.

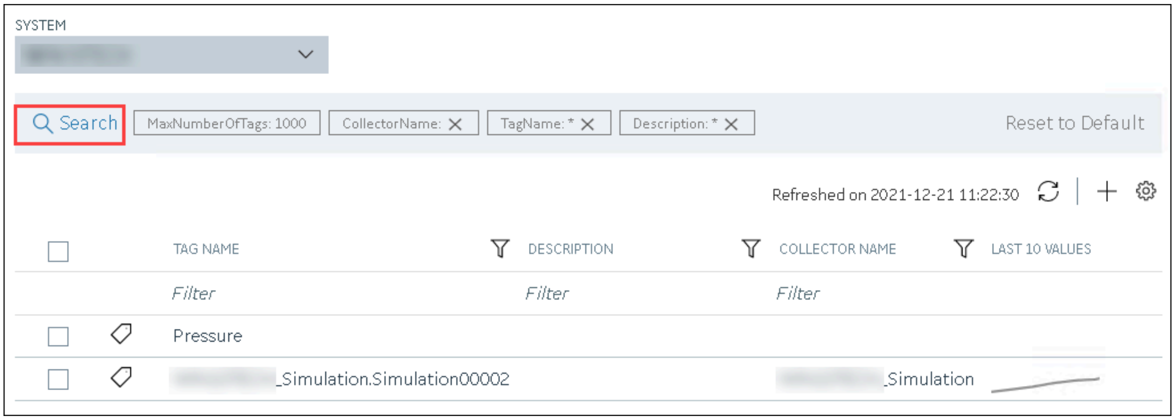
> **Note:**
> If a tag name contains a comma or a semicolon, you cannot view the tag alias.

- **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 97)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.

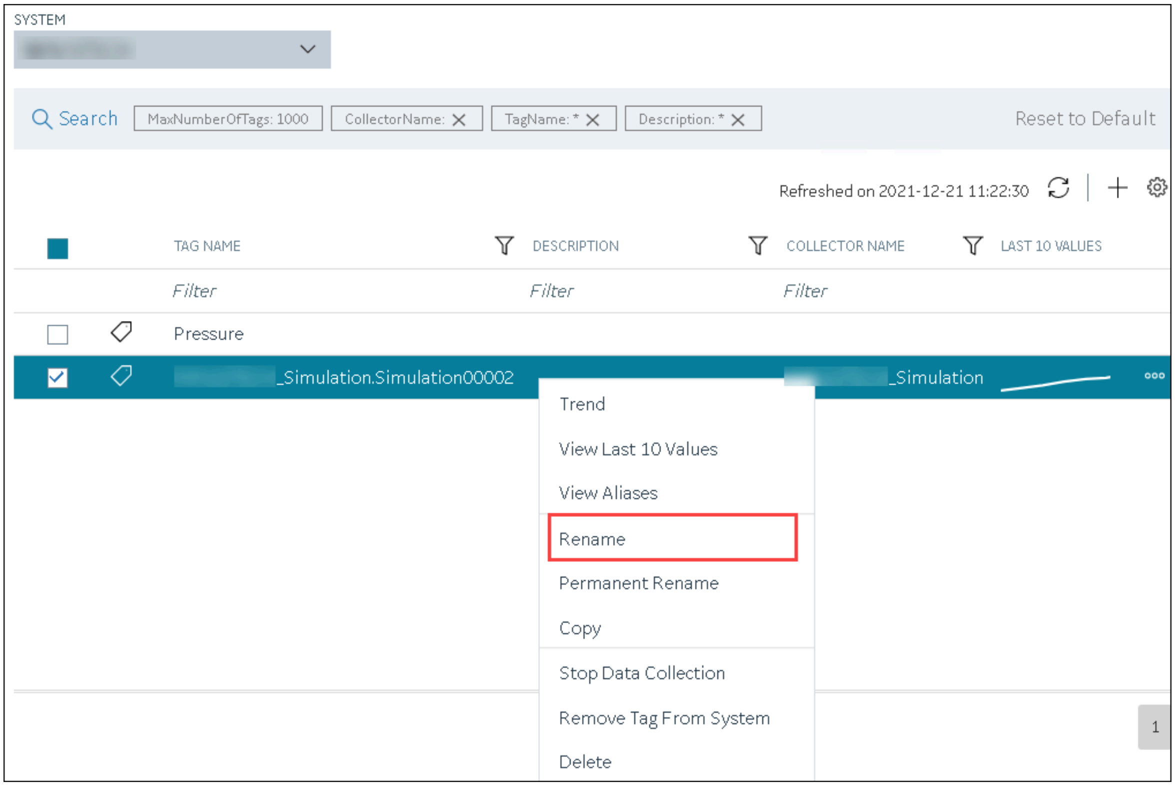
   A list of all the tags appears.

3. If you want to narrow down the search results, select **Search**.

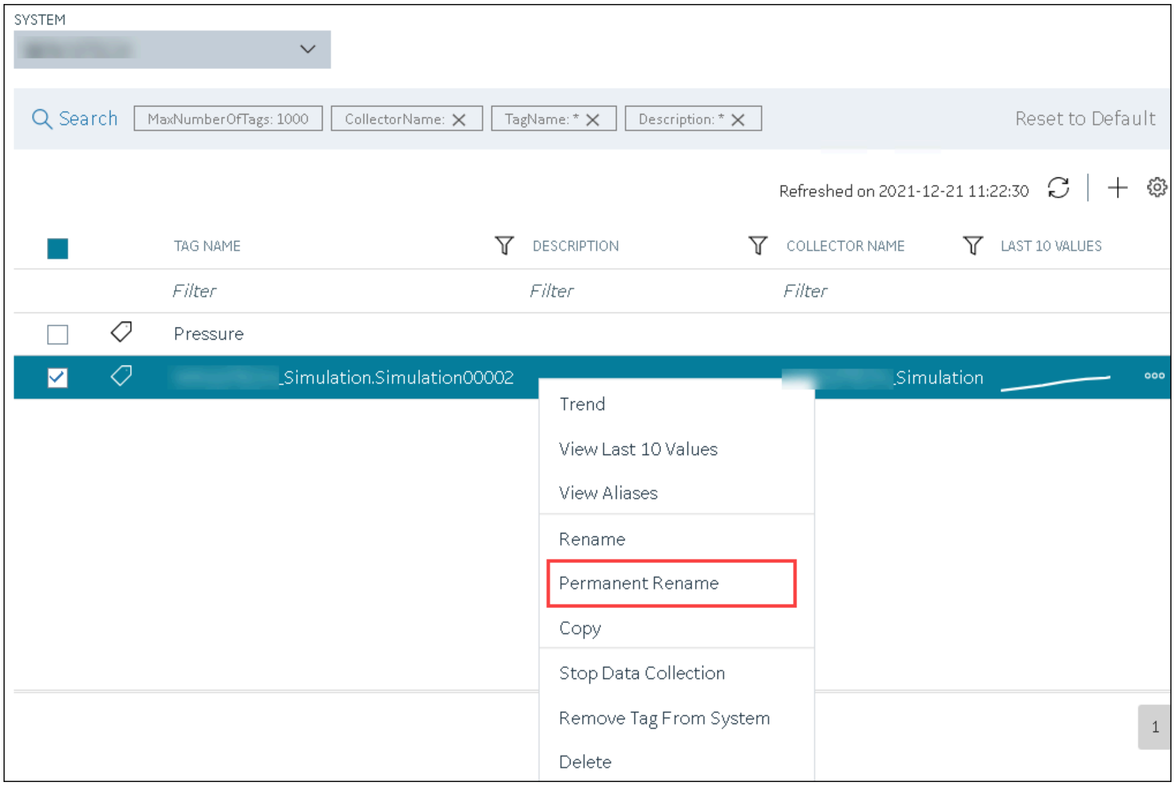


Enter the search criteria, and then select **Search**. You can add more attributes by selecting **Add Attribute**. You can enter a name or a value partially or use the wildcard character asterisk (*). The list of tags are filtered based on the search criteria.

4. Right-click the tag that you want to rename (or select ooo).
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:

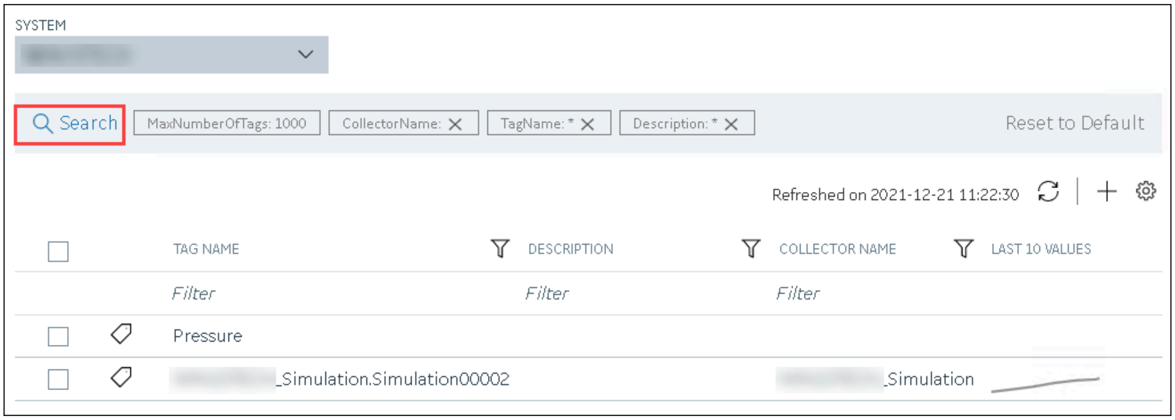
- If the tag is used as a trigger, reassign the trigger.
- Restart the collector instance *(on page 728)*.

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.

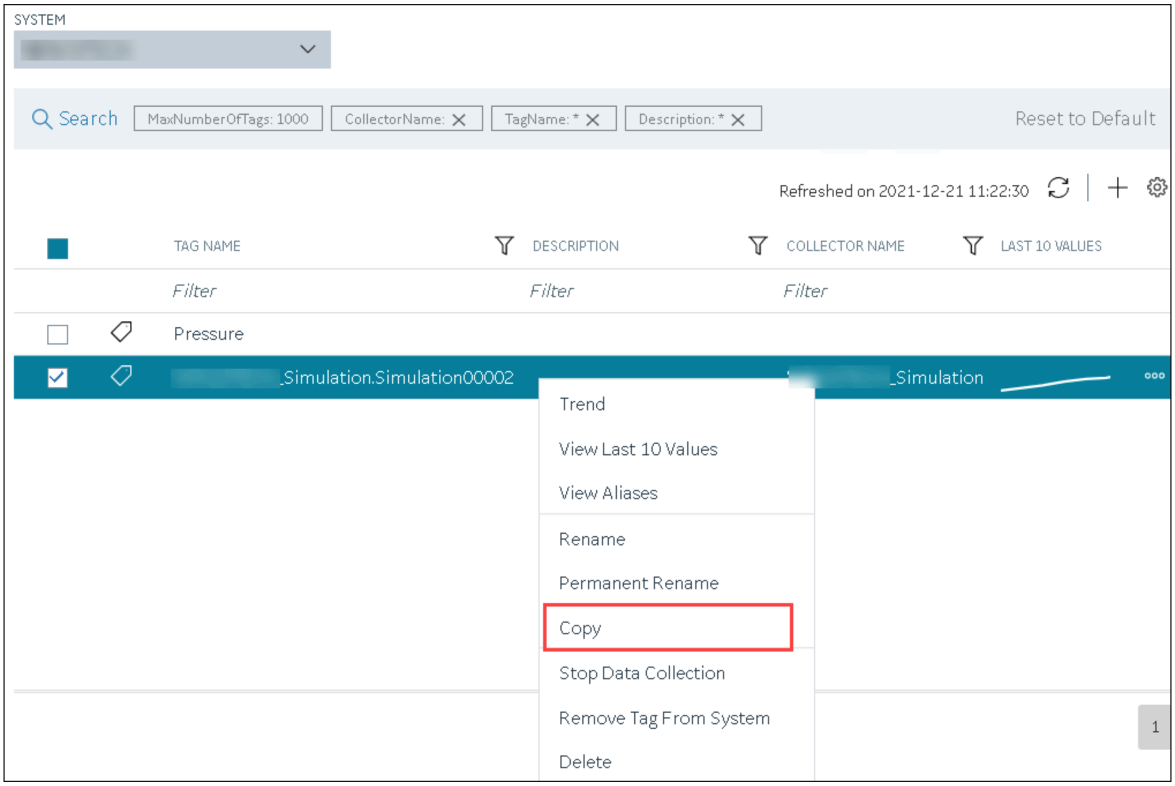
1. Access Configuration Hub *(on page 97)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.
   A list of all the tags appears.

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**.

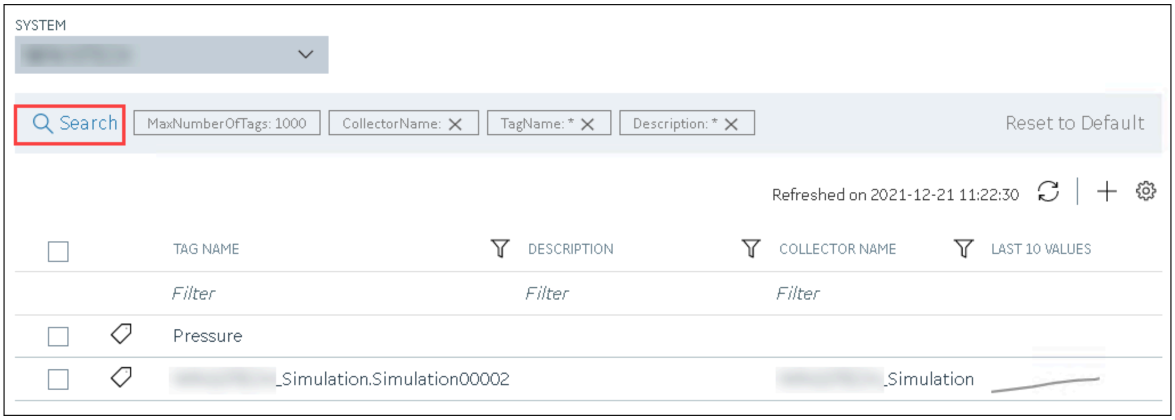
   The tag is copied, inheriting the properties of the original tag. In addition, data collection begins for the tag.

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 785)* or delete it *(on page 786)*.

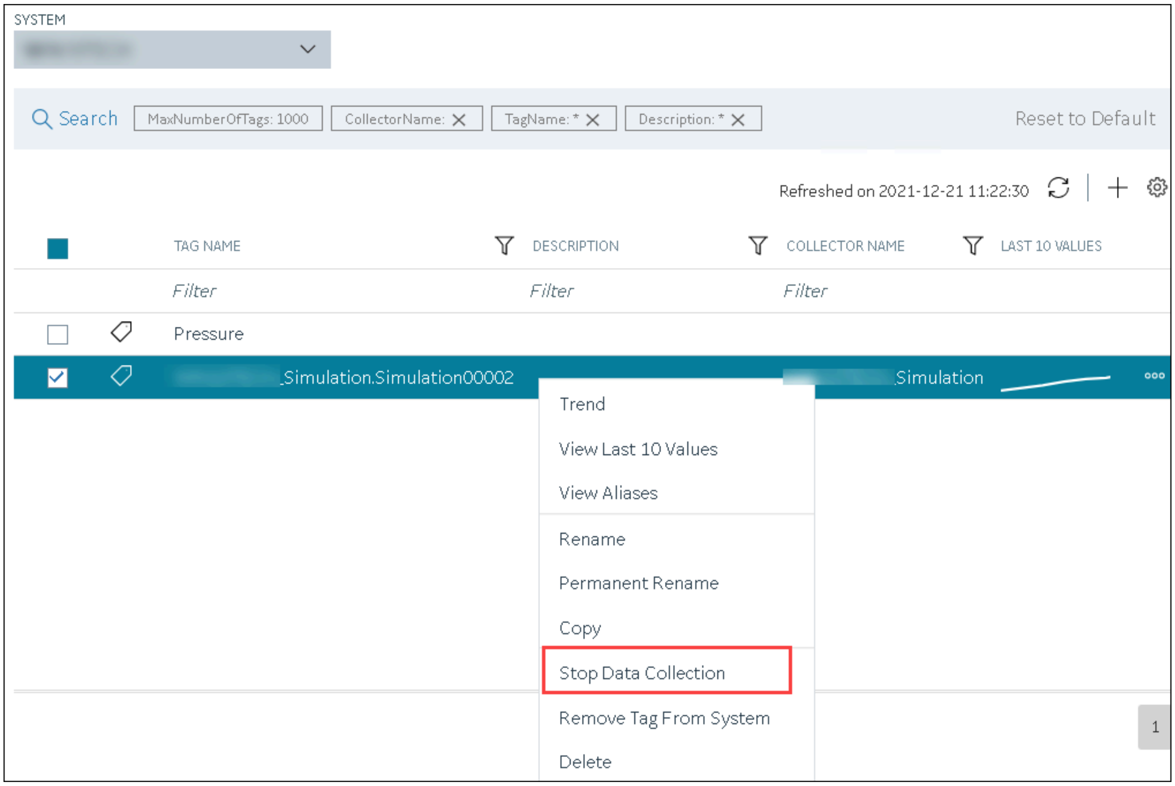
1. Access Configuration Hub *(on page 97)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.

   A list of all the tags appears.
3. If you want to narrow down the search results, select **Search**.

TAG NAME	DESCRIPTION	COLLECTOR NAME	LAST 10 VALUES
Filter	Filter	Filter	
Pressure			
_Simulation.Simulation00002		_Simulation	

   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.

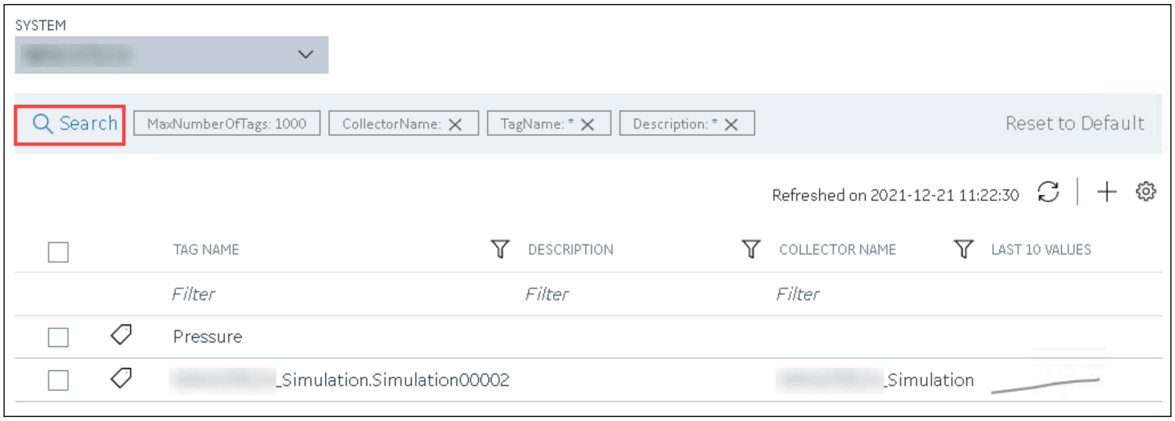
5. Select **Stop**.

   Data collection is stopped for the tag.

Start the Data Collection of a Tag

1. Access Configuration Hub *(on page 97)*.
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**.

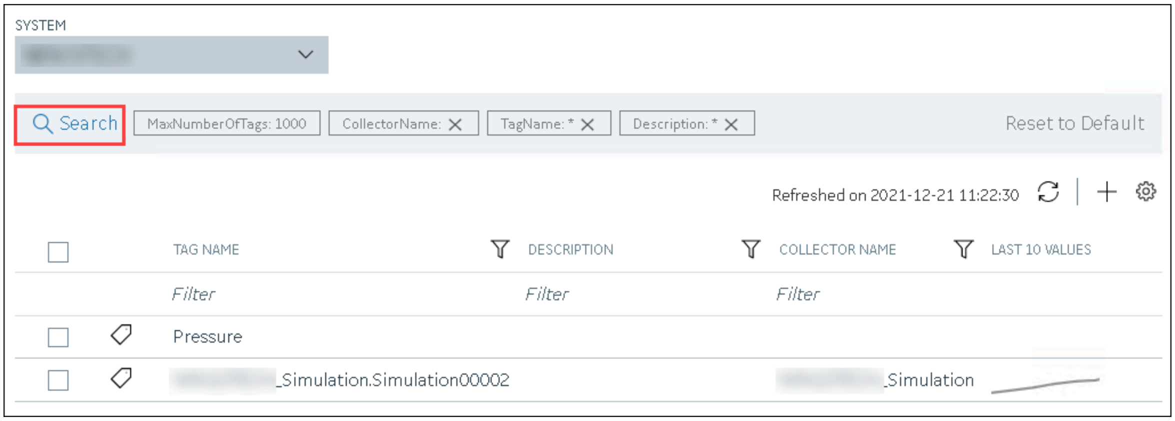
   Data collection is started for the tag.

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 786)*. Or, you can choose to stop the data collection *(on page 783)* for the tag, which allows you to resume the data collection *(on page 784)* later.

1. Access Configuration Hub *(on page 97)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.

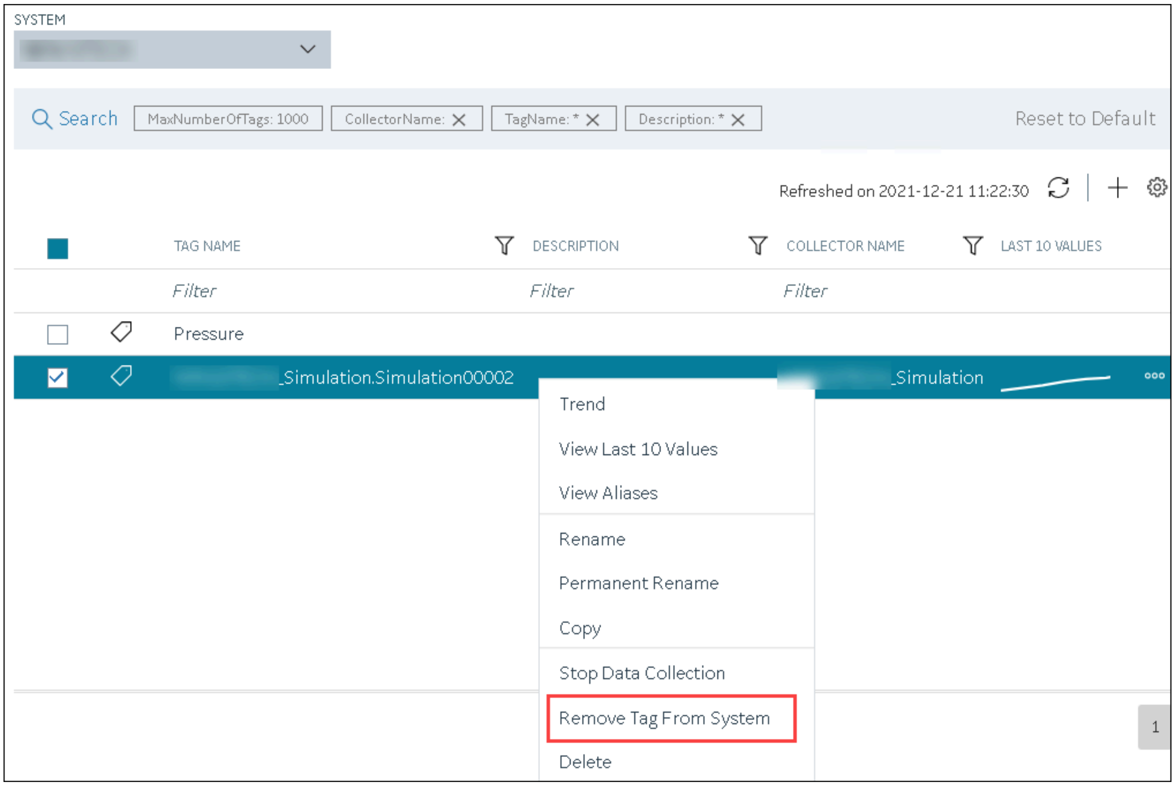
   A list of all the tags appears.
3. If you want to narrow down the search results, select **Search**.



□		TAG NAME	DESCRIPTION	COLLECTOR NAME	LAST 10 VALUES
		Filter	Filter	Filter	
□		Pressure			
□		[redacted]_Simulation.Simulation00002		[redacted]_Simulation	

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 want to remove (or select ⋯), and then select **Remove Tag From System**.



SYSTEM [redacted]

Search MaxNumberOfTags: 1000 CollectorName: TagName: * Description: *

Refreshed on 2021-12-21 11:22:30

		TAG NAME	DESCRIPTION	COLLECTOR NAME	LAST 10 VALUES
		Filter	Filter	Filter	
□		Pressure			
☑		[redacted]_Simulation.Simulation00002		[redacted]_Simulation	

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.

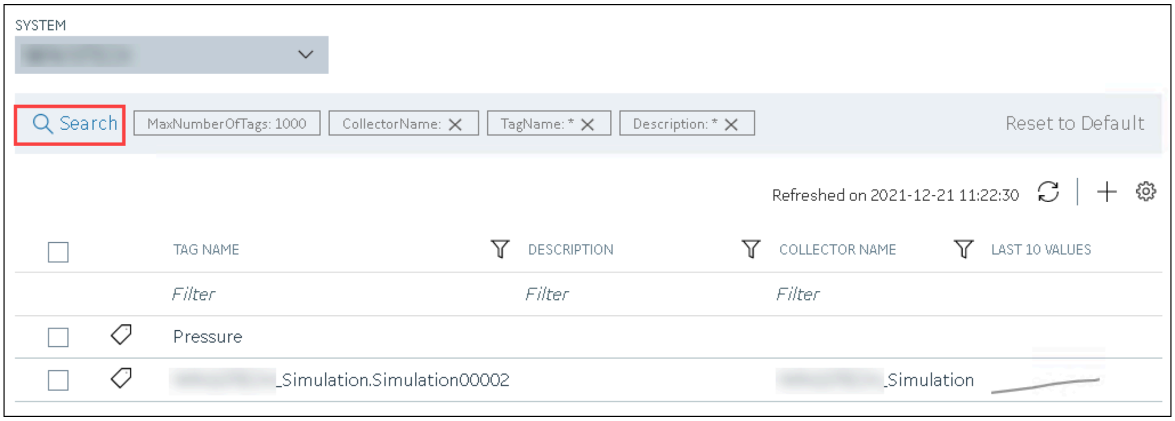
Delete a Tag

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 785)*. Or, you can choose to stop the data collection *(on page 783)* for the tag, which allows you to resume the data collection *(on page 784)* later.

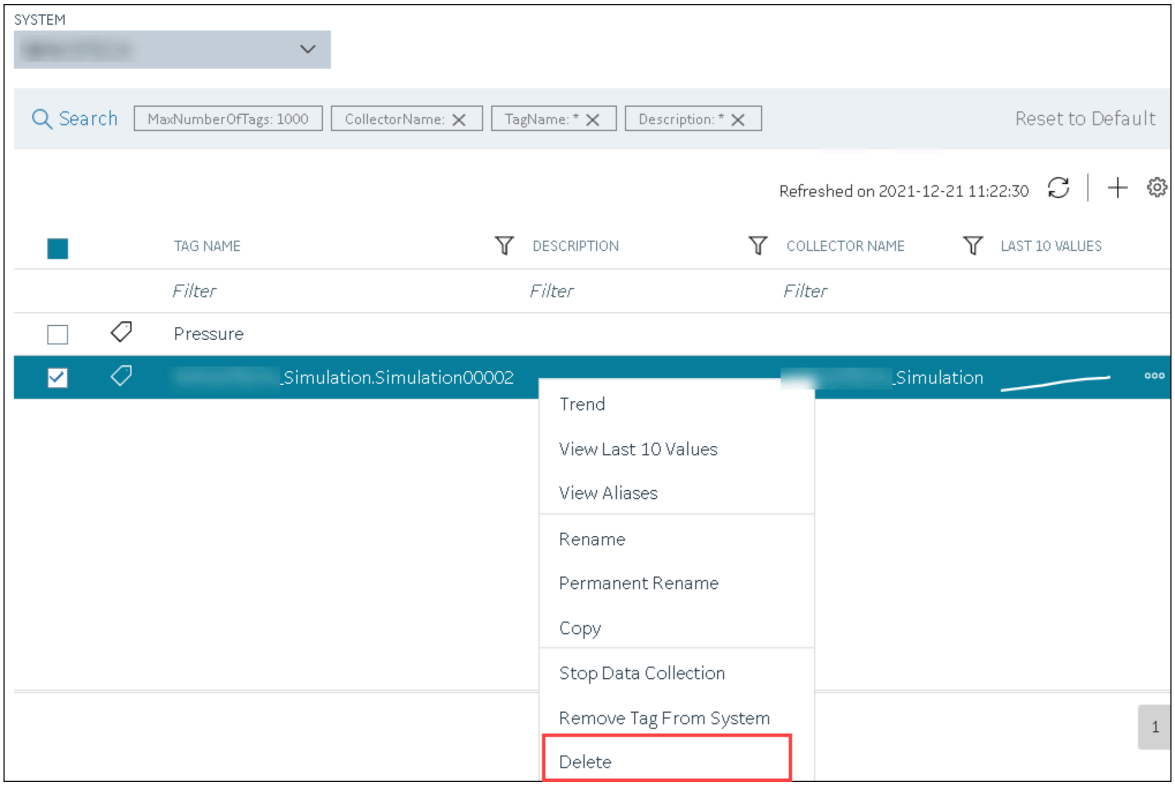
1. Access Configuration Hub *(on page 97)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Tags**.

   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 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)*.

> **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:

$$5000\times\frac{1}{1}\times\frac{1}{5}\times\frac{5}{100}\times\frac{4}{1}\times\frac{3600}{1}\times\frac{24}{1}\times\frac{1}{1024\times1024}\times30=494\frac{MB}{Month}$$

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

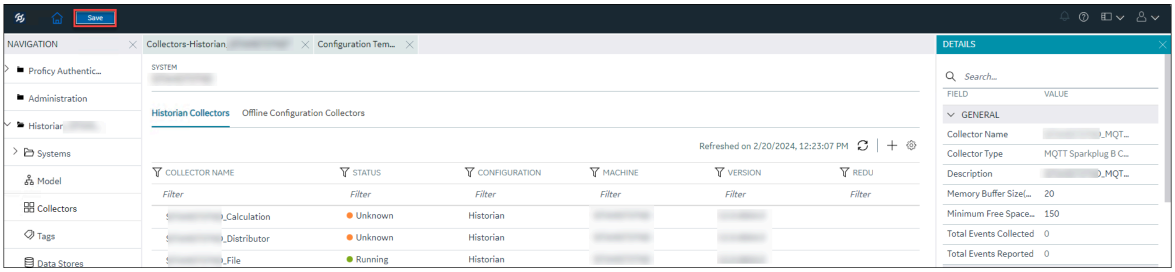
An archive size calculator tool is available to estimate archive size and collector compression based on a tag that has already been configured or based on your inputs. Log on to http://digitalsupport.ge.com to download this tool and other GE Intelligent Platforms freeware product solutions.

Access an Archive

1. Access Configuration Hub *(on page 97)*.
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: ◦ **Current**: The archive is actively accepting data. ◦ **Active**: The archive contains data but is not currently accepting data. ◦ **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**.



SYSTEM

[redacted]

Historian Collectors | Offline Configuration Collectors

Refreshed on 2/20/2024, 12:23:07 PM

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSION	REDU
Filter	Filter	Filter	Filter	Filter	Filter
[redacted]_Calculation	● Unknown	Historian	[redacted]	[redacted]	
[redacted]_Distributor	● Unknown	Historian	[redacted]	[redacted]	
[redacted]_File	● Running	Historian	[redacted]	[redacted]	

FIELD	VALUE
GENERAL	
Collector Name	[redacted]_MQT...
Collector Type	MQTT Sparkplug B C...
Description	[redacted]_MQT...
Memory Buffer Size(...	20
Minimum Free Space...	150
Total Events Collected	0
Total Events Reported	0

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 793)*.

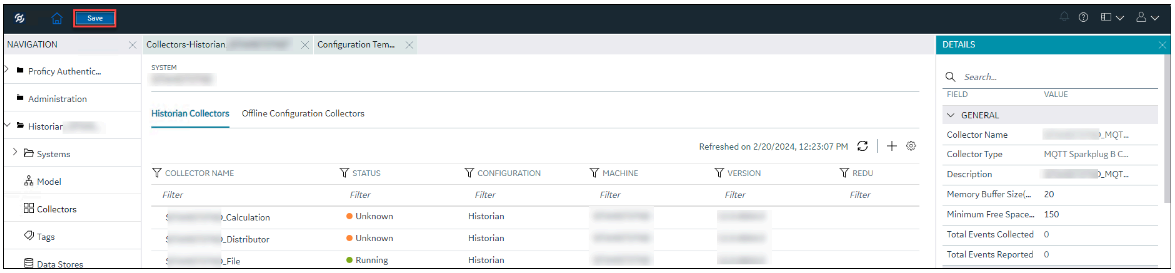
1. Access Configuration Hub *(on page 97)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Archives**.
   The **Data Stores** section appears.
3. Select the data store in which you want to create archives automatically.
   The **DETAILS** section displays 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. Select one of the following options: ◦ **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**). ◦ **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 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

Field	Description
	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**.



SYSTEM
[redacted]

Refreshed on 2/20/2024, 12:23:07 PM

COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VERSION	REDU
Filter	*Filter*	*Filter*	*Filter*	*Filter*	*Filter*
[redacted]_Calculation	● Unknown	Historian	[redacted]	[redacted]	
[redacted]_Distributor	● Unknown	Historian	[redacted]	[redacted]	
[redacted]_File	● Running	Historian	[redacted]	[redacted]	

FIELD	VALUE
GENERAL	
Collector Name	[redacted]_MQT...
Collector Type	MQTT Sparkplug B C...
Description	[redacted]_MQT...
Memory Buffer Size(...	20
Minimum Free Space...	150
Total Events Collected	0
Total Events Reported	0

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 791)*.

1. Access Configuration Hub *(on page 97)*.
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

Field	Description
	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. **Important:** 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)*.

This topic describes how to back up an archive manually. You can also back up an archive automatically *(on page)*.

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 97)*.
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.

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 97)*.
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 97)*.
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 97)*.
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 97)*.
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.

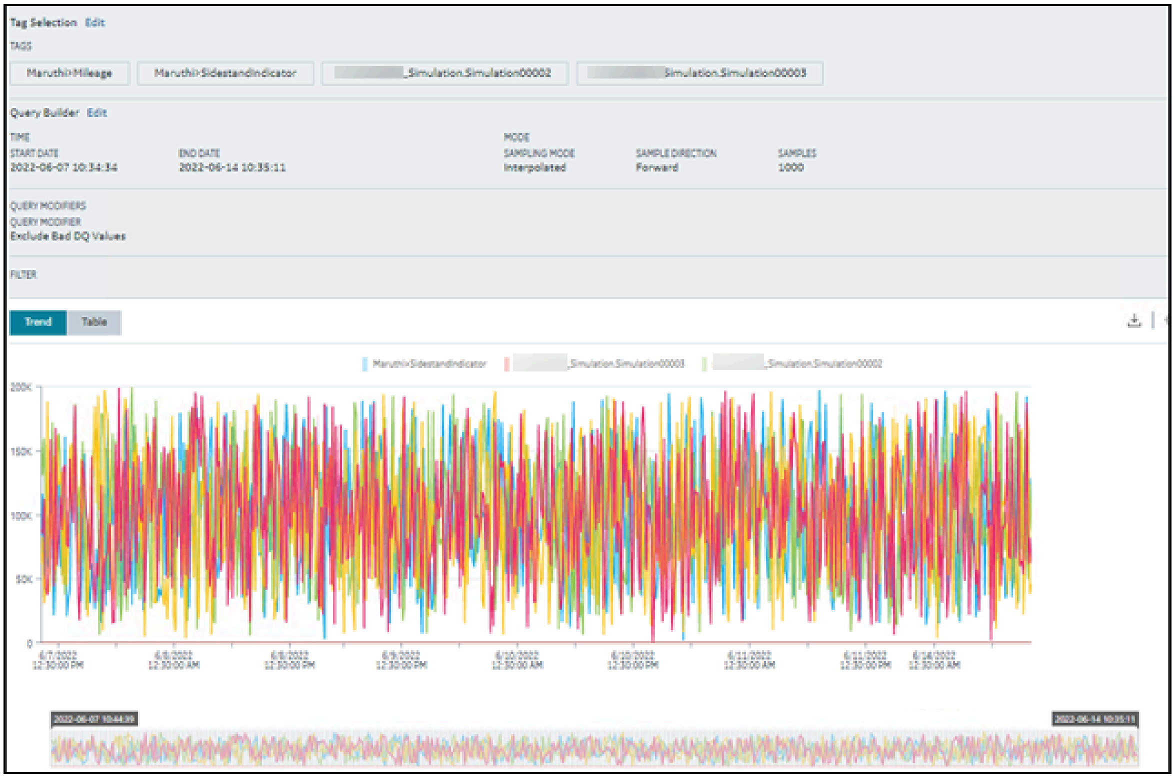
Field	Description
START DATE/TIME	Enter the start date and time for the query.
END DATE/TIME	Enter the end date and time for the query.
SAMPLING MODES	Select the sampling mode for the query: ◦ **Calculated** *(on page 1406)*: Returns the result of a calculation on tag values (for example, count, maximum, delta). ◦ **Current Value** *(on page 1378)*: Returns a single sample containing the current value of the tag. Retrieves the timestamp, value, and quality. ◦ **Interpolated** *(on page 1381)*: Returns the interpolated tag values. ◦ **Interpolated to Raw** ◦ **Lab** *(on page 1380)*: 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. ◦ **Lab to Raw** ◦ **Raw By Number** *(on page 1390)*: 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 1390)*: 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. ◦ **Trend** *(on page 1398)*: Returns significant points, which are raw samples. These points are determined by finding the raw minimum and raw maximum values within each interval. ◦ **Trend to Raw** ◦ **Trend to Raw2** ◦ **Trend2** *(on page 1402)*: 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 1406)*.
SAMPLING DIRECTION	The direction in which you want to retrieve the query results: ◦ **Forward**: Returns query results starting from the start data to the end date. ◦ **Backward**: Returns query results starting from the end date to the start data.
SAMPLE INCREMENT	Identifies the amount of data samples in each sample: ◦ **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. ◦ **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 1492)*.
Filters	You can enter your filter conditions using the Filter tag, the Filter Condition, and the Filter Comparison Value. In the Filter Tag Name field, select the tag you want to enable filtering with. In the Condition field, select your comparison condition. 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.



Tag Selection Edit

TAGS

Maruthi>Mileage | Maruthi>SidestandIndicator | [redacted]_Simulation.Simulation00002 | [redacted]Simulation.Simulation00003

Query Builder Edit

TIME

START DATE: 2022-06-07 10:34:34

END DATE: 2022-06-14 10:35:11

MODE

SAMPLING MODE: Interpolated

SAMPLE DIRECTION: Forward

SAMPLES: 1000

QUERY MODIFIERS

QUERY MODIFIER: Exclude Bad DQ Values

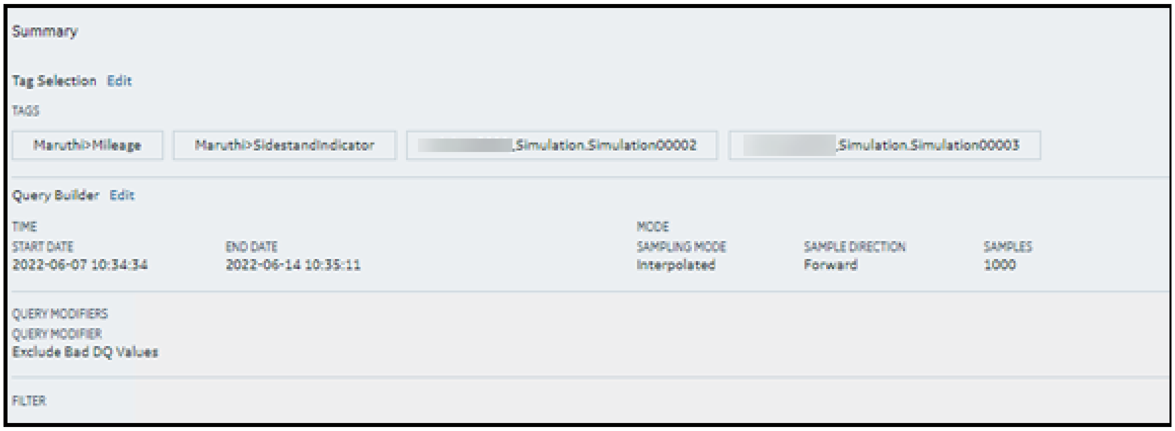
FILTER

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.



Summary

Tag Selection Edit

TAGS

Maruthi>Mileage | Maruthi>SidestandIndicator | [redacted]_Simulation.Simulation00002 | [redacted]_Simulation.Simulation00003

Query Builder Edit

TIME		MODE		
START DATE	END DATE	SAMPLING MODE	SAMPLE DIRECTION	SAMPLES
2022-06-07 10:34:34	2022-06-14 10:35:11	Interpolated	Forward	1000

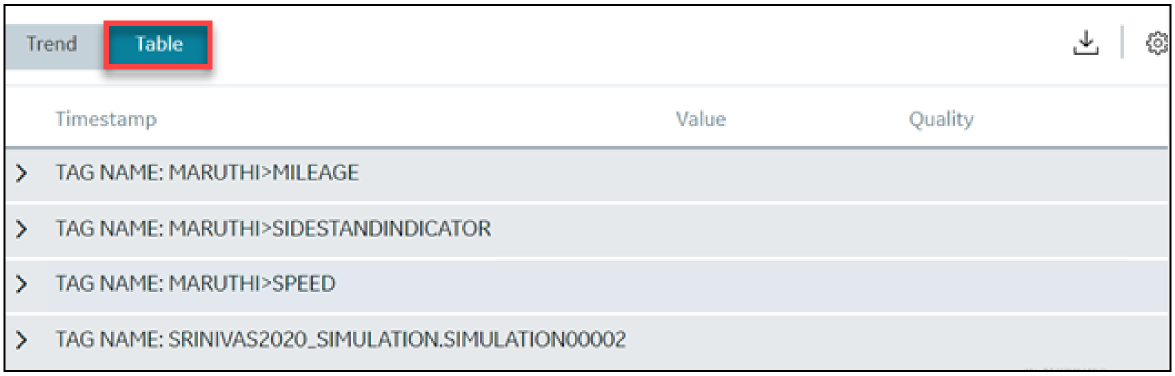
QUERY MODIFIERS

QUERY MODIFIER

Exclude Bad DQ Values

FILTER

If you want to view the results in a table, select **Table**.



Timestamp	Value	Quality
TAG NAME: MARUTHI>MILEAGE		
TAG NAME: MARUTHI>SIDESTANDINDICATOR		
TAG NAME: MARUTHI>SPEED		
TAG NAME: SRINIVAS2020_SIMULATION.SIMULATION00002		

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 804)*.

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.

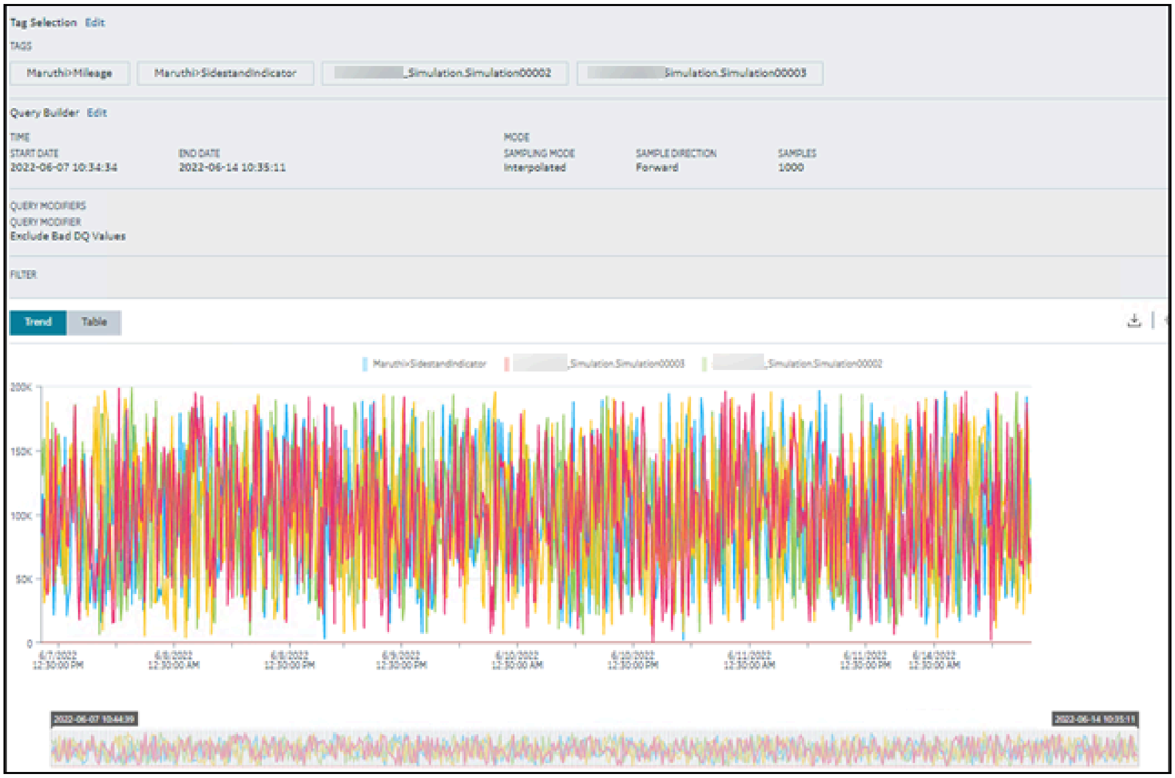
Writing data involves the following steps:

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 97)*.
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. 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.



Tag Selection Edit

TAGS

Maruthi>Mileage | Maruthi>SidestandIndicator | [redacted]_Simulation.Simulation00002 | [redacted]Simulation.Simulation00003

Query Builder Edit

TIME

START DATE 2022-06-07 10:34:34

END DATE 2022-06-14 10:35:11

MODE

SAMPLING MODE Interpolated

SAMPLE DIRECTION Forward

SAMPLES 1000

QUERY MODIFIERS

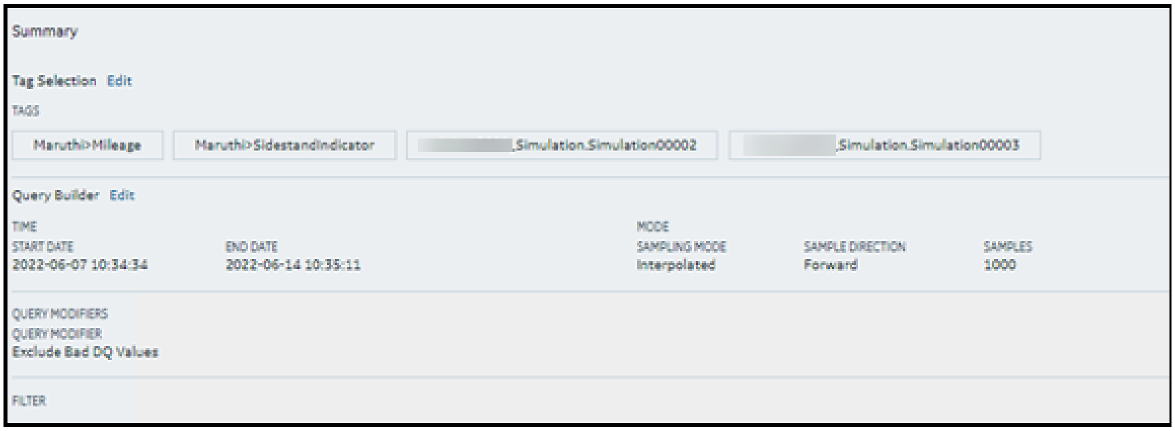
QUERY MODIFIER Exclude Bad DQ Values

FILTER

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.



Summary

Tag Selection Edit

TAGS

Maruthi>Mileage | Maruthi>SidestandIndicator | [redacted]_Simulation.Simulation00002 | [redacted]_Simulation.Simulation00003

Query Builder Edit

TIME

START DATE 2022-06-07 10:34:34

END DATE 2022-06-14 10:35:11

MODE

SAMPLING MODE Interpolated

SAMPLE DIRECTION Forward

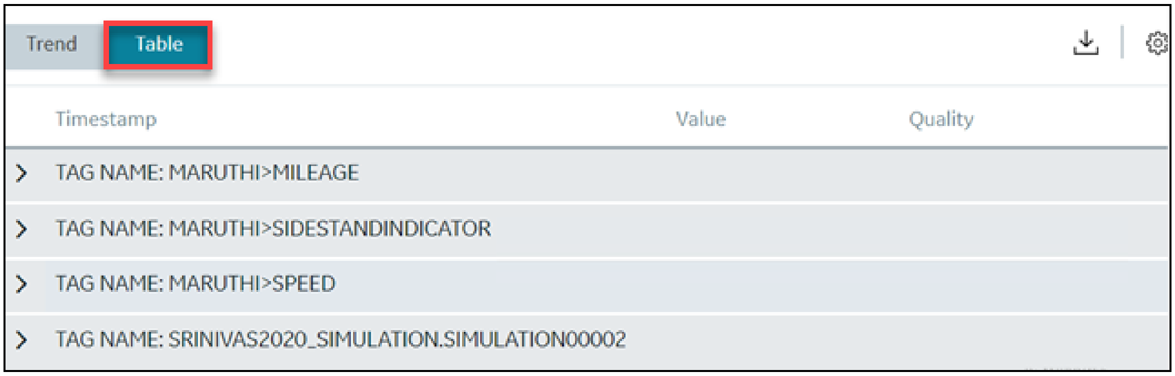
SAMPLES 1000

QUERY MODIFIERS

QUERY MODIFIER Exclude Bad DQ Values

FILTER

If you want to view the results in a table, select **Table**.



	Timestamp	Value	Quality
>	TAG NAME: MARUTHI>MILEAGE		
>	TAG NAME: MARUTHI>SIDESTANDINDICATOR		
>	TAG NAME: MARUTHI>SPEED		
>	TAG NAME: SRINIVAS2020_SIMULATION.SIMULATION00002		

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 804)*.

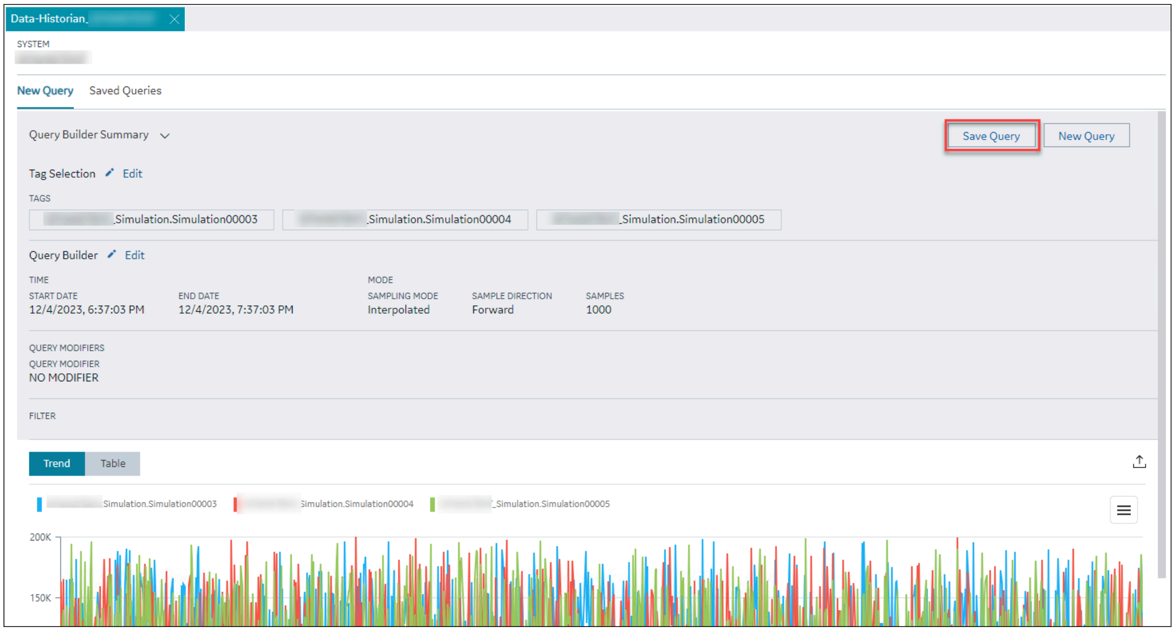
Save a Query

This topic describes how to save a Query to read tags data.

1. Access Configuration Hub *(on page 97)*.
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 801)*.
5. Select **Generate Report**.

   The query results are plotted on a trend chart.



SYSTEM
[redacted]

Tag Selection

TAGS
[redacted]_Simulation.Simulation00003 [redacted]_Simulation.Simulation00004 [redacted]_Simulation.Simulation00005

Query Builder

TIME
START DATE 12/4/2023, 6:37:03 PM
END DATE 12/4/2023, 7:37:03 PM

MODE
SAMPLING MODE Interpolated
SAMPLE DIRECTION Forward
SAMPLES 1000

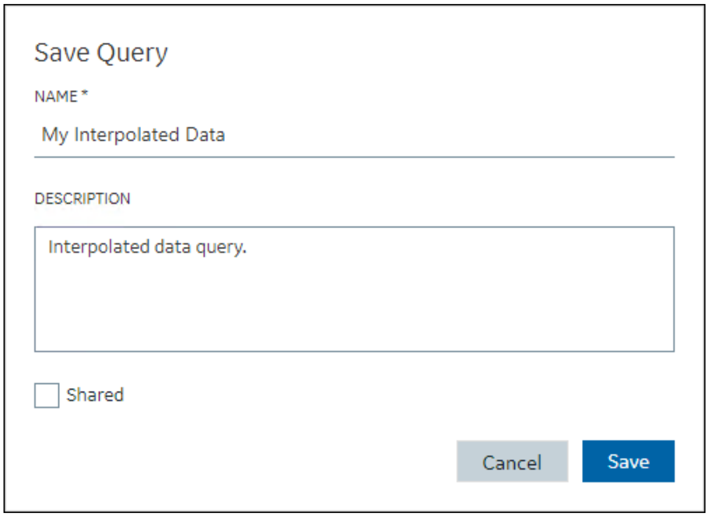
QUERY MODIFIERS
QUERY MODIFIER
NO MODIFIER

FILTER

[redacted]Simulation.Simulation00003 [redacted]Simulation.Simulation00004 [redacted]_Simulation.Simulation00005

6. Select **Save Query**.

   The **Save Query** window appears.



NAME *
My Interpolated Data

DESCRIPTION
Interpolated data query.

□ Shared

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. **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.

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 807)* the query.

Run a Saved Query

After you create and save a query, you can run the query and see the query results.

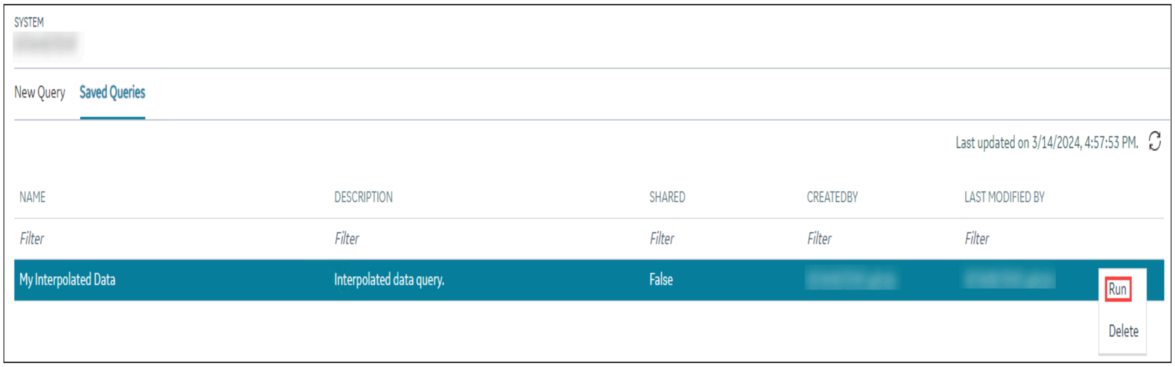
This topic describes how to run a saved query.

1. Access Configuration Hub *(on page 97)*.
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**.



Last updated on 3/14/2024, 4:57:53 PM.

NAME	DESCRIPTION	SHARED	CREATEDBY	LAST MODIFIED BY
Filter	*Filter*	*Filter*	*Filter*	*Filter*
My Interpolated Data	Interpolated data query.	False	[redacted]	[redacted]

The query results are plotted on a trend chart.

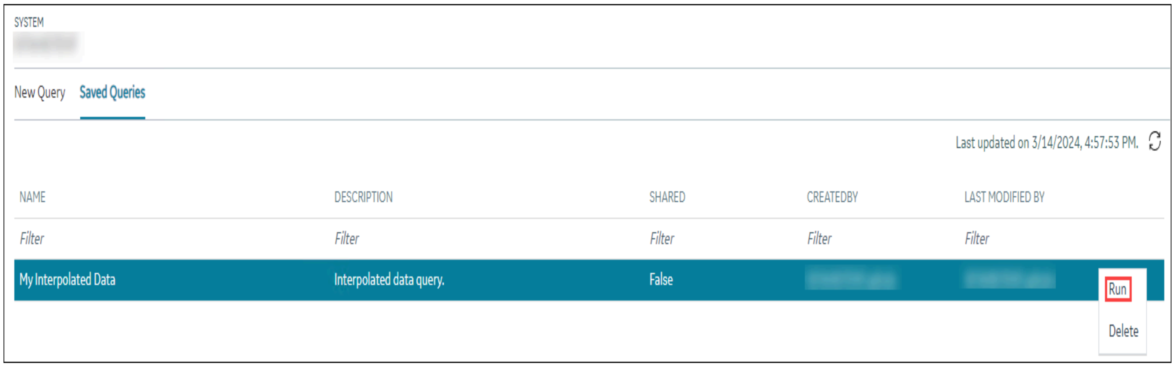
Update a Saved Query

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 97)*.
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**.



SYSTEM

[redacted]

Last updated on 3/14/2024, 4:57:53 PM.

NAME	DESCRIPTION	SHARED	CREATEDBY	LAST MODIFIED BY
Filter	*Filter*	*Filter*	*Filter*	*Filter*
My Interpolated Data	Interpolated data query.	False	[redacted]	[redacted]

The query results are plotted on a trend chart.

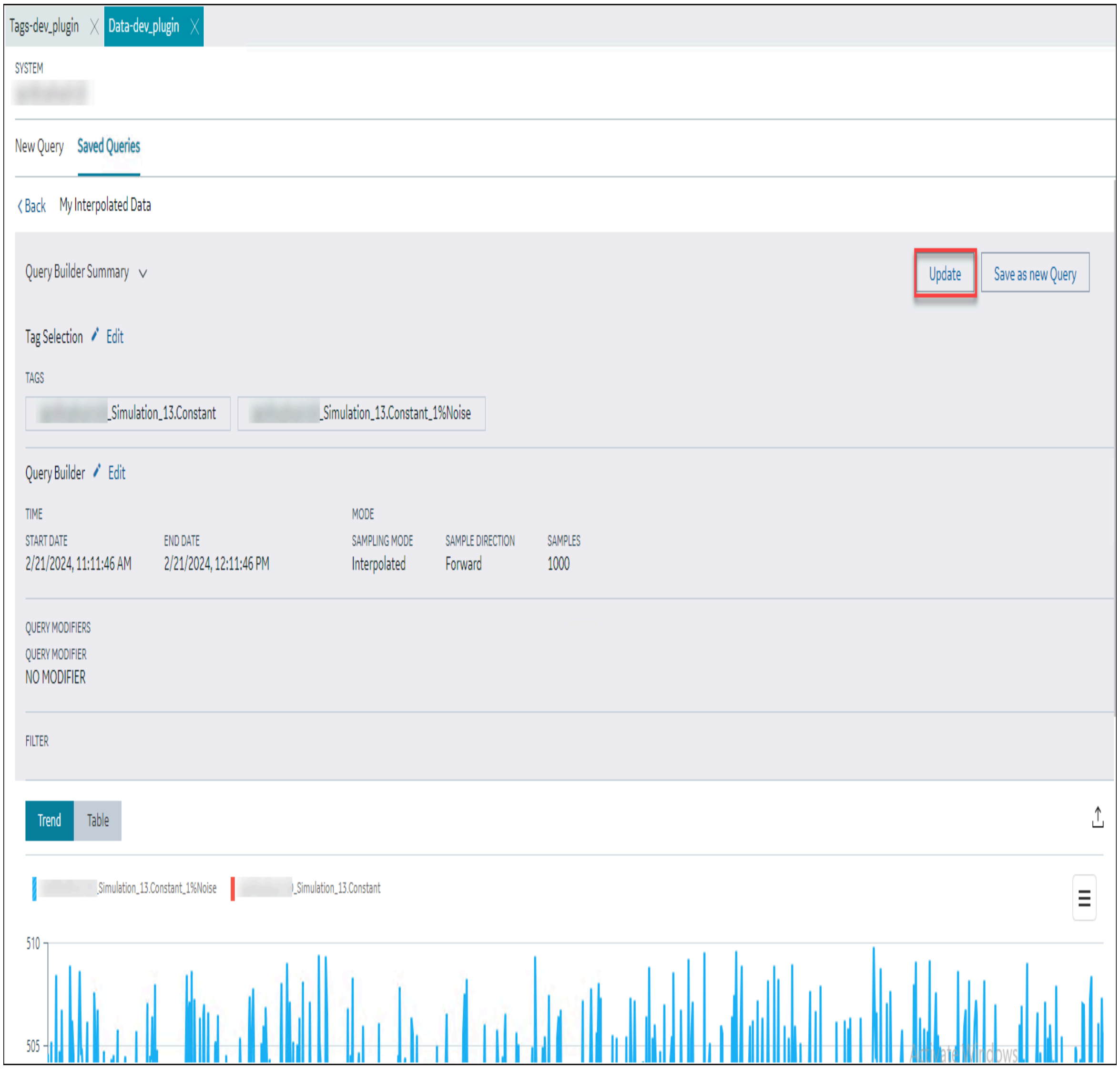
6. Select the **Edit** beside **Tag Selection** and **Query Builder** to edit tags and query criteria.

    a. Edit the tag selections or query criteria as needed.

    b. Select**Generate Report**.

    The **Query Builder Summary** appears.

7. Select **Update**.



SYSTEM
[redacted]

My Interpolated Data

Tag Selection

TAGS
[redacted]_Simulation_13.Constant
[redacted]_Simulation_13.Constant_1%Noise

Query Builder

TIME

START DATE
2/21/2024, 11:11:46 AM

END DATE
2/21/2024, 12:11:46 PM

MODE

SAMPLING MODE
Interpolated

SAMPLE DIRECTION
Forward

SAMPLES
1000

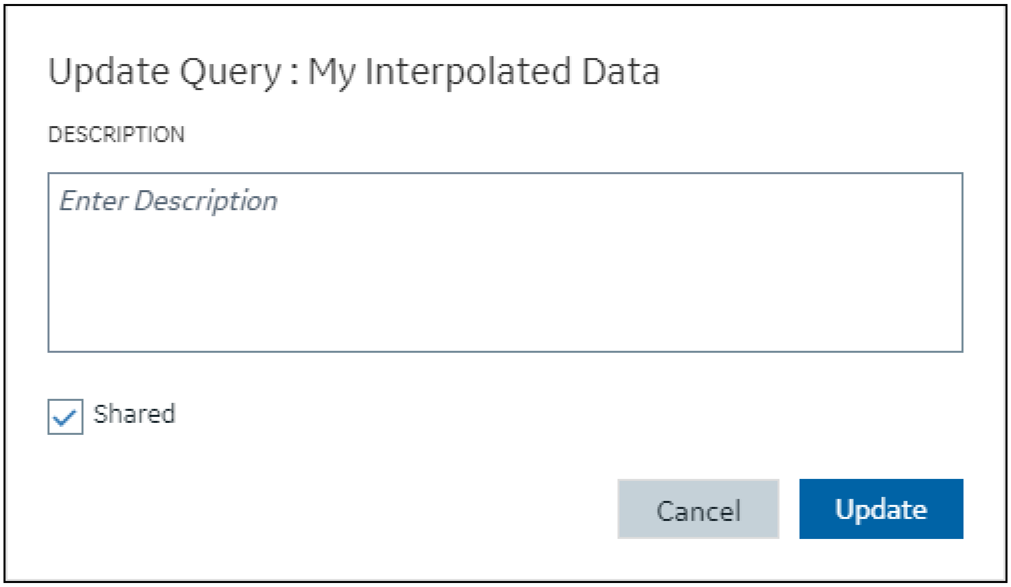
QUERY MODIFIERS

QUERY MODIFIER
NO MODIFIER

FILTER

[redacted]_Simulation_13.Constant_1%Noise
[redacted]_Simulation_13.Constant

The **Update Query:<Query Name>** window appears.



Update Query : My Interpolated Data

DESCRIPTION

Enter Description

☑ Shared

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**

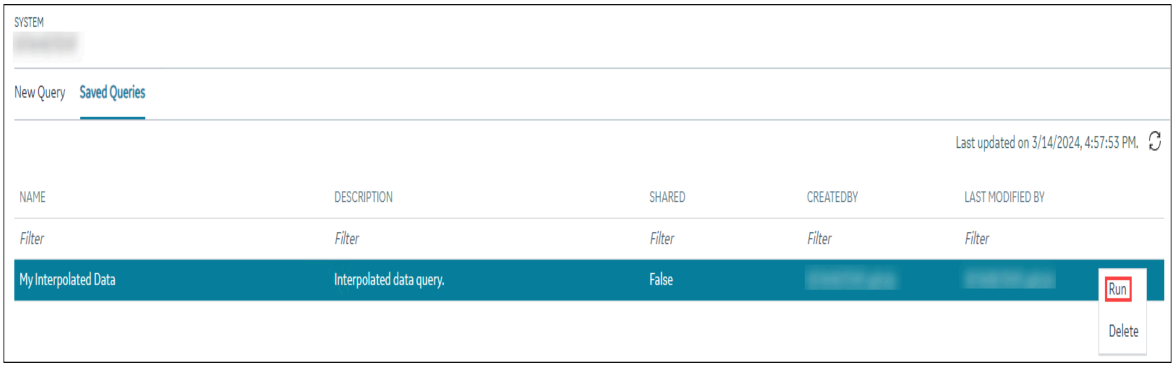
Update a Saved Query and Save it as a New Query

This topic describes how to update a saved query and save it as a new query.

1. Access Configuration Hub *(on page 97)*.
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**.



SYSTEM
[redacted]

Last updated on 3/14/2024, 4:57:53 PM.

NAME	DESCRIPTION	SHARED	CREATEDBY	LAST MODIFIED BY
Filter	Filter	Filter	Filter	Filter
My Interpolated Data	Interpolated data query.	False	[redacted]	[redacted]

The query results are plotted on a trend chart.

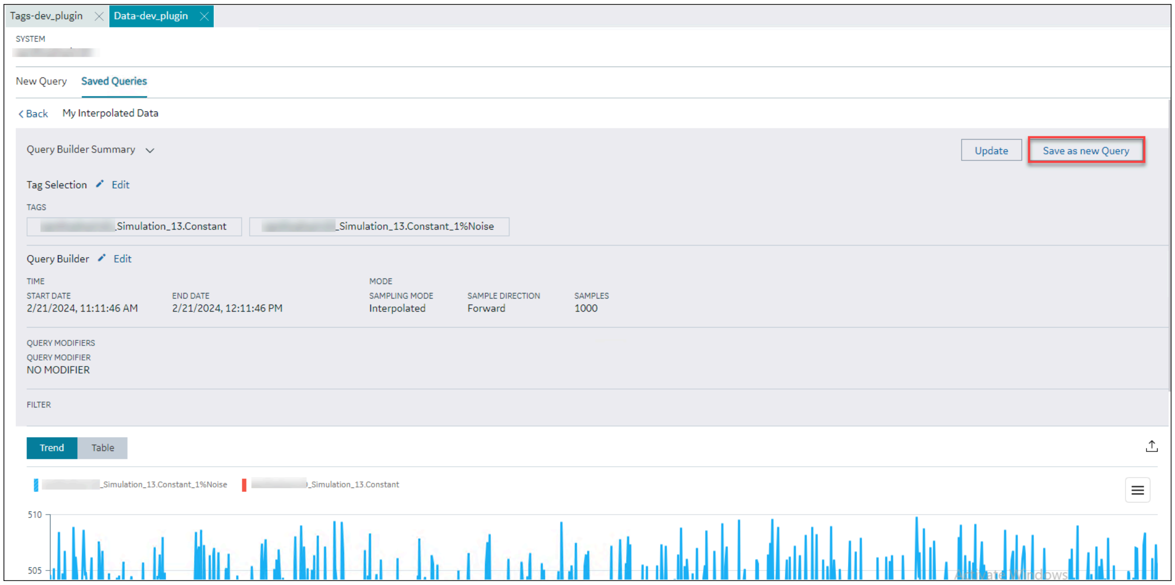
6. Select the **Edit** beside **Tag Selection** and **Query Builder** to edit tags and query criteria.

    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**.



SYSTEM
[redacted]

TAGS
[redacted]_Simulation_13.Constant
[redacted]_Simulation_13.Constant_1%Noise

TIME
START DATE 2/21/2024, 11:11:46 AM
END DATE 2/21/2024, 12:11:46 PM

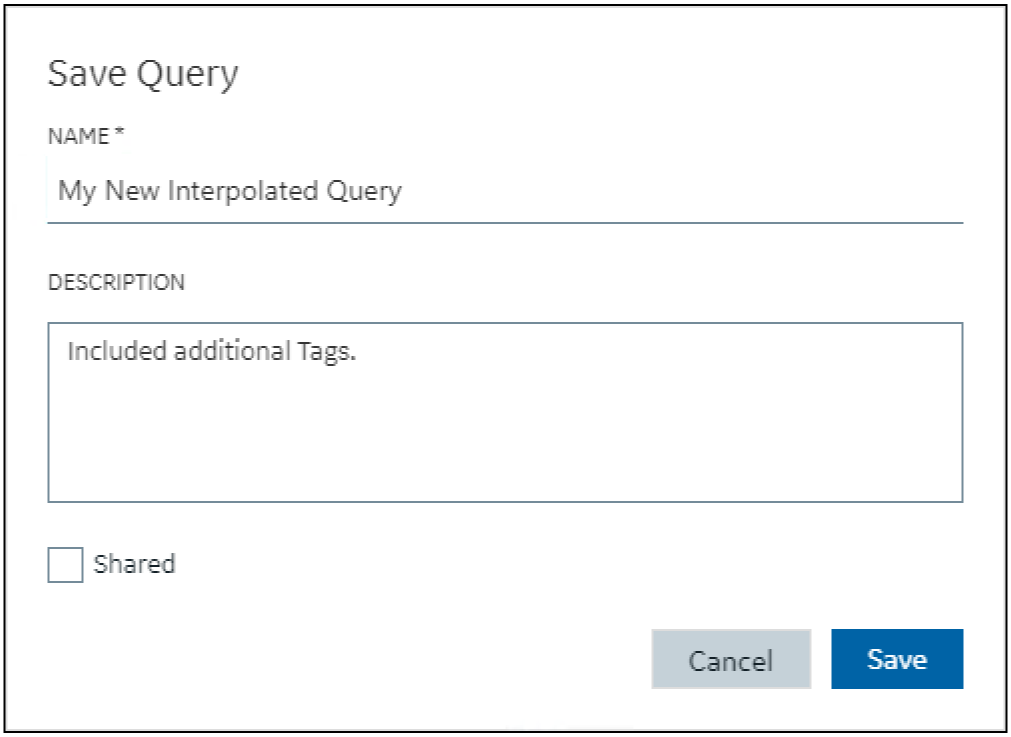
MODE
SAMPLING MODE Interpolated
SAMPLE DIRECTION Forward
SAMPLES 1000

QUERY MODIFIERS
QUERY MODIFIER NO MODIFIER

FILTER

Trend
Table
[redacted]_Simulation_13.Constant_1%Noise
[redacted]_Simulation_13.Constant
510
505

The **Save Query** window appears.



Save Query

NAME *

My New Interpolated Query

DESCRIPTION

Included additional Tags.

□ Shared

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 97)*.
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 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:

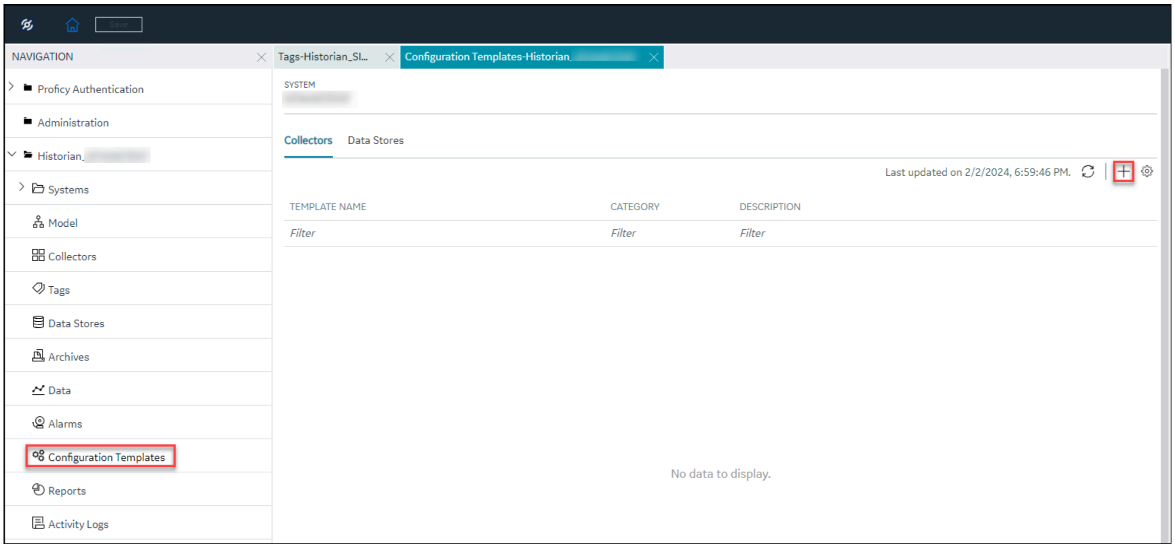
- Update some common configurations of a data store or collector.
- Save time from configuring common parameters.

Create a Configuration Template for Collectors

This topic describes how to create a configuration template for collectors.

1. Access Configuration Hub *(on page 97)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select**Configuration Templates**.

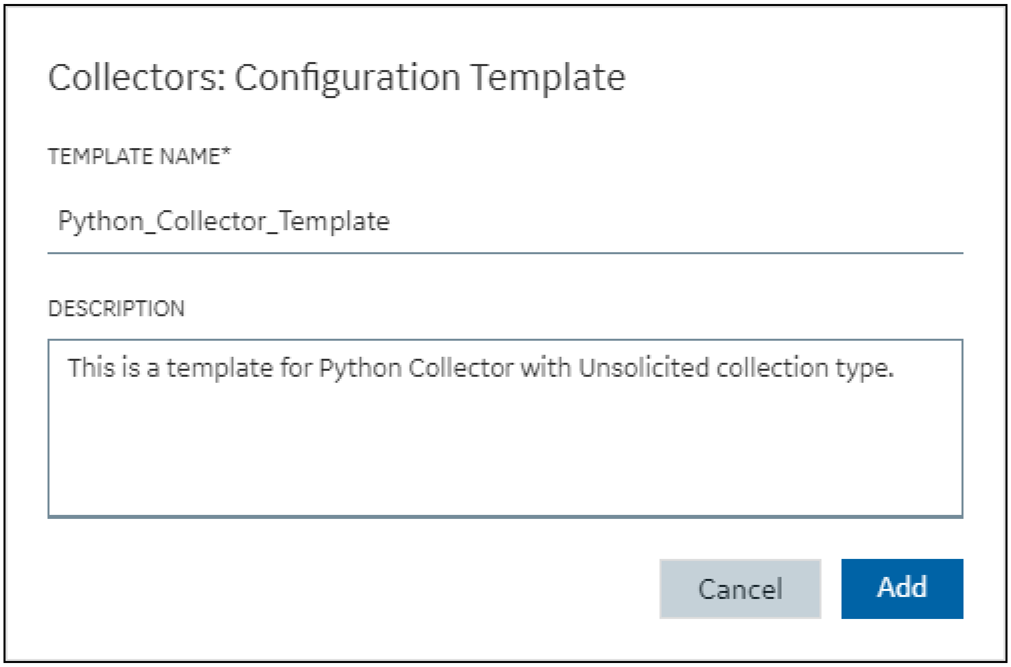
   The **Collectors** tab appears, providing you to option to create template.



Last updated on 2/2/2024, 6:59:46 PM.

3. Select +.

   The **Collectors:Configuration Template** window appears.



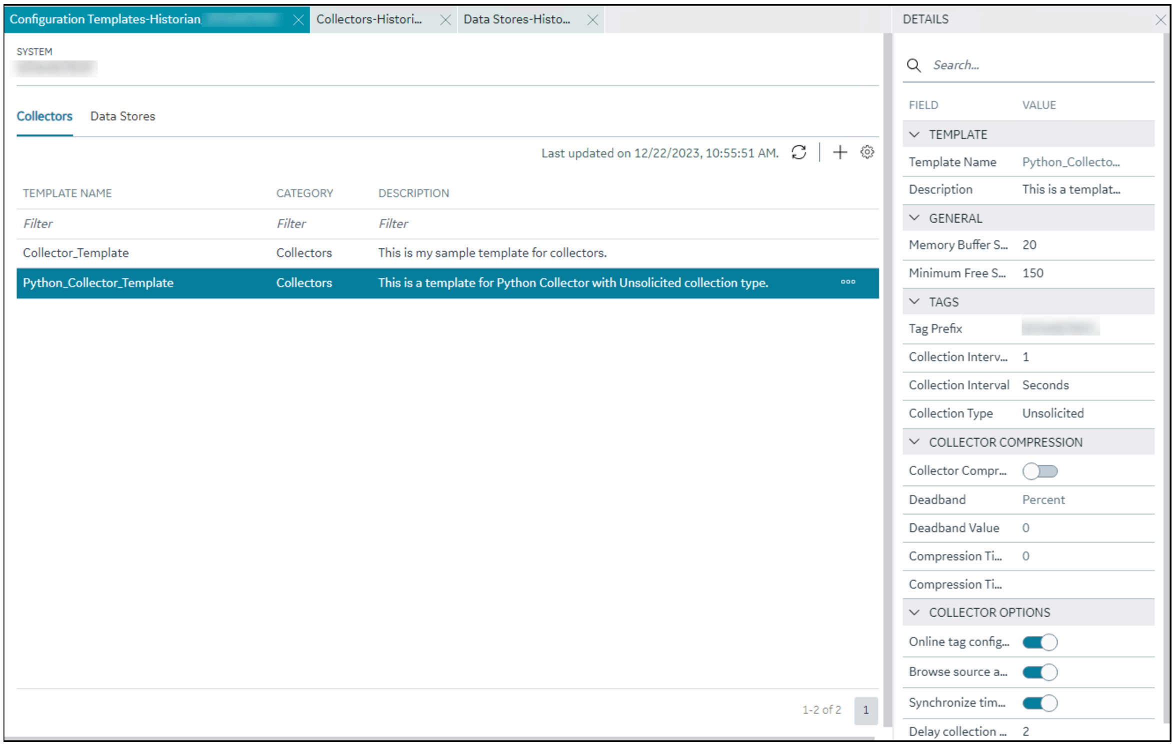
Collectors: Configuration Template

TEMPLATE NAME*: Python_Collector_Template

DESCRIPTION: This is a template for Python Collector with Unsolicited collection type.

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.



Configuration Templates-Historian [redacted] | Collectors-Histori... | Data Stores-Histo...

SYSTEM
[redacted]

Collectors | Data Stores

Last updated on 12/22/2023, 10:55:51 AM.

TEMPLATE NAME	CATEGORY	DESCRIPTION
Filter	*Filter*	*Filter*
Collector_Template	Collectors	This is my sample template for collectors.
Python_Collector_Template	Collectors	This is a template for Python Collector with Unsolicited collection type.

1-2 of 2 | 1

DETAILS

Search...

FIELD	VALUE
TEMPLATE	
Template Name	Python_Collecto...
Description	This is a templat...
GENERAL	
Memory Buffer S...	20
Minimum Free S...	150
TAGS	
Tag Prefix	[redacted]
Collection Interv...	1
Collection Interval	Seconds
Collection Type	Unsolicited
COLLECTOR COMPRESSION	
Collector Compr...	○
Deadband	Percent
Deadband Value	0
Compression Ti...	0
Compression Ti...	
COLLECTOR OPTIONS	
Online tag config...	●
Browse source a...	●
Synchronize tim...	●
Delay collection ...	2

7. Select the row that contains the created template.

   The configuration details appear in the **DETAILS** section.

Table 31. 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 32. 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 outages. To estimate the size you need for this buffer, you need to know how fast the collec-

Field	Description
	tor 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. 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 33. 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. 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. ◦ For polled data collection, this value represents the time required to complete a poll of tags in the collector. ◦ 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. You can set this value for each tag as well.

Field	Description
	Important: 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: ◦ **Polled:** Data is collected based on a scheduled time interval. This type of data collection is supported only for: ▪ The Calculation collector ▪ The HAB collector ▪ The iFIX collector ▪ The OPC Classic DA collector ▪ The OPC UA DA collector ▪ The Python collector ▪ The Simulation collector ▪ The Windows Performance collector ◦ **Unsolicited:** Data is collected based on an event. This type of data collection is supported only for: ▪ The Calculation collector ▪ The HAB collector ▪ The MQTT collector ▪ The MQTT Sparkplug B collector ▪ The ODBC collector ▪ The OPC Classic DA collector ▪ The OPC Classic HDA collector ▪ The OPC UA DA collector ▪ The OSI PI collector ▪ The OSI PI distributor

Field	Description
	- The Python collector - The Server-to-Server collector - The Server-to-Server distributor - The Wonderware Collector

Table 34. The Collector Compression Section

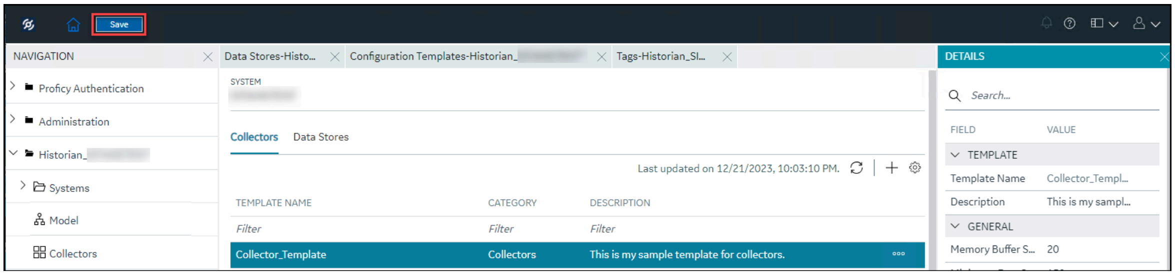
Field	Description
Collector Compression	Indicates whether you want to apply collector compression, which is a smoothing filter to data retrieved from the data source. By ignoring small changes in values that fall within a deadband centered around the last reported value, only significant changes are stored in Historian, thus consuming less archive storage space. For more information, refer to About Collector and Archive Compression *(on page 742)*.
Deadband	Indicates whether you want to apply a deadband based on the percentage of values or on absolute values. 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. **Note:** 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. 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. For more information, refer to About Collector and Archive Compression *(on page 742)*.
Compression Timeout	The time for one poll cycle for which collector compression is not used, thus sending all the samples to Historian. 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. For more information, refer to About Collector and Archive Compression *(on page 742)*.
Compression Timeout Interval	The units of measure for compression timeout.

Table 35. 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. **Note:** ◦ 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). ◦ 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.



Last updated on 12/21/2023, 10:03:10 PM.

TEMPLATE NAME	CATEGORY	DESCRIPTION
Filter	Filter	Filter
Collector_Template	Collectors	This is my sample template for collectors.

FIELD	VALUE
TEMPLATE	
Template Name	Collector_Templ...
Description	This is my sampl...
GENERAL	
Memory Buffer S...	20

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 737)*.

Apply Configuration Template to a Collector

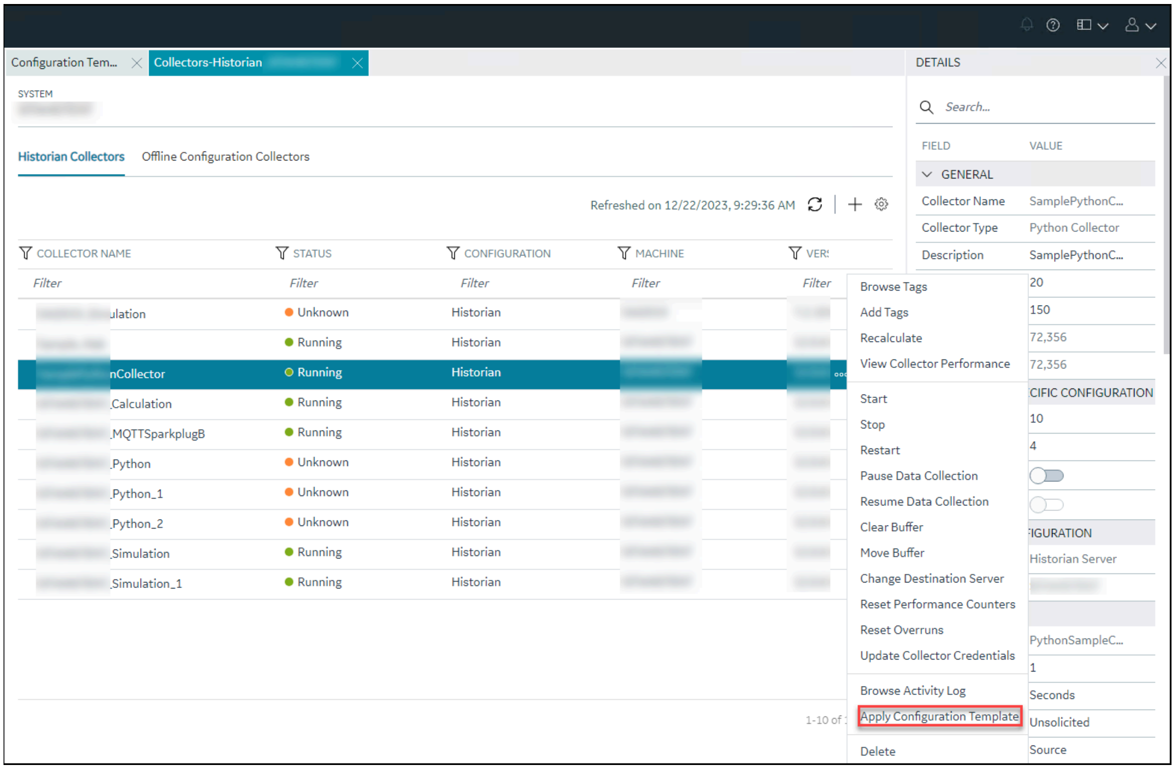
You can apply the created template to collectors as needed. You will be prompted to confirm whether you want to overwrite few of the configuration values with the values in the template.

- Ensure that you have a collector instance added *(on page 556)*.
- Ensure that you created a configuration template for collectors *(on page 813)*.

This topic describes how to apply a configuration template to a collector.

1. Access Configuration Hub *(on page 97)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.

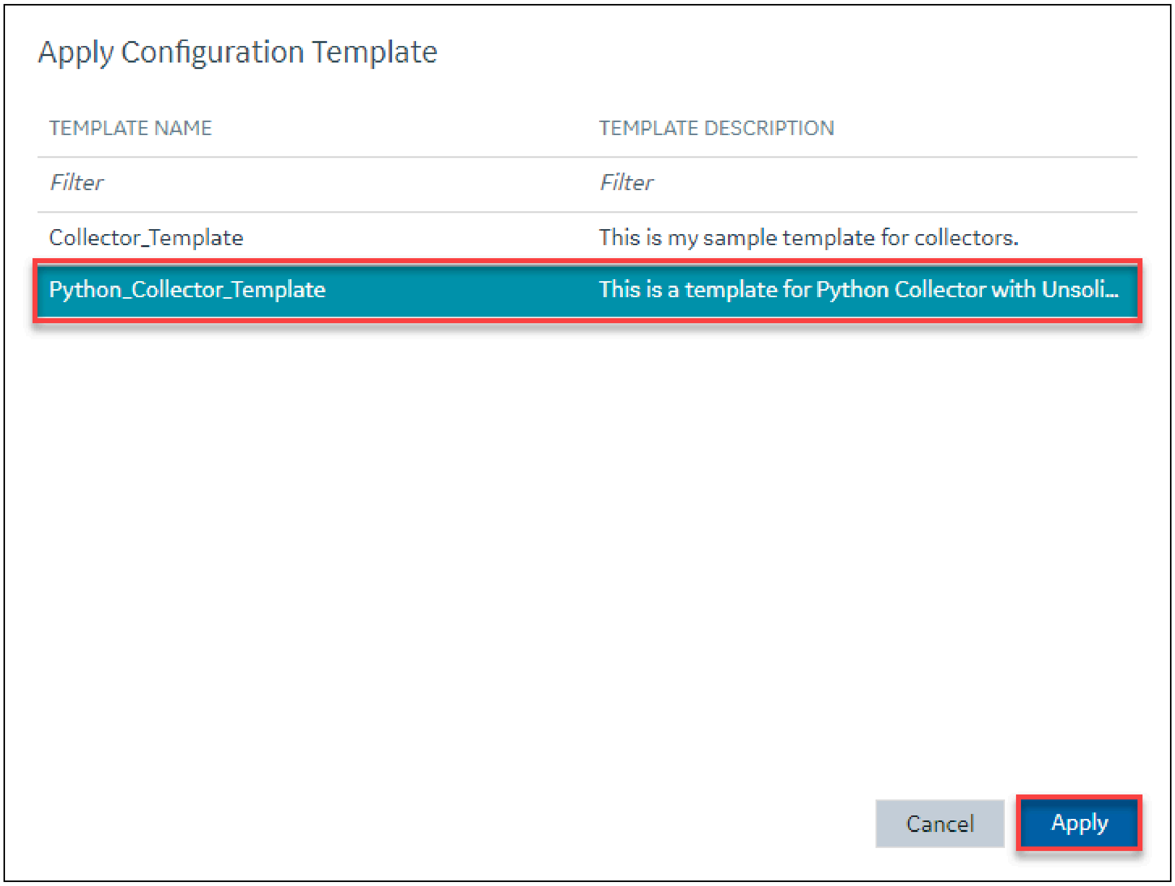
   A list of collectors appears.
3. Right-click the collector (or select ⚬⚬⚬), and then select **Apply Configuration Template**.



COLLECTOR NAME	STATUS	CONFIGURATION	MACHINE	VER:
Filter	Filter	Filter	Filter	Filter
ulation	Unknown	Historian		
	Running	Historian		
nCollector	Running	Historian		
_Calculation	Running	Historian		
_MQTTSparkplugB	Running	Historian		
_Python	Unknown	Historian		
_Python_1	Unknown	Historian		
_Python_2	Unknown	Historian		
_Simulation	Running	Historian		
_Simulation_1	Running	Historian		

FIELD	VALUE
Collector Name	SamplePythonC...
Collector Type	Python Collector
Description	SamplePythonC...
	20
	150
	72,356
	72,356
	10
	4
	Historian Server
	PythonSampleC...
	1
	Seconds
	Unsolicited
	Source

The **Apply Configuration Template** window appears, listing the available templates.



Apply Configuration Template

TEMPLATE NAME	TEMPLATE DESCRIPTION
Filter	*Filter*
Collector_Template	This is my sample template for collectors.
Python_Collector_Template	This is a template for Python Collector with Unsoli...

4. Select **Apply**.

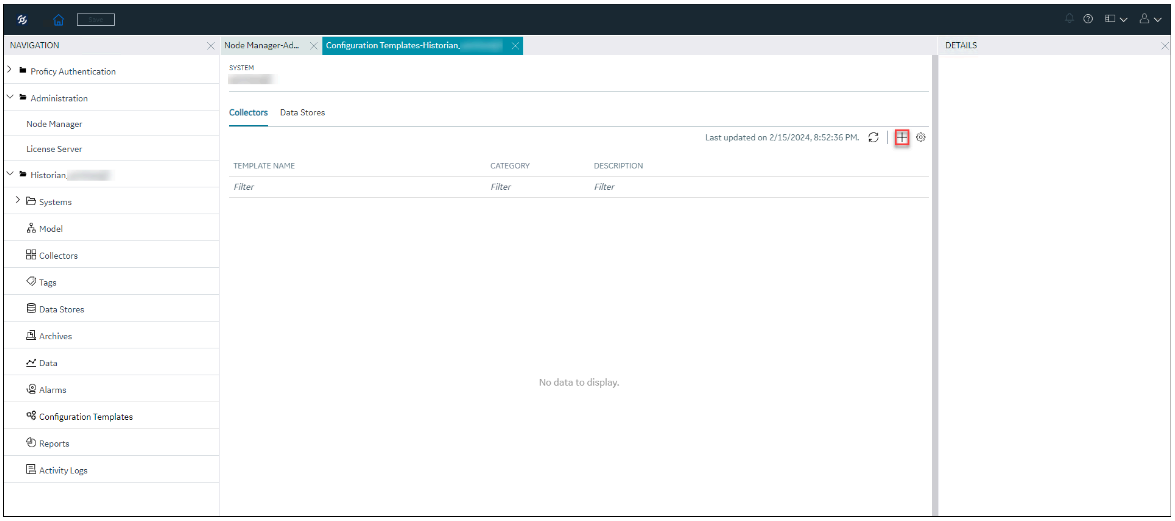
   A confirmation window appears, prompting you to confirm whether you want to overwrite few of the configuration values with the values in the template.
5. Select **Ok**.
6. In the upper-left corner, select **Save**.

   The configurations in the template are applied to the collector.

Create a Configuration Template for Data Stores

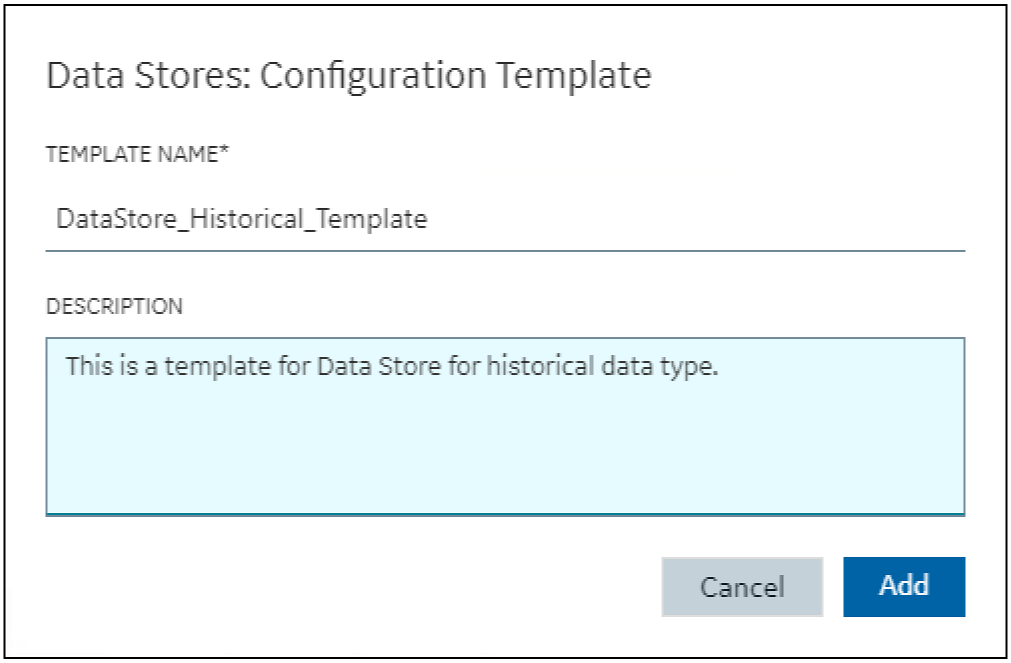
This topic describes how to create a configuration template for data stores.

1. Access Configuration Hub *(on page 97)*.
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.



Data Stores: Configuration Template

TEMPLATE NAME*

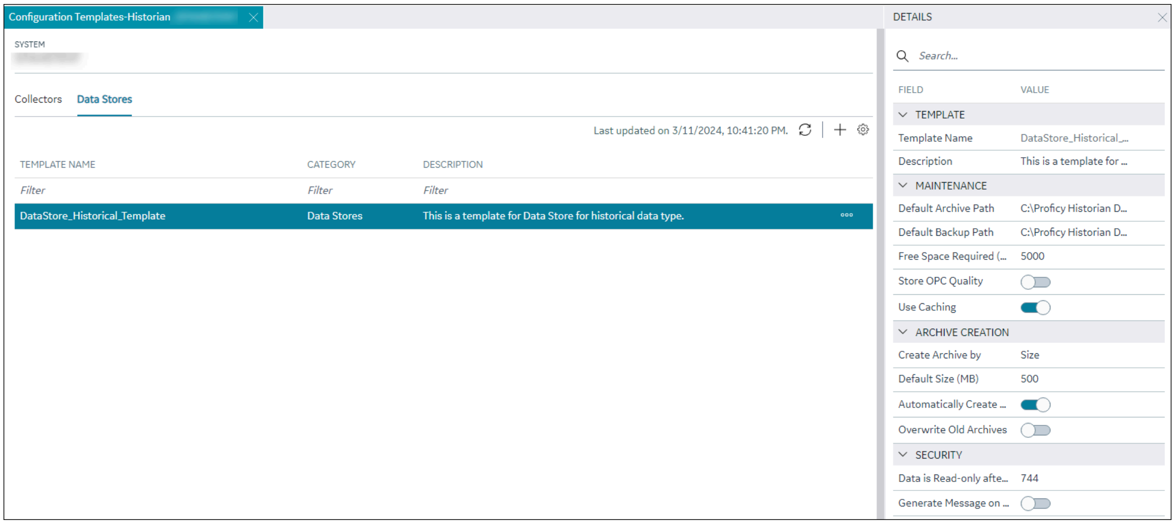
DataStore_Historical_Template

DESCRIPTION

This is a template for Data Store for historical data type.

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.



Last updated on 3/11/2024, 10:41:20 PM.

TEMPLATE NAME	CATEGORY	DESCRIPTION
Filter	*Filter*	*Filter*
DataStore_Historical_Template	Data Stores	This is a template for Data Store for historical data type.

DETAILS

Search...

FIELD	VALUE
TEMPLATE	
Template Name	DataStore_Historical_...
Description	This is a template for ...
MAINTENANCE	
Default Archive Path	C:\Proficy Historian D...
Default Backup Path	C:\Proficy Historian D...
Free Space Required (...	5000
Store OPC Quality	○
Use Caching	●
ARCHIVE CREATION	
Create Archive by	Size
Default Size (MB)	500
Automatically Create ...	●
Overwrite Old Archives	○
SECURITY	
Data is Read-only afte...	744
Generate Message on ...	○

8. Select the row that contains the created template.

   The data store details appear in the **DETAILS** section.

Table 36. 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 37. 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. 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.

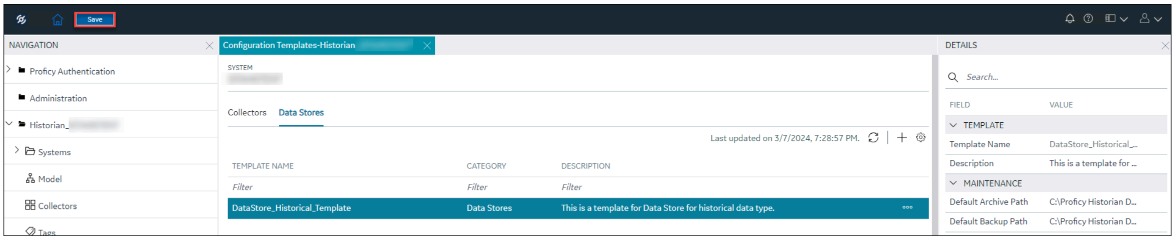
Table 38. Archive Creation

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. Select one of the following options: ◦ **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**). ◦ **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 791)* 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. 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 39. 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.



SYSTEM [redacted]

Last updated on 3/7/2024, 7:28:57 PM.

TEMPLATE NAME	CATEGORY	DESCRIPTION
Filter	Filter	Filter
DataStore_Historical_Template	Data Stores	This is a template for Data Store for historical data type.

FIELD	VALUE
TEMPLATE	
Template Name	DataStore_Historical_...
Description	This is a template for ...
MAINTENANCE	
Default Archive Path	C:\Proficy Historian D...
Default Backup Path	C:\Proficy Historian D...

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 585)*.

Apply the Configuration Template to a Data Store

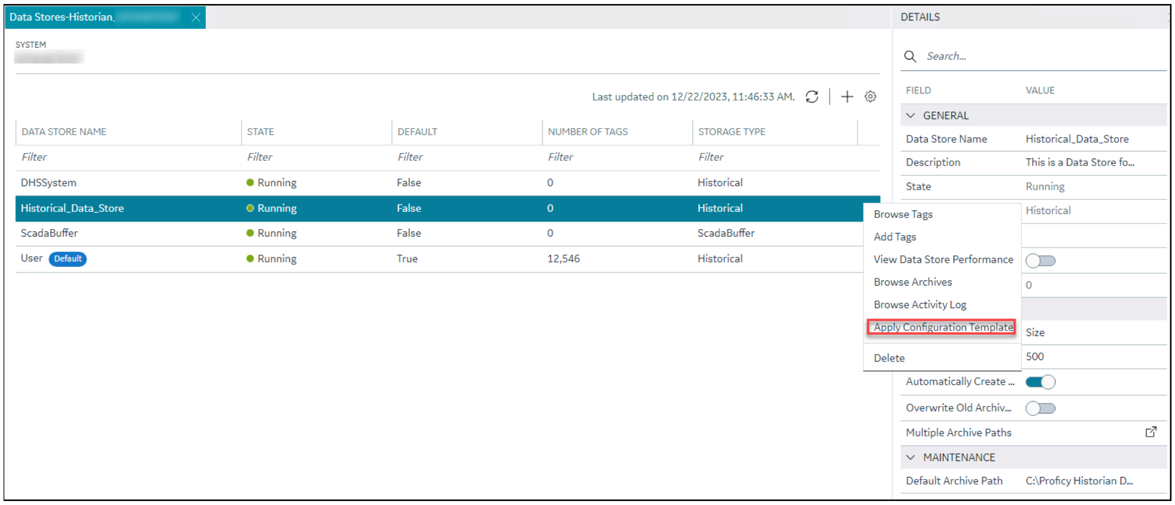
You can apply the created template to user-created data store as needed. You will be prompted to confirm whether you want to overwrite few of the configuration values with the values in the template.

- Ensure that you have a data store created *(on page 575)*.
- Ensure that you have a configuration template for data stores *(on page 823)*.

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 97)*.
2. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Data Stores**.

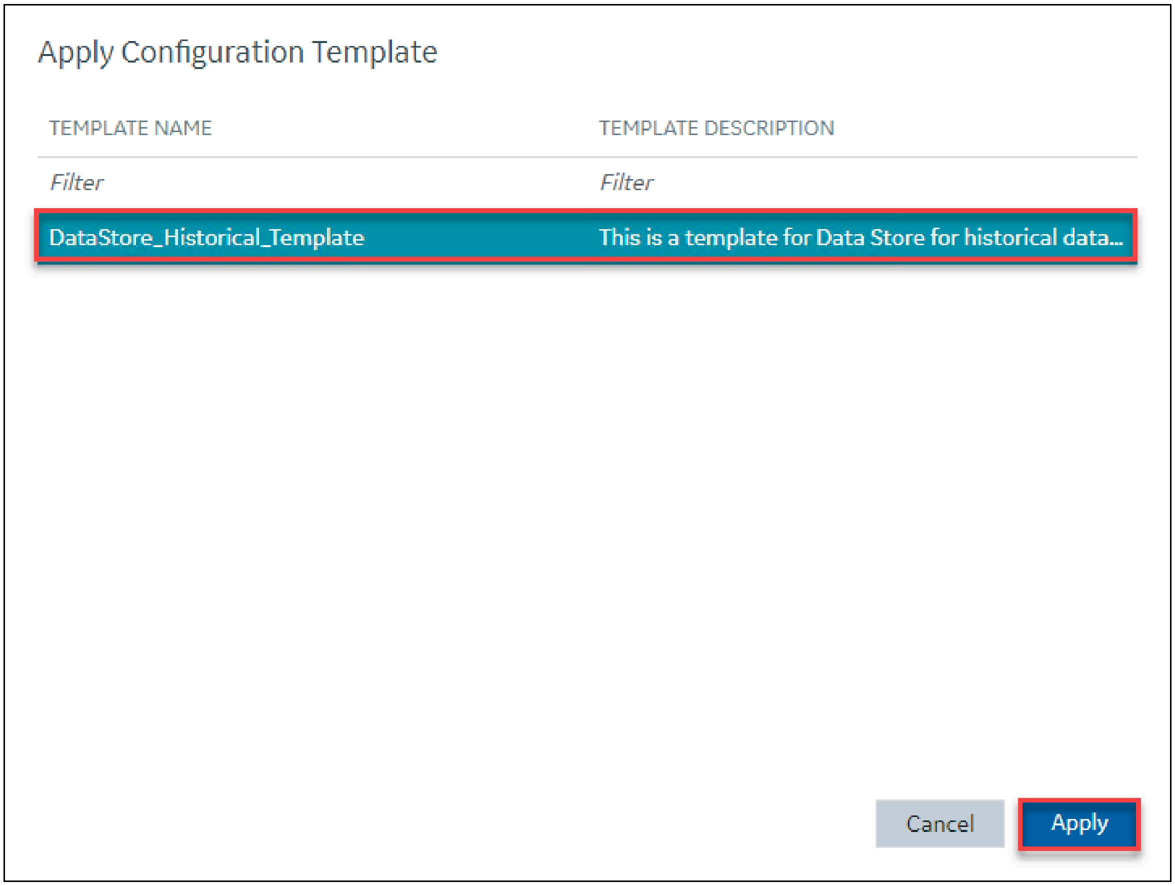
   The **Data Stores** section appears.
3. Right-click the data store (or select ⚬⚬⚬), and then select **Apply Configuration Template**.



DATA STORE NAME	STATE	DEFAULT	NUMBER OF TAGS	STORAGE TYPE
Filter	*Filter*	*Filter*	*Filter*	*Filter*
DHSSystem	● Running	False	0	Historical
Historical_Data_Store	● Running	False	0	Historical
ScadaBuffer	● Running	False	0	ScadaBuffer
User Default	● Running	True	12,546	Historical

FIELD	VALUE
GENERAL	
Data Store Name	Historical_Data_Store
Description	This is a Data Store fo...
State	Running
	Historical
	○
	0
	Size
	500
Automatically Create ...	●
Overwrite Old Archiv...	○
Multiple Archive Paths	
MAINTENANCE	
Default Archive Path	C:\Proficy Historian D...

The **Apply Configuration Template** window appears, listing the available templates.



Apply Configuration Template

TEMPLATE NAME	TEMPLATE DESCRIPTION
Filter	*Filter*
DataStore_Historical_Template	This is a template for Data Store for historical data...

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.

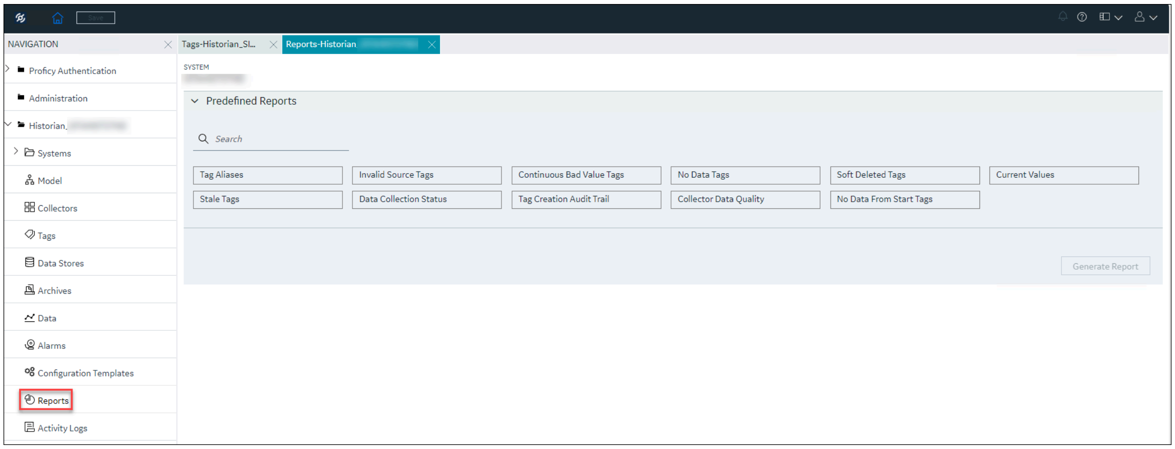
For more information on generating reports, refer to generate reports *(on page 830)*.

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 97)*.
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. **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 ◦ 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. ◦ TIME INTERVAL- The duration based on which the report must be generated. The available options are, ▪ 1 hour ▪ 8 hours ▪ 1 day ▪ 1 week ▪ 1 month	◦ TAG NAME ◦ 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. ◦ TIME INTERVAL-The duration based on which the report must be generated. The available options are, ▪ 1 hour ▪ 8 hours ▪ 1 day ▪ 1 week ▪ 1 month ▪ Custom-Selecting this option enables a cus-	◦ TAG NAME ◦ DESCRIPTION ◦ VALUE ◦ TIME STAMP ◦ 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. ◦ TIME INTERVAL- The duration based on which the report must be generated. The available options are, ▪ 1 hour ▪ 8 hours ▪ 1 day ▪ 1 week ▪ 1 month ▪ Custom- Selecting this option enables a custom start and end date/time selector. You can select or	◦ TAG NAME ◦ DESCRIPTION ◦ DELETION TIME ◦ 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. ◦ 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 ◦ ENGINEERING UNITS (DESCRIPTION) ◦ QUALITY ◦ VALUE ◦ 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 ◦ 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. ◦ DATA COLLECTION- The status of da-	◦ TAG NAME ◦ DESCRIPTION ◦ 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, ▪ On ▪ 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. ◦ TIME INTERVAL- The duration based on which the report must be generated. The available options are, ▪ 1 hour ▪ 8 hours ▪ 1 day ▪ 1 week ▪ 1 month ▪ 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 ◦ DESCRIPTION ◦ CREATION TIME ◦ CREATED BY

Report	Description	Input Options	Data Available in the Report
		on which the report must be generated. ◦ CREATION TIME- The time of creation based on which the report must be generated. ◦ 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 ◦ DESCRIPTION ◦ TOTAL TAG COUNT ◦ BAD TAGS COUNT ◦ BAD TAGS PERCENT ◦ GOOD TAGS COUNT ◦ 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 ◦ 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 837)*, or save the generated report as a PDF file *(on page 837)*.

Export the Generated Report as a CSV File

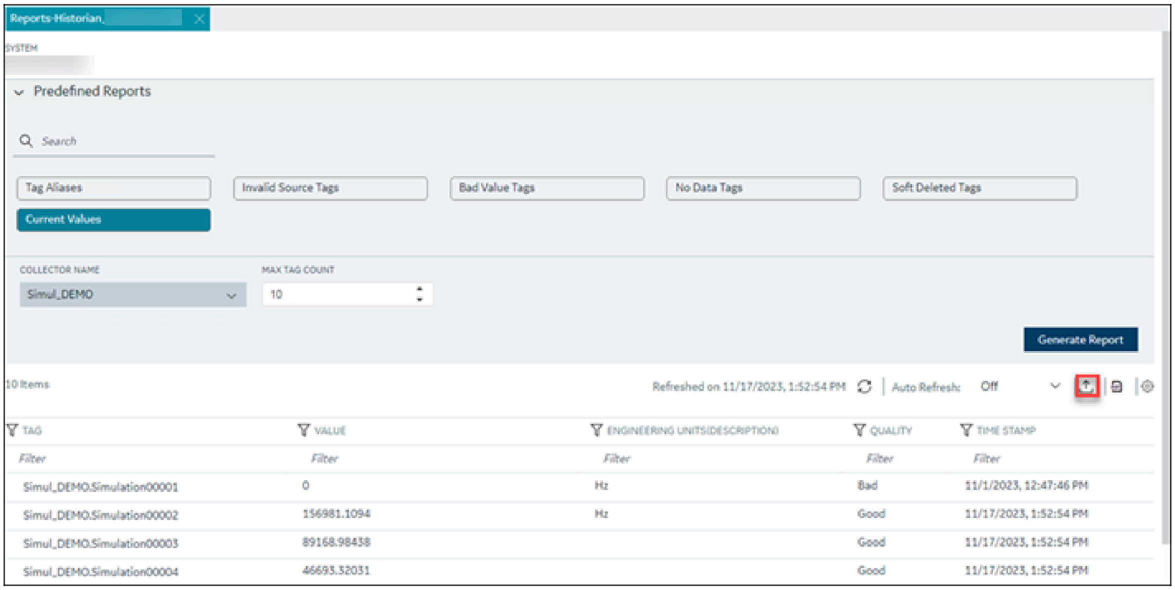
This topic describes how to export and save a report as CSV file.

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 830)* 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.



COLLECTOR NAME: Simul_DEMO

MAX TAG COUNT: 10

10 Items

Refreshed on 11/17/2023, 1:52:54 PM

Auto Refresh: Off

TAG	VALUE	ENGINEERING UNITS(DESCRIPTION)	QUALITY	TIME STAMP
Filter	Filter	Filter	Filter	Filter
Simul_DEMO.Simulation00001	0	Hz	Bad	11/1/2023, 12:47:46 PM
Simul_DEMO.Simulation00002	156981.1094	Hz	Good	11/17/2023, 1:52:54 PM
Simul_DEMO.Simulation00003	89168.98438		Good	11/17/2023, 1:52:54 PM
Simul_DEMO.Simulation00004	46693.32031		Good	11/17/2023, 1:52:54 PM

The file will be saved in the <System Name>_Reports_Export_Details_with_<ReportName> format. For example, **SYSTEMADMIN_Reports_Export_Details_with_CurrentValues**.

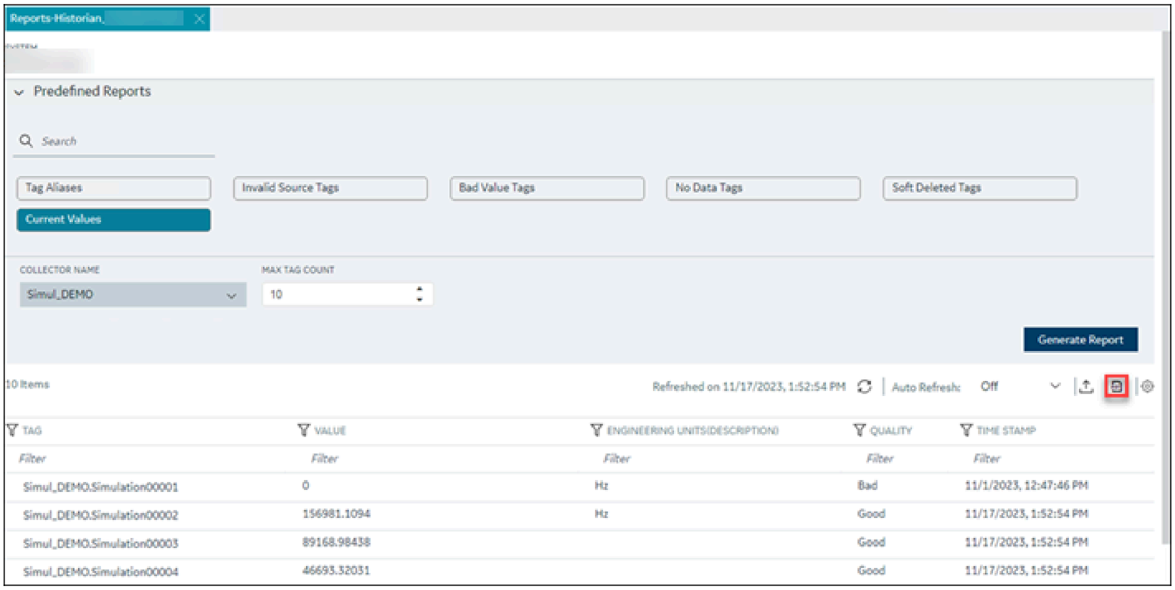
Save the Generated Report as a PDF File

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 830)* 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.



COLLECTOR NAME: Simul_DEMO
MAX TAG COUNT: 10

10 Items
Refreshed on 11/17/2023, 1:52:54 PM

TAG	VALUE	ENGINEERING UNITS(DESCRIPTION)	QUALITY	TIME STAMP
Filter	Filter	Filter	Filter	Filter
Simul_DEMO.Simulation00001	0	Hz	Bad	11/1/2023, 12:47:46 PM
Simul_DEMO.Simulation00002	156981.1094	Hz	Good	11/17/2023, 1:52:54 PM
Simul_DEMO.Simulation00003	89168.98438		Good	11/17/2023, 1:52:54 PM
Simul_DEMO.Simulation00004	46693.32031		Good	11/17/2023, 1:52:54 PM

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 97)*.
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:
   1. In the **NAVIGATION** section, under the Configuration Hub plugin for Historian, select **Collectors**.
      A list of collectors in the system appears.
   2. 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:
   1. In the **NAVIGATION** section, select **Historian_<system_name> > Tags**.
      A list of tags in the system appears.
   2. 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:
   1. In the upper-right corner of the main section, select ▽.
   2. 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: ▪ **Configuration**: Includes modifying a collector instance or a tag configuration. ▪ **Connections**: Includes system connections and creating a collector interface.

Field	Description
	▪ **General**: Includes pausing or resuming data collection. ▪ **Performance**: Includes creating or closing an archive and moving buffer files. ▪ **ServiceControl**: Includes starting or stopping a collector interface. 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)* and Troubleshooting the Historian Server *(on page)*.

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)* on the machine on which you have installed Web-based Clients, install the Proficy Authentication certificate *(on page)*, 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.

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 10. Using the REST APIs

Overview of the Historian REST APIs

The REST APIs are used to fetch data from the Historian database. You can fetch data such as the latest data point of a tag or data points for a duration. Using these APIs, you can also work on aggregation techniques like average, minimum, and maximum values.

The REST APIs will be available automatically when you deploy Proficy Historian for Cloud *(on page 44)*. No additional steps are required.

Get an Authorization Token

1. Deploy Proficy Historian for AWS *(on page 43)*.
2. Install an API platform (such as Postman).

To connect the Historian server with the REST APIs, you must get an authorization token.

1. Access an API platform, and request for an authorization token.
2. Enter values as described in the following table, and submit.

> **Note:**
> The names of these fields may vary depending on the API platform that you are using.

Field	Description
Token name	Provide a name for the token.
Grant type	Select Client Credentials.
Access token URL	Enter a value in the following format: `https://<NLB DNS>:9090/oauth/token` **Tip:** To find the NLB DNS: a. Access the EKS cluster on which you have deployed Proficy Historian for AWS. b. Access the EC2 instance. c. In the navigation pane, under **Load Balancing**, select **Load Balancers**.

Field	Description
	d. Select the load balancer for which you want to find the DNS. e. In the **Description** section, copy the DNS name.
Client ID	Enter `historian_rest_api`.
Client secret	Enter the value you entered in the **Password** field under **Proficy Authentication Configuration** when you created the stack.

The token is generated. You can use this token to use REST APIs.

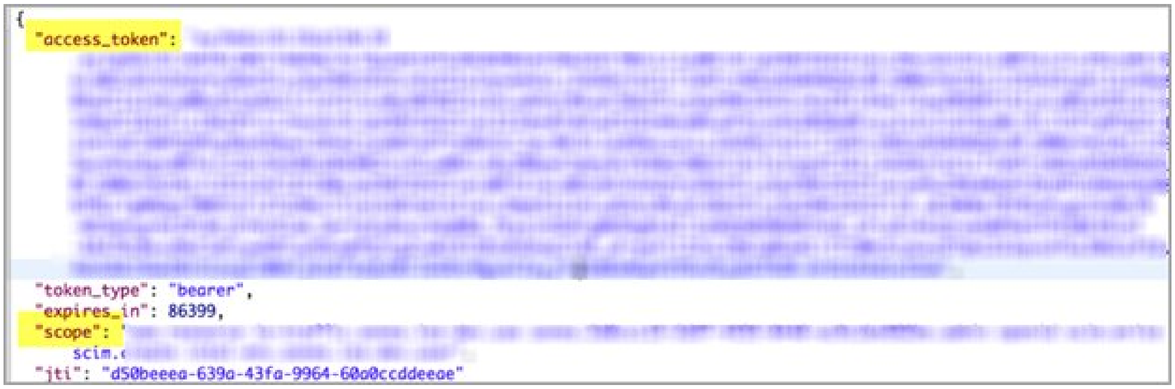
3. Use the following command to authenticate the REST API authentication with password credentials:

```
curl -d
 "client_id=<value>&client_secret=<value>&grant_type=password&username=<value>&password=<value>&token_format=op
aque&response_type=token" https://nlb-dns:9090/oauth/token
```

For example:

```
curl -d
 "client_id=server1.admin&client_secret=adminsecret&grant_type=password&username=user1&password=pwd123&token_fo
rmat=opaque&response_type=token" https://nlb-dns:9090/oauth/token
```

The following image shows an example of OAuth access. The actual token text is blurred for security concerns.



```
{
  "access_token": [redacted]
  "token_type": "bearer",
  "expires_in": 86399,
  "scope": [redacted]
      scim.[redacted]
  "jti": "d50beeea-639a-43fa-9964-60a0ccddeeae"
```

Client applications can access data using service the REST API endpoints. Your application makes an HTTP request and parses the response. You can use any web-development language to access the APIs.

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 846)*
- Start and End timestamps *(on page 846)*
- TagSamples *(on page 847)*
- DataSamples *(on page 849)*
- SamplingModeType *(on page 850)*
- Direction *(on page 852)*
- CalculationModeType *(on page 852)*
- FilterModeType *(on page 858)*
- ReturnDataFields *(on page 858)*
- Payload *(on page 859)*
- Error Code Definitions *(on page 865)*

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://<NLB DNS:9090>:9090/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. 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. 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:** • **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. • **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. • **Byte** (one byte) – Stores integer values. Valid values for the byte data type are -128 to +127. • **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 • **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.

Property Name	Property Type	Description
		• **SingleInteger** (two bytes) – Stores whole numbers, without decimal places. Valid values for the single integer data type are -32767 to +32767. • **DoubleInteger** (four bytes) – Stores whole numbers, without decimal places. Valid values for the double integer data type are -2147483648 to +2147483648. • **FixedString** (Configured by user) – Stores string data of a fixed size. Valid values are between 0 and 255 bytes. • **Float** – Single float. • **Integer** – Single integer. • **MultiField** – Stores string data that has multiple words. • **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). • **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. • **Time** – Returns or sets the type of time stamping applied to data at collection time. • **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). • **Undefined** – Data type is not defined. • **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). • **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.

Property Name	Property Type	Description
		• **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. • **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 See Error Code Definition *(on page 865)* for more information.
Samples	Data Sample	See DataSample Parameter *(on page 849)* 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). 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 (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".

Property Name	Property Type	Description
		Value and Status • **0** – Bad • **1** – Uncertain • **2** – NA • **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. 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.	3
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 End-	5

Properties	Description	Value
	Time criteria is ignored for this sampling mode.	
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
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: • 1 – true • 0 – false 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
StandardDeviation	Retrieves the time-weighted standard deviation for each calculation interval.	2
Total	Retrieves the time-weighted rate total for each calculation interval. 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

Calculation Mode Type	Description	Value
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. 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. 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. 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.	6
RawAverage	Retrieves the arithmetic average of all good quality raw samples for each calculation interval. 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. 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	7

Calculation Mode Type	Description	Value
	percentage of good is always 100, even if the interval contains bad quality samples. RawAverage is useful for calculating an accurate average when a sufficient number of raw samples are collected.	
RawStandardDeviation	Retrieves the arithmetic standard deviation of raw values for each calculation interval. For Value, any raw point of bad data quality is ignored. 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. RawStandardDeviation is useful for calculating an accurate standard deviation when a sufficient number of raw samples are collected.	8
RawTotal	Retrieves the arithmetic total (sum) of sampled values for each interval. Value is the sum of the good quality values of all raw samples in the interval. All bad quality samples are ignored. 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. 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.	9

Calculation Mode Type	Description	Value
	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.	
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
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. 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.	16

Calculation Mode Type	Description	Value
	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.	
FirstRawValue	Retrieves the first good raw sample value for a given interval. Value is the value of the raw sample, or zero if there are no good raw samples in the interval. 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. The Raw sample has a quality of Good, Bad, or Uncertain, and that is converted to a 0 or 100 percent.	17
FirstRawTime	Retrieves the first good raw timestamp for a given interval. Value is the timestamp of the sample or the year 1969 if there are no good raw samples in the interval. 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.	18

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.	
LastRawValue	Retrieves the last good raw sample value for a given time interval. Value is the value of the raw sample or zero if there are no good raw samples in the interval. 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. The Raw sample has a quality of Good, Bad, or Uncertain, and that is converted to a 0 or 100 percent.	19
LastRawTime	Retrieves the last good timestamp of the last value for a given time interval. Value is the timestamp of the sample or the year 1969 if there are no good raw samples in the interval. 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. The Raw sample has a quality of Good, Bad, or Uncertain, and that is converted to a 0 or 100 percent.	20

Calculation Mode Type	Description	Value
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 collected sample or an interval time stamp. When specified in the query, returns the TimeStamp property.	1 (0001)
Value	The collected value or sampled value; the data type of the value will be the same data type as the tag's raw data.	2 (0010)
Quality	When specified in the query, returns the Quality property. Each sample in Current Value and Raw query retrieval has a quality of: • Good (3) • Not Available (2) • Uncertain (1) • Bad (0) 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	
Comment	String	

DataType : ihDataType	SignedIntegral	Type definition is an enumerated type "ihDataType".
		`{`
		`ihDataTypeUndefined = 0,`
		`ihScaled,`
		`ihFloat,`
		`ihDoubleFloat,`
		`ihInteger,`
		`ihDoubleInteger,`
		`ihFixedString,`
		`ihVariableString,`
		`ihBlob,`
		`ihTime,`
		`ihInt64,`
		`ihUInt64,`
		`ihUInt32,`
		`ihUInt16,`
		`ihByte,`
		`ihBool,`

Property Name	Property Type	Description

		`ihMultiField,`
		`ihArray,`
		`ihMaxDataType`
		`} ihDataType;`

FixedStringLength	UnsignedChar	
CollectorName	String	
SourceAddress	String	

CollectionType : ihCollectionType	SignedIntegral	Type definition is an enumerated type "ihCollectionType".
		`{`
		`ihUnsolicited = 1,`
		`ihPolled`
		`} ihCollectionType;`

CollectionInterval	SignedIntegral	
CollectionOffset	UnsignedLong	
LoadBalancing	Boolean	

TimeStampType : ihTimeStampType	SignedIntegral	Type definition is an enumerated type "ihTimeStampType".
		`{`
		`ihSource = 1,`
		`ihInterface,`
		`} ihTimeStampType;`

HiEngineeringUnits	Double	
LoEngineeringUnits	Double	
InputScaling	Boolean	
HiScale	Double	
LoScale	Double	

Property Name	Property Type	Description
CollectorCompression	Boolean	
CollectorDeadbandPercentRange	Float	
ArchiveCompression	Boolean	
ArchiveDeadbandPercentRange	Float	
General1	String	
General2	String	
General3	String	
General4	String	
General5	String	
ReadSecurityGroup	String	
WriteSecurityGroup	String	
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".
		`{`
		`ihInterfaceUndefined = 0,`
		`ihIFix,`
		`ihRandom,`
		`ihOPC,`
		`ihFile,`
		`ihIFixLabData,`
		`ihManualEntry,`
		`ihOther,`
		`ihCalcEngine,`
		`ihServerToServer,`
		`ihPI,`

Property Name	Property Type	Description

		`ihOPCAE,`
		`ihCIMPE,`
		`ihPIDistributor,`
		`ihCIMME,`
		`ihPerfTag,`
		`ihCustom,`
		`ihServerToServerDistributor,`
		`ihWindowsPerfMon,`
		`} ihInterfaceType;`

UTCBias	SignedIntegral	
AverageCollectionTime	Boolean	Used for Get Tag Properties API.
CalculationDependencies	StringArray	
CollectionDisabled	Boolean	
ArchiveCompressionTimeout	UnsignedLong	
CollectorCompressionTimeout	UnsignedLong	
SpikeLogic	Boolean	
SpikeLogicOverride	Boolean	
CollectorAbsoluteDeadbanding	Boolean	
CollectorAbsoluteDeadband	Double	
ArchiveAbsoluteDeadbanding	Boolean	
ArchiveAbsoluteDeadband	Double	
StepValue	Boolean	

TimeResolution : ihTimeResolution	SignedIntegral	Type definition is an enumerated type "ihTimeResolution".
		`{`
		`ihSeconds = 0,`
		`ihMilliseconds,`

Property Name	Property Type	Description

		`ihMicroseconds,`
		`ihNanoseconds`
		`} ihTimeResolution;`

ConditionCollectionEnabled	Boolean	
ConditionCollectionTriggerTag	String	

ConditionCollectionComparison : ihConditionCollectionComparison	SignedIntegral	Type definition is an enumerated type "ihConditionCollectionComparison".
		`{`
		`ihConditionComparisonUndefined = 0,`
		`ihConditionComparisonEqual,`
		`ihConditionComparisonLessThan,`
		`ihConditionComparisonLessThanEqual,`
		`ihConditionComparisonGreaterThan,`
		`ihConditionComparisonGreaterThanEqual,`
		`ihConditionComparisonNotEqual`
		`} 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	

Property Name	Property Type	Description
NumberOfElements	SignedIntegral	

DataDensity : ihTagDataDensity	SignedIntegral	Type definition is an enumerated type "ihTagDataDensity".
		`{`
		`ihDataDensityUndefined = 0,`
		`ihDataDensityMinimum = 1,`
		`ihDataDensityMedium = 4,`
		`ihDataDensityMaximum = 7`
		`} ihTagDataDensity;`

CalcType : ihTagCalcType	SignedIntegral	Type definition is an enumerated type "ihCalcType".
		`{`
		`ihRawTag = 0,`
		`ihAnalyticTag = 1,`
		`ihPythonExprTag = 2`
		`} 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 40. 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.

Table 40. Error Code Definitions (continued)

Error Code Value:	Error Code Definition
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.
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.

Table 40. Error Code Definitions (continued)

Error Code Value:	Error Code Definition
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
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.

Table 40. Error Code Definitions (continued)

Error Code Value:	Error Code Definition
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.
ihSTATUS_INTERFACE_START_FAIL = -50	Starting the collector has failed.
ihSTATUS_INTERFACE_STOP_FAIL = -51	Stopping the collector has failed.

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://<NLB DNS>:9090/historian-rest-api/v1/tags/{nameMask}/{maxNumber}`
SAMPLE URI	`https://<NLB DNS>:9090/historian-rest-api/v1/tags?name-Mask=*&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 41. Query Parameters

Parameter	Description	Required?	Values
`nameMask`	Tagmask that searches for all tags that match the mask and applies the remaining criteria to	Optional	String

Table 41. Query Parameters (continued)

Parameter	Description	Required?	Values
	retrieve data. The mask can include wildcards, such as asterisks (*).		
`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. If a negative number is used, then 0 is used for the maxNumber. 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 0 by default

Table 42. 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: • ALT_SENSOR • tagName1 • 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).

Parameter Name	Data Type	Default	Description
`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.
`lastmodified`	String	`1970-01-01T00:00:00.000Z`	`>=` 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` Must be > `1`	If invalid, no data is returned.

Parameter Name	Data Type	Default	Description
`pagesize`	Integer	`128` Max `512` 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. • `first` – First page `tags list` URL (can be null if count is 0). • `last` - Last page `tags list` URL (can be null if count is 0). • `prev` – Previous page `tags list` URL (can be null if current page is 1). • Next - Next page `tags list` URL (can be null if current page is last page).

Table 43. 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**
	`https://<NLB DNS>:9090/historian-rest-api/v1/datapoints/raw/{tagNames}/{start}/{end}/{direction}/{count}`
	POST
	`https://<NLB DNS>:9090/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. `https://<NLB DNS>:9090/historian-rest-api/datapoints/raw/tagName1/2013-10-02T11:30:00.111Z/2013-10-02T11:31:11.111Z/0/100https://<NLB DNS>:9090/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` **Raw By Time** The count value equals 0.

	`https://<NLB DNS>:9090/historian-rest-api/datapoints/raw/tagName1/2013-10-02T11:30:00.111Z/2013-10-02T11:31:11.111Z/0/0https://<NLB DNS>:9090/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**
	Count value is a non-zero positive number, and end time is greater than start time.
	`https://<NLB DNS>:9090/historian-rest-api/datapoints/raw/2013-10-02T11:30:00.111Z/2013-10-02T11:31:11.111Z/0/100https://<NLB DNS>:9090/historian-rest-api/datapoints/raw/start=2013-10-02T11:30:00.111Z&end=2013-10-02T11:31:11.111Z&count=100&direction=0`
	Raw By Time
	The count value equals 0.
	`https://<NLB DNS>:9090/historian-rest-api/datapoints/raw/2013-10-02T11:30:00.111Z/2013-10-02T11:31:11.111Z/0/0https://<NLB DNS>:9090/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 44. 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 from the starting time) of data sampling from the archive. The default value is Forward (1).	Yes	Integer, with a value such as 1.
Count	Count of archive values within each calculation interval.	Yes	Integer, with a value such as 100.

Table 45. Response Parameters

Parameter	Data Type	Required?	Description
ErrorCode	Integer	Yes	For example, 0.
ErrorMessage	String	Yes	For example, NULL.

Table 45. Response Parameters (continued)

Parameter	Data Type	Required?	Description

Data	String	Yes	The object container for the following parameters:
			DataType
			DoubleFloat, which stores decimal values up to 15 places.
			ErrorCode
			Value is 0, which means the operation was successful.
			TagName
			Example: TagName1.
			Samples
			Provides TimeStamp, Value, Quality, and DtatAttributes for each sample. Where, DataAttributes is the detailed information regarding the Quality.
			For example,
			`"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**
	`https://<NLB DNS>:9090/historian-rest-api/v1/datapoints/interpolated/{tagNames}/{start}/{end}/{count}/{intervalMs}`
	POST
	`https://<NLB DNS>:9090/historian-rest-api/v1/datapoints/interpolated/{start}/{end}/{count}/{intervalMs}`

SAMPLE GET URI:	`https://<NLB DNS>:9090/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://<NLB DNS>:9090/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 46. 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	Yes	DateTime

Table 46. Query Parameters (continued)

Parameter	Description	Required?	Values
	as YYYY-MM-DDTHH:mm:ss.SSSZ).		
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 47. 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: **DataType** DoubleFloat, which stores decimal values up to 15 places. **ErrorCode** Value is 0, which means the operation was successful. **TagName** Example is TagName1. **Samples**

Table 47. Response Parameters (continued)

Parameter	Data Type	Required?	Description
			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**
	`https://<NLB DNS>:9090/historian-rest-api/v1/datapoints/raw/{tagNames}/{start}/{end}/{direction}/{count}`
	POST
	`https://<NLB DNS>:9090/historian-rest-api/v1/datapoints/currentvalue`

SAMPLE GET URI:	`https://<NLB DNS>:9090/historian-rest-api/v1/datapoints/currentvalue?tagNames=tagName1`
SAMPLE POST URI:	`https://<NLB DNS>:9090/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 48. Query Parameters

Parameter	Description	Required?	Values
TagNames	Queries the specified tag names.	Yes	String

Table 49. 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: **DataType** DoubleFloat, which stores decimal values up to 15 places. **ErrorCode** Value is 0, which means the operation was successful. **TagName** Example is TagName1. **Samples** 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**
	`https://<NLB DNS>:9090/historian-rest-api/v1/datapoints/calculated/{tagNames}/{start}/{end}/{calculationMode}/{count}/{intervalMs}`
	POST
	`https://<NLB DNS>:9090/historian-rest-api/v1/datapoints/calculated/{start}/{end}/{calculationMode}/{count}/{intervalMs}`

SAMPLE GET URI:	**Number of Samples**
	`https://<NLB DNS>:9090/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
	`https://<NLB DNS>:9090/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**
	`https://<NLB DNS>:9090/historian-rest-api/v1/datapoints/calculated/2013-10-02T11:30:00.111Z/2013-10-02T11:31:11.111Z/1/100/1000`
	Time Range for List of Tags
	`https://<NLB DNS>:9090/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 50. 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
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.
IntervalMS	Interval in milliseconds.		64-bit signed integer, with a value such as 1000.

Table 51. 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: **DataType** DoubleFloat, which stores decimal values up to 15 places. **ErrorCode** Value is 0, which means the operation was successful. **TagName** Example is TagName1. **Samples** 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** `https://<NLB DNS>:9090/historian-rest-api/v1/datapoints/sampled` **POST** `https://<NLB DNS>:9090/historian-rest-api/v1/datapoints/sampled`
SAMPLE GET URI:	`https://<NLB DNS>:9090/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://<NLB DNS>:9090/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 52. 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 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 53. Response Parameters

Parameter	Data Type	Required?	Description
ErrorCode	Integer	Yes	For example, 0.

Table 53. Response Parameters (continued)

Parameter	Data Type	Required?	Description
ErrorCode	String	Yes	For example, NULL.
Data	String	Yes	The object container for the following parameters: **DataType** DoubleFloat, which stores decimal values up to 15 places. **ErrorCode** Value is 0, which means the operation was successful. **TagName** Example is TagName1. **Samples** 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**

	`https://<NLB DNS>:9090/historian-rest-api/v1/datapoints/trend` **POST** `https://<NLB DNS>:9090/historian-rest-api/v1/datapoints/trend`
SAMPLE GET URI:	`https://<NLB DNS>:9090/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://<NLB DNS>:9090/historian-rest-api/v1/datapoints/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 54. 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

Table 54. 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
Sampling-Mode	Also known as SamplingModeType.	Optional	Integer, with a value such as 1.
Calculation-Mode	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 55. 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: **TagName** Name of the tag, such as ahistfile.Simulation00001.

Table 55. Response Parameters (continued)

Parameter	Data Type	Required?	Description
			TagSource Location where tags are being searched for. **DataType** Float, which stores decimal values up to 6 places. **Trend** Provides TimeStamp, Value and Quality for each sample. For example, 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://<NLB DNS>:9090/historian-rest-api/v1/tags/addtag`

SAMPLE URI:	`https://<NLB DNS>:9090/historian-rest-api/v1/tags/addtag`
	`Payload:`
	`{`
	`    "Name" : "SampleTag",`
	`    "DataType" : 3`
	`}`

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://<NLB DNS>:9090/historian-rest-api/v1/tags/addtag`

Table 56. Query Parameters

Parameter	Description	Required?	Values
Payload	JSON array of PropertyName and PropertyValue.	Yes. "Name" and "DataType" properties are required. All other properties are optional.	Multidata types. See Payload Parameter *(on page 859)* for a list of tag properties 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://<NLB DNS>:9090/historian-rest-api/v1/tags/addtags`

SAMPLE URI:	`https://<NLB DNS>:9090/historian-rest-api/v1/tags/addtags`
	
	`Payload:`
	
	`[`
	`{`
	`    "Name" : "SampleTag1",`

	`    "DataType" : 3`
	`},`
	`{`
	`    "Name" : "SampleTag2",`
	`    "DataType" : 3`
	`}`
	`]`

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://<NLB DNS>:9090/historian-rest-api/v1/tags/addtags`

Table 57. 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 859)* for a list of tag properties used to update a tag configuration.

Table 58. 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://<NLB DNS>:9090/historian-rest-api/v1/tags/properties/tagName`

SAMPLE DELETE URI:	`https://<NLB DNS>:9090/historian-rest-api/v1/tags/properties/tagName`
	
	`Payload:`
	`{`
	`    "PropertyName" : "PropertyValue"`
	`}`

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://<NLB DNS>:9090/historian-rest-api/v1/tags/properties/tagName`

Table 59. 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 859)* for a list of tag properties used to update a tag configuration.

Table 60. 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://<NLB DNS>:9090/historian-rest-api/v1/tags/properties/tagName This URI returns all tag properties.

URI: (POST)	https://<NLB DNS>:9090/historian-rest-api/v1/tags/properties/tagName
	Payload
	{
	"PropertyName1" : 1,
	"PropertyName2" : 1
	}

SAMPLE GET URI:	https://<NLB DNS>:9090/historian-rest-api/v1/tags/properties/tagName

SAMPLE POST URI:	https://<NLB DNS>:9090/historian-rest-api/v1/tags/properties/tagName
	Payload:
	{
	    "Description" : 1
	}

SAMPLE cURL GET COMMAND:	curl -i -H "Accept: application/json" -i -H "Content-Type: application/json" -H "Authorization: Bearer <TOKEN>” -X GET https://<NLB DNS>:9090/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://<NLB DNS>:9090/historian-rest-api/v1/tags/properties/tagName

Table 61. Query Parameters

Parameter	Description	Required?	Values
tagName	Tag name for which properties need to be retrieved.	Yes	String

Table 61. Query Parameters (continued)

Parameter	Description	Required?	Values
Payload	JSON array of PropertyName and boolean (true/false).	At least one property must be provided.	Multi data types. See Payload Parameter *(on page 859)* 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 62. 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://<NLB DNS>:9090/historian-rest-api/v1/tags/tagName?{permanentDelete}`
SAMPLE DELETE URI:	`https://<NLB DNS>:9090/historian-rest-api/v1/tags/tagName?permanentDelete=true`

SAMPLE CURL COMMAND:	`curl -i -H "Authorization: Bearer <TOKEN>" -X DELETE https://<NLB DNS:9090>:443/historian-rest-api/v1/tags/tagName?permanentDelete=<true|false>`

Table 63. Query Parameters

Parameter	Description	Required?	Values
tagName	Name of the tag to be deleted.	Yes	String
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 64. 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://<NLB DNS>:9090/historian-rest-api/v1/datapoints/ configuration/{maxDataQueryResultSize}`
SAMPLE URI:	`https://<NLB DNS>:9090/historian-rest-api/v1/datapoints/ configuration?maxDataQueryResultSize=6000`

SAMPLE CURL COMMAND:	`curl -i -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>” https://<nodename>:443/ historian-rest-api/v1/datapoints/configuration? maxDataQueryResultSize=<Number_Of_Query_Results>`

Table 65. Query Parameters

Parameter	Description	Required?	Values
maxDataQueryResultSize	Maximum samples that should be configured as part of Query Results.	Yes	Integer

Table 66. 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://<NLB DNS>:9090/historian-rest-api/v1/tags/tagrename/oldtagname/newtagname?{truerename}`
SAMPLE URI	`https://<NLB DNS>:9090/historian-rest-api/v1/tags/tagrename/GDW14NV2E.Simulation0000101/GDW14NV2E.Simulation0000101newname?truerename=<true | false>`

METHOD:	PUT
SAMPLE CURL COMMAND	`curl -i -H "Accept: application/json" -i -H "Content-Type: application/json"-H "Authorization: Bearer <TOKEN> -X PUT https://<NLB DNS>:9090/historian-rest-api/v1/tags/tagrename/<oldtagname>/<newtagname>?truerename=<true | false>`

Table 67. 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. Creates an alias if the value entered is false.	Optional (false is default)	Boolean (true or false)

Table 68. 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://<NLB DNS>:9090/historian-rest-api /v1/datapoints/create

SAMPLE URI	https://<NLB DNS>:9090/historian-rest-api
	/v1/datapoints/create
	
	Payload
	
	{
	"TagName": "GDW14NV2E.Simulation00015",
	"samples": [
	{
	
	"TimeStamp":
	"2019-09-17T15:58:00.000Z",
	
	"Value": "1",
	
	"Quality": 3
	}
	]
	}

SAMPLE RESPONSE	{
	
	"ErrorCode": 0,
	
	"ErrorMessage": ""
	
	}

SAMPLE CURL COMMAND	curl -i -H "Accept: application/json" -i -H "Content-Type: application/json" -H "Authorization: Bearer <TOKEN>” -d “{ \”TagName\”:\”GDW14NV2E.Simulation00015\”,\"samples\":[{\"TimeStamp\": \ "2019-09-17T15:58:00.000Z\",\"Value\": \"1\",\"Quality\": 3}]}” -X POST

Method	POST
	`https://<NLB DNS>:9090/historian-rest-api/v1/datapoints/create`

Table 69. 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 70. 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.

REST APIs for Managing Tag Data

This topic provides REST APIs that you can use to manage tags. You can add, access, modify, rename, and delete tags.

Before You Begin: Get an authorization token *(on page 844)*.

Table 71. Access Raw Data Using the GET Method

Parameter	Value
Method	GET
URI	`https://<NLB DNS of Proficy Historian for Cloud>:9090/historian-rest-api/v1/datapoints/raw/<tag names>/<start>/<end>/<direction>/<count>` **Tip:** To find the NLB DNS:

Table 71. Access Raw Data Using the GET Method (continued)

Parameter	Value
	1. Access the EKS cluster on which you have deployed Proficy Historian for AWS. 2. Access the EC2 instance. 3. In the navigation pane, under **Load Balancing**, select **Load Balancers**. 4. Select the load balancer for which you want to find the DNS. 5. In the **Description** section, copy the DNS name.
Authorization	Bearer *<token>*
Content type	application/json

Sample URI	- **Raw By Number:** Retrieves data samples up to a specified number.
	`https://<NLB DNS of Proficy Historian for Cloud>:9090/historian-rest-api/datapoints/raw/tag-Name1/2013-10-02T11:30:00.111Z/2013-10-02T11:31:11.111Z/0/100`
	- **Raw By Time:** Retrieves data samples up to a specified time.
	`https://<NLB DNS of Proficy Historian for Cloud>:9090/historian-rest-api/datapoints/raw/tag-Name1/2013-10-02T11:30:00.111Z/2013-10-02T11:31:11.111Z/0/0`

Sample cURL commands	- **Raw By Number:** `curl -i -H "Accept: application/json" -H "Authorization: Bearer <token>http://<NLB DNS of Proficy Historian for Cloud>:9090/historian-rest-api/v1/datapoints/raw/<tag name>/<start time>/<end time>/<direction>/<count>` - **Raw By Time:**

Table 71. Access Raw Data Using the GET Method (continued)

Parameter	Value
	`curl -i -H "Accept: application/json" -H "Authorization: Bearer <token>http://<NLB DNS of Proficy Historian for Cloud>:9090/historian-rest-api/v1/datapoints/raw/<tag name>/<start time>/<end time>/<direction>/0`
Query parameters	- **Tag names:** The names of tags whose data you want to retrieve. - **Start time:** The time from which you want to retrieve data. Enter a value in the ISO date format (YYYY-MM-DDTHH:mm:ss.SSSZ). - **End time:** The time up to which you want to retrieve data. Enter a value in the ISO date format (YYYY-MM-DDTHH:mm:ss.SSSZ). - **Direction:** The direction (forward or backward) from the start time for which you want to retrieve data. - **Count:** The count of the data samples within each calculation interval.

Table 72. Access Raw Data Using the POST Method

Parameter	Value
Method	POST
URI	`https://<NLB DNS of Proficy Historian for Cloud>:9090/historian-rest-api/v1/datapoints/raw/<start>/<end>/<direction>/<count>`
Authorization	Bearer *<token>*
Content type	application/json
Sample URI	- **Raw By Number:** Retrieves data samples up to a specified number. `https://<NLB DNS of Proficy Historian for Cloud>:9090/historian-rest-api/datapoints/raw/2013-10-02T11:30:00.111Z/2013-10-02T11:31:11.111Z/0/100`

Table 72. Access Raw Data Using the POST Method (continued)

Parameter	Value
	- **Raw By Time:** Retrieves data samples up to a specified time. `https://<NLB DNS of Proficy Historian for Cloud>:9090/historian-rest-api/datapoints/raw/2013-10-02T11:30:00.111Z/2013-10-02T11:31:11.111Z/0/0`

Sample cURL commands	- **Raw By Number:**
	`curl X POST -i -H "Content-Type: application/json" -H "Accept: application/json" -H "Authorization: Bearer <token>-d {\tagNames\:\<tag name>;<tag name>\}http://<NLB DNS of Proficy Historian for Cloud>/historian-rest-api/v1/datapoints/raw/<tag name><start time>/<end time>/<direction>/<count>`
	- **Raw By Time:**
	`curl X POST -i -H "Content-Type: application/json" -H "Accept: application/json" -H "Authorization: Bearer <token>-d {\tagNames\:\<tag name>;<tag name>\}http://<NLB DNS of Proficy Historian for Cloud>/historian-rest-api/v1/ datapoints/raw?start=<start time>&end=<end time>&direction=<direction>&count=<count>`

Query parameters	- **Tag names:** The names of tags whose data you want to retrieve. - **Start time:** The time from which you want to retrieve data. Enter a value in the ISO date format (YYYY-MM-DDTHH:mm:ss.SSSZ). - **End time:** The time up to which you want to retrieve data. Enter a value in the ISO date format (YYYY-MM-DDTHH:mm:ss.SSSZ). - **Direction:** The direction (forward or backward) from the start time for which you want to retrieve data. - **Count:** The count of the data samples within each calculation interval.

Table 72. Access Raw Data Using the POST Method (continued)

Parameter	Value

Body	{
	"ErrorCode" : 0,
	"ErrorMessage": null,
	"Data":[{
	"DataType":"*<value>*",
	"ErrorCode":0,
	"TagName":"*<value>*",
	"Samples":
	[
	{ "TimeStamp":"*<value>*","Value":"*<value>*","Quality":*<value>* },
	{ "TimeStamp":"*<value>*","Value":"*<value>*","Quality":*<value>* },
	{ "TimeStamp":"*<value>*","Value":"*<value>*","Quality":*<value>* }
	]
	}]
	}

Example	{
	"ErrorCode" : 0,
	"ErrorMessage": null,
	"Data":[{
	"DataType":"DoubleFloat",
	"ErrorCode":0,
	"TagName":"TagName1",
	"Samples":
	[
	{ "TimeStamp":"2013-10-02T11:30:00.111Z","Value":"34.26155","Quality":3 },
	{ "TimeStamp":"2013-10-02T11:30:10.111Z","Value":"37.26155","Quality":3 },
	{ "TimeStamp":"2013-10-02T11:31:00.111Z","Value":"33.26155","Quality":3 }
	]
	}]
	}

Table 73. Access Interpolated Data

Parameter	Value
Method	GET
URI	`https://<NLB DNS of Proficy Historian for Cloud>:9090/historian-rest-api/v1/datapoints/interpolated/<start>/<end>/<count>/<interval in milliseonds>`
Authorization	Bearer *<token>*
Content type	application/json
Sample URI	`https://<NLB DNS of Proficy Historian for Cloud>:9090/historian-rest-api/v1/datapoints/interpolated/tagName1/2013-10-02T11:30:00.111111Z/2013-10-02T11:31:11.111Z/100/10000`
Sample cURL commands	`curl -i -H "Accept: application/json" -H "Authorization: Bearer <token>http://<NLB DNS of Proficy Historian for Cloud>:9090/historian-rest-api/v1/datapoints/interpolated/<tag name>/<start time>/<end time>/<count>/<interval in milliseconds>`
Query parameters	- **Tag names:** The names of tags whose data you want to retrieve. - **Start time:** The time from which you want to retrieve data. Enter a value in the ISO date format (YYYY-MM-DDTHH:mm:ss.SSSZ). - **End time:** The time up to which you want to retrieve data. Enter a value in the ISO date format (YYYY-MM-DDTHH:mm:ss.SSSZ). - **Interval:** The interval in milliseconds. - **Count:** The count of the data samples within each calculation interval.

Body	{
	"ErrorCode" : 0,
	"ErrorMessage": null,
	"Data":[{
	"DataType":"*<value>*",
	"ErrorCode":0,
	"TagName":"*<value>*",

Table 73. Access Interpolated Data (continued)

Parameter	Value

	`"Samples":`
	`[`
	`{ "TimeStamp":"<value>","Value":"<value>","Quality":<value> },`
	`{ "TimeStamp":"<value>","Value":"<value>","Quality":<value> },`
	`{ "TimeStamp":"<value>","Value":"<value>","Quality":<value> }`
	`]`
	`}]`
	`}`

Example	`{`
	`"ErrorCode" : 0,`
	`"ErrorMessage": null,`
	`"Data":[{`
	`"DataType":"DoubleFloat",`
	`"ErrorCode":0,`
	`"TagName":"TagName1",`
	`"Samples":`
	`[`
	`{ "TimeStamp":"2013-10-02T11:30:00.111Z","Value":"34.26155","Quality":3 },`
	`{ "TimeStamp":"2013-10-02T11:30:10.111Z","Value":"37.26155","Quality":3 },`
	`{ "TimeStamp":"2013-10-02T11:31:00.111Z","Value":"33.26155","Quality":3 }`
	`]`
	`}]`
	`}`

Table 74. Update Interpolated Data

Parameter	Value
Method	POST
URI	`https://<NLB DNS of Proficy Historian for Cloud>:9090/historian-rest-api/v1/datapoints/interpolated/<start>/<end>/<count>/<interval in milliseonds>`

Table 74. Update Interpolated Data (continued)

Parameter	Value
Authorization	Bearer *<token>*
Content type	application/json
Sample URI	`https://<NLB DNS of Proficy Historian for Cloud>:9090/historian-rest-api/v1/datapoints/interpolated/2013-10-02T11:30:00.111111Z/2013-10-02T11:31:11.111Z/100/10000`
Sample cURL commands	`curl -i -H "Accept: application/json" -H "Authorization: Bearer <token>http://<NLB DNS of Proficy Historian for Cloud>:9090/historian-rest-api/v1/datapoints/interpolated/<start time>/<end time>/<count>/<interval in milliseconds>`
Query parameters	• **Tag names:** The names of tags whose data you want to retrieve. • **Start time:** The time from which you want to retrieve data. Enter a value in the ISO date format (YYYY-MM-DDTHH:mm:ss.SSSZ). • **End time:** The time up to which you want to retrieve data. Enter a value in the ISO date format (YYYY-MM-DDTHH:mm:ss.SSSZ). • **Interval:** The interval in milliseconds. • **Count:** The count of the data samples within each calculation interval.

Body	`{`
	`"ErrorCode" : 0,`
	`"ErrorMessage": null,`
	`"Data":[{`
	`"DataType":"<value>",`
	`"ErrorCode":0,`
	`"TagName":"<value>",`
	`"Samples":`
	`[`
	`{ "TimeStamp":"<value>","Value":"<value>","Quality":<value> },`
	`{ "TimeStamp":"<value>","Value":"<value>","Quality":<value> },`

Table 74. Update Interpolated Data (continued)

Parameter	Value

	`{ "TimeStamp":"<value>","Value":"<value>","Quality":<value> }`
	`]`
	`}]`
	`}`

Example	`{`
	`"ErrorCode" : 0,`
	`"ErrorMessage": null,`
	`"Data":[{`
	`"DataType":"DoubleFloat",`
	`"ErrorCode":0,`
	`"TagName":"TagName1",`
	`"Samples":`
	`[`
	`{ "TimeStamp":"2013-10-02T11:30:00.111Z","Value":"34.26155","Quality":3 },`
	`{ "TimeStamp":"2013-10-02T11:30:10.111Z","Value":"37.26155","Quality":3 },`
	`{ "TimeStamp":"2013-10-02T11:31:00.111Z","Value":"33.26155","Quality":3 }`
	`]`
	`}]`
	`}`

Table 75. Access Current Data

Parameter	Value
Method	GET
URI	`https://<NLB DNS of Proficy Historian for Cloud>:9090/historian-rest-api/v1/datapoints/currentvalue`
Authorization	Bearer *<token>*
Content type	application/json

Table 75. Access Current Data (continued)

Parameter	Value
Sample URI	`https://<NLB DNS of Proficy Historian for Cloud>:9090/historian-rest-api/v1/datapoints/currentvalue?tagNames=tagName1`
Sample cURL commands	`curl -i -H "Accept: application/json" -H "Authorization: Bearer <token>http://<NLB DNS of Proficy Historian for Cloud>:9090/historian-rest-api/v1/datapoints/currentvalue/<tag name>`
Query parameters	**Tag names:** The names of tags whose data you want to retrieve.

Body	{
	"ErrorCode" : 0,
	"ErrorMessage": null,
	"Data":[{
	"DataType":"DoubleFloat",
	"ErrorCode":0,
	"TagName":"TagName1",
	
	"Samples":[{ "TimeStamp":"<value>","Value":"<value>","Quality":<value> }]
	}]
	}

Example	{
	"ErrorCode" : 0,
	"ErrorMessage": null,
	"Data":[{
	"DataType":"DoubleFloat",
	"ErrorCode":0,
	"TagName":"TagName1",
	
	"Samples":[{ "TimeStamp":"2014-01-01T12:00:00Z","Value":"34.26155","Quality":3 }]
	}]
	}

Table 76. Update Current Data

Parameter	Value
Method	POST
URI	`https://<NLB DNS of Proficy Historian for Cloud>:9090/historian-rest-api/v1/datapoints/currentvalue`
Authorization	Bearer *<token>*
Content type	application/json
Sample URI	`https://<NLB DNS of Proficy Historian for Cloud>:9090/historian-rest-api/v1/datapoints/currentvalue`
Sample cURL command	`curl -i X POST -H "Content-Type: application/json" -H "Accept: application/json" -H "Authorization: Bearer <token>-d {\tagNames\:\<tag name>\}http://<NLB DNS of Proficy Historian for Cloud>:9090/historian-rest-api/v1/datapoints/currentvalue`
Query parameters	**Tag names:** The names of tags whose data you want to retrieve.

Body	`{`
	`"ErrorCode" : 0,`
	`"ErrorMessage": null,`
	`"Data":[{`
	`"DataType":"<value>",`
	`"ErrorCode":0,`
	`"TagName":"<value>",`
	`"Samples":`
	`[`
	`{ "TimeStamp":"<value>","Value":"<value>","Quality":<value> },`
	`{ "TimeStamp":"<value>","Value":"<value>","Quality":<value> },`
	`{ "TimeStamp":"<value>","Value":"<value>","Quality":<value> }`
	`]`
	`}]`
	`}`

Example	`{`
	`"ErrorCode" : 0,`
	`"ErrorMessage": null,`
	`"Data":[{`

Table 76. Update Current Data (continued)

Parameter	Value

	`"DataType":"DoubleFloat",`
	`"ErrorCode":0,`
	`"TagName":"TagName1",`
	`"Samples":`
	`[`
	`{ "TimeStamp":"2013-10-02T11:30:00.111Z","Value":"34.26155","Quality":3 },`
	`{ "TimeStamp":"2013-10-02T11:30:10.111Z","Value":"37.26155","Quality":3 },`
	`{ "TimeStamp":"2013-10-02T11:31:00.111Z","Value":"33.26155","Quality":3 }`
	`]`
	`}]`
	`}`

Table 77. Access Sampled Data

Parameter	Value
Method	GET
URI	`https://<NLB DNS of Proficy Historian for Cloud>:9090/historian-rest-api/v1/datapoints/sampled/<tag names>/<start time>/<end time>/<direction>/<count>/<interval>`
Authorization	Bearer *<token>*
Content type	application/json
Sample URI	`https://<NLB DNS of Proficy Historian for Cloud>:9090/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 cURL commands	`curl -i -H "Accept: application/json" -H "Authorization: Bearer <token>http://<NLB DNS of Proficy Historian for Cloud>:8843/historian-rest-api/v1/datapoints/sampled/<tag name>/<start time>/<end time>/<direction>/<count>/<interval>`

Table 77. Access Sampled Data (continued)

Parameter	Value
Query parameters	- **Tag names:** The names of tags whose data you want to retrieve. - **Start time:** The time from which you want to retrieve data. Enter a value in the ISO date format (YYYY-MM-DDTHH:mm:ss.SSSZ). - **End time:** The time up to which you want to retrieve data. Enter a value in the ISO date format (YYYY-MM-DDTHH:mm:ss.SSSZ). - **Interval:** The interval in milliseconds. - **Direction:** The direction (forward or backward) from the start time for which you want to retrieve data. - **Count:** The count of the data samples within each calculation interval.

Body	`{`
	`"ErrorCode" : 0,`
	`"ErrorMessage": null,`
	`"Data":[{`
	`"DataType":"DoubleFloat",`
	`"ErrorCode":0,`
	`"TagName":"TagName1",`
	
	`"Samples":[ { "TimeStamp":"`*`<value>`*`","Value":"`*`<value>`*`","Quality":`*`<value>`*` } ]`
	`}]`
	`}`

Example	`{`
	`"ErrorCode" : 0,`
	`"ErrorMessage": null,`
	`"Data":[{`
	`"DataType":"DoubleFloat",`
	`"ErrorCode":0,`
	`"TagName":"TagName1",`

Table 77. Access Sampled Data (continued)

Parameter	Value
	"Samples":[{ "TimeStamp":"2014-01-01T12:00:00Z","Value":"34.26155","Quality":3 }] }] }

Table 78. Update Sampled Data

Parameter	Value
Method	POST
URI	https://*<NLB DNS of Proficy Historian for Cloud>*:9090/historian-rest-api/v1/datapoints/sampled/*<tag names>/<start time>/<end time>*/*<direction>*/*<count>*/<interval>
Authorization	Bearer *<token>*
Content type	application/json
Sample URI	https://*<NLB DNS of Proficy Historian for Cloud>*:9090/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 cURL command	curl -i X POST -H "Content-Type: application/json" -H "Accept: application/json" -H "Authorization: Bearer *<token>*-d { \tagNames\:*<value>*\, \start\: *<value>*\, \end\: *<value>*\, \samplingMode\: *<value>*, \calculationMode\: *<value>*, \direction\: *<value>*, \count\: *<value>*, \returnDataFields\: *<value>*, \intervalMs\: *<value>*, \queryModifier\: *<value>*, \filterMode\: *<value>*, \filterExpression\: *<value>*\}http://*<NLB DNS of Proficy Historian for Cloud>*:9090/historian-rest-api/v1/datapoints/sampled

Table 78. Update Sampled Data (continued)

Parameter	Value
Query parameters	• **Tag names:** The names of tags whose data you want to retrieve. • **Start time:** The time from which you want to retrieve data. Enter a value in the ISO date format (YYYY-MM-DDTHH:mm:ss.SSSZ). • **End time:** The time up to which you want to retrieve data. Enter a value in the ISO date format (YYYY-MM-DDTHH:mm:ss.SSSZ). • **Interval:** The interval in milliseconds. • **Sampling mode:** The sampling mode using which you want to retrieve data. • **Direction:** The direction (forward or backward) from the start time for which you want to retrieve data. • **Count:** The count of the data samples within each calculation interval.

Body	`{`
	`"ErrorCode" : 0,`
	`"ErrorMessage": null,`
	`"Data":[{`
	`"DataType":"<value>",`
	`"ErrorCode":0,`
	`"TagName":"<value>",`
	`"Samples":`
	`[`
	`{ "TimeStamp":"<value>","Value":"<value>","Quality":<value> },`
	`{ "TimeStamp":"<value>","Value":"<value>","Quality":<value> },`
	`{ "TimeStamp":"<value>","Value":"<value>","Quality":<value> }`
	`]`
	`}]`
	`}`

Example	`{`
	`"ErrorCode" : 0,`
	`"ErrorMessage": null,`

Table 78. Update Sampled Data (continued)

Parameter	Value

	"Data":[{
	"DataType":"DoubleFloat",
	"ErrorCode":0,
	"TagName":"TagName1",
	"Samples":
	[
	{ "TimeStamp":"2013-10-02T11:30:00.111Z","Value":"34.26155","Quality":3 },
	{ "TimeStamp":"2013-10-02T11:30:10.111Z","Value":"37.26155","Quality":3 },
	{ "TimeStamp":"2013-10-02T11:31:00.111Z","Value":"33.26155","Quality":3 }
	]
	}]
	}

Table 79. Access Trend Data

Parameter	Value
Method	GET
URI	https://*<NLB DNS of Proficy Historian for Cloud>*:9090/historian-rest-api/v1/datapoints/trend/*<tag names>*/*<start time>*/*<end time>*/*<direction>*/*<count>*/<interval>
Authorization	Bearer *<token>*
Content type	application/json
Sample URI	https://*<NLB DNS of Proficy Historian for Cloud>*:9090/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 cURL commands	curl -i -H "Accept: application/json" -H "Authorization: Bearer *<token>*http://*<NLB DNS of Proficy Historian for Cloud>*:8843/

Table 79. Access Trend Data (continued)

Parameter	Value
	`historian-rest-api/v1/datapoints/trend/<tag name>/<start time>/<end time>/<direction>/<count>/<interval>`
Query parameters	- **Tag names:** The names of tags whose data you want to retrieve. - **Start time:** The time from which you want to retrieve data. Enter a value in the ISO date format (YYYY-MM-DDTHH:mm:ss.SSSZ). - **End time:** The time up to which you want to retrieve data. Enter a value in the ISO date format (YYYY-MM-DDTHH:mm:ss.SSSZ). - **Sampling mode:** The sampling mode that you want to use to retrieve data. - **Calculation mode:** The calculation mode that you want to use to retrieve data. - **Interval:** The interval in milliseconds. - **Direction:** The direction (forward or backward) from the start time for which you want to retrieve data. - **Count:** The count of the data samples within each calculation interval.

Body	`{`
	`    "ErrorCode" : 0,`
	`    "ErrorMessage": null,`
	`    "Data":[{`
	`     "DataType":"DoubleFloat",`
	`     "ErrorCode":0,`
	`     "TagName":"TagName1",`
	
	` "Trend":[ { "TimeStamp":"<value>","Value":"<value>","Quality":<value> } ]`
	`     }]`
	`}`

Example	`{`
	`    "ErrorCode" : 0,`
	`    "ErrorMessage": null,`

Table 79. Access Trend Data (continued)

Parameter	Value

	"Data":[{
	"DataType":"DoubleFloat",
	"ErrorCode":0,
	"TagName":"TagName1",
	"Trend":[{ "TimeStamp":"2014-01-01T12:00:00Z","Value":"34.26155","Quality":3 }]
	}]
	}

Table 80. Update Sampled Data

Parameter	Value
Method	POST
URI	https://<*NLB DNS of Proficy Historian for Cloud*>:9090/historian-rest-api/v1/datapoints/trend/<*tag names*>/<*start time*>/<*end time*>/<*direction*>/<*count*>/<interval>
Authorization	Bearer <*token*>
Content type	application/json
Sample URI	https://<*NLB DNS of Proficy Historian for Cloud*>:9090/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 cURL command	curl -i X POST -H "Content-Type: application/json" -H "Accept: application/json" -H "Authorization: Bearer <*token*>-d { \tagNames\:\<*value*>\, \start\: \<*value*>\, \end\: \<*value*>\, \trend \: <*value*>, \calculationMode\: <*value*>, \direction\: <*value*>, \count\: <*value*>, \returnDataFields\: <*value*>, \intervalMs\: <*value*>, \queryModifier\: <*value*>, \filterMode\: <*value*>, \filterExpression\: \<*value*>\}http://<*NLB DNS of Proficy Historian for Cloud*>:9090/historian-rest-api/v1/datapoints/sampled

Table 80. Update Sampled Data (continued)

Parameter	Value
Query parameters	• **Tag names:** The names of tags whose data you want to retrieve. • **Start time:** The time from which you want to retrieve data. Enter a value in the ISO date format (YYYY-MM-DDTHH:mm:ss.SSSZ). • **End time:** The time up to which you want to retrieve data. Enter a value in the ISO date format (YYYY-MM-DDTHH:mm:ss.SSSZ). • **Interval:** The interval in milliseconds. • **Sampling mode:** The sampling mode using which you want to retrieve data. • **Calculation mode:** The calculation mode using which you want to retrieve data. • **Direction:** The direction (forward or backward) from the start time for which you want to retrieve data. • **Count:** The count of the data samples within each calculation interval.

Body	`{`
	`     "ErrorCode" : 0,`
	`     "ErrorMessage": null,`
	`     "Data":[{`
	`      "DataType":"<value>",`
	`      "ErrorCode":0,`
	`      "TagName":"<value>",`
	`      "Samples":`
	`        [`
	`          { "TimeStamp":"<value>","Value":"<value>","Quality":<value> },`
	`          { "TimeStamp":"<value>","Value":"<value>","Quality":<value> },`
	`          { "TimeStamp":"<value>","Value":"<value>","Quality":<value> }`
	`        ]`
	`     }]`
	`}`

Table 80. Update Sampled Data (continued)

Parameter	Value

Example	`{`
	`    "ErrorCode" : 0,`
	`    "ErrorMessage": null,`
	`    "Data":[{`
	`     "DataType":"DoubleFloat",`
	`     "ErrorCode":0,`
	`     "TagName":"TagName1",`
	`     "Samples":`
	`       [`
	` { "TimeStamp":"2013-10-02T11:30:00.111Z","Value":"34.26155","Quality":3 },`
	` { "TimeStamp":"2013-10-02T11:30:10.111Z","Value":"37.26155","Quality":3 },`
	` { "TimeStamp":"2013-10-02T11:31:00.111Z","Value":"33.26155","Quality":3 }`
	`       ]`
	`    }]`
	`}`

Table 81. Limit Query Results

Parameter	Value
Method	GET
URI	`https://<NLB DNS of Proficy Historian for Cloud>:9090/historian-rest-api/v1/datapoints/configuration/maxDataQueryResultSize=<value>` The minimum number of query result size to enter is 1000.
Authorization	Bearer *<token>*
Content type	application/json
Sample URI	`https://<NLB DNS of Proficy Historian for Cloud>:9090/historian-rest-api/v1/datapoints/configuration/maxDataQueryResultSize=6000`

Table 81. Limit Query Results (continued)

Parameter	Value
Sample cURL commands	`curl -i -H "Accept: application/json" -H "Authorization: Bearer <token>https://<NLB DNS of Proficy Historian for Cloud>:9090/historian-rest-api/v1/datapoints/configuration?maxDataQueryResultSize=<value>`
Query parameters	**maxDataQueryResultSize:** The maximum number of query results that you want to retrieve.

Managing Collector Instances

The Create Collector Instance API

Using the Create Collector Instance API, you can create a collector instance.

METHOD	POST
URI	`https://<NLB DNS:9090>/historian-rest-api/v1/collector/createnewinstance`

SAMPLE URI	`https://<NLB DNS:9090>/historian-rest-api/v1/collector/createnewinstance`
	Payload
	`{`
	`"mode":1,`
	`"CollectorSystemName":"xyz",`
	`"InterfaceDescription":"xyz_Simulation_<IP address>_2",`
	`"DataPathDirectory":"C:\\Proficy Historian Data",`
	`"CollectorDestination":"Historian",`
	`"winUserName":"","winPassword":"",`
	`"InterfaceSubType":"",`
	`"DestinationHistorianUserName":"abc",`
	`"DestinationHistorianPassword":"password",`

	`"DestinationHistorian":"<IP address>",`
	`"General1":"",`
	`"General2":"",`
	`"General3":"123",`
	`"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.

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 “{ \"mode\":1,\"CollectorSystemName\":\"xyz\",`

	\"InterfaceDescription\":\"xyz_Simulation_<IP address>_2\",\"DataPathDirectory\":\"C:\ \Proficy Historian Data\", \"CollectorDestination\":\"Historian\", \"winUserName\":\"\",\"winPassword\":\"\", \"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://<NLB DNS:9090>/historian-rest-api/v1/collector/creat enewinstance

Table 82. 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://<NLB DNS:9090>/historian-rest-api/v1/collector/insta ncedetails/<interface name>
SAMPLE QUERY PARAM GET URL	https://<NLB DNS:9090>/historian-rest-api/v1/collector/insta ncedetails/<interface name>

SAMPLE RESPONSE	`{`
	`"ErrorCode":0,`
	`"ErrorMessage":null,`
	`"Data":`
	`{`
	`"CloudDestination":"",`
	`"InterfaceSubType":"",`
	`"CollectorSystemName":"xyz",`
	`"Type":2,`
	`"DefaultCompression":false,`
	`"CloudInformationLogLevel":0,`
	`"InterfaceDataDir":"C:\\Proficy Historian Data",`
	`"SourceServer":"",`
	`"Username":"",`
	`"Password":"",`
	`"DestinationType":"Historian",`
	`"DestinationServer":"abc",`
	`"DebugLogLevel":0,`
	`"InterfaceInstallDrive":"C",`
	`"ConnnectionString":"xyz"`
	`}`
	`}`

SAMPLE cURL COMMAND	`curl -i -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>” https://<NLB DNS:9090>/historian-rest-api/v1/collector/instancedetails/xyz_Simulation_<IP address>_2`

Table 83. Query Parameters

Parameter	Description	Required?	Values
`interface name`	The interface name of the collector whose details you want to view.	Yes	String

Table 84. 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://<NLB DNS:9090>/historian-rest-api/v1/collector/editInstance`

SAMPLE URI	`https://<NLB DNS:9090>/historian-rest-api/v1/collector/editInstance`
	Payload
	`{"InterfaceName":"<source server>_<type of the collector>_<destination server>",`
	`"messageCompression":0,`
	`"azureLogLevel":1,`
	`"debugMode":0,`
	`"CollectorDestination":"Predix",`
	`"DestinationHistorian":"abc",`
	`"mode":1,`
	`"CloudDestinationAddress":"wss://def.run.abc.ice.predix.io/v1/stream/messages",`
	`"IdentityIssuer":"https://1234.predix-uaa.run.abc.ice.predix.io/oauth/token",`
	`"ClientID":"xyz",`
	`"ClientSecret":"123",`
	`"ZoneID":"1234",`
	`"Proxy":"http://1.2.3.4:80",`

	"ProxyUserName":"", "ProxyPassword":"", "DatapointAttribute1":"", "DatapointAttribute2":"", "DatapointAttribute3":"", "DatapointAttribute4":""} **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.

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>*\", \"InterfaceName\":\"*<source server>_<type of the collector>_<destination server>*\", \"messageCompression\":0,\"azureLogLevel\":1,

	`\"debugMode\":0,\"CollectorDestination\":`
	`\"Predix\",\"DestinationHistorian\":\"abc\",`
	`\"mode\":1,`
	`\"CloudDestinationAddress\":`
	`\"wss://wss://def.run.abc.ice.predix.io/v1/stream/messages\",`
	`\"IdentityIssuer\":`
	`\"https://1234.predix-uaa.run.abc.ice.predix.io/oauth/token/",`
	`\"ClientID\":\"HistQA\",\"ClientSecret\":`
	`\"Gei321itc\",\"ZoneID\":\"1234\",`
	`\"Proxy\":`
	`\"http://1.2.3.4:80\",\"ProxyUserName\":`
	`\"\",\"ProxyPassword\":\"\",`
	`\"DatapointAttribute1\":\"\",\"DatapointAttribute2\":\"\",\"DatapointAttribute3\":\"\",`
	`\"DatapointAttribute4\":\"\"}” -X PUT`
	`https://<NLB DNS:9090>/historian-rest-api/v1/collector/editInstance`

Table 85. 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://<NLB DNS:9090>/historian-rest-api/v1/collector/azureloglevel`

SAMPLE URI	`https://<NLB DNS:9090>/historian-rest-api/v1/collector/azureloglevel`
	Payload
	`{"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://<NLB DNS:9090>/historian-rest-api/v1/collector/azureloglevel`

Table 86. 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://<NLB DNS:9090>/historian-rest-api/v1/installcomponentdetails/collectorType/collectorSubType/machine`
SAMPLE URI	`https://<NLB DNS:9090>/historian-rest-api/v1/installcomponentdetails/2/-/abc`

SAMPLE RESPONSE	`{`
	`"ErrorCode":0,`
	`"ErrorMessage":null,`
	`"Data":`
	`{`
	`"InterfaceInstallDrive":"C",`
	`"InterfaceDataDir":"C:\\Proficy Historian Data",`
	`"CertPathDir":"NONE"`
	`}`
	`}`

SAMPLE cURL COMMAND	`curl -i -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>” https://<NLB DNS:9090>/historian-rest-api/v1/installcomponentdetails/2/-/abc`

Table 87. 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://<NLB DNS:9090>/historian-rest-api/v1/collector/messagecompression`

SAMPLE REQUEST	`{`
	`"InterfaceName":"<source server>_<type of the collector>_<destination server>",`
	`"messageCompression":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>\",`
	`\"messageCompression\":1}” -X PUT`
	`https://<NLB DNS:9090>/historian-rest-api/v1/collector/messagecompression`

Table 88. Query Parameters

Parameter	Description	Required?	Values
`interface name`	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 89. 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://<NLB DNS:9090>/historian-rest-api/v1/collector/deleteinstance`

SAMPLE REQUEST	`https://<NLB DNS:9090>/historian-rest-api/v1/collector/deleteinstance`
	`Payload`
	`{`
	`"InterfaceName":"<source server>_<type of the collector>_<destination server>",`
	`"deleteTags":true`
	`}`

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>\",\"deleteTags\":true}” -X PUT`

	`https://<NLB`
	`DNS:9090>/historian-rest-api/v1/collector/delet`
	`einstance`

Table 90. Query Parameters

Parameter	Description	Required?	Values
`interface name`	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 91. 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 Collectors

The Start Collector API

Using the Start Collector API, you can start a collector.

METHOD	PUT

URI	`https://<NLB`
	`DNS:9090>/historian-rest-api/v1/collector/start`

SAMPLE URI	Sample URI for the service mode:
	`https://<NLB`
	`DNS:9090>/historian-rest-api/v1/collector/start`
	`Payload`
	`{`

	`"interfaceName":"<source server>_<type of the collector>_<destination server>",`
	`"mode":1`
	`}`
	Sample URI for the command line mode:
	`https://<NLB DNS:9090>/historian-rest-api/v1/collector/start`
	
	`Payload`
	
	`{`
	`"interfaceName":"<source server>_<type of the collector>_<destination server>",`
	`"mode":2,`
	`“winUserName”:””,`
	`“winPassword”:””`
	`}`

SAMPLE RESPONSE	`{`
	`"ErrorCode": 0,`
	`"ErrorMessage": ""`
	`"Data": "Collector Start Initiated"`
	`}`

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://<NLB DNS:9090>/historian-rest-api/v1/collector/start`

Query parameters include the Payload parameter, which is a JSON file, which contains the following properties.

Table 92. 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 • 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 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.
`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://<NLB DNS:9090>/historian-rest-api/v1/collector/stop`

SAMPLE URI	https://<NLB
	DNS:9090>/historian-rest-api/v1/collector/stop
	
	Payload
	
	{
	“interfaceName":"*<source server>_<type of the collector>_<destination server>*"
	“winUserName”:”TestAdmin”,
	“winPassword”:”TestAdminPassword”
	}

SAMPLE RESPONSE	{
	    "ErrorCode": 0,
	    "ErrorMessage": null,
	    "Data": "Collector Stop Initiated"
	}

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>*\”}” -X PUT https://<NLB DNS:9090>/historian-rest-api/v1/collector/stop

Query parameters include the Payload parameter, which is a JSON file, which contains the following properties.

Table 94. 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 com-	String

Table 94. Query Parameters (continued)

Parameter	Description	Required?	Values
		mand-line mode)	
`winPassword`	The Windows password	Yes (only if you want to use the command-line mode)	String

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.
`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://<NLB`
	`DNS:9090>/historian-rest-api/v1/collector/rest`
	`art`

SAMPLE URI	`https://<NLB`
	`DNS:9090>/historian-rest-api/v1/collector/rest`
	`art`
	
	`Payload`
	`{`
	`  “interfaceName":"`*`<source server>_<type of the`*
	`collector>_<destination server>``"`
	`   "winUserName":"",`

	`"winPassword":""`
	`}`

SAMPLE RESPONSE	`{`
	`"ErrorCode": 0,`
	`"ErrorMessage": null,`
	`"Data": "Collector Restart Initiated"`
	`}`

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>\”}”`
	`-X PUT https://<NLB`
	`DNS:9090>/historian-rest-api/v1/collector/rest`
	`art`

Query parameters include the Payload parameter, which is a JSON file, which contains the following properties.

Table 96. 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 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.
`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://<NLB`
	`DNS:9090>/historian-rest-api/v1/collector/pause`
	`collection/{interfaceName}`

SAMPLE URI	`https://<NLB`
	`DNS:9090>/historian-rest-api/v1/collector/pause`
	`collection/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://<NLB`
	`DNS:9090>/historian-rest-api/v1/collector/pause`
	`collection/RSSERVER2012-02_Simulation`

Table 98. 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://<NLB DNS:9090>/historian-rest-api/v1/collector/resumecollection/{interfaceName}`
SAMPLE URI	`https://<NLB DNS:9090>/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://<NLB DNS:9090>/historian-rest-api/v1/collector/resumecollection/RSSERVER2012-02_Simulation`

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.
`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://<NLB DNS:9090>/historian-rest-api/v1/tags/addcomment`
SAMPLE URI	`https://<NLB DNS:9090>/historian-rest-api/v1/tags/addcomment`

	Payload
	{
	"tagName":"rsserver2012-02.Simulation00003",
	"comment":"Retest",
	"timeStamp":"2020-04-22T00:00:00.000Z"
	}

SAMPLE RESPONSE	{
	"ErrorCode": 0,
	"ErrorMessage": null,
	"Data": ""
	}

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://<NLB DNS>:9090/historian-rest-api/v1/tags/addcomment

Query parameters include the Payload parameter, which is a JSON file, which contains the following properties.

Table 100. 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 101. Response Parameters

Parameter	Data Type	Required?	Description
`ErrorCode`	Integer	Yes	For example, ErrorCode = 0 implies the operation was successful.

Table 101. Response Parameters (continued)

Parameter	Data Type	Required?	Description
`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.

<table>
<tr><td>METHOD</td><td>GET</td></tr>
<tr><td>URI</td><td><pre>https://<NLB
 DNS:9090>/historian-rest-api/v1/tags/
comments/{tagNames}/{start}/{end}</pre></td></tr>
<tr><td>SAMPLE QUERY PARAM GET URI</td><td><pre>https://<NLB DNS:9090>/historian-rest-api/
v1/
tags/comments/?tagNames=rsserver2012-02.Simulati
on00003;
rsserver2012-02.Simulation00004&start=2020-04-19
T00:
00:00.000Z&end=2020-04-24T00:00:00.000Z</pre>
<b>Note:</b>
The query parameter supports multiple tags.</td></tr>
<tr><td>SAMPLE RESPONSE</td><td><pre>{
    "ErrorCode": 0,
    "ErrorMessage": null,
    "Data": [
        {
            "TagName": "Motor Temperature",
            "ErrorCode": 0,
            "Comments": [
                {
                    "TimeStamp":
 "2020-04-22T00:00:00.000Z",</pre></td></tr>
</table>

```
                    "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://<NLB DNS:9090> /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 102. Query Parameters

Parameter	Description	Required?	Values
`tagNames`	The names of the tag as a semicolon-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 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 Set Debug Mode API

Using the Set Debug Mode API, you can set the debug mode of a collector.

METHOD	PUT
URI	`https://<NLB DNS:9090>/historian-rest-api/v1/collector/setdebugmode`

SAMPLE URI	`https://<NLB DNS:9090>/historian-rest-api/v1/collector/setdebugmode`
	
	`Payload`

	`{`
	`“interfaceName":"<source server>_<type of the collector>_<destination server>",`
	`“debugMode”:255`
	`}`

SAMPLE RESPONSE	`{`
	`"ErrorCode": 0,`
	`"ErrorMessage": ""`
	`}`

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>\”,`
	`\”debugMode\”}” -X PUT https://<NLB DNS:9090>/historian-rest-api/v1/collector/setdebugmode`

Query parameters include the Payload parameter, which is a JSON file, which contains the following properties.

Table 104. 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 • 255: Debug log level

Table 105. Response Parameters

Parameter	Data Type	Required?	Description
`ErrorCode`	Integer	Yes	For example, ErrorCode = 0 implies the operation was successful.

Table 105. 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://<NLB DNS:9090>/historian-rest-api/v1/collector/buffercontrol`

SAMPLE URI	Sample URI for moving buffer files:
	`https://<NLB DNS:9090>/historian-rest-api/v1/collector/buffercontrol`
	
	`Payload`
	`{`
	`  “interfaceName":"<source server>_<type of the collector>_<destination server>",`
	`  “bufferMode”:2,`
	`     "winUserName":"Administrator",`
	`        "winPassword":"xxxxxxx",`
	`  "bufferPath": "C:\\Users\\bufffiles"`
	`}`
	Sample URI for deleting buffer files:
	`https://<NLB DNS:9090>/historian-rest-api/v1/collector/buffercontrol`

	Payload
	{
	“interfaceName":"<source server>_<type of the
	collector>_<destination server>",
	“bufferMode”:1
	"winUserName":"Administrator",
	"winPassword":"xxxxxxx",
	}

SAMPLE RESPONSE	Sample response for moving buffer files:
	{
	"ErrorCode": 0,
	"ErrorMessage": null,
	"Data": "BufferFiles Move Initiated.
	Collector is in the Stopped state."
	}
	Sample response for deleting buffer files:
	{
	"ErrorCode": 0,
	"ErrorMessage": null,
	"Data": "BufferFiles Delete Initiated.
	Collector is in the Stopped state."
	}

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>\”, \”bufferMode \”:1,\”bufferPath \”:\” C:\\Users\\bufffiles\”}” -X PUT https://<NLB DNS:9090>/historian-rest-api/v1/collector/buffe rcontrol

Query parameters include the Payload parameter, which is a JSON file, which contains the following properties.

Table 106. 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. • 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` **Note:** 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 106. Query Parameters (continued)

Parameter	Description	Required?	Values
		mand-line mode)	

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.
`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.

METHOD	PUT

URI	`https://<NLB`
	`DNS:9090>/historian-rest-api/v1/collector/historiannodechange`

SAMPLE URI	`https://<NLB`
	`DNS:9090>/historian-rest-api/v1/collector/historiannodechange`
	
	`Payload`
	`{`
	`“interfaceName":"<source server>_<type of the collector>_<destination server>",`
	`“mode”:2,`
	`"winUserName":"TestAdministrator",`
	`"winPassword":"TestPassword",`
	`"historianNode":"VMHISTWEBAUTO",`
	`"historianUserName":" TestAdministrator2 ",`

	`"historianpassword":" TestPassword2"`
	`}`

SAMPLE RESPONSE	`{`
	`"ErrorCode": 0,`
	`"ErrorMessage": ""`
	`}`

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>\”,`
	`\”userName`
	`winUserName\”:\”tesrt\”,\”winpPassword`
	`\”:\”password\”,`
	`\”historianNode \”:\”nodename\”,\”`
	`historianUserName \”:\”husername\”,`
	`\” historianpassword \”:\”hpassword\”}” -X PUT`
	`https://<NLB`
	`DNS:9090>/historian-rest-api/v1/collector/`
	`historiannodechange`

Query parameters include the Payload parameter, which is a JSON file, which contains the following properties.

Table 108. 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 • 2: command-line mode
`winUserName`	The Windows username.	Yes (only if you want to use the com-	String

Table 108. Query Parameters (continued)

Parameter	Description	Required?	Values
		mand-line mode)	
`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 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 Get Collector Version API

Using the Get Collector Version API, you can view the version number of a collector.

METHOD	GET

URI	`https://<NLB DNS:9090>/historian-rest-api/v1/collector/version/{interfaceName}`
SAMPLE QUERY PARAM GET URL	`https://<NLB DNS:9090>/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://<NLB DNS:9090>/historian-rest-api/v1/collector/version/RSSERVER2012-02_Simulation`

Table 110. Query Parameters

Parameter	Description	Required?	Values
`interfaceName`	The interface name of the collector.	Yes	String

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.
`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://<NLB DNS:9090>/historian-rest-api/v1/collector/status/{interfaceName}`
SAMPLE GET URI	`https://<NLB DNS:9090>/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://<NLB DNS:9090>/historian-rest-api/v1/collector/status/RSSERVER2012-02_Simulation`

Table 112. 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://<NLB DNS:9090>/historian-rest-api/v1/collectormanagerlist`
SAMPLE QUERY PARAM GET URL	`https://<NLB DNS:9090>/historian-rest-api/v1/collectormanagerlist`
SAMPLE RESPONSE	{ "ErrorCode": 0, "ErrorMessage": null, "Data": [{ "Name": "CollectorManager::abc", "IPAddress": "[::ffff :<IP address>]", "Status": 1, "ComputerName": "abc " }, { "Name": "CollectorManager::xyz", "IPAddress": "[::ffff:<IP address>]", "Status": 1, "ComputerName": "xyz" }, { "Name": "CollectorManager::abc", "IPAddress": "[::ffff:<IP address>]", "Status": 1, "ComputerName": "abc" }, { "Name": "CollectorManager::123", "IPAddress": "[::ffff:<IP address>]", "Status": 1,

	`"ComputerName": "123"`
	`}`
	`]`
	`}`

SAMPLE cURL COMMAND	`curl -i -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>” https://<NLB DNS:9090>/historian-rest-api/v1/collectormanagerlist`

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 Collector Mode API

Using the Collector Mode API, you can view the running mode of a collector.

METHOD	GET
URI	`https://<NLB DNS:9090>/historian-rest-api/v1/collector/mode/<collector interface name>`
SAMPLE QUERY PARAM GET URL	`https://<NLB DNS:9090>/historian-rest-api/v1/collector/mode/<collector interface name>`

SAMPLE RESPONSE	`{`
	`"ErrorCode": 0,`
	`"ErrorMessage": null,`
	`"Data": {`
	`"RunningMode": "Service Mode"`
	`}`
	`}`

SAMPLE cURL COMMAND	`curl -i -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>” https://<NLB DNS:9090>/historian-rest-api/v1/collector/mode/<host name>_Simulation_<IP address>_2`

Table 114. 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://<NLB DNS:9090>/historian-rest-api/v1/collector/details`
SAMPLE QUERY PARAM GET URL	`https://<NLB DNS:9090>/historian-rest-api/v1/collector/details`

SAMPLE RESPONSE	`{`
	`"ErrorCode":0,`
	`"ErrorMessage":null,`
	`"Data":`
	`[`
	`{`
	`"Name":"<value>",`
	`"ComputerName":"<value>",`
	`"Status":"Running",`
	`"ReportRate":0,`
	`"MaximumEventRate":0,`
	`"MinimumEventRate":0,`

	`"OutOfOrderEvents":0,`
	`"Overruns":0,`
	`"OverrunsPercent":0,`
	`"TotalEventsCollected":0,`
	`"TotalEventsReported":0,`
	`"LastDataValue":"\\\"1970-01-01T00:00:00.000Z\\`
	`\"",`
	`"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://<NLB`
	`DNS:9090>/historian-rest-api/v1/collector/deta`
	`ils`

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 Offline Collectors API

Using the Offline Collectors API, you can view a list of offline collectors.

METHOD	GET

URI	`https://<NLB`
	`DNS:9090>/historian-rest-api/v1/offlinecollect`
	`ors`

SAMPLE QUERY PARAM GET URL	`https://<NLB DNS:9090>/historian-rest-api/v1/offlinecollectors`

SAMPLE RESPONSE	`{`
	`"ErrorCode": 0,`
	`"ErrorMessage": null,`
	`"Data": [`
	`{`
	`"Name": "DISTMACHINE1_Simulation",`
	`"ComputerName": "DISTMACHINE1",`
	`"Status": "Stopped"`
	`},`
	`{`
	`"Name": "NPI212611749M1_Simulation",`
	`"ComputerName": "NPI212611749M1",`
	`"Status": "Stopped"`
	`},`
	`{`
	`"Name": "NPI212611749M1_Mqtt",`
	`"ComputerName": "NPI212611749M1",`
	`"Status": "Stopped"`
	`}`
	`]`
	`}`

SAMPLE cURL COMMAND	`curl -i -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>”`
	`https://<NLB DNS:9090>/historian-rest-api/v1/offlinecollectors`

Table 116. 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	`https://<NLB DNS:9090>/historian-rest-api/v1/datastores?dataStoreMask=`
SAMPLE QUERY PARAM GET URL	`https://<NLB DNS:9090>/historian-rest-api/v1/datastores?dataStoreMask=*`

SAMPLE RESPONSE	`{`
	`"ErrorCode": 0,`
	`"ErrorMessage": null,`
	`"Data": [`
	`{`
	`"Description": "The System Data Store.",`
	`"Id": "D3C23639-81CD-40F7-9CB0-37484FC5190D",`
	`"IsDefault": false,`
	`"IsSystem": true,`
	`"Name": "System",`
	`"NumberOfTags": 0,`

```
            "State": 2,

            "DHSStorageName": "System Storage",

            "StorageType": 0,

            "Links": [

                {

                    "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,
```

<table>
<tr><td></td><td><pre>
            "DHSStorageName": "xyz",

            "StorageType": 0,

            "Links": [

                {

                    "Rel": "self",

                    "Href": "/datastore/User"

                }

            ]

        }

    ]

}

   }

}
</pre></td></tr>
<tr><td>SAMPLE cURL COMMAND</td><td><pre>
curl -i -H "Accept: application/json" -H

 "Authorization: Bearer <TOKEN>”

https://<NLB

 DNS:9090>/historian-rest-api/v1/datastores?data

StoreMask=*
</pre></td></tr>
</table>

Table 117. Query Parameters

Parameter	Description	Required?	Values
`dataStoreMask`	The value of the data store mask.	No	String

Table 118. 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://<NLB DNS:9090>/historian-rest-api/v1/storage/datastores?storageName=`
SAMPLE QUERY PARAM GET URL	`https://<NLB DNS:9090>/historian-resr-api/v1/storage/datastores?storageName=xx`

SAMPLE RESPONSE	`{`
	`"ErrorCode": 0,`
	`"ErrorMessage": null,`
	`"Data": [`
	`{`
	`"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",
                    "Href": "/datastore/User"
                }
            ]
        }
    ]
}
```

SAMPLE cURL COMMAND

```
curl -i -H "Accept: application/json" -H
 "Authorization: Bearer <TOKEN>”
https://<NLB
 DNS:9090>/historian-rest-api/v1/storage/datasto
res?storageName=xx
```

Table 119. Query Parameters

Parameter	Description	Required?	Values
`storageName`	The name of the location whose data stores you want to view.	Yes	String

Table 120. 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://<NLB DNS:9090>/historian-rest-api/v1/datastoretostorage`

SAMPLE PATH PARAM GET URI	`https://<NLB DNS:9090>/historian-rest-api/v1/datastoretostorage`
	`Payload`
	`{`
	`"dataStoreName": "abc",`
	`"storageName": "storage1",`
	`"description": "test",`

	"isDefault": true }

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 “{ \”dataStoreName \”:\”name\”,\" storageName \": \"sname\",\"description \":\" des\",\" isDefault \":false}” -X POST https://<NLB DNS:9090>/historian-rest-api/v1/datastoretostor age

Table 121. Query Parameters

Parameter	Description	Required?	Values
`Payload`	Contains the details of the data store in the JSON format.	Yes	Multiple

Table 122. 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://<NLB`
	` DNS:9090>/historian-rest-api/v1/datastore`

SAMPLE URI	`https://<NLB`
	` DNS:9090>/historian-rest-api/v1/datastore`
	
	`Payload`
	
	`{`
	
	`"dataStoreName": "testdatastore"`
	
	`}`

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`
	` “{ \”dataStoreName \”:\”name\”}” -X DELETE`
	`https://<NLB`
	` DNS:9090>/historian-rest-api/v1/datastore`

Table 123. 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://<NLB DNS:9090>/historian-rest-api/v1/dataStore/<data store name>`

SAMPLE URI	`https://<NLB DNS:9090>/historian-rest-api/v1/dataStore/mirror1DS1`
	`Payload`
	`{`
	`"Description": "testing",`
	`"IsDefault": true,`
	`"IsSystem": false,`
	`"Name": "mirror1DS1",`
	`"NumberOfTags": 0,`
	`"State": 2,`
	`"DHSStorageName": "mirror1",`
	`"StorageType": 0,`
	`}`

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 “{ \” Description\”:\”des\”,\"IsDefault \": true, \” IsSystem \”:false, \” Name\”:\” mirror1DS1\”,\”NumberOfTags \”:0,\”State\”:2, \”DHSStorageName\”:\”mirror1\”,\”StorageType \”:0,\}” -X PUT https://<NLB DNS:9090>/historian-rest-api/v1/dataStore/mirror1DS1`

Table 124. 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 125. 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://<NLB DNS:9090>/historian-rest-api/v1/storage/<location name>`

SAMPLE URI

```
https://<NLB
 DNS:9090>/historian-rest-api/v1/storage/NPI2126
11749M1

Payload

{

            "StorageName": "NPI212611749M1",

            "StorageType": 0,

            "NumberOfDataStores": 5,

            "NumberOfArchivers": 1,

            "DataStores": [

                "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"] }

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 “{ \”StorageName\”:\”name\”,\"StorageType\": 0, \”NumberOfDataStores\”:5,\” NumberOfArchivers\”:1, \”IsDefault\”:true,\”ArchiverServices\”: [\"DataArchiver_NPI212611749M1\", \"DataArchiver_distamchine1"\"]}” -X PUT https://<NLB DNS:9090>/historian-rest-api/v1/storage/NPI2126 11749M1

Table 126. 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 127. 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 Systems

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://<NLB DNS:9090>/historian-rest-api/v1/serverproperties`
SAMPLE QUERY PARAM GET URL	`https://<NLB DNS:9090>/historian-rest-api/v1/serverproperties`

SAMPLE RESPONSE	`{`
	`    "ErrorCode": 0,`
	`    "ErrorMessage": null,`
	`    "Data": {`
	`        "Storages": [`

```
           {
               "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": "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"`
	`}`
	`]`
	`}`
	`}`

SAMPLE cURL COMMAND	`curl -i -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>” https://<NLB DNS:9090>/historian-rest-api/v1/serverproperties`

Table 128. 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	`https://<NLB DNS:9090>/historian-rest-api/v1/systemstats`

SAMPLE QUERY PARAM GET URL	`https://<NLB`
	` DNS:9090>/historian-rest-api/v1/systemstats`

SAMPLE RESPONSE	`{`
	`    "ErrorCode": 0,`
	`    "ErrorMessage": null,`
	`    "Data": {`
	`        "Utilization": {`
	`            "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": ""

        },

        "TotalCollectors": {

            "TotalCollectors": 1,

            "RunningCollectors": 1,

            "StoppedCollectors": 0,

            "UnknownCollectors": 0

        },

        "Licence": {

            "ActualDataStores": 3,

            "MaxDataStores": 200,

            "ActualTags": 0,

            "MaxTags": 2147483647,

            "ActualUsers": 0,
```

	`"MaxUsers": 1000`
	`}`
	`}`
	`}`

SAMPLE cURL COMMAND	`curl -i -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>" https://<NLB DNS:9090>/historian-rest-api/systemstats`

Table 129. 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
	`https://<NLB DNS:9090>/historian-rest-api/v1/performancecounter/perftagdata/PerfTag_AverageEventRate/-/-/starttime/endtime/interval`
	Receive Rate
	`https://<NLB DNS:9090>/historian-rest-api/v1/performancecounter/perftagdata/`

	`PerfTag_AverageReadRawRate/-/-/starttime/endtime/interval`
SAMPLE GET URI	`https://<NLB DNS:9090>/historian-rest-api/v1/performancecounter/perftagdata/PerfTag_AverageEventRate/-/-/2020-12-15T11:19:01.719Z/2020-12-15T12:19:01.719Z/360000`

SAMPLE RESPONSE	`{`
	`"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`
	`}`
	`]`
	`}`
	`]`
	`}`

SAMPLE cURL COMMAND	`curl -i -H "Accept: application/json" -H "Authorization: Bearer <TOKEN>” https://<NLB DNS:9090>/historian-rest-api/v1/performancecounter/perftagdata/PerfTag_AverageEventRate/-/-/2020-12-15T11:19:01.719Z/2020-12-15T12:19:01.719Z/360000`

Table 130. 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://<NLB DNS:9090>/v1/localopcservers/<machine name>`
SAMPLE QUERY PARAM GET URL	`http://<NLB DNS:9090>/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://<NLB DNS:9090>/historian-rest-api/v1/localopcservers/xyz`

Table 131. Query Parameters

Parameter	Description	Required?	Values
`machine name`	The machine name of the OPC server.	Yes	String

Table 132. 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://<NLB`
	`DNS:9090>/v1/localopcaeservers/<machine name>`

SAMPLE QUERY PARAM GET URL	`http://<NLB`
	`DNS:9090>/v1/localopcaeservers/<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://<NLB`
	`DNS:9090>/historian-rest-api/v1/localopcaeserve`
	`rs/abc`

Table 133. Query Parameters

Parameter	Description	Required?	Values
`machine name`	The machine name of the OPC Alarms and Events server.	Yes	String

Table 134. 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://<NLB`
	`DNS:9090>/v1/localopchdaservers/<machine name>`

SAMPLE QUERY PARAM GET URL	`http://<NLB`
	`DNS:9090>/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://<NLB`
	`DNS:9090>/historian-rest-api/v1/localopchdaserv`
	`ers/xyz`

Table 135. Query Parameters

Parameter	Description	Required?	Values
`machine name`	The machine name of the OPC HDA server.	Yes	String

Table 136. 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.

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.

This topic describes how to access the Historian REST APIs.

1. Access the following URL: `https://<NLB DNS>:9090/historian-rest-api/swagger-ui.html`
   The Swagger UI appears.
2. Select **Authorize**. The Available authorizations window appears.
3. 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. The default is **ih-CloudHistAdmin**.
password	Enter the web-based client login password.
client_id	Enter client id: **historian_rest_api_cloud**.
client_secret	Enter the web-based client login password.

4. You can now access the REST APIs for Historian.

Chapter 11. Using The Excel Add-In for Historian

About 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 and data — all with familiar Excel commands, macros, and computational techniques.
- You can create dynamic reports that you can share among users.

Excel Add-In Conventions

The Excel Add-In uses several conventions that allow you to take full advantage of the features of the Historian Excel Add-In:

- You can select tags and times either by cell references or by manually entering the values.
- You can select multiple statistics or attributes.
- 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.
- 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.

Installation

Install the Historian Excel Add-in Using the Installer

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

- Microsoft® Excel® 2021 (32 & 64 bit)
- Microsoft® Excel® 2019 (32 & 64 bit)
- Microsoft® Excel® 2016 (32 & 64 bit)
- Microsoft® Excel® 2013 (32 & 64 bit)

This topic describes how to install Excel Add-In using the installer. You can also install it at a command prompt *(on page 117)*.

1. Run the `InstallLauncher.exe` file. Contact the AWS support team for the installer.
2. Select **Historian Excel Add-in**.
   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 993)*.

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
2. Install Excel Add-in using the installer *(on page 117)* 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 during the installation. You can then use this template to install Excel Add-in at a command prompt on other machines.

This topic describes how to install the Excel Addin for Historian at a command prompt. You can also install it using the installer *(on page 117)*.

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 993)*.

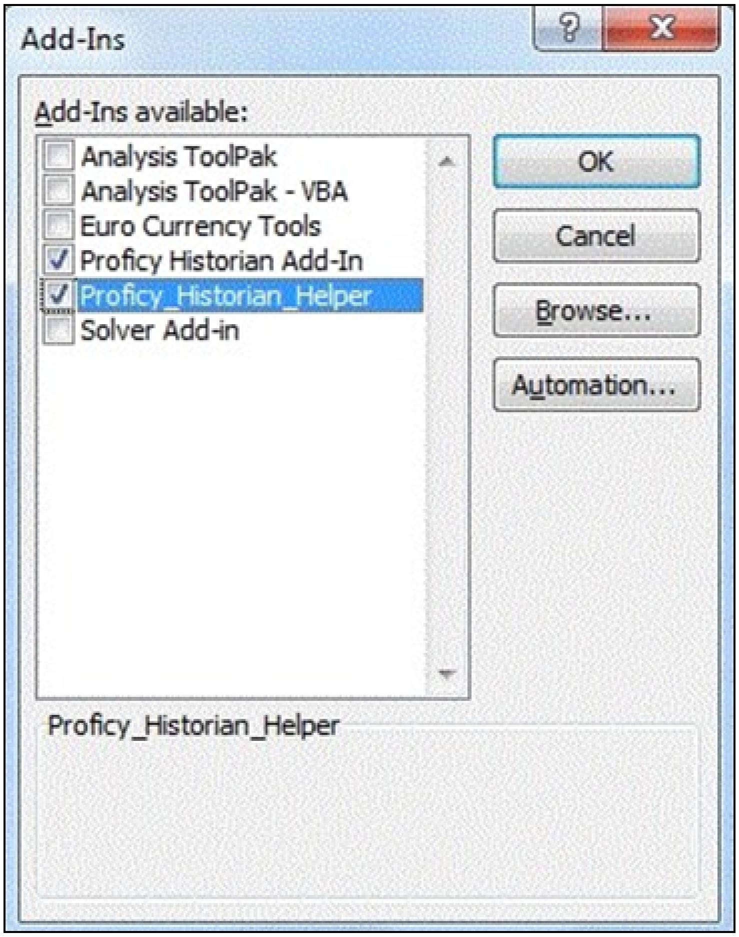
Activate Excel Add-In

Install Excel Add-In *(on page 117)*.

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**.

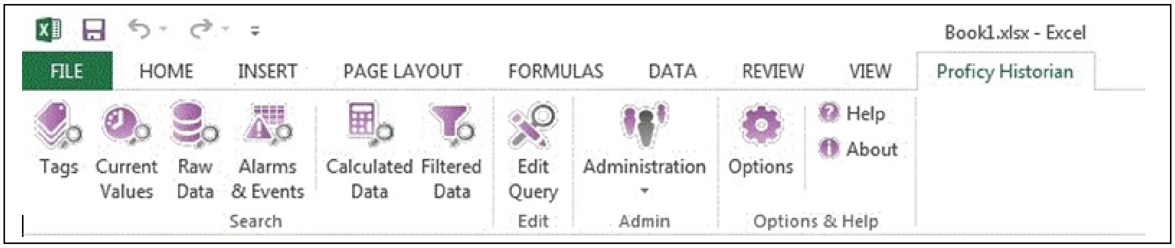
   The **Add-Ins** window appears.

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.



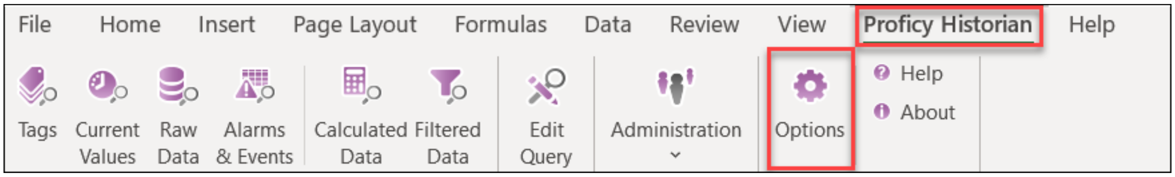
Connect the Excel Add-in with a Historian server *(on page 994)*.

Connect the Excel Add-in with the Historian Server

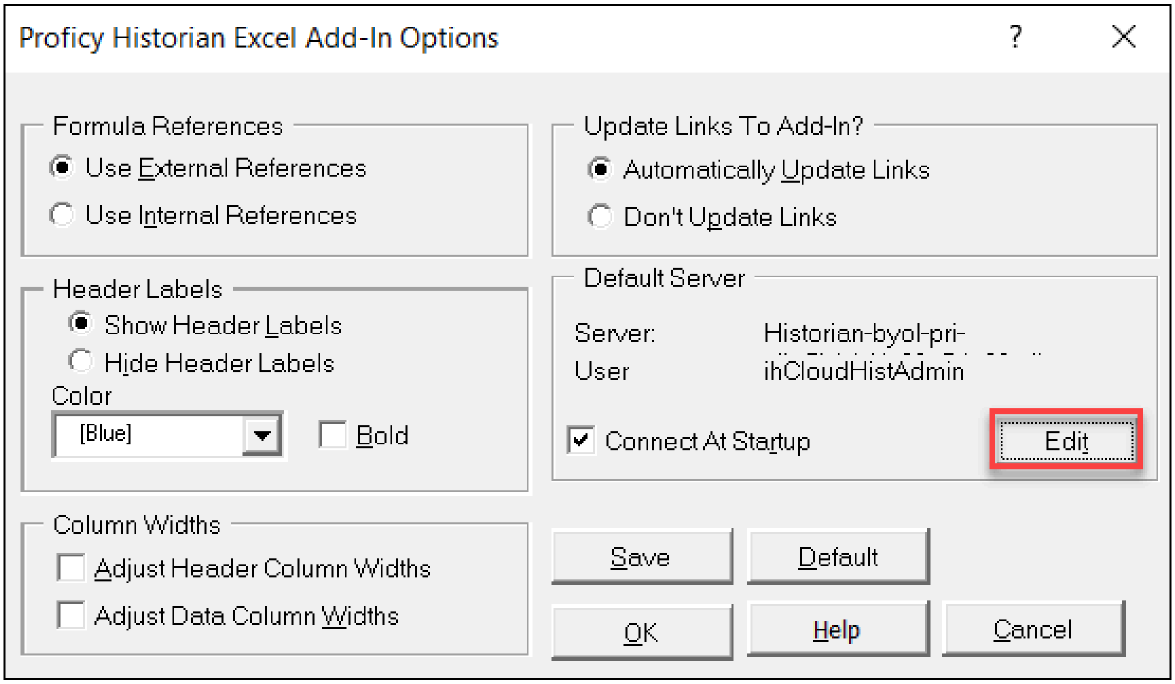
Activate the Excel Add-in for Historian *(on page 993)*.

This topic describes how to add an NLB DNS in the Excel Add-in for Historian. You can add multiple NLBs; however, you can connect with a single NLB at a time.

1. Open an Excel worksheet.
2. Select **Proficy Historian > Options**.



The **Proficy Historian Excel Add-in Options** window appears.
3. In the **Default Server** section, select **Edit**.



Proficy Historian Excel Add-In Options

Formula References
- ● Use External References
- ○ Use Internal References

Header Labels
- ● Show Header Labels
- ○ Hide Header Labels

Color: [Blue] □ Bold

Column Widths
- □ Adjust Header Column Widths
- □ Adjust Data Column Widths

Update Links To Add-In?
- ● Automatically Update Links
- ○ Don't Update Links

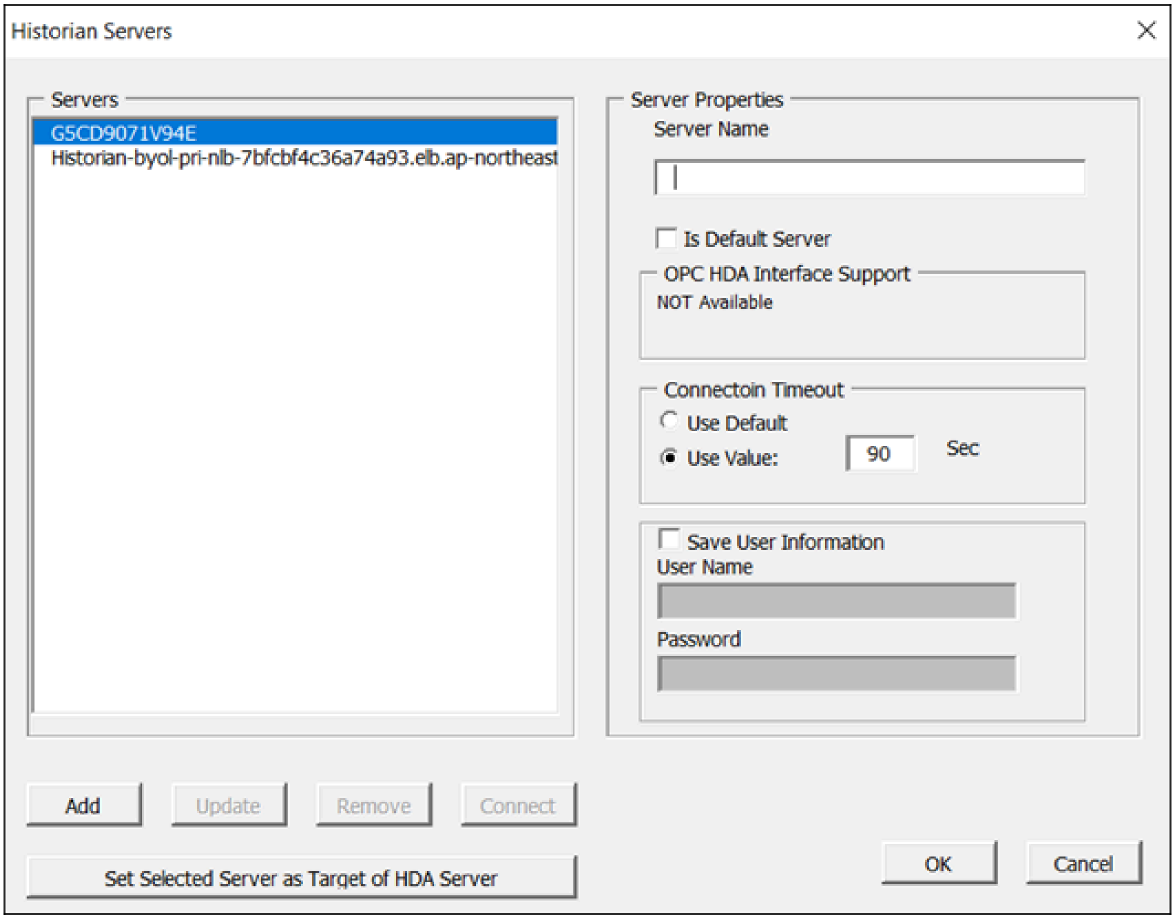
Default Server

Server: Historian-byol-pri-

User ihCloudHistAdmin

☑ Connect At Startup

The **Historian Servers** window appears.



Historian Servers

Servers
- G5CD9071V94E
- Historian-byol-pri-nlb-7bfcbf4c36a74a93.elb.ap-northeast

Server Properties

Server Name:

□ Is Default Server

OPC HDA Interface Support: NOT Available

Connectoin Timeout
- ○ Use Default
- ● Use Value: 90 Sec

□ Save User Information

User Name:

Password:

4. Enter values as described in the following table.

Field	Description
Server Name	Enter the Amazon Network Load Balancer (NLB) DNS. **Tip:** To find the NLB DNS: a. Access the EKS cluster on which you have deployed Proficy Historian for AWS. b. Access the EC2 instance. c. In the navigation pane, under **Load Balancing**, select **Load Balancers**. d. Select the load balancer for which you want to find the DNS. e. In the **Description** section, copy the DNS name.
Is Default Server	Select this check box if you want to set this Historian server as the default one.
Connection Timeout	Specify the time after which the connection will time out.
User Name	Enter the username to connect to Proficy Historian for AWS.
Password	Enter the password to connect to Proficy Historian for AWS. **Tip:** This is the value you entered in the **Password** field under **Proficy Authentication Configuration** when you created the stack.

5. Select **OK**.

   The Excel Addin is connected to the Historian server. You can now query data or manage tags.

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 998)* and calculated data *(on page 1001)*.

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 1011)*.
   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:

Field	Description
	◦ **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. ◦ **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. ◦ **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 1003)*.
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.

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 1011)*.
   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.

Field	Description
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 1003)*.
Sampling Type	Select the sampling type.
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: ◦ **By Interval**: Using this option, you can query the data for a specific interval. 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. ◦ **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.

Query 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 1011)*.
   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 1003)*.
Sampling Type	Select the sampling type.
Calculation	Select a calculation algorithm. This field is enabled only if you select **Calculated Sampling** in the **Sampling Type** field.

Field	Description
Sampling Interval	Select one of the following options: ◦ **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. ◦ **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.

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-select 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`	The `ONLYGOOD` modifier excludes bad and uncertain data quality values from retrieval and calculations. 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 with those cases.
`INCLUDEREPLACED`	Normally,when you query raw data, any values that have been replaced with a different value for the same timestamp are not returned. 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. This modifier is only useful with the rawbytime or rawbynumber retrieval. Do not use it with any other sampling or calculation mode.
`INCLUDEDELETED`	The `INCLUDEDELETED` modifier 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.

Query Modifier	Results
	This modifier is only useful with the rawbytime or rawbynumber retrieval. Do not use it with any other sampling or calculation mode.
`ONLYIFCONNECTED/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
`ONLYRAW`	The `ONLYRAW` modifier 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. 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. Use this query modifier with calculation modes only, not with raw or sampled retrieval like interpolated 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. 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
`INCLUDEBAD`	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. You can use `INCLUDEBAD` with any sampling or calculation mode.
`FILTERINCLUDEBAD`	Normally, while filtering, we use only good data quality values. When we use `FILTERINCLUDEBAD`, the bad data quality values are considered when filtering to determine time ranges. This query modifier is not always recommended.
`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.
`HONORENDTIME`	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 offpage. 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 (backwards or forwards), the end time must be older than the start time for a backwards direction or later than the start time for a forwards direction. Use this query modifier only with the `RawByNumber` sampling mode.
`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. Using this query modifier is recommended when: • The time interval is great than 1 minute. • The collection interval is greater than 1 second.

Query Modifier	Results
	• 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.

Export 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 or data, 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. Select **Administration > Export Raw Data** from the **Historian** menu.
   The **Export Data from Historian** window appears.
2. 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.
3. 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.

4. Optionally, you can select the tag name from the **Advance Tag Search** window.
   See Search for a Tag (Advanced) *(on page 1011)*
5. 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 1003)*.
6. 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 1031)*.

7. In the **Sampling Type** section, select a type from the drop-down list.
8. 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.
9. 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.
10. 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.
11. 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.
12. In the **Export Options** section, select one of three options:
    - **To New Worksheet**
    - **To CSV File** or
    - **To XML File**
13. 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.
14. Select **OK** to initiate the export. Select **Cancel** to abort the operation and close the window.

Import 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.

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, and current or read-only status). 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. **Tip:** To return details for more than one item, specify a sub-string 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 `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. **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 1011)*.

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 1010)*.

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 1038)*.
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**.

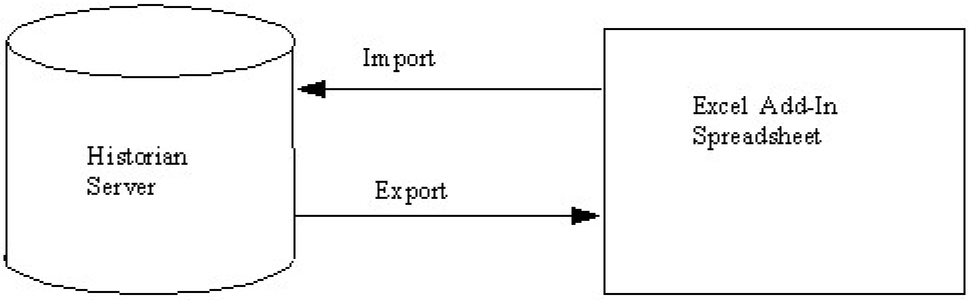
       The criteria list is loaded to the **Advanced Tag Search** window.

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 1013)* in bulk, and then import them *(on page 1014)*.



> **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. **Note:** You cannot export multiple tags when tagnames are read from multiple cells. If you specify a range of tag-names to read from multiple cells in the **Tag 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 1014)*.

   > **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 1013)*.

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).

> **Note:**
> Do not add or update the spare configurations as the data may get corrupted or overwritten. For example, the **Spare 5** field is used by the Server-to-Server collector 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 1012)* 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.

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 any formula links that may be broken due to differences among users in Microsoft Office installation. The default setting enables this feature. 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. 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**

Field	Description
	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 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.

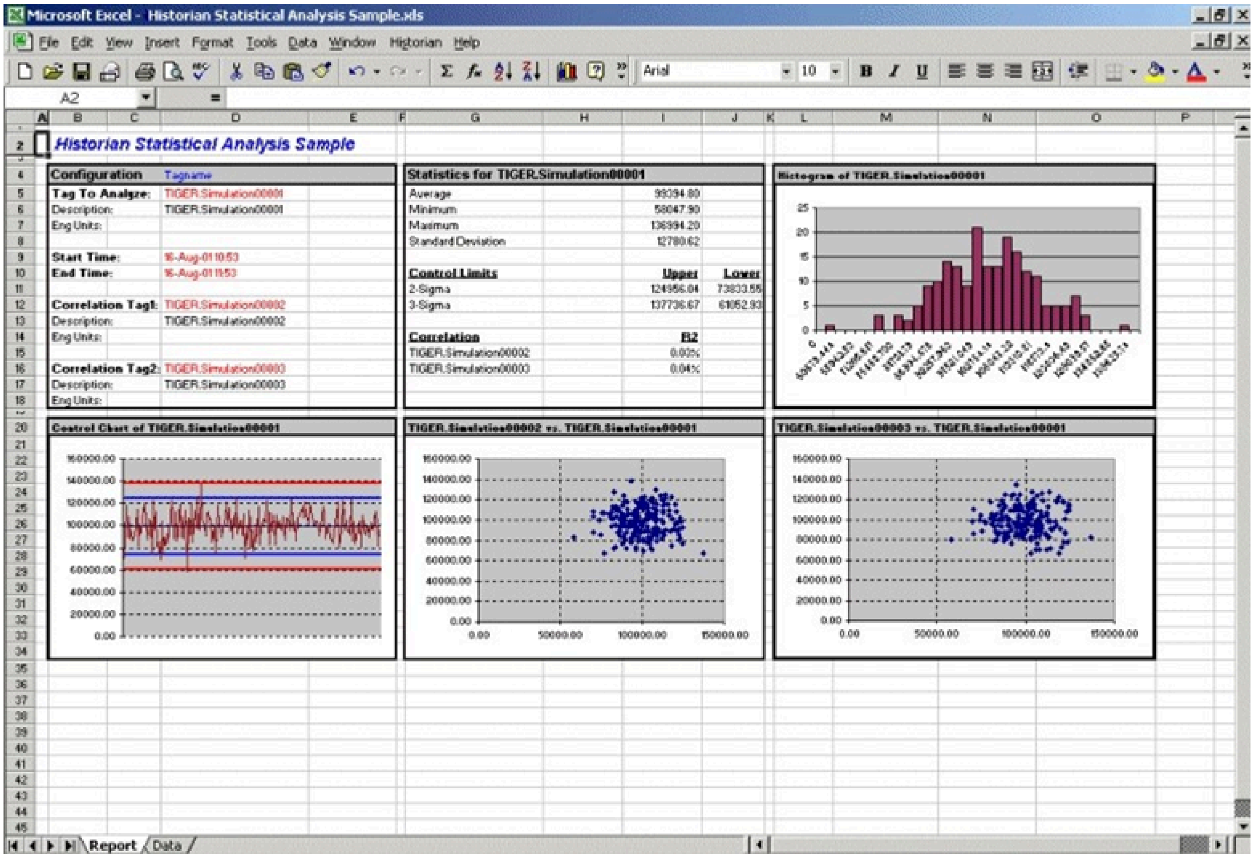
In addition, when you create an instance of the Simulation collector, it prompts you for the number of simulation tags it should create (but you must still add the tags for collection using one of the two methods above). The default is 1000. Do not enter a value less than 30.

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.

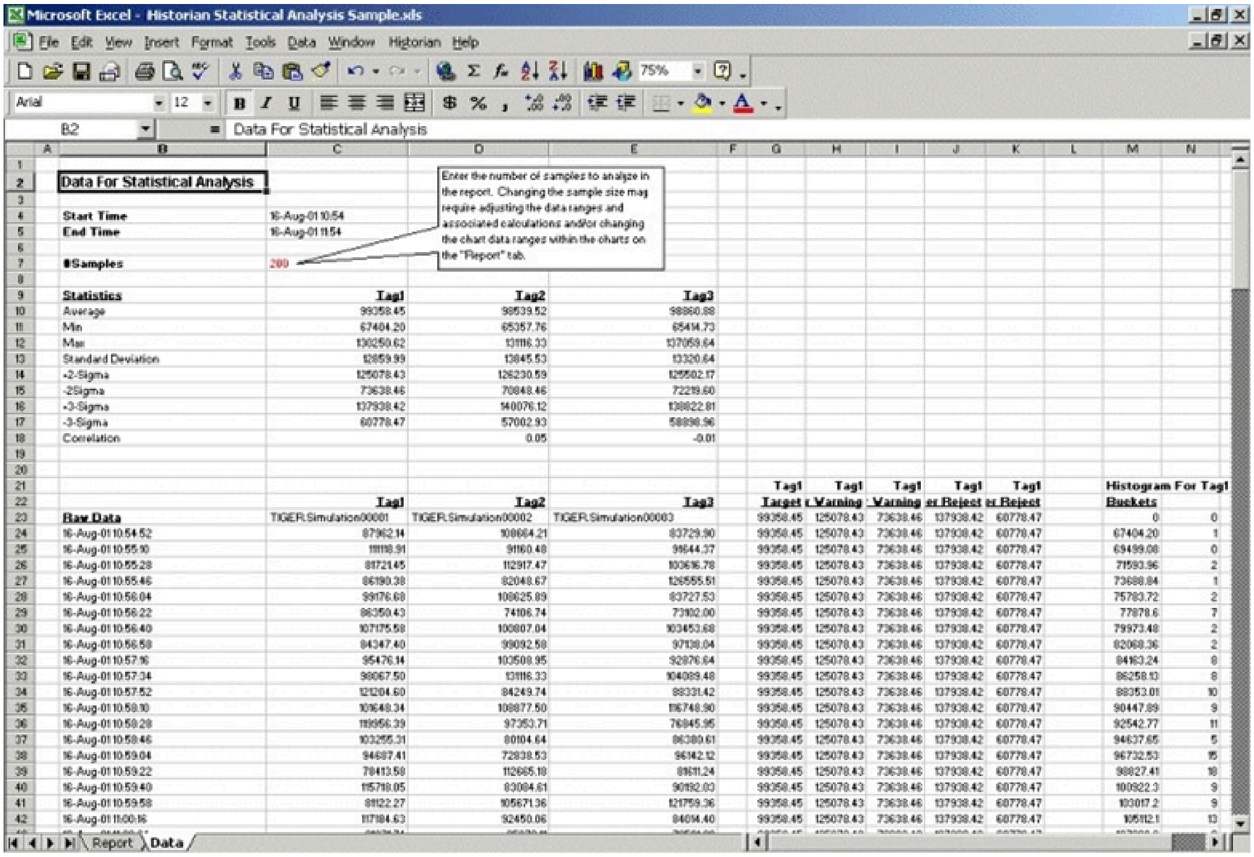
The chart at the lower left is a plot of the main variable vs. time with sigma control limits indicated by the straight lines. The two charts to the right are scatter diagrams that show the correlation between the main variable and two other variables. The chart at the top right is a histogram of data values of the main variable that shows how the data points are distributed.


Microsoft Excel - Historian Statistical Analysis Sample.xls
File Edit View Insert Format Tools Data Window Historian Help
Arial
10
A2
=
Historian Statistical Analysis Sample
Histogram of TIGER.Simulation00001
25
20
15
10
5
0
Control Chart of TIGER.Simulation00001
160000.00
140000.00
120000.00
100000.00
80000.00
60000.00
40000.00
20000.00
0.00
TIGER.Simulation00002 vs. TIGER.Simulation00001
TIGER.Simulation00003 vs. TIGER.Simulation00001
0.00
50000.00
100000.00
150000.00
Report
Data

Configuration	Tagname
Tag To Analyze:	TIGER.Simulation00001
Description:	TIGER.Simulation00001
Eng Units:	
Start Time:	16-Aug-01 10:53
End Time:	16-Aug-01 11:53
Correlation Tag1:	TIGER.Simulation00002
Description:	TIGER.Simulation00002
Eng Units:	
Correlation Tag2:	TIGER.Simulation00003
Description:	TIGER.Simulation00003
Eng Units:	

Statistics for TIGER.Simulation00001		
Average	99394.80	
Minimum	58047.90	
Maximum	136994.20	
Standard Deviation	12780.62	
Control Limits	**Upper**	**Lower**
2-Sigma	124956.04	73833.55
3-Sigma	137736.67	61052.93
Correlation	**R2**	
TIGER.Simulation00002	0.03%	
TIGER.Simulation00003	0.04%	

The following figure shows the worksheet associated with the sample report that contains the data used to generate the report.

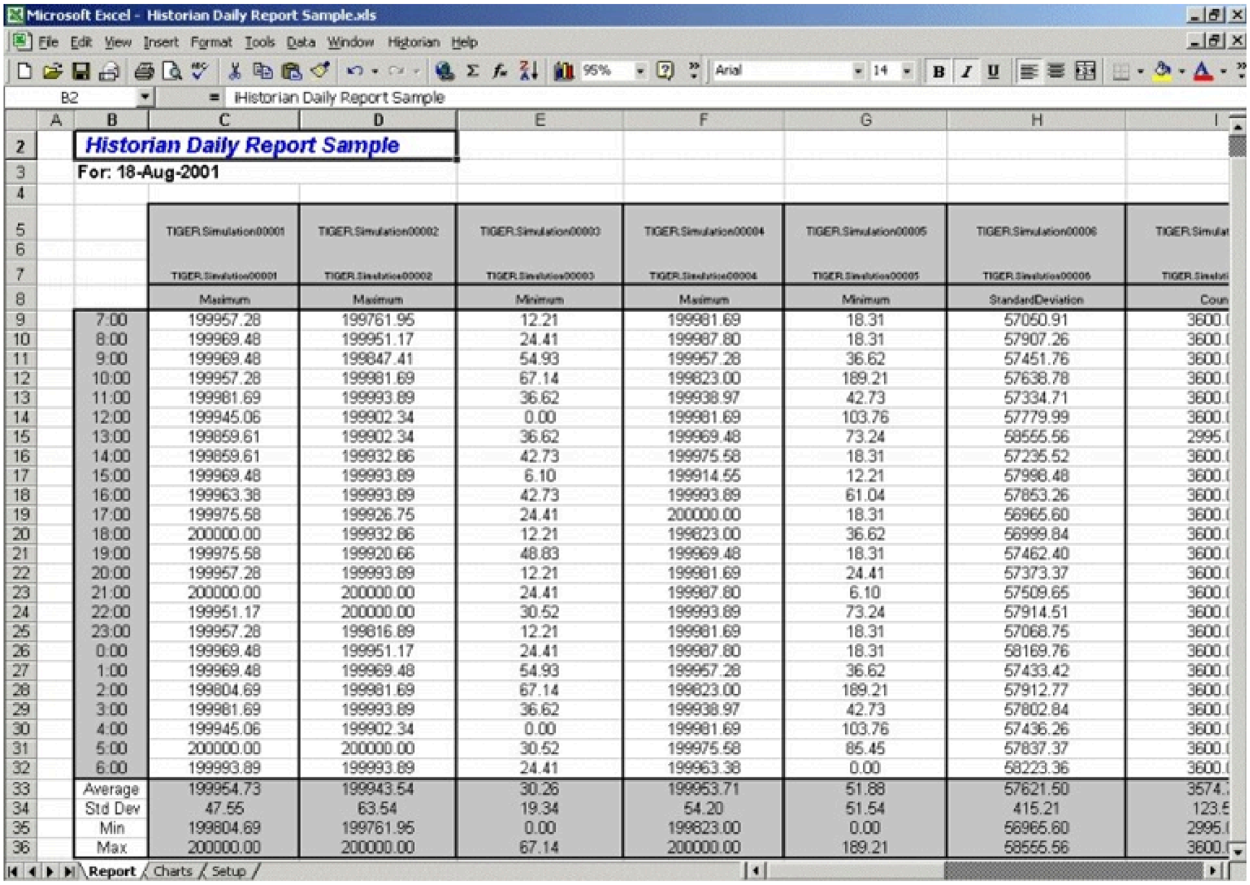

Microsoft Excel - Historian Statistical Analysis Sample.xls
File Edit View Insert Format Tools Data Window Historian Help
75%
Arial
12
B2
Data For Statistical Analysis
Report
Data

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1														
2		Data For Statistical Analysis												
3														
4		**Start Time**	16-Aug-01 10:54											
5		**End Time**	16-Aug-01 11:54											
6														
7		**#Samples**	200											
8														
9		**Statistics**	**Tag1**	**Tag2**	**Tag3**									
10		Average	99358.45	98539.52	98860.88									
11		Min	67404.20	65357.76	65414.73									
12		Max	130250.62	131116.33	137059.64									
13		Standard Deviation	12859.99	13845.53	13320.64									
14		+2-Sigma	125078.43	126230.59	125502.17									
15		-2Sigma	73638.46	70848.46	72219.60									
16		+3-Sigma	137938.42	140076.12	138822.81									
17		-3-Sigma	60778.47	57002.93	58898.96									
18		Correlation		0.05	-0.01									
19														
20														
21							**Tag1**	**Tag1**	**Tag1**	**Tag1**	**Tag1**		**Histogram For Tag1**	
22			**Tag1**	**Tag2**	**Tag3**		**Target**	**r Warning**	**Warning**	**er Reject**	**er Reject**		**Buckets**	
23		**Raw Data**	TIGER.Simulation00001	TIGER.Simulation00002	TIGER.Simulation00003		99358.45	125078.43	73638.46	137938.42	60778.47		0	0
24		16-Aug-01 10:54:52	87962.14	108664.21	83729.90		99358.45	125078.43	73638.46	137938.42	60778.47		67404.20	1
25		16-Aug-01 10:55:10	111118.91	91160.48	98644.37		99358.45	125078.43	73638.46	137938.42	60778.47		69499.08	0
26		16-Aug-01 10:55:28	81721.45	112917.47	100616.78		99358.45	125078.43	73638.46	137938.42	60778.47		71593.96	2
27		16-Aug-01 10:55:46	86190.38	82048.67	126555.51		99358.45	125078.43	73638.46	137938.42	60778.47		73688.84	1
28		16-Aug-01 10:56:04	99176.68	108625.89	83727.53		99358.45	125078.43	73638.46	137938.42	60778.47		75783.72	2
29		16-Aug-01 10:56:22	86350.43	74106.74	73102.00		99358.45	125078.43	73638.46	137938.42	60778.47		77878.6	7
30		16-Aug-01 10:56:40	107175.58	100807.04	103453.68		99358.45	125078.43	73638.46	137938.42	60778.47		79973.48	2
31		16-Aug-01 10:56:58	84347.40	99092.58	97138.04		99358.45	125078.43	73638.46	137938.42	60778.47		82068.36	2
32		16-Aug-01 10:57:16	95476.14	103508.95	92876.64		99358.45	125078.43	73638.46	137938.42	60778.47		84163.24	8
33		16-Aug-01 10:57:34	98067.50	131116.33	104089.48		99358.45	125078.43	73638.46	137938.42	60778.47		86258.13	8
34		16-Aug-01 10:57:52	121204.60	84249.74	88331.42		99358.45	125078.43	73638.46	137938.42	60778.47		88353.01	10
35		16-Aug-01 10:58:10	101648.34	108877.50	116748.90		99358.45	125078.43	73638.46	137938.42	60778.47		90447.89	9
36		16-Aug-01 10:58:28	119956.39	97353.71	76845.95		99358.45	125078.43	73638.46	137938.42	60778.47		92542.77	11
37		16-Aug-01 10:58:46	103255.31	80104.64	86380.61		99358.45	125078.43	73638.46	137938.42	60778.47		94637.65	5
38		16-Aug-01 10:59:04	94687.41	72838.53	96142.12		99358.45	125078.43	73638.46	137938.42	60778.47		96732.53	15
39		16-Aug-01 10:59:22	78413.58	112665.18	89611.24		99358.45	125078.43	73638.46	137938.42	60778.47		98827.41	18
40		16-Aug-01 10:59:40	115718.05	83084.61	90192.03		99358.45	125078.43	73638.46	137938.42	60778.47		100922.3	9
41		16-Aug-01 10:59:58	89122.27	105671.36	121759.36		99358.45	125078.43	73638.46	137938.42	60778.47		103017.2	9
42		16-Aug-01 11:00:16	117184.63	92450.06	84014.40		99358.45	125078.43	73638.46	137938.42	60778.47		105112.1	13

Enter the number of samples to analyze in the report. Changing the sample size may require adjusting the data ranges and associated calculations and/or changing the chart data ranges within the charts on the "Report" tab.

Daily Performance Sample Report

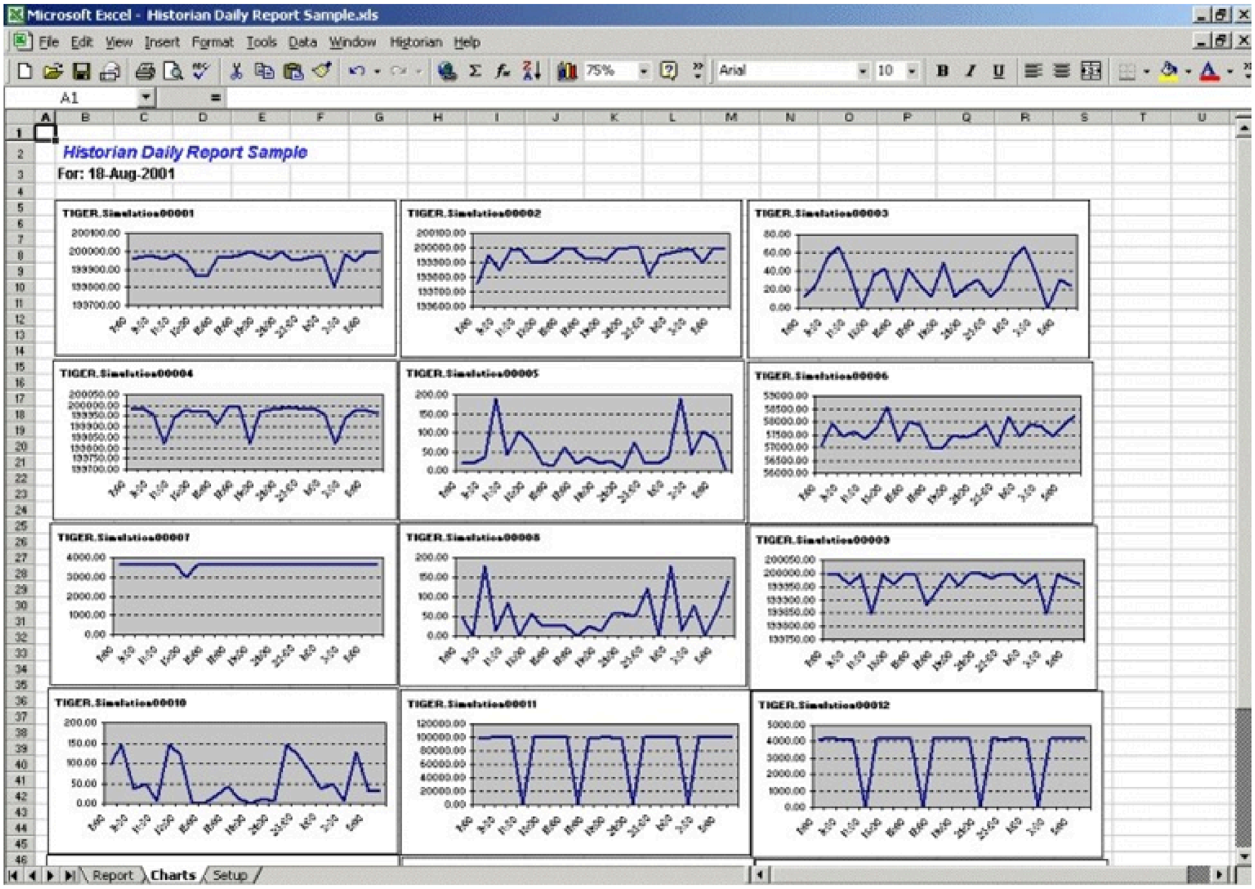
This sample report shows how the measured values and selected statistical properties of specified tags have varied in the last 24 hours. This sample is an example of a typical daily performance report in an industrial plant.


Microsoft Excel - Historian Daily Report Sample.xls
File Edit View Insert Format Tools Data Window Historian Help
95%
Arial
14
B2

= Historian Daily Report Sample

	A	B	C	D	E	F	G	H	I
2		**Historian Daily Report Sample**							
3		For: 18-Aug-2001							
4									
5			TIGER.Simulation00001	TIGER.Simulation00002	TIGER.Simulation00003	TIGER.Simulation00004	TIGER.Simulation00005	TIGER.Simulation00006	TIGER.Simulat
6									
7			TIGER.Simulation00001	TIGER.Simulation00002	TIGER.Simulation00003	TIGER.Simulation00004	TIGER.Simulation00005	TIGER.Simulation00006	TIGER.Simulati
8			Maximum	Maximum	Minimum	Maximum	Minimum	StandardDeviation	Coun
9		7:00	199957.28	199761.95	12.21	199981.69	18.31	57050.91	3600.
10		8:00	199969.48	199951.17	24.41	199987.80	18.31	57907.26	3600.
11		9:00	199969.48	199847.41	54.93	199957.28	36.62	57451.76	3600.
12		10:00	199957.28	199981.69	67.14	199823.00	189.21	57638.78	3600.
13		11:00	199981.69	199993.89	36.62	199938.97	42.73	57334.71	3600.
14		12:00	199945.06	199902.34	0.00	199981.69	103.76	57779.99	3600.
15		13:00	199859.61	199902.34	36.62	199969.48	73.24	58555.56	2995.
16		14:00	199859.61	199932.86	42.73	199975.58	18.31	57235.52	3600.
17		15:00	199969.48	199993.89	6.10	199914.55	12.21	57998.48	3600.
18		16:00	199963.38	199993.89	42.73	199993.89	61.04	57853.26	3600.
19		17:00	199975.58	199926.75	24.41	200000.00	18.31	56965.60	3600.
20		18:00	200000.00	199932.86	12.21	199823.00	36.62	56999.84	3600.
21		19:00	199975.58	199920.66	48.83	199969.48	18.31	57462.40	3600.
22		20:00	199957.28	199993.89	12.21	199981.69	24.41	57373.37	3600.
23		21:00	200000.00	200000.00	24.41	199987.80	6.10	57509.65	3600.
24		22:00	199951.17	200000.00	30.52	199993.89	73.24	57914.51	3600.
25		23:00	199957.28	199816.89	12.21	199981.69	18.31	57068.75	3600.
26		0:00	199969.48	199951.17	24.41	199987.80	18.31	58169.76	3600.
27		1:00	199969.48	199969.48	54.93	199957.28	36.62	57433.42	3600.
28		2:00	199804.69	199981.69	67.14	199823.00	189.21	57912.77	3600.
29		3:00	199981.69	199993.89	36.62	199938.97	42.73	57802.84	3600.
30		4:00	199945.06	199902.34	0.00	199981.69	103.76	57436.26	3600.
31		5:00	200000.00	200000.00	30.52	199975.58	85.45	57837.37	3600.
32		6:00	199993.89	199993.89	24.41	199963.38	0.00	58223.36	3600.
33		Average	199954.73	199943.54	30.26	199953.71	51.88	57621.50	3574.
34		Std Dev	47.55	63.54	19.34	54.20	51.54	415.21	123.5
35		Min	199804.69	199761.95	0.00	199823.00	0.00	56965.60	2995.
36		Max	200000.00	200000.00	67.14	200000.00	189.21	58555.56	3600.

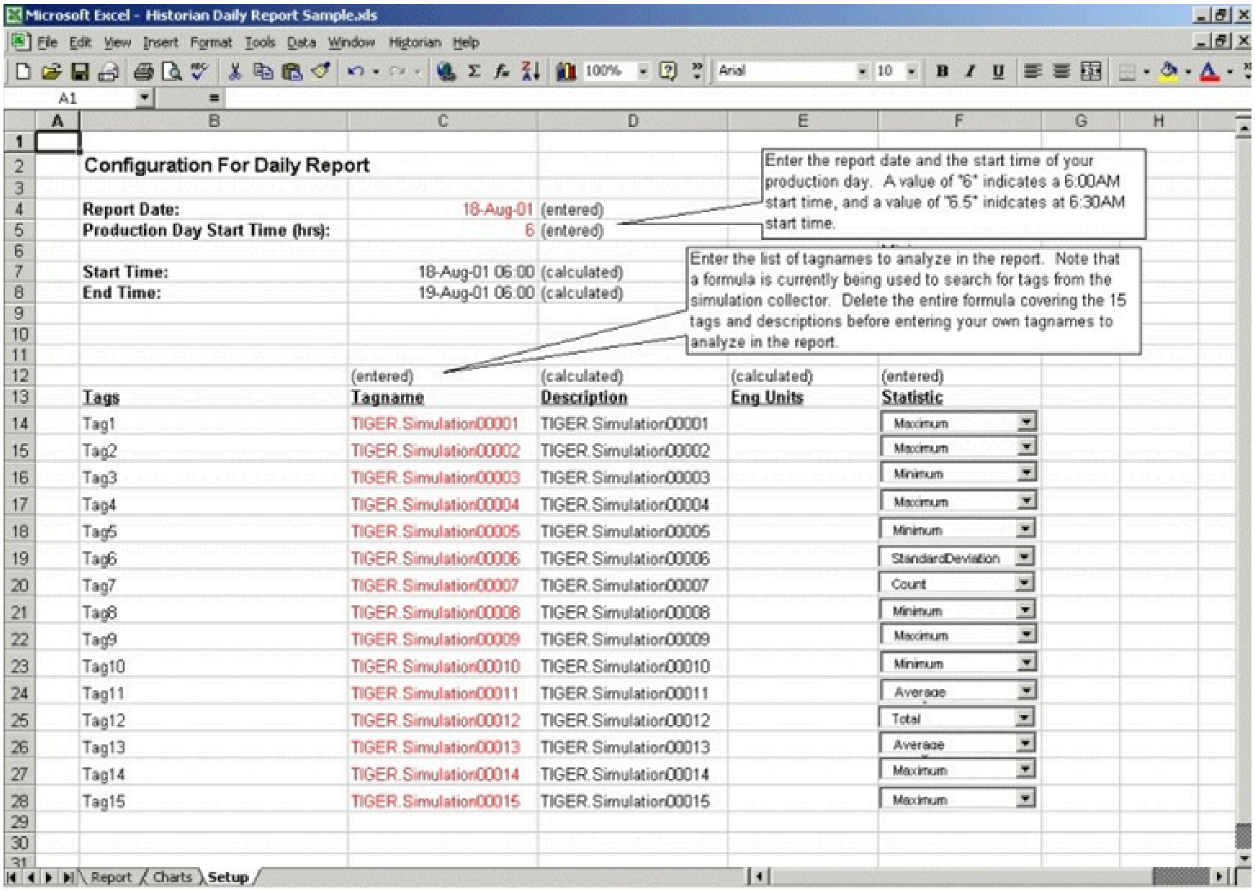
The report shown in the following image is a collection of chart plots of the data displayed in the report of the previous image.



Historian Daily Report Sample

For: 18-Aug-2001

The following figure shows the worksheet used to set up the Daily Sample Report. Edit the worksheet to adapt this report to your application.


Microsoft Excel - Historian Daily Report Sample.xls
File Edit View Insert Format Tools Data Window Historian Help
100%
Arial
10
A1
Report
Charts
Setup

Configuration For Daily Report

Report Date: 18-Aug-01 (entered)
Production Day Start Time (hrs): 6 (entered)

Start Time: 18-Aug-01 06:00 (calculated)
End Time: 19-Aug-01 06:00 (calculated)

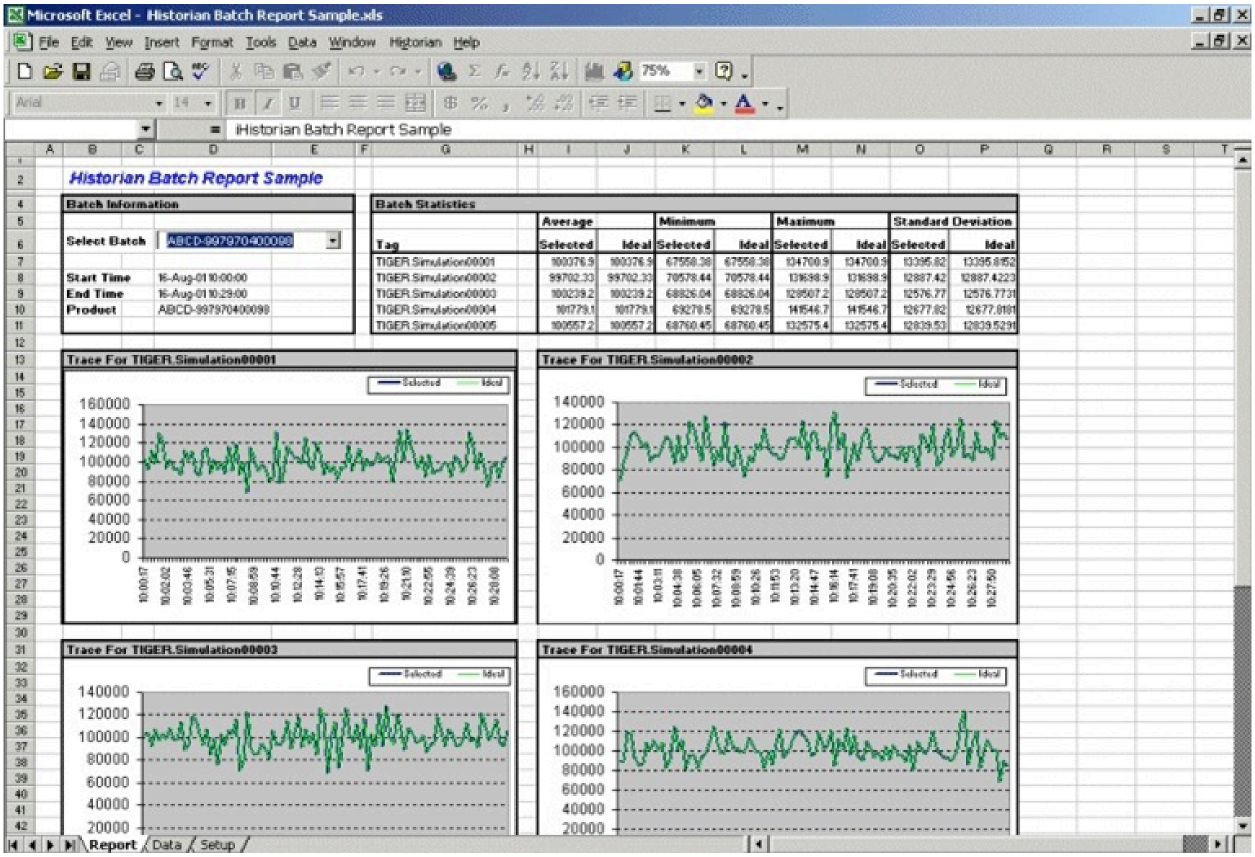
Enter the report date and the start time of your production day. A value of "6" indicates a 6:00AM start time, and a value of "6.5" inidcates at 6:30AM start time.

Enter the list of tagnames to analyze in the report. Note that a formula is currently being used to search for tags from the simulation collector. Delete the entire formula covering the 15 tags and descriptions before entering your own tagnames to analyze in the report.

	(entered)	(calculated)	(calculated)	(entered)
Tags	**Tagname**	**Description**	**Eng Units**	**Statistic**
Tag1	TIGER.Simulation00001	TIGER.Simulation00001		Maximum
Tag2	TIGER.Simulation00002	TIGER.Simulation00002		Maximum
Tag3	TIGER.Simulation00003	TIGER.Simulation00003		Minimum
Tag4	TIGER.Simulation00004	TIGER.Simulation00004		Maximum
Tag5	TIGER.Simulation00005	TIGER.Simulation00005		Minimum
Tag6	TIGER.Simulation00006	TIGER.Simulation00006		StandardDeviation
Tag7	TIGER.Simulation00007	TIGER.Simulation00007		Count
Tag8	TIGER.Simulation00008	TIGER.Simulation00008		Minimum
Tag9	TIGER.Simulation00009	TIGER.Simulation00009		Maximum
Tag10	TIGER.Simulation00010	TIGER.Simulation00010		Minimum
Tag11	TIGER.Simulation00011	TIGER.Simulation00011		Average
Tag12	TIGER.Simulation00012	TIGER.Simulation00012		Total
Tag13	TIGER.Simulation00013	TIGER.Simulation00013		Average
Tag14	TIGER.Simulation00014	TIGER.Simulation00014		Maximum
Tag15	TIGER.Simulation00015	TIGER.Simulation00015		Maximum

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.


Microsoft Excel - Historian Batch Report Sample.xls
File Edit View Insert Format Tools Data Window Historian Help
Arial 14 75%
Report / Data / Setup

iHistorian Batch Report Sample

Historian Batch Report Sample

Batch Information

Select Batch	ABCD-997970400098
Start Time	16-Aug-01 10:00:00
End Time	16-Aug-01 10:29:00
Product	ABCD-997970400098

Batch Statistics

	Average		Minimum		Maximum		Standard Deviation	
Tag	Selected	Ideal	Selected	Ideal	Selected	Ideal	Selected	Ideal
TIGER.Simulation00001	100376.9	100376.9	67558.38	67558.38	134700.9	134700.9	13395.82	13395.8152
TIGER.Simulation00002	99702.33	99702.33	70578.44	70578.44	131698.9	131698.9	12887.42	12887.4223
TIGER.Simulation00003	100239.2	100239.2	68826.04	68826.04	128507.2	128507.2	12576.77	12576.7731
TIGER.Simulation00004	101779.1	101779.1	69278.5	69278.5	141546.7	141546.7	12677.82	12677.8181
TIGER.Simulation00005	100557.2	100557.2	68760.45	68760.45	132575.4	132575.4	12839.53	12839.5291

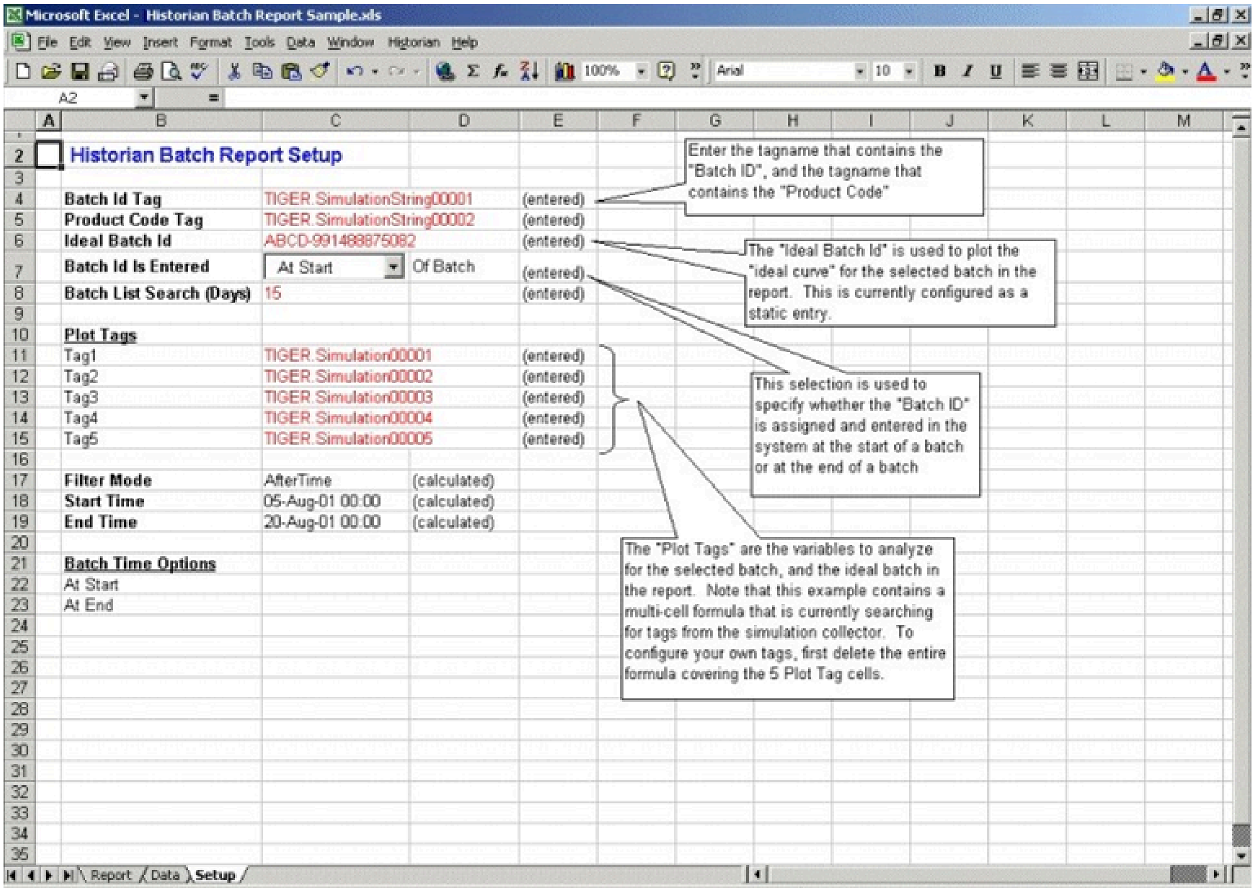
Trace For TIGER.Simulation00001

Trace For TIGER.Simulation00002

Trace For TIGER.Simulation00003

Trace For TIGER.Simulation00004

This is the configuration worksheet used to generate the report shown in the previous image. Modify this worksheet to adapt it to your requirements.


Microsoft Excel - Historian Batch Report Sample.xls
File Edit View Insert Format Tools Data Window Historian Help
100%
Arial
10
A2
Report
Data
Setup

Historian Batch Report Setup

Batch Id Tag	TIGER.SimulationString00001		(entered)
Product Code Tag	TIGER.SimulationString00002		(entered)
Ideal Batch Id	ABCD-991488875082		(entered)
Batch Id Is Entered	At Start	Of Batch	(entered)
Batch List Search (Days)	15		(entered)
Plot Tags			
Tag1	TIGER.Simulation00001		(entered)
Tag2	TIGER.Simulation00002		(entered)
Tag3	TIGER.Simulation00003		(entered)
Tag4	TIGER.Simulation00004		(entered)
Tag5	TIGER.Simulation00005		(entered)
Filter Mode	AfterTime	(calculated)	
Start Time	05-Aug-01 00:00	(calculated)	
End Time	20-Aug-01 00:00	(calculated)	
Batch Time Options			
At Start			
At End			

Enter the tagname that contains the "Batch ID", and the tagname that contains the "Product Code"

The "Ideal Batch Id" is used to plot the "ideal curve" for the selected batch in the report. This is currently configured as a static entry.

This selection is used to specify whether the "Batch ID" is assigned and entered in the system at the start of a batch or at the end of a batch

The "Plot Tags" are the variables to analyze for the selected batch, and the ideal batch in the report. Note that this example contains a multi-cell formula that is currently searching for tags from the simulation collector. To configure your own tags, first delete the entire formula covering the 5 Plot Tag cells.

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,pFilterTag,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.
`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: • `Equal`: Filter condition is True when the `FilterTag` is equal to the comparison value. • `EqualFirst`: Filter condition is True when the `FilterTag` is equal to the first comparison value. • `EqualLast`: Filter condition is True when the `FilterTag` is equal to the last comparison value. • `NotEqual`: Filter condition is True when the `FilterTag` is NOT equal to the comparison value. • `LessThan`: Filter condition is True when the `FilterTag` is less than the comparison value. • `GreaterThan`: Filter condition is True when the `FilterTag` is greater than the comparison value. • `LessThanEqual`: Filter condition is True when the `FilterTag` is less than or equal to the comparison value. • `GreaterThanEqual`: Filter condition is True when the `FilterTag` is greater than or equal to the comparison value. • `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. • `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. • `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. • `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: • `AND` condition • `OR` condition • Combination of both `AND` and `OR` 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: • `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 the 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). • 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.
`pFilterTag`	The single tagname used when applying the filter criteria.

Parameter	Description
`pNumberOfSamples`	Number of samples from the archive to retrieve. 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 used by the query.
`p-SearchText`	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: • Connections • Configuration

Parameter	Description
	• General • Services • Performance • 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
`Filter Tag`	The single tag name used when applying the filter criteria. **Note:** 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.
`Filter Expression`	An expression that includes one or more filter conditions. The types of conditions used are: • AND Condition • OR Condition • Combination of both AND and OR FilterExpression can be used instead of FilterTag, FilterComparisonMode and FilterValue parameters. There is no maximum length for a filter expression.
`Filter Mode`	The type of time filter:

Parameters	Description
	• **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). 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.
`Filter Comparison Mode`	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.

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. - **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. 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. **Note:** Filter Comparison Mode is only used if Filter Tag is filled in.
`FilterComparison Value`	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). **Note:** 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 minutes 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.

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. **Note:** 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. **Note:** Do not use wildcards in this field.
Comment	Comments that is applied to the tag. **Note:** Do not use wildcards in this field.
Source Address	The address for the selected tag in the data store.

Criteria	Description
	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.

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 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 12. Using the 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.

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:

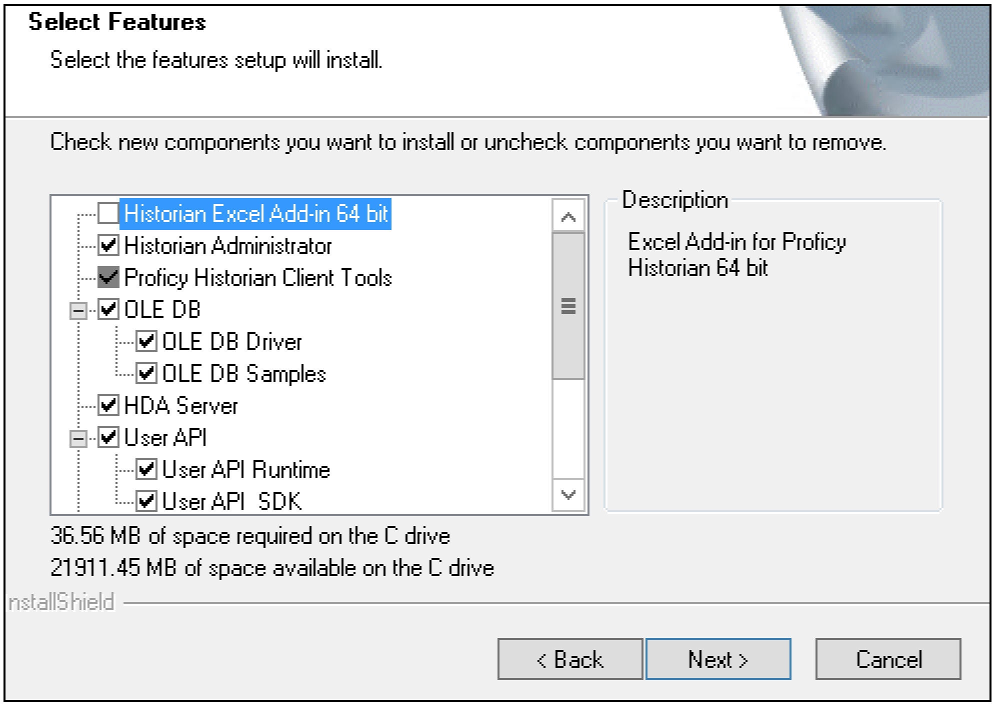
- Client Tools
- Historian Administrator

- 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 1046)*.

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.



Select Features

Select the features setup will install.

Check new components you want to install or uncheck components you want to remove.

- ☐ Historian Excel Add-in 64 bit
- ☑ Historian Administrator
- ☑ Proficy Historian Client Tools
- ☑ OLE DB
  - ☑ OLE DB Driver
  - ☑ OLE DB Samples
- ☑ HDA Server
- ☑ User API
  - ☑ User API Runtime
  - ☑ User API SDK

Description

Excel Add-in for Proficy Historian 64 bit

36.56 MB of space required on the C drive

21911.45 MB of space available on the C drive

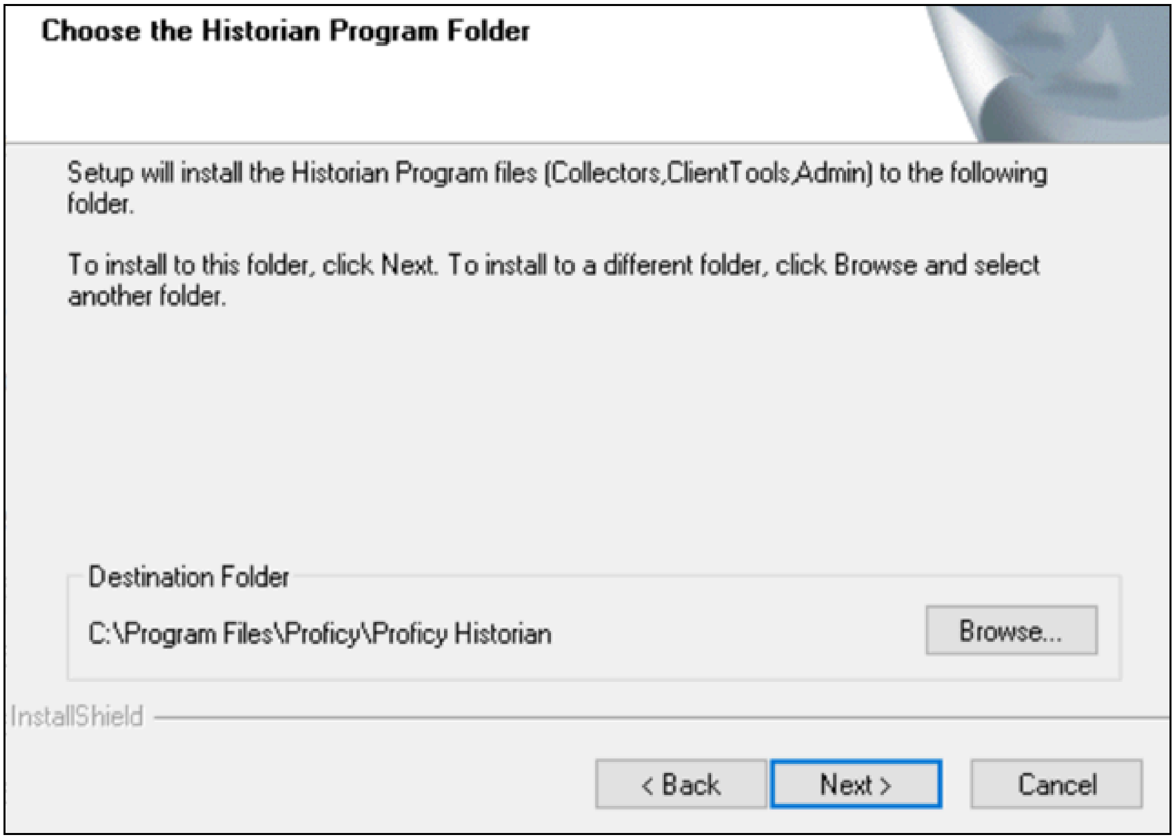
By default, the check boxes for components such as **Historian Administrator**, **HDA Server**, **OLE DB**, and **User API and SDK** are selected. If you do not want to install them at this time, clear the check boxes. You cannot, however, clear the **Proficy Historian Client Tools** check box.

> **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.

3. Select **Next**.
   The **Choose the Historian Program Folder** page appears.



Choose the Historian Program Folder

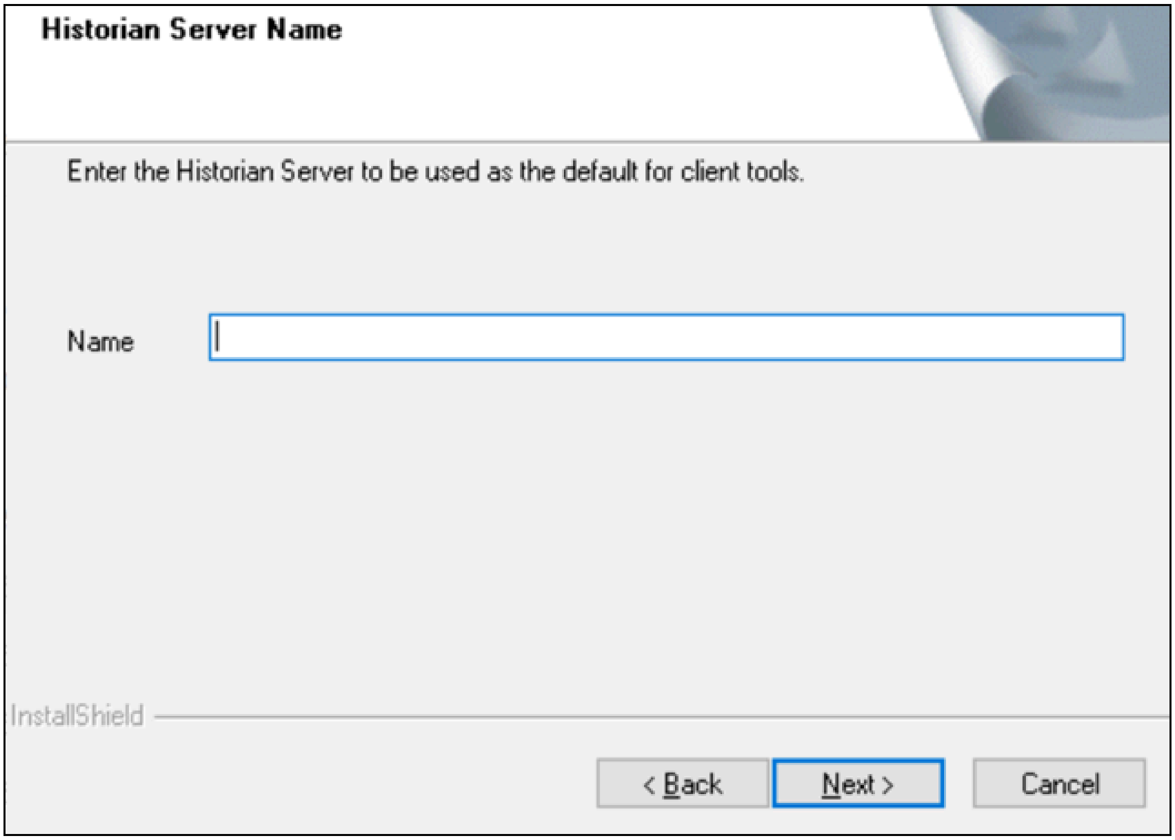
Setup will install the Historian Program files (Collectors,ClientTools,Admin) to the following folder.

To install to this folder, click Next. To install to a different folder, click Browse and select another folder.

Destination Folder

C:\Program Files\Proficy\Proficy Historian

4. As needed, change the destination folder of Client Tools, or leave the default folder, and then select **Next**.
   The **Historian Server Name** page appears.



Historian Server Name

Enter the Historian Server to be used as the default for client tools.

Name

5. Enter the Amazon Network Load Balancer (NLB) DNS.

> **Tip:**
>
> To find the NLB DNS:
>
> a. Access the EKS cluster on which you have deployed Proficy Historian for AWS.
> b. Access the EC2 instance.
> c. In the navigation pane, under **Load Balancing**, select **Load Balancers**.
> d. Select the load balancer for which you want to find the DNS.
> e. In the **Description** section, copy the DNS name.

6. 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 1042)* 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 (such as 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 993)*.

Connect to a Historian Server

1. Install Client Tools *(on page 1042)*, 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 1047)*
- Import Historian Data into Crystal Reports *(on page 1054)*
- Import Historian data into Microsoft Excel *(on page 1059)*

Run the following command:

```
Provider=iHOLEDB.iHistorian.1;PersistSecurity Info=False;

USER ID=[<Historian server username>];

Password=[<Historian server password>];

Data Source=[<NLB DNS>]
```

Working with Clients

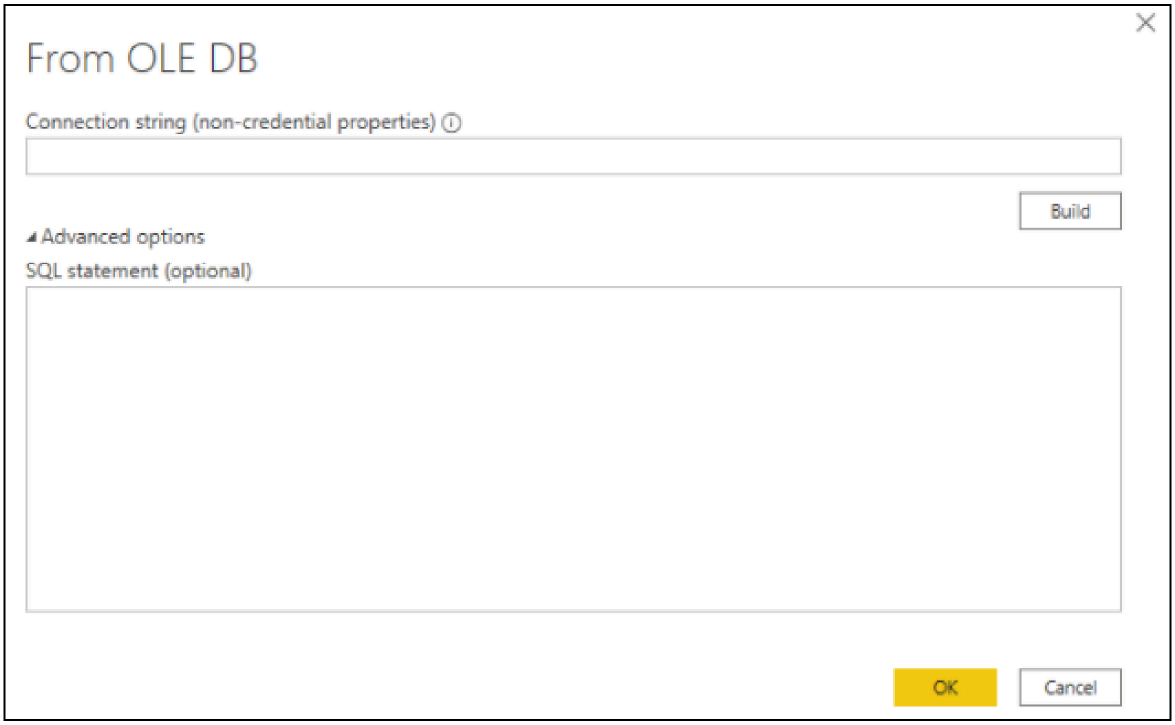
Power BI Desktop

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.

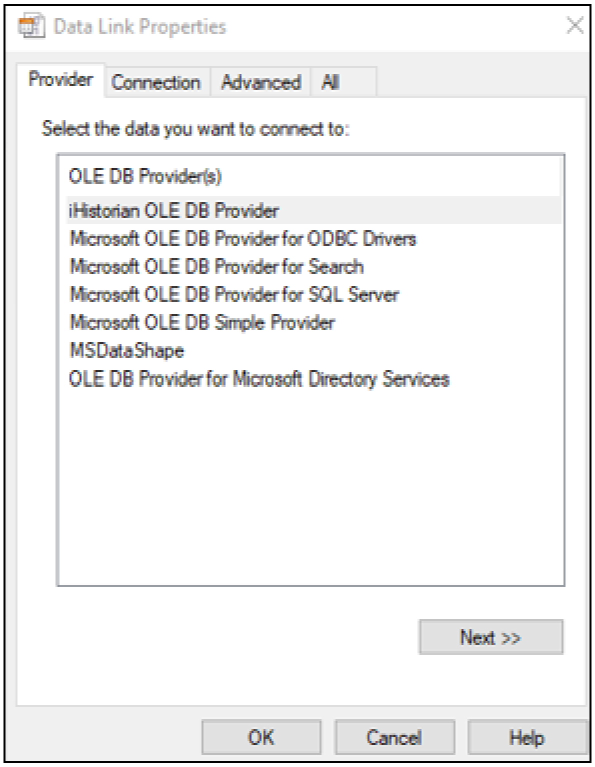
This topic describes how to import Historian data into Power BI Desktop.

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.

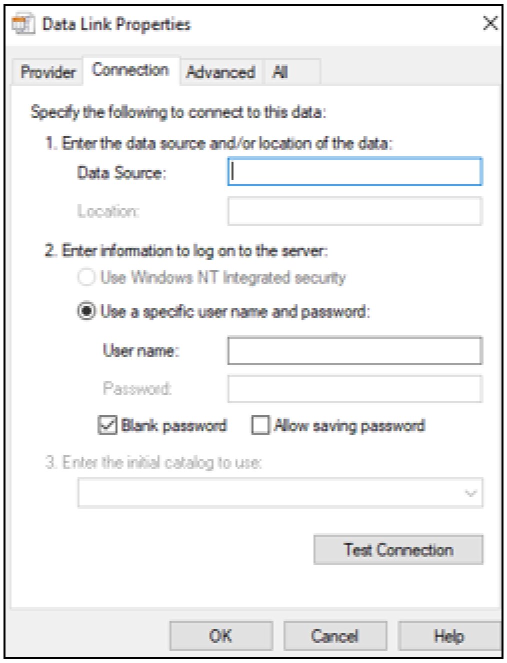


Select the data you want to connect to:

OLE DB Provider(s)
iHistorian OLE DB Provider
Microsoft OLE DB Provider for ODBC Drivers
Microsoft OLE DB Provider for Search
Microsoft OLE DB Provider for SQL Server
Microsoft OLE DB Simple Provider
MSDataShape
OLE DB Provider for Microsoft Directory Services

4. Select **Next**.

   The **Connection** section appears.

   Data Link Properties

   Provider | Connection | Advanced | All

   Specify the following to connect to this data:

   1. Enter the data source and/or location of the data:

   Data Source:

   Location:

   2. Enter information to log on to the server:

   ○ Use Windows NT Integrated security

   ● Use a specific user name and password:

   User name:

   Password:

   ☑ Blank password ☐ Allow saving password

   3. Enter the initial catalog to use:

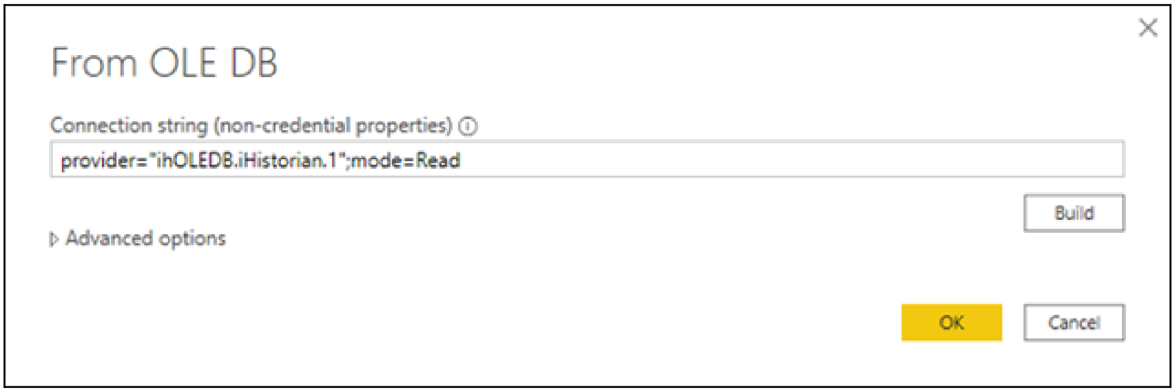
5. Leave the default values as is, and select **Test Connection.**

   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;PersistSecurity Info=False; USER ID=[<Historian server username>]; Password=[<Historian server password>]; Data Source=[<NLB DNS>]`

   From OLE DB

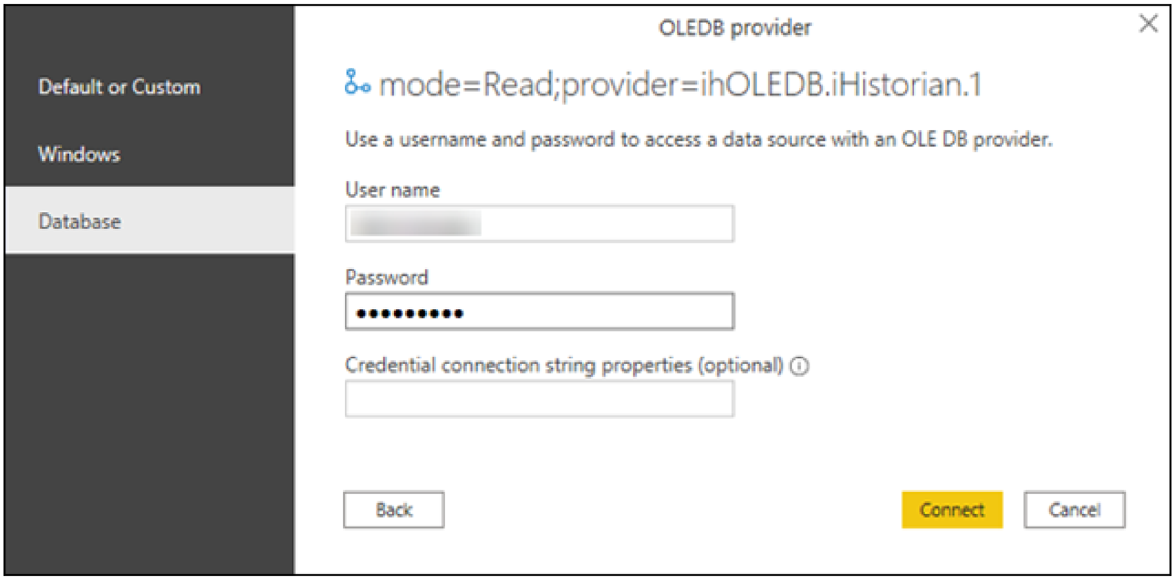
   Connection string (non-credential properties) ⓘ

   provider="ihOLEDB.iHistorian.1";mode=Read

7. Select **OK**.

The **OLE DB Provider** window appears.

8. In the **Database** section, enter `ihCloudHistAdmin` as the username, and enter the password that you provided in the **Proficy Authentication Configuration** field while deploying Proficy Historian for AWS, and then select **Connect**.



mode=Read;provider=ihOLEDB.iHistorian.1

Use a username and password to access a data source with an OLE DB provider.

User name

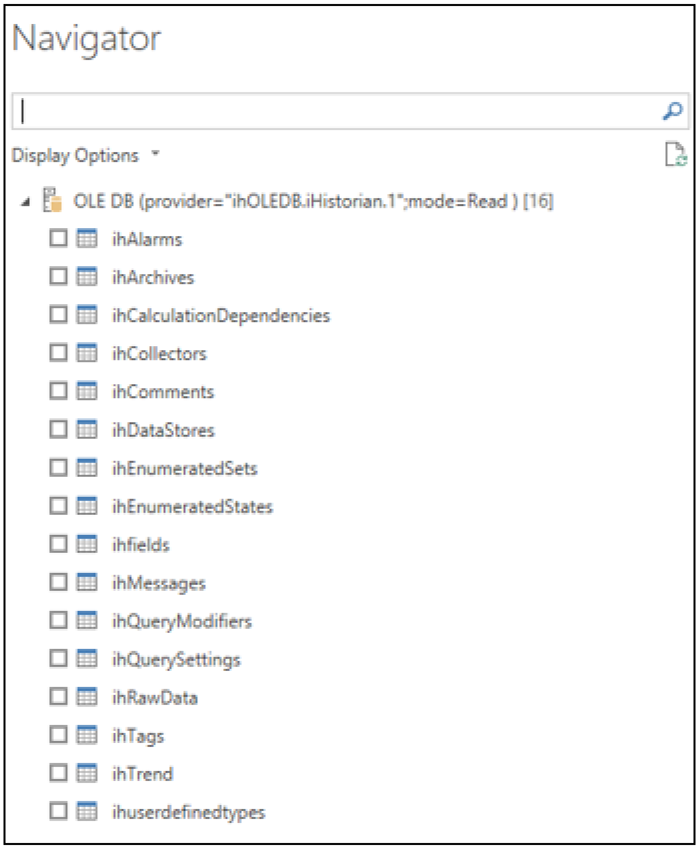
[redacted]

Password

●●●●●●●●●●

Credential connection string properties (optional)

A list of Historian tables appears in the **Navigator** section.

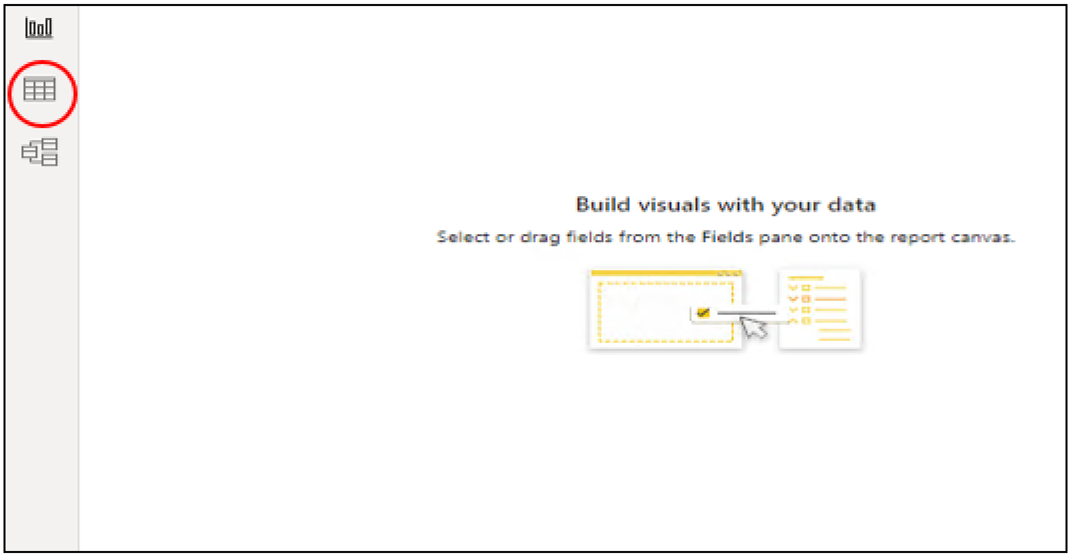


OLE DB (provider="ihOLEDB.iHistorian.1";mode=Read) [16]

- □ ihAlarms
- □ ihArchives
- □ ihCalculationDependencies
- □ ihCollectors
- □ ihComments
- □ ihDataStores
- □ ihEnumeratedSets
- □ ihEnumeratedStates
- □ ihfields
- □ ihMessages
- □ ihQueryModifiers
- □ ihQuerySettings
- □ ihRawData
- □ ihTags
- □ ihTrend
- □ ihuserdefinedtypes

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 ▦.



Build visuals with your data

Select or drag fields from the Fields pane onto the report canvas.

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.

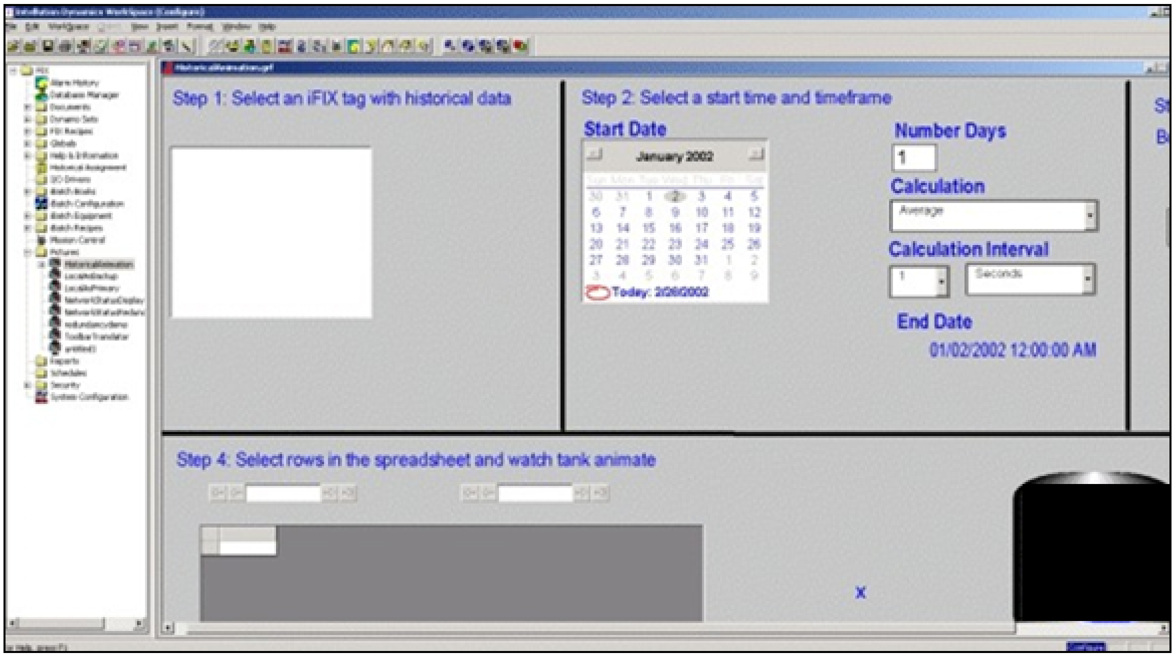
Access the iFIX Sample Picture

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.



Step 1: Select an iFIX tag with historical data

Step 2: Select a start time and timeframe

Start Date: January 2002

Today: 2/26/2002

Number Days: 1

Calculation: Average

Calculation Interval: 1 Seconds

End Date: 01/02/2002 12:00:00 AM

Step 4: Select rows in the spreadsheet and watch tank animate

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 1056)* for instructions.
- Analysis of the ihTrend and ihAlarm tables in Crystal Reports is not supported.

Table 137. Crystal Reports Samples

File Name	Description
`SimpleCrystal80Report.rpt`	Contains values cast from **Variant** to **Float**.
`MultipleServers-Subreport.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.
`iFIX1_CROSSTAB_OLEDB.rpt`	Contains data from iFIX Sample System converted from the iFIX Historical ODBC driver to OLE DB.
`iFIX1_DAILY_OLEDB.rpt`	Contains data from iFIX Sample System converted from the iFIX Historical ODBC driver to OLE DB.

Connect to the Historian Server

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. In the **Data Source** field, enter the Amazon Network Load Balancer (NLB) DNS.

    > **Tip:**
    >
    > To find the NLB DNS:
    >
    > i. Access the EKS cluster on which you have deployed Proficy Historian for AWS.
    > ii. Access the EC2 instance.
    > iii. In the navigation pane, under **Load Balancing**, select **Load Balancers**.
    > iv. Select the load balancer for which you want to find the DNS.
    > v. In the **Description** section, copy the DNS 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 1054)*.

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. In the **Data Source** field, enter the Amazon Network Load Balancer (NLB) DNS.

> **Tip:**
> To find the NLB DNS:
>
> i. Access the EKS cluster on which you have deployed Proficy Historian for AWS.
> ii. Access the EC2 instance.
> iii. In the navigation pane, under **Load Balancing**, select **Load Balancers**.
> iv. Select the load balancer for which you want to find the DNS.
> v. In the **Description** section, copy the DNS name.

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 1056)* for instructions.

11. Repeat the previous step for each field that you want to add, and then select **Finish**.
    The Crystal Report is generated.

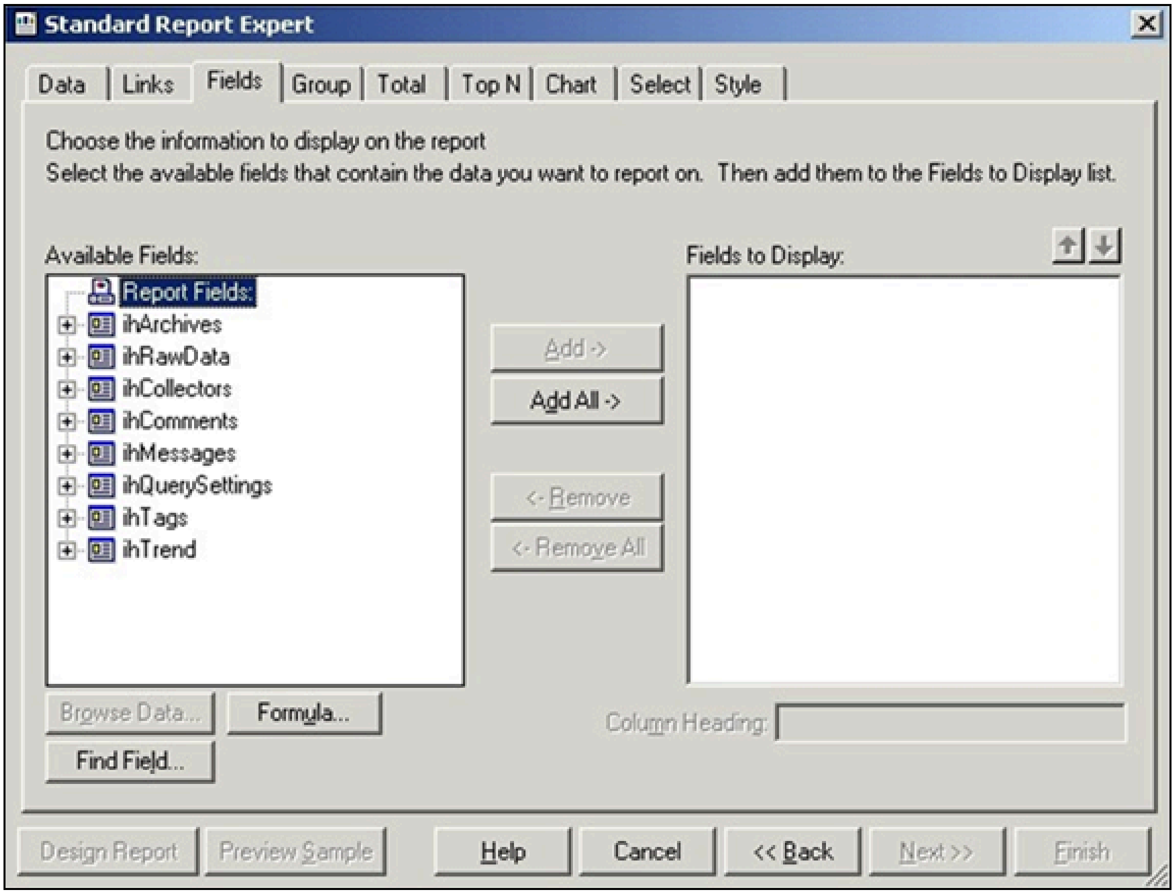
Format Decimal Point Precision

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.

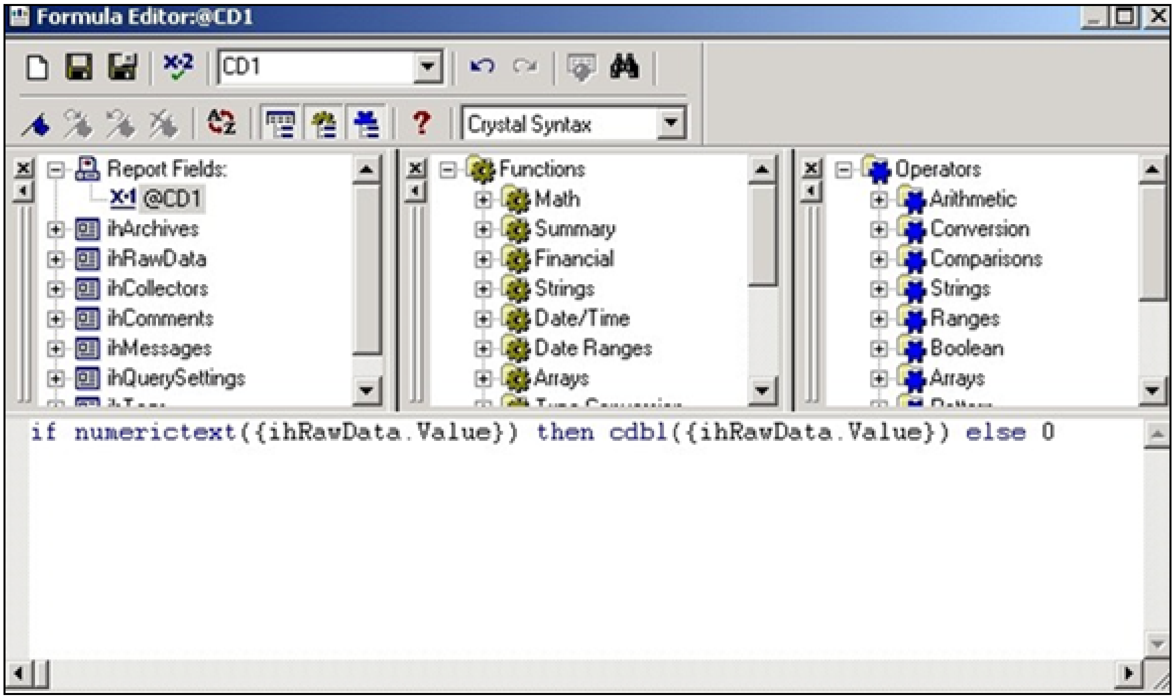
   The **Formula Editor** section appears.

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
```



```
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 1059)* or using a UDL file *(on page 1061)*.

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 138. 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 1061)* or by using the sample UDL file *(on page 1062)*.

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. In the **Data Source** field, enter the Amazon Network Load Balancer (NLB) DNS..

    > **Tip:**
    > To find the NLB DNS:
    > i. Access the EKS cluster on which you have deployed Proficy Historian for AWS.
    > ii. Access the EC2 instance.
    > iii. In the navigation pane, under **Load Balancing**, select **Load Balancers**.
    > iv. Select the load balancer for which you want to find the DNS.
    > v. In the **Description** section, copy the DNS name.

    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 opens.

   > **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 1063)*.

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 1059)* or using the sample UDL file *(on page 1062)*.

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. In the **Data Source** field, enter the Amazon Network Load Balancer (NLB) DNS..

        > **Tip:**
        > To find the NLB DNS:
        > i. Access the EKS cluster on which you have deployed Proficy Historian for AWS.
        > ii. Access the EC2 instance.
        > iii. In the navigation pane, under **Load Balancing**, select **Load Balancers**.
        > iv. Select the load balancer for which you want to find the DNS.
        > v. In the **Description** section, copy the DNS name.

    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 1063)*.

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 1059)* or by creating a .UDL file *(on page 1061)*.

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 1063)*.

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 1073)* 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 139. 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_databound-grid.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 1105)*.

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 1069)*.

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 Amazon Network Load Balancer (NLB) DNS, and then select **Provider Options**.

   > **Tip:**
   > To find the NLB DNS:
   >
   > a. Access the EKS cluster on which you have deployed Proficy Historian for AWS.
   > b. Access the EC2 instance.
   > c. In the navigation pane, under **Load Balancing**, select **Load Balancers**.
   > d. Select the load balancer for which you want to find the DNS.
   > e. In the **Description** section, copy the DNS name.

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 1068)*. 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 1068)*.

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 1068)*.

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'
   ```

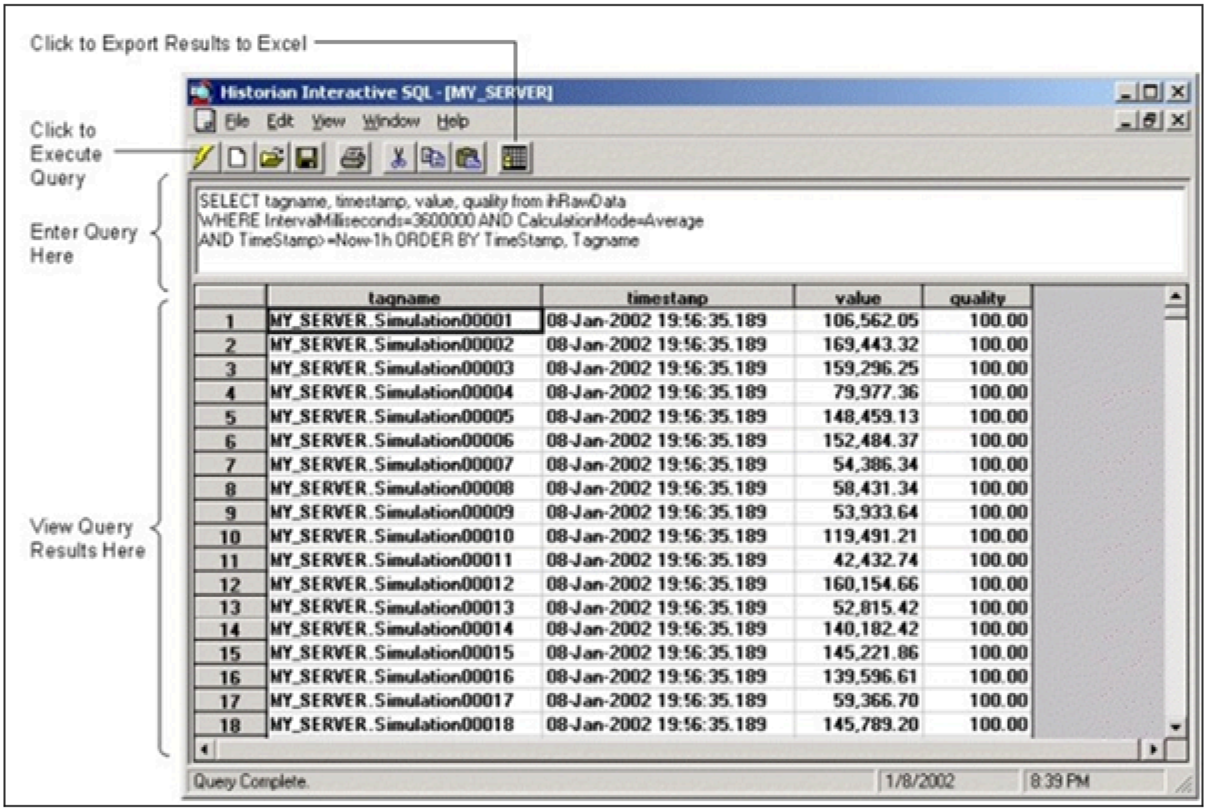
   b. When you create the stored procedure in **Enterprise Manager**, include the following lines before the create procedure command to avoid an error:

   ```
   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.


Click to Export Results to Excel
Historian Interactive SQL - [MY_SERVER]
File Edit View Window Help
Click to Execute Query
Enter Query Here
View Query Results Here
Query Complete.
1/8/2002
8:39 PM

```
SELECT tagname, timestamp, value, quality from ihRawData
WHERE IntervalMilliseconds=3600000 AND CalculationMode=Average
AND TimeStamp>=Now-1h ORDER BY TimeStamp, Tagname
```

	tagname	timestamp	value	quality
1	MY_SERVER.Simulation00001	08-Jan-2002 19:56:35.189	106,562.05	100.00
2	MY_SERVER.Simulation00002	08-Jan-2002 19:56:35.189	169,443.32	100.00
3	MY_SERVER.Simulation00003	08-Jan-2002 19:56:35.189	159,296.25	100.00
4	MY_SERVER.Simulation00004	08-Jan-2002 19:56:35.189	79,977.36	100.00
5	MY_SERVER.Simulation00005	08-Jan-2002 19:56:35.189	148,459.13	100.00
6	MY_SERVER.Simulation00006	08-Jan-2002 19:56:35.189	152,484.37	100.00
7	MY_SERVER.Simulation00007	08-Jan-2002 19:56:35.189	54,386.34	100.00
8	MY_SERVER.Simulation00008	08-Jan-2002 19:56:35.189	58,431.34	100.00
9	MY_SERVER.Simulation00009	08-Jan-2002 19:56:35.189	53,933.64	100.00
10	MY_SERVER.Simulation00010	08-Jan-2002 19:56:35.189	119,491.21	100.00
11	MY_SERVER.Simulation00011	08-Jan-2002 19:56:35.189	42,432.74	100.00
12	MY_SERVER.Simulation00012	08-Jan-2002 19:56:35.189	160,154.66	100.00
13	MY_SERVER.Simulation00013	08-Jan-2002 19:56:35.189	52,815.42	100.00
14	MY_SERVER.Simulation00014	08-Jan-2002 19:56:35.189	140,182.42	100.00
15	MY_SERVER.Simulation00015	08-Jan-2002 19:56:35.189	145,221.86	100.00
16	MY_SERVER.Simulation00016	08-Jan-2002 19:56:35.189	139,596.61	100.00
17	MY_SERVER.Simulation00017	08-Jan-2002 19:56:35.189	59,366.70	100.00
18	MY_SERVER.Simulation00018	08-Jan-2002 19:56:35.189	145,789.20	100.00

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:

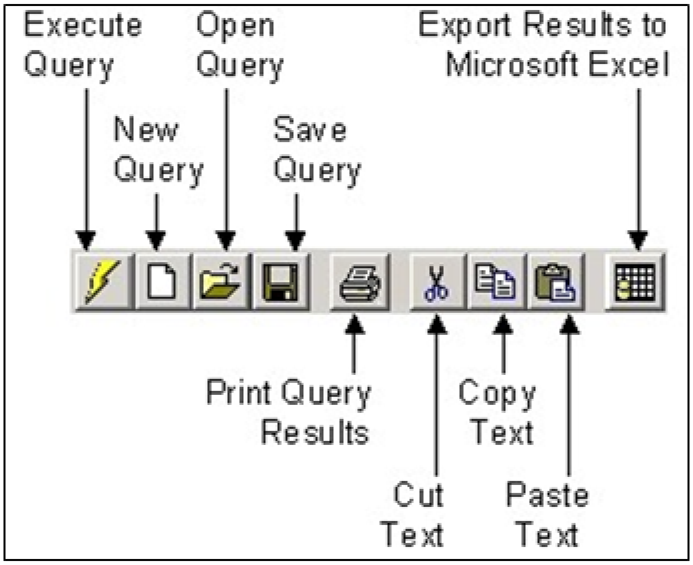
- Test SQL syntax before using it in an application.
- Troubleshoot OLE DB connections or Historian errors.
- Perform more complex searching or filtering of data than you can in the Historian SDK and administration applications.
- Retrieve data from any available Historian server.
- Save and access queries.
- 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.



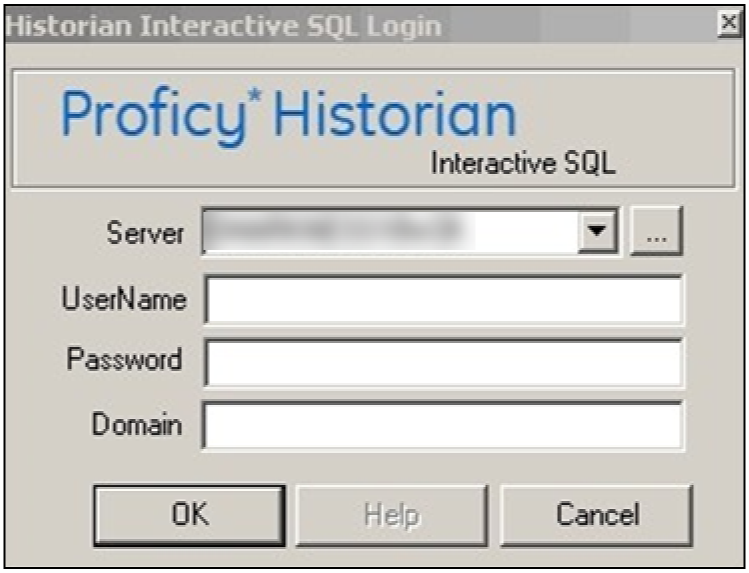
Access the Historian Interactive SQL Application

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. In the **Server** field, enter the Amazon Network Load Balancer (NLB) DNS.

> **Tip:**
>
> To find the NLB DNS:
>
> a. Access the EKS cluster on which you have deployed Proficy Historian for AWS.
> b. Access the EC2 instance.
> c. In the navigation pane, under **Load Balancing**, select **Load Balancers**.
> d. Select the load balancer for which you want to find the DNS.
> e. In the **Description** section, copy the DNS name.

3. Enter the username and password to connect to the server. If you do not enter user credentials, the currently logged-in user is considered. Leave the Domain field blank.
4. Select **OK**.

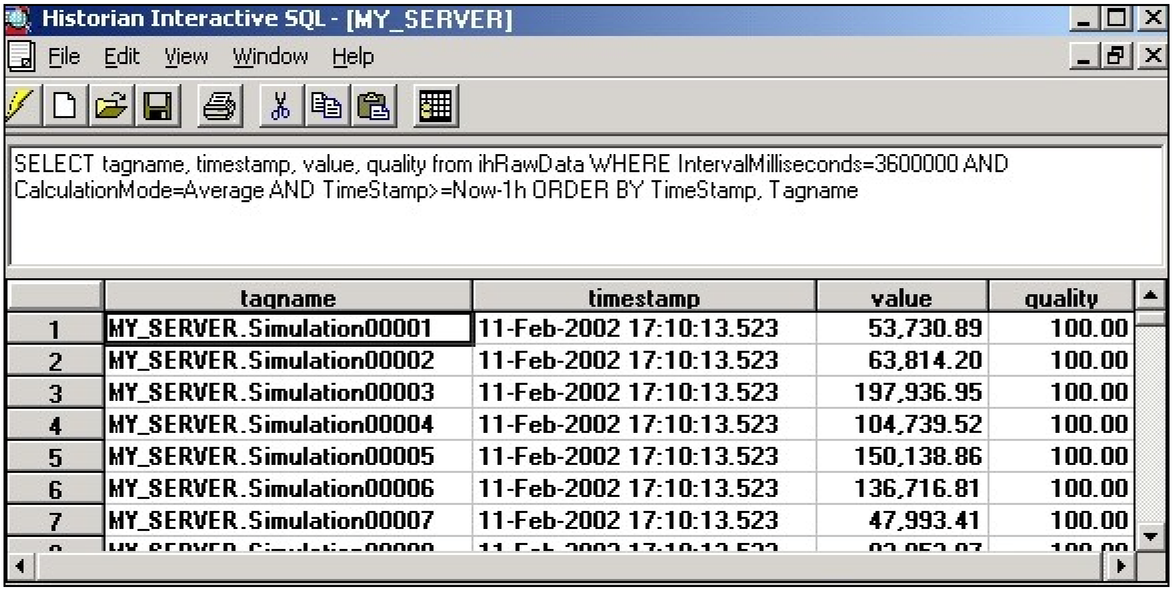
   A new session of the Historian Interactive SQL application appears, and it is connected to the server that you have specified. The session begins with the default values for SET variables *(on page 1098)*.

> **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 1075)*.
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.



```
SELECT tagname, timestamp, value, quality from ihRawData WHERE IntervalMilliseconds=3600000 AND
CalculationMode=Average AND TimeStamp>=Now-1h ORDER BY TimeStamp, Tagname
```

	tagname	timestamp	value	quality
1	MY_SERVER.Simulation00001	11-Feb-2002 17:10:13.523	53,730.89	100.00
2	MY_SERVER.Simulation00002	11-Feb-2002 17:10:13.523	63,814.20	100.00
3	MY_SERVER.Simulation00003	11-Feb-2002 17:10:13.523	197,936.95	100.00
4	MY_SERVER.Simulation00004	11-Feb-2002 17:10:13.523	104,739.52	100.00
5	MY_SERVER.Simulation00005	11-Feb-2002 17:10:13.523	150,138.86	100.00
6	MY_SERVER.Simulation00006	11-Feb-2002 17:10:13.523	136,716.81	100.00
7	MY_SERVER.Simulation00007	11-Feb-2002 17:10:13.523	47,993.41	100.00

4. Select ⚡ or press Ctrl+E.

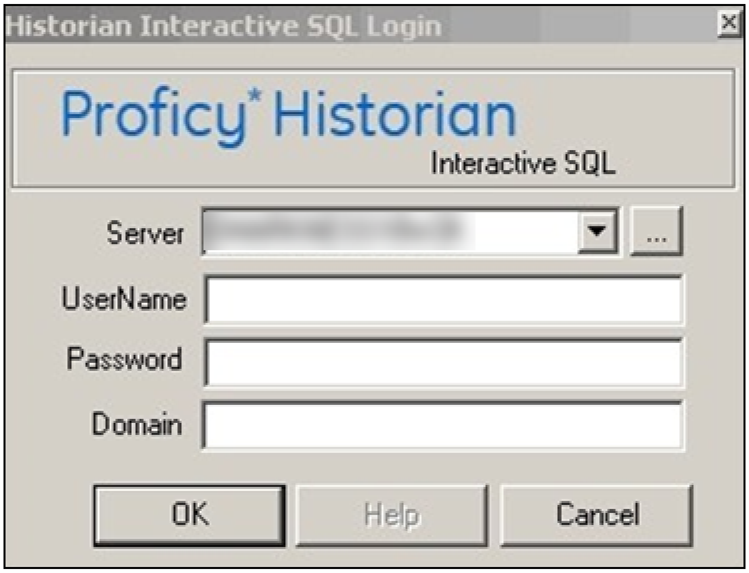
   The query results appear.

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 1075)*.
2. Select **File > New**.

   The **Historian Interactive SQL Login** window appears.



3. In the **Server** field, enter the Amazon Network Load Balancer (NLB) DNS.

> **Tip:**
>
> To find the NLB DNS:
>
> a. Access the EKS cluster on which you have deployed Proficy Historian for AWS.
> b. Access the EC2 instance.
> c. In the navigation pane, under **Load Balancing**, select **Load Balancers**.
> d. Select the load balancer for which you want to find the DNS.
> e. In the **Description** section, copy the DNS name.

4. Enter the username and password to connect to the server. If you do not enter user credentials, the currently logged-in user is considered. Leave the Domain field blank.
5. Select **OK**.

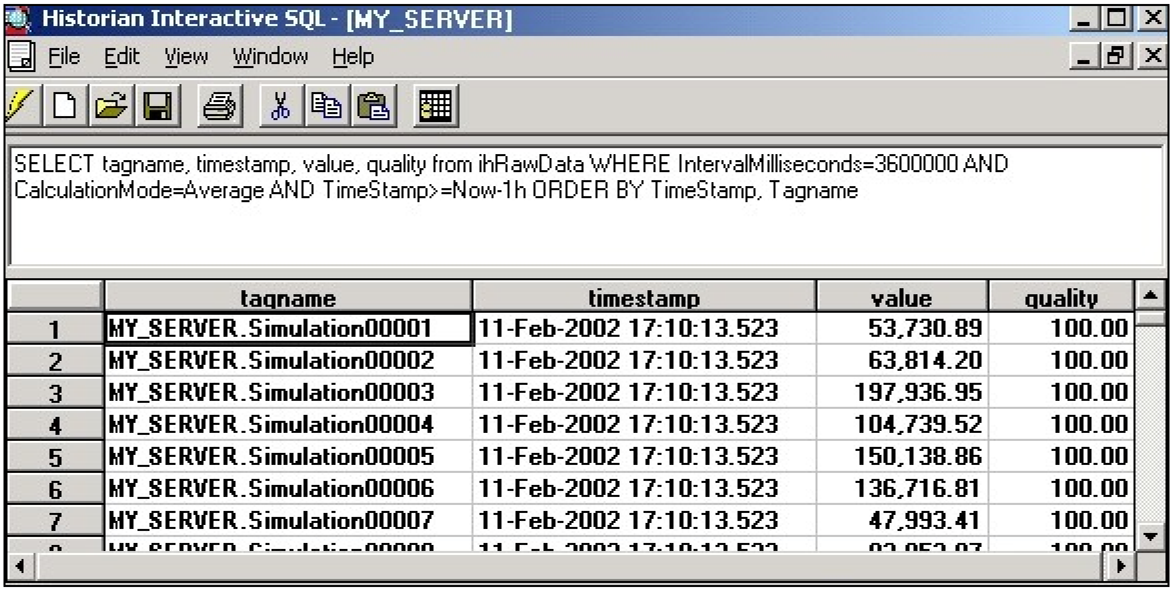
   A new session of the Historian Interactive SQL application appears, and it is connected to the server that you have specified. The session begins with the default values for SET variables *(on page 1098)*.

> **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 1075)*.
2. Enter your query into the **Query Entry** field.



```
SELECT tagname, timestamp, value, quality from ihRawData WHERE IntervalMilliseconds=3600000 AND
CalculationMode=Average AND TimeStamp>=Now-1h ORDER BY TimeStamp, Tagname
```

	tagname	timestamp	value	quality
1	MY_SERVER.Simulation00001	11-Feb-2002 17:10:13.523	53,730.89	100.00
2	MY_SERVER.Simulation00002	11-Feb-2002 17:10:13.523	63,814.20	100.00
3	MY_SERVER.Simulation00003	11-Feb-2002 17:10:13.523	197,936.95	100.00
4	MY_SERVER.Simulation00004	11-Feb-2002 17:10:13.523	104,739.52	100.00
5	MY_SERVER.Simulation00005	11-Feb-2002 17:10:13.523	150,138.86	100.00
6	MY_SERVER.Simulation00006	11-Feb-2002 17:10:13.523	136,716.81	100.00
7	MY_SERVER.Simulation00007	11-Feb-2002 17:10:13.523	47,993.41	100.00

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 1077)*.
2. Select ▦.

   The query results are exported to an Excel spreadsheet.

Format the date and time *(on page 1063)* 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.

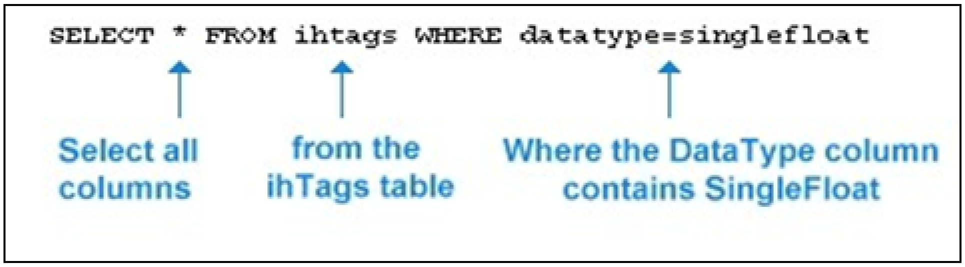
Supported SQL Syntax

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.



```
SELECT * FROM ihtags WHERE datatype=singlefloat
```

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 1115)* .

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 1115)*.

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.

Abbreviation	Description
`ASC`	Specifies that the values must be sorted in ascending order, from lowest value to highest value.

Abbreviation	Description
`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 140. 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.
`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)
```

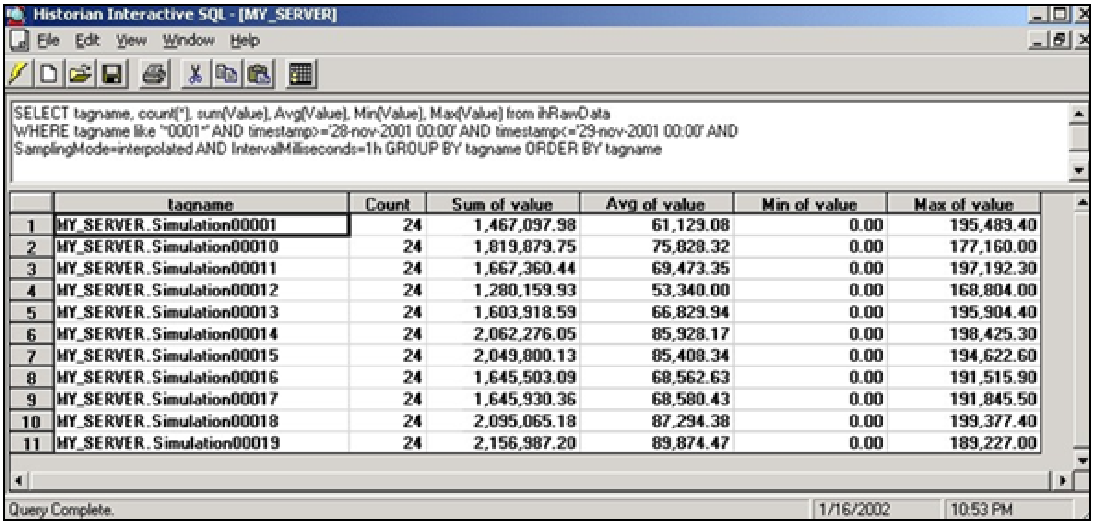
Example 1: Retrieve the Total Number of Tags

```
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.



```
SELECT tagname, count(*), sum(Value), Avg(Value), Min(Value), Max(Value) from ihRawData
WHERE tagname like '*0001*' AND timestamp>='28-nov-2001 00:00' AND timestamp<='29-nov-2001 00:00' AND
SamplingMode=interpolated AND IntervalMilliseconds=1h GROUP BY tagname ORDER BY tagname
```

	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

Query Complete. 1/16/2002 10:53 PM

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 141. 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 142. 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.

Table 142. Supported Join Operations (continued)

Supported Join Feature	Description
Cross Join	Returns all rows from the left table. Each row from the left table is combined with all rows from the right table.
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
`thurs`	The previous Thursday at midnight

Time Segment	Meaning
`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 143. Relative Date and Time Shortcuts

Time Segment	Meaning
`s`	Second
`m`	Minute
`h`	Hour
`d`	Day
`w`	Week
`micro`	Microsecond

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 144. Expression Comparisons

Comparison Symbol	Meaning
`<`	Less Than
`>`	Greater Than
`<=`	Less Than or Equal
`>=`	Greater Than or Equal
`=`	Equal

Table 144. 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 1175)*.

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 145. SET Statement Variables

Variable	Description	Default Setting
`StartTime`	A valid date and time string, such as: • `StartTime = '14-sep-200111:00:00'` • `StartTime = Now -1h` • `StartTime = '02/01/199811:00:00'` • `StartTime = {ts '2002-06-20 15:34:08'}` • `StartTime = '7/12/201112:03:16.100000'`	Two hours prior to execution of the query.
`EndTime`	A valid date and time string, such as: `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: • `CurrentValue` • `Interpolated` • `InterpolatedtoRaw` • `RawByTime` • `RawByNumber` • `Calculated` • `Lab` • `LabtoRaw` • `Trend` • `TrendtoRaw` • `Trend2` • `TrendtoRaw2` • `RawByFilterToggle`	`Calculated`

Table 145. SET Statement Variables (continued)

Variable	Description	Default Setting
`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` - `Backward`	`Forward`
`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`. 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. For example: - If you run a query with `'IntervalMilliseconds = 100'`, it returns samples in 100-millisecond intervals. - 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: - `Average` - `StandardDeviation` - `Total` - `Minimum` - `MaximumCount` - `RawAverage` - `RawStandardDeviation` - `RawTotal` - `MinimumTime` - `MaximumTime` - `Count` - `TimeGood`	`Average`

Table 145. SET Statement Variables (continued)

Variable	Description	Default Setting
	- `FirstRawValue` - `FirstRawTime` - `LastRawValue` - `LastRawTime` - `TagStats`	
`FilterTag`	A valid tagname used to define the filter. For example: `FilterTag = 'SimulationString00001'` 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: - `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. `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: - `Equal` - `EqualFirst` - `EqualLast` - `NotEqual` - `LessThan` - `GreaterThan`	`Equal`

Table 145. SET Statement Variables (continued)

Variable	Description	Default Setting
	- `LessThanEqual` - `GreaterThanEqual` - `AllBitsSet` - `AnyBitSet` - `AnyBitNotSet` - `AllBitsNotSet` 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`. `FilterExpression = 'BatchID=B1'` 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. For example: `FilterValue = 'ABCD-1086031382099'` 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: - `Client` - `Server` - Explicit bias number (number of minutes from GMT) For example, an explicit bias number of `300` represents 300 minutes from GMT.	`Client`

Table 145. SET Statement Variables (continued)

Variable	Description	Default Setting
	Note: Time zones are not supported on Windows 9x computers.	
`DaylightSavingTime`	Indicates whether Daylight Saving Time logic should be applied to timestamps. Valid values: • `True` • `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 1107)*
- Cannot Log Into the Historian Interactive SQL Application *(on page 1107)*
- Cannot Get Historian OLE DB provider Data *(on page 1107)*
- Samples Do Not Run *(on page 1108)*
- Time Zones Do Not Work *(on page 1108)*
- Cannot Get String Data From the ihRawData Table *(on page 1108)*
- Timestamps Include Only the Previous Two Hours *(on page 1109)*
- Row Count Less Than Expected *(on page 1109)*
- Linked Server Not Working *(on page 1109)*
- SET Not Applied to SELECT When Using a Linked Server *(on page 1109)*
- Client Crashes When Using Historian OLE DB provider *(on page 1110)*

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. Access Historian Administrator *(on page 1194)*.
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 1054)* 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 1097)* 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 1058)*.

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 1121)*	Contains Historian tag configuration information.
ihArchives Table *(on page 1128)*	Contains Historian archive configuration information, plus performance statistics for each archive.
ihCol-	Contains configuration and status information for each collector connected to the Historian server.

Table Name	Description
lectors Table *(on page 1131)*	
ihMessages Table *(on page 1137)*	Contains Historian messages such as alerts, informational topics, and connection information contained in the audit log.
ihRawData Table *(on page 1140)*	Contains collected data for each tag in the Historian server. It contains not just raw data, but also calculated and interpolated data.
ihComments Table *(on page 1151)*	Contains the comments associated with the Historian data.

Table Name	Description
ihTrend Table *(on page 1159)*	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 1175)*	Contains a set of parameters that apply to all queries you make in that session, unless overridden by a `WHERE` clause.
ihCalculationDependencies *(on page 1181)*	Contains the calculation dependencies for tags.
ihAlarms Table *(on page 1182)*	Contains collected alarms and events data.

Table Name	Description
ihEnumeratedSets Table *(on page 1185)*	Contains information about enumerated sets.
ihEnumeratedStates Table *(on page 1187)*	Contains information about enumerated states.
ihUserDefinedTypes Table *(on page 1189)*	Contains information about user-defined data types.

Table Name	Description
ihFields Table *(on page 1191)*	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 146. 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. **Note:** There is no length limit for Historian tag names in the Data Archiver. However, differen applications may have their own limits.
`Description`	`VT_BSTR`	User description of the tag.
`EngUnits`	`VT_BSTR`	Engineering units description of the tag.

Column Name	Data Type	Description
`Comment`	`VT_-BSTR`	User comment associated with the selected tag.
`DataType`	`VT_-BSTR`	The data type of the tag: - `Scaled` - `SingleFloat` - `DoubleFloat` - `SingleInteger` - `DoubleInteger` - `Quad Integer` - `Unsigned Single Integer` - `Unsigned Double Integer` - `Unsigned Quad Integer` - `Byte` - `Boolean` - `FixedString` - `VariableString` - `BLOB` The data type returned in this column is the data type that you defined in Historian Administra cation.
`FixedStringLength`	`VT_UI1`	Zero unless the data type is `FixedString`. If the data type is `FixedString`, this number represen maximum length of the string value.
`CollectorName`	`VT_-BSTR`	Name of the collector responsible for collecting data for the specified tag.
`SourceAddress`	`VT_-BSTR`	Address used to identify the tag at the data source. For iFIX systems, this is the NTF (`Node.Tag`
`CollectionType`	`VT_-BSTR`	Type of collection used to acquire data for the tag: - **Unsolicited:** The collector accepts data from the source whenever the source presents - **Polled:** The collector acquires data from a source on a periodic schedule determined b lector.

Column Name	Data Type	Description
		Note: Not all collectors support unsolicited collection.
`CollectionInterval`	`VT_I4`	The time interval, in milliseconds, between readings of data from this tag. For polled collection, this field represents the time between samples. For unsolicited collectio 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 distri 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: • **Source:** The source delivers the timestamp along with the data sample. • **Collector:** The collector delivers the timestamp along with the collected data.
`HiEngineeringUnits`	`VT_R8`	The high end of the engineering units range. Used only for scaled data types and input scaled
`LoEngineeringUnits`	`VT_R8`	The low end of the engineering units range. Used only for scaled data types and input scaled t
`InputScaling`	`VT_BOOL`	Indicates whether the measurement should be converted to an engineering units value. When `False`, the measurement is interpreted as a raw measurement. When set to `True`, the system converts the value to engineering units by scaling the value betw `HiScale` and `LoScale` columns. If not enabled, the system assumes the measurement is alread ed 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. Collector compression applies a smoothing filter to incoming data by ignoring incremental ch values that fall within a deadband centered around the last collected value. The collector pass archiver) any new value that falls outside the deadband and then centers the deadband aroun value.

Column Name	Data Type	Description
`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.
`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. Refer to "Implementing Historian Security" in the *Getting Started with Historian* manual for def 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. Refer to "Implementing Historian Security" in the *Getting Started with Historian* manual for def the various security levels and groups.
`AdministratorSecurityGroup`	`VT_BSTR`	The name of the Windows security group responsible for controlling configuration changes fo
`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 m onds.

Column Name	Data Type	Description
`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: • `Undefined` • `iFIX` • `Simulation` • `OPC` • `File` • `iFIXLabData` • `ManualEntry` • `Simulation` • `Other`
`StoreMilliseconds`	VT_BOOL	Indicates whether milliseconds are recorded in timestamps. If not enabled, the time resolution is in seconds instead of milliseconds. Maximum data comp achieved when this option is set to `False`. This is the optimum setting for most applications. **Note:** `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 pressed as an offset from UTC (Universal Time Coordinated) in minutes. UTC is the international time standard, the current term for what was commonly referred to as wich Mean Time (GMT).
`AverageCollectionTime`	`VT_I4`	The average time it takes to execute the calculation tag since you started the Calculation colle
`CollectionDisabled`	`VT_I4`	Indicates whether collection is enabled (`0`) or disabled (`1`) for the tag. The default setting is en
`CollectorCompressionTimeout`	`VT_I4`	Indicates the maximum amount of time the collector will wait between sending samples to th This time is kept per tag, as different tags report to the archiver at different times.

Column Name	Data Type	Description
		This value should be in increments of your collection interval, and not less. Ideally, this field is used for polled data values. It can be used with unsolicited data, but when you are dependent on the data source for the value to change. With unsolicited data, since His ly records the value when it changes, the actual time before the timeout might exceed the con timeout.
`ArchiveCompressionTimeout`	`VT_I4`	Indicates the maximum amount of time from the last stored point before another point is stor value does not exceed the archive compression deadband. The data archiver treats the incoming sample after the timeout occurs as if it exceeded comp then stores the pending sample.
`TimeZone`	`VT_BSTR`	The type of time zone used: • Client • Server • 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 n 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.

Column Name	Data Type	Description
`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 abou via the `ihEnumeratedSet` table.
`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. 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 a namic, 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 147. ihArchives Table

Column Name	Data Type	Description
`ArchiveName`	`VT_BSTR`	Name of the archive for the current server if the authenticated user is a member of Historian Administrators group.

Table 147. ihArchives Table (continued)

Column Name	Data Type	Description
`ArchiveStatus`	`VT_BSTR`	The status of the specified archive: - `Undefined` - `Empty` - `NotEmpty`
`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.
`FileSizeCurrentDisk`	`VT_I4`	The actual space on the hard disk (in MB) for the specified archive.
`FileSizeCurrent`	`VT_I4`	The size of the archive file that is currently being used (in MB) for the specified archive.
`FileSizeTarget`	`VT_I4`	The target size of the specified archive file (in MB).
`StartTime`	`VT_DBTimeStamp`	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-`	The end time of the specified archive. This represents the latest timestamp (including date and time) for any tag contained in the archive.

Table 147. ihArchives Table (continued)

Column Name	Data Type	Description
	TimeStamp	
LastBackup	VT_DBTimeStamp	The date and time the most recent online backup was performed on this archive.
LastBackupUser	VT_BSTR	The name of the user who performed the most recent online backup.
LastModified	VT_DBTimeStamp	The date and time that the archive was last modified. The time structure includes milliseconds.
LastModifiedUser	VT_BSTR	The username of the Windows user who last modified the archive.
TimeZone	VT_BSTR	The type of time zone used: • Client • Server • Explicit bias number (number of minutes from GMT).
DaylightSavingTime	VT_BOOL	Indicates whether Daylight Saving Time logic should be applied to timestamps.

Table 147. ihArchives Table (continued)

Column Name	Data Type	Description
`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.
`DataStoreName`	`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 148. 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.

Table 148. ihCollectors Table (continued)

Column Name	Data Type	Description
`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: • `Unknown` • `Starting` • `Running` • `Stopping` • `Stopped`
`CollectorType`	`VT_BSTR`	The type of collector responsible for collecting data for the tag: • `Undefined` • `iFIX` • `Simulation` • `OPC` • `OPC AE` • `File` • `iFIXLabData` • `ManualEntry` • `Simulation` • `Calculation` • `ServerToServer` • `Other`
`MaximumDiskFreeBuffer-Size`	`VT_I4`	The maximum size (in MB) of the disk buffer for outgoing data.
`MaximumMemoryBuffer-Size`	`VT_I4`	The maximum size of the memory buffer (in MB) for outgoing data. The memory buffer stores data during short-term or momentary interruptions of the server connec-

Table 148. ihCollectors Table (continued)

Column Name	Data Type	Description
		tion. 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`. **Note:** 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 148. 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:

Table 148. ihCollectors Table (continued)

Column Name	Data Type	Description
		• **Unsolicited:** The collector accepts data from the source whenever the source presents the data. • **Polled:** The collector acquires data from a source on a periodic schedule determined by the collector. **Note:** 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: • **Source:** The source delivers the timestamp along with the data sample. • **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. 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.

Table 148. ihCollectors Table (continued)

Column Name	Data Type	Description
`DefaultSpikeMultiplier`	`VT_R4`	The default Spike Logic multiplier.
`DefaultSpikeInterval`	`VT_I4`	The default Spike Logic interval.
`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: • Client • Server • 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 149. 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.

Table 149. ihMessages Table (continued)

Column Name	Data Type	Description
`Microseconds`	`VT_I4`	The microsecond portion of the date and time for the message.
`Topic`	`VT_BSTR`	The topic name of the message: - `AlertTopics` - `AllTopics` - `ConfigurationAudit` - `Connections` - `General` - `MessageTopicMax` - `MessageTopics` - `Performance` - `ServiceControl` - `Security` - `Undefined`
`Username`	`VT_BSTR`	Name of the Windows user who generated the message, or who the message is associated with.
`MessageNumber`	`VT_I4`	Message number for the message. A message number is a unique identifier associated with the message template.
`MessageString`	`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.
`TimeZone`	`VT_BSTR`	The type of time zone used: - Client - Server - Explicit bias number (number of minutes from GMT)

Table 149. ihMessages Table (continued)

Column Name	Data Type	Description
`DaylightSavingTime`	`VT_BOOL`	Indicates whether Daylight Saving Time logic should be applied to timestamps.
`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 150. ihRawData Table

Column Name	Data Type	Description
`Tagname`	`VT_BSTR`	Tagname property of the tag. **Note:** 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.
`Microseconds`	`VT_DBTimeStamp`	The microsecond interval for the data sample.

Table 150. ihRawData Table (continued)

Column Name	Data Type	Description
`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. For raw sampled data, data values are: • `Good` • `Bad` • `Uncertain` • `Not Available` • `Really Unknown` This column also includes the subquality of the data value, if it exists: • `NonSpecific` • `ConfigError` • `NotConnected` • `DeviceFail` • `SensorFail` • `LastKnownValue` • `CommFailure` • `OutOfService` • `ScaledOutOfRange` • `OffLine` • `NoValue` • `Really Unknown`
`OPCQualityValid`	`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.

Table 150. ihRawData Table (continued)

Column Name	Data Type	Description
`OPCQuality`	`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.
`SamplingMode`	`VT_BSTR`	The mode used to sample data from the archive: • `CurrentValue`: Retrieves the current value. Time frame criteria are ignored. • `Interpolated`: Retrieves evenly spaced interpolated values based on interval or `NumberOfSamples` and the time frame criteria. • `RawByTime`: Retrieves raw archive values based on time frame criteria. • `RawByNumber`: Retrieves raw archive values based on the `StartTime`, `NumberOfSamples`, and `Direction` criteria. **Note:** `EndTime` criteria are ignored for this sampling mode. • `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. • `Calculated`: Retrieves evenly spaced calculated values based on `NumberOfSamples`, interval, the time frame criteria, and the `CalculationMode` criteria. • `Lab`: Returns actual collected values without interpolation. • `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. • `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 150. 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. - `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. - `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. - `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. 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. **Note:** 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. **Note:** The `NumberofSamples` and `IntervalMilliseconds` columns are mutually exclusive. If `NumberofSamples` is used, `IntervalMilliseconds` is not used.

Table 150. ihRawData Table (continued)

Column Name	Data Type	Description
`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: • `Average` • `Count` • `Maximum` • `MaximumTime` • `Minimum` • `MinimumTime` • `StandardDeviation` • `Total` • `RawAverage` • `RawStandardDeviation` • `RawTotal` • `TimeGood` • `FirstRawValue` • `FirstRawTime` • `LastRawValue` • `LastRawTime` • `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: • `ExactTime`: Retrieves data for the exact times that the filter condition is `True`. • `BeforeTime`: Retrieves data from the time of the last `False` filter condition to the time of the next `True` condition. • `AfterTime`: Retrieves data from the time of the last `True` filter condition to the next `False` condition. • `BeforeAndAfterTime`: Retrieves data from the time of the last `False` filter condition to the next `False` condition.

Table 150. ihRawData Table (continued)

Column Name	Data Type	Description
		This 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.
`FilterComparisonMode`	`VT_BSTR`	The type of comparison to be made on the filter comparison value: • `Equal`: Filter condition is `True` when the `FilterTag` value is equal to the comparison value. • `EqualFirst`: Filter condition is `True` when the `FilterTag` value is equal to the first comparison value. • `EqualLast`: Filter condition is `True` when the `FilterTag` value is equal to the last comparison value. • `NotEqual`: Filter condition is `True` when the `FilterTag` value is not equal to the comparison value. • `LessThan`: Filter condition is `True` when the `FilterTag` value is less than the comparison value. • `GreaterThan`: Filter condition is `True` when the `FilterTag` value is greater than the comparison value. • `LessThanEqual`: Filter condition is `True` when the `FilterTag` value is less than or equal to the comparison value. • `GreaterThanEqual`: Filter condition is `True` when the `FilterTag` value is greater than or equal to the comparison value. • `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`. • `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`. • `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`. • `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`. 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`.

Table 150. ihRawData Table (continued)

Column Name	Data Type	Description
`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: • `AND` • `OR` • Combination of `AND` and `OR` `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: • Client • Server • 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.
`timestamp`	`VT_DBTimeStamp`	The date and time for the data sample (based on the timestamp of collector)
`timestampseconds`	`VT_I4`	The date and time for the data sample.
`microseconds`	`VT_I4`	The microsecond interval for the data sample.
`samplingmode`	`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

Column Name	Data Type	Description
`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 151. ihComments Table

Column Name	Data Type	Description
`Tagname`	`VT_BSTR`	Tagname property of the tag. **Note:** 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 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: • `String` • `Read-only` • `Optional`
`SamplingMode`	`VT_BSTR`	The mode used to sample data from the archive:

Table 151. ihComments Table (continued)

Column Name	Data Type	Description
		- `CurrentValue`: Retrieves the current value. Time frame criteria are ignored. - `Interpolated`: Retrieves evenly spaced interpolated values based on interval or `NumberOfSamples` and time frame criteria. - `RawByTime`: Retrieves raw archive values based on time frame criteria. - `RawByNumber`: Retrieves raw archive values based on the `StartTime`, `NumberOfSamples`, and `Direction` criteria. **Note:** `EndTime` criteria are ignored for this sampling mode. - `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. - `Calculated`: Retrieves evenly spaced calculated values based on `NumberOfSamples`, interval, time frame, and `CalculationMode` criteria. - `Lab`: Returns actual collected values without interpolation. - `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 period does not evenly divide by the interval length, Historian ignores any leftover

Table 151. ihComments Table (continued)

Column Name	Data Type	Description
		values at the end instead of putting them into a smaller interval. • `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. • `InterpolatedtoRaw`: Provides raw data in place of interpolated data when the number of samples requested is less than the number of available samples. • `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. • `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

Table 151. ihComments Table (continued)

Column Name	Data Type	Description
		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. • `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. 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. **Note:** 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. **Note:** The `NumberofSamples` and `IntervalMilliseconds` columns are mutually exclusive.

Table 151. ihComments Table (continued)

Column Name	Data Type	Description
		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: • `ExactTime`: Retrieves data for the exact times that the filter condition is `True`. • `BeforeTime`: Retrieves data from the time of the last `False` filter condition to the time of the next `True` condition. • `AfterTime`: Retrieves data from the time of the last `True` filter condition to the time of the next `False` condition. • `BeforeAndAfterTime`: Retrieves data from the time of the last `False` filter condition to the time of the next `False` condition. 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 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:

Table 151. ihComments Table (continued)

Column Name	Data Type	Description
		- `Equal`: Filter condition is `True` when the `FilterTag` value is equal to the comparison value. - `EqualFirst`: Filter condition is `True` when the `FilterTag` value is equal to the first comparison value. - `EqualLast`: Filter condition is `True` when the `FilterTag` value is equal to the last comparison value. - `NotEqual`: Filter condition is `True` when the `FilterTag` value is not equal to the comparison value. - `LessThan`: Filter condition is `True` when the `FilterTag` value is less than the comparison value. - `GreaterThan`: Filter condition is `True` when the `FilterTag` value is greater than the comparison value. - `LessThanEqual`: Filter condition is `True` when the `FilterTag` value is less than or equal to the comparison value. - `GreaterThanEqual`: Filter condition is `True` when the `FilterTag` value is greater than or equal to the comparison value. - `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`. - `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`. - `AnyBitNotSet`: Filter condition is `True` when the binary `FilterTag` value is not equal to any one of the bits in the condition. It is rep-

Table 151. ihComments Table (continued)

Column Name	Data Type	Description
		resented as `!~` to be used in `FilterExpression`. - `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`. 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: - `AND` - `OR` - Combination of `AND` and `OR` 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:

Table 151. ihComments Table (continued)

Column Name	Data Type	Description
		• Client • Server • 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 152. 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: • `CurrentValue`: Retrieves the current value. Time frame criteria are ignored. • `Interpolated`: Retrieves evenly spaced interpolated values based on interval or `NumberOfSamples` and time frame criteria. • `RawByTime`: Retrieves raw archive values based on time frame criteria. • `RawByNumber`: Retrieves raw archive values based on the `StartTime`, `NumberOfSamples`, and `Direction` criteria. **Note:** `EndTime` criteria are ignored for this mode.

Table 152. IhTrend Table (continued)

Column Name	Data Type	Description
		• `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. • `Calculated`: Retrieves evenly spaced calculated values based on `NumberOfSamples`, interval, time frame, and `CalculationMode` criteria. • `Lab`: Returns actual collected values without interpolation. • `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. • `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. • `InterpolatedtoRaw`: Provides raw data in place of interpolated data when the number of samples is less than the number of available samples. • `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 point this mode returns all available raw data points with no further processing. This mode is therefor used instead of `Trend` when the number of actual data samples is less than the requested number of samples. • `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. • `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.

Table 152. IhTrend Table (continued)

Column Name	Data Type	Description
`NumberOfSamples`	`VT_I4`	Number of samples to retrieve from the archive. Samples will be evenly spaced within the start and end times defined for most sampling modes. For th `RawByNumber` mode, this column determines the maximum number of values to retrieve. For the `RawByTime` mode, this column is ignored. **Note:** 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. **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 t perform on archive data: • `Average` • `Count` • `Maximum` • `MaximumTime` • `Minimum` • `MinimumTime` • `StandardDeviation` • `Total` • `RawAverage` • `RawStandardDeviation` • `RawTotal` • `TimeGood` • `FirstRawValue` • `FirstRawTime`

Table 152. IhTrend Table (continued)

Column Name	Data Type	Description
		- `LastRawValue` - `LastRawTime` - `TagStats`
`FilterTag`	`VT_BSTR`	Tagname used to define the filter, if specified. Only a single tag can be specified. Wildcards are not supported.
`FilterMode`	`VT_BSTR`	The type of time filter: - `ExactTime`: Retrieves data for the exact times that the filter condition is `True`. - `BeforeTime`: Retrieves data from the time of the last `False` filter condition to the time of the next `True` condition. - `AfterTime`: Retrieves data from the time of the last `True` filter condition to the time of the next `False` condition. - `BeforeAndAfterTime`: Retrieves data from the time of the last `False` filter condition to the time of the next `False` condition. This value 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 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: - `Equal`: Filter condition is `True` when the `FilterTag` value is equal to the comparison value. - `EqualFirst`: Filter condition is `True` when the `FilterTag` value is equal to the first comparison value. - `EqualLast`: Filter condition is `True` when the `FilterTag` value is equal to the last comparison valu - `NotEqual`: Filter condition is `True` when the `FilterTag` value is not equal to the comparison value. - `LessThan`: Filter condition is `True` when the `FilterTag` value is less than the comparison value. - `GreaterThan`: Filter condition is `True` when the `FilterTag` value is greater than the comparison value. - `LessThanEqual`: Filter condition is `True` when the `FilterTag` value is less than or equal to the comparison value.

Table 152. IhTrend Table (continued)

Column Name	Data Type	Description
		- `GreaterThanEqual`: Filter condition is `True` when the `FilterTag` value is greater than or equal to th comparison value. - `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`. - `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`. - `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`. - `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`. 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: - `AND` - `OR` - Combination of `AND` and `OR` 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: - Client - Server - Explicit bias number (number of minutes from GMT)
`Daylight-`	`VT_BOOL`	Indicates whether Daylight Saving Time logic should be applied to timestamps.

Table 152. IhTrend 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 lim to the number of rows returned.
`*TagID*.Value`	`VT_VARIANT`	The value of the data for the specified tag ID.
`*TagID*.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. For raw sampled data, data values are: • `Good` • `Bad` • `Uncertain` • `Not Available` • `Really Unknown` This column also includes the subquality of the data value, if it exists: • `NonSpecific` • `ConfigError` • `NotConnected` • `DeviceFail` • `SensorFail` • `LastKnownValue` • `CommFailure` • `OutOfService` • `ScaledOutOfRange` • `OffLine` • `NoValue` • `Really Unknown`

Table 152. IhTrend Table (continued)

Column Name	Data Type	Description
`*TagID*.Tagname`	`VT_BSTR`	Tagname property of the specified tag ID.
`*TagID*.Description`	`VT_BSTR`	User description for the specified tag ID.
`*TagID*.EngUnits`	`VT_BSTR`	Engineering unit description for the specified tag ID.
`*TagID*.Comment`	`VT_BSTR`	User comment associated with the specified tag ID.
`*TagID*.DataType`	`VT_BSTR`	The data type for the specified tag ID: • `Scaled` • `SingleFloat` • `DoubleFloat` • `SingleInteger` • `DoubleInteger` • `QuadInteger` • `UnsignedSingleInteger` • `UnsignedDoubleInteger` • `UnsignedQuadInteger` • `FixedString` • `VariableString` • `Byte` • `Boolean` • `BLOB` • `Time` • `Undefined`

Table 152. IhTrend Table (continued)

Column Name	Data Type	Description
		The data type returned in this column is the data type that you defined in Historian Administrator application.
`*TagID*.FixedStringLength`	`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.
`*TagID*.CollectorName`	`VT_BSTR`	The name of the collector responsible for collecting data for the specified tag ID.
`*TagID*.SourceAddress`	`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`).
`*TagID*.CollectionType`	`VT_BSTR`	Type of collection used to acquire data for the tag: • **Unsolicited:** The collector accepts data from the source whenever the source presents the data • **Polled:** The collector acquires data from a source on a periodic schedule determined by the collector. **Note:** Not all collectors support unsolicited collection.
`*TagID*.CollectionInterval`	`VT_I4`	The time interval, in milliseconds, between readings of data from this tag. For polled collection, this field represents the time between samples. For unsolicited collection, this fie represents the minimum time allowed between samples.
`*TagID*.CollectionOffset`	`VT_I4`	The time shift from midnight, in milliseconds, for collection of data from this tag.

Table 152. IhTrend Table (continued)

Column Name	Data Type	Description
`*Tag-ID*.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 for the specified tag ID. This is sometimes called phase shifting.
`*Tag-ID*.TimeStampType`	`VT_BSTR`	The timestamp type applied to data samples at collection time: • `Source`: The source delivers the timestamp along with the data sample. • `Collector`: The collector delivers the timestamp along with the collected data.
`*Tag-ID*.HiEngineeringUnits`	`VT_R8`	The high end of the engineering units range. Used only for scaled data types and input scaled tags.
`*Tag-ID*.LoEngineeringUnits`	`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. 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.
`*Tag-ID*.LoScale`	`VT_R8`	The low-end value of the input scaling range used for the tag.

Table 152. IhTrend Table (continued)

Column Name	Data Type	Description
`TagID.CollectorCompression`	`VT_BOOL`	Indicates whether collector compression is enabled for the specified tag ID. 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 th archiver) any new value that falls outside the deadband and then centers the deadband around the nev value.
`TagID.CollectorDeadbandPercentRange`	`VT_R4`	The current value of the compression deadband.
`TagID.ArchiveCompression`	`VT_BOOL`	Indicates whether archive collector compression is enabled for the tag.
`TagID.ArchiveDeadbandPercentRange`	`VT_R4`	The current value of the archive compression deadband.
`TagID.CollectorGeneral1`	`VT_BSTR`	The general (or spare) configuration fields for the specified tag ID.
`TagID.CollectorGeneral2`	`VT_BSTR`	The general (or spare) configuration fields for the specified tag ID.

Table 152. IhTrend Table (continued)

Column Name	Data Type	Description
TagID.CollectorGeneral3	VT_BSTR	The general (or spare) configuration fields for the specified tag ID.
TagID.CollectorGeneral4	VT_BSTR	The general (or spare) configuration fields for the specified tag ID.
TagID.CollectorGeneral5	VT_BSTR	The general (or spare) configuration fields for the specified tag ID.
TagID.ReadSecurityGroup	VT_BSTR	The name of the Windows security group that controls the reading of data for the specified tag ID. Refer to "Implementing Historian Security" in the *Getting Started with Historian* manual for definitions o the various security levels and groups.
TagID.WriteSecurityGroup	VT_BSTR	The name of the Windows security group that controls the writing of data for the specified tag ID. Refer to "Implementing Historian Security" in the *Getting Started with Historian* manual for definitions o the various security levels and groups.
TagID.AdministratorSecurityGroup	VT_BSTR	The name of the Windows security group responsible for controlling configuration changes for the specified tag ID.
TagID.Calculation	VT_BSTR	The equation for the calculation performed for the specified tag ID.

Table 152. IhTrend Table (continued)

Column Name	Data Type	Description
TagID.LastModified	VT_DBTimeStamp	The date and time that the tag configuration was last modified. The time structure includes milliseconds.
TagID.LastModifiedUser	VT_BSTR	The username of the Windows user who last modified the tag configuration.
TagID.CollectorType	VT_BSTR	The type of collector responsible for collecting data for the specified tag ID: • Undefined • iFIX • Simulation • OPC • File • iFIXLabData • ManualEntry • Simulation • Other
TagID.StoreMilliseconds	VT_BOOL	Indicates whether time resolution in milliseconds is enabled for the specified tag ID. 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.
TagID.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. UTC is the international time standard, the current term for what was commonly referred to as Greenwich Mean Time (GMT).
TagID.AverageCollectionTime	VT_I4	The average time it takes to execute the calculation tag since you started the Calculation collector for the specified tag ID.

Table 152. IhTrend Table (continued)

Column Name	Data Type	Description
TagID.CollectionDisabled	VT_I4	Indicates whether collection is enabled (0) or disabled (1) for the specified tag ID. The default setting is enabled (0).
TagID.CollectorCompressionTimeout	VT_I4	Indicates the maximum amount of time the collector will wait between sending samples to the archive This time is kept per tag, as different tags report to the archiver at different times. This value should be in increments of your collection interval, and not less. Ideally, this field is used for polled data values. It can be used with unsolicited data, but when you do s 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.
TagID.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.
TagID.InterfaceAbsoluteDeadbanding	VT_BOOL	Indicates whether absolute collector deadband is enabled for the specified tag ID.
TagID.InterfaceAbsoluteDeadband	VT_R8	Indicates the value for absolute collector deadband.
TagID.Archive-	VT_BOOL	Indicates whether absolute archive deadband is enabled for the specified tag ID.

Table 152. IhTrend Table (continued)

Column Name	Data Type	Description
`AbsoluteDeadbanding`		
TagID`.ArchiveAbsoluteDeadband`	`VT_R8`	Indicates the value for absolute archive deadband.
TagID`.SpikeLogic`	`VT_BOOL`	Indicates whether Spike Logic is enabled on the collector.
TagID`.SpikeLogicOverride`	`VT_BOOL`	Indicates whether the Spike Logic setting for the specified tag ID overrides the collector setting (`True`) 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 1080)* 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 153. ihQuerySettings Table

Column Name	Data Type	Description
`StartTime`	`VT_DBTimeStamp`	The start time of the query. This represents the earliest timestamp for any tag contained in the query. If no `StartTime` value is specified, the start time is two hours prior to execution of the query.
`EndTime`	`VT_DBTimeStamp`	The end time of the query. This represents the latest timestamp for any tag contained in the query. If no `EndTime` value is specified, the end time is the time that you execute the query.
`SamplingMode`	`VT_BSTR`	The mode of sampling data from the archive: • `CurrentValue`: Retrieves the current value. Time frame criteria are ignored. • `Interpolated`: Retrieves evenly spaced interpolated values based on interval or `NumberOfSamples` and time frame criteria. • `RawByTime`: Retrieves raw archive values based on time frame criteria. • `RawByNumber`: Retrieves raw archive values based on the `StartTime`, `NumberOfSamples`, and `Direction` criteria.

Table 153. ihQuerySettings Table (continued)

Column Name	Data Type	Description
		Note: `EndTime` criteria are ignored for this mode. • `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. • `Calculated`: Retrieves evenly spaced calculated values based on `NumberOfSamples`, interval, time frame, and `CalculationMode` criteria. • `Lab`: Returns actual collected values without interpolation. • `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. • `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. • `InterpolatedtoRaw`: Provides raw data in place of interpolated data when the number of samples is less than the number of available samples. • `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. • `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

Table 153. ihQuerySettings Table (continued)

Column Name	Data Type	Description
		therefore used instead of `Trend2` when the number of actual data samples is less than the requested number of samples. • `LabtoRaw`: Provides raw data for the selected calculated data when the number of requested samples is less than the number of available samples. `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. 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. **Note:** 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. **Note:** The `NumberofSamples` and `IntervalMilliseconds` columns are mutually exclusive. If `IntervalMilliseconds` is used, `NumberofSamples` 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: • `Average` • `Count` • `Maximum` • `MaximumTime`

Table 153. ihQuerySettings Table (continued)

Column Name	Data Type	Description
		- Minimum - MinimumTime - OPCQOr and OPCQAnd - StandardDeviation - StateCount - StateTime - Total - RawAverage - RawStandardDeviation - RawTotal - TimeGood - FirstRawValue - FirstRawTime - LastRawValue - LastRawTime - TagStats The default value is Average.
FilterTag	VT_BSTR	Tagname used to define the filter, if specified. Only a single tag can be specified. Wildcards are not supported.
FilterMode	VT_BSTR	The type of time filter: - ExactTime: Retrieves data for the exact times that the filter condition is True. - BeforeTime: Retrieves data from the time of the last False filter condition to the time of the next True condition. - AfterTime: Retrieves data from the time of the last True filter condition to the time of the next False condition. - BeforeAndAfterTime: Retrieves data from the time of the last False filter condition to the time of the next False condition.

Table 153. ihQuerySettings Table (continued)

Column Name	Data Type	Description
		This value 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 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: • `Equal`: Filter condition is `True` when the `FilterTag` value is equal to the comparison value. • `EqualFirst`: Filter condition is `True` when the `FilterTag` value is equal to the first comparison value. • `EqualLast`: Filter condition is `True` when the `FilterTag` value is equal to the last comparison value. • `NotEqual`: Filter condition is `True` when the `FilterTag` value is not equal to the comparison value. • `LessThan`: Filter condition is `True` when the `FilterTag` value is less than the comparison value. • `GreaterThan`: Filter condition is `True` when the `FilterTag` value is greater than the comparison value. • `LessThanEqual`: Filter condition is `True` when the `FilterTag` value is less than or equal to the comparison value. • `GreaterThanEqual`: Filter condition is `True` when the `FilterTag` value is greater than or equal to the comparison value. • `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`. • `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`. • `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`. • `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`. 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`.

Table 153. ihQuerySettings Table (continued)

Column Name	Data Type	Description
`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: - `AND` - `OR` - Combination of `AND` and `OR` 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: - Client - Server - 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. 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:

Table 153. ihQuerySettings Table (continued)

Column Name	Data Type	Description
		- `Alarms` - `Alarm_History` - `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 154. 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.
`DependentTagname`	`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 155. 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 th 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 Historian.
`AlarmType`	`VT_BSTR`	The alarm type:

Table 155. ihAlarms Table (continued)

Column Name	Data Type	Description
		- `Alarms`: In Historian, the full life cycle of an alarm is stored as a single record in the alarm arch - `Alarm_History`: The separate transitions for all alarms. One row per transition is returned. - `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` v ue comprises an alarm.
`SubCondition`	`VT_BSTR`	The OPC subcondition of the alarm. Does not apply to event data. This value represents the state of 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.
`AckTime`	`VT_DBTimeStamp`	The time the alarm was acknowledged. Does not apply to event data.
`Microseconds`	`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:

Table 155. ihAlarms Table (continued)

Column Name	Data Type	Description
		- Client - Server - 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 returned by the current query.
`User-Defined Variable #X`	`VT_VARIANT`	User-defined variables. This is a dynamic list of columns that varies based on the collectors running 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 156. ihEnumeratedSets Table

Column Name	Data Type	Description
`SetName`	`VT_BSTR`	The name of the set.

Table 156. ihEnumeratedSets Table (continued)

Column Name	Data Type	Description
`Description`	`VT_BSTR`	The description of the set.
`NumberofStates`	`VT_I4`	The number of states a set contains.
`NumberofTagReferences`	`VT_I4`	The number of tags with which a set is associated.
`SetDataType`	`VT_BSTR`	The data type of the set.
`AdministratorSecurityGroup`	`VT_BSTR`	The security group to which the set belongs.
`LastModifiedUser`	`VT_BSTR`	Indicates which user last modified the set.
`LastModi-`	`VT_DB-`	Indicates the last time the set was modified.

Table 156. ihEnumeratedSets Table (continued)

Column Name	Data Type	Description
`fiedTime`	`TimeStamp`	
`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 157. ihEnumeratedStates Table

Column Name	Data Type	Description
`SetName`	`VT_BSTR`	The name of the set.
`Description`	`VT_BSTR`	The description of the set.

Table 157. ihEnumeratedStates Table (continued)

Column Name	Data Type	Description
`NumberofStates`	`VT_I4`	The number of states a set contains.
`NumberofTagReferences`	`VT_I4`	The number of tags with which a set is associated.
`SetDataType`	`VT_BSTR`	The data type of the set.
`AdministratorSecurityGroup`	`VT_BSTR`	The security group to which the set belongs.
`LastModifiedUser`	`VT_BSTR`	Indicates which user last modified the set.
`LastModifiedTime`	`VT_DBTimeStamp`	Indicates the last time the set was modified.

Table 157. ihEnumeratedStates Table (continued)

Column Name	Data Type	Description
`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 158. 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.

Table 158. ihUserDefinedTypes Table (continued)

Column Name	Data Type	Description
`StoreFieldQuality`	`VT_BOOL`	Indicates whether the field-level quality is stored.
`NumberofFields`	`VT_I4`	The number of fields a user-defined type contains.
`NumberofTagReferences`	`VT_I4`	The number of tags with which a user-defined type is associated.
`AdministratorSecurityGroup`	`VT_BSTR`	The security group to which the user-defined type belongs.
`LastModifiedUser`	`VT_BSTR`	Indicates which user last modified the user-defined type.
`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 159. 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.
`MasterField`	`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 13. Using Historian Administrator

Historian Administrator

Introduction to 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.
- 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.
- Perform archive maintenance, including:
  - Setting archive size.
  - Selecting options and parameters.
  - Accessing security parameters.
  - Adding archives.
- 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.
    - Displaying performance trends for selected collectors.

> **Note:**
> You can back up and restore archives directly using Elastic File System (EFS). If you try to back up archives using Historian Administrator, and error occurs.

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 Using the Installer *(on page 1195)*.
- Create a Windows user on the Historian server *(on page 120)*.
- 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, collector maintenance, tag maintenance, and help pages.
- **Tag Maintenance**: Contains tag names, parameters, and controls.
- **Collector Maintenance**: Contains collector names, parameters, and controls.
- **Data Store Maintenance**: Contains archive names, parameters, alarms, security, and controls.
- **Message Search**: Not applicable

Installing Historian Administrator

Install Historian Administrator Using the Installer

If you already have Historian Administrator on your machine (installed using on-premises Proficy Historian), you can just change the destination to the NLB DNS, and begin using it:

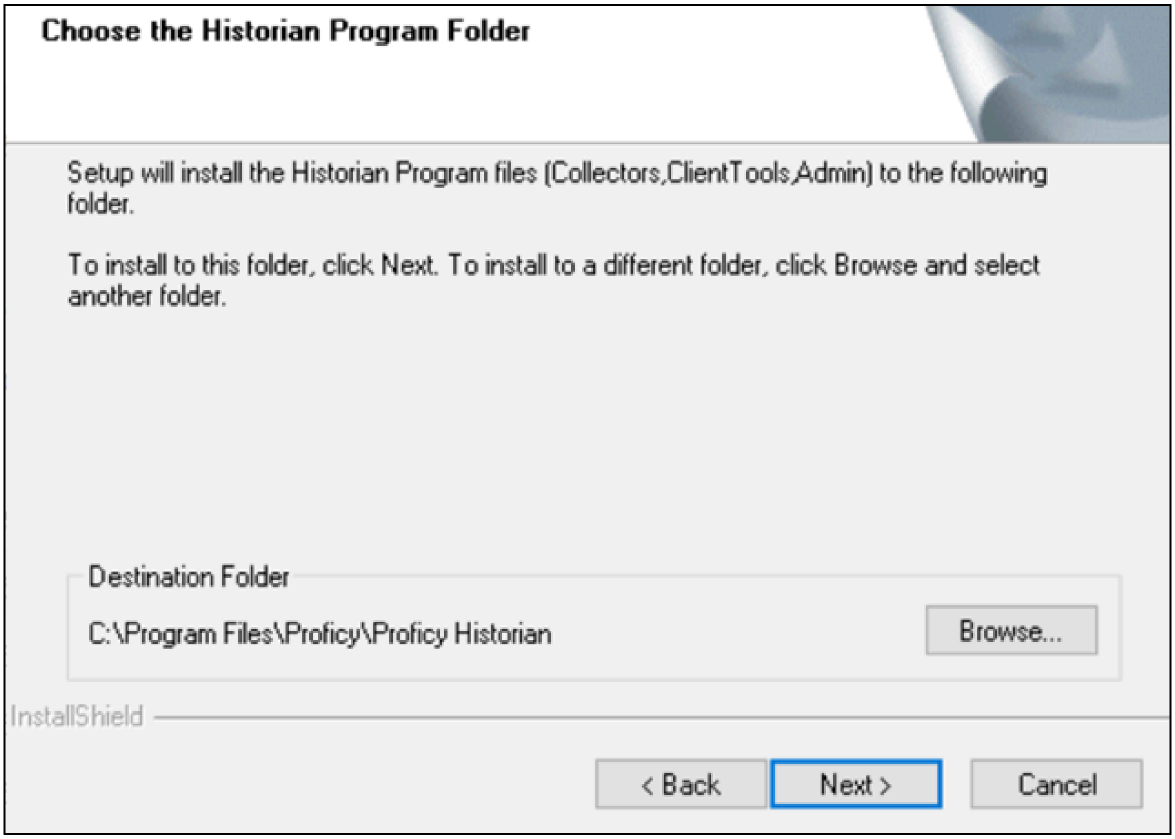
1. Select **Main**.

   A login window appears.

2. Provide the NLB DNS, username, password, and domain information, and then select **OK**.

This topic describes how to install Historian Administrator using the installer. You can also install it at a command prompt *(on page 1197)*.

1. Run the `InstallLauncher.exe` file. Contact the support team for this installer.
2. Select **Install Client Tools**.
   The **Select Features** page appears, displaying a list of components.
3. Select the **Historian Administrator** check box.
4. Select **Next**.
   The **Choose the Historian Program Folder** page appears.



Choose the Historian Program Folder

Setup will install the Historian Program files (Collectors,ClientTools,Admin) to the following folder.

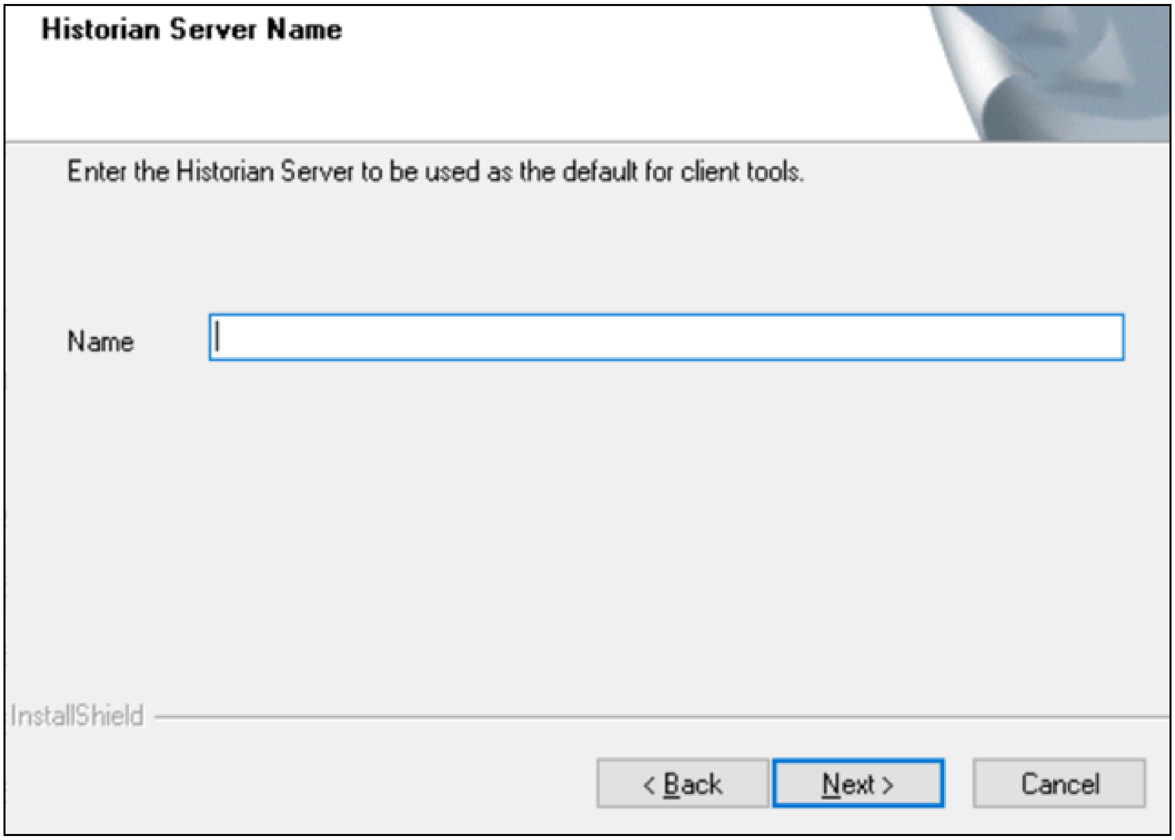
To install to this folder, click Next. To install to a different folder, click Browse and select another folder.

Destination Folder

C:\Program Files\Proficy\Proficy Historian

5. As needed, change the destination folder of Historian Administrator, or leave the default folder, and then select **Next**.

   The **Historian Server Name** page appears.



Historian Server Name

Enter the Historian Server to be used as the default for client tools.

Name

6. Enter the NLB DNS of Proficy Historian for AWS that you want to use with Historian Administrator, and then select **Next**.

> **Tip:**
>
> To find the NLB DNS:
>
> a. Access the EKS cluster on which you have deployed Proficy Historian for AWS.
> b. Access the EC2 instance.
> c. In the navigation pane, under **Load Balancing**, select **Load Balancers**.
> d. Select the load balancer for which you want to find the DNS.
> e. In the **Description** section, copy the DNS name.

7. When you are asked to reboot your system, select **Yes**.

Install Historian Administrator at a Command Prompt

Install Historian Administrator using the installer *(on page 1195)* 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 Historian Administrator at a command prompt on other machines.

1. Copy the `setup.iss` file to the machine on which you want to install Historian Administrator 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**.

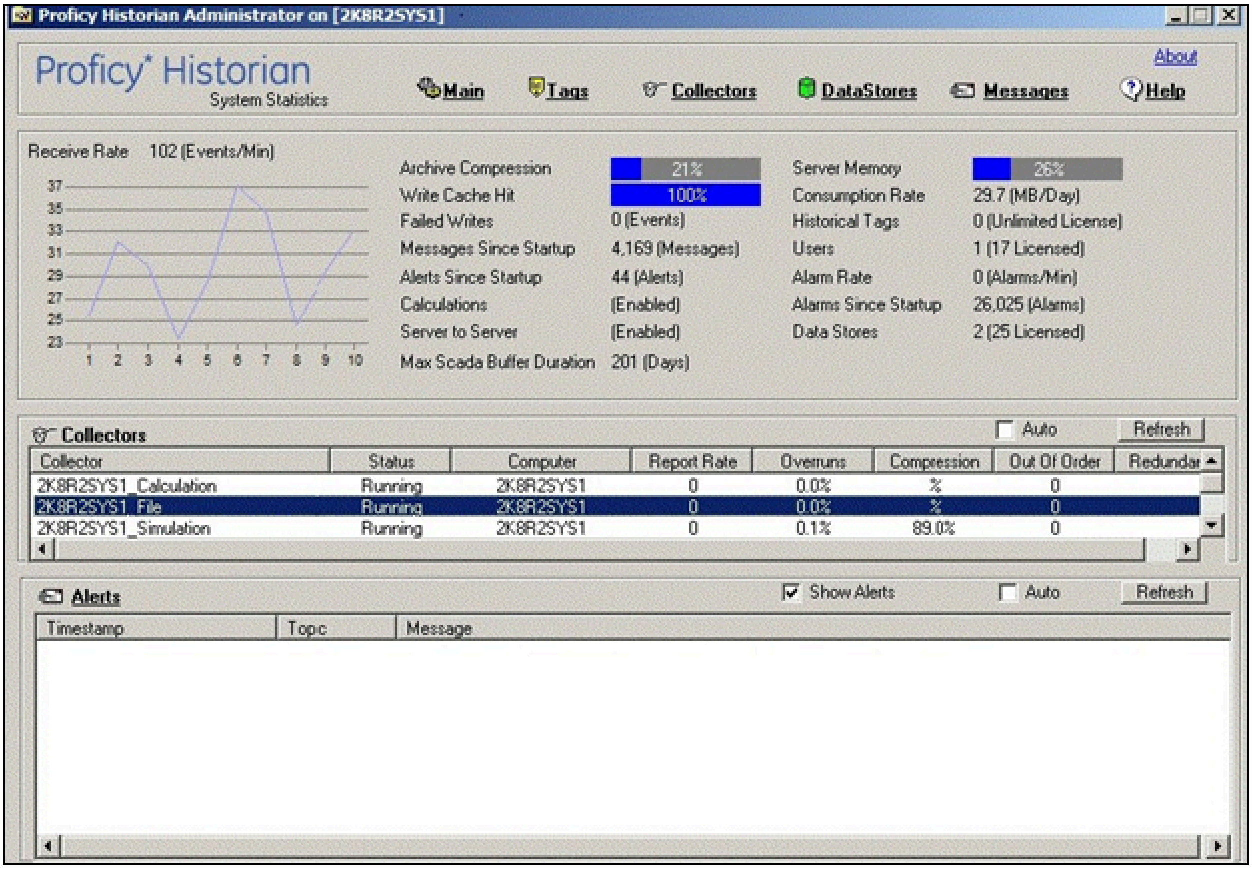
Historian Administrator is installed.

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 1199)*
- The **Collectors** section *(on page 1202)*



Receive Rate 102 (Events/Min)

Archive Compression	21%	Server Memory	26%
Write Cache Hit	100%	Consumption Rate	29.7 (MB/Day)
Failed Writes	0 (Events)	Historical Tags	0 (Unlimited License)
Messages Since Startup	4,169 (Messages)	Users	1 (17 Licensed)
Alerts Since Startup	44 (Alerts)	Alarm Rate	0 (Alarms/Min)
Calculations	(Enabled)	Alarms Since Startup	26,025 (Alarms)
Server to Server	(Enabled)	Data Stores	2 (25 Licensed)
Max Scada Buffer Duration	201 (Days)		

Collectors □ Auto Refresh

Collector	Status	Computer	Report Rate	Overruns	Compression	Out Of Order	Redundar
2K8R2SYS1_Calculation	Running	2K8R2SYS1	0	0.0%	%	0	
2K8R2SYS1_File	Running	2K8R2SYS1	0	0.0%	%	0	
2K8R2SYS1_Simulation	Running	2K8R2SYS1	0	0.1%	89.0%	0	

Alerts ☑ Show Alerts □ Auto Refresh

Timestamp	Topc	Message

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)	Not applicable
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. 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	Not applicable
Alerts Since Startup	Not applicable
Calculations	Not applicable
Server-to-Server	Displays the value **Enabled** if the Server-to-Server collector is licensed on the software key.
Alarms since Startup	Not applicable

Field	Description
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)	Not applicable
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. **Note:** 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. 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. 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

Field	Description
	are other conditions. For example, if the Historian server is busy, you will not be able to connect using this feature. **Note:** 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	Not applicable
SCADA Tags	Displays the number of CIMPLICITY or iFIX tags.
Tags Consumed by Arrays	Not applicable

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:

Field	Description
	• **Running**: Indicates that the collector is running. • **Stopped**: Indicates that the collector is not collecting data. • **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	Not applicable
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. 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). 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. 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.

Field	Description
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.
Redundancy	Not applicable

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**.

This topic describes the fields in each subsection in the **Archive Details** section.

Note:
You cannot back up and restore data using Historian Administrator. This is because the backup and restore functions are performed using Elastic File System (EFS).

The Status Subsection

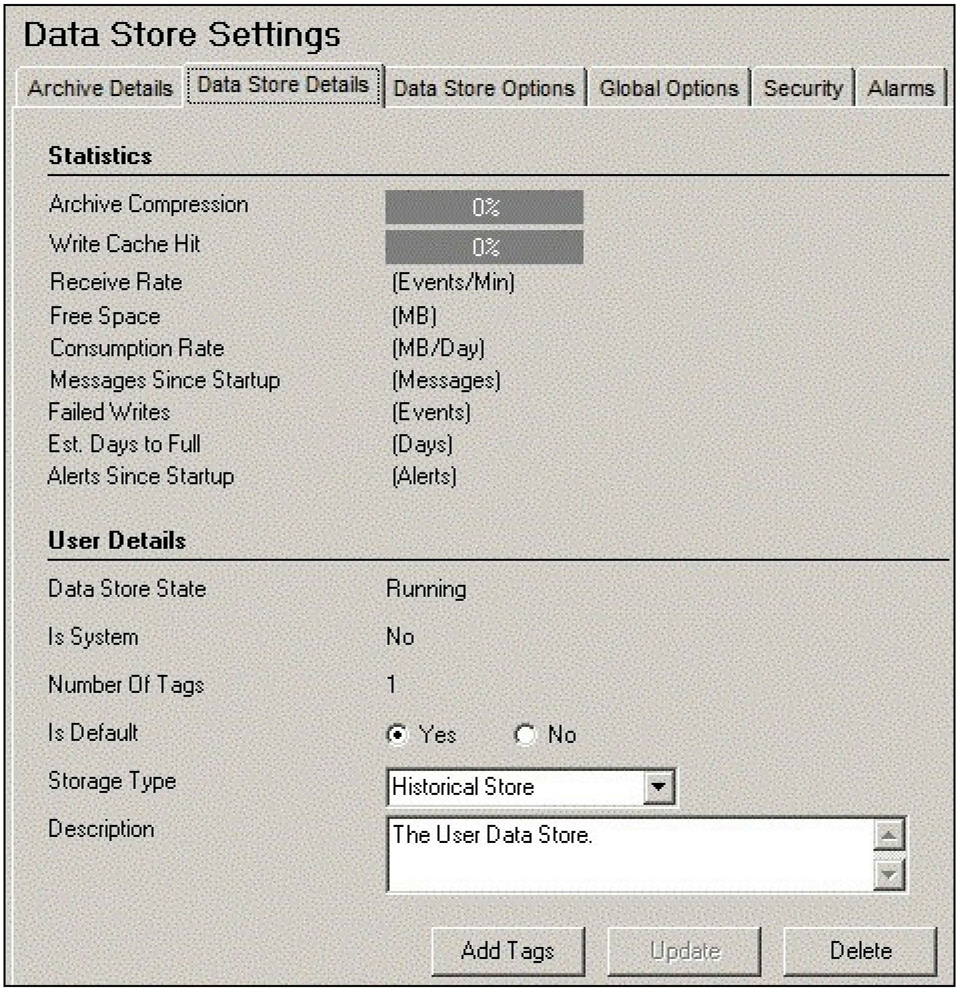
Field	Description
Status	The current operating state of the archive. This field contains one of the following values: • **Current**: Indicates that the archive is actively accepting data. • **Active**: Indicates that the archive contains data but is not currently accepting data. • **Empty**: Indicates 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 160. Resources

Field	Description
File Location	The path and name of the archive file.
File Size	The size (in MB) of the archive file. **Note:** 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. **Note:** 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.


Data Store Settings

Statistics

Archive Compression	0%
Write Cache Hit	0%
Receive Rate	(Events/Min)
Free Space	(MB)
Consumption Rate	(MB/Day)
Messages Since Startup	(Messages)
Failed Writes	(Events)
Est. Days to Full	(Days)
Alerts Since Startup	(Alerts)

User Details

Data Store State	Running
Is System	No
Number Of Tags	1
Is Default	● Yes ○ No
Storage Type	Historical Store
Description	The User Data Store.

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. 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

Field	Description
	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	Not applicable
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. 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.
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

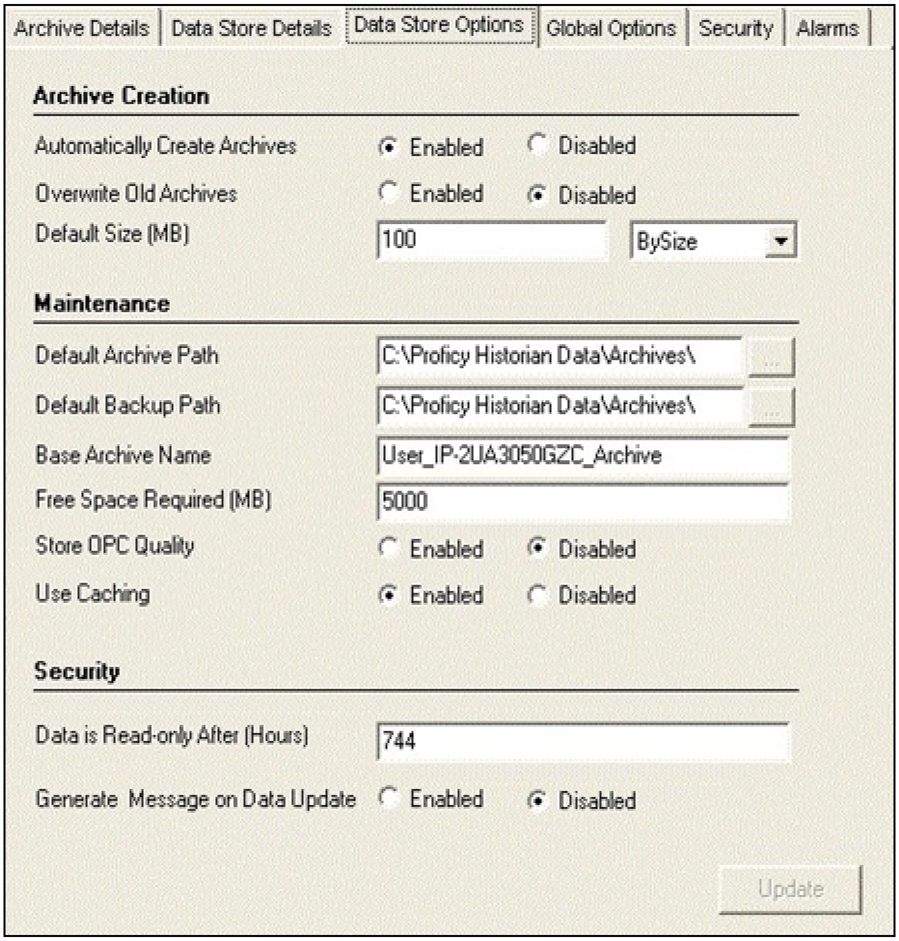
Field	Description
	days until full. After the archive is full, a new archive must be created (could be automatic). To increase this value, you must reduce the consumption rate. To ensure that collection is not interrupted, make sure that the **Automatically Create Archives** option is enabled in the **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	Not applicable

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. **Note:** 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**.
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.



Archive Creation

Automatically Create Archives ● Enabled ○ Disabled

Overwrite Old Archives ○ Enabled ● Disabled

Default Size (MB) 100 BySize

Maintenance

Default Archive Path C:\Proficy Historian Data\Archives\

Default Backup Path C:\Proficy Historian Data\Archives\

Base Archive Name User_IP-2UA3050GZC_Archive

Free Space Required (MB) 5000

Store OPC Quality ○ Enabled ● Disabled

Use Caching ● Enabled ○ Disabled

Security

Data is Read-only After (Hours) 744

Generate Message on Data Update ○ Enabled ● Disabled

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. **Note:** 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

Field	Description
	data, exercise caution in using this feature. We recommend that you back up the archive using EFS so that you can restore it later. **Note:** 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: • **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. • **Days**: A new archive file is created after the number of days that you specify in the **Archive Duration** field that will appear. • **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 200.
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. **Note:** 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	Not applicable
Base Archive Name	A prefix that you want to add to all the archive files.

Field	Description
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.
Store OPC Quality	Indicates whether to store the OPC data quality. **Note:** 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. **Note:** This option is not available for SCADA Buffer data stores.

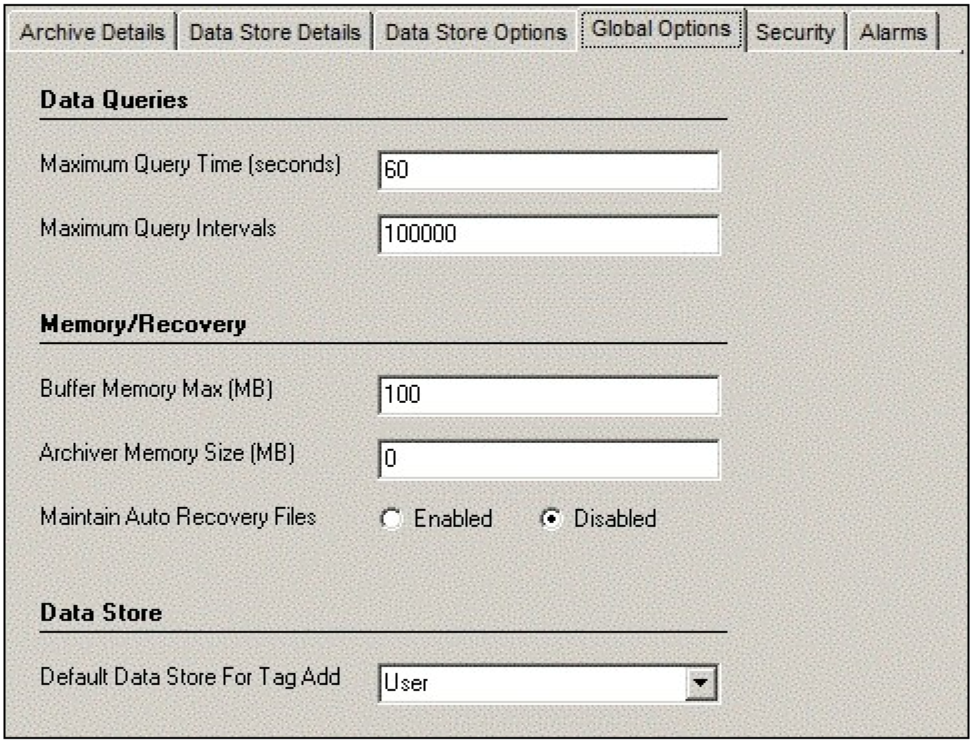
The Security Subsection

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. **Note:** 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.

Field	Description
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. **Note:** 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.



Data Queries

Maximum Query Time (seconds): 60

Maximum Query Intervals: 100000

Memory/Recovery

Buffer Memory Max (MB): 100

Archiver Memory Size (MB): 0

Maintain Auto Recovery Files: ○ Enabled ● Disabled

Data Store

Default Data Store For Tag Add: User

The Data Queries Subsection

Field	Description
Maximum Query Time (seconds)	Specifies the maximum time that a data point or 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.

Field	Description
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. 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. **Note:** • 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. • 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.

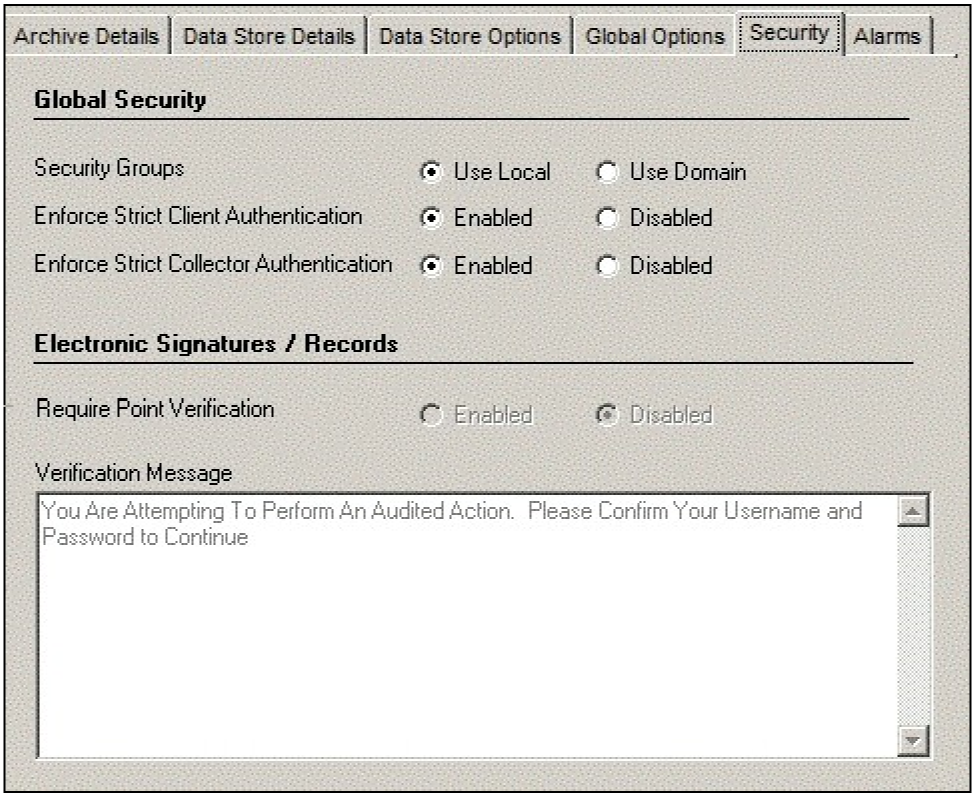
Field	Description
	Note: These files are managed internally by Historian, and should not be used as backup files. To create multiple archives at the same time, select the **Disabled** option. By default, this field is set to **Disabled** on a 64-bit operating system. On a large-scale system, we recommend 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



Global Security

Security Groups ● Use Local ○ Use Domain

Enforce Strict Client Authentication ● Enabled ○ Disabled

Enforce Strict Collector Authentication ● Enabled ○ Disabled

Electronic Signatures / Records

Require Point Verification ○ Enabled ● Disabled

Verification Message

You Are Attempting To Perform An Audited Action. Please Confirm Your Username and Password to Continue

.

The Global Security Subsection

Field	Description
Security Groups	Indicates whether to use the local security groups or the domain security groups. **Note:** 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.

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. **Note:** • The audit features are not dependent on this feature being enabled. Historian audits all user actions regardless of whether this option is enabled. • If you plan to create multiple archives at the same time, select the **Disabled** option. 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. **Note:** 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.

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.

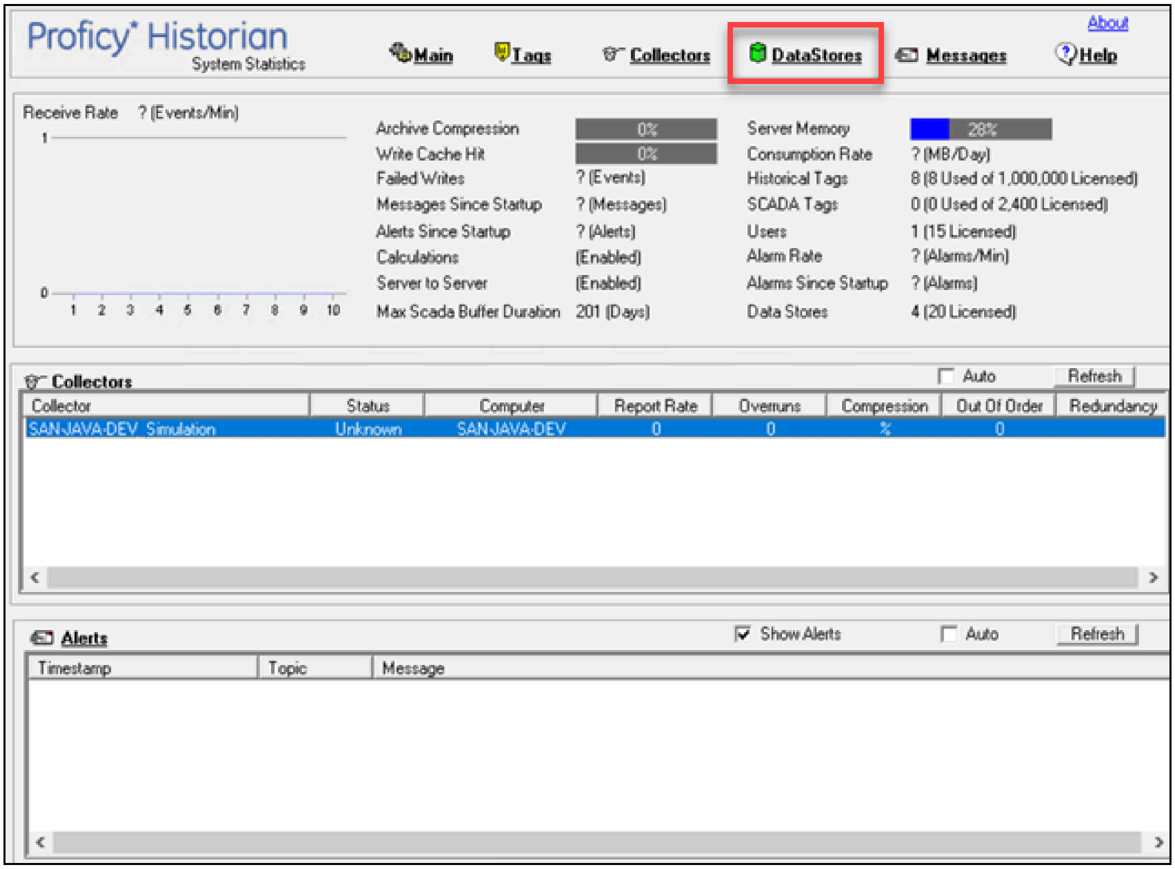
- **System**: Stores performance tags. This is only for internal usage within Historian, and you cannot add tags to this data store. You must not rename or delete the system data store.
- **User**: Stores tag data. This is a default data store. You can rename and delete a user data store as long as there 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 1194)*.
2. Select **DataStores**.

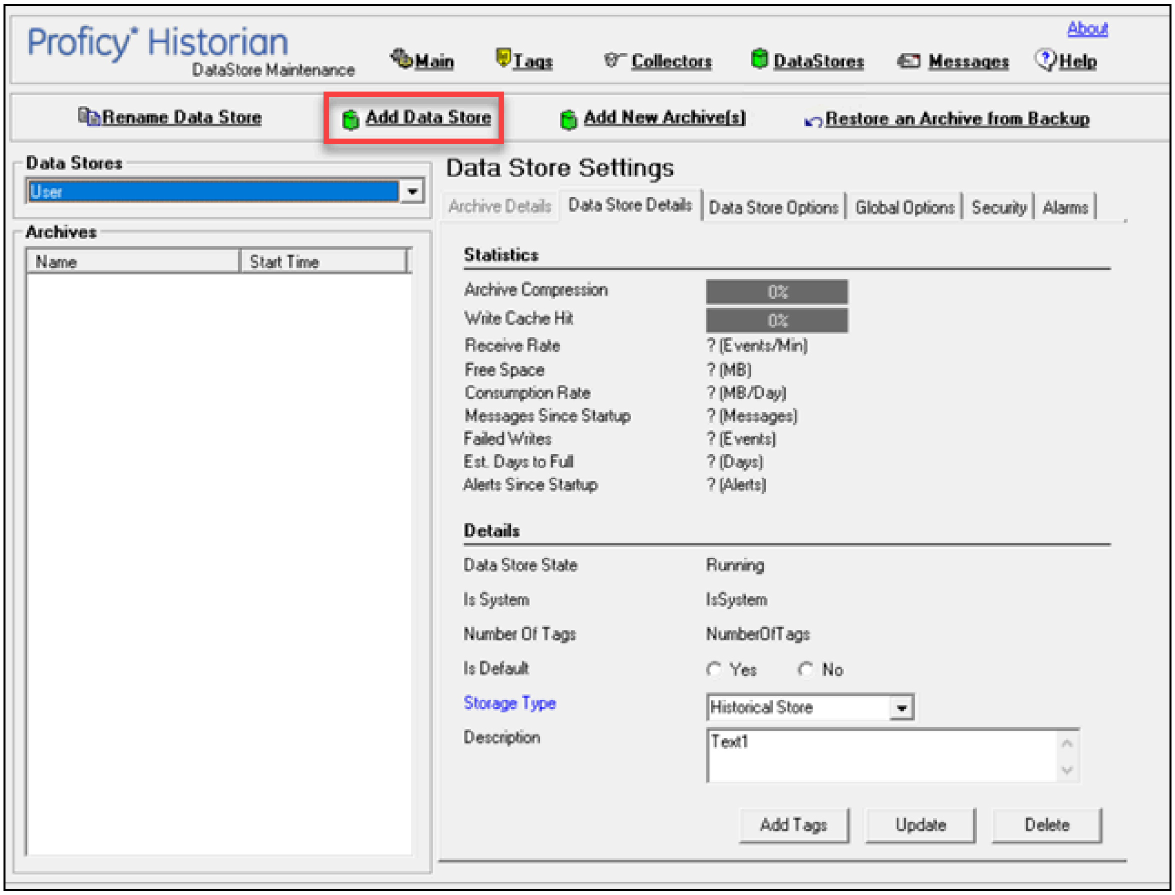


Archive Compression	0%	Server Memory	28%
Write Cache Hit	0%	Consumption Rate	? (MB/Day)
Failed Writes	? (Events)	Historical Tags	8 (8 Used of 1,000,000 Licensed)
Messages Since Startup	? (Messages)	SCADA Tags	0 (0 Used of 2,400 Licensed)
Alerts Since Startup	? (Alerts)	Users	1 (15 Licensed)
Calculations	(Enabled)	Alarm Rate	? (Alarms/Min)
Server to Server	(Enabled)	Alarms Since Startup	? (Alarms)
Max Scada Buffer Duration	201 (Days)	Data Stores	4 (20 Licensed)

Collector	Status	Computer	Report Rate	Overruns	Compression	Out Of Order	Redundancy
SAN-JAVA-DEV_Simulation	Unknown	SAN-JAVA-DEV	0	0	%	0	

Timestamp	Topic	Message

3. Select **Add Data Store**.



Statistics

Archive Compression 0%
Write Cache Hit 0%
Receive Rate ? (Events/Min)
Free Space ? (MB)
Consumption Rate ? (MB/Day)
Messages Since Startup ? (Messages)
Failed Writes ? (Events)
Est. Days to Full ? (Days)
Alerts Since Startup ? (Alerts)

Details

Data Store State Running
Is System IsSystem
Number Of Tags NumberOfTags
Is Default ○ Yes ○ No
Storage Type Historical Store
Description Text1

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.

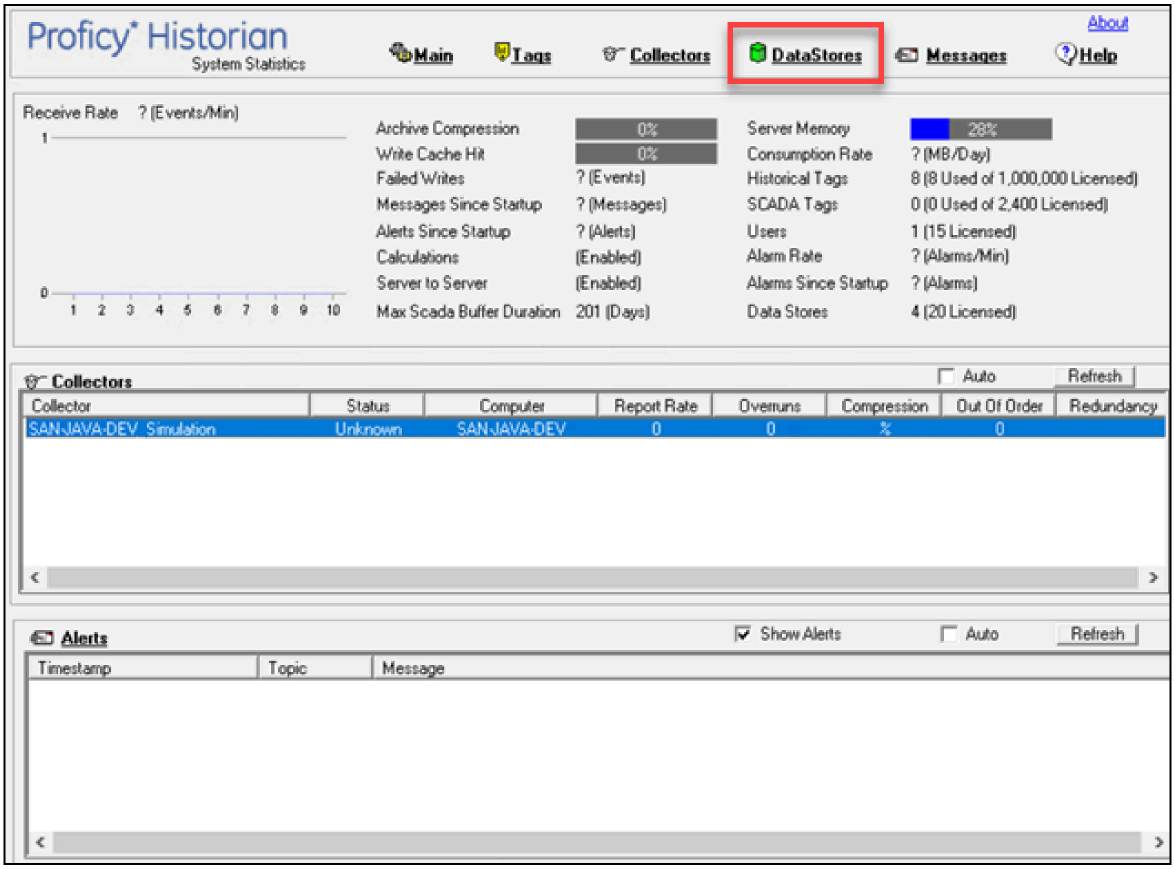
5. Select **OK**.

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 using EFS.

Rename a Data Store

1. Access Historian Administrator *(on page 1194)*.
2. Select **DataStores**.

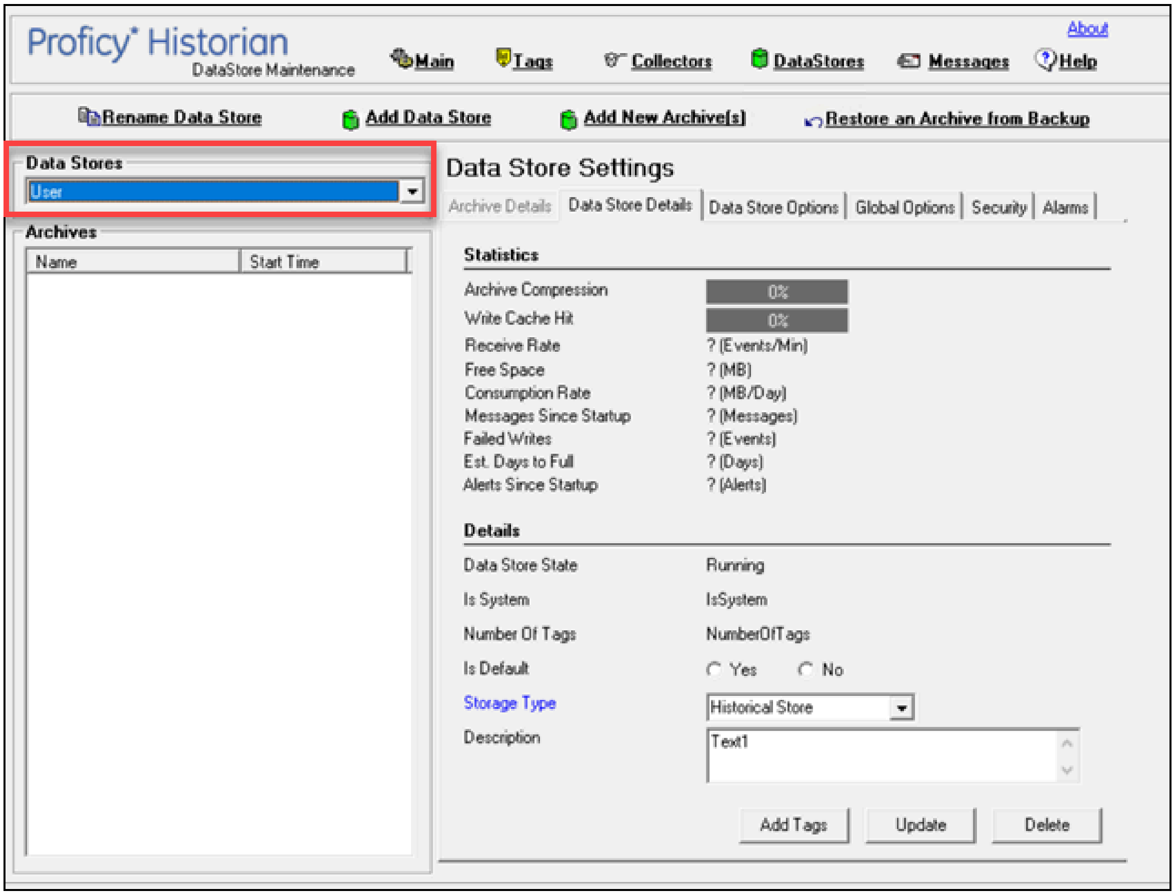


Archive Compression	0%	Server Memory	28%
Write Cache Hit	0%	Consumption Rate	? (MB/Day)
Failed Writes	? (Events)	Historical Tags	8 (8 Used of 1,000,000 Licensed)
Messages Since Startup	? (Messages)	SCADA Tags	0 (0 Used of 2,400 Licensed)
Alerts Since Startup	? (Alerts)	Users	1 (15 Licensed)
Calculations	(Enabled)	Alarm Rate	? (Alarms/Min)
Server to Server	(Enabled)	Alarms Since Startup	? (Alarms)
Max Scada Buffer Duration	201 (Days)	Data Stores	4 (20 Licensed)

Collector	Status	Computer	Report Rate	Overruns	Compression	Out Of Order	Redundancy
SAN-JAVA-DEV_Simulation	Unknown	SAN-JAVA-DEV	0	0	%	0	

Timestamp	Topic	Message

3. In the **Data Stores** field, select the data store that you want to rename.



Data Stores: User

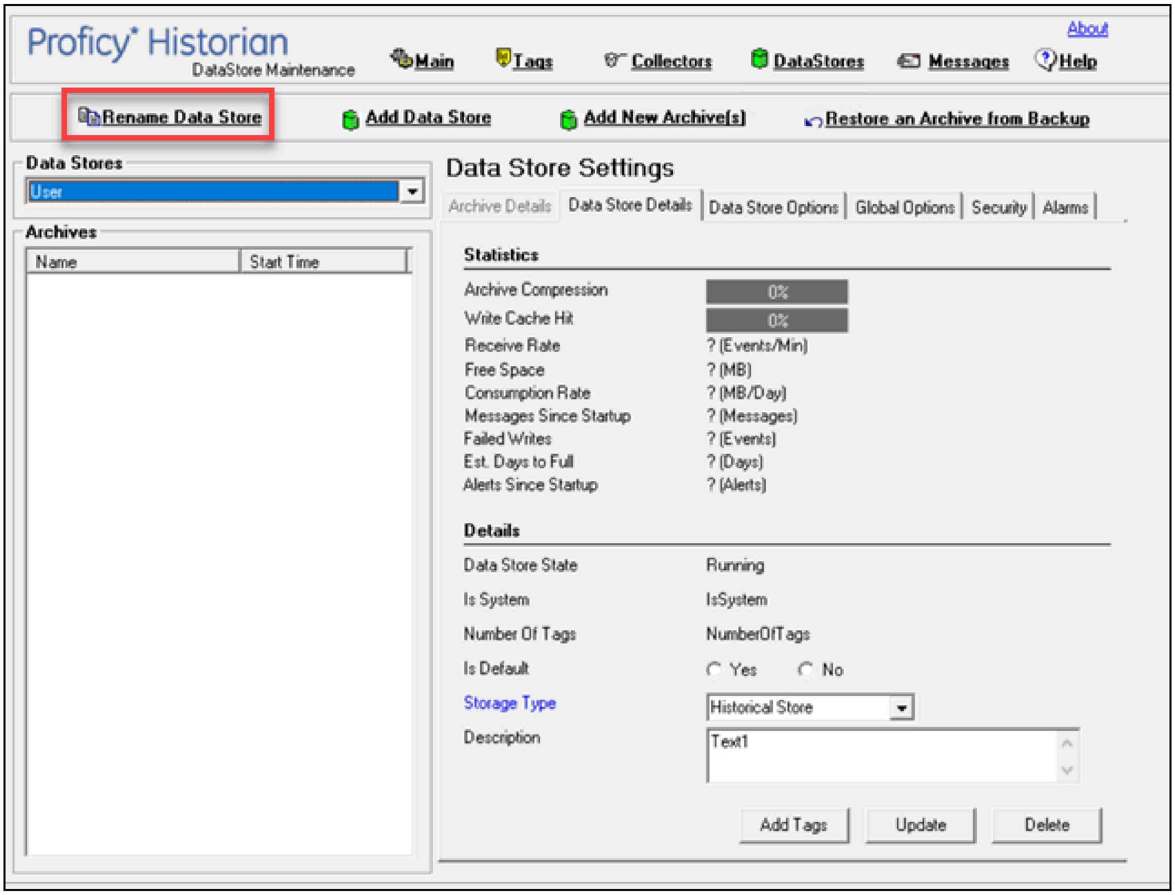
Statistics

Archive Compression	0%
Write Cache Hit	0%
Receive Rate	? (Events/Min)
Free Space	? (MB)
Consumption Rate	? (MB/Day)
Messages Since Startup	? (Messages)
Failed Writes	? (Events)
Est. Days to Full	? (Days)
Alerts Since Startup	? (Alerts)

Details

Data Store State	Running
Is System	IsSystem
Number Of Tags	NumberOfTags
Is Default	○ Yes ○ No
Storage Type	Historical Store
Description	Text1

4. Select **Rename Data Store**.



Data Store Settings

Data Stores: User

Statistics

Archive Compression	0%
Write Cache Hit	0%
Receive Rate	? (Events/Min)
Free Space	? (MB)
Consumption Rate	? (MB/Day)
Messages Since Startup	? (Messages)
Failed Writes	? (Events)
Est. Days to Full	? (Days)
Alerts Since Startup	? (Alerts)

Details

Data Store State	Running
Is System	IsSystem
Number Of Tags	NumberOfTags
Is Default	○ Yes ○ No
Storage Type	Historical Store
Description	Text1

The **Rename New Data Store** window appears.

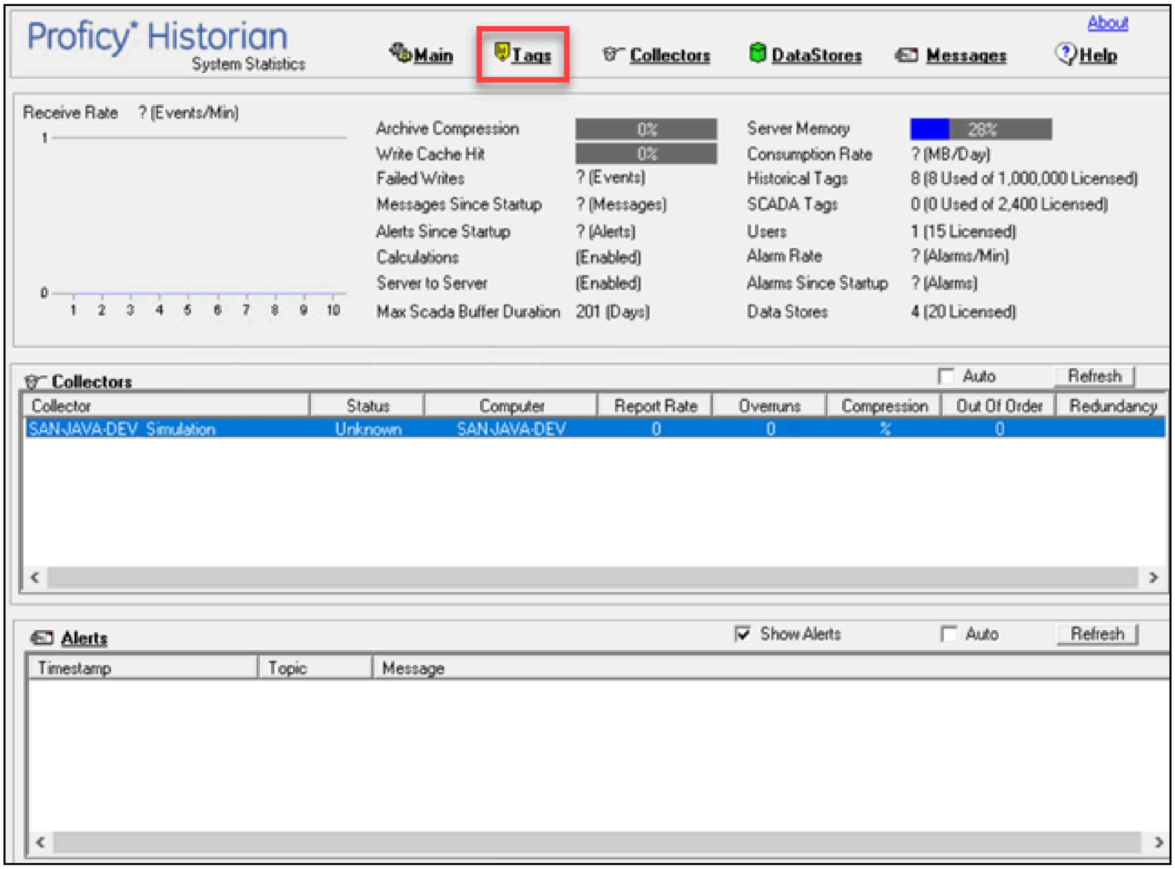
5. In the **New Data Store Name** field, enter the new name. The following special characters cannot be used in data store names: `/\ \ * ? < > |`
6. Select **Rename**.

   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 1194)*.
2. Select **Tags**.



Receive Rate ? (Events/Min)

Archive Compression	0%	Server Memory	28%
Write Cache Hit	0%	Consumption Rate	? (MB/Day)
Failed Writes	? (Events)	Historical Tags	8 (8 Used of 1,000,000 Licensed)
Messages Since Startup	? (Messages)	SCADA Tags	0 (0 Used of 2,400 Licensed)
Alerts Since Startup	? (Alerts)	Users	1 (15 Licensed)
Calculations	(Enabled)	Alarm Rate	? (Alarms/Min)
Server to Server	(Enabled)	Alarms Since Startup	? (Alarms)
Max Scada Buffer Duration	201 (Days)	Data Stores	4 (20 Licensed)

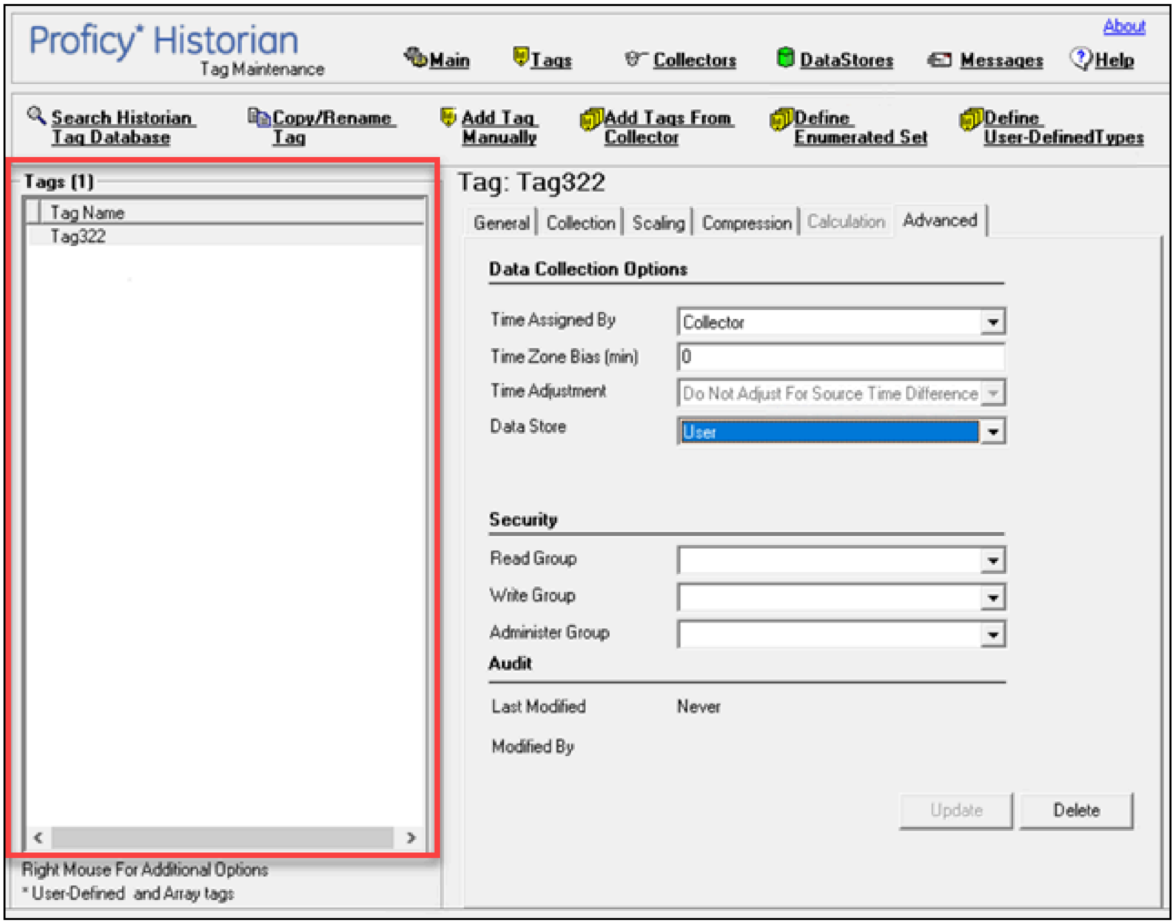
Collectors □ Auto Refresh

Collector	Status	Computer	Report Rate	Overruns	Compression	Out Of Order	Redundancy
SAN-JAVA-DEV_Simulation	Unknown	SAN-JAVA-DEV	0	0	%	0	

Alerts ☑ Show Alerts □ Auto Refresh

Timestamp	Topic	Message

3. Select the tag that you want to move to a different data store.



Tag: Tag322

Data Collection Options

Time Assigned By: Collector

Time Zone Bias (min): 0

Time Adjustment: Do Not Adjust For Source Time Difference

Data Store: User

Security

Read Group:

Write Group:

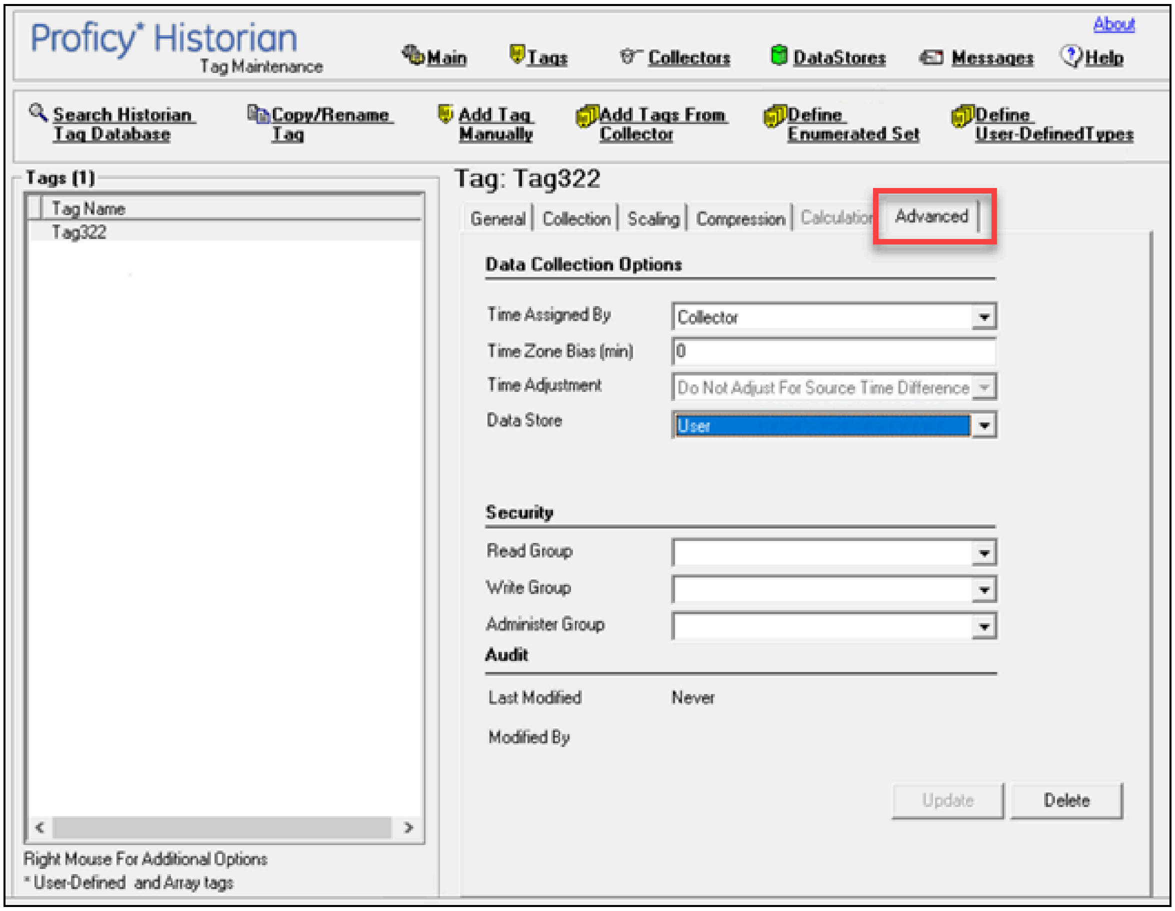
Administer Group:

Audit

Last Modified: Never

Modified By:

4. Select **Advanced**.



Tag: Tag322

Data Collection Options

Time Assigned By: Collector

Time Zone Bias (min): 0

Time Adjustment: Do Not Adjust For Source Time Difference

Data Store: User

Security

Read Group:

Write Group:

Administer Group:

Audit

Last Modified: Never

Modified By:

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

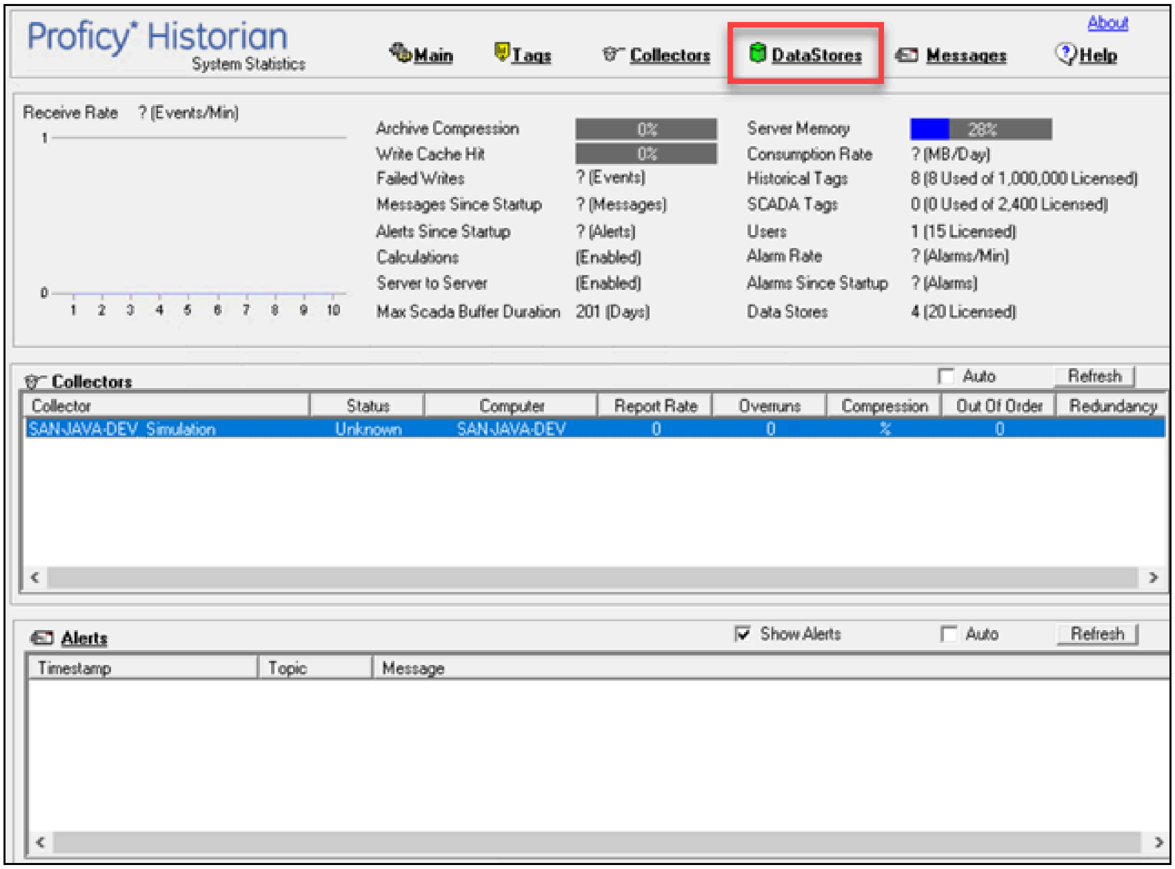
You can delete a data store when it is no longer needed.

Note:

- You can only delete user data stores. You must not 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 1194)*.
2. Select **DataStores**.



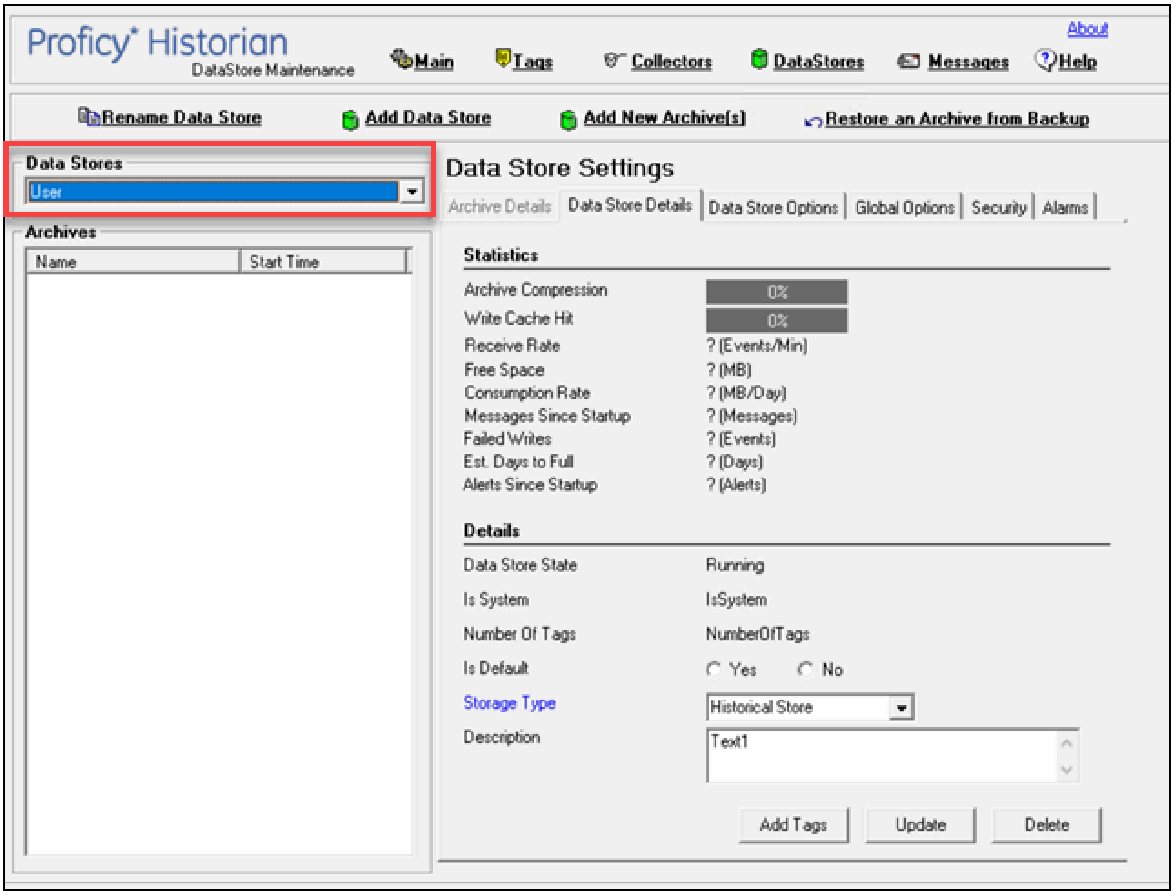
Receive Rate ? (Events/Min)

Archive Compression	0%	Server Memory	28%
Write Cache Hit	0%	Consumption Rate	? (MB/Day)
Failed Writes	? (Events)	Historical Tags	8 (8 Used of 1,000,000 Licensed)
Messages Since Startup	? (Messages)	SCADA Tags	0 (0 Used of 2,400 Licensed)
Alerts Since Startup	? (Alerts)	Users	1 (15 Licensed)
Calculations	(Enabled)	Alarm Rate	? (Alarms/Min)
Server to Server	(Enabled)	Alarms Since Startup	? (Alarms)
Max Scada Buffer Duration	201 (Days)	Data Stores	4 (20 Licensed)

Collector	Status	Computer	Report Rate	Overruns	Compression	Out Of Order	Redundancy
SAN-JAVA-DEV_Simulation	Unknown	SAN-JAVA-DEV	0	0	%	0	

Timestamp	Topic	Message

3. In the **Data Stores** field, select the data store that you want to delete.



Data Stores: User

Data Store Settings

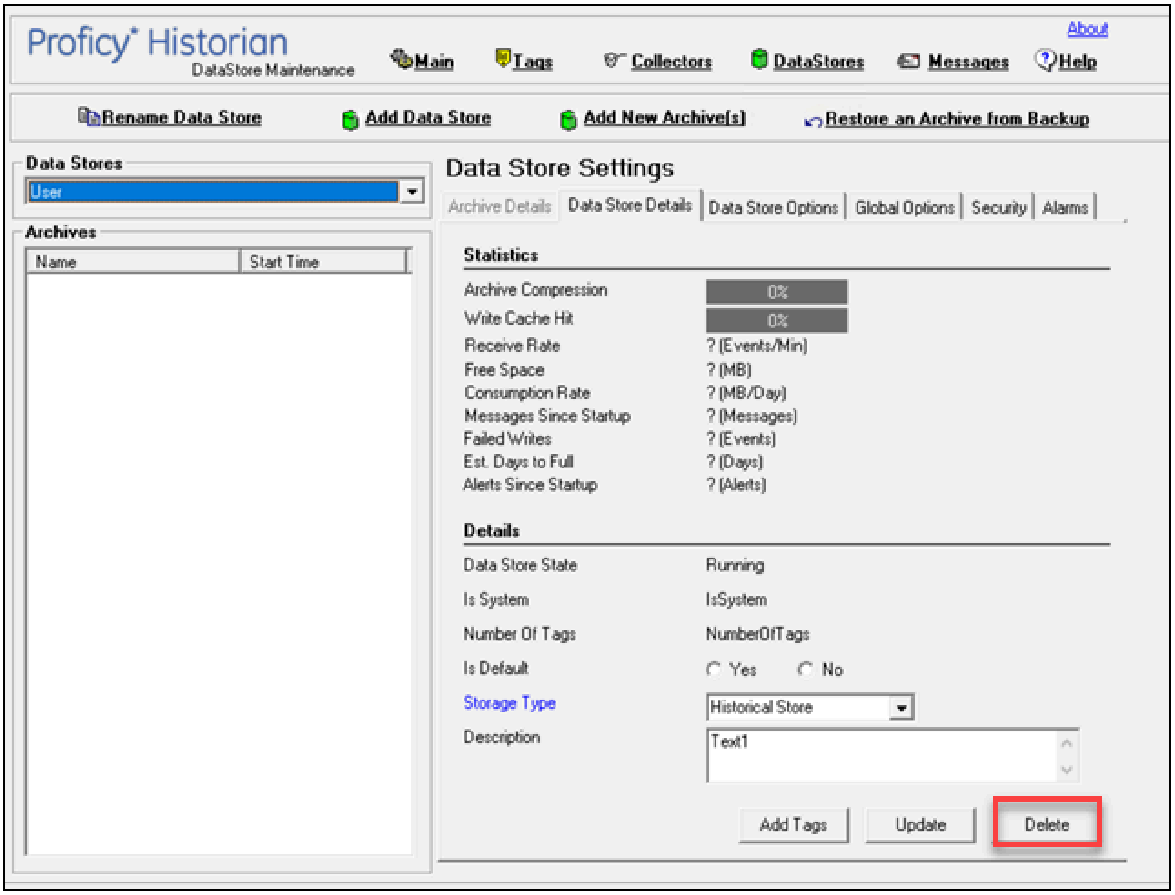
Statistics

Archive Compression	0%
Write Cache Hit	0%
Receive Rate	? (Events/Min)
Free Space	? (MB)
Consumption Rate	? (MB/Day)
Messages Since Startup	? (Messages)
Failed Writes	? (Events)
Est. Days to Full	? (Days)
Alerts Since Startup	? (Alerts)

Details

Data Store State	Running
Is System	IsSystem
Number Of Tags	NumberOfTags
Is Default	○ Yes ○ No
Storage Type	Historical Store
Description	Text1

4. Select **Delete**.



Data Store Settings

Statistics

Archive Compression	0%
Write Cache Hit	0%
Receive Rate	? (Events/Min)
Free Space	? (MB)
Consumption Rate	? (MB/Day)
Messages Since Startup	? (Messages)
Failed Writes	? (Events)
Est. Days to Full	? (Days)
Alerts Since Startup	? (Alerts)

Details

Data Store State	Running
Is System	IsSystem
Number Of Tags	NumberOfTags
Is Default	○ Yes ○ No
Storage Type	Historical Store
Description	Text1

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 1231)*.

Important:

- 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.
- Ensure that the number of archive files does not exceed 1024. Otherwise, the archiver will crash. This is because 1024 is the default number of file descriptors a process can open on Linux. We recommend that you create archive files daily or by size so that you can monitor the number of archive files 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

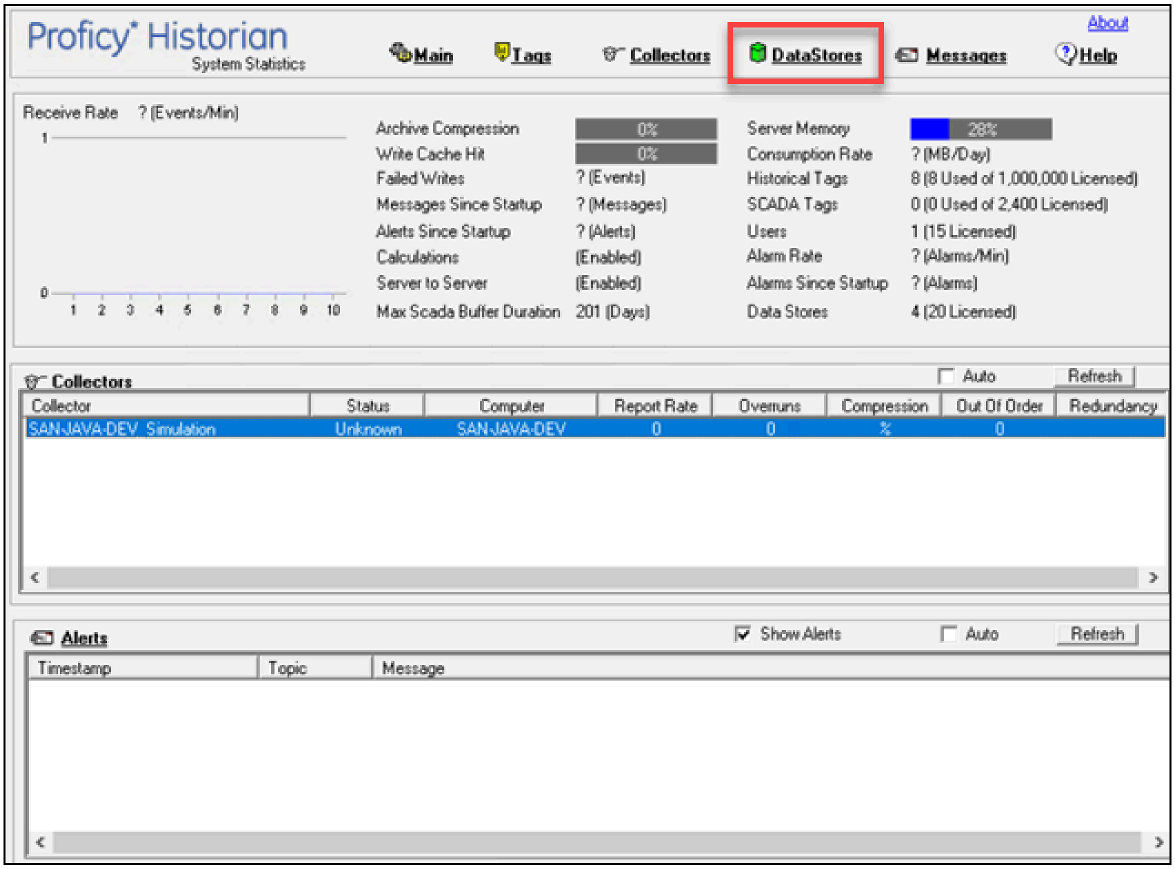
An archive size calculator tool is available to estimate archive size and collector compression based on a tag that has already been configured or based on your inputs. Log on to http://digitalsupport.ge.com to download this tool and other GE Intelligent Platforms freeware product solutions.

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 1235)*. 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 1194)*.
2. Select **DataStores**.



Receive Rate ? (Events/Min)

Archive Compression	0%	Server Memory	28%
Write Cache Hit	0%	Consumption Rate	? (MB/Day)
Failed Writes	? (Events)	Historical Tags	8 (8 Used of 1,000,000 Licensed)
Messages Since Startup	? (Messages)	SCADA Tags	0 (0 Used of 2,400 Licensed)
Alerts Since Startup	? (Alerts)	Users	1 (15 Licensed)
Calculations	(Enabled)	Alarm Rate	? (Alarms/Min)
Server to Server	(Enabled)	Alarms Since Startup	? (Alarms)
Max Scada Buffer Duration	201 (Days)	Data Stores	4 (20 Licensed)

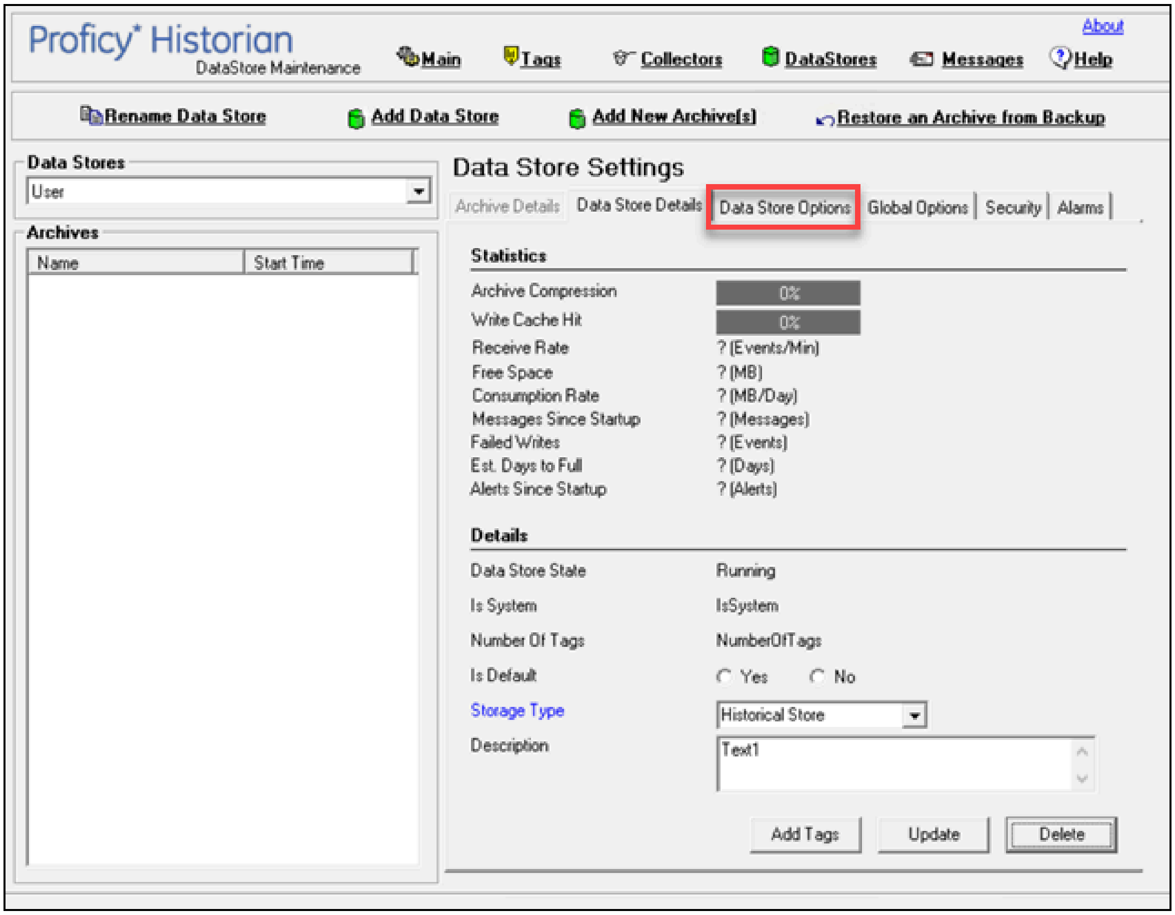
Collectors □ Auto Refresh

Collector	Status	Computer	Report Rate	Overruns	Compression	Out Of Order	Redundancy
SAN-JAVA-DEV_Simulation	Unknown	SAN-JAVA-DEV	0	0	%	0	

Alerts ☑ Show Alerts □ Auto Refresh

Timestamp	Topic	Message

3. Select **Data Store Options**.



Data Stores: User

Statistics

Archive Compression	0%
Write Cache Hit	0%
Receive Rate	? (Events/Min)
Free Space	? (MB)
Consumption Rate	? (MB/Day)
Messages Since Startup	? (Messages)
Failed Writes	? (Events)
Est. Days to Full	? (Days)
Alerts Since Startup	? (Alerts)

Details

Data Store State	Running
Is System	IsSystem
Number Of Tags	NumberOfTags
Is Default	○ Yes ○ No
Storage Type	Historical Store
Description	Text1

4. Enter values as described in the following table.

Field	Description
Automatically Create Archives	Select **Enabled**. **Important:** ◦ 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. ◦ Ensure that the number of archive files does not exceed 1024. Otherwise, the archiver will crash. This is because 1024 is the default number of file descriptors a process can open on Linux. We

Field	Description
	! recommend that you create archive files daily or by size so that you can monitor the number of archive files created.
Overwrite Old Archives	Specify whether you want to overwrite old archives with new ones. Exercise caution in enabling this option. We recommend that you back up 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. If, however, you want to create archives after a duration, select **Days** or **Hours**, and then enter the value in the **Archive Duration** field.
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. 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	Not applicable. Use EFS to back up and restore archives.
Base Archive Name	Not applicable. Use EFS to back up and restore archives.
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

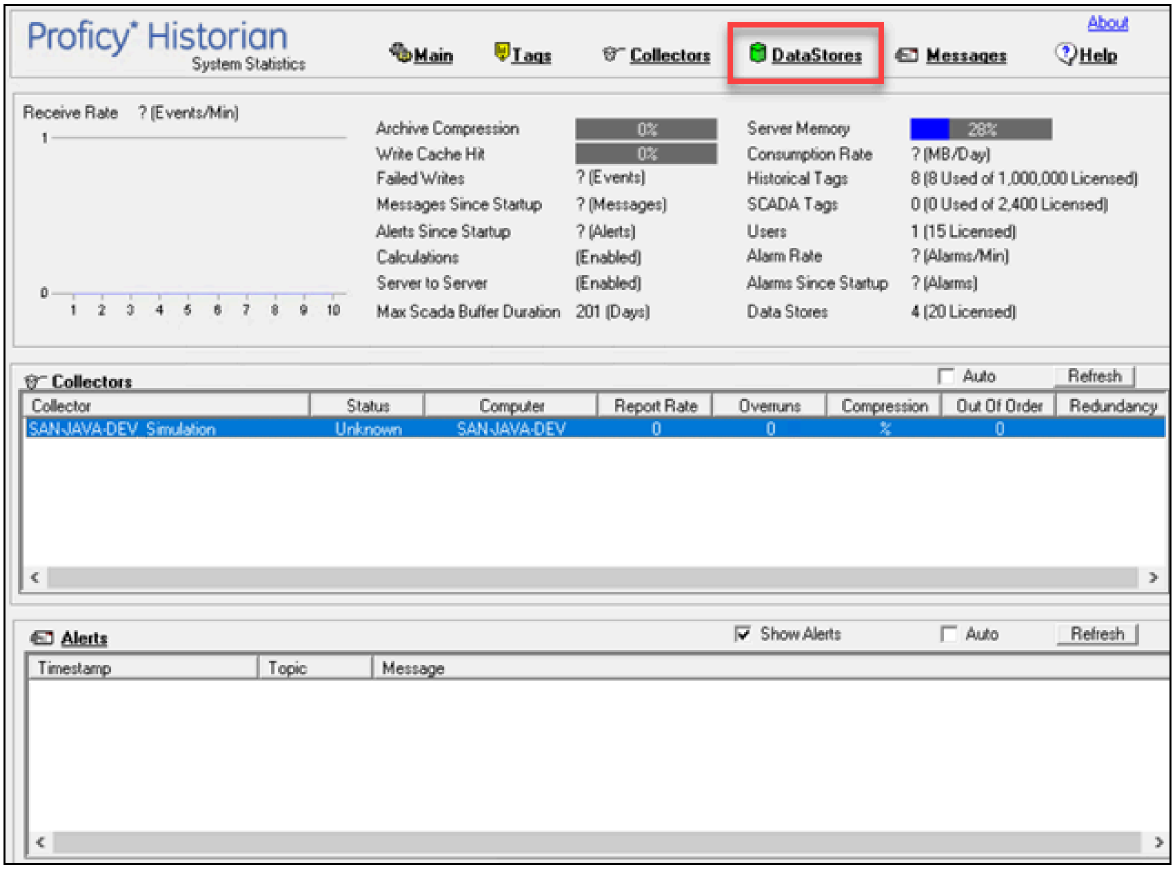
If you want to create multiple archives at the same time, access Historian Administrator, and set values for the following fields:

Field	Value
The **Details** Section	
File Attribute	**Read/Write**
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 1231)*. When the current archive is full, a new archive is used (in a sequential order).

You can create multiple archives at the same time.

1. Access Historian Administrator *(on page 1194)*.
2. Select **DataStores**.

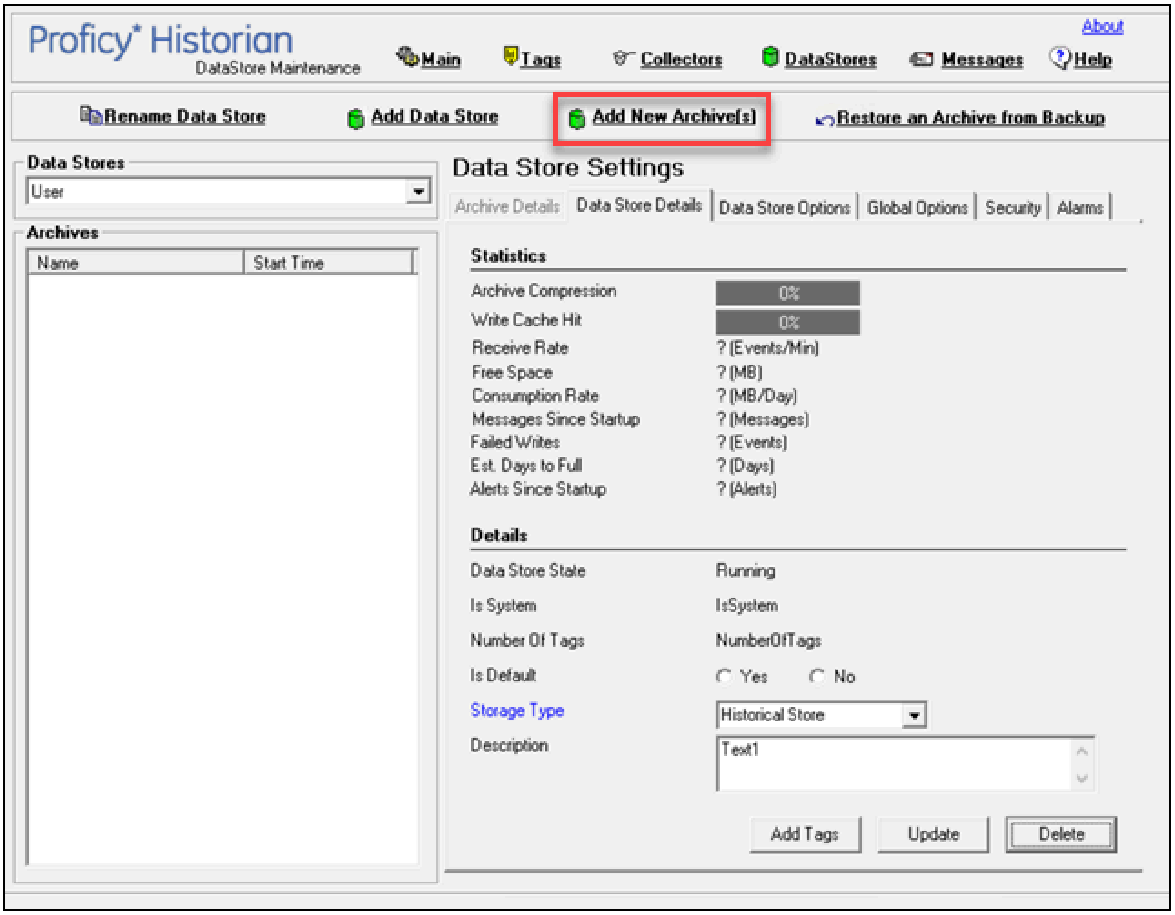


Archive Compression	0%	Server Memory	28%
Write Cache Hit	0%	Consumption Rate	? (MB/Day)
Failed Writes	? (Events)	Historical Tags	8 (8 Used of 1,000,000 Licensed)
Messages Since Startup	? (Messages)	SCADA Tags	0 (0 Used of 2,400 Licensed)
Alerts Since Startup	? (Alerts)	Users	1 (15 Licensed)
Calculations	(Enabled)	Alarm Rate	? (Alarms/Min)
Server to Server	(Enabled)	Alarms Since Startup	? (Alarms)
Max Scada Buffer Duration	201 (Days)	Data Stores	4 (20 Licensed)

Collector	Status	Computer	Report Rate	Overruns	Compression	Out Of Order	Redundancy
SAN-JAVA-DEV_Simulation	Unknown	SAN-JAVA-DEV	0	0	%	0	

Timestamp	Topic	Message

3. Select **Add New Archive(s)**.



Data Stores: User

Archives

Name	Start Time

Data Store Settings

Statistics

Archive Compression	0%
Write Cache Hit	0%
Receive Rate	? (Events/Min)
Free Space	? (MB)
Consumption Rate	? (MB/Day)
Messages Since Startup	? (Messages)
Failed Writes	? (Events)
Est. Days to Full	? (Days)
Alerts Since Startup	? (Alerts)

Details

Data Store State	Running
Is System	IsSystem
Number Of Tags	NumberOfTags
Is Default	○ Yes ○ No
Storage Type	Historical Store
Description	Text1

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
	Important: Ensure that the number of archive files does not exceed 1024. Otherwise, the archiver will crash. This is because 1024 is the default number of file descriptors a process can open on Linux. We recommend that you create archive files daily or by size so that you can monitor the number of archive files created.
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. **Note:** 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.

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.

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 1246)*.

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.

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

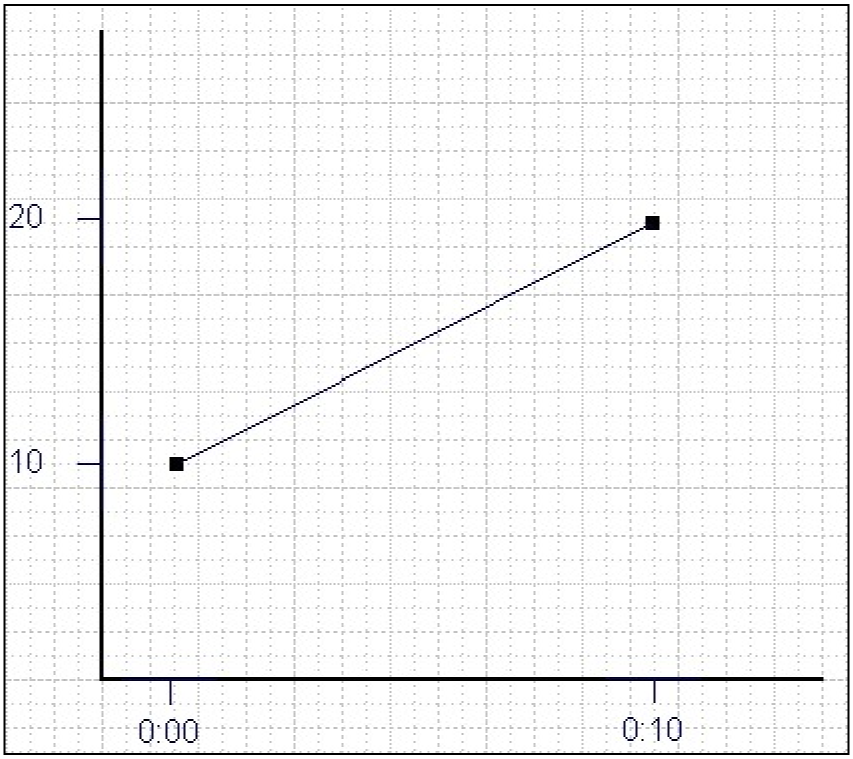
For example, a collector reads a value X once per second, with a compression deadband of 1.0. If the value of X is 10.0 for a number of seconds starting at 0:00:00 and jumps to 20.0 at 0:00:10, the data samples read would be:

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
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)

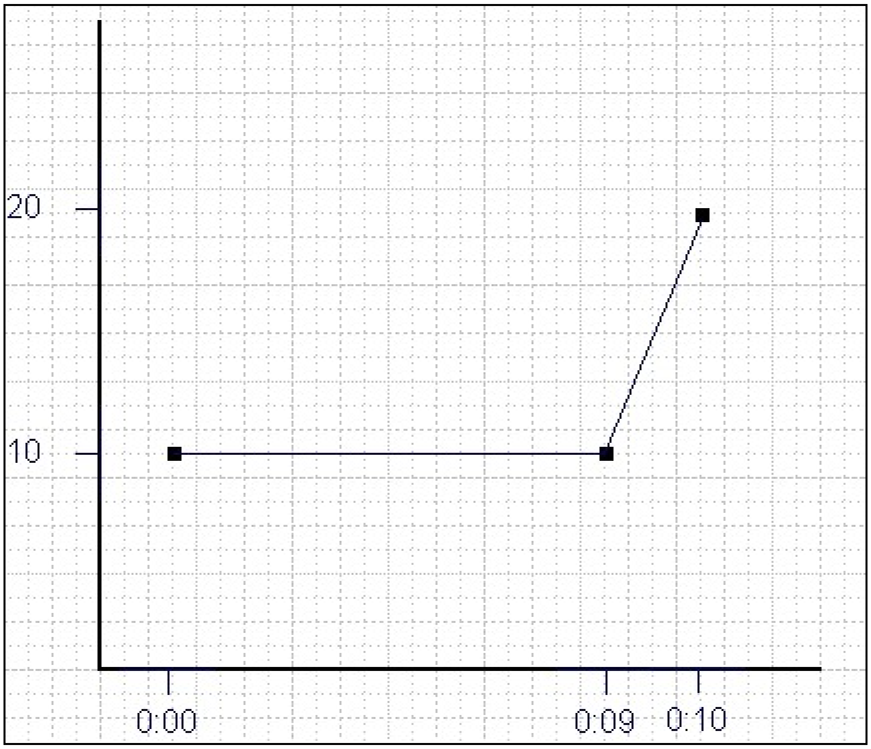
However, without the marker value, if this data were to be put into a chart, it would look like the data value **ramped** over 10 seconds from a value of 10.0 to 20.0, as shown in the following chart.



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. 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 1246)*.

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.

For instructions on setting the scaling parameters, refer to Access/Modify a Tag *(on page 1246)*.

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 1246)*.

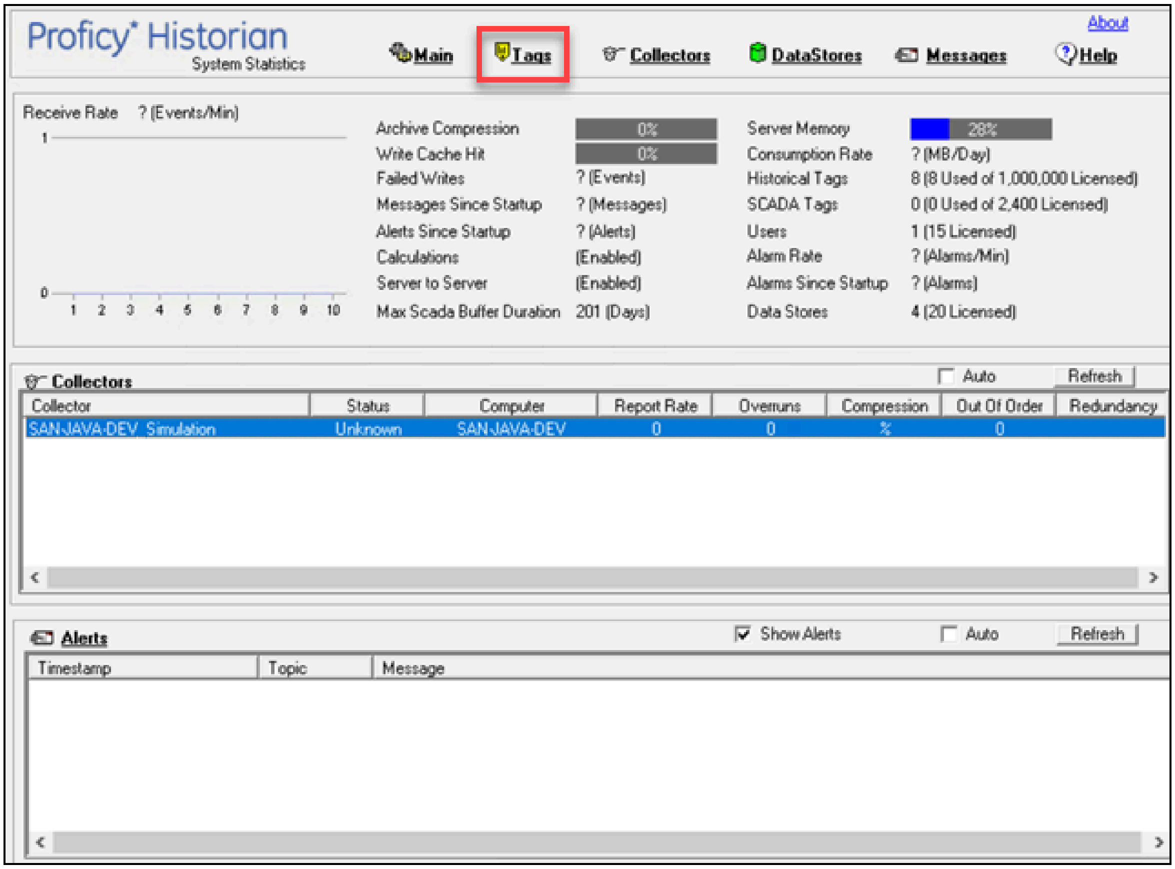
Access/Modify a Tag

T modify a tag, you must be a member of the administrator's group with tag-level security (that is, the iH Security Admins or the iH Tag Admins group).

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 1194)*.
2. Select **Tags**.



Archive Compression	0%	Server Memory	28%
Write Cache Hit	0%	Consumption Rate	? (MB/Day)
Failed Writes	? (Events)	Historical Tags	8 (8 Used of 1,000,000 Licensed)
Messages Since Startup	? (Messages)	SCADA Tags	0 (0 Used of 2,400 Licensed)
Alerts Since Startup	? (Alerts)	Users	1 (15 Licensed)
Calculations	(Enabled)	Alarm Rate	? (Alarms/Min)
Server to Server	(Enabled)	Alarms Since Startup	? (Alarms)
Max Scada Buffer Duration	201 (Days)	Data Stores	4 (20 Licensed)

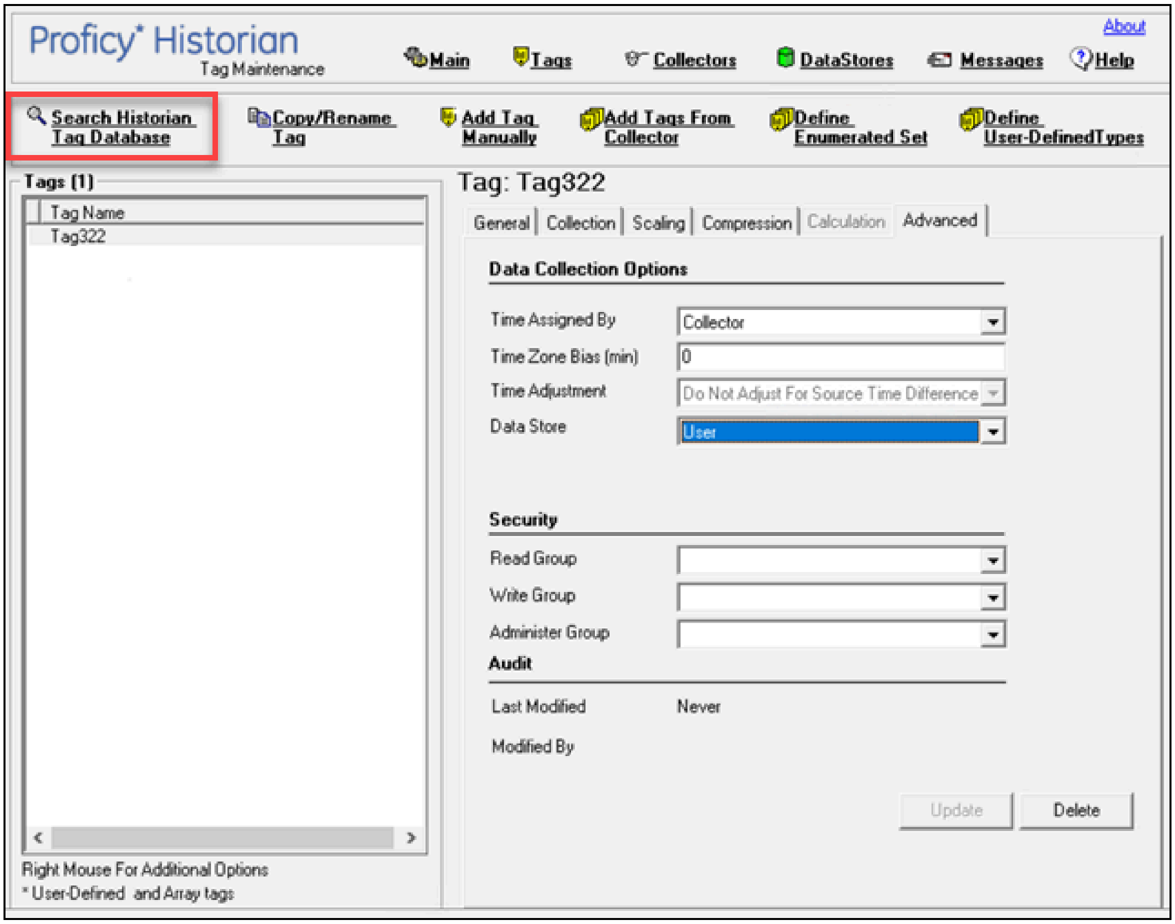
Collectors

Collector	Status	Computer	Report Rate	Overruns	Compression	Out Of Order	Redundancy
SAN-JAVA-DEV_Simulation	Unknown	SAN-JAVA-DEV	0	0	%	0	

Alerts

Timestamp	Topic	Message

3. Select **Search Historian Tag Database**.



Tag: Tag322

Data Collection Options

Time Assigned By: Collector

Time Zone Bias (min): 0

Time Adjustment: Do Not Adjust For Source Time Difference

Data Store: User

Security

Read Group:

Write Group:

Administer Group:

Audit

Last Modified: Never

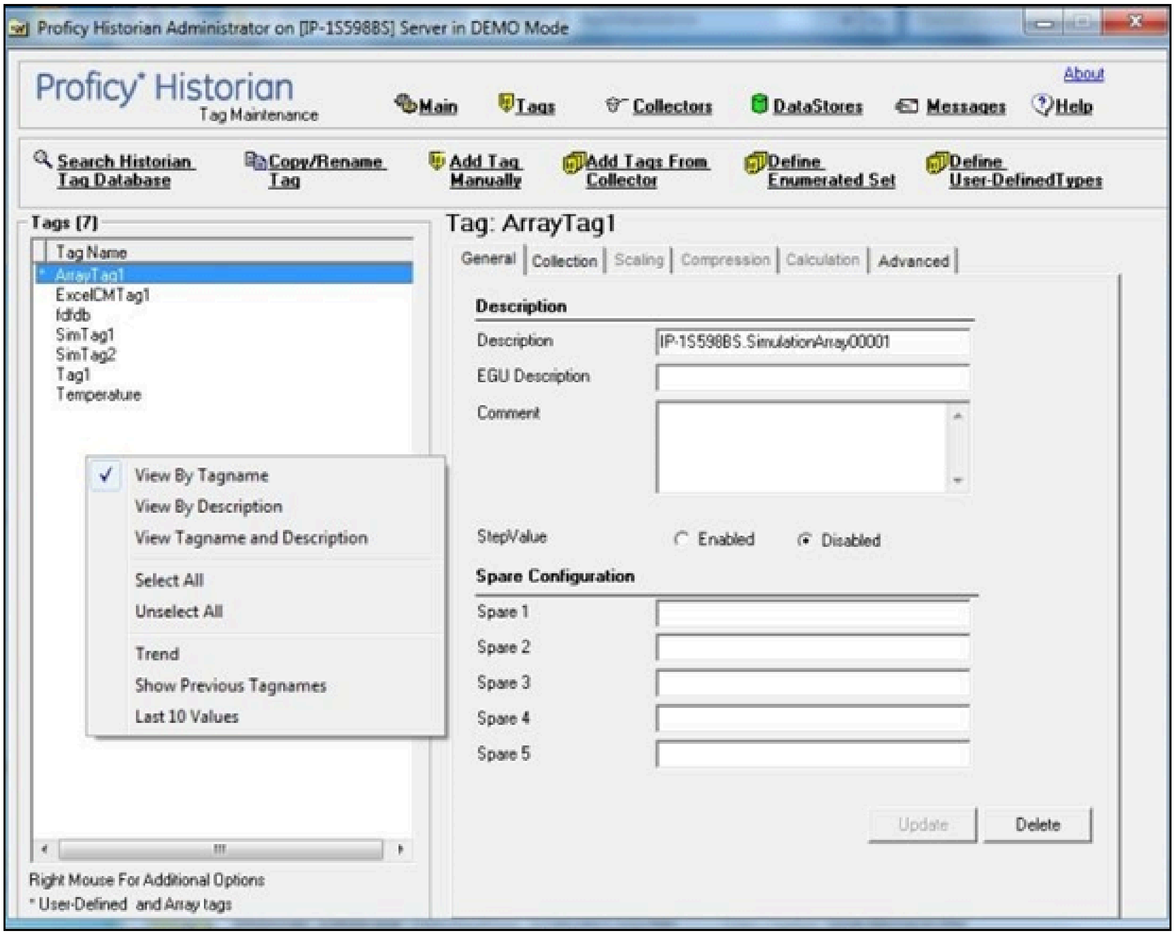
Modified By:

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.



Tag: ArrayTag1

Description

Description: IP-1S598BS.SimulationArray00001

EGU Description:

Comment:

StepValue: ○ Enabled ● Disabled

Spare Configuration

Spare 1:

Spare 2:

Spare 3:

Spare 4:

Spare 5:

6. As needed, modify values as described in the following tables, and then select **Update**.

Table 161. 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: Do not add or update the spare configurations as the data may get corrupted or overwritten. For example, the **Spare 5** field is used by the Server-to-Server collector for internal purposes.

Table 162. 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. For Python Expression tags, this field contains the full applicable JSON configuration, which includes an indication of the source address. **Note:** 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. 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. **Note:** 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.

Field	Description
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	Not applicable
Collection	Indicates whether data collection is enable or disable 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. **Note:** 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 1245)* is enabled.
Trigger Tag	The name of the trigger tag used in the condition.

Field	Description
Comparison	The comparison operator that you want to use in the condition. Select one of the following options: ◦ **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. ◦ **< =**: Setting condition as trigger tag value less than or equal to the compare value. ◦ **> =** Setting condition as trigger tag value greater than or equal to the compare value. ◦ **<**: Setting condition as trigger tag value less than the compare value. ◦ **>**: Setting condition as trigger tag value greater than the compare value. ◦ **=**: Setting condition as trigger tag value equals compare value. ◦ **!=**: 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 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 163. The Scaling Section

Field	Description
Hi Engineering Units	The current value of the upper range limit of the span for this tag. 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.
	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, 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.
	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`

Field	Description

	`Lo Scale Value = - 1`
	`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 164. The Compression Section

Field	Description
Collector Compression	Indicates whether collector compression *(on page 1239)* 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. **Note:** 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. **Example:** 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.) 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

Field	Description
	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 1302)*.
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. 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. 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. 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.

Field	Description
Archive Compression	Indicates whether archive compression *(on page 1239)* 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. The data archiver treats the incoming sample after the timeout occurs as if it exceeded compression. It then stores the pending sample.

Table 165. 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. 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. 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. **Note:** This field is disabled for Calculation and Server-to-Server tags.

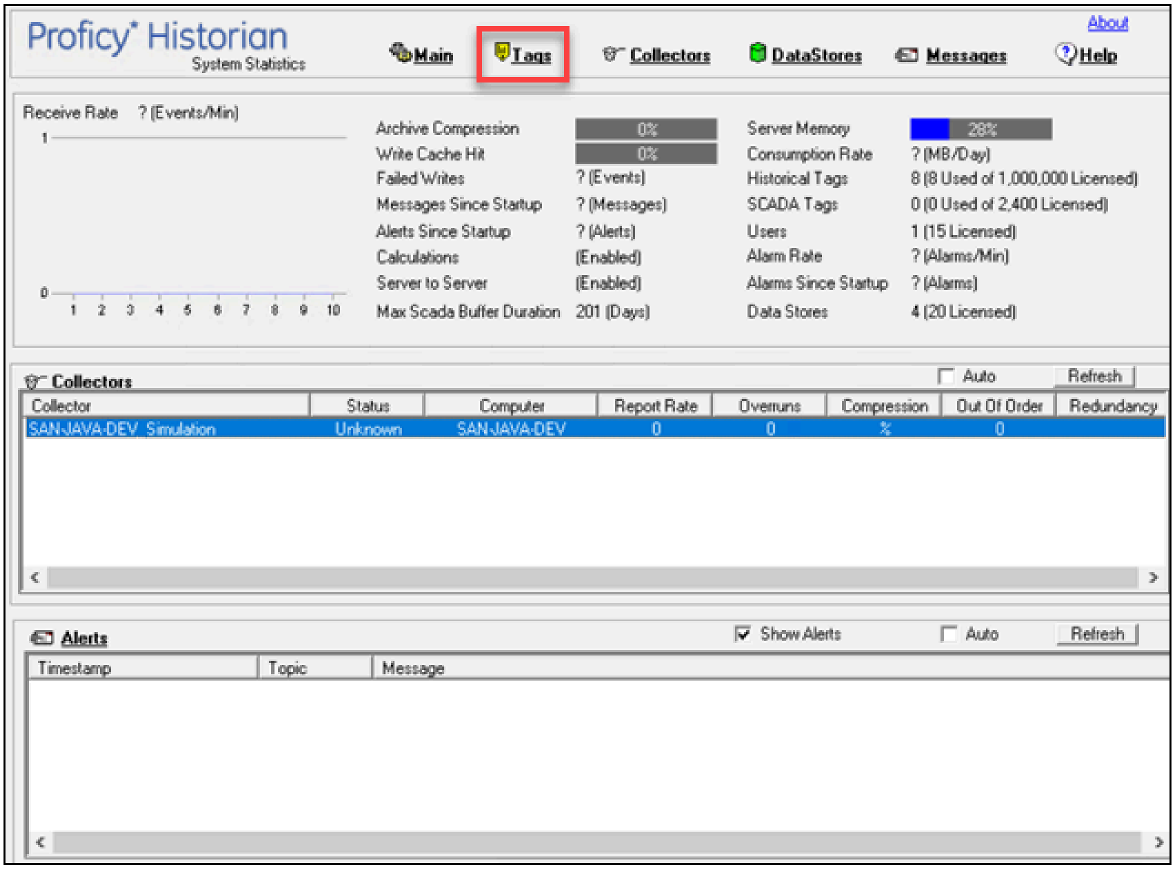
Field	Description
Time Zone Bias	The number of minutes from GMT that should be used to translate timestamps when retrieving data from this tag. For example, the time zone bias for Eastern Standard time is -300 minutes (GMT-5). This field is not used during collection. Use this option if a particular 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. **Note:** 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.
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

- Ensure that you are a member of the administrator's group with tag-level security (that is, the iH Security Admins or the iH Tag Admins group).
- Create a collector instance 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 1261)*.

1. Access Historian Administrator *(on page 1194)*.
2. Select **Tags**.



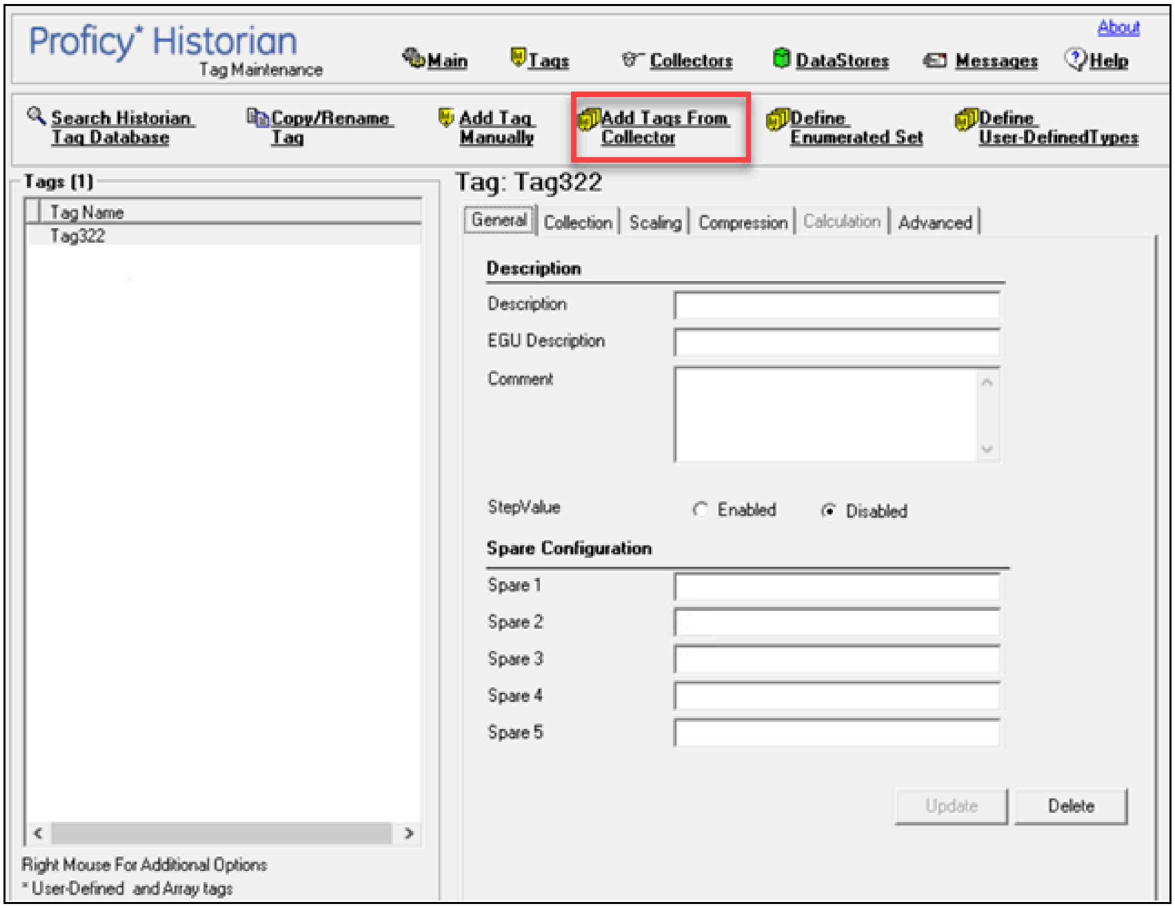
Receive Rate ? (Events/Min)

Archive Compression	0%	Server Memory	28%
Write Cache Hit	0%	Consumption Rate	? (MB/Day)
Failed Writes	? (Events)	Historical Tags	8 (8 Used of 1,000,000 Licensed)
Messages Since Startup	? (Messages)	SCADA Tags	0 (0 Used of 2,400 Licensed)
Alerts Since Startup	? (Alerts)	Users	1 (15 Licensed)
Calculations	(Enabled)	Alarm Rate	? (Alarms/Min)
Server to Server	(Enabled)	Alarms Since Startup	? (Alarms)
Max Scada Buffer Duration	201 (Days)	Data Stores	4 (20 Licensed)

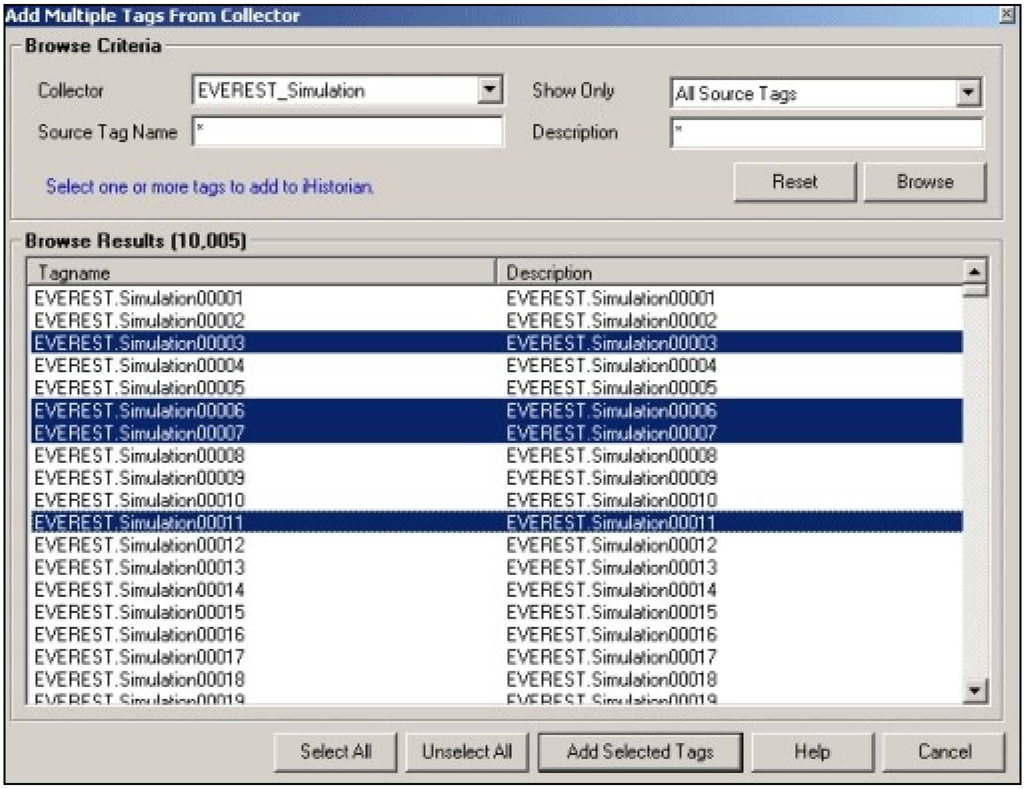
Collector	Status	Computer	Report Rate	Overruns	Compression	Out Of Order	Redundancy
SAN-JAVA-DEV_Simulation	Unknown	SAN-JAVA-DEV	0	0	%	0	

Timestamp	Topic	Message

3. Select **Add Tags from Collector**.



The **Add Multiple Tags from Collector** window appears.



Collector: EVEREST_Simulation

Show Only: All Source Tags

Source Tag Name: *

Description: *

Select one or more tags to add to iHistorian.

Tagname	Description
EVEREST.Simulation00001	EVEREST.Simulation00001
EVEREST.Simulation00002	EVEREST.Simulation00002
EVEREST.Simulation00003	EVEREST.Simulation00003
EVEREST.Simulation00004	EVEREST.Simulation00004
EVEREST.Simulation00005	EVEREST.Simulation00005
EVEREST.Simulation00006	EVEREST.Simulation00006
EVEREST.Simulation00007	EVEREST.Simulation00007
EVEREST.Simulation00008	EVEREST.Simulation00008
EVEREST.Simulation00009	EVEREST.Simulation00009
EVEREST.Simulation00010	EVEREST.Simulation00010
EVEREST.Simulation00011	EVEREST.Simulation00011
EVEREST.Simulation00012	EVEREST.Simulation00012
EVEREST.Simulation00013	EVEREST.Simulation00013
EVEREST.Simulation00014	EVEREST.Simulation00014
EVEREST.Simulation00015	EVEREST.Simulation00015
EVEREST.Simulation00016	EVEREST.Simulation00016
EVEREST.Simulation00017	EVEREST.Simulation00017
EVEREST.Simulation00018	EVEREST.Simulation00018
EVEREST.Simulation00019	EVEREST.Simulation00019

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

You must be a member of the administrator's group with tag-level security (that is, the iH Security Admins or the iH Tag Admins group).

This topic describes how to create a tag manually. You can also add tags from source *(on page 1257)*; 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
- Simulation Collector
- Server-to-Server Collector
- OSI PI Collector
- OSI PI Distributor

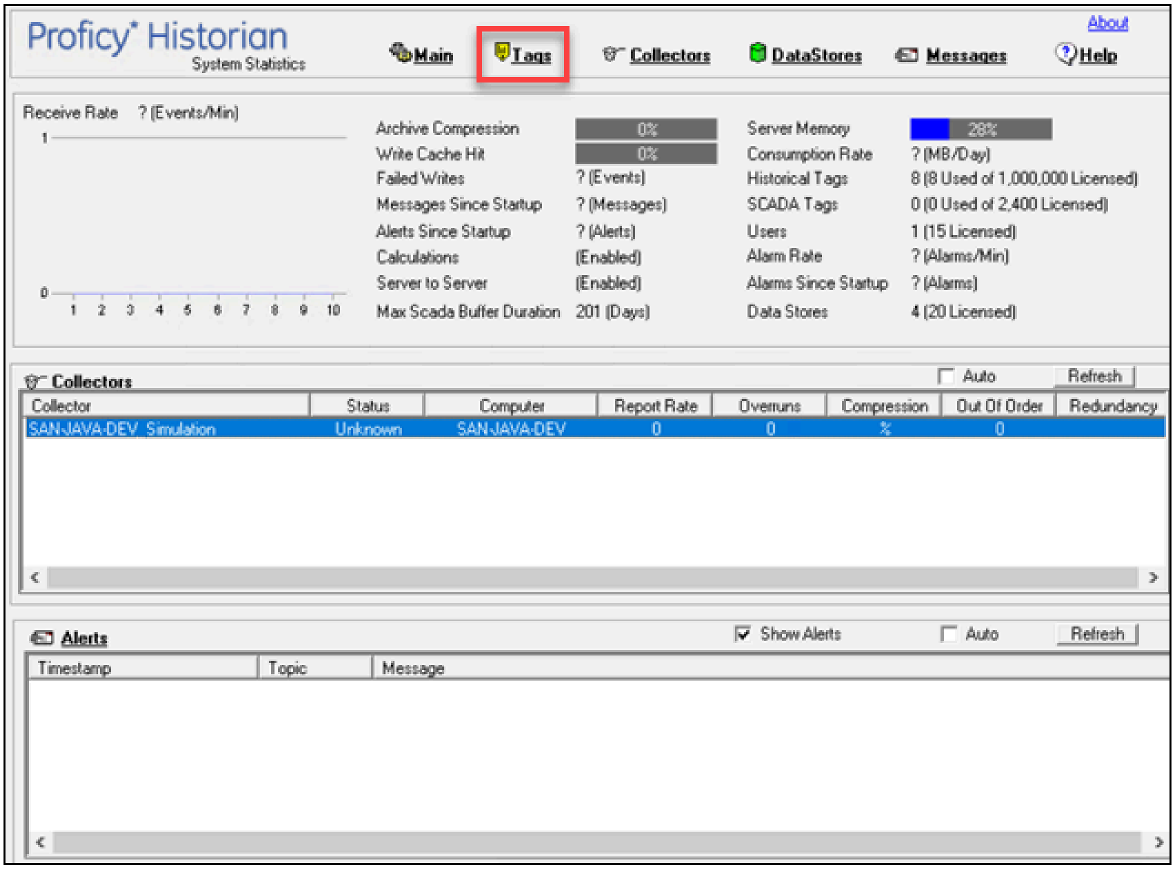
The dynamic collector update feature ensures that any modifications to the tag configuration do not affect all the tags in a collector. Tags that stop data collection may record zero data and bad quality without restarting the collector. Tags that do not stop data collection do not record bad data samples 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 1194)*.
2. Select **Tags**.



Archive Compression	0%	Server Memory	28%
Write Cache Hit	0%	Consumption Rate	? (MB/Day)
Failed Writes	? (Events)	Historical Tags	8 (8 Used of 1,000,000 Licensed)
Messages Since Startup	? (Messages)	SCADA Tags	0 (0 Used of 2,400 Licensed)
Alerts Since Startup	? (Alerts)	Users	1 (15 Licensed)
Calculations	(Enabled)	Alarm Rate	? (Alarms/Min)
Server to Server	(Enabled)	Alarms Since Startup	? (Alarms)
Max Scada Buffer Duration	201 (Days)	Data Stores	4 (20 Licensed)

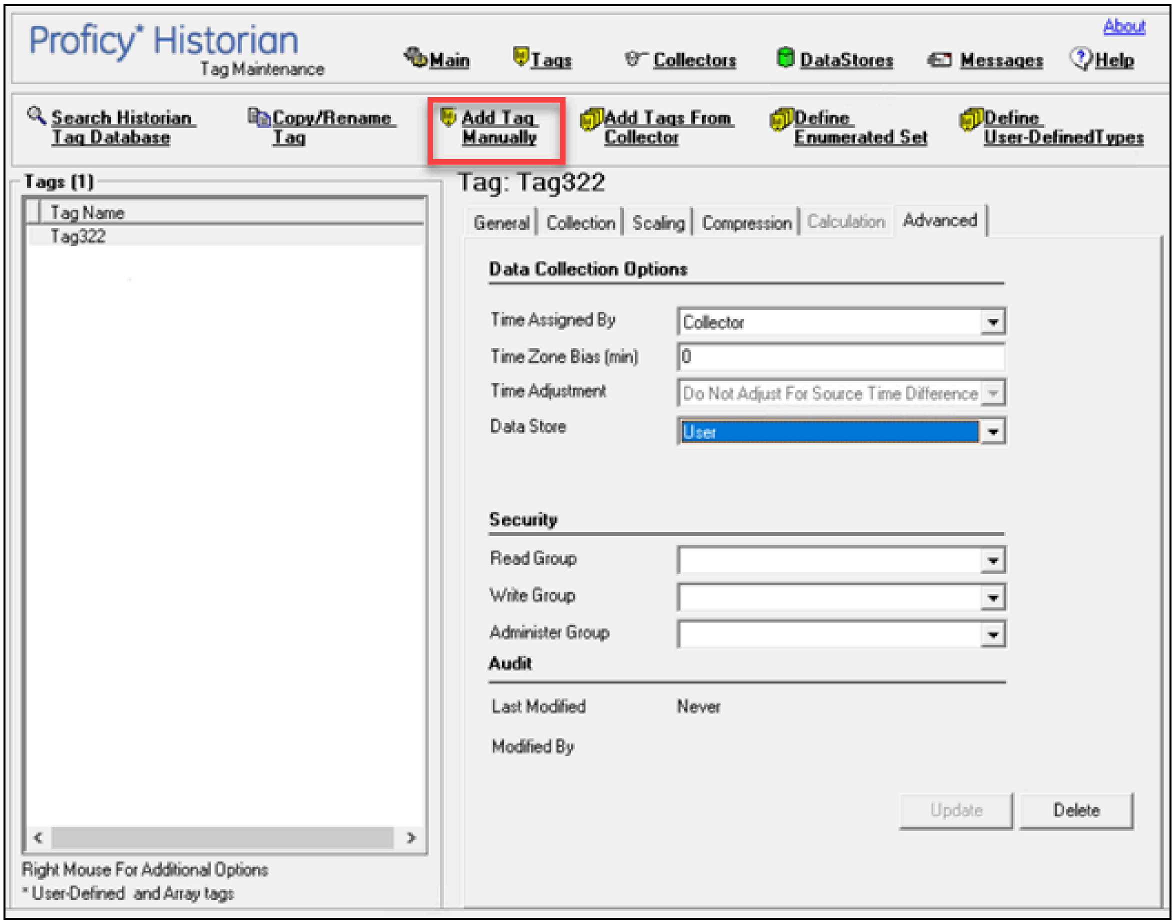
Collectors ☐ Auto Refresh

Collector	Status	Computer	Report Rate	Overruns	Compression	Out Of Order	Redundancy
SANJAVA-DEV_Simulation	Unknown	SANJAVA-DEV	0	0	%	0	

Alerts ☑ Show Alerts ☐ Auto Refresh

Timestamp	Topic	Message

3. Select **Add Tag Manually**.



Tag: Tag322

Data Collection Options

Time Assigned By: Collector
Time Zone Bias (min): 0
Time Adjustment: Do Not Adjust For Source Time Difference
Data Store: User

Security

Read Group:
Write Group:
Administer Group:

Audit

Last Modified: Never
Modified By:

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	Not applicable

Field	Description
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.

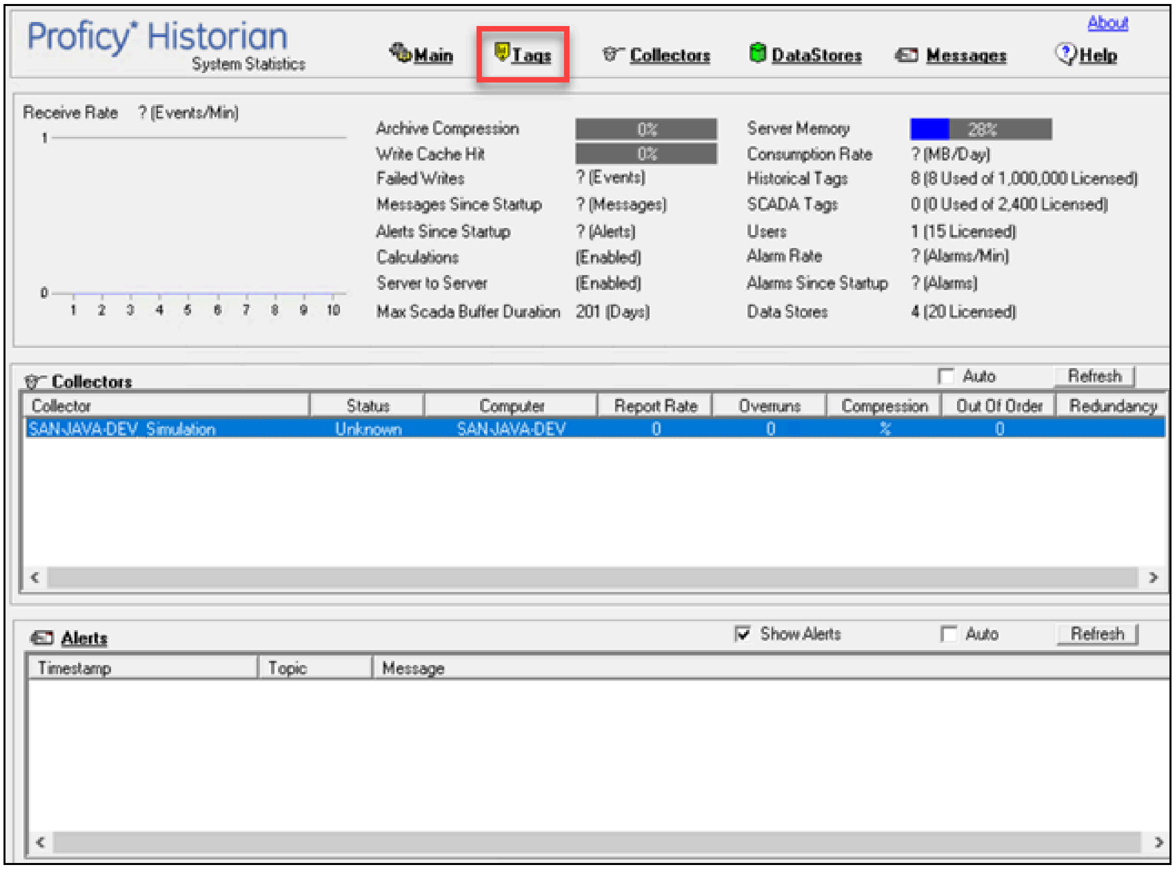
Note:
If you add a tag for a Server-to-Server collector, set the **Time Adjustment** field for the tag to **Adjust for Source Time Difference** after you add the tag. This field is available under **Tags > Advanced**. This is applicable only for polled data collection.

5. Select **OK**.
   The tag is created.

Copy a Tag

You must be a member of the administrator's group with tag-level security (that is, the iH Security Admins or the iH Tag Admins group).

1. Access Historian Administrator *(on page 1194)*.
2. Select **Tags**.



Receive Rate ? (Events/Min)

Archive Compression	0%	Server Memory	28%
Write Cache Hit	0%	Consumption Rate	? (MB/Day)
Failed Writes	? (Events)	Historical Tags	8 (8 Used of 1,000,000 Licensed)
Messages Since Startup	? (Messages)	SCADA Tags	0 (0 Used of 2,400 Licensed)
Alerts Since Startup	? (Alerts)	Users	1 (15 Licensed)
Calculations	(Enabled)	Alarm Rate	? (Alarms/Min)
Server to Server	(Enabled)	Alarms Since Startup	? (Alarms)
Max Scada Buffer Duration	201 (Days)	Data Stores	4 (20 Licensed)

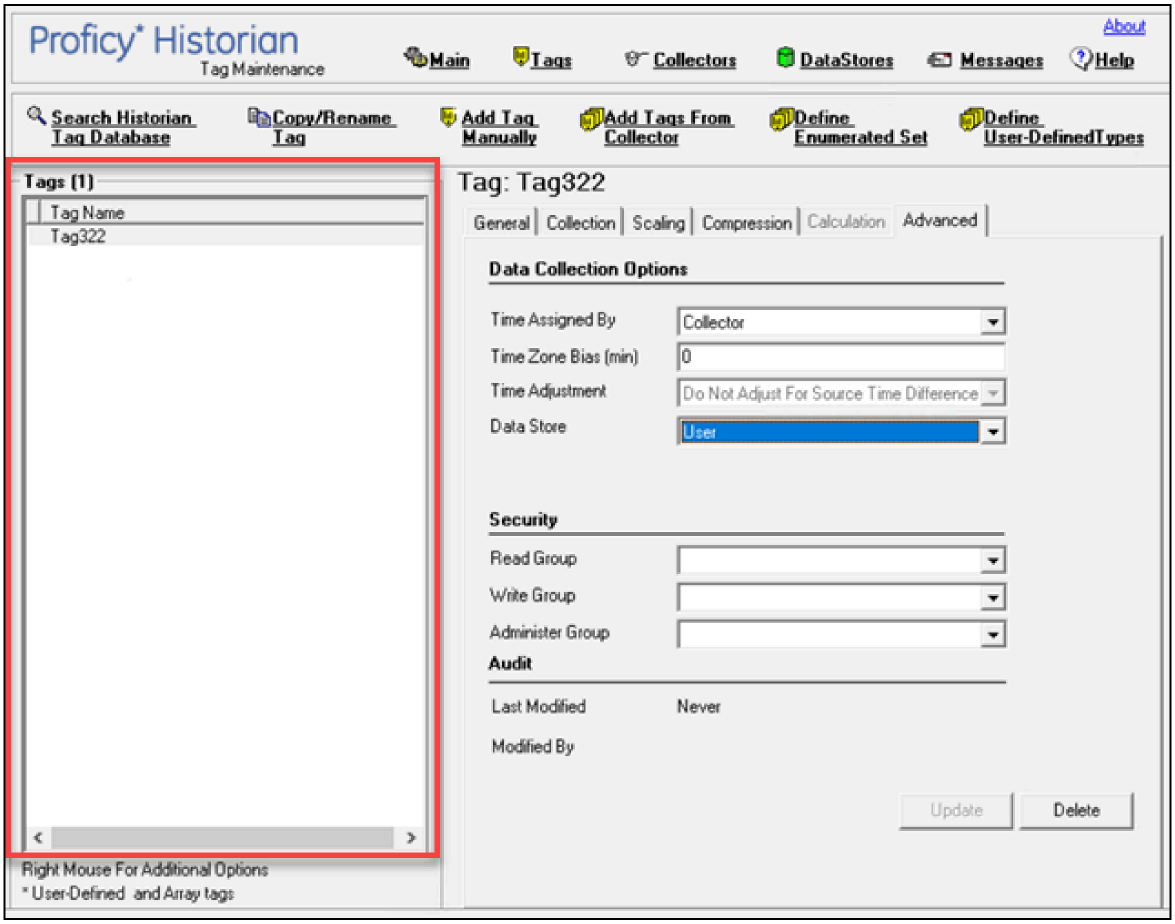
Collectors

Collector	Status	Computer	Report Rate	Overruns	Compression	Out Of Order	Redundancy
SAN-JAVA-DEV_Simulation	Unknown	SAN-JAVA-DEV	0	0	%	0	

Alerts

Timestamp	Topic	Message

3. Select the tag that you want to copy.



Tags (1)

Tag Name
Tag322

Right Mouse For Additional Options
* User-Defined and Array tags

Tag: Tag322

Data Collection Options

Time Assigned By	Collector
Time Zone Bias (min)	0
Time Adjustment	Do Not Adjust For Source Time Difference
Data Store	User

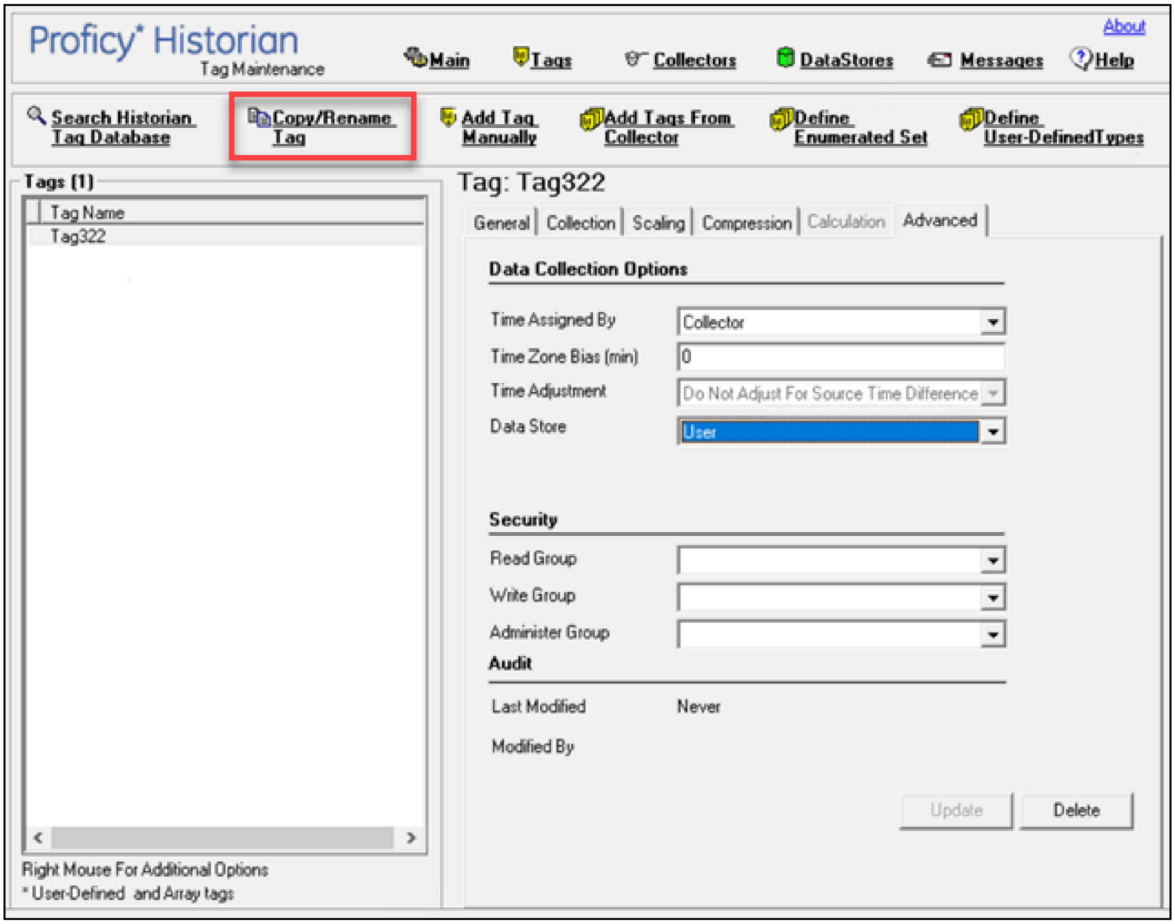
Security

Read Group	
Write Group	
Administer Group	

Audit

Last Modified	Never
Modified By	

4. Select **Copy/Rename Tag**.



Tag: Tag322

Data Collection Options

Time Assigned By: Collector

Time Zone Bias (min): 0

Time Adjustment: Do Not Adjust For Source Time Difference

Data Store: User

Security

Read Group:

Write Group:

Administer Group:

Audit

Last Modified: Never

Modified By:

The **Copy/RenameTag** window appears.

5. Select **Copy**.
6. Enter a new tag name.
7. Select **OK**.

   The tag is copied.

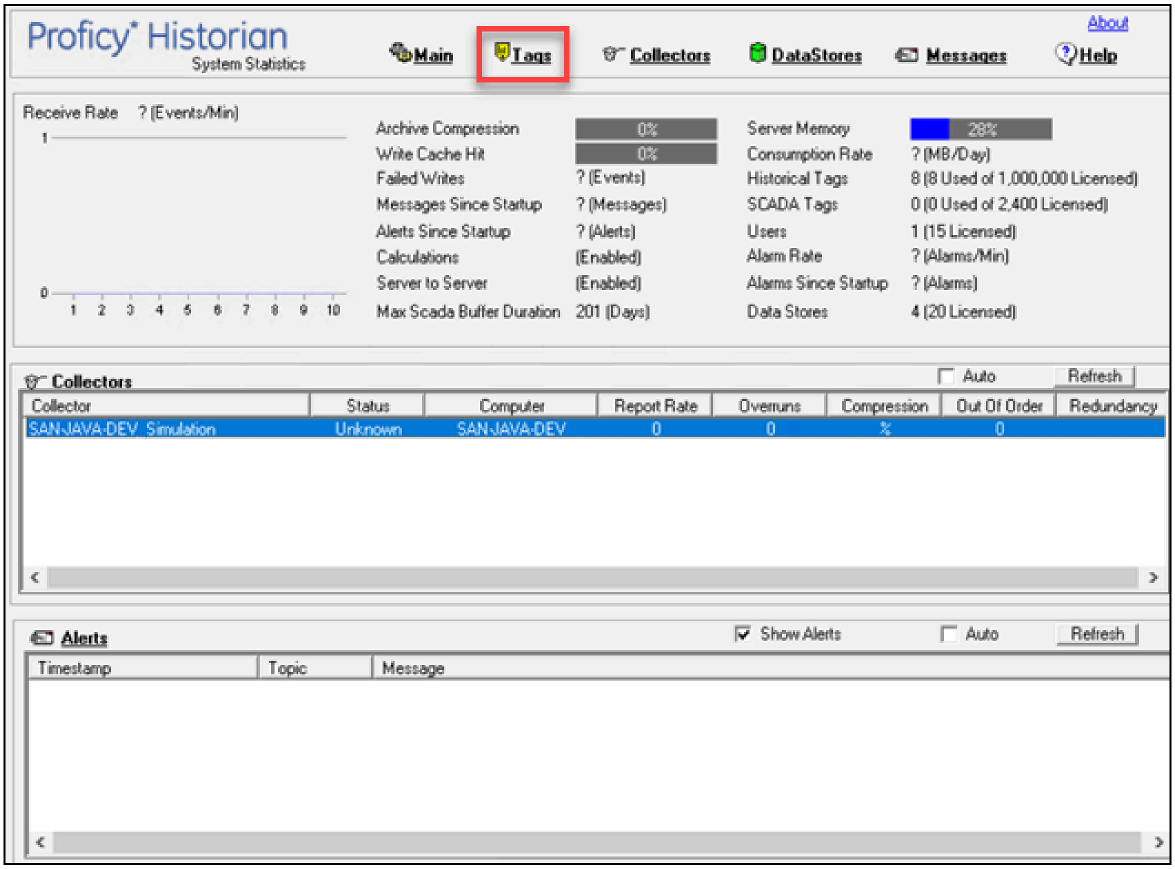
Rename a Tag

- You must be a member of the administrator's group with tag-level security (that is, the iH Security Admins or the iH Tag Admins group).
- 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 1194)*.
2. Select **Tags**.



Receive Rate ? (Events/Min)

Archive Compression	0%	Server Memory	28%
Write Cache Hit	0%	Consumption Rate	? (MB/Day)
Failed Writes	? (Events)	Historical Tags	8 (8 Used of 1,000,000 Licensed)
Messages Since Startup	? (Messages)	SCADA Tags	0 (0 Used of 2,400 Licensed)
Alerts Since Startup	? (Alerts)	Users	1 (15 Licensed)
Calculations	(Enabled)	Alarm Rate	? (Alarms/Min)
Server to Server	(Enabled)	Alarms Since Startup	? (Alarms)
Max Scada Buffer Duration	201 (Days)	Data Stores	4 (20 Licensed)

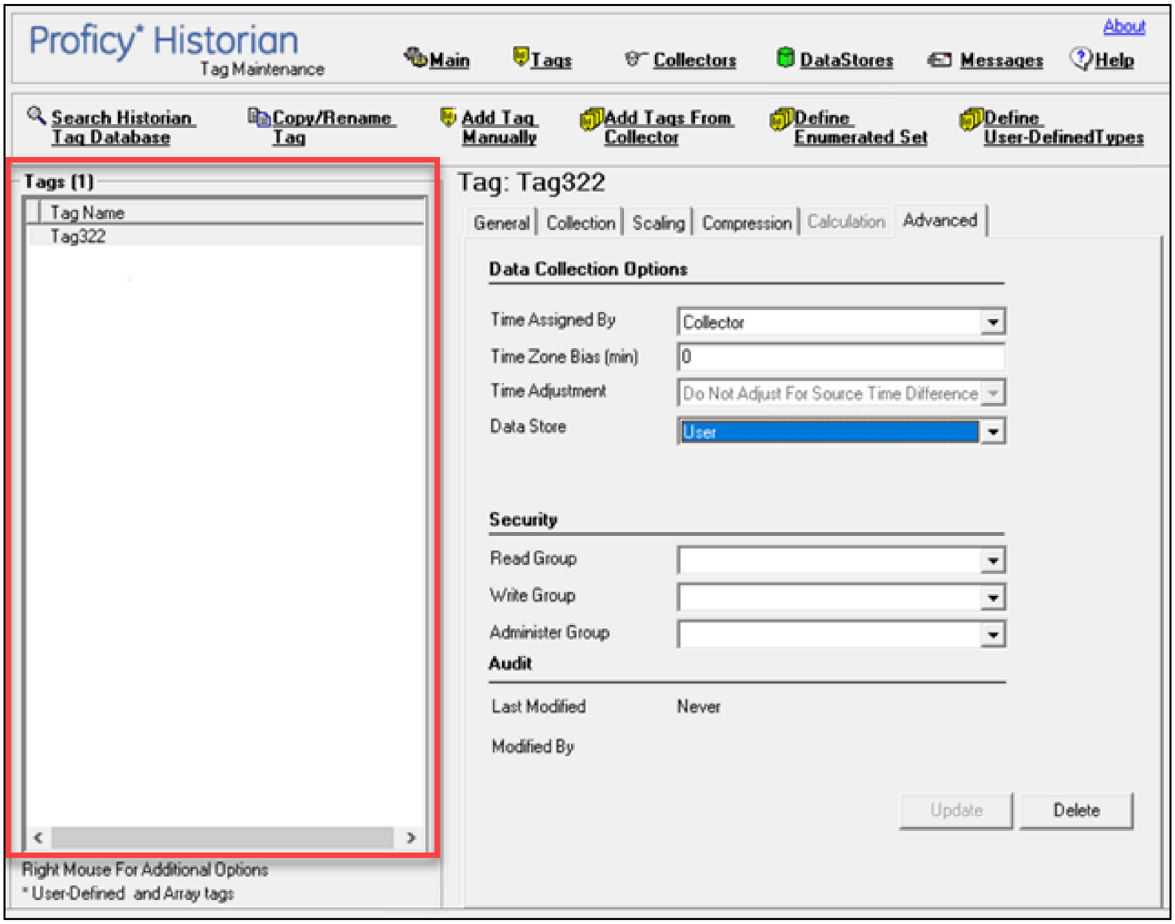
Collectors □ Auto Refresh

Collector	Status	Computer	Report Rate	Overruns	Compression	Out Of Order	Redundancy
SAN-JAVA-DEV_Simulation	Unknown	SAN-JAVA-DEV	0	0	%	0	

Alerts ☑ Show Alerts □ Auto Refresh

Timestamp	Topic	Message

3. Select the tag that you want to rename.



Tag322

Tag: Tag322

Data Collection Options

Time Assigned By: Collector

Time Zone Bias (min): 0

Time Adjustment: Do Not Adjust For Source Time Difference

Data Store: User

Security

Read Group

Write Group

Administer Group

Audit

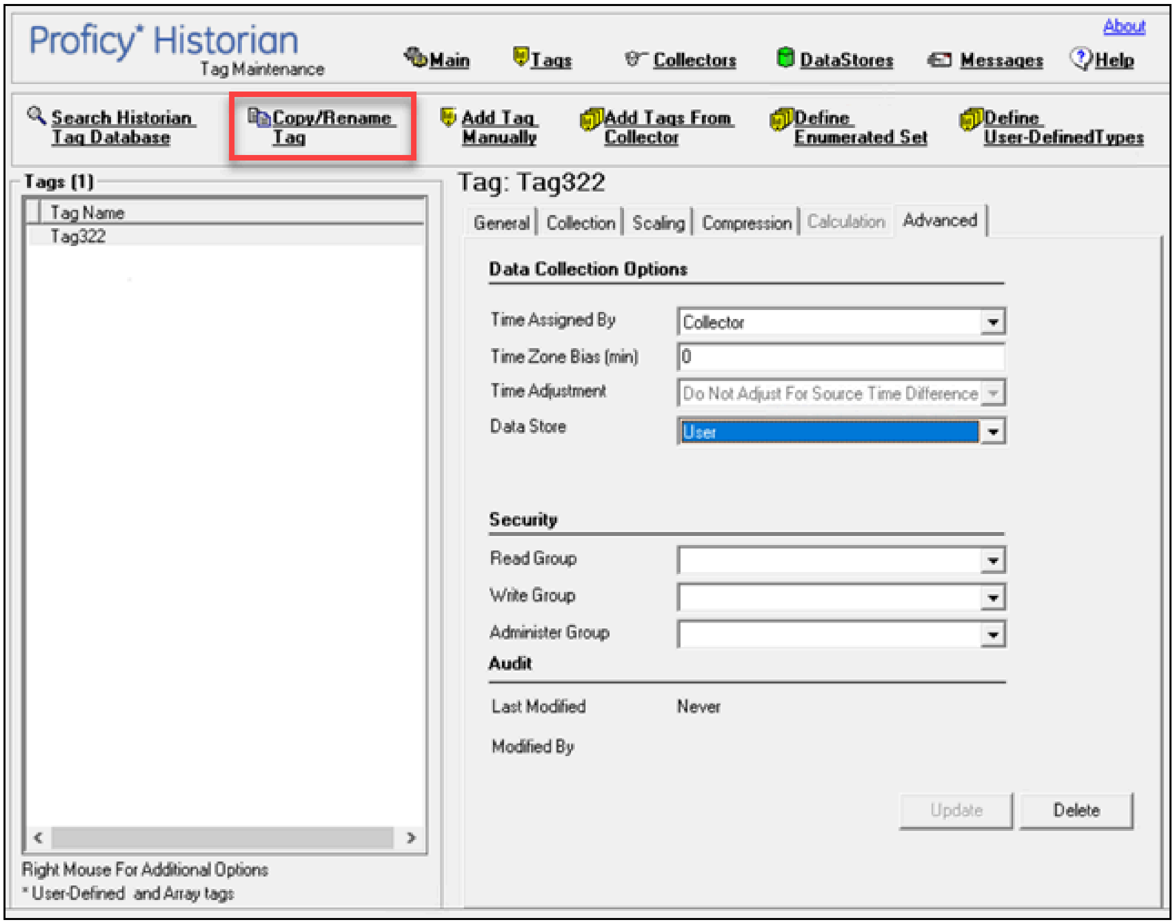
Last Modified: Never

Modified By

Right Mouse For Additional Options

* User-Defined and Array tags

4. Select **Copy/Rename Tag** .



Tag: Tag322

Data Collection Options

Time Assigned By: Collector

Time Zone Bias (min): 0

Time Adjustment: Do Not Adjust For Source Time Difference

Data Store: User

Security

Read Group:

Write Group:

Administer Group:

Audit

Last Modified: Never

Modified By:

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**.

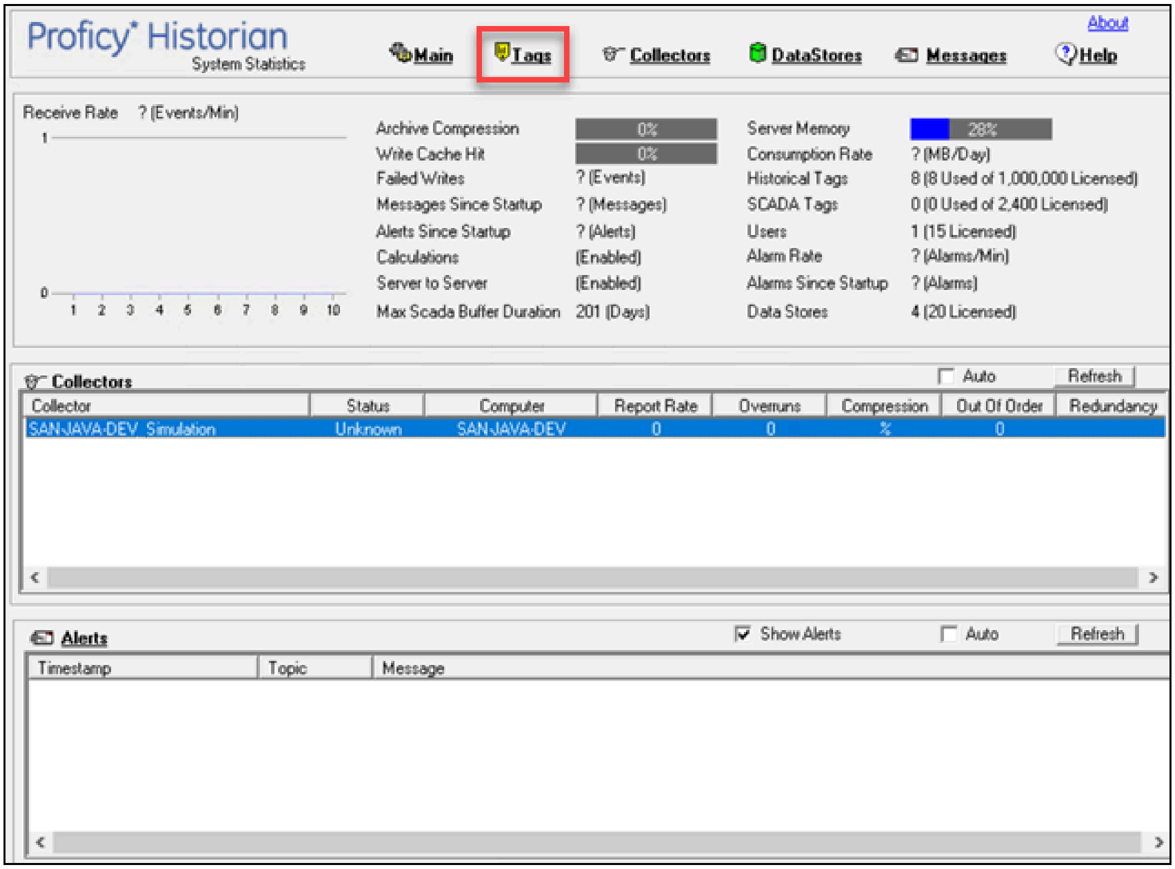
If you have renamed the tag permanently:

- 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.

1. Access Historian Administrator *(on page 1194)*.
2. Select **Tags**.

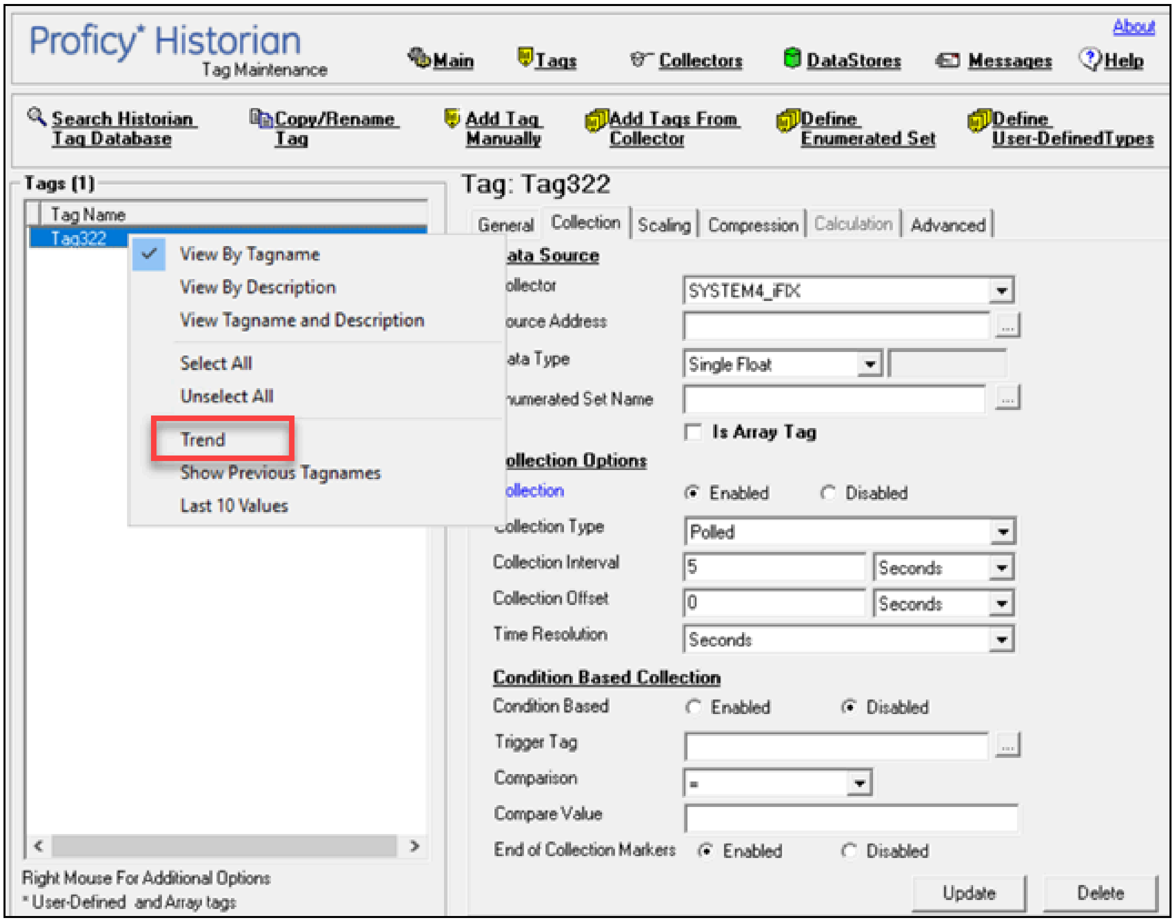


Archive Compression	0%	Server Memory	28%
Write Cache Hit	0%	Consumption Rate	? (MB/Day)
Failed Writes	? (Events)	Historical Tags	8 (8 Used of 1,000,000 Licensed)
Messages Since Startup	? (Messages)	SCADA Tags	0 (0 Used of 2,400 Licensed)
Alerts Since Startup	? (Alerts)	Users	1 (15 Licensed)
Calculations	(Enabled)	Alarm Rate	? (Alarms/Min)
Server to Server	(Enabled)	Alarms Since Startup	? (Alarms)
Max Scada Buffer Duration	201 (Days)	Data Stores	4 (20 Licensed)

Collector	Status	Computer	Report Rate	Overruns	Compression	Out Of Order	Redundancy
SAN-JAVA-DEV_Simulation	Unknown	SAN-JAVA-DEV	0	0	%	0	

Timestamp	Topic	Message

3. Right-click the tag whose trend chart you want to access, and then select **Trend**.



Tag: Tag322

ata Source

ollector: SYSTEM4_iFIX

ource Address:

ata Type: Single Float

numerated Set Name:

□ Is Array Tag

ollection Options

ollection: ● Enabled ○ Disabled

ollection Type: Polled

Collection Interval: 5 Seconds

Collection Offset: 0 Seconds

Time Resolution: Seconds

Condition Based Collection

Condition Based: ○ Enabled ● Disabled

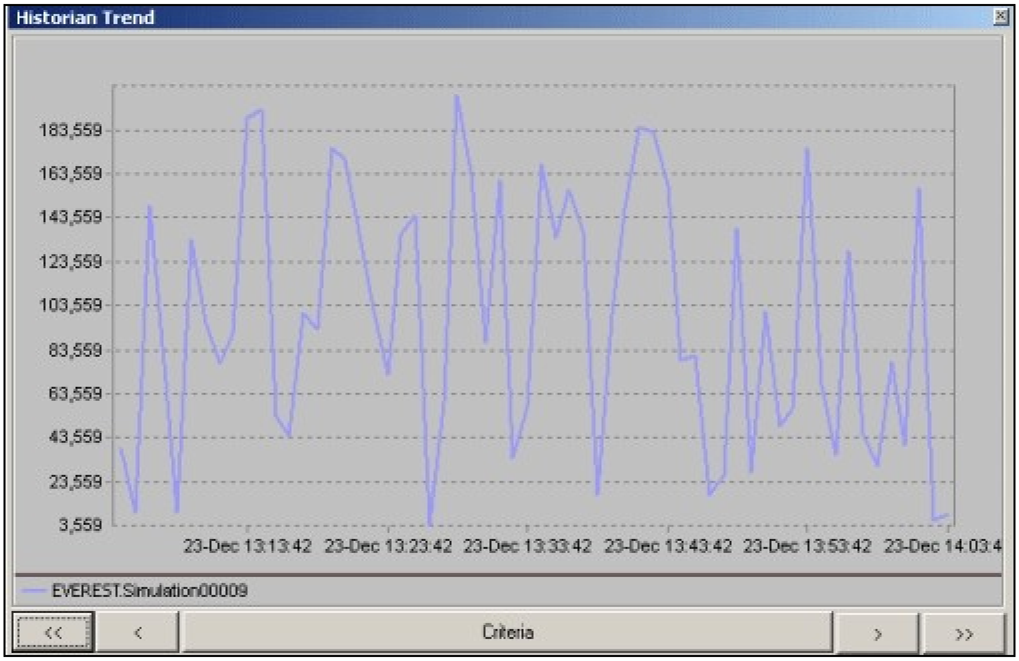
Trigger Tag:

Comparison: =

Compare Value:

End of Collection Markers: ● Enabled ○ Disabled

The trend chart of the tag values appears.



Note:

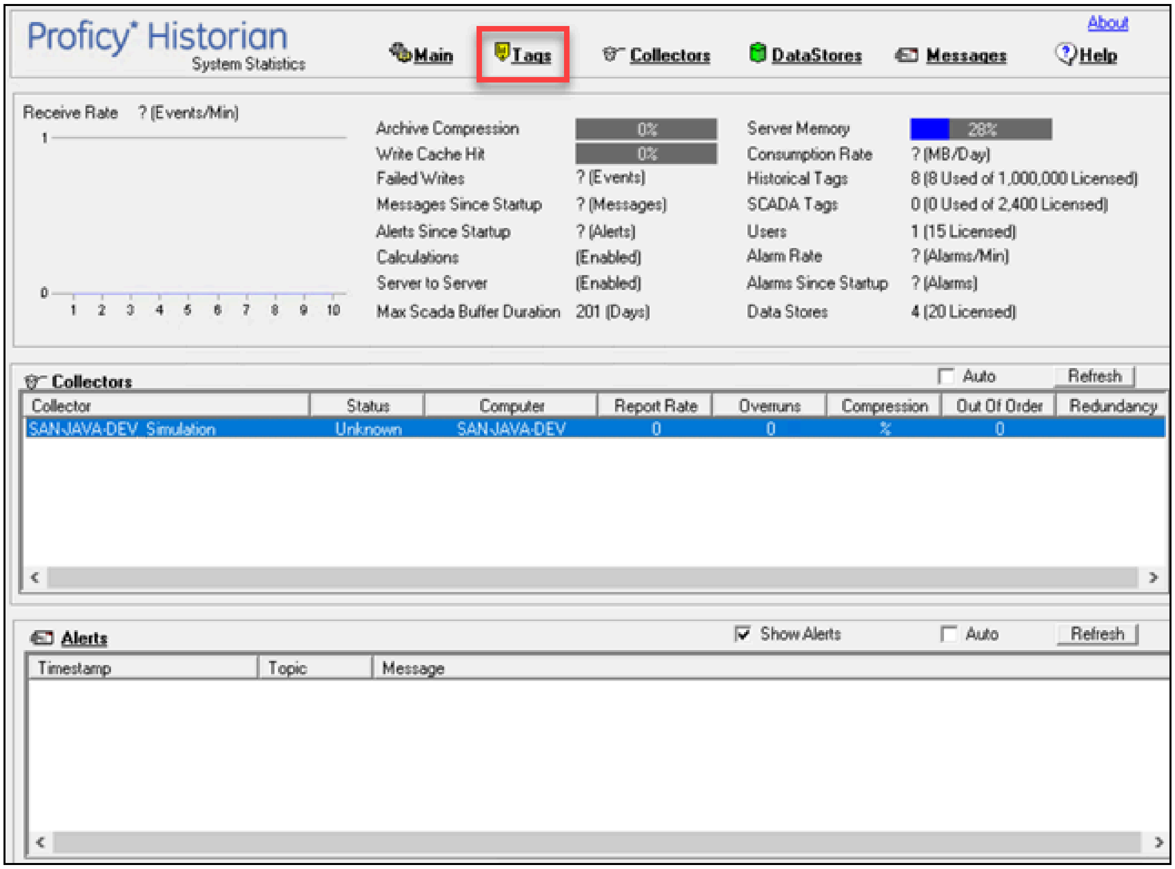
To change the criteria, select **Criteria**, and then enter values as described in the following table.

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 1194)*.
2. Select **Tags**.



Receive Rate ? (Events/Min)

Archive Compression	0%	Server Memory	28%
Write Cache Hit	0%	Consumption Rate	? (MB/Day)
Failed Writes	? (Events)	Historical Tags	8 (8 Used of 1,000,000 Licensed)
Messages Since Startup	? (Messages)	SCADA Tags	0 (0 Used of 2,400 Licensed)
Alerts Since Startup	? (Alerts)	Users	1 (15 Licensed)
Calculations	(Enabled)	Alarm Rate	? (Alarms/Min)
Server to Server	(Enabled)	Alarms Since Startup	? (Alarms)
Max Scada Buffer Duration	201 (Days)	Data Stores	4 (20 Licensed)

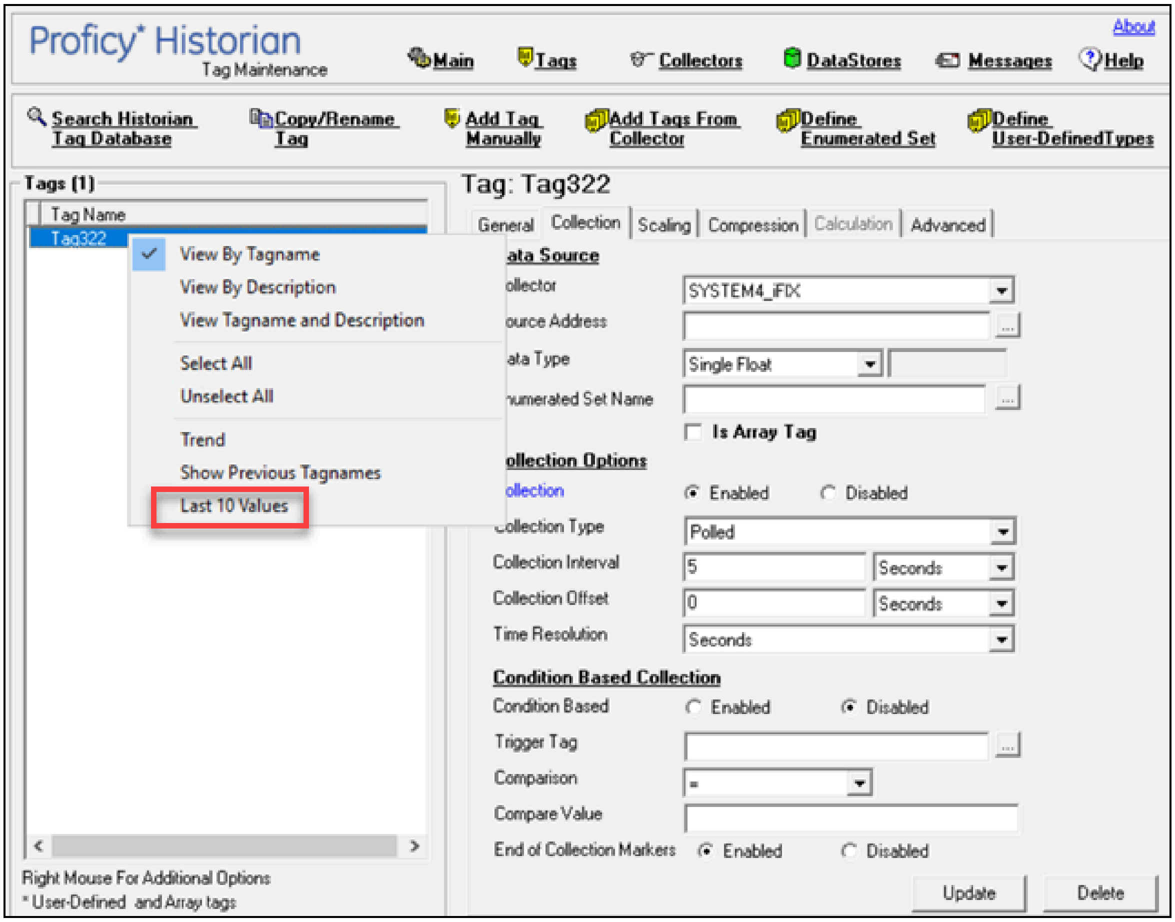
Collectors □ Auto Refresh

Collector	Status	Computer	Report Rate	Overruns	Compression	Out Of Order	Redundancy
SAN-JAVA-DEV_Simulation	Unknown	SAN-JAVA-DEV	0	0	%	0	

Alerts ☑ Show Alerts □ Auto Refresh

Timestamp	Topic	Message

3. Right-click the tag whose last 10 values you want to access, and then select **Last 10 Values**.



Tag: Tag322

ata Source

ollector: SYSTEM4_iFIX

ource Address:

ata Type: Single Float

numerated Set Name:

□ Is Array Tag

ollection Options

ollection: ● Enabled ○ Disabled

ollection Type: Polled

Collection Interval: 5 Seconds

Collection Offset: 0 Seconds

Time Resolution: Seconds

Condition Based Collection

Condition Based: ○ Enabled ● Disabled

Trigger Tag:

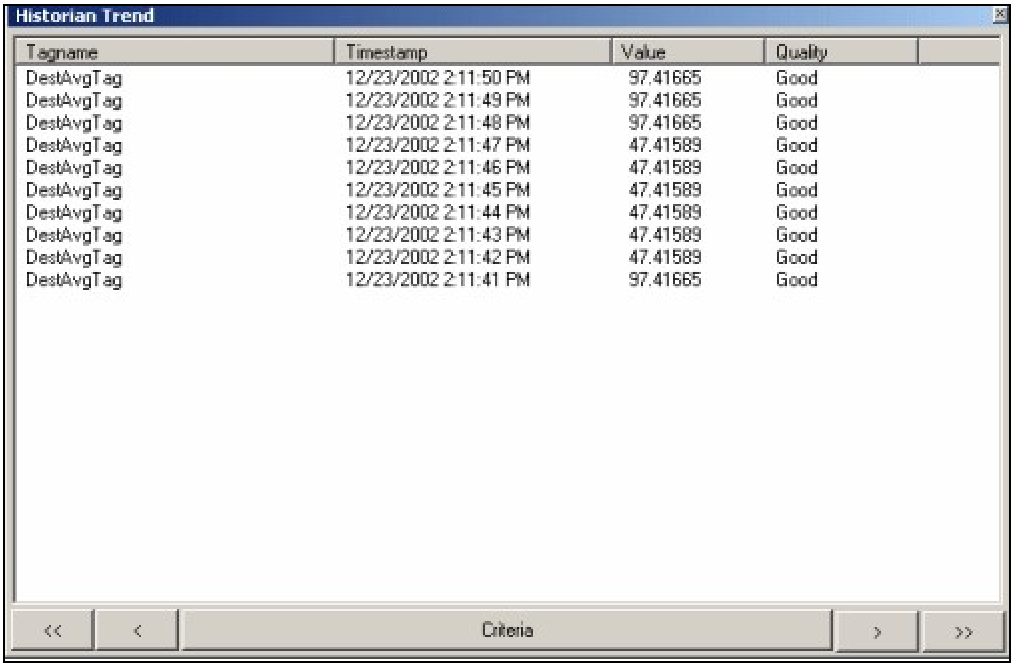
Comparison: =

Compare Value:

End of Collection Markers: ● Enabled ○ Disabled

Right Mouse For Additional Options
* User-Defined and Array tags

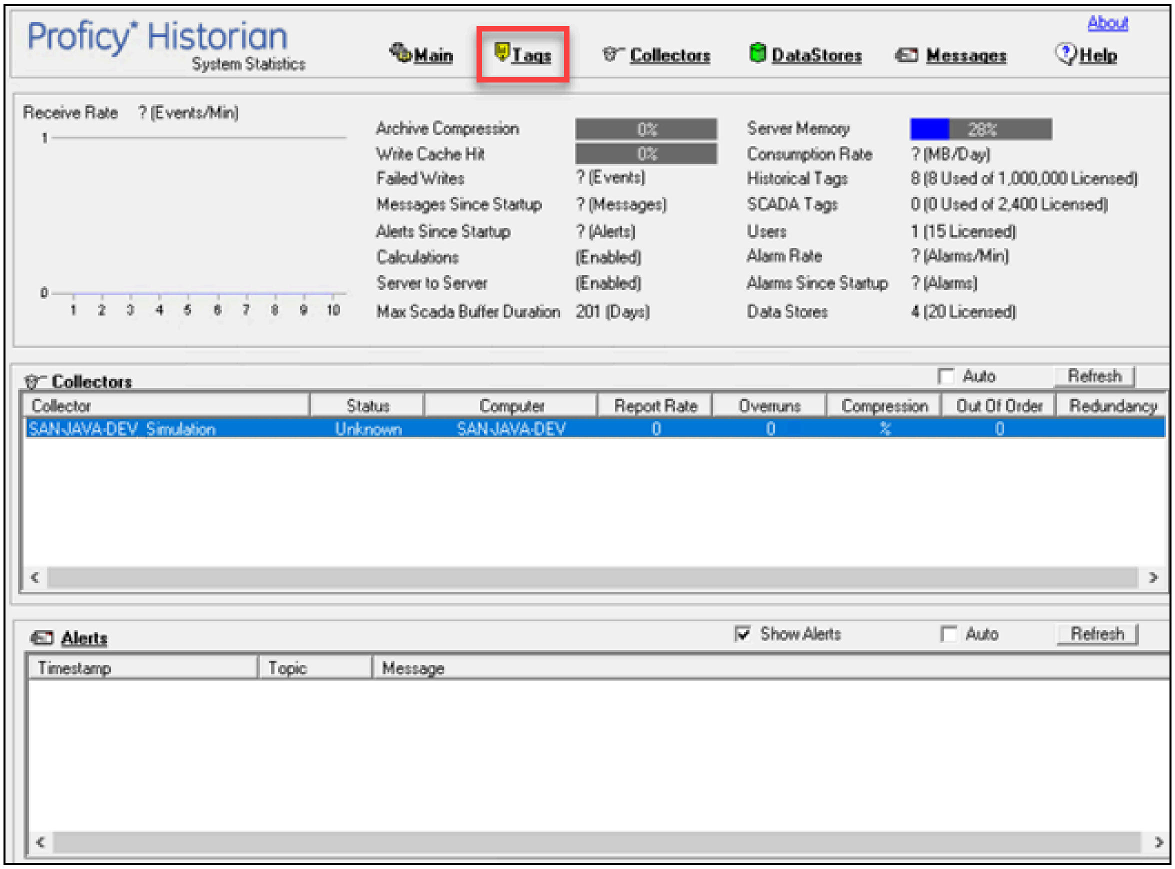
The last 10 values of the tag appear.



Tagname	Timestamp	Value	Quality
DestAvgTag	12/23/2002 2:11:50 PM	97.41665	Good
DestAvgTag	12/23/2002 2:11:49 PM	97.41665	Good
DestAvgTag	12/23/2002 2:11:48 PM	97.41665	Good
DestAvgTag	12/23/2002 2:11:47 PM	47.41589	Good
DestAvgTag	12/23/2002 2:11:46 PM	47.41589	Good
DestAvgTag	12/23/2002 2:11:45 PM	47.41589	Good
DestAvgTag	12/23/2002 2:11:44 PM	47.41589	Good
DestAvgTag	12/23/2002 2:11:43 PM	47.41589	Good
DestAvgTag	12/23/2002 2:11:42 PM	47.41589	Good
DestAvgTag	12/23/2002 2:11:41 PM	97.41665	Good

Stop Data Collection

1. Access Historian Administrator *(on page 1194)*.
2. Select **Tags**.



Receive Rate ? (Events/Min)

Archive Compression	0%	Server Memory	28%
Write Cache Hit	0%	Consumption Rate	? (MB/Day)
Failed Writes	? (Events)	Historical Tags	8 (8 Used of 1,000,000 Licensed)
Messages Since Startup	? (Messages)	SCADA Tags	0 (0 Used of 2,400 Licensed)
Alerts Since Startup	? (Alerts)	Users	1 (15 Licensed)
Calculations	(Enabled)	Alarm Rate	? (Alarms/Min)
Server to Server	(Enabled)	Alarms Since Startup	? (Alarms)
Max Scada Buffer Duration	201 (Days)	Data Stores	4 (20 Licensed)

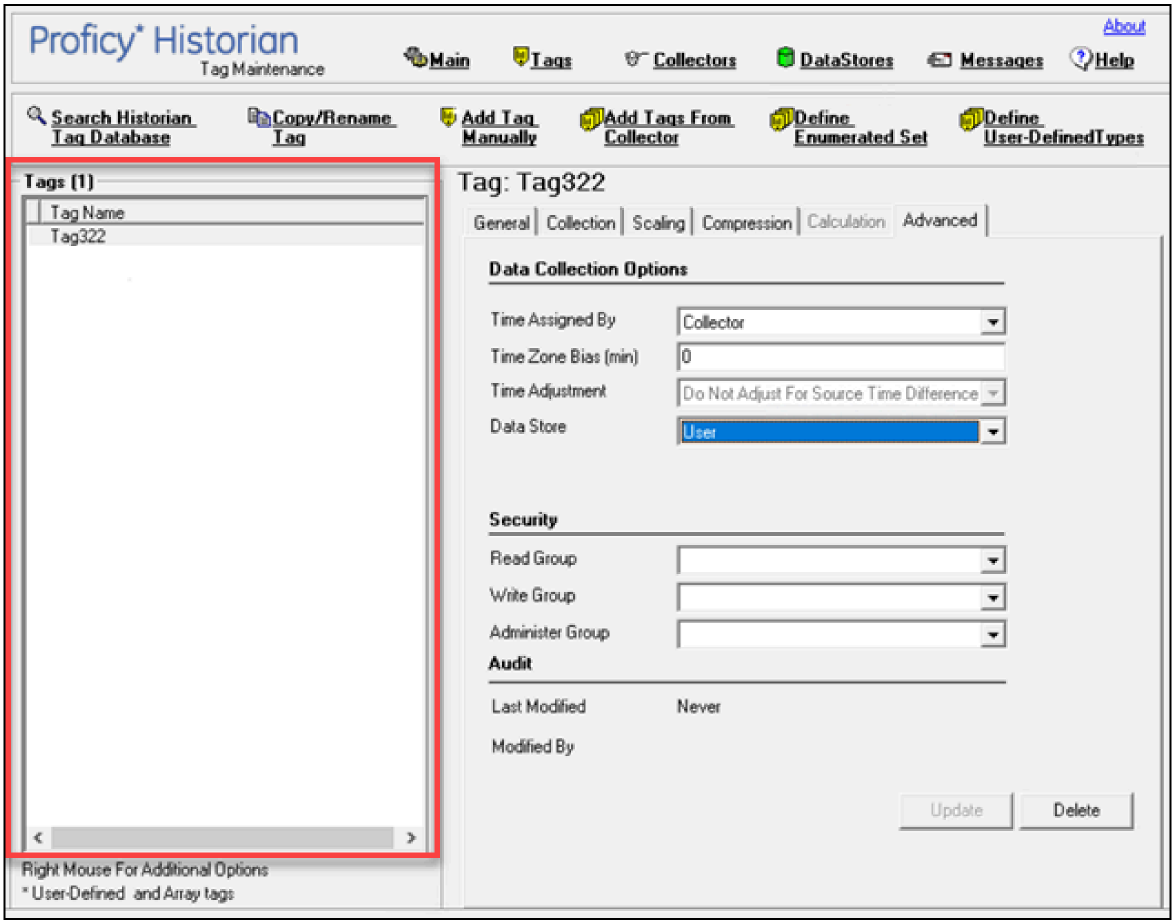
Collectors

Collector	Status	Computer	Report Rate	Overruns	Compression	Out Of Order	Redundancy
SAN-JAVA-DEV_Simulation	Unknown	SAN-JAVA-DEV	0	0	%	0	

Alerts

Timestamp	Topic	Message

3. Select the tag whose data collection you want to stop.



Data Collection Options

Time Assigned By: Collector

Time Zone Bias (min): 0

Time Adjustment: Do Not Adjust For Source Time Difference

Data Store: User

Security

Read Group

Write Group

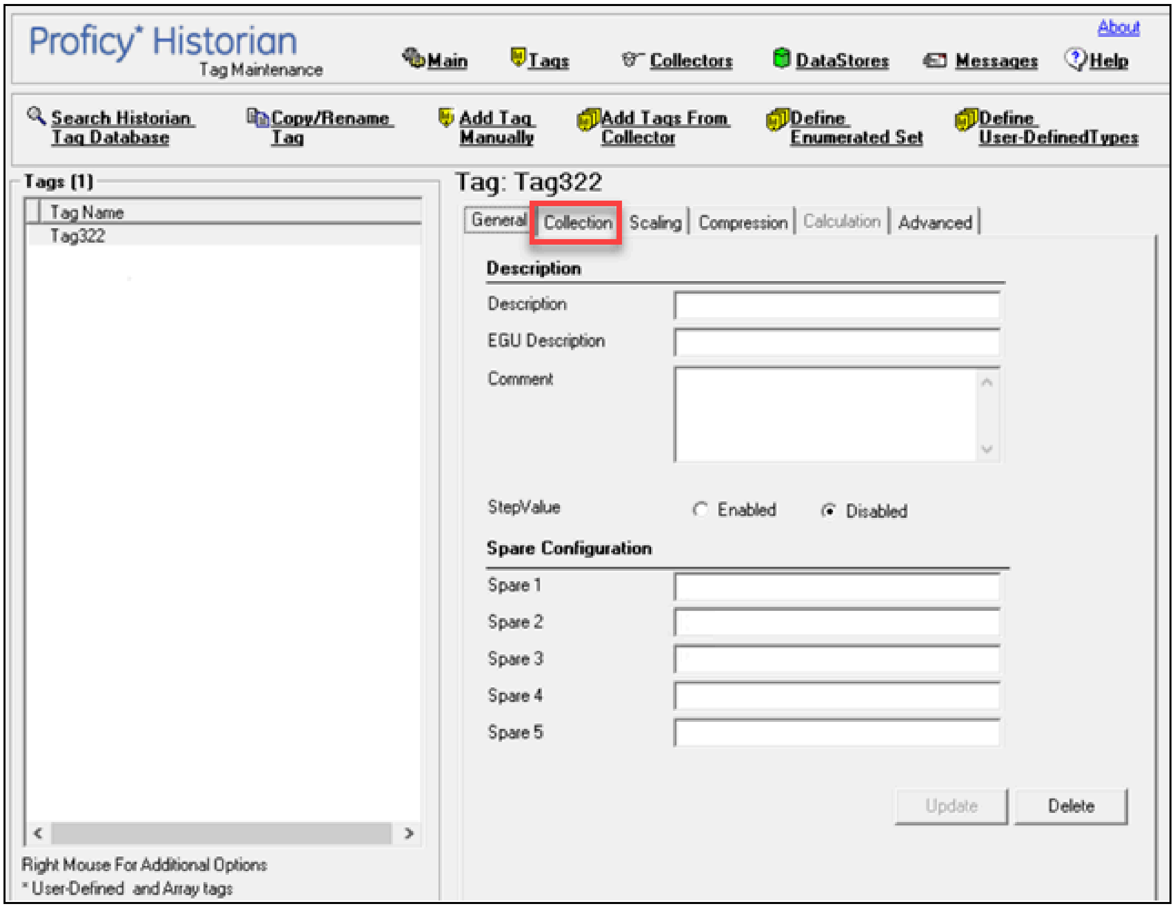
Administer Group

Audit

Last Modified: Never

Modified By

4. Select **Collection**.



Tags (1)

Tag Name
Tag322

Tag: Tag322

Description

Description

EGU Description

Comment

StepValue ○ Enabled ● Disabled

Spare Configuration

Spare 1

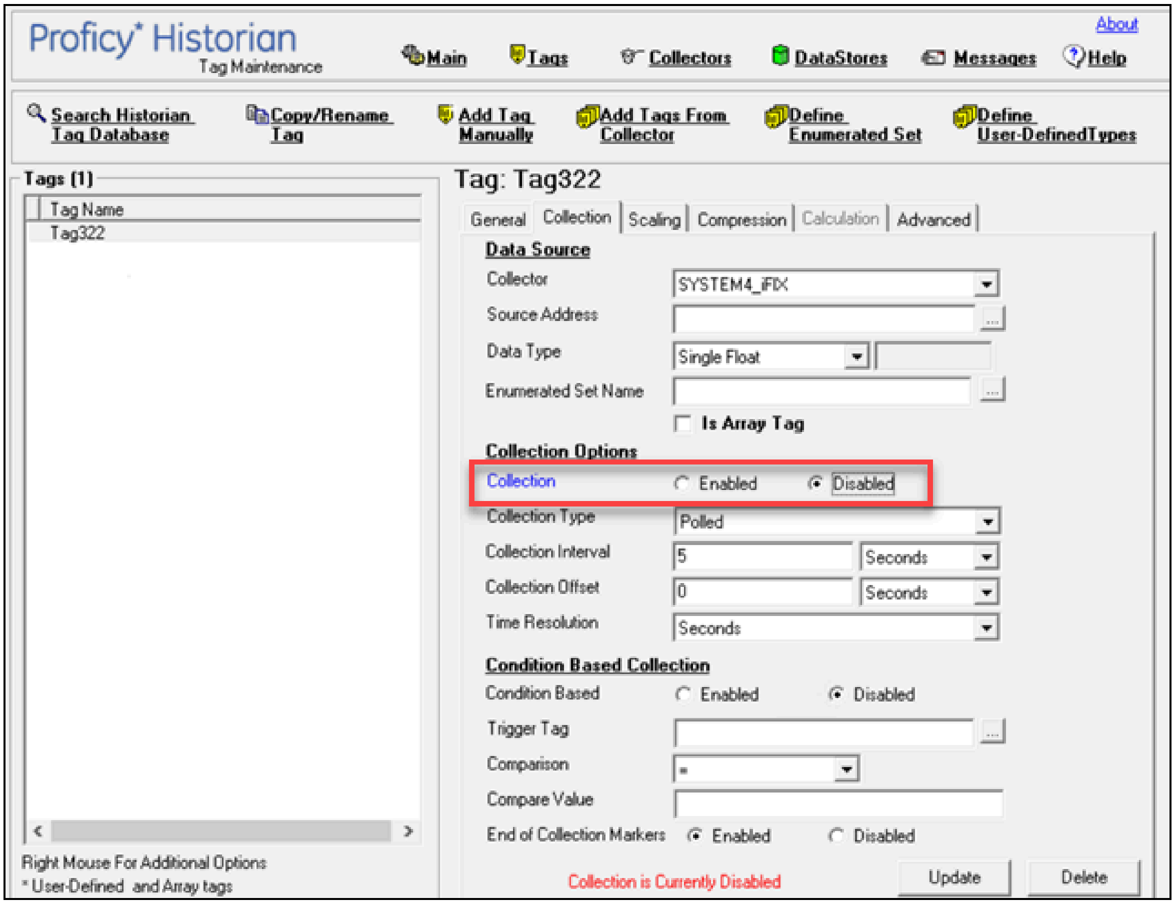
Spare 2

Spare 3

Spare 4

Spare 5

5. For the **Collection** field, select **Disabled**.



Tags (1)

Tag Name
Tag322

Right Mouse For Additional Options
* User-Defined and Array tags

Tag: Tag322

Data Source

Collector: SYSTEM4_iFIX
Source Address:
Data Type: Single Float
Enumerated Set Name:
□ Is Array Tag

Collection Options

Collection ○ Enabled ● Disabled
Collection Type: Polled
Collection Interval: 5 Seconds
Collection Offset: 0 Seconds
Time Resolution: Seconds

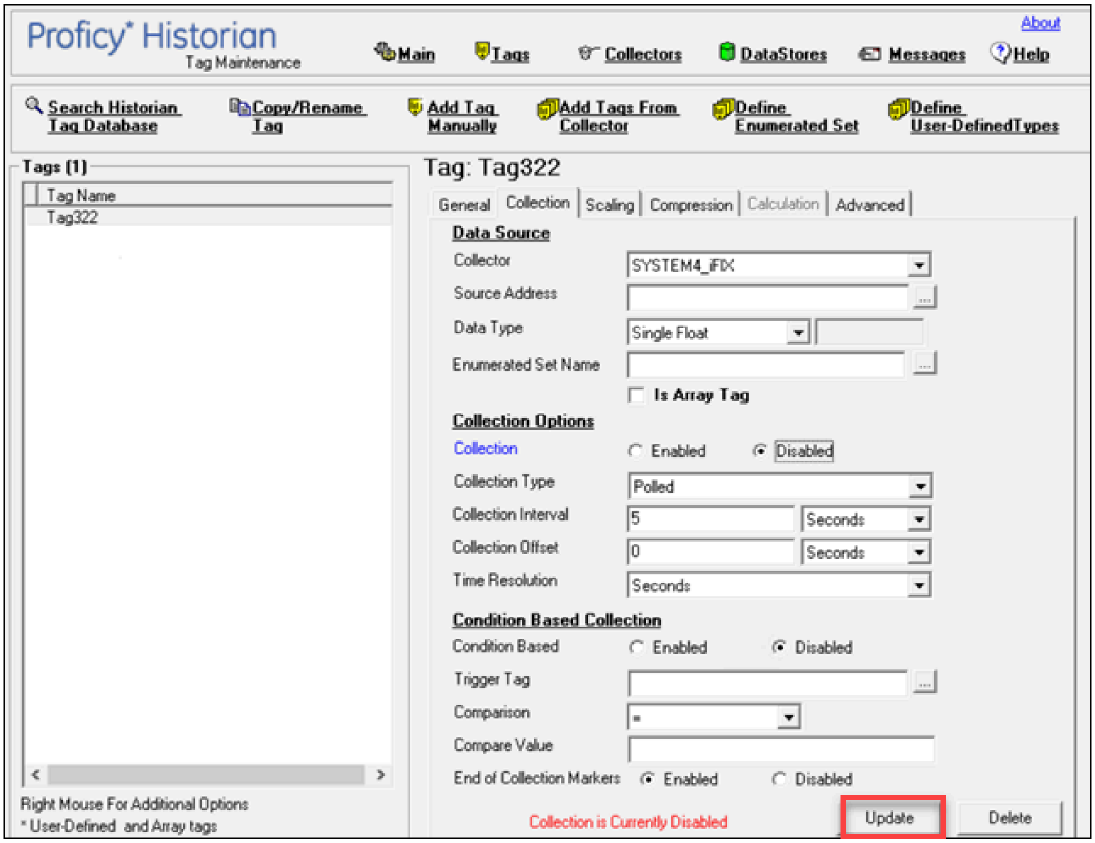
Condition Based Collection

Condition Based ○ Enabled ● Disabled
Trigger Tag:
Comparison: =
Compare Value:
End of Collection Markers ● Enabled ○ Disabled

Collection is Currently 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**.



Tags (1)

Tag Name
Tag322

Right Mouse For Additional Options
* User-Defined and Array tags

Tag: Tag322

Data Source

Collector: SYSTEM4_iFIX
Source Address:
Data Type: Single Float
Enumerated Set Name:
□ Is Array Tag

Collection Options

Collection: ○ Enabled ● Disabled
Collection Type: Polled
Collection Interval: 5 Seconds
Collection Offset: 0 Seconds
Time Resolution: Seconds

Condition Based Collection

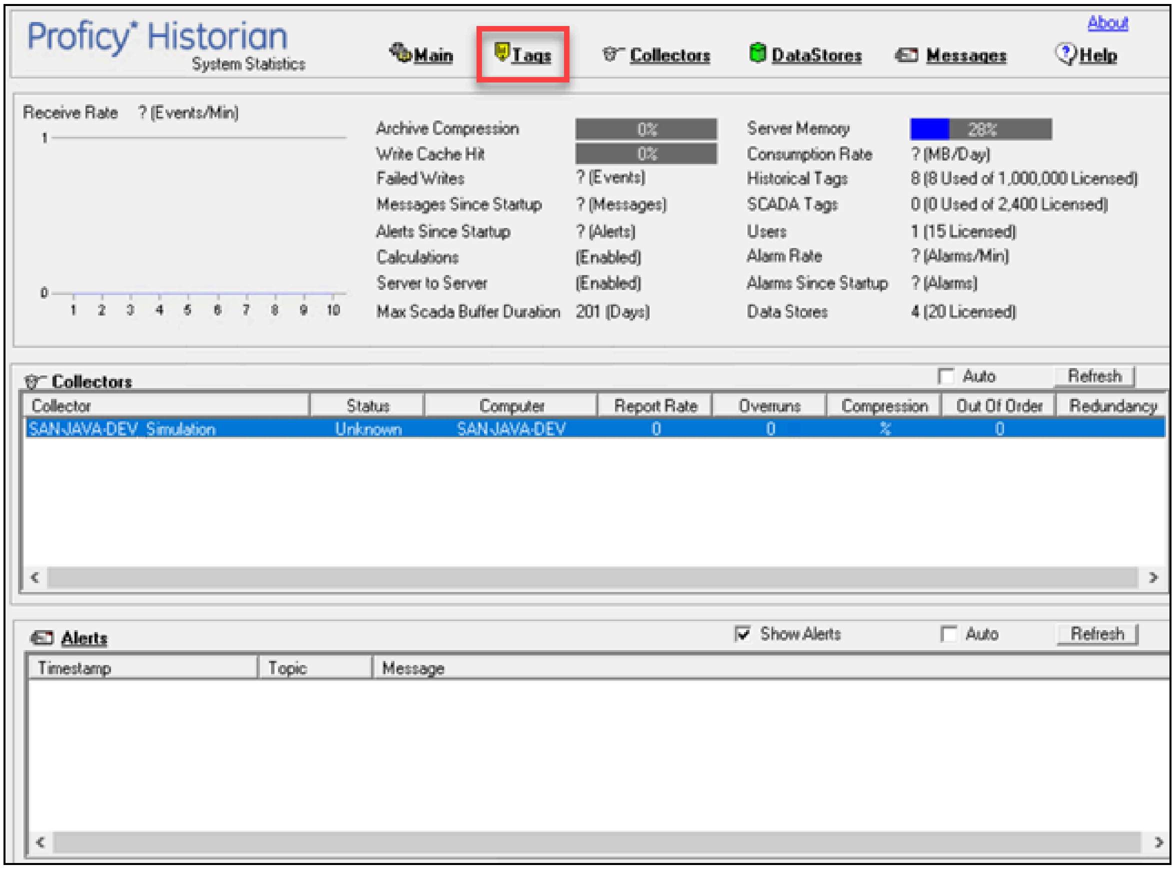
Condition Based: ○ Enabled ● Disabled
Trigger Tag:
Comparison: =
Compare Value:
End of Collection Markers: ● Enabled ○ Disabled

Collection is Currently Disabled

The data collection for the tag is stopped.

Resume Data Collection

1. Access Historian Administrator *(on page 1194)*.
2. Select **Tags**.



Receive Rate ? (Events/Min)

Archive Compression	0%	Server Memory	28%
Write Cache Hit	0%	Consumption Rate	? (MB/Day)
Failed Writes	? (Events)	Historical Tags	8 (8 Used of 1,000,000 Licensed)
Messages Since Startup	? (Messages)	SCADA Tags	0 (0 Used of 2,400 Licensed)
Alerts Since Startup	? (Alerts)	Users	1 (15 Licensed)
Calculations	(Enabled)	Alarm Rate	? (Alarms/Min)
Server to Server	(Enabled)	Alarms Since Startup	? (Alarms)
Max Scada Buffer Duration	201 (Days)	Data Stores	4 (20 Licensed)

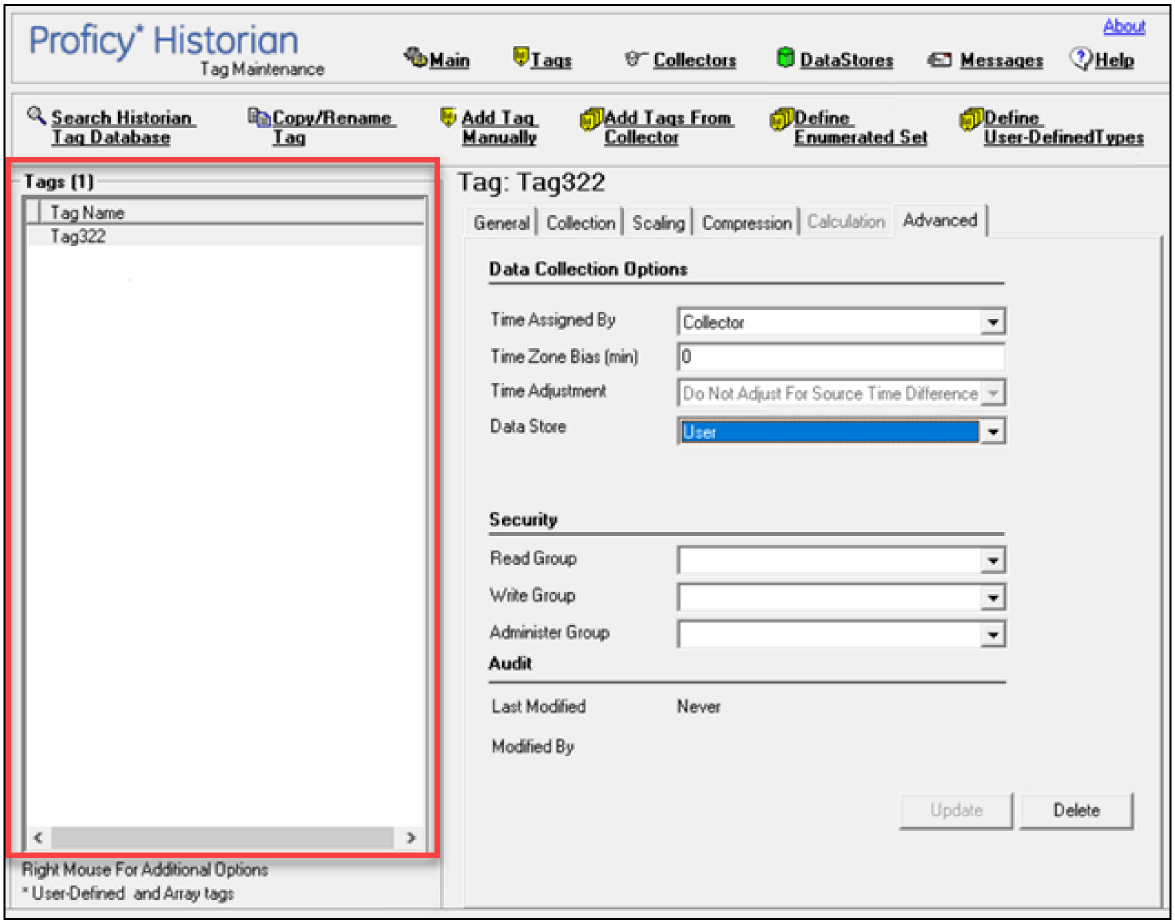
Collectors □ Auto Refresh

Collector	Status	Computer	Report Rate	Overruns	Compression	Out Of Order	Redundancy
SAN-JAVA-DEV_Simulation	Unknown	SAN-JAVA-DEV	0	0	%	0	

Alerts ☑ Show Alerts □ Auto Refresh

Timestamp	Topic	Message

3. Select the tag whose data collection you want to resume.



Tag: Tag322

Data Collection Options

Time Assigned By: Collector

Time Zone Bias (min): 0

Time Adjustment: Do Not Adjust For Source Time Difference

Data Store: User

Security

Read Group:

Write Group:

Administer Group:

Audit

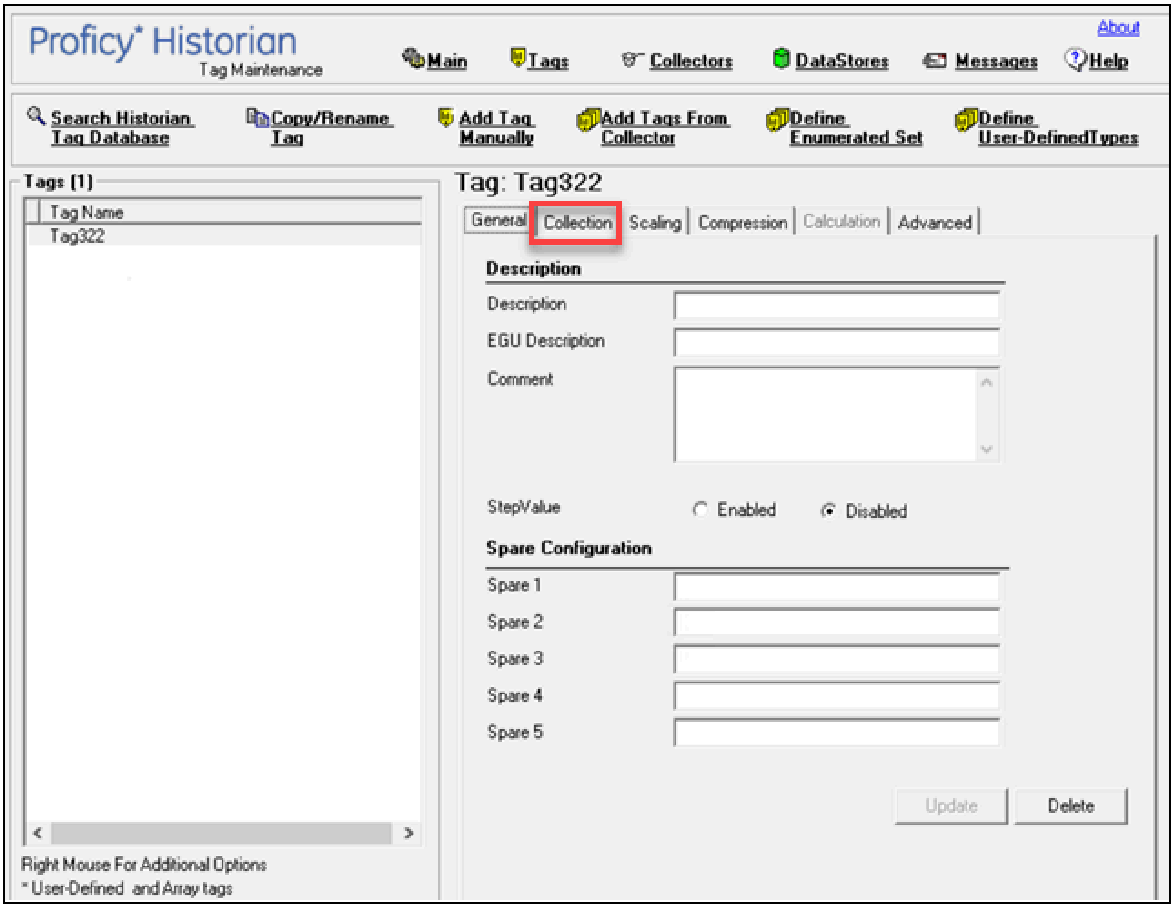
Last Modified: Never

Modified By:

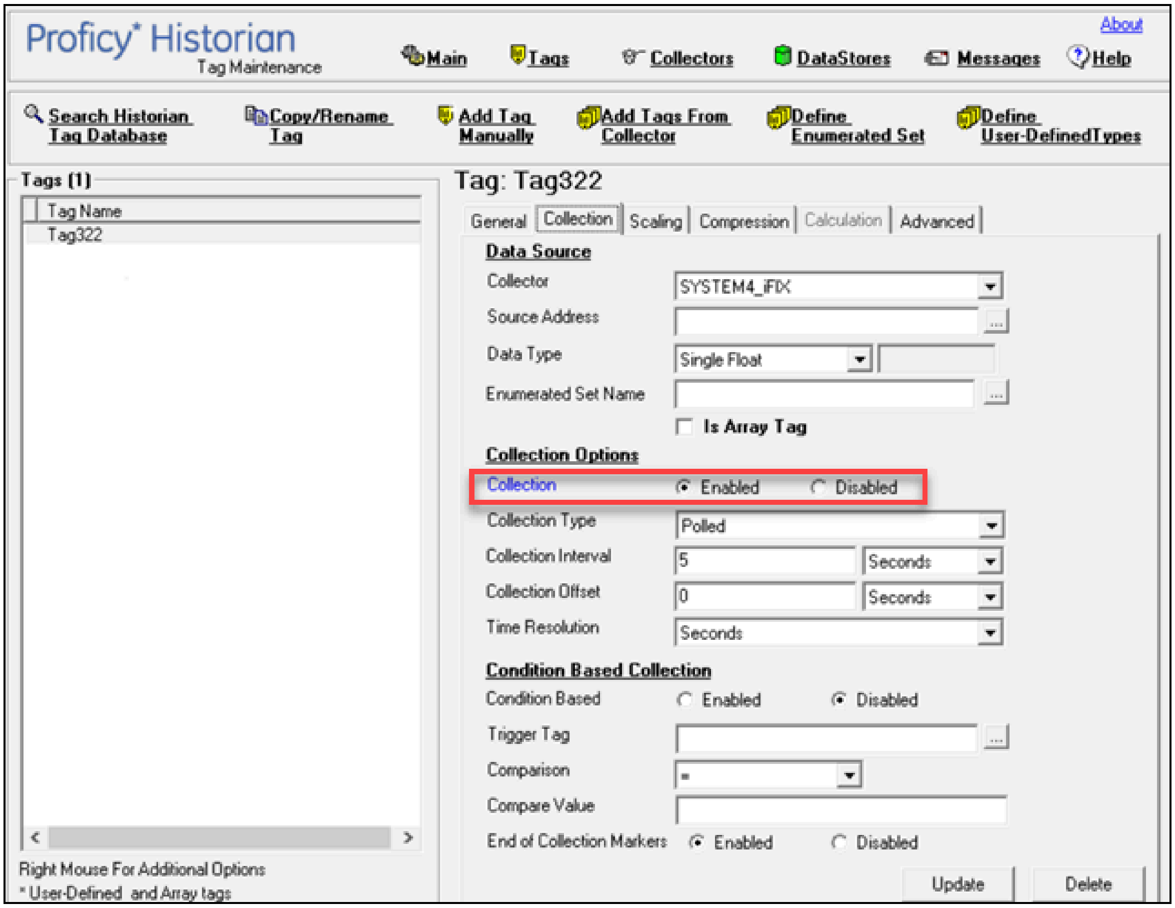
Right Mouse For Additional Options

* User-Defined and Array tags

4. Select **Collection**.



5. For the **Collection** field, select **Enabled**.



Tag: Tag322

Data Source

Collector: SYSTEM4_iFIX

Source Address:

Data Type: Single Float

Enumerated Set Name:

□ Is Array Tag

Collection Options

Collection: ● Enabled ○ Disabled

Collection Type: Polled

Collection Interval: 5 Seconds

Collection Offset: 0 Seconds

Time Resolution: Seconds

Condition Based Collection

Condition Based: ○ Enabled ● Disabled

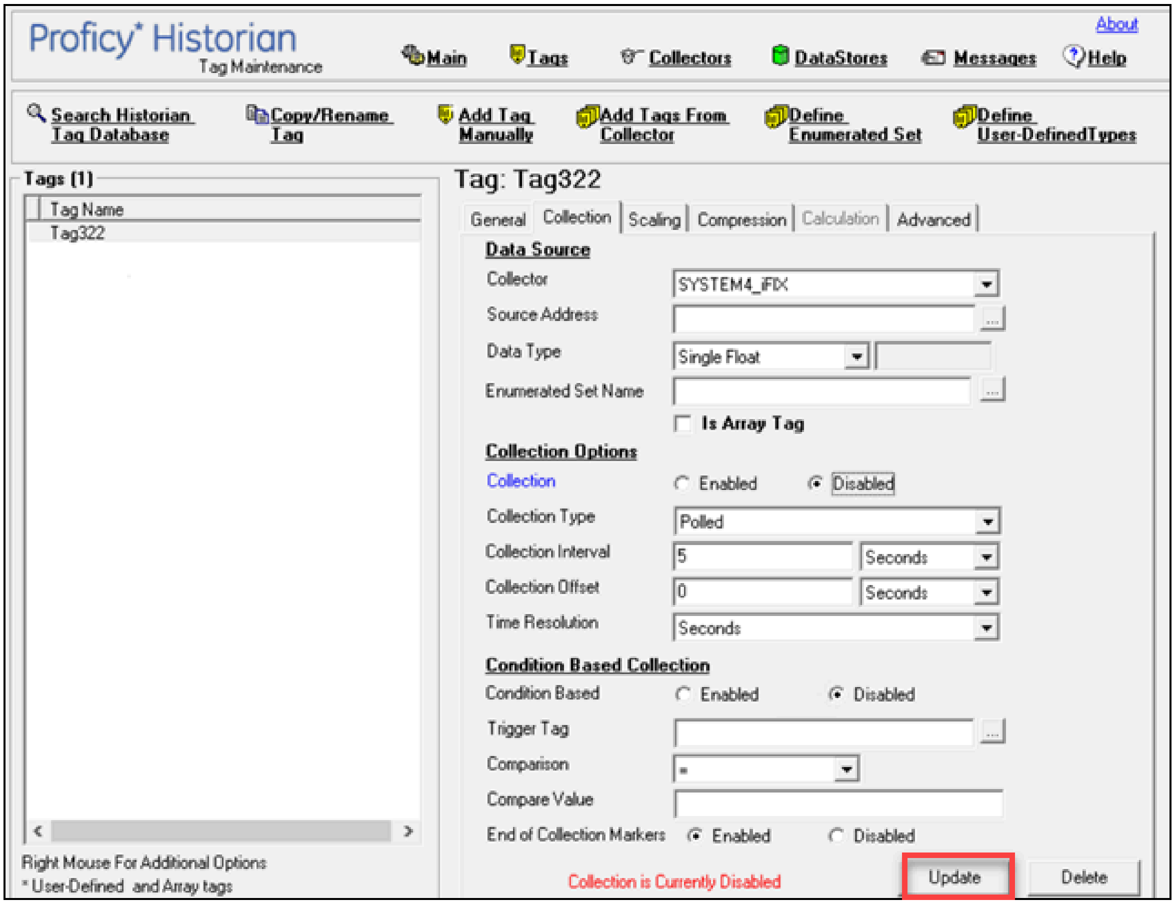
Trigger Tag:

Comparison: =

Compare Value:

End of Collection Markers: ● Enabled ○ Disabled

6. Select **Update**.



Tag: Tag322

Data Source

Collector SYSTEM4_iFIX

Source Address

Data Type Single Float

Enumerated Set Name

□ Is Array Tag

Collection Options

Collection ○ Enabled ● Disabled

Collection Type Polled

Collection Interval 5 Seconds

Collection Offset 0 Seconds

Time Resolution Seconds

Condition Based Collection

Condition Based ○ Enabled ● Disabled

Trigger Tag

Comparison =

Compare Value

End of Collection Markers ● Enabled ○ Disabled

Collection is Currently Disabled

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 1289)*.

Whenever you delete/remove tags, the following collectors reload only the modified tag(s) without restarting the collectors.

- OPC Collector
- iFIX Collector
- Simulation Collector
- Server-to-Server Collector
- OSI PI Collector

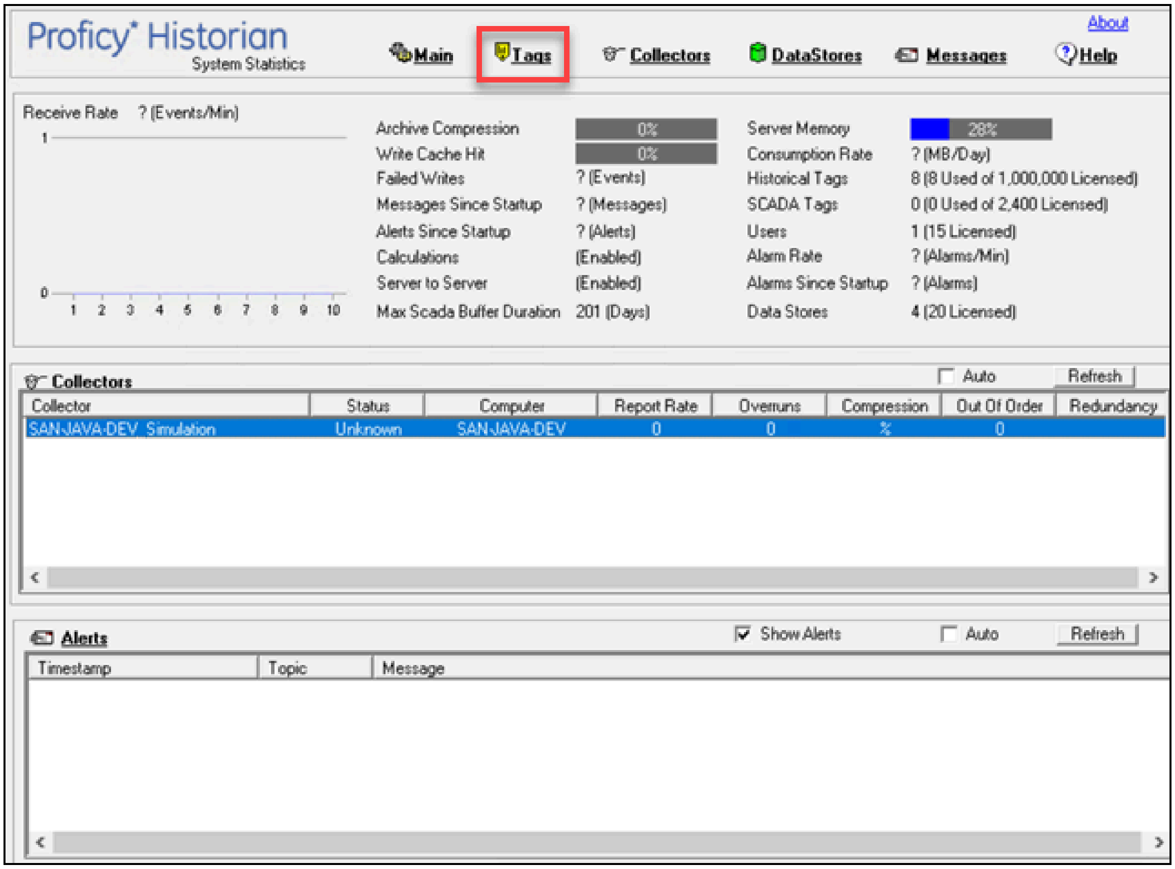
By default, the On-line Tag Configuration Changes option is enabled, which allows a tag to stop and restart data collection without restarting the collector. If you disable the On-line Tag Configuration Changes option, any changes you make to the tags do not affect collection until you restart the collector. 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 1194)*.
2. Select **Tags**.



Receive Rate ? (Events/Min)

Archive Compression	0%	Server Memory	28%
Write Cache Hit	0%	Consumption Rate	? (MB/Day)
Failed Writes	? (Events)	Historical Tags	8 (8 Used of 1,000,000 Licensed)
Messages Since Startup	? (Messages)	SCADA Tags	0 (0 Used of 2,400 Licensed)
Alerts Since Startup	? (Alerts)	Users	1 (15 Licensed)
Calculations	(Enabled)	Alarm Rate	? (Alarms/Min)
Server to Server	(Enabled)	Alarms Since Startup	? (Alarms)
Max Scada Buffer Duration	201 (Days)	Data Stores	4 (20 Licensed)

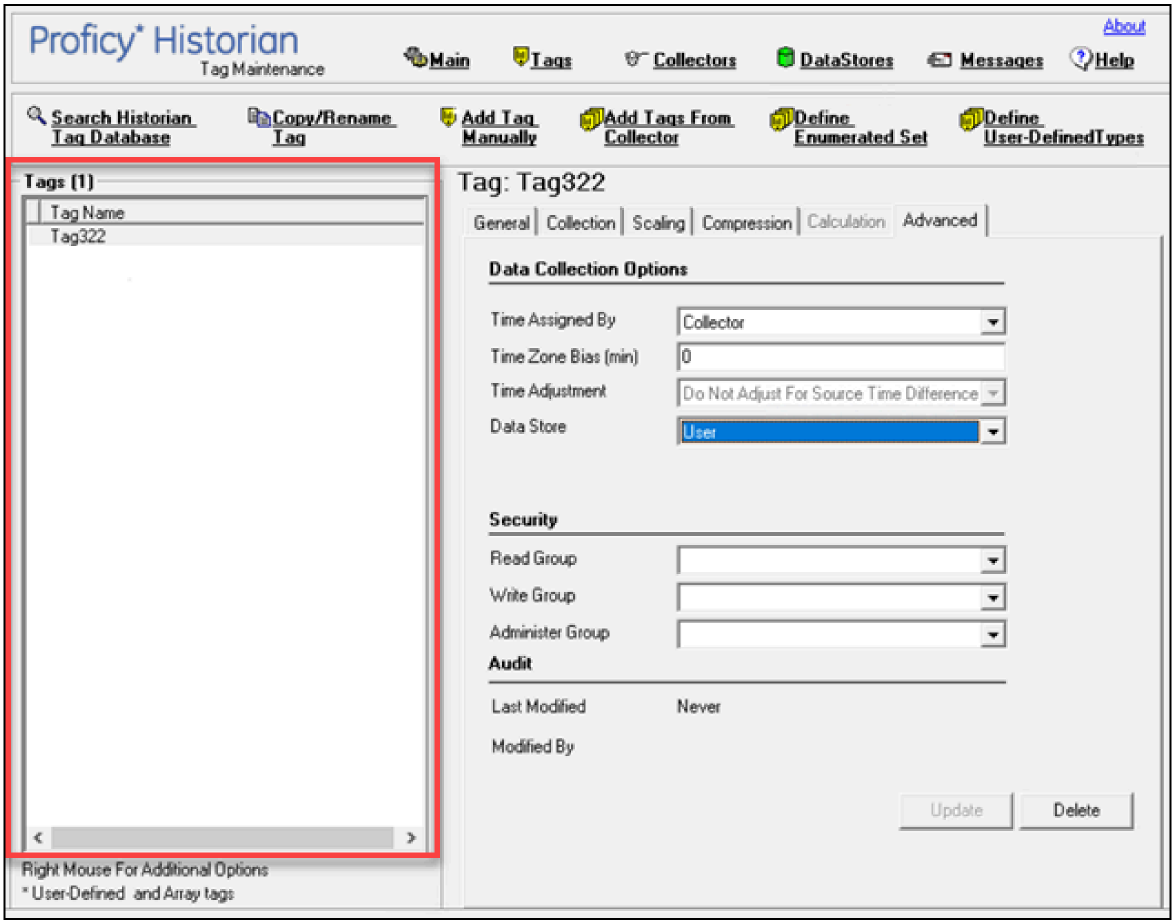
Collectors ☐ Auto Refresh

Collector	Status	Computer	Report Rate	Overruns	Compression	Out Of Order	Redundancy
SAN-JAVA-DEV_Simulation	Unknown	SAN-JAVA-DEV	0	0	%	0	

Alerts ☑ Show Alerts ☐ Auto Refresh

Timestamp	Topic	Message

3. Select the tag that you want to remove.



Tag: Tag322

Data Collection Options

Time Assigned By: Collector

Time Zone Bias (min): 0

Time Adjustment: Do Not Adjust For Source Time Difference

Data Store: User

Security

Read Group:

Write Group:

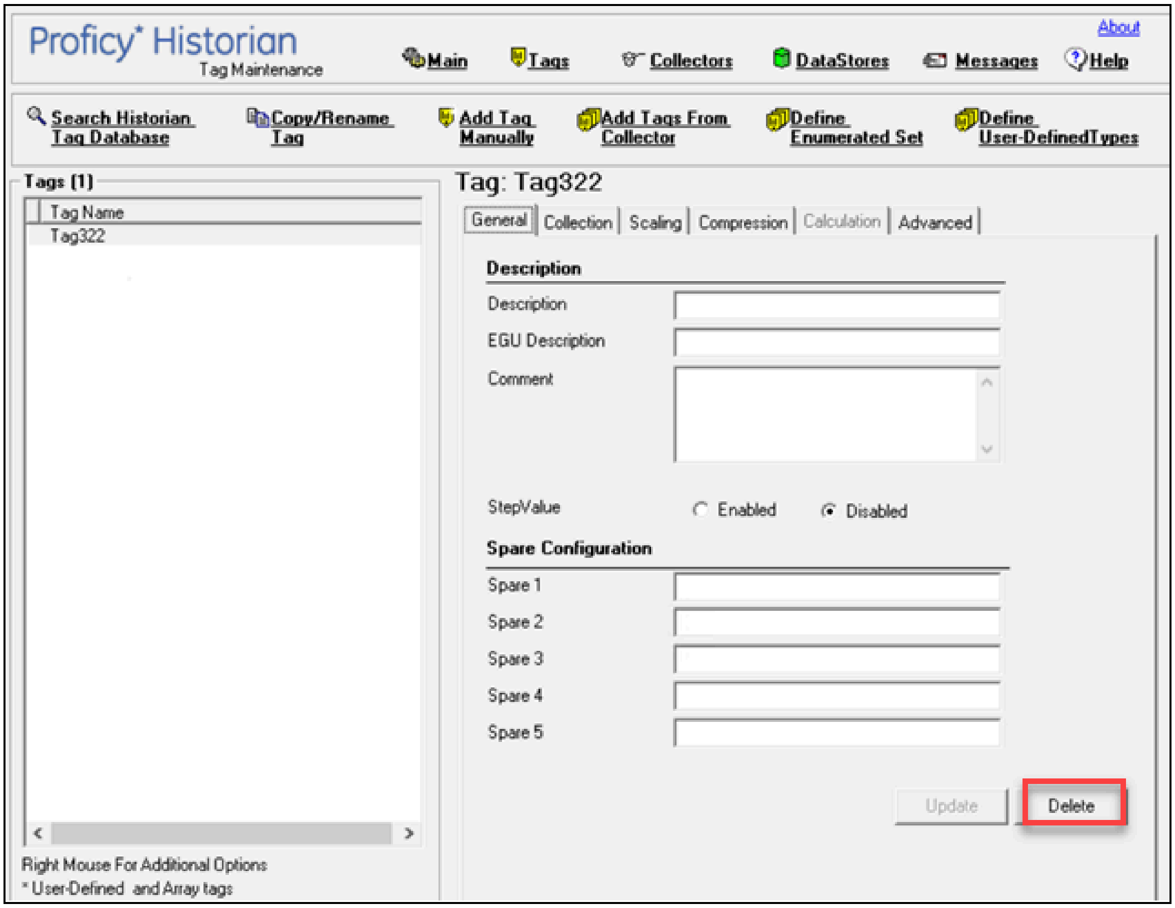
Administer Group:

Audit

Last Modified: Never

Modified By:

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.

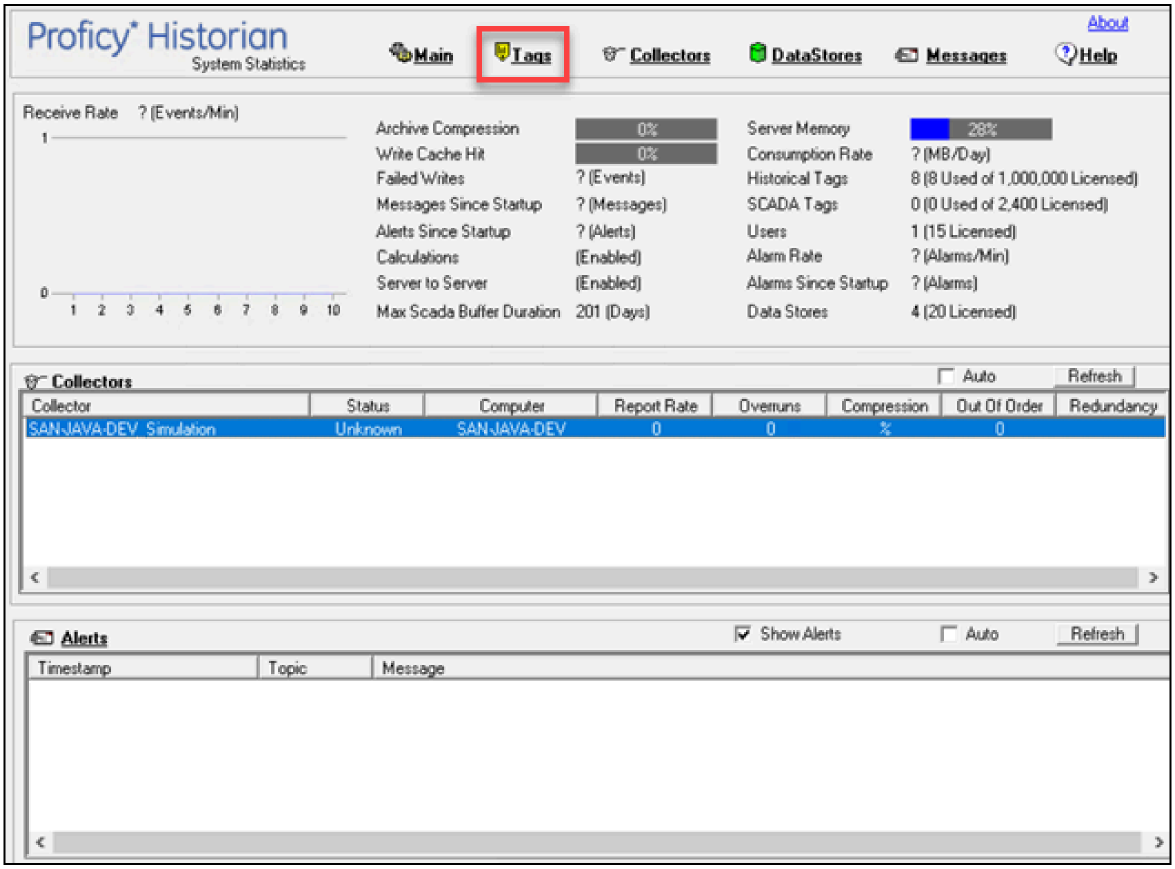
6. Select **Yes**.

   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 1285)*.

1. Access Historian Administrator *(on page 1194)*.
2. Select **Tags**.



Receive Rate ? (Events/Min)

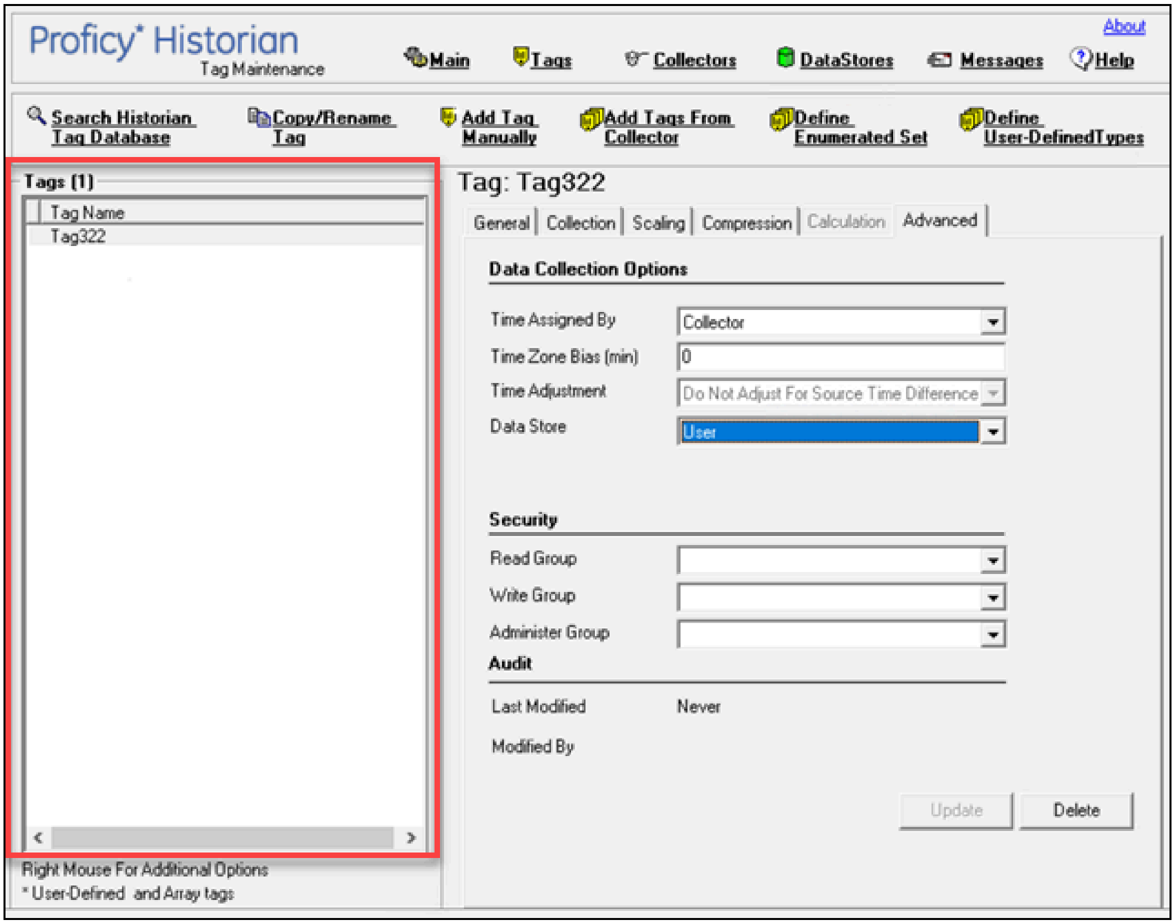
Archive Compression 0%
Write Cache Hit 0%
Failed Writes ? (Events)
Messages Since Startup ? (Messages)
Alerts Since Startup ? (Alerts)
Calculations (Enabled)
Server to Server (Enabled)
Max Scada Buffer Duration 201 (Days)

Server Memory 28%
Consumption Rate ? (MB/Day)
Historical Tags 8 (8 Used of 1,000,000 Licensed)
SCADA Tags 0 (0 Used of 2,400 Licensed)
Users 1 (15 Licensed)
Alarm Rate ? (Alarms/Min)
Alarms Since Startup ? (Alarms)
Data Stores 4 (20 Licensed)

Collector	Status	Computer	Report Rate	Overruns	Compression	Out Of Order	Redundancy
SAN-JAVA-DEV_Simulation	Unknown	SAN-JAVA-DEV	0	0	%	0	

Timestamp	Topic	Message

3. Select the tag that you want to delete.



Tag: Tag322

Data Collection Options

Time Assigned By: Collector

Time Zone Bias (min): 0

Time Adjustment: Do Not Adjust For Source Time Difference

Data Store: User

Security

Read Group:

Write Group:

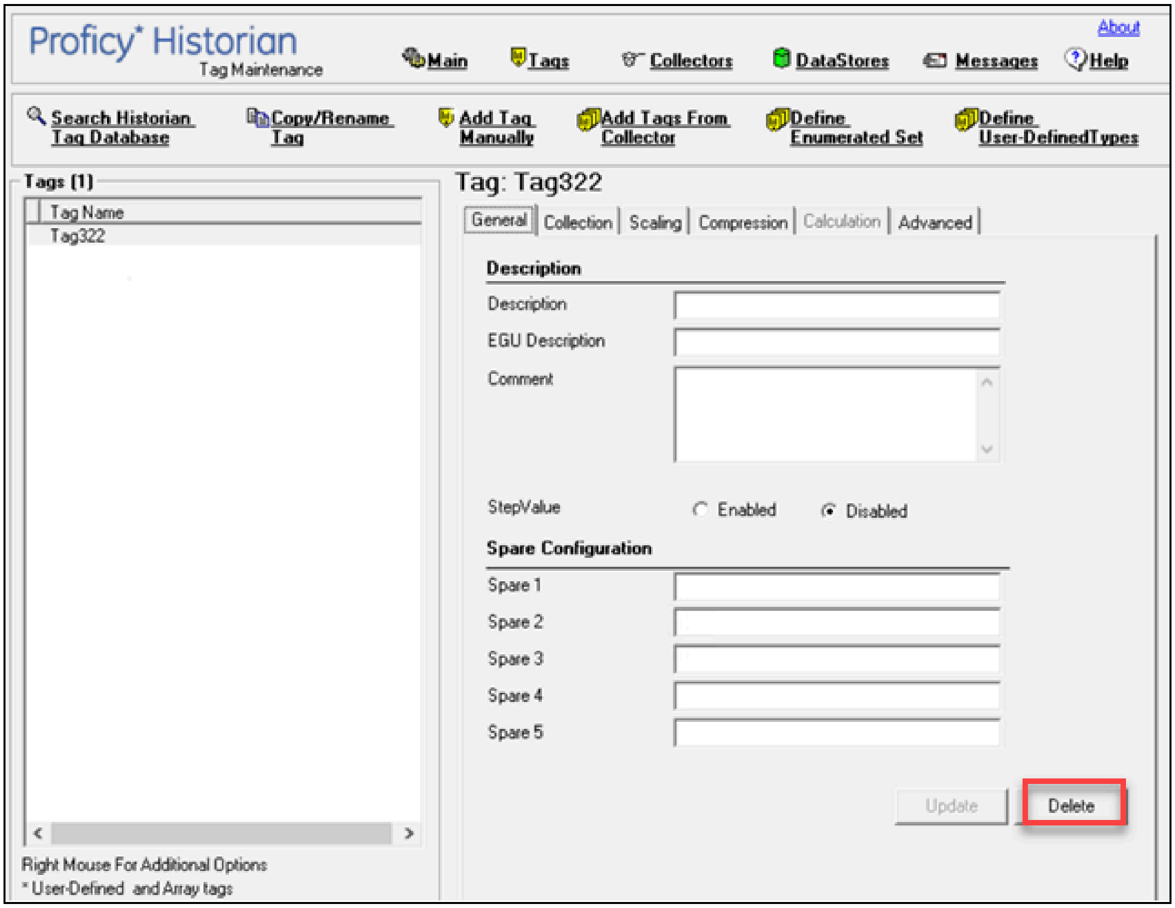
Administer Group:

Audit

Last Modified: Never

Modified By:

4. Select **Delete**.



Tag: Tag322

Description

Description

EGU Description

Comment

StepValue ○ Enabled ● Disabled

Spare Configuration

Spare 1

Spare 2

Spare 3

Spare 4

Spare 5

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.

Managing Collectors

About Historian Data Collectors

A collector collects tag data from various data sources.

How tag data is stored if using collectors of on-premises Proficy Historian (TLS encryption is not used):

1. Collectors send a request to AWS Network Load Balancer (NLB) to write tag data.
2. NLB sends the request to Data Archiver. If user authentication is needed, Data Archiver sends the request to Proficy Authentication, which verifies the user credentials stored in PostgreSQL. After authentication, NLB confirms to the collectors that data can be sent.

3. Data collected by the collector instances is sent to NLB.
4. NLB sends the data to Data Archiver directly. After authentication, Data Archiver stores the data in EFS in .iha files.

How tag data is stored if using Historian Collectors for Cloud (TLS encryption is used):

1. Collectors send a request to AWS NLB to write tag data. Since the request is encrypted, port 443 is used.
2. NLB decrypts the request and sends it to Data Archiver. If user authentication is needed, Data Archiver sends the request to Proficy Authentication, which verifies the user credentials stored in PostgreSQL. After authentication, NLB confirms to the collectors that data can be sent.
3. Data collected by the collector instances is encrypted and sent to NLB using port 443.
4. NLB decrypts the data and sends it to Data Archiver. After authentication, Data Archiver stores the data in EFS in .iha files.

How data is retrieved:

1. Clients (that is, Excel Addin, the Web Admin console, the REST Query service, or Historian Administrator) send a request to NLB to retrieve data.
2. NLB sends the request to Data Archiver, which retrieves data from EFS. If, however, user authentication is needed, Data Archiver sends the request to Proficy Authentication, which verifies the user credentials stored in PostgreSQL. After authentication, data is retrieved from EFS.

To send data using a collector, you must:

1. Install collectors *(on page)*.

   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.

> **Note:**
> The following collectors are not supported by Proficy Historian for AWS:
>
> - The File collector
> - The HAB collector

- The iFIX Alarms and Events collector
- The OPC Classic Alarms and Events collector

What does SQ mean in the cloud output and what are the sub-quality values?

The full form of SQ is Sub Quality; the values range from 1 to 13.

The following are the sub-quality values:

ihOPCNonspecific = 0

ihOPCConfigurationError=1

ihOPCNotConnected=2

ihOPCDeviceFailure=3

ihOPCSensorFailure=4

ihOPCLastKnownValue=5

ihOPCCommFailure=6

ihOPCOutOfService=7

ihScaledOutOfRange=8

ihOffLine=9

ihNoValue=10

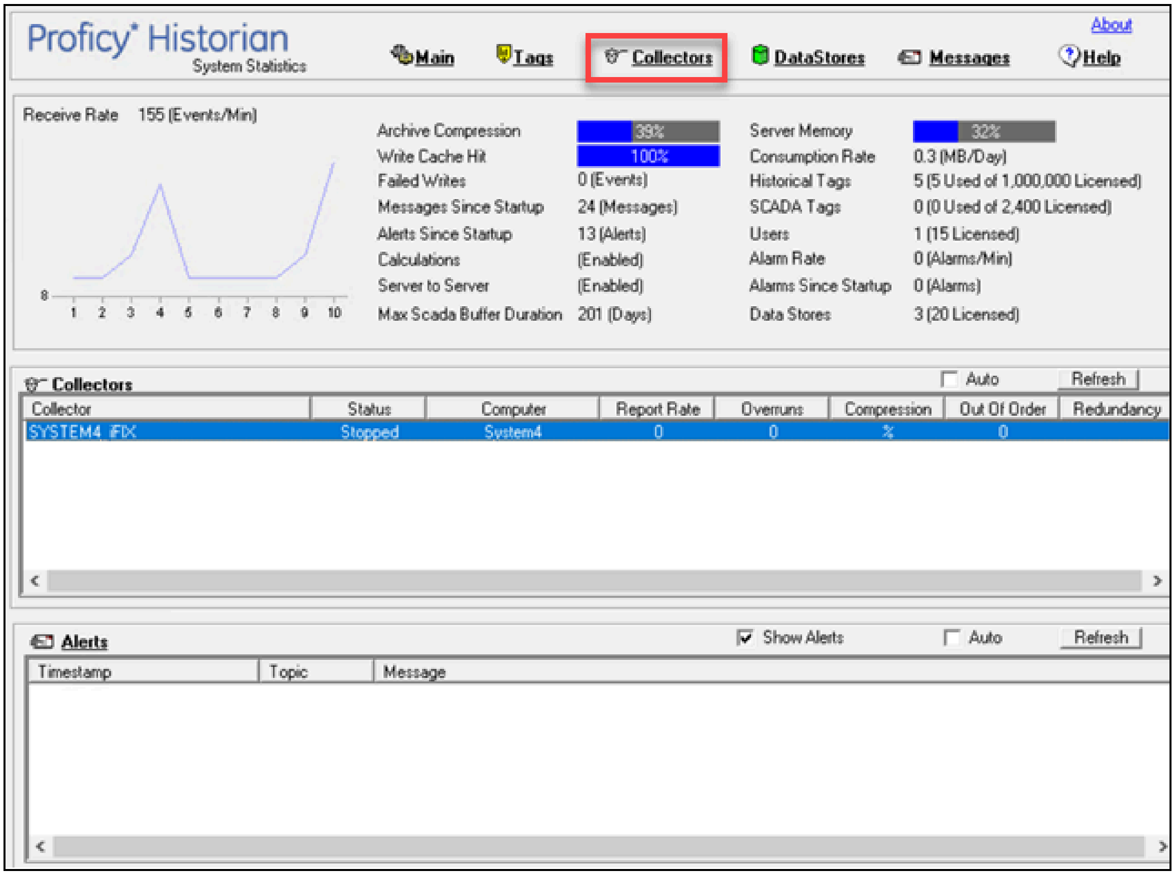
ihCalculationError=11

ihConditionCollectionHalted=12

ihCalculationTimeout=13

Access/Modify a Collector

1. Access Historian Administrator *(on page 1194)*.
2. Select **Collectors**.



Receive Rate 155 (Events/Min)

Archive Compression	39%	Server Memory	32%
Write Cache Hit	100%	Consumption Rate	0.3 (MB/Day)
Failed Writes	0 (Events)	Historical Tags	5 (5 Used of 1,000,000 Licensed)
Messages Since Startup	24 (Messages)	SCADA Tags	0 (0 Used of 2,400 Licensed)
Alerts Since Startup	13 (Alerts)	Users	1 (15 Licensed)
Calculations	(Enabled)	Alarm Rate	0 (Alarms/Min)
Server to Server	(Enabled)	Alarms Since Startup	0 (Alarms)
Max Scada Buffer Duration	201 (Days)	Data Stores	3 (20 Licensed)

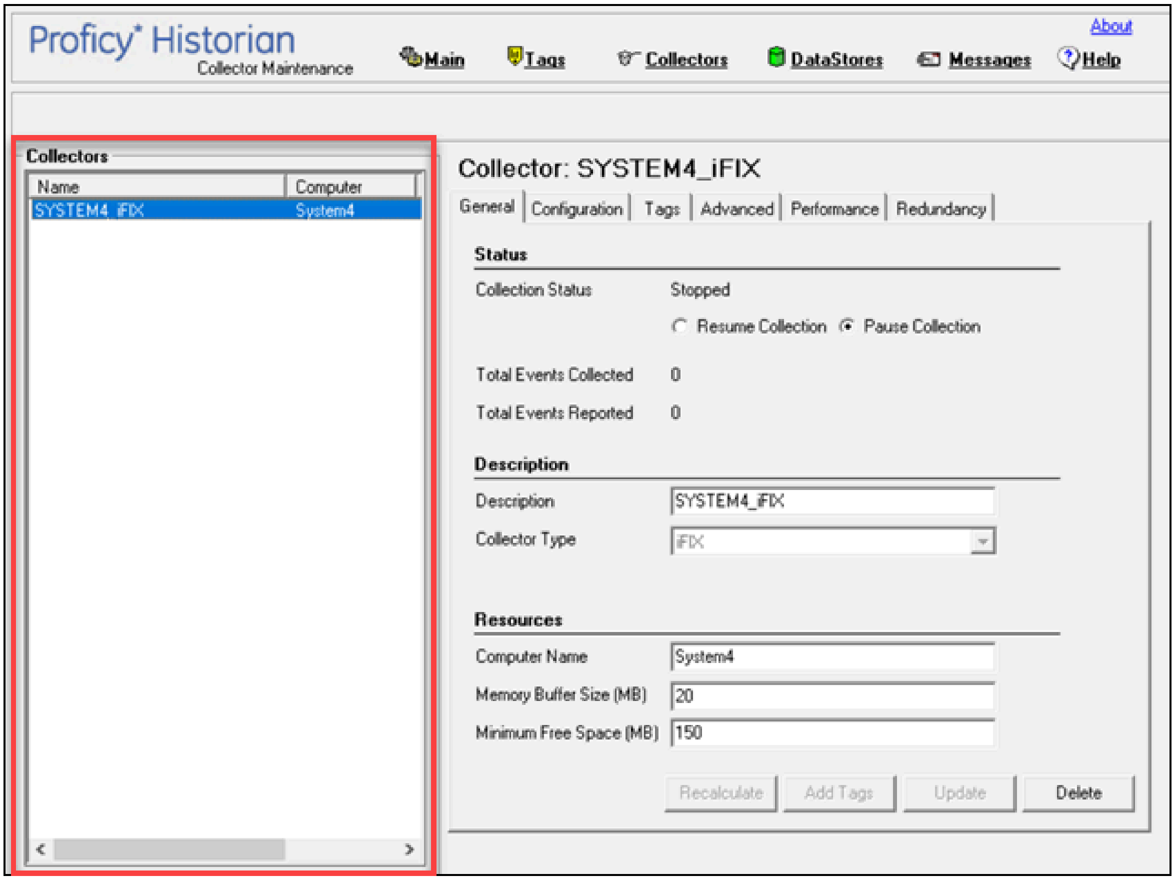
Collectors

Collector	Status	Computer	Report Rate	Overruns	Compression	Out Of Order	Redundancy
SYSTEM4_iFIX	Stopped	System4	0	0	%	0	

Timestamp	Topic	Message

A list of collectors appears.

3. Select the collector whose details you want to access/modify.



Collectors

Name	Computer
SYSTEM4_iFIX	System4

Collector: SYSTEM4_iFIX

Status

Collection Status: Stopped

○ Resume Collection ● Pause Collection

Total Events Collected: 0

Total Events Reported: 0

Description

Description: SYSTEM4_iFIX

Collector Type: iFIX

Resources

Computer Name: System4

Memory Buffer Size (MB): 20

Minimum Free Space (MB): 150

4. As needed, modify values as described in the following tables, and then select **Update**.

Table 166. The General Section

Field	Description
Collection Status Field	The current operating status of the collector. Contains one of the following values: ◦ **Running:** The collector is operating and collecting data. ◦ **Stopped:** The collector is in pause mode and not collecting data. ◦ **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. You can specify whether you want to pause or resume data collection.
Total Events Collected	Not applicable
Total Events Reported	Not applicable
Description	The name of the collector.

Field	Description
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. **Note:** 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 167. The Tags Section

Field	Description
Add Prefix to Tag	A prefix that is automatically added to all tag names when you add tags.
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. **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.
Collection Type	Indicates whether this collector is configured for polled or unsolicited data collection.

Field	Description
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 1239)* 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 previously archived sample is inserted into the archive in front of the spike value. For more information, refer to Enable Spike Logic *(on page 1302)*.

Table 168. 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

Field	Description
	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	Not applicable
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. 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. 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). 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. 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). 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 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.

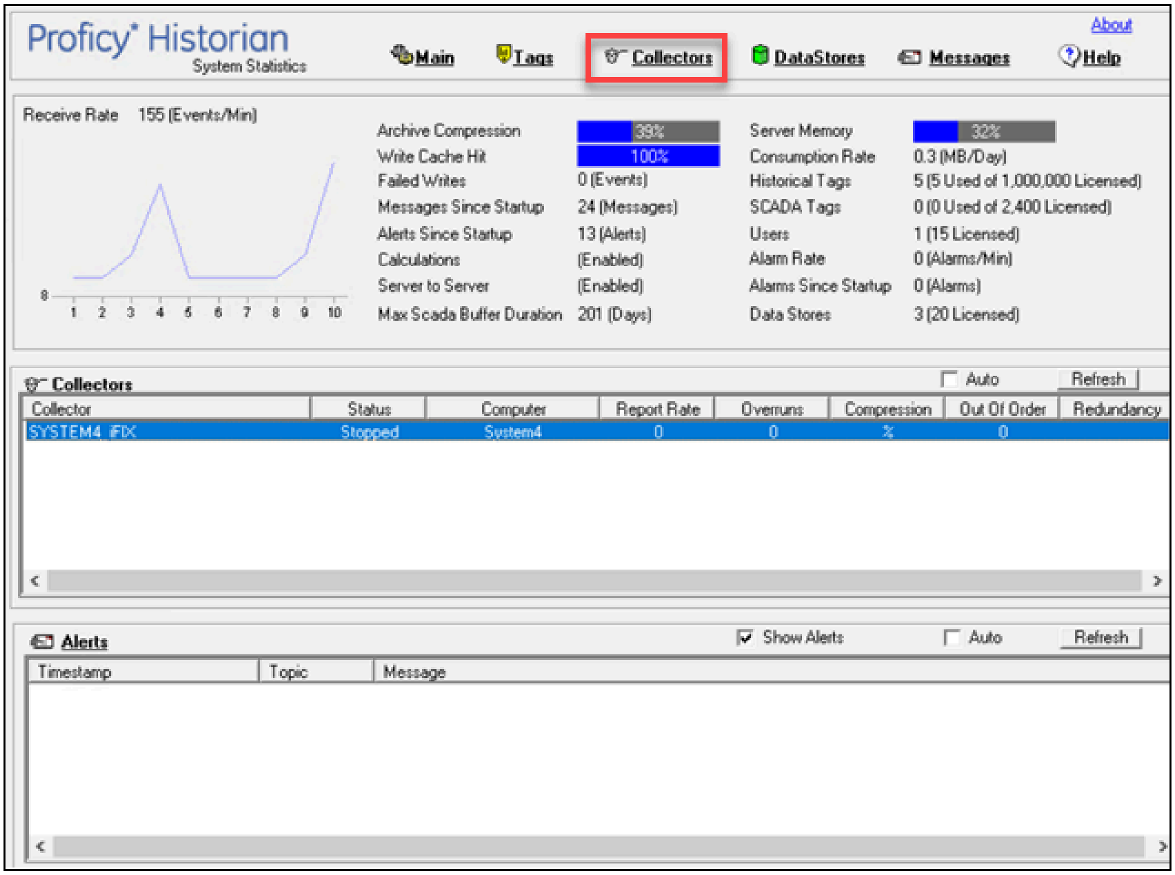
Table 169. The Performance Section

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	Not applicable

Delete a Collector

When you delete a collector, all of its tags are deleted from the Historian database.

1. Access Historian Administrator *(on page 1194)*.
2. Select **Collectors**.



Receive Rate 155 (Events/Min)

Archive Compression	39%	Server Memory	32%
Write Cache Hit	100%	Consumption Rate	0.3 (MB/Day)
Failed Writes	0 (Events)	Historical Tags	5 (5 Used of 1,000,000 Licensed)
Messages Since Startup	24 (Messages)	SCADA Tags	0 (0 Used of 2,400 Licensed)
Alerts Since Startup	13 (Alerts)	Users	1 (15 Licensed)
Calculations	(Enabled)	Alarm Rate	0 (Alarms/Min)
Server to Server	(Enabled)	Alarms Since Startup	0 (Alarms)
Max Scada Buffer Duration	201 (Days)	Data Stores	3 (20 Licensed)

Collectors

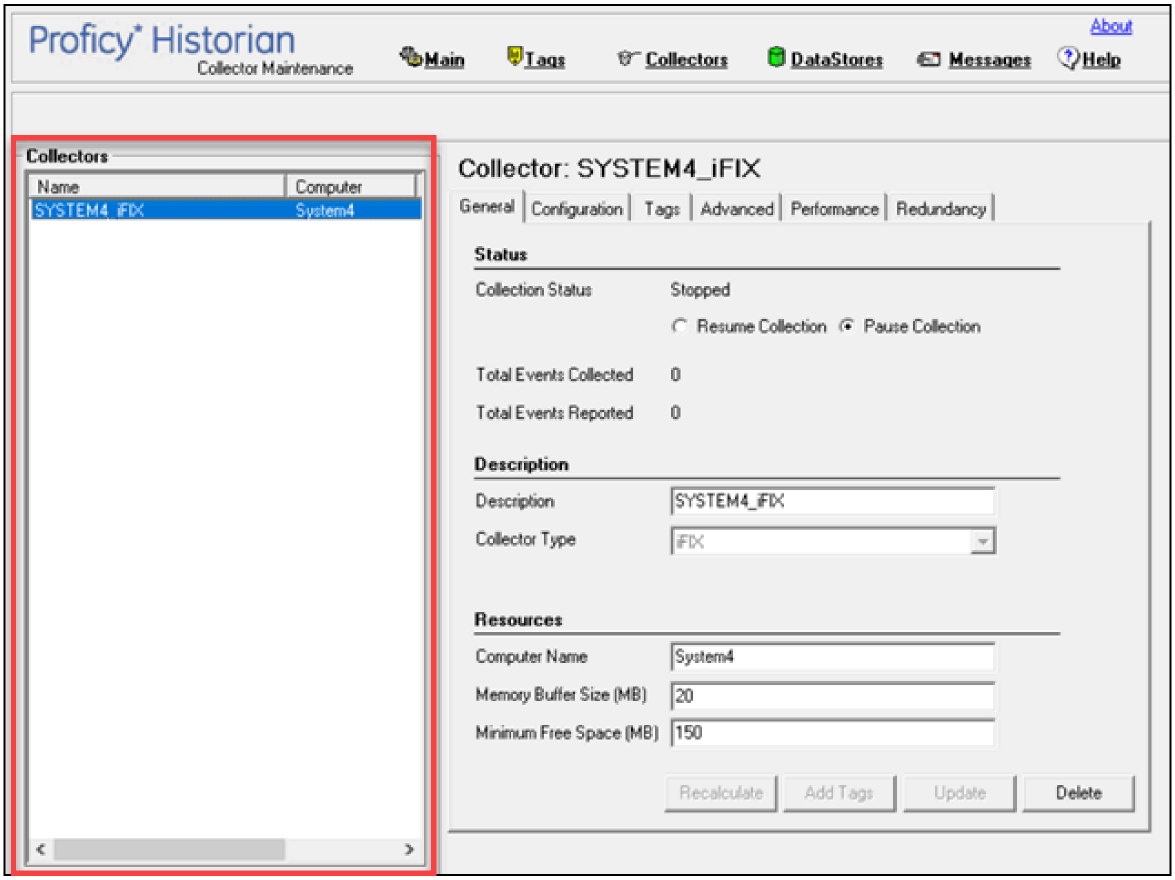
Collector	Status	Computer	Report Rate	Overruns	Compression	Out Of Order	Redundancy
SYSTEM4_iFIX	Stopped	System4	0	0	%	0	

Alerts

Timestamp	Topic	Message

A list of collectors appears.

3. Select the collector whose details you want to delete.



Collectors

Name	Computer
SYSTEM4_iFIX	System4

Collector: SYSTEM4_iFIX

Status

Collection Status: Stopped

○ Resume Collection ● Pause Collection

Total Events Collected: 0

Total Events Reported: 0

Description

Description: SYSTEM4_iFIX

Collector Type: iFIX

Resources

Computer Name: System4

Memory Buffer Size (MB): 20

Minimum Free Space (MB): 150

4. Select **Delete**.

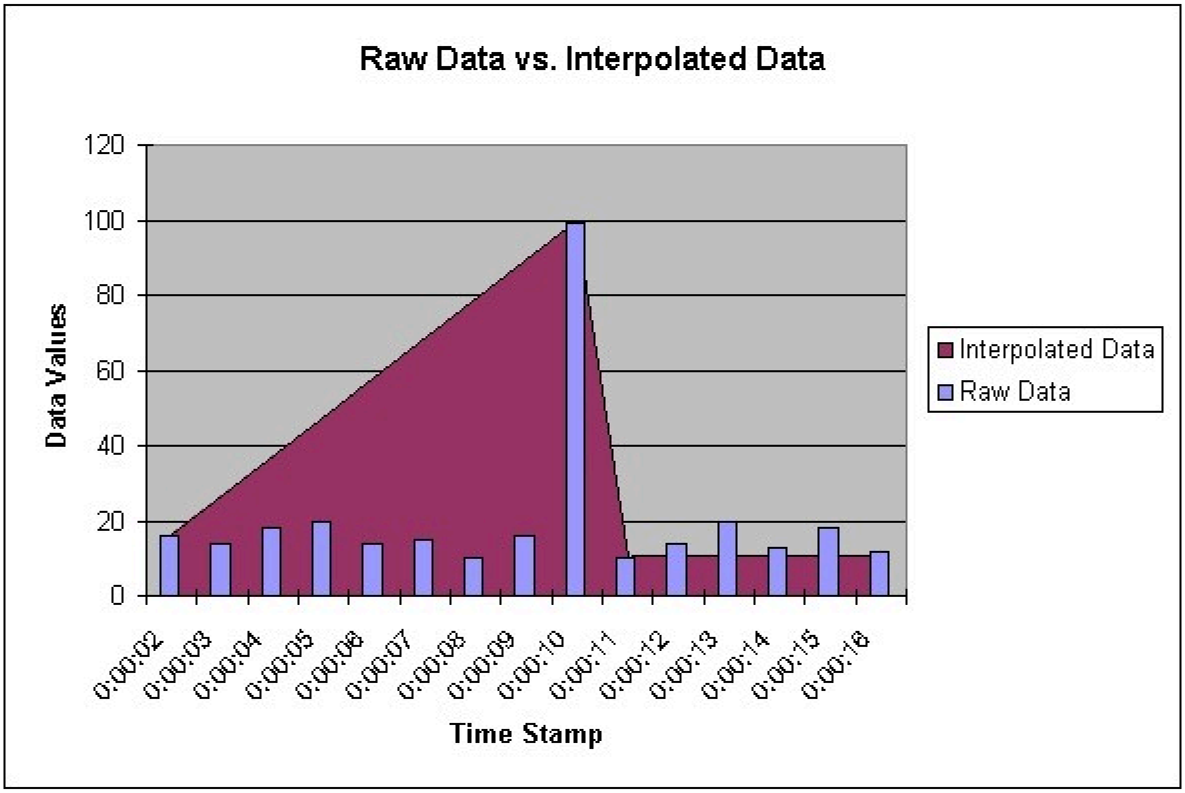
   A message appears, asking you to confirm that you want to delete the collector.
5. Select **Yes**.

   The collector is deleted.

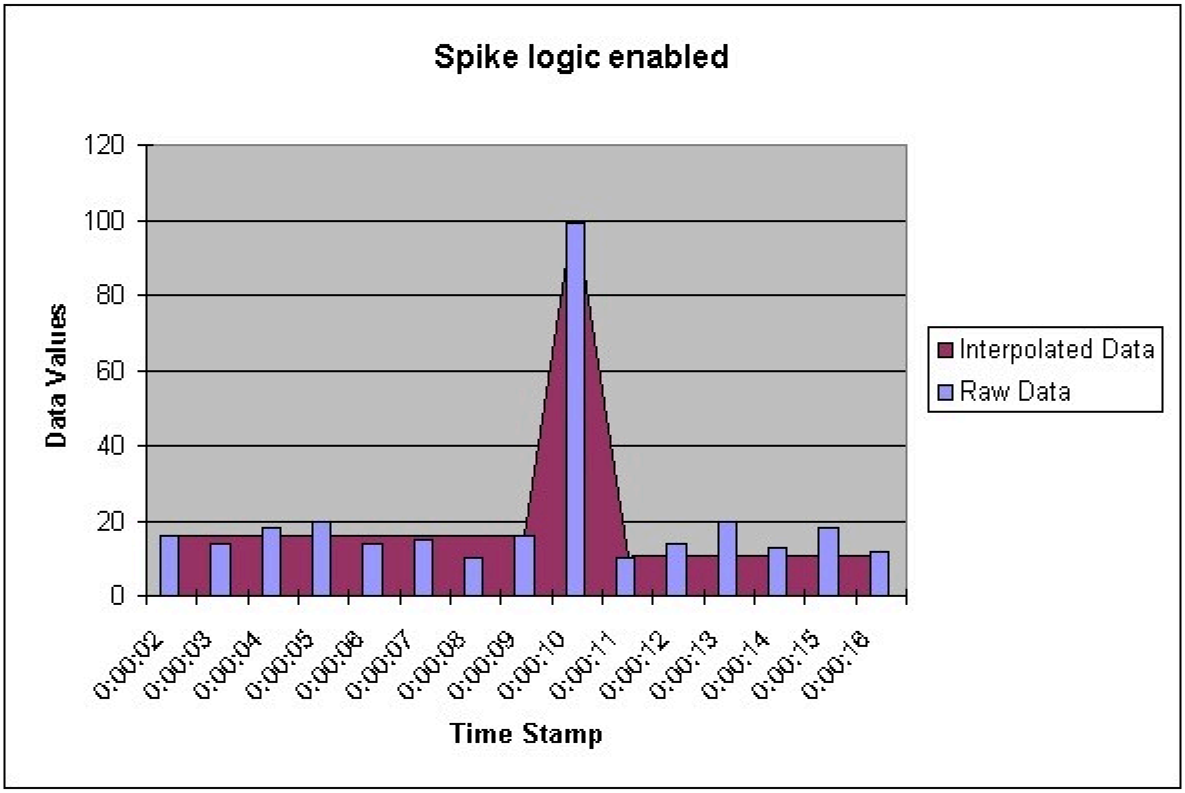
Enable Spike Logic

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.

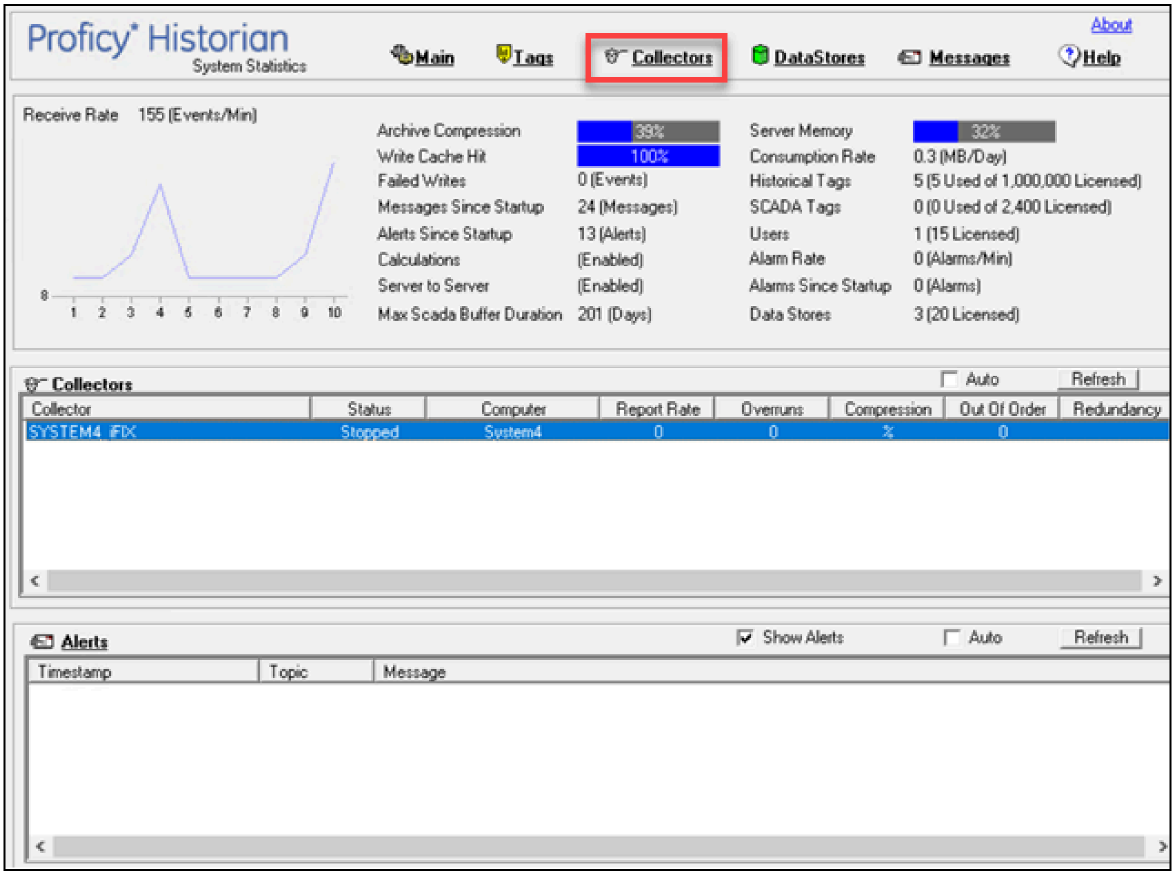
Unfortunately, in the event of a spike in data values, an unrealistic set of samples is created when the data is plotted. Instead of showing the results of compression (the same values over a series of intervals), a rising or falling slope is created in the chart. This gives the impression that values 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 1194)*.
2. Select **Collectors**.



Receive Rate 155 (Events/Min)

Archive Compression	39%	Server Memory	32%
Write Cache Hit	100%	Consumption Rate	0.3 (MB/Day)
Failed Writes	0 (Events)	Historical Tags	5 (5 Used of 1,000,000 Licensed)
Messages Since Startup	24 (Messages)	SCADA Tags	0 (0 Used of 2,400 Licensed)
Alerts Since Startup	13 (Alerts)	Users	1 (15 Licensed)
Calculations	(Enabled)	Alarm Rate	0 (Alarms/Min)
Server to Server	(Enabled)	Alarms Since Startup	0 (Alarms)
Max Scada Buffer Duration	201 (Days)	Data Stores	3 (20 Licensed)

Collectors □ Auto Refresh

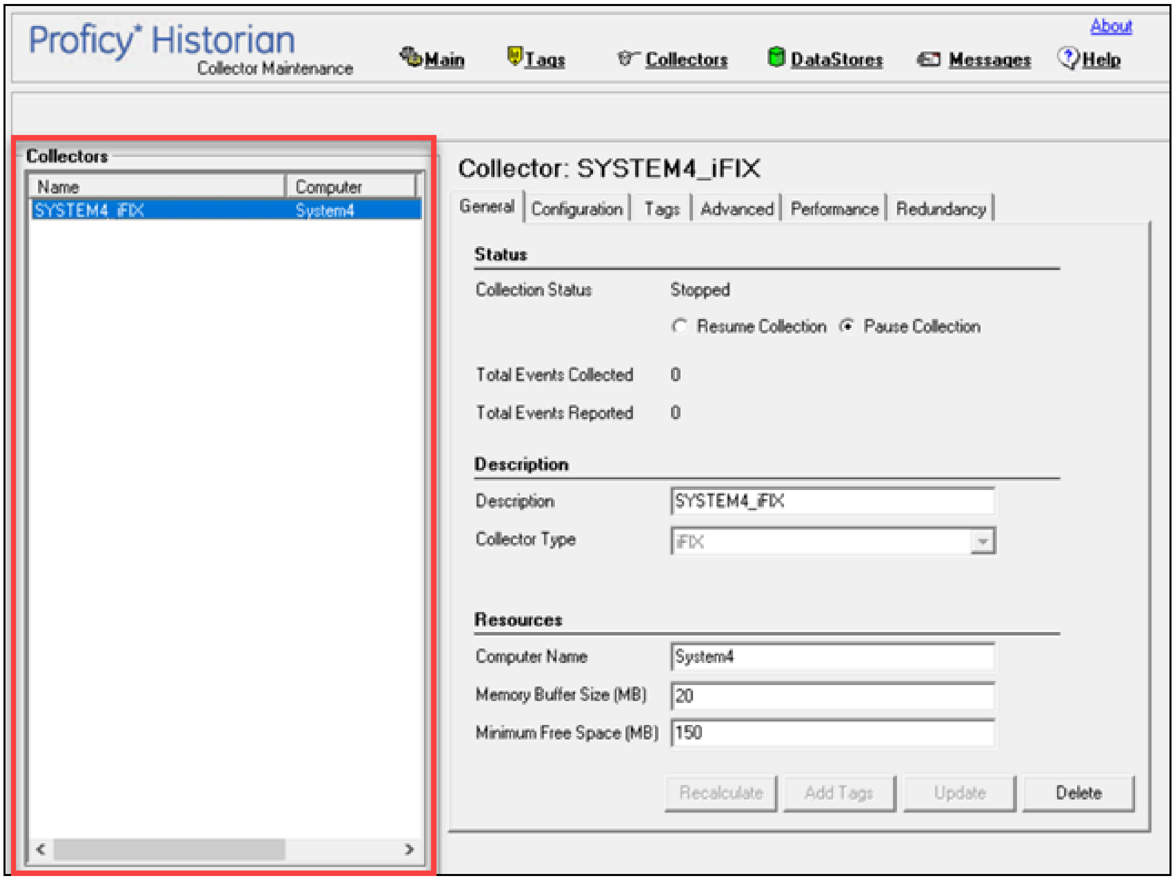
Collector	Status	Computer	Report Rate	Overruns	Compression	Out Of Order	Redundancy
SYSTEM4_iFIX	Stopped	System4	0	0	%	0	

Alerts ☑ Show Alerts □ Auto Refresh

Timestamp	Topic	Message

A list of collectors appears.

3. Select the collector to which you want to apply spike logic.



Name	Computer
SYSTEM4_iFIX	System4

Collector: SYSTEM4_iFIX

Status

Collection Status Stopped

○ Resume Collection ● Pause Collection

Total Events Collected 0

Total Events Reported 0

Description

Description SYSTEM4_iFIX

Collector Type iFIX

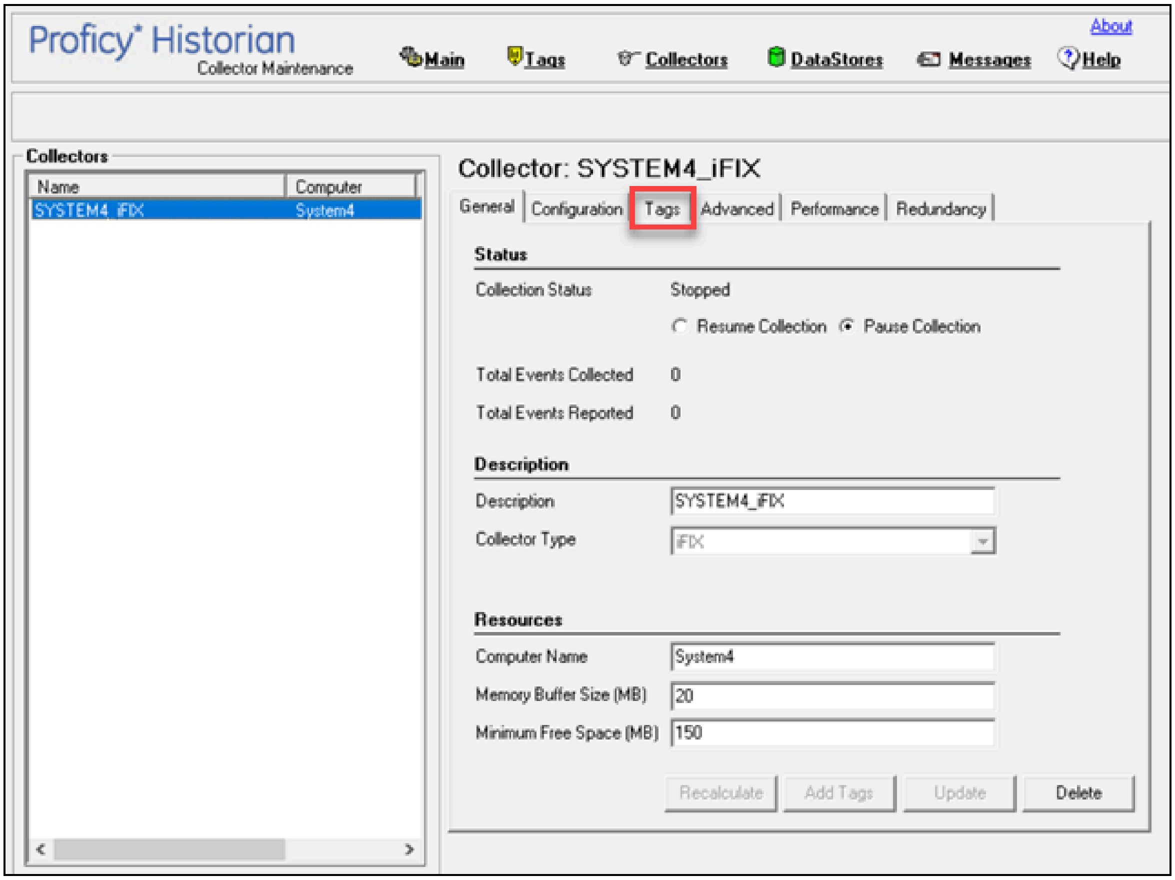
Resources

Computer Name System4

Memory Buffer Size (MB) 20

Minimum Free Space (MB) 150

4. Select **Tags**.



Collectors

Name	Computer
SYSTEM4_iFIX	System4

Collector: SYSTEM4_iFIX

Status

Collection Status: Stopped

○ Resume Collection ● Pause Collection

Total Events Collected: 0

Total Events Reported: 0

Description

Description: SYSTEM4_iFIX

Collector Type: iFIX

Resources

Computer Name: System4

Memory Buffer Size (MB): 20

Minimum Free Space (MB): 150

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.

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)
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)

Data Type	Size	Description	Valid Values
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.

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.

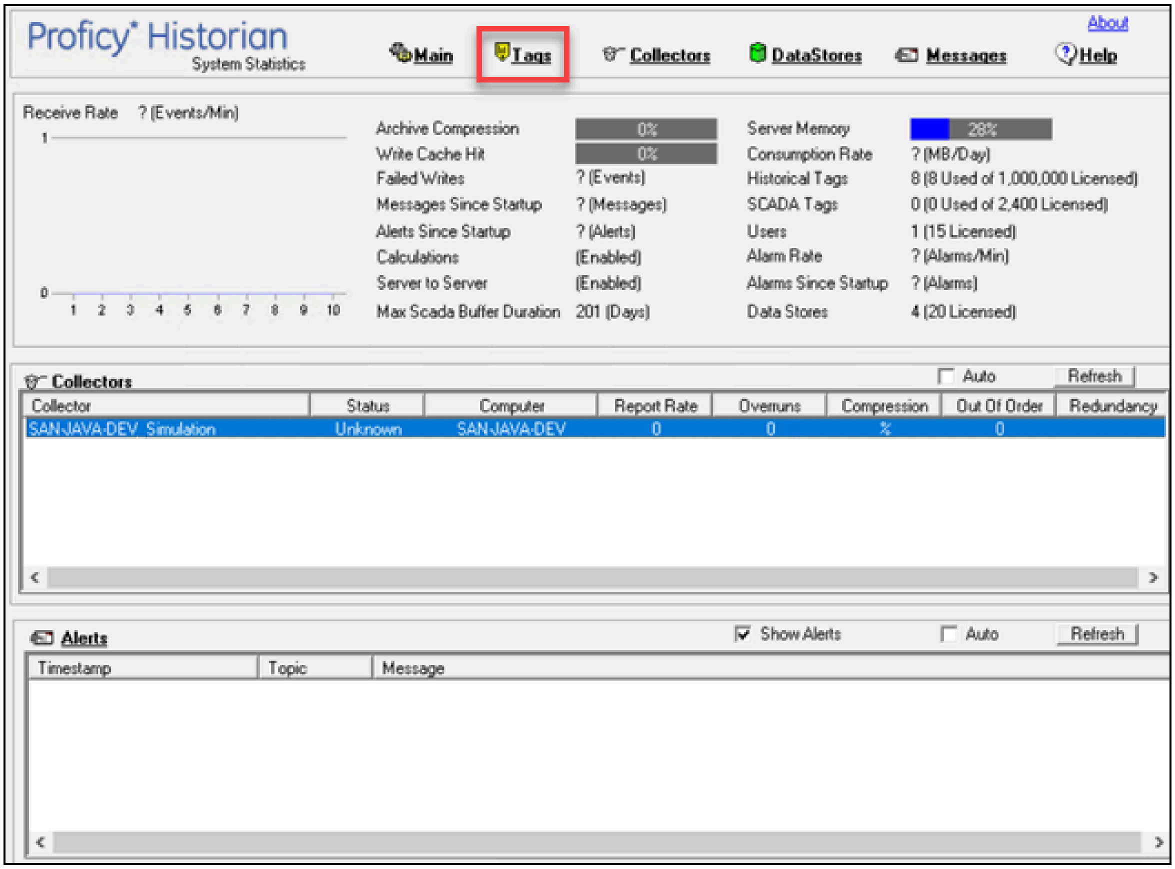
> **Important:**
> Values that are outside of the EGU range of a scaled data type tag are stored as bad, scaledoutofrange in Historian. Changing either the High or Low EGU tags does not affect existing data, but only affects the new data with new timestamps. You cannot correct values for scaled data 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 1194)*.
2. Select **Tags**.



Receive Rate ? (Events/Min)

Archive Compression	0%	Server Memory	28%
Write Cache Hit	0%	Consumption Rate	? (MB/Day)
Failed Writes	? (Events)	Historical Tags	8 (8 Used of 1,000,000 Licensed)
Messages Since Startup	? (Messages)	SCADA Tags	0 (0 Used of 2,400 Licensed)
Alerts Since Startup	? (Alerts)	Users	1 (15 Licensed)
Calculations	(Enabled)	Alarm Rate	? (Alarms/Min)
Server to Server	(Enabled)	Alarms Since Startup	? (Alarms)
Max Scada Buffer Duration	201 (Days)	Data Stores	4 (20 Licensed)

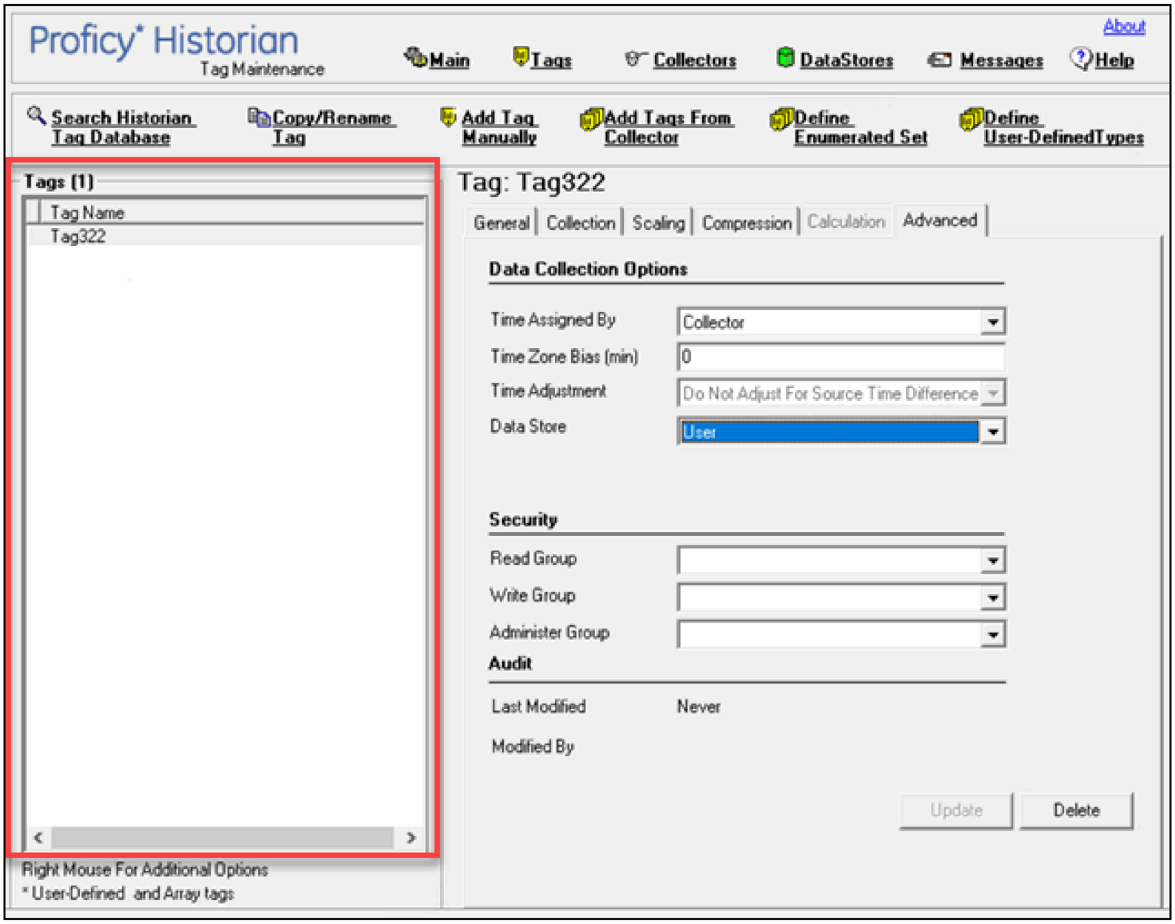
Collectors □ Auto Refresh

Collector	Status	Computer	Report Rate	Overruns	Compression	Out Of Order	Redundancy
SAN-JAVA-DEV_Simulation	Unknown	SAN-JAVA-DEV	0	0	%	0	

Alerts ☑ Show Alerts □ Auto Refresh

Timestamp	Topic	Message

3. Select the tag for which you want to set a fixed string data type.



Data Collection Options

Time Assigned By: Collector

Time Zone Bias (min): 0

Time Adjustment: Do Not Adjust For Source Time Difference

Data Store: User

Security

Read Group:

Write Group:

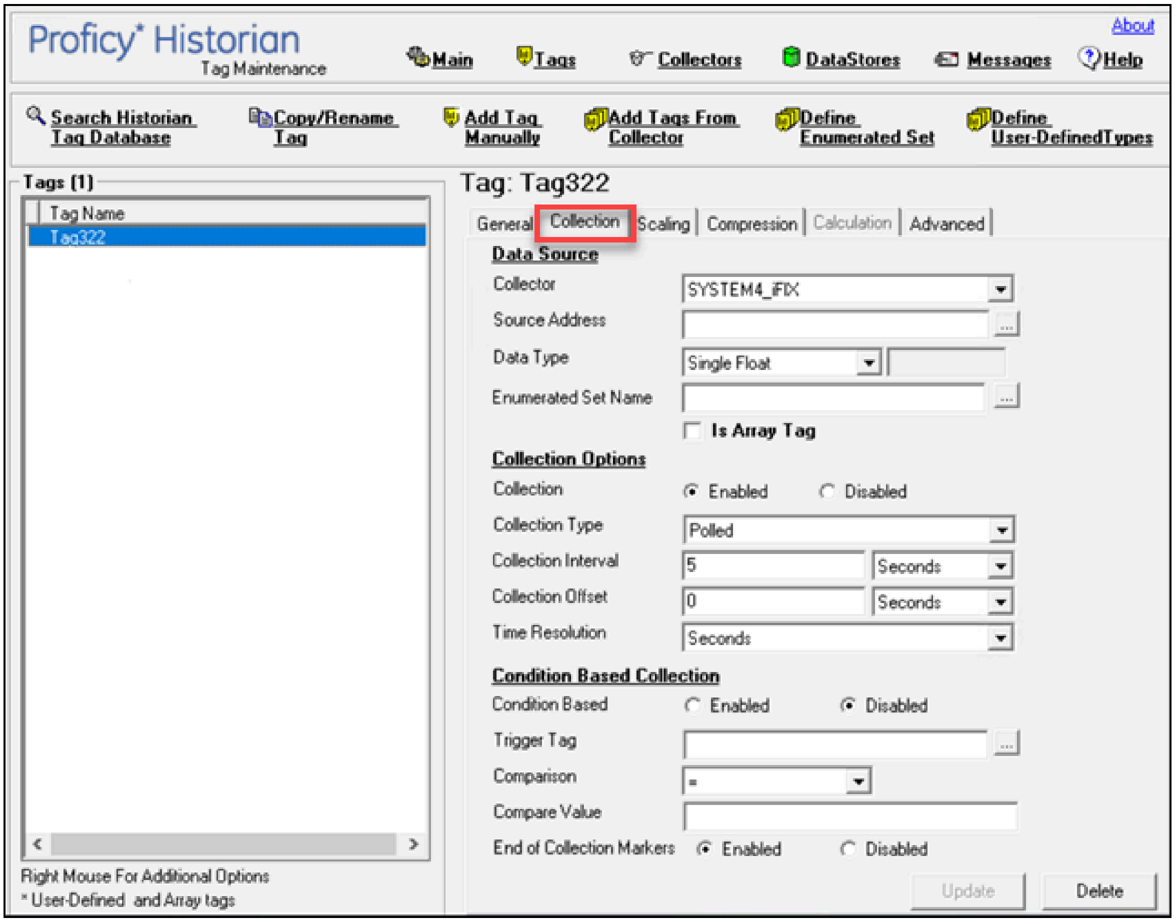
Administer Group:

Audit

Last Modified: Never

Modified By:

4. Select **Collection**.



Tag: Tag322

Data Source

Collector: SYSTEM4_iFIX
Source Address:
Data Type: Single Float
Enumerated Set Name:
□ Is Array Tag

Collection Options

Collection: ● Enabled ○ Disabled
Collection Type: Polled
Collection Interval: 5 Seconds
Collection Offset: 0 Seconds
Time Resolution: Seconds

Condition Based Collection

Condition Based: ○ Enabled ● Disabled
Trigger Tag:
Comparison: =
Compare Value:
End of Collection Markers: ● Enabled ○ Disabled

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.

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.

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 170. System Statistics Performance Indicators

Field	Recommended Action
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. 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	Not applicable

On a regular schedule, examine and analyze the performance indicators displayed in the **Performance** section.

Table 171. Collector Performance Indicators

Field	Recommended Actions
Avg. Event Rate Chart	Not applicable
Compression Chart	Is compression effectiveness acceptable? 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.

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.
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, Simulation 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 logging off 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.

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

The Non-Web Administrator offers a configurable timeout. This configurable timeout determines how long the Non-Web 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 with Registry Keys

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.

> **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.

Configuring the throttle is easy, but it 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 with the NumIntervalsFlush registry key

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 172. Historian Error Codes and Messages

Number	Description
-32	Operation not permitted
-31	The requested data store was not found
-29	A supplied argument is outside the valid range
-28	A supplied argument is NULL
-27	A supplied argument is invalid
-25	Attempted data delete outside allowed modification interval
-24	Data Retrieval Count Exceeded
-23	Invalid Server Version
-21	Calculation Circular Reference
-20	Not Licensed
-19	Duplicate Interface
-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

Table 172. Historian Error Codes and Messages (continued)

Number	Description
-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
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

Table 172. Historian Error Codes and Messages (continued)

Number	Description
16	Interface Modified
17	Interface Deleted
18	Archive Added
19	Archive Add Failure Time Overlap
20	Archive Deleted
21	Archive Overwritten
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
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

Table 172. Historian Error Codes and Messages (continued)

Number	Description
44	Option Modified
45	Write Processing Stopped
46	Write Processing Resumed
47	Interface Status Unknown
48	Archive Closed
49	Interface Stopped
50	Interface Started

Scheduled Software Performance Impact

Running continuous disk scan software applications such as anti-virus scans, 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.

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.

Error Message	Suggested Action
	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) unsuccessfully connected at 07/17/2001 07:58:18.980 PM.	Bad password for user account. Enter correct password.
[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.

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.

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.

Chapter 14. 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:

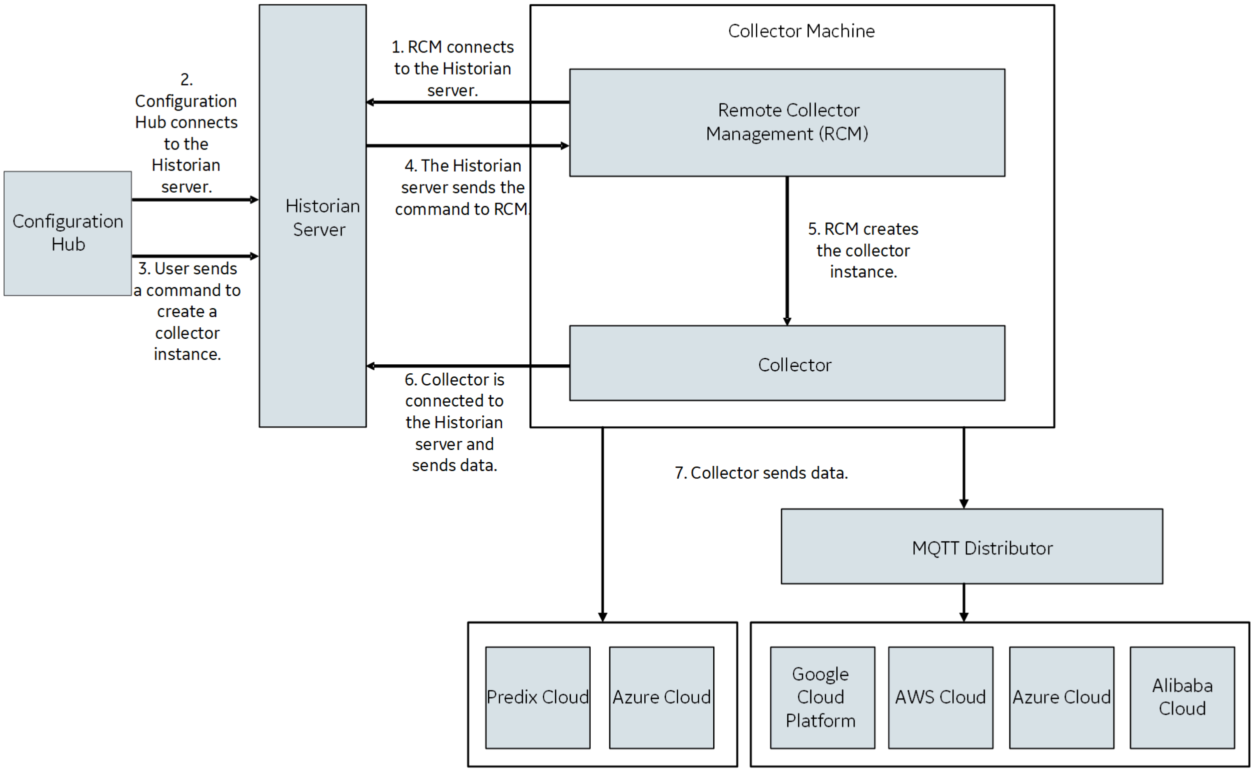
- Accessing a collector machine physically to manage the collector is no longer required.
- Security is enabled. That is, only members of the iH Security Admins, iH Tag Admins, and the iH Collector Admins security groups can manage the collectors remotely.
- Works with the older versions of collectors as well (V5.5 and later).

Features

- Add *(on page 1335)*, modify *(on page 1337)*, or delete *(on page 1355)* a collector instance.
- Start *(on page 725)*, stop *(on page 727)*, or restart *(on page 728)* a collector.
- Pause *(on page 729)* or resume *(on page 730)* the data collection of a collector.
- Delete *(on page 731)* or move *(on page 732)* the buffer files of a collector.
- Change the destination server of a collector *(on page 734)*.

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.


Collector Machine
1. RCM connects to the Historian server.
2. Configuration Hub connects to the Historian server.
Remote Collector Management (RCM)
4. The Historian server sends the command to RCM.
Configuration Hub
Historian Server
3. User sends a command to create a collector instance.
5. RCM creates the collector instance.
Collector
6. Collector is connected to the Historian server and sends data.
7. Collector sends data.
MQTT Distributor
Predix Cloud
Azure Cloud
Google Cloud Platform
AWS Cloud
Azure Cloud
Alibaba Cloud

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 1331)*.

1. Run the `InstallLauncher.exe` file.
2. Select **Install Remote Management Agents**.

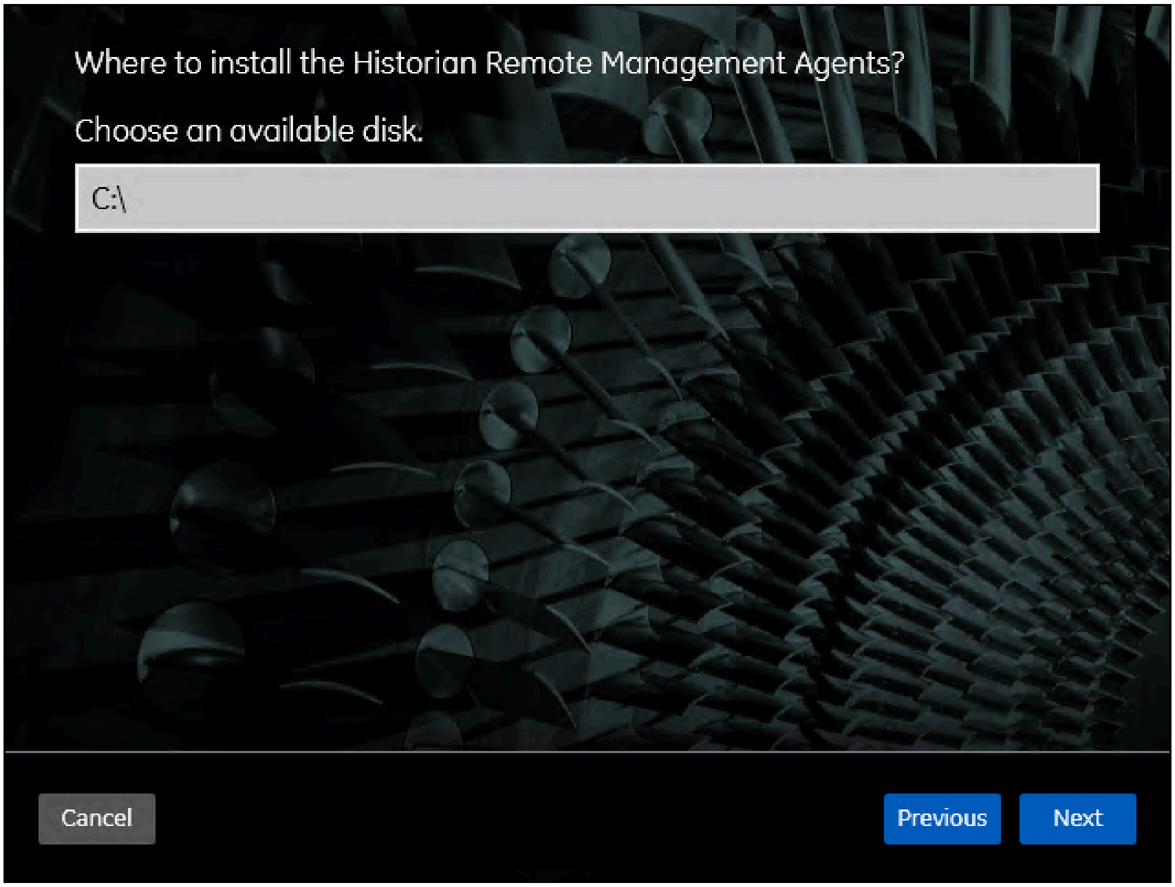
The welcome page appears.

3. Select **Next**.

   The license agreement appears.

4. Select the **Accept** check box, and then select **Next**.

   The default installation drive appears.



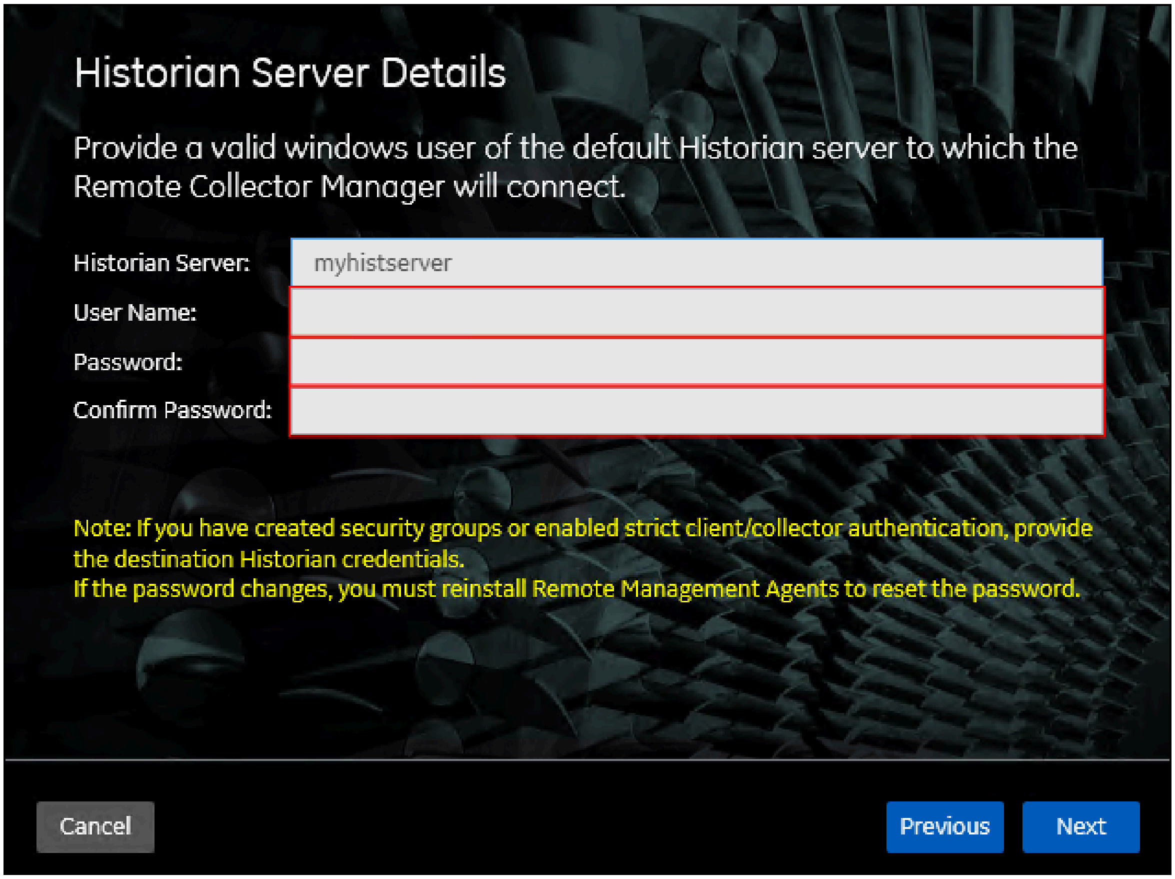
Where to install the Historian Remote Management Agents?

Choose an available disk.

C:\

5. If needed, modify the installation drive, and then select **Next**.

   The destination Historian server page appears.



Historian Server Details

Provide a valid windows user of the default Historian server to which the Remote Collector Manager will connect.

Historian Server: myhistserver

User Name:

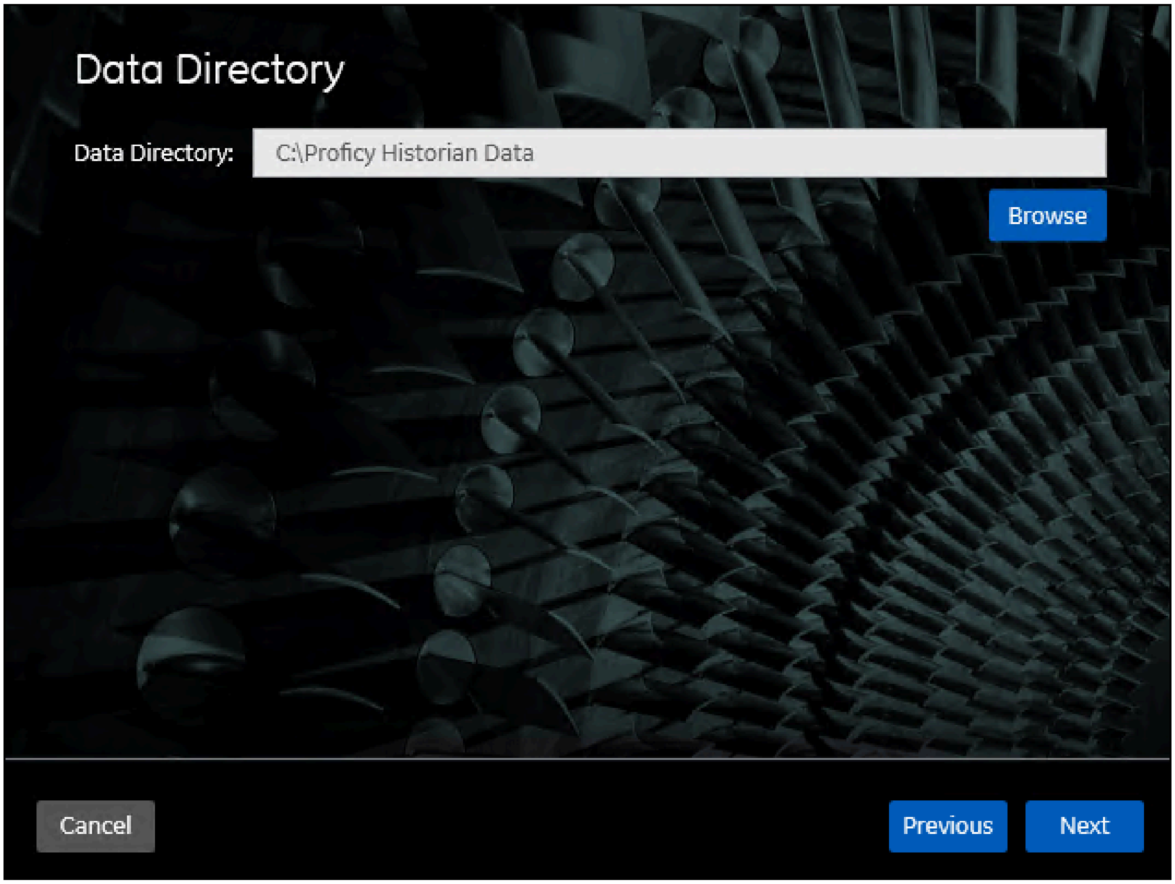
Password:

Confirm Password:

Note: If you have created security groups or enabled strict client/collector authentication, provide the destination Historian credentials.
If the password changes, you must reinstall Remote Management Agents to reset the password.

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

   The **Data Directory** page appears.



Data Directory: C:\Proficy Historian Data

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 1358)*.

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 1328)*.

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 1358)*.

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 142)*.

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 717)*.

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 1334)*, 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 1339)*. You can also specify values for the general parameters *(on page 1352)*.

Add *(on page 1335)* or modify *(on page 1337)* the collector instance using the sample JSON file that you have created.

Add a Collector Instance

- Install collectors *(on page 90)*.
- 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 300)* 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 556)*. If you want to add an offline collector instance, refer to Add an Offline Collector Instance *(on page 1357)*.

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 1334)*.

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 1339)*.

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 1334)*.

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 1339)*.

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 741)*.
- 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)*.

Modify a Collector Instance

Stop the collector *(on page)* 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 722)*.

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 1334)*.

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 1339)*.

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 1334)*.

    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 1339)*.

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 173. Destination: Historian

Parameter	Description
CollectorDestination	The type of the configuration to specify the tags whose data you want to collect. • If you want to use Historian Administrator to specify the tags for data collection, enter `Historian`. For information, refer to About Tags *(on page 741)*. • 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)*.
winUserName	The username to connect to the machine on which Historian Administrator is installed. A value is required only if: • You want to use Historian Administrator to specify the tags for data collection. • Historian security groups are used. • 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 173. Destination: Historian (continued)

Parameter	Description
	- You want to use Historian Administrator to specify the tags for data collection. - Historian security groups are used. - 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: - < (less than) - > (greater than) - : (colon) - " (double quote) - / (forward slash) - \ (backslash) - | (vertical bar or pipe) - ? (question mark) - * (asterisk)
InterfaceDescription	The description of the collector instance.
InterfaceSubType	The subtype of the collector. For information, refer to Collector Type and Subtype *(on page)*.
Type	The type of the collector. For information, refer to Collector Type and Subtype *(on page)*. 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 173. Destination: Historian (continued)

Parameter	Description
	• 1: Creates the collector instance with the credentials of the local user. • 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 174. 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 174. 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 `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.
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 174. 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: • < (less than) • > (greater than) • : (colon) • " (double quote) • / (forward slash) • \ (backslash) • | (vertical bar or pipe) • ? (question mark) • * (asterisk)
InterfaceDescription	The description of the collector instance.
InterfaceSubType	The subtype of the collector. For information, refer to Collector Type and Subtype *(on page)*.
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 174. Destination: Predix TimeSeries (continued)

Parameter	Description
Type	The type of the collector. For information, refer to Collector Type and Subtype *(on page)*. 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: • You want to use Historian Administrator to specify the tags for data collection. • Historian security groups are used. • 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: • You want to use Historian Administrator to specify the tags for data collection. • Historian security groups are used. • 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: • 1: Creates the collector instance with the credentials of the local user. • 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-Proficy
 Authentication.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 175. 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: • < (less than) • > (greater than) • : (colon) • " (double quote) • / (forward slash) • \ (backslash) • | (vertical bar or pipe) • ? (question mark) • * (asterisk)
InterfaceDescription	The description of the collector instance.
InterfaceSubType	The subtype of the collector. For information, refer to Collector Type and Subtype *(on page).*

Table 175. 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)*. A value is required.
winUserName	The username to connect to the machine on which Historian Administrator is installed. A value is required only if: • You want to use Historian Administrator to specify the tags for data collection. • Historian security groups are used. • 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: • You want to use Historian Administrator to specify the tags for data collection. • Historian security groups are used. • 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 175. Destination: MQTT (continued)

Parameter	Description
	• 1: Creates the collector instance with the credentials of the local user. • 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 176. 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: • < (less than) • > (greater than)

Table 176. Destination: Azure IoT Hub (continued)

Parameter	Description
	• : (colon) • " (double quote) • / (forward slash) • \ (backslash) • | (vertical bar or pipe) • ? (question mark) • * (asterisk)
InterfaceDescription	The description of the interface.
InterfaceSubType	The subtype of the collector. For information, refer to Collector Type and Subtype *(on page)*.
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: • HTTP • MQTT • AMQP • MQTT_OVER_WEBSOCKETS • AMQP_OVER_WEBSOCKETS

Table 176. Destination: Azure IoT Hub (continued)

Parameter	Description
	For information on which protocol to use, refer to Protocols and Port Numbers *(on page 184)*.
Type	The type of the collector. For information, refer to Collector Type and Subtype *(on page)*. A value is required.
winUserName	The username to connect to the machine on which Historian Administrator is installed. A value is required only if: • You want to use Historian Administrator to specify the tags for data collection. • Historian security groups are used. • 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: • You want to use Historian Administrator to specify the tags for data collection. • Historian security groups are used. • 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: • 1: Creates the collector instance with the credentials of the local user. • 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. • 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 • General3 - optional. Used for thread count. Default value: 5 • General4 - optional. Used for the Cygnet debug mode. Default value: 0 • 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 • 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. General2 - required. Used for the source topic. General3 - required. Used for the source port.
The ODBC collector	• General1 - required. Used for the ODBC server. • General2 - required. Used for the ODBC username. • General3 - required. Used for the ODBC password. • General4 - optional. Used for recovery time (hr). Default value: 0. • 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. • General2 - optional. Used for recovery time (hr). Default value: 24
The OPC UA DA collector	• General1 - required. Used for the OPC UA server URI. • General2 - optional. Used for secured connectivity. Default value: false • General3 - optional. Used to enable user security. Default value: false • General4 - optional. Used for username when security is enabled. General5 - optional. Used for password when security is enabled.
The OSI PI collector	• General1 - required. Used for the OSI PI server name. • General2 - optional. Used for the OSI PI username. Default value: piadmin • General3 - optional. Used for the OSI PI password. • General4 - optional. Used for max recovery time (hr). Default value: 4 • 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. General2 - optional. Used for the OSI PI username. Default value: piadmin • General3 - optional. Used for the OSI PI password.

Collector Type	Applicable General Parameters
	• General4 - optional. Used for max recovery time (hr). Default value: 4 • 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 • 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 • General2 - optional. Used for max recovery time (hr). Default value: 4 • General3 - required. Used for the source server name • General4 - optional. Used for message replication (0 or 1) and alarm replication (0 or 1). Enter 0 or 1 only for alarm replication. • General5 - optional. Used for prefix to messages.
The Windows Performance collector	None
The Wonderware collector	• General1 - required. Used for the Wonderware server. • General2 - required. Used for the Wonderware username. • General3 - required. Used for the Wonderware password. • General4 - optional. Used for the recovery time (hr). Default value: 0 • General5 - optional. Used for Throttle (milliseconds). Default value: 100

Delete a Collector Instance

Stop the collector *(on page)* 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 739)*. If you want to delete an offline collector, refer to Delete an Offline Collector Instance *(on page 1357)*.

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 1334)*.
   For information on the parameters, refer to Collector Instance Parameters *(on page 1339)*.

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 1328)*.

   > **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 725)*, stop *(on page 727)*, or restart *(on page 728)* a collector.
- Pause *(on page 729)* or resume *(on page 730)* the data collection of a collector.
- Delete *(on page 731)* or move *(on page 732)* the buffer files of a collector.
- Change the destination server of a collector *(on page 734)*.

> **Note:**
> You can also perform these tasks using REST APIs *(on page)*.

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 15. 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.

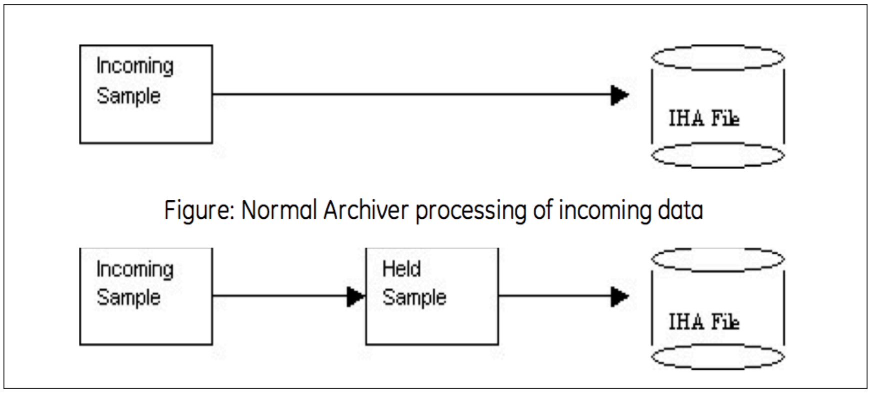


Figure: Normal Archiver processing of incoming data

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 1372)* or Command Line *(on page 1372)*.

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 1376)*.

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 97)*.
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 1373)* or Command Line *(on page 1372)*.

    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 1376)*.

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)

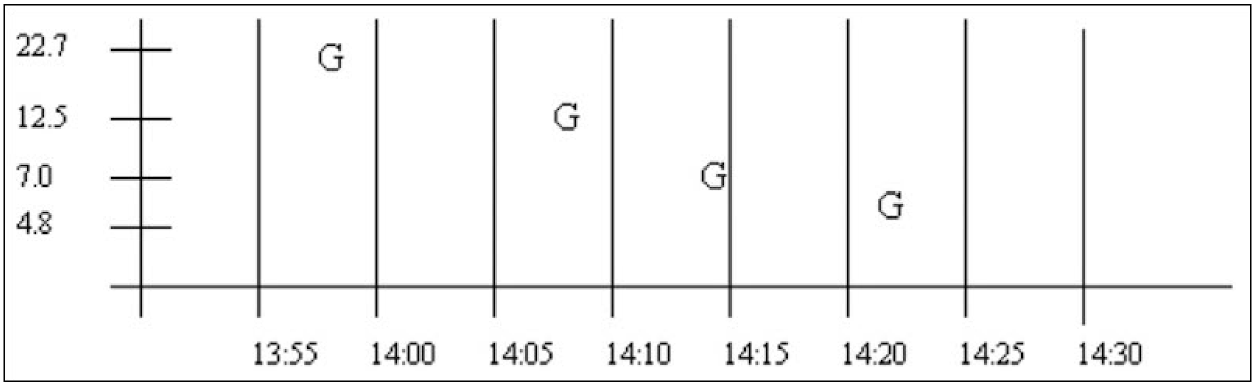
`Interpolated value` = 12.5 + ((-5.5/6)*2) = 10.67

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
```

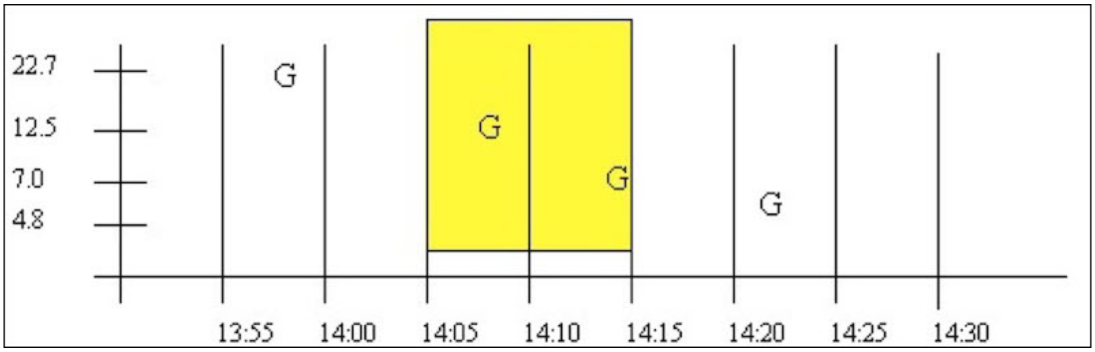
Timestamp	Value	Quality
29-Mar-2002 13:55:00.000	0.00	0.00
29-Mar-2002 14:00:00.000	21.57	100.00
29-Mar-2002 14:05:00.000	15.90	100.00
29-Mar-2002 14:10:00.000	10.67	100.00
29-Mar-2002 14:15:00.000	6.73	100.00
29-Mar-2002 14:20:00.000	5.35	100.00
29-Mar-2002 14:25:00.000	4.80	100.00
29-Mar-2002 14:30:00.000	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.

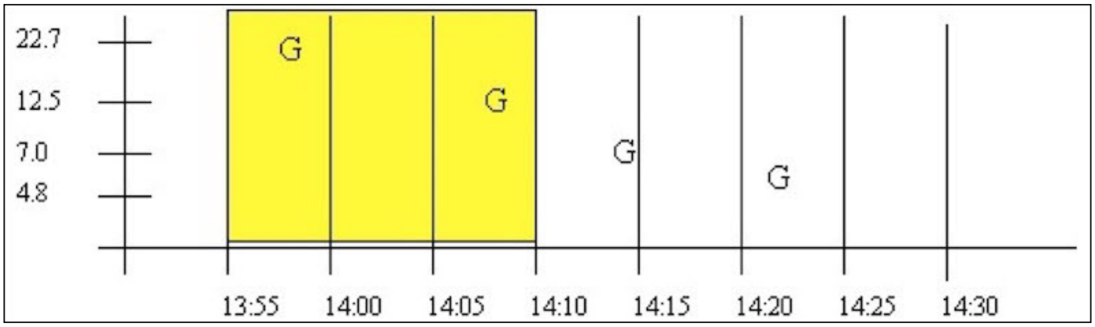
Case 1: Good Data Samples Before and After the Interval Timestamp

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.



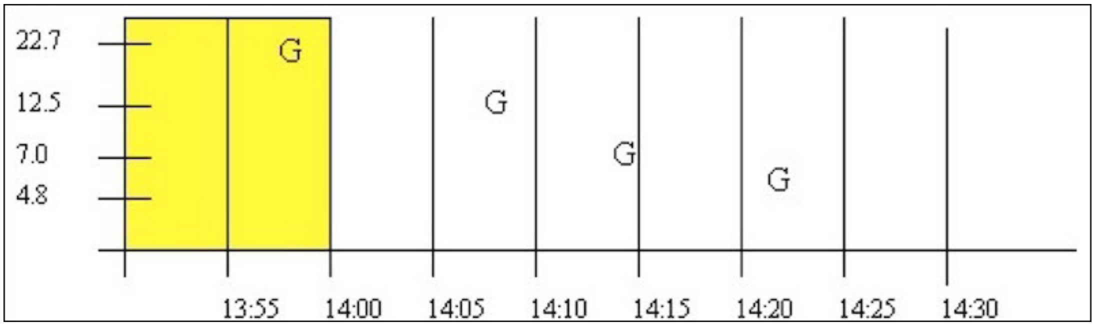
Case 1a: Good Data Samples between the Interval Timestamp and the Start and End Time

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:



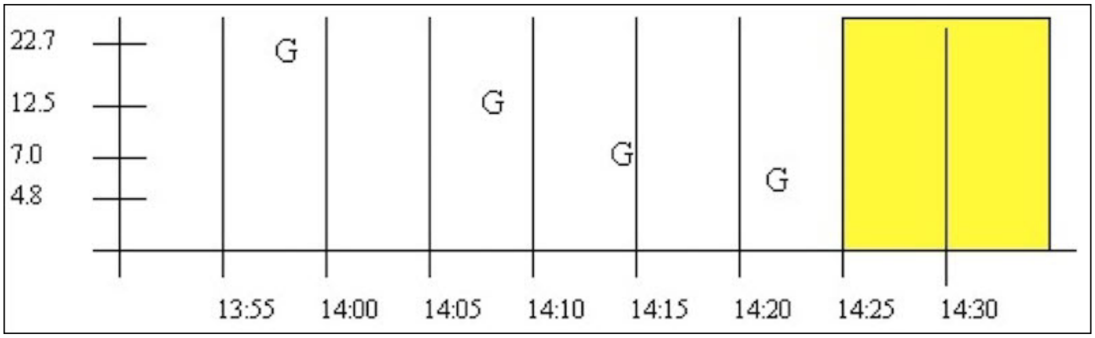
Case 2: No Good Data between Start Time and Interval Timestamp

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.

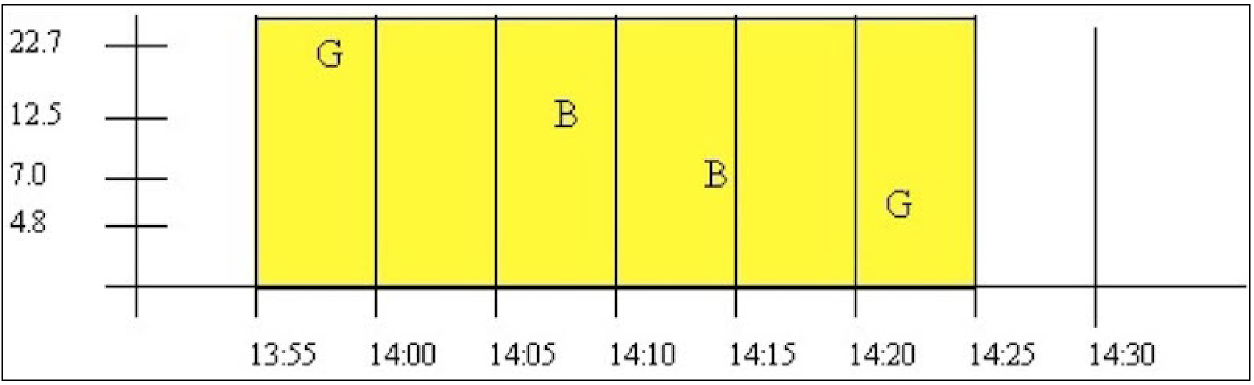
Interpolated and Lab sampling determine the percent good using the same procedure, resulting in a value of either 100 or 0 (though the determined value may be different for each mode even with the same data). Intermediate percent good values are determined only for Calculated retrieval modes.

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.

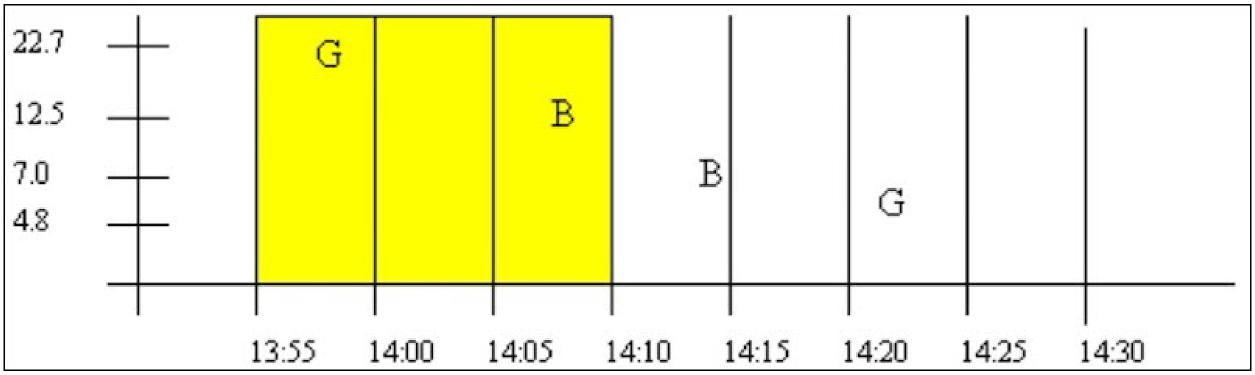
This interval from 14:10 to 14:25 starts with a bad data quality sample but ends with a good sample, so the results are the same. That is, the query:

```
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

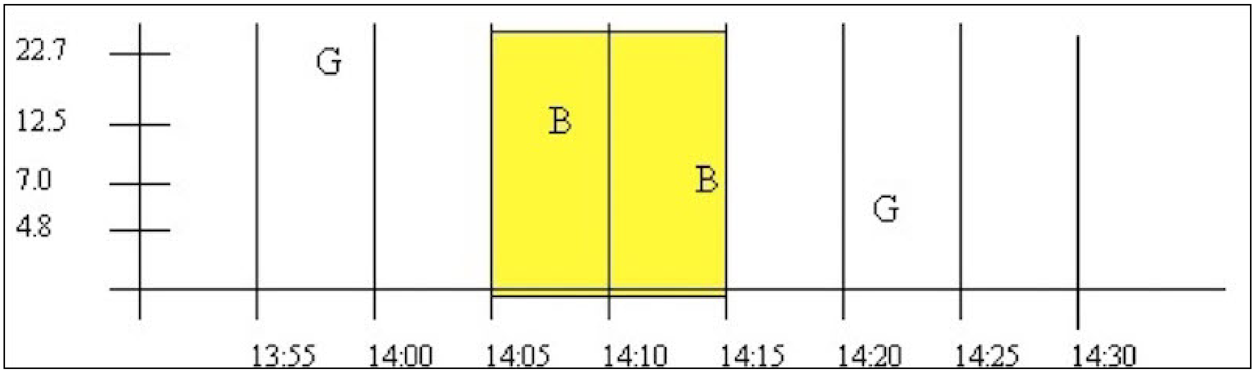
This example shows some data patterns that result in a percent good of 0. An interval ending with a bad data quality sample, always results in a percent good of 0 for the interval.



Timestamp	Value	Quality
29-Mar-2002 14:10:00.000	0.00	0.00

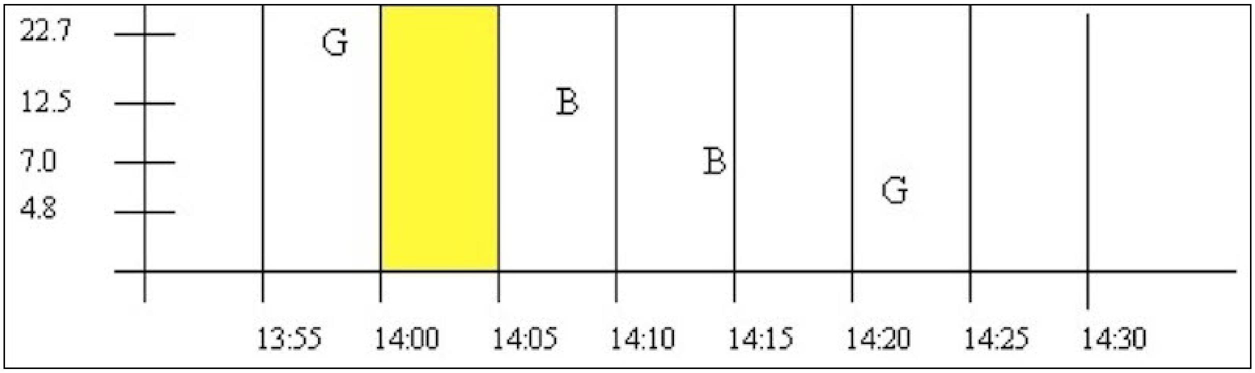
```
select timestamp, value, quality from ihrawdata where samplingmode=interpolated and timestamp >=
'29-Mar-2002 13:55' and timestamp <= '29-Mar-2002 14:10' and tagname  = baddqtag and numberofsamples = 1
```

Timestamp	Value	Quality
29-Mar-2002 14:10:00.000	0.00	0.00



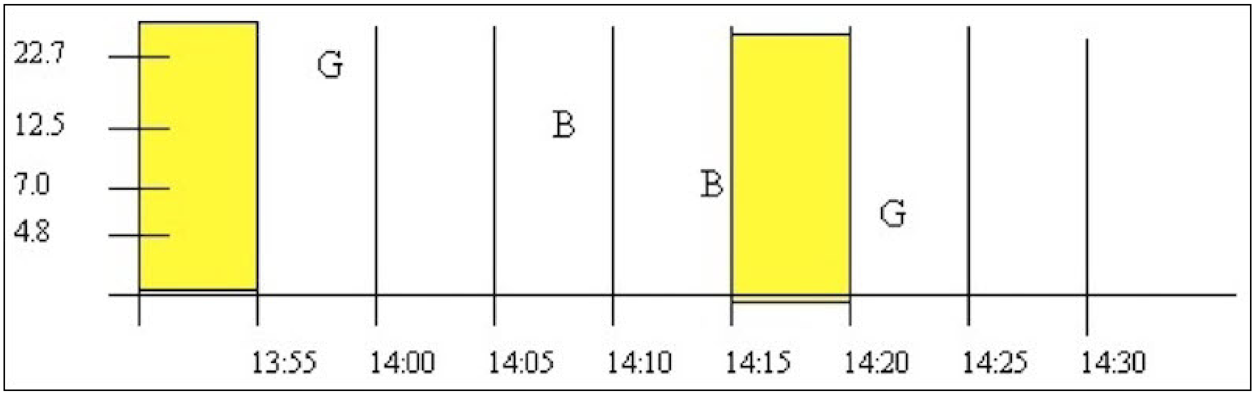
Example: Interpolated and Lab retrieval of an empty interval

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 177. 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. 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. 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. 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. 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. 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. For example, if you specify 12 samples, Historian will divide the time period into 6 intervals, because 12/2 = 6. 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. 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. 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. **Note:** 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. See the note in Minimum for additional information.
Maximum	Displays the maximum value in a specified interval.

Calculation Mode	Results
	Note: 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. 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. 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: • DELTAPOS *(on page 1449)* • DELTANEG *(on page 1465)* • DELTA *(on page 1474)*
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 1449)*
- DELTANEG *(on page 1465)*
- DELTA *(on page 1474)*

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 1449)*
- DELTANEG *(on page 1465)*
- DELTA *(on page 1474)*

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 384)* 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 384)* 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 384)* in Configuration Hub.

Delta Max Negative RPH

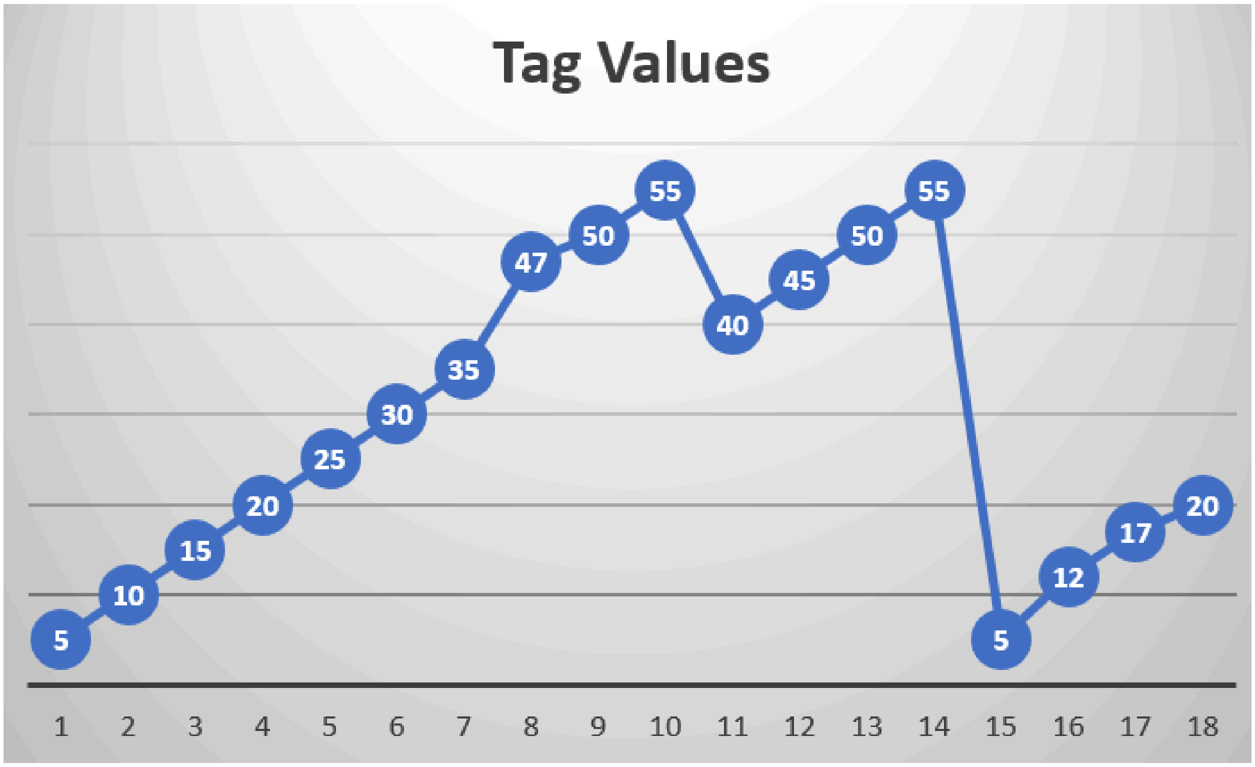
The maximum rate per hour between two consecutive data points in the negative direction. If two consecutive data points exceed this value, they are not considered in a delta query. You can set this value while specifying tags for data collection *(on page 384)* in Configuration Hub.

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

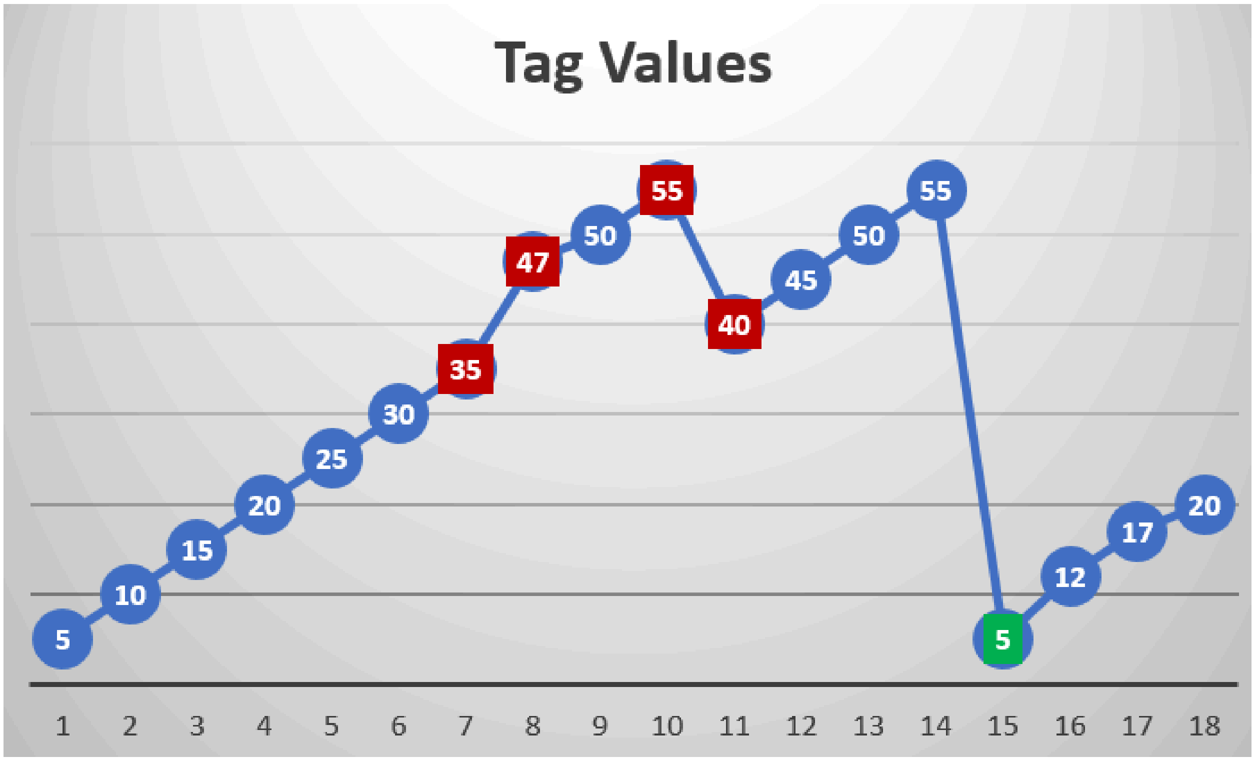
Suppose the following data points are received for the tag:



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

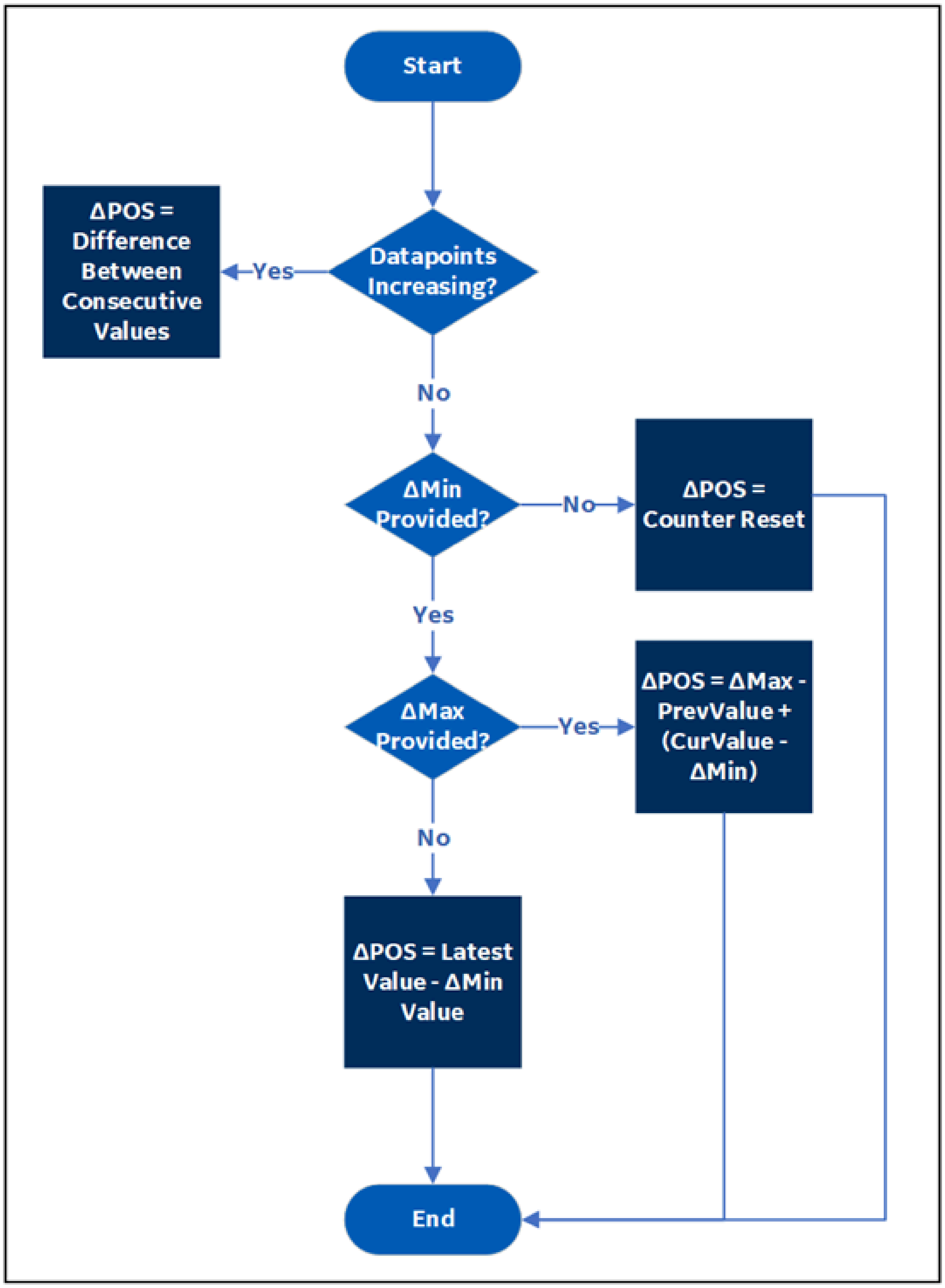
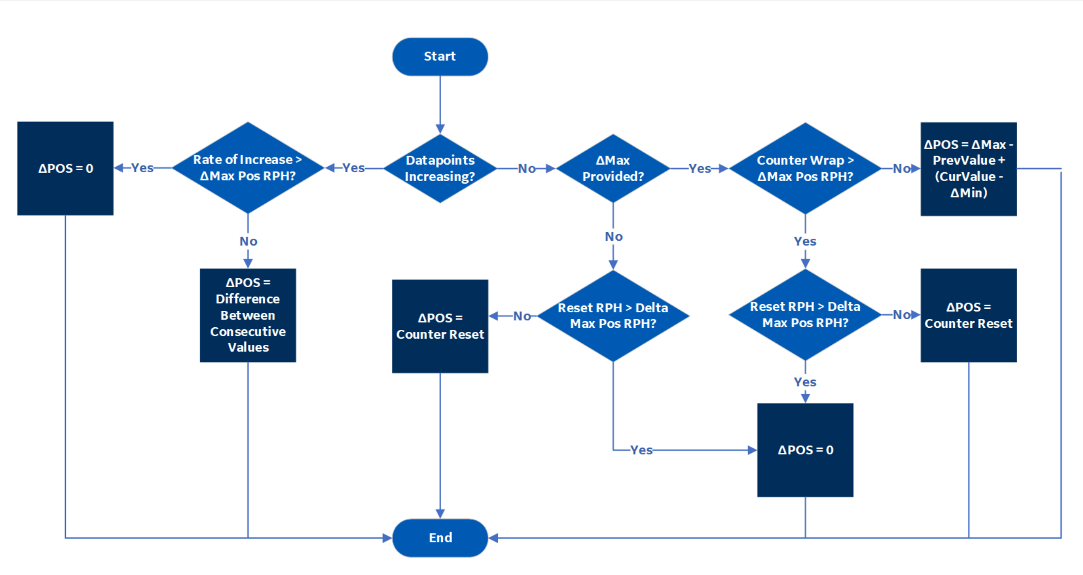

Start
Datapoints Increasing?
Yes
ΔPOS = Difference Between Consecutive Values
No
ΔMin Provided?
No
ΔPOS = Counter Reset
Yes
ΔMax Provided?
Yes
ΔPOS = ΔMax - PrevValue + (CurValue - ΔMin)
No
ΔPOS = Latest Value - ΔMin Value
End

Figure 3. DELTAPOS Calculation When Delta Max Positive RPH is Provided


Start
ΔPOS = 0
Yes
Rate of Increase > ΔMax Pos RPH?
Yes
Datapoints Increasing?
No
ΔMax Provided?
Yes
Counter Wrap > ΔMax Pos RPH?
No
ΔPOS = ΔMax - PrevValue + (CurValue - ΔMin)
No
No
Yes
ΔPOS = Difference Between Consecutive Values
ΔPOS = Counter Reset
No
Reset RPH > Delta Max Pos RPH?
Reset RPH > Delta Max Pos RPH?
No
ΔPOS = Counter Reset
Yes
Yes
ΔPOS = 0
End

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	Value	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	Value	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	Value	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	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	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	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	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	Value	Data Quality
19-Dec-2021 14:01:10	0	Good

Time Stamp	Value	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	Value	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	Value	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	Value	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

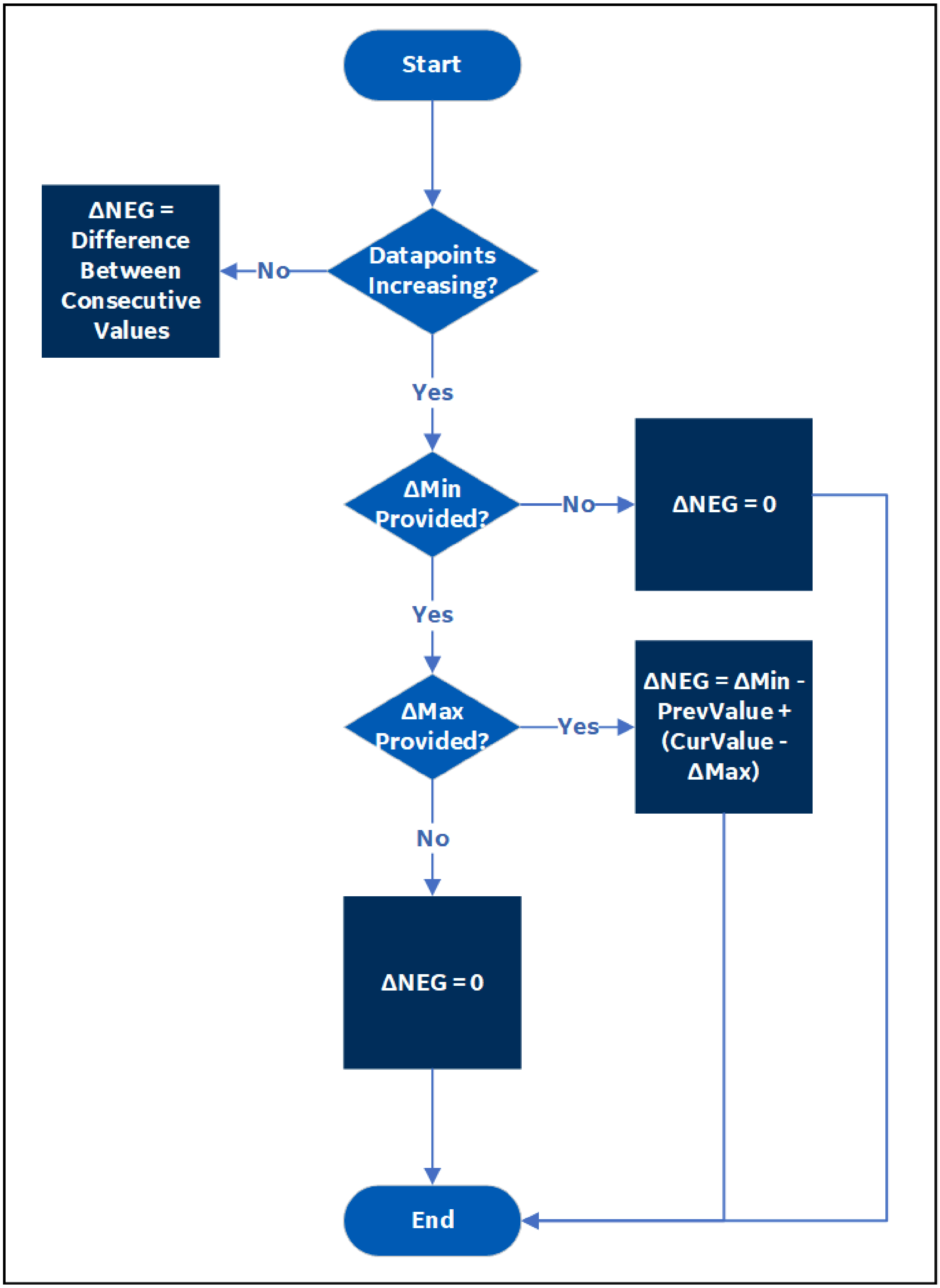
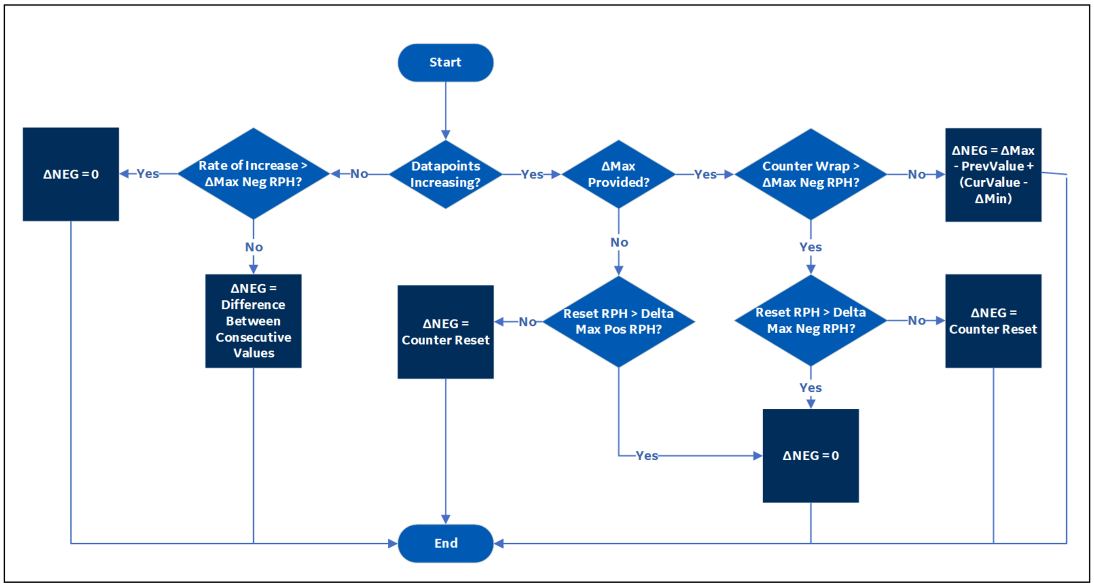


Figure 5. DELTANEG Calculation When Delta Max Negative RPH is Provided


Start
ΔNEG = 0
Yes
Rate of Increase > ΔMax Neg RPH?
No
Datapoints Increasing?
Yes
ΔMax Provided?
Yes
Counter Wrap > ΔMax Neg RPH?
No
ΔNEG = ΔMax - PrevValue + (CurValue - ΔMin)
No
No
Yes
ΔNEG = Difference Between Consecutive Values
ΔNEG = Counter Reset
No
Reset RPH > Delta Max Pos RPH?
Reset RPH > Delta Max Neg RPH?
No
ΔNEG = Counter Reset
Yes
Yes
ΔNEG = 0
End

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	Value	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	Value	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
```

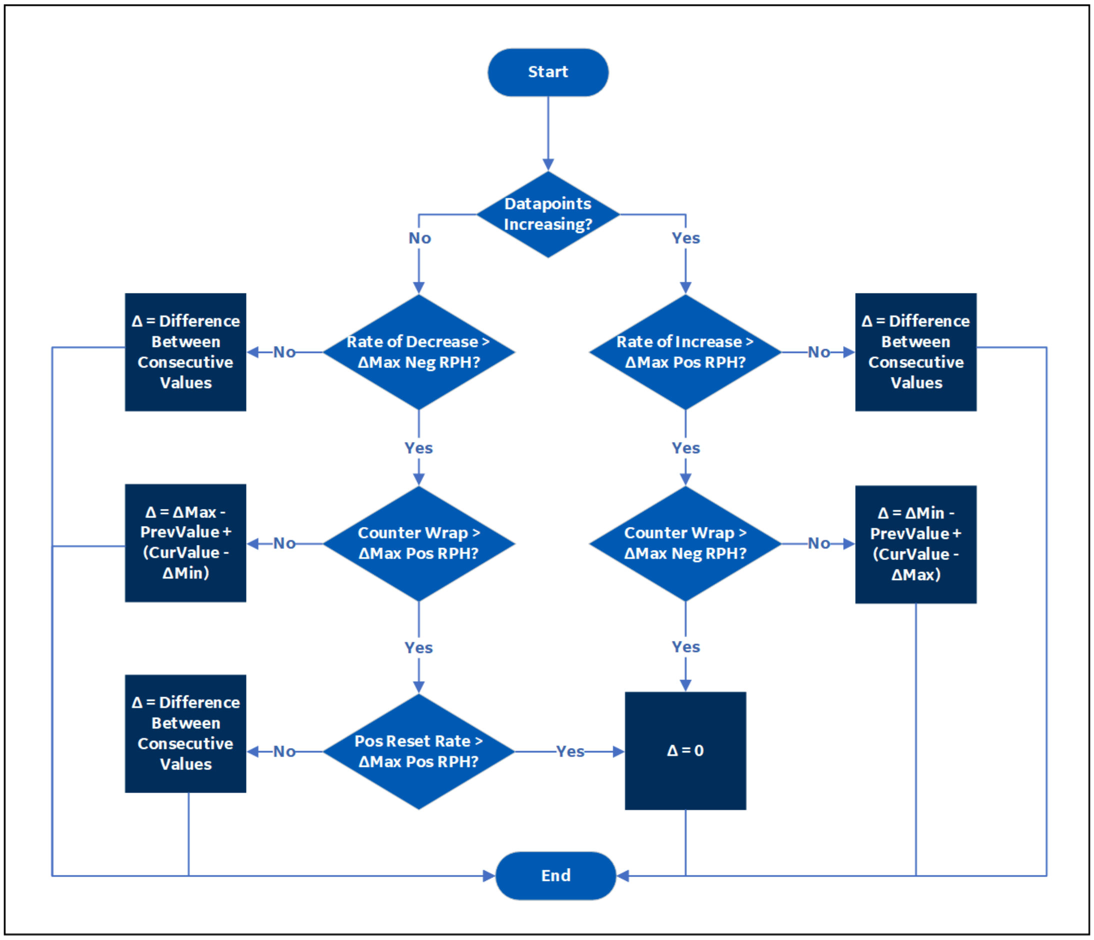
For data that is in the increasing trend, since you have not provided values for Delta Max, Delta Min, and Delta Max Negative RPH, DELTANEG = 0. For the other cases, DELTANEG is the difference between the values.

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


Start
Datapoints Increasing?
No
Yes
Rate of Decrease > ΔMax Neg RPH?
Rate of Increase > ΔMax Pos RPH?
Δ = Difference Between Consecutive Values
No
Yes
Counter Wrap > ΔMax Pos RPH?
Counter Wrap > ΔMax Neg RPH?
Δ = ΔMax - PrevValue + (CurValue - ΔMin)
Δ = ΔMin - PrevValue + (CurValue - ΔMax)
Pos Reset Rate > ΔMax Pos RPH?
Δ = 0
End

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
```

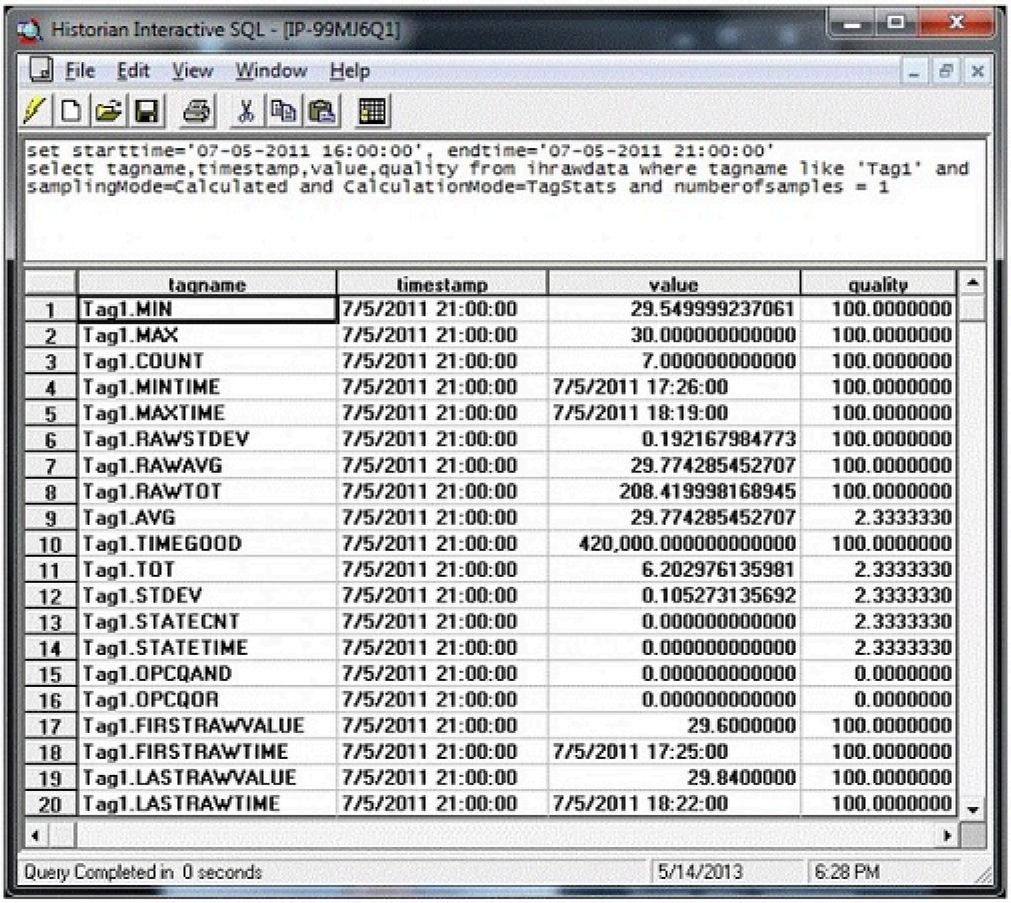
he following output is retrieved.

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`.

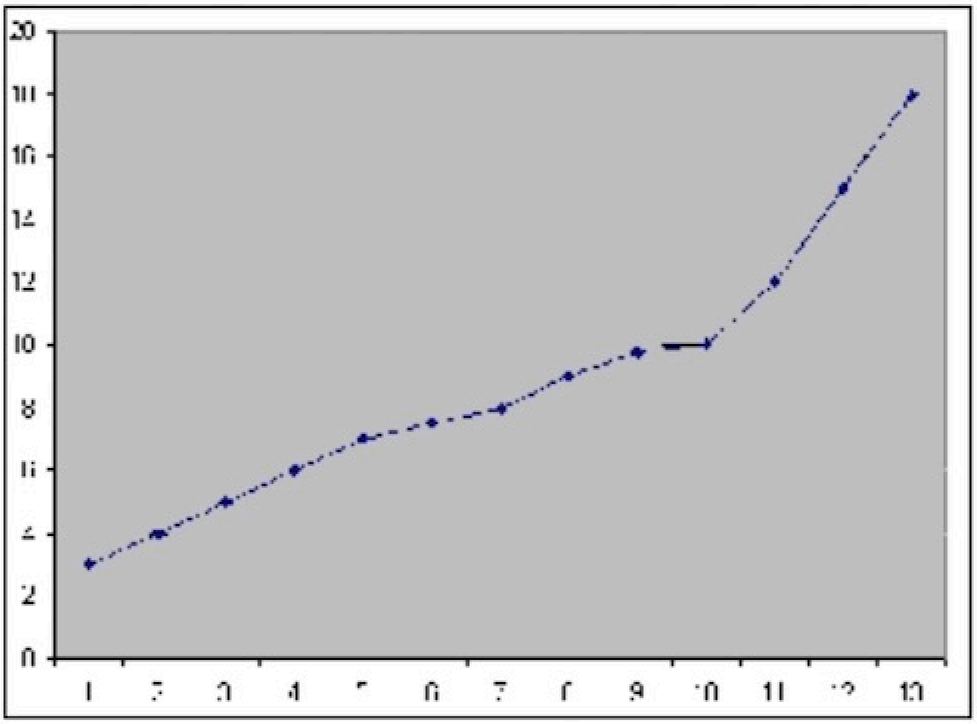

Historian Interactive SQL - [IP-99MJ6Q1]
File
Edit
View
Window
Help
Query Completed in 0 seconds
5/14/2013
6:28 PM

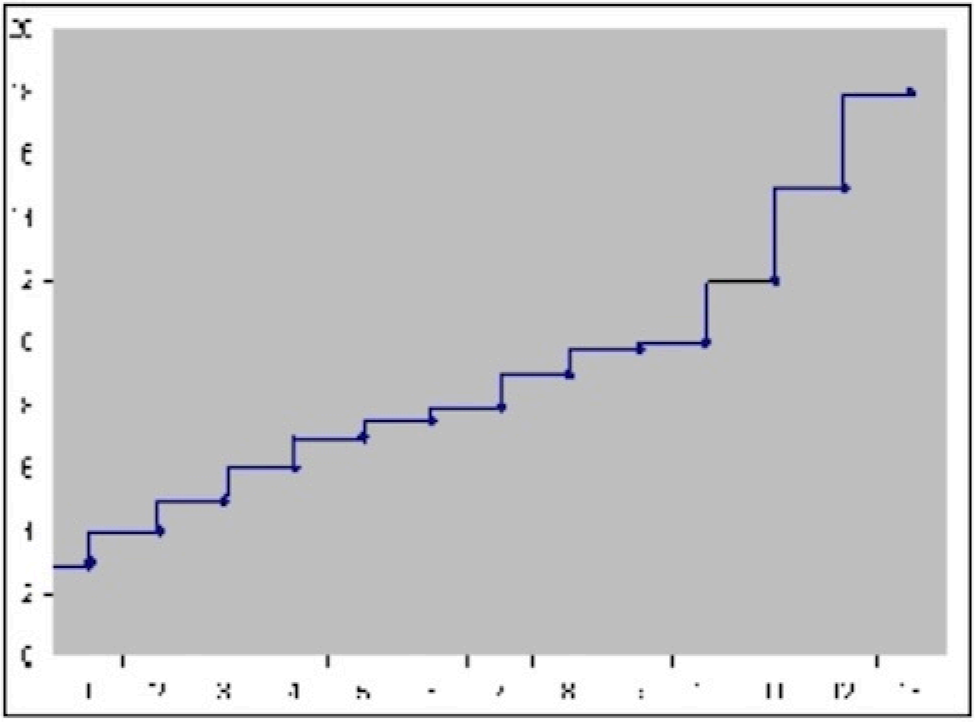
```
set starttime='07-05-2011 16:00:00', endtime='07-05-2011 21:00:00'
select tagname,timestamp,value,quality from ihrawdata where tagname like 'Tag1' and
samplingMode=Calculated and CalculationMode=TagStats and numberofsamples = 1
```

	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 1520)*

Related information

Creating a Data Store *(on page 1519)*

Deleting a Data Store *(on page 1519)*

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 waited 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. 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 16. Monitoring

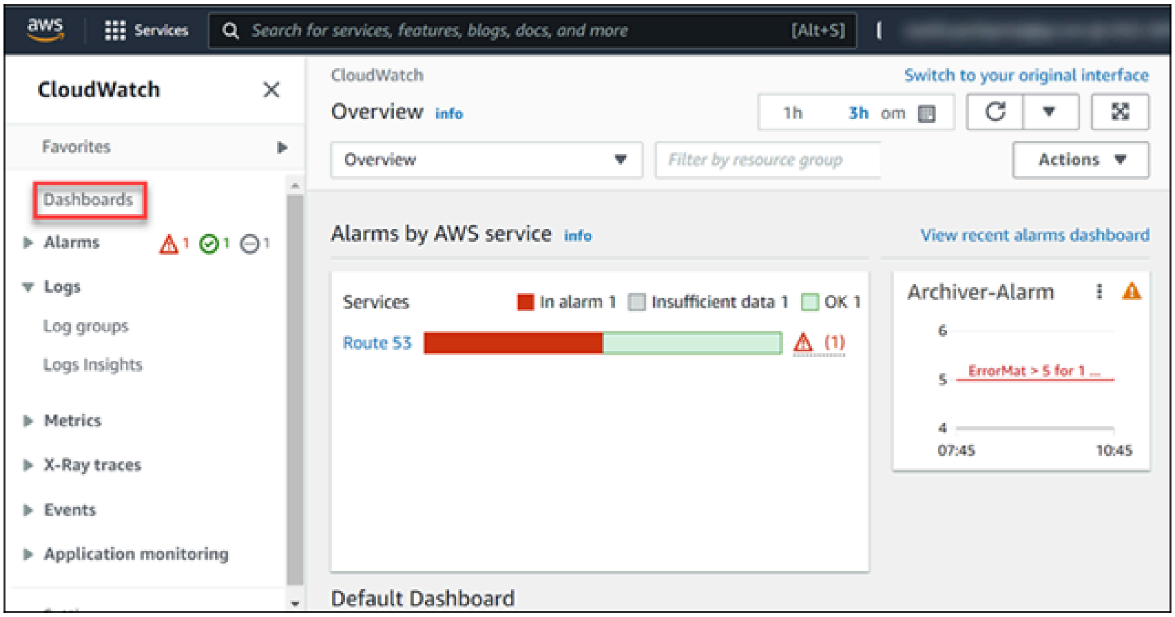
Access Logs

Deploy Proficy Historian for AWS *(on page 43)*.

You can access and analyze logs using CloudWatch. Historian also provides a custom dashboard, which contains a few important widgets to monitor memory utilization, etc.

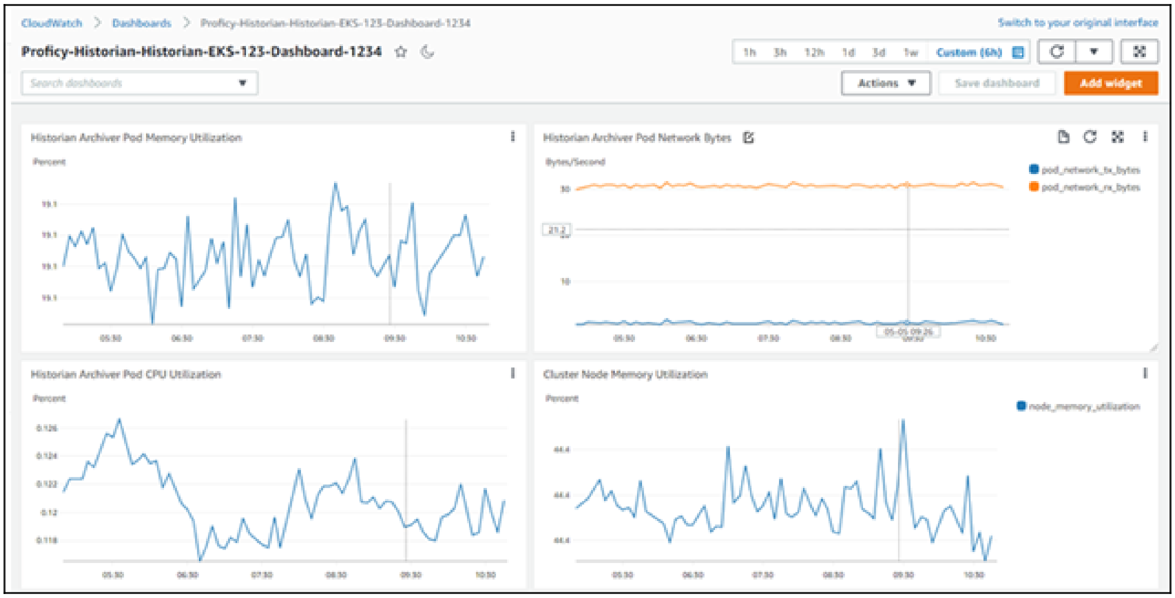
1. Log in to AWS Management Console.
2. Under **All Services > Management & Governance**, select **CloudWatch**.

   The **CloudWatch** page appears.
3. In the left section, select **Dashboards**.

4. From the list of dashboards, select the one that contains the EKS cluster name.

   The following widgets appear.
   - Historian Archiver Pod Memory Utilization
   - Historian Archiver Pod Network Bytes
   - Historian Archiver Pod CPU Utilization
   - Cluster Node Memory Utilization
   - Cluster Node CPU Utilization



You can add more widgets as needed. For information on dashboards and widgets, refer to https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/CloudWatch_Dashboards.html.

5. Under **Logs**, select **Log groups**.
6. From the list of log groups, select the one that contains the EKS cluster name.

   The logs generated by Data Archiver appear.

   The logs are retained for the retention period you set while deploying Proficy Historian for AWS (by default, 30 days). You can change the retention period by selecting the link in the **Retention** column in the list of log groups.

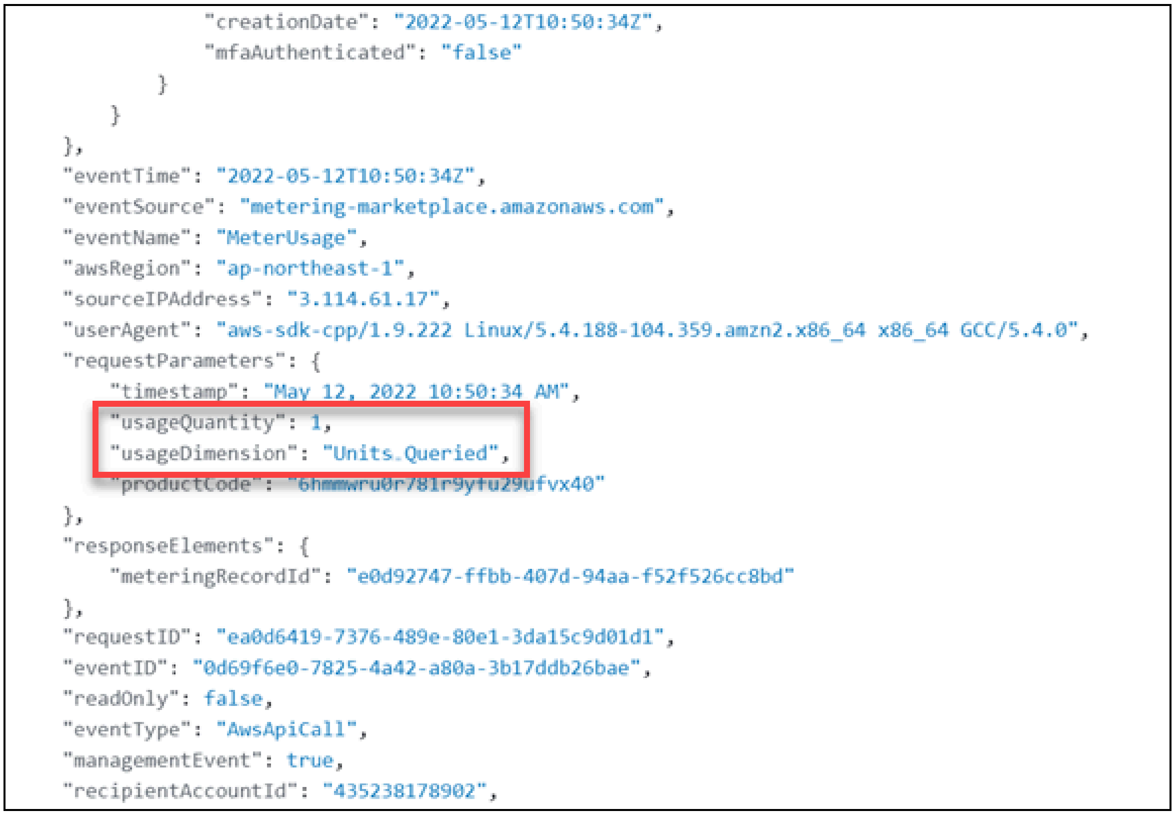
Access Events

Historian uses AWS CloudTrail to capture the consumption of data. Using CloudTrail, you can view the following metrics:

- **The amount of tag data fetched from Data Archiver**

  An event is generated every hour in CloudTrail, containing the total number of data samples (in millions) that are *fetched* from Data Archiver in that hour.

  For example, suppose you plot the trend chart of tag values in the past hour, and 1,045,000 samples are fetched from Data Archiver. In the event, the usageQuantity is set to `1` (indicating 1 million), and the usageDimension is set to `Units_Queried` (indicating that this event is for *fetching* data from Data Archiver). The remaining 45,000 samples are added to the ones collected in the next hour.

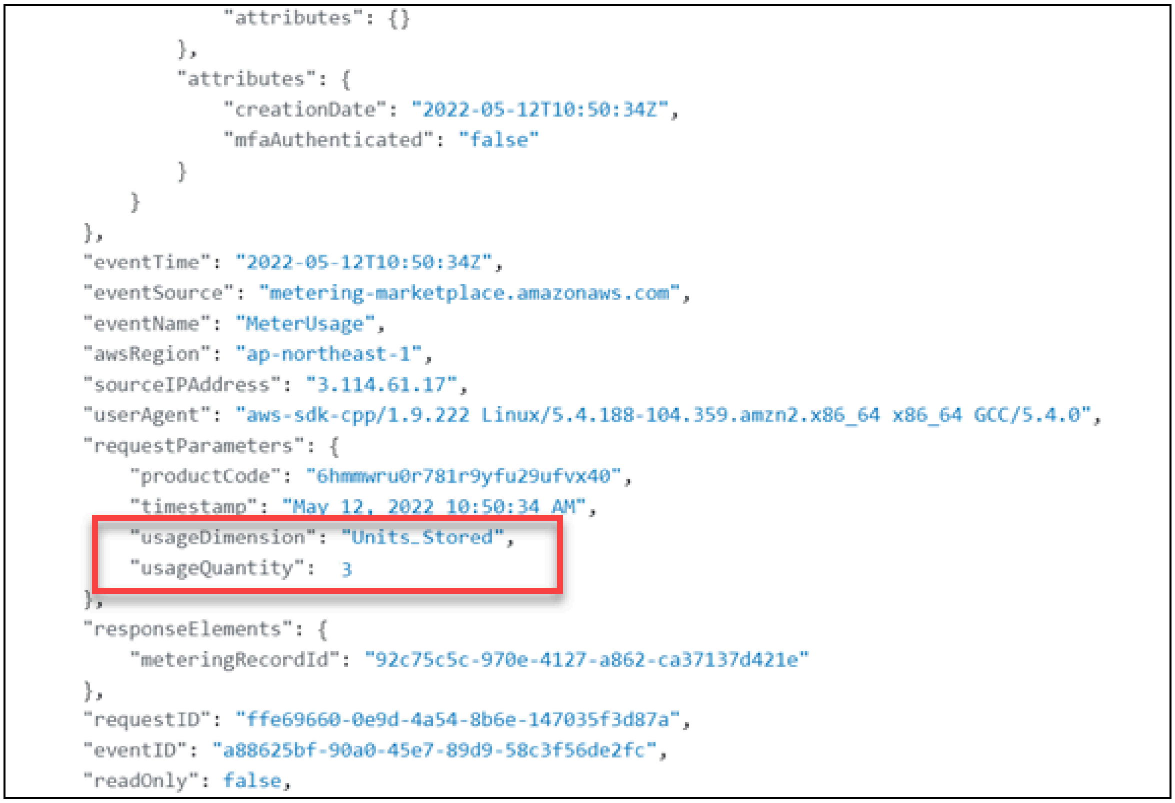


```
                "creationDate": "2022-05-12T10:50:34Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2022-05-12T10:50:34Z",
    "eventSource": "metering-marketplace.amazonaws.com",
    "eventName": "MeterUsage",
    "awsRegion": "ap-northeast-1",
    "sourceIPAddress": "3.114.61.17",
    "userAgent": "aws-sdk-cpp/1.9.222 Linux/5.4.188-104.359.amzn2.x86_64 x86_64 GCC/5.4.0",
    "requestParameters": {
        "timestamp": "May 12, 2022 10:50:34 AM",
        "usageQuantity": 1,
        "usageDimension": "Units_Queried",
        "productCode": "6hmmwru0r781r9yfu29ufvx40"
    },
    "responseElements": {
        "meteringRecordId": "e0d92747-ffbb-407d-94aa-f52f526cc8bd"
    },
    "requestID": "ea0d6419-7376-489e-80e1-3da15c9d01d1",
    "eventID": "0d69f6e0-7825-4a42-a80a-3b17ddb26bae",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "435238178902",
```

- **The amount of tag data written to Data Archiver**

  An event is generated every hour, containing the total number of samples (in millions) that are *written* to Data Archiver.

  For example, if you have 1000 tags, each one set to a resolution of 1 second, 3,600,000 data samples are written to Data Archiver per hour. In the event, usageQuantity is set to `3`, and usageDimension is set to `Units_Stored`. The remaining 600,000 samples are added to the next hour.

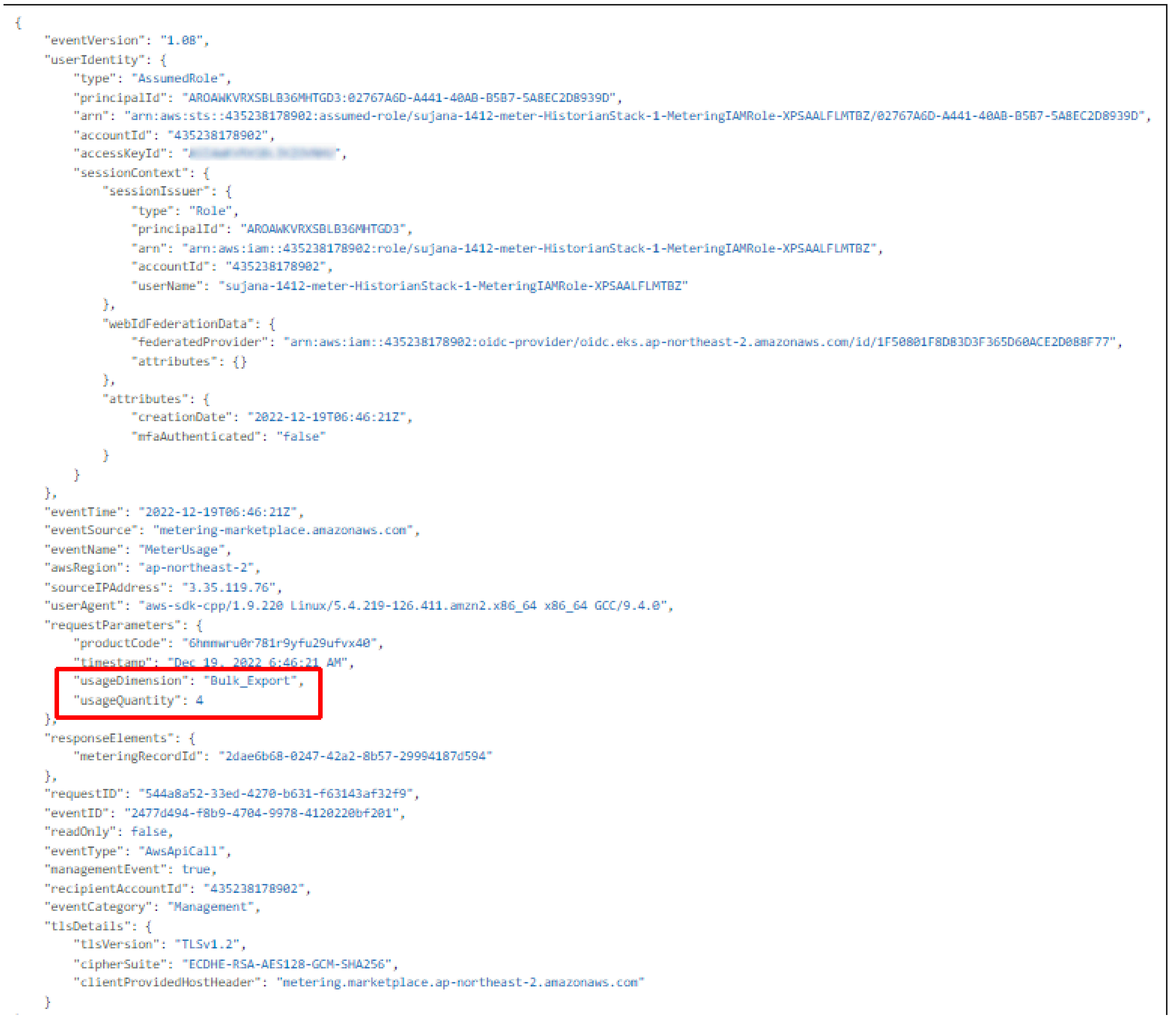


```
                "attributes": {}
            },
            "attributes": {
                "creationDate": "2022-05-12T10:50:34Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2022-05-12T10:50:34Z",
    "eventSource": "metering-marketplace.amazonaws.com",
    "eventName": "MeterUsage",
    "awsRegion": "ap-northeast-1",
    "sourceIPAddress": "3.114.61.17",
    "userAgent": "aws-sdk-cpp/1.9.222 Linux/5.4.188-104.359.amzn2.x86_64 x86_64 GCC/5.4.0",
    "requestParameters": {
        "productCode": "6hmmwru0r781r9yfu29ufvx40",
        "timestamp": "May 12, 2022 10:50:34 AM",
        "usageDimension": "Units_Stored",
        "usageQuantity":  3
    },
    "responseElements": {
        "meteringRecordId": "92c75c5c-970e-4127-a862-ca37137d421e"
    },
    "requestID": "ffe69660-0e9d-4a54-8b6e-147035f3d87a",
    "eventID": "a88625bf-90a0-45e7-89d9-58c3f56de2fc",
    "readOnly": false,
```

- **The amount of tag data exported in bulk to Amazon S3 in the Parquet or CSV file format**

  An event is generated every hour, containing the total number of samples (in millions) that are *exported* from Data Archiver to Amazon S3 in the Parquet or CSV file format.

For example, if 4,785,000 data samples are exported per hour, the usageQuantity is set to `4`, and usageDimension is set to `Bulk_Export`. The remaining 785,000 samples are added to the next hour.



```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAWKVRXSBLB36MHTGD3:02767A6D-A441-40AB-B5B7-5A8EC2D8939D",
        "arn": "arn:aws:sts::435238178902:assumed-role/sujana-1412-meter-HistorianStack-1-MeteringIAMRole-XPSAALFLMTBZ/02767A6D-A441-40AB-B5B7-5A8EC2D8939D",
        "accountId": "435238178902",
        "accessKeyId": "[redacted]",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAWKVRXSBLB36MHTGD3",
                "arn": "arn:aws:iam::435238178902:role/sujana-1412-meter-HistorianStack-1-MeteringIAMRole-XPSAALFLMTBZ",
                "accountId": "435238178902",
                "userName": "sujana-1412-meter-HistorianStack-1-MeteringIAMRole-XPSAALFLMTBZ"
            },
            "webIdFederationData": {
                "federatedProvider": "arn:aws:iam::435238178902:oidc-provider/oidc.eks.ap-northeast-2.amazonaws.com/id/1F50801F8D83D3F365D60ACE2D088F77",
                "attributes": {}
            },
            "attributes": {
                "creationDate": "2022-12-19T06:46:21Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2022-12-19T06:46:21Z",
    "eventSource": "metering-marketplace.amazonaws.com",
    "eventName": "MeterUsage",
    "awsRegion": "ap-northeast-2",
    "sourceIPAddress": "3.35.119.76",
    "userAgent": "aws-sdk-cpp/1.9.220 Linux/5.4.219-126.411.amzn2.x86_64 x86_64 GCC/9.4.0",
    "requestParameters": {
        "productCode": "6hmmwru0r781r9yfu29ufvx40",
        "timestamp": "Dec 19, 2022 6:46:21 AM",
        "usageDimension": "Bulk_Export",
        "usageQuantity": 4
    },
    "responseElements": {
        "meteringRecordId": "2dae6b68-0247-42a2-8b57-29994187d594"
    },
    "requestID": "544a8a52-33ed-4270-b631-f63143af32f9",
    "eventID": "2477d494-f8b9-4704-9978-4120220bf201",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "435238178902",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.2",
        "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
        "clientProvidedHostHeader": "metering.marketplace.ap-northeast-2.amazonaws.com"
    }
```

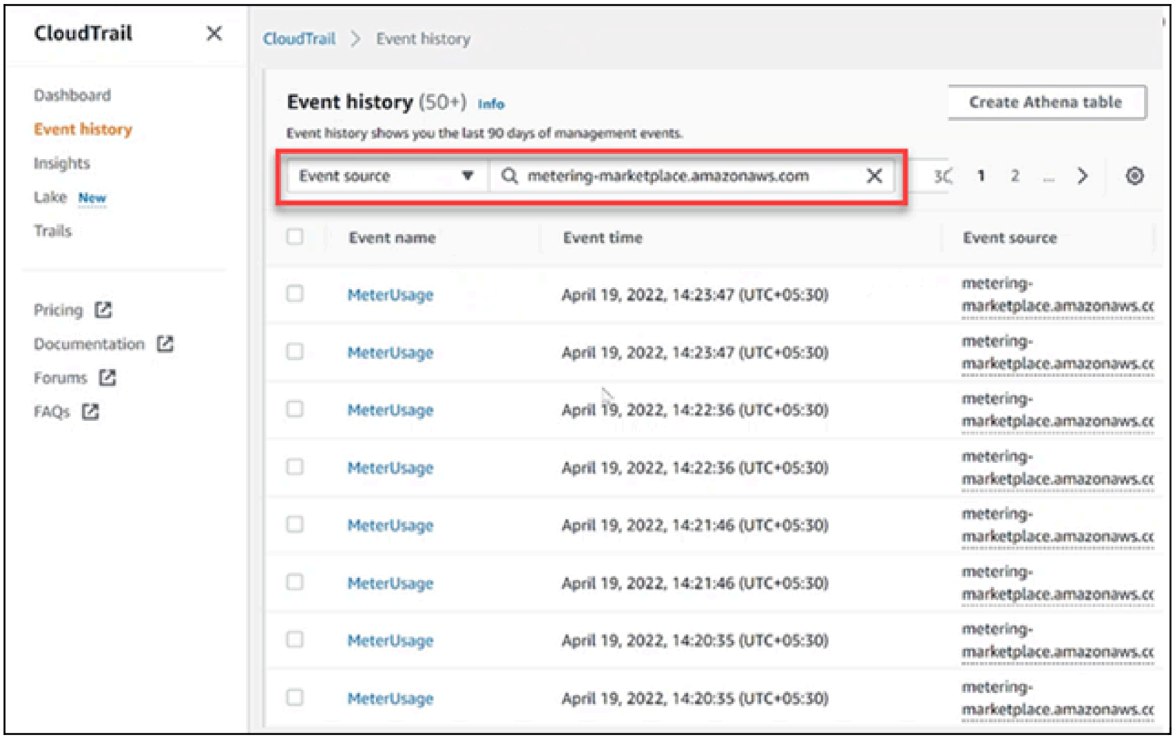
Events are also generated when Data Archiver is started or stopped.

Depending on the data samples written, fetched, or exported from Data Archiver, the price *(on page 29)* of using Proficy Historian for Cloud is calculated. You can then make the payment *(on page 29)*. If, however, you are using the BYOL model (that is, you have applied a Historian license *(on page 56)*), you are not required to pay for the consumption.

Note:
The price for exporting data is different from that of fetching or sending data to Data Archiver. It is applicable only if you are using the consumption-based model for Proficy Historian for Cloud.

Also, you must deploy the scheduled export feature separately. For more information, refer to About Sending Data to Amazon S3 *(on page 129)*.

1. Log in to the AWS console.
2. Under **All Services > Management & Governance**, select **CloudTrail**.
3. In the left section, select **Event history**.
4. Set the **Event source** to **metering-marketplace.amazonaws.com**.



Event source ▼ metering-marketplace.amazonaws.com

□	Event name	Event time	Event source
□	MeterUsage	April 19, 2022, 14:23:47 (UTC+05:30)	metering-marketplace.amazonaws.cc
□	MeterUsage	April 19, 2022, 14:23:47 (UTC+05:30)	metering-marketplace.amazonaws.cc
□	MeterUsage	April 19, 2022, 14:22:36 (UTC+05:30)	metering-marketplace.amazonaws.cc
□	MeterUsage	April 19, 2022, 14:22:36 (UTC+05:30)	metering-marketplace.amazonaws.cc
□	MeterUsage	April 19, 2022, 14:21:46 (UTC+05:30)	metering-marketplace.amazonaws.cc
□	MeterUsage	April 19, 2022, 14:21:46 (UTC+05:30)	metering-marketplace.amazonaws.cc
□	MeterUsage	April 19, 2022, 14:20:35 (UTC+05:30)	metering-marketplace.amazonaws.cc
□	MeterUsage	April 19, 2022, 14:20:35 (UTC+05:30)	metering-marketplace.amazonaws.cc

A list of read/write/export events appears. If needed, you can filter the events for a specific duration.

5. Select the event whose details you want to access.

   The **Meter Usage** page appears, displaying the event details. In the **Event Record** section, the usageDimension parameter indicates whether the event is for fetching, writing, or exporting data. The usageQuantity parameter contains the number of samples that are fetched, written, or exported in that duration.

Chapter 17. Troubleshooting

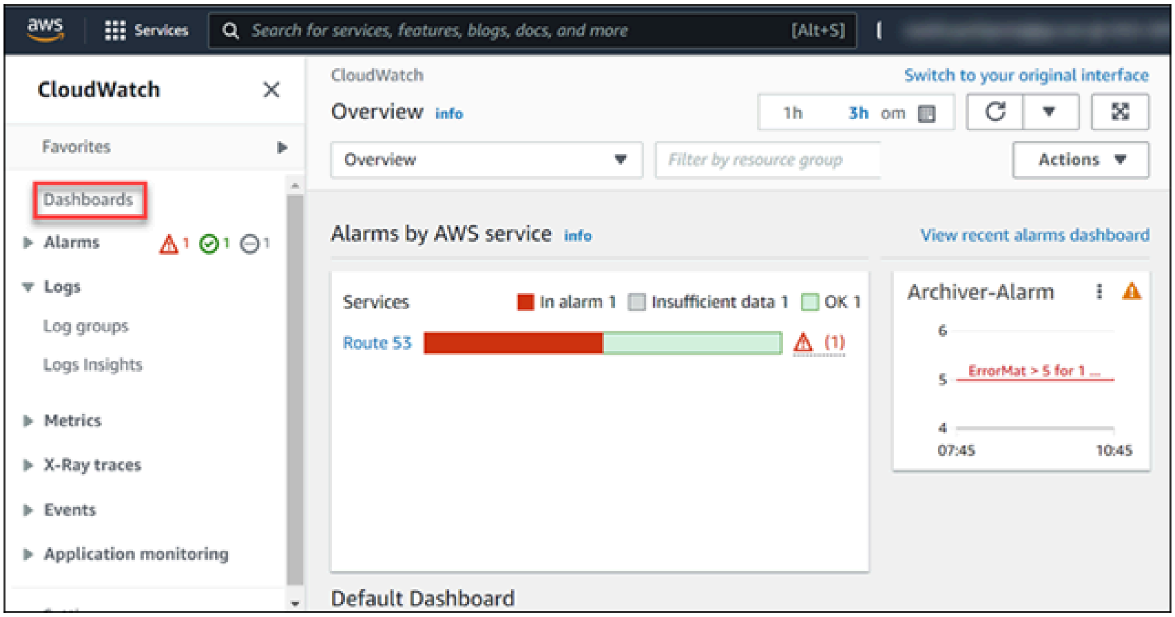
Access Logs

Deploy Proficy Historian for AWS *(on page 43)*.

You can access and analyze logs using CloudWatch. Historian also provides a custom dashboard, which contains a few important widgets to monitor memory utilization, etc.

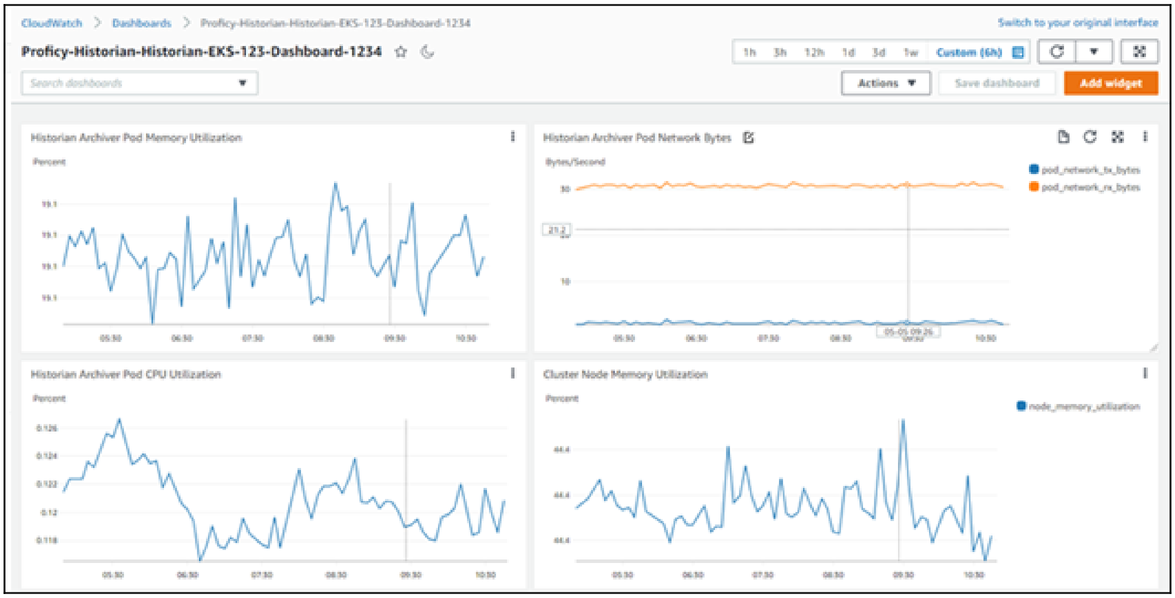
1. Log in to AWS Management Console.
2. Under **All Services > Management & Governance**, select **CloudWatch**.

   The **CloudWatch** page appears.
3. In the left section, select **Dashboards**.



4. From the list of dashboards, select the one that contains the EKS cluster name.

   The following widgets appear.
   - Historian Archiver Pod Memory Utilization
   - Historian Archiver Pod Network Bytes
   - Historian Archiver Pod CPU Utilization
   - Cluster Node Memory Utilization
   - Cluster Node CPU Utilization



You can add more widgets as needed. For information on dashboards and widgets, refer to https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/CloudWatch_Dashboards.html.

5. Under **Logs**, select **Log groups**.
6. From the list of log groups, select the one that contains the EKS cluster name.

   The logs generated by Data Archiver appear.

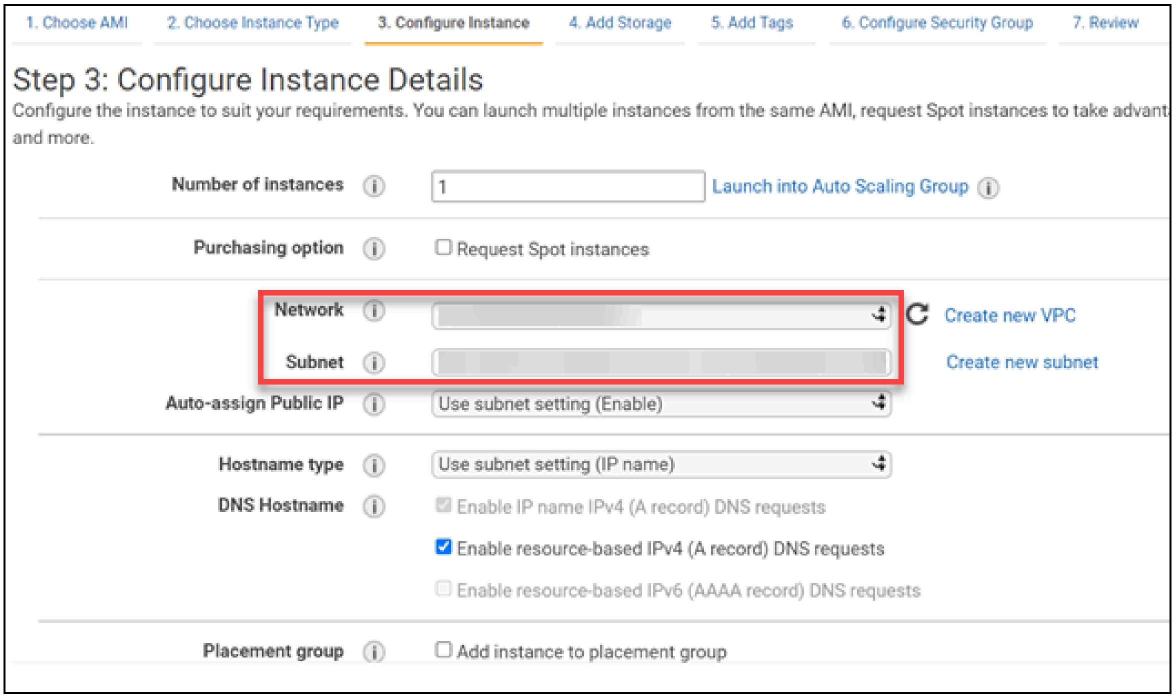
   The logs are retained for the retention period you set while deploying Proficy Historian for AWS (by default, 30 days). You can change the retention period by selecting the link in the **Retention** column in the list of log groups.

Access Archives

Deploy Proficy Historian for AWS *(on page 43)*.

Archives and log files are stored in Elastic File System (EFS). This topic describes how to access them for troubleshooting, monitoring, etc.

1. Launch an EC2 instance. While launching the EC2 instance, under **Configure Instance Details**, in the **Network** field, select the VPC in which you have deployed Proficy Historian for AWS. In addition, in the **Subnet** field, select the public subnet of the VPC.



Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advant and more.

Number of instances: 1 — Launch into Auto Scaling Group

Purchasing option: □ Request Spot instances

Network — Create new VPC

Subnet — Create new subnet

Auto-assign Public IP: Use subnet setting (Enable)

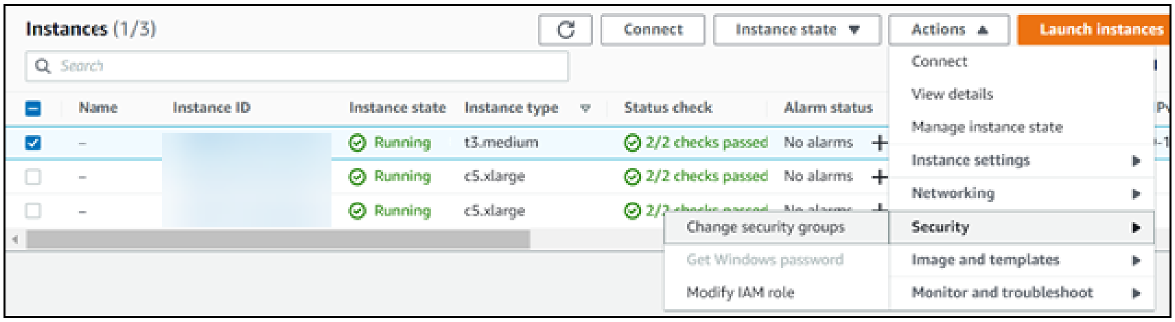
Hostname type: Use subnet setting (IP name)

DNS Hostname:
☑ Enable IP name IPv4 (A record) DNS requests
☑ Enable resource-based IPv4 (A record) DNS requests
□ Enable resource-based IPv6 (AAAA record) DNS requests

Placement group: □ Add instance to placement group

Note down the path to the .pem key file and the public IP address or DNS of the EC2 instance.

2. In the list of EC2 instances, select the check box corresponding to the one that you have created.
3. Select **Actions > Security > Change security groups**.

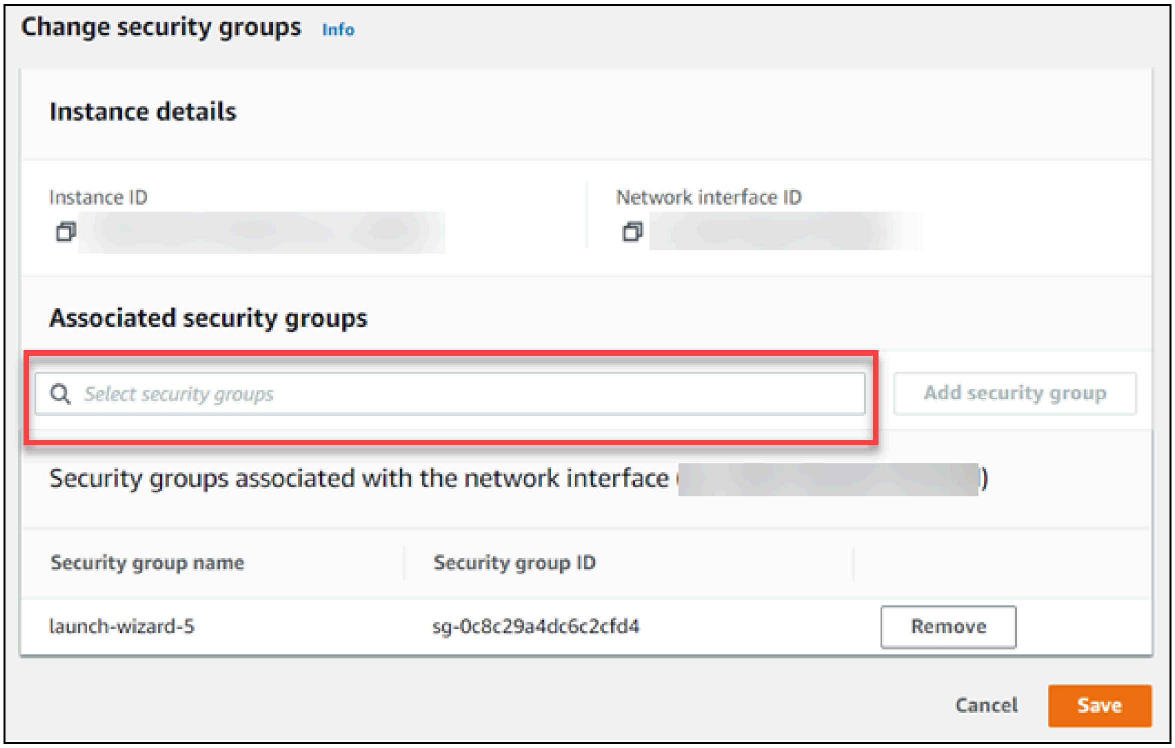


Instances (1/3)

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status
☑	-		Running	t3.medium	2/2 checks passed	No alarms
□	-		Running	c5.xlarge	2/2 checks passed	No alarms
□	-		Running	c5.xlarge	2/2 checks passed	No alarms

The **Change security groups** window appears.

4. In the **Select security groups** field, select the EKS cluster that you have created while deploying Proficy Historian for AWS, select **Add security group**, and then select **Save**.



Change security groups Info

Instance details

Instance ID [redacted]

Network interface ID [redacted]

Associated security groups

Select security groups

Add security group

Security groups associated with the network interface ([redacted])

Security group name	Security group ID	
launch-wizard-5	sg-0c8c29a4dc6c2cfd4	Remove

Cancel Save

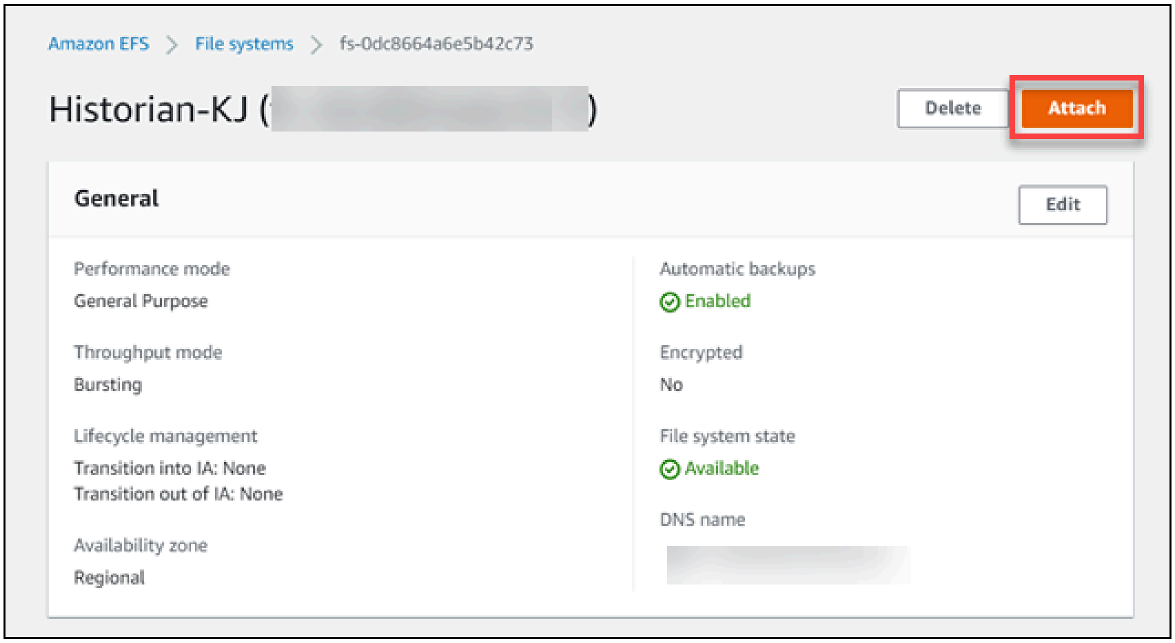
CAUTION:

Do not remove any security group.

5. From the machine using which you want to access archive files, run the following command to connect to the EC2 instance:

```
ssh -i <path to your .pem key file> ec2-user@<public IP/DNS of the EC2 instance>
```

6. Create a directory in the EC2 instance.
7. Access the AWS console.
8. Select **EFS**, and then select the EFS instance that contains the EKS cluster name.
9. Select **Attach**.



Amazon EFS > File systems > fs-0dc8664a6e5b42c73

Historian-KJ ([redacted])

General

Performance mode
General Purpose

Throughput mode
Bursting

Lifecycle management
Transition into IA: None
Transition out of IA: None

Availability zone
Regional

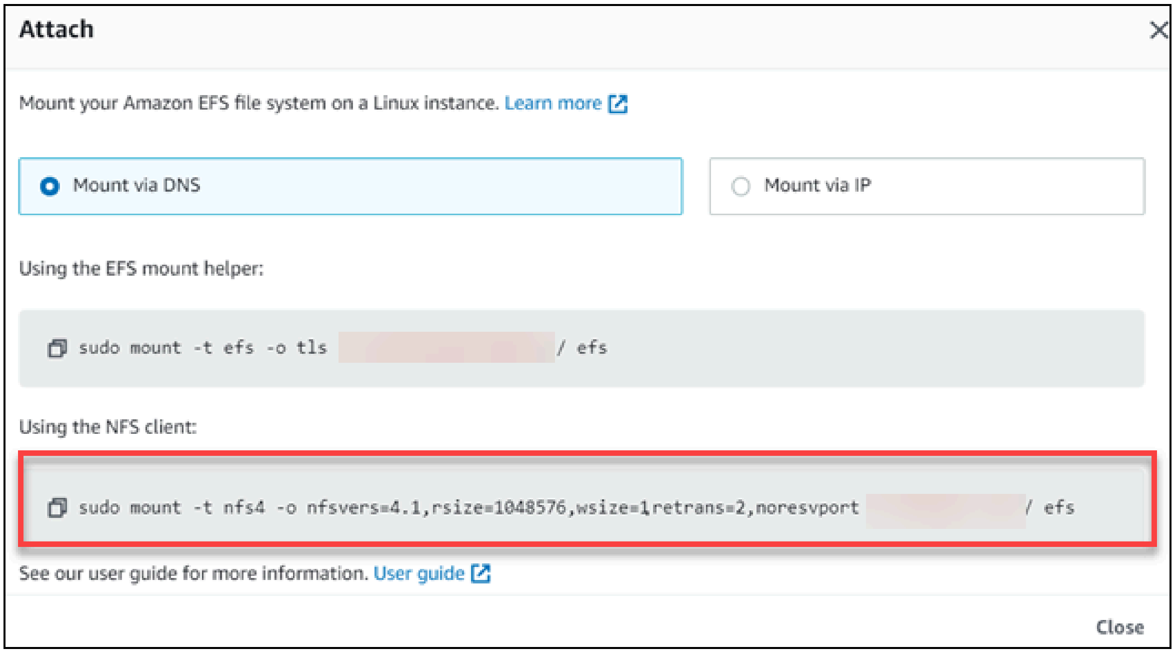
Automatic backups
✓ Enabled

Encrypted
No

File system state
✓ Available

DNS name
[redacted]

10. Copy the value in the **Using the NFS client** field.



Attach

Mount your Amazon EFS file system on a Linux instance. Learn more

● Mount via DNS ○ Mount via IP

Using the EFS mount helper:

```
sudo mount -t efs -o tls [redacted]/ efs
```

Using the NFS client:

```
sudo mount -t nfs4 -o nfsvers=4.1,rsize=1048576,wsize=1retrans=2,noresvport [redacted]/ efs
```

See our user guide for more information. User guide

11. Paste the value in the EC2 instance terminal.
12. Access the directory that you have created in step 6.

    The directory contains a folder named historian, which contains the archives and log files.

Troubleshoot Connection Issues

Unable to Connect the Web Admin Console with the Historian Server

Suggested solution: If you restart the EC2 instance on which you have deployed the Web Admin console, you must generate the configuration file once again to connect the Web Admin console with the Historian server.

Unable to Start the iFIX Collector

Description: Sometimes, an error occurs while starting the iFIX collector, stating that the collector fails to start and prompting you to delete the collector. This happens because the iFIX collector service is created with the default path set to `C:\Program Files (x86)\GE\iFIX\ihFIXCollector.exe`, which does not exist.

Workaround:

1. Select No in the error message.
2. In iFIX System Configuration (SCU), set the task parameters as follows:
   - **Filename**: Enter `<installation drive>:\Program Files (x86)\GE Digital \Historian iFix Collector`.
   - **Command Line**: Enter `NOSERVICE REG=<collector name>`.

Unable to Change the Destination of a Collector

If you try to change the destination of a collector from an on-premise Historian server to the cloud counterpart, an error occurs, stating, "Unable To Start the Collector Instance...Please check the parameters provided are valid and edit the Collector Instance...Press any key to exit the Application".

This issue occurs only if the collector name does not contain a prefix of the collector type.

Workaround:

1. Access the registry entry for the collector instance.
2. Rename the instance as follows, and close the registry: *<collector type>-<original name of the instance>*

   For example, if the instance name of a Simulation collector is `sim1`, rename it `SimulationCollector-sim1`.

   Use one of the following prefixes based on the collector type:

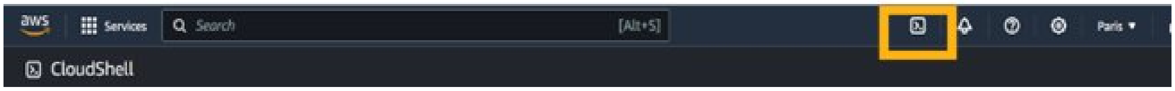
    - SimulationCollector-
    - OPCCollector-
    - OPCUACollector-
    - OPCHDACollector-
    - PiCollector-
    - PIDistributor-
    - iFixCollector-
    - ServerToServerCollector-
    - MQTTCollector-
3. Change the deistination of the collector instance. When asked for the name of the collector instance, provide the *original* name (without the prefix). In the case of the previous example, enter `sim1`.

   An error message appears, but it is expected; you can ignore it.
4. Access the registry entry for the collector instance, and undo the changes you made in step 2 (that is, the collector instance should now contain the original name).
5. Close the registry, and start the collector instance.

Restart a Pod

The following steps explain how to restart a pod in your own environment

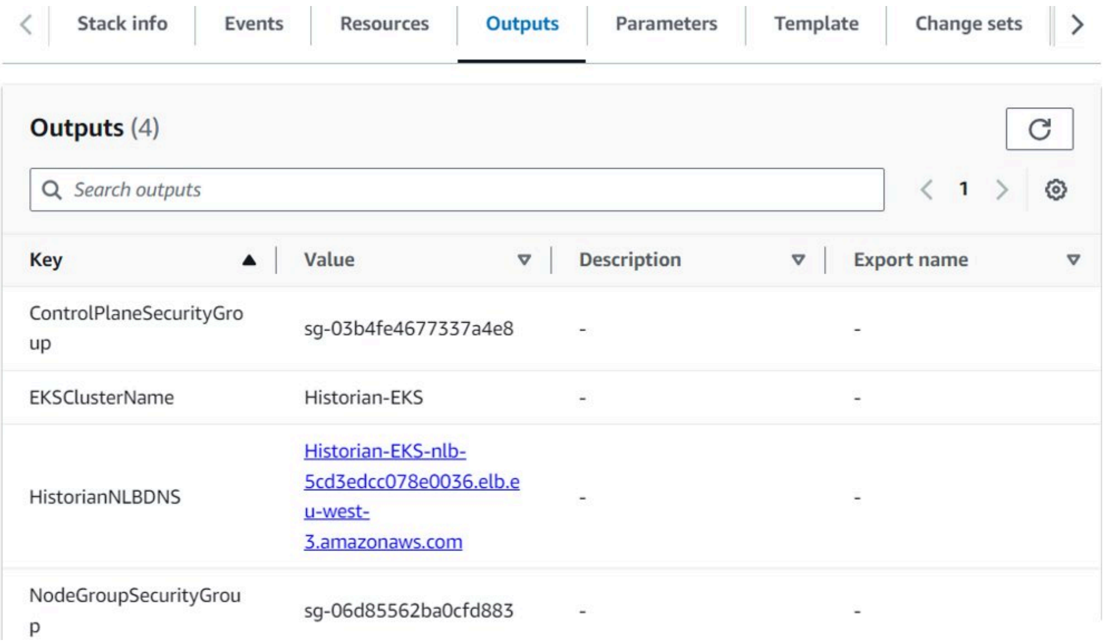
1. Log in to AWS Management Console.
2. Click the icon to open AWS Cloudshell as shown in the following figure.

3. Run the following command to get your EKS cluster authority:

   ```
   aws eks update-kubeconfig --name <EKS Cluster Name> --region <region name >
   ```

   After you complete the deployment, you can get EKS cluster name in the output like this:

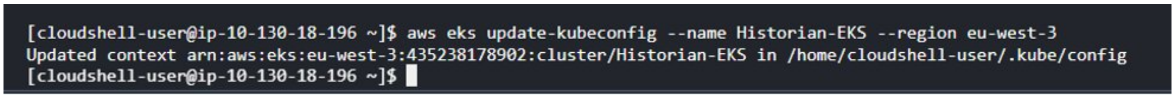


Outputs (4)

Key	Value	Description	Export name
ControlPlaneSecurityGroup	sg-03b4fe4677337a4e8	-	-
EKSClusterName	Historian-EKS	-	-
HistorianNLBDNS	Historian-EKS-nlb-5cd3edcc078e0036.elb.eu-west-3.amazonaws.com	-	-
NodeGroupSecurityGroup	sg-06d85562ba0cfd883	-	-

The region name is the region where the EKS cluster is deployed.

The output of the following command will show something like the following:

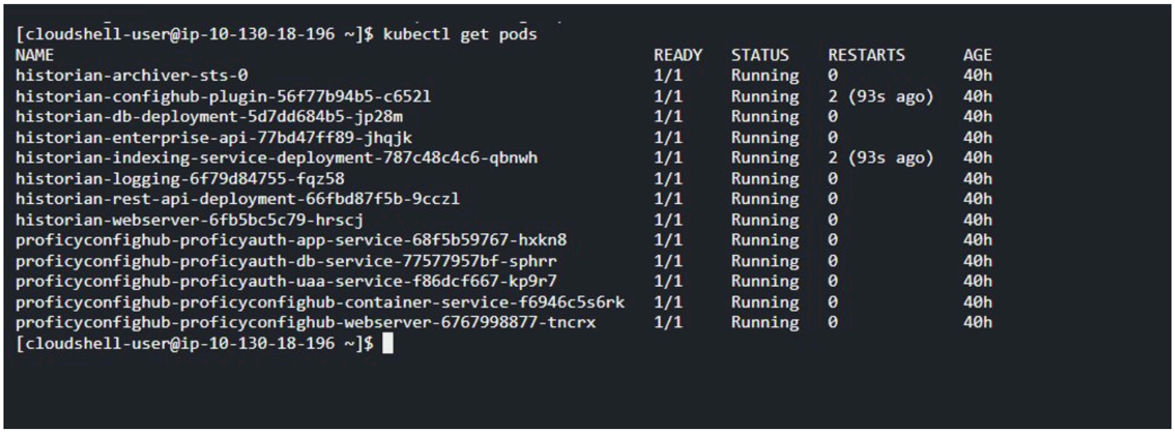


```
[cloudshell-user@ip-10-130-18-196 ~]$ aws eks update-kubeconfig --name Historian-EKS --region eu-west-3
Updated context arn:aws:eks:eu-west-3:435238178902:cluster/Historian-EKS in /home/cloudshell-user/.kube/config
[cloudshell-user@ip-10-130-18-196 ~]$ 
```

4. Run the following command to get all running pods into your EKS cluster:

```
Kubectl get pods
```

The output of the previous command will generate a result similar to the following:



```
[cloudshell-user@ip-10-130-18-196 ~]$ kubectl get pods
NAME                                                           READY   STATUS    RESTARTS      AGE
historian-archiver-sts-0                                       1/1     Running   0             40h
historian-confighub-plugin-56f77b94b5-c652l                    1/1     Running   2 (93s ago)   40h
historian-db-deployment-5d7dd684b5-jp28m                       1/1     Running   0             40h
historian-enterprise-api-77bd47ff89-jhqjk                      1/1     Running   0             40h
historian-indexing-service-deployment-787c48c4c6-qbnwh         1/1     Running   2 (93s ago)   40h
historian-logging-6f79d84755-fqz58                             1/1     Running   0             40h
historian-rest-api-deployment-66fbd87f5b-9cczl                 1/1     Running   0             40h
historian-webserver-6fb5bc5c79-hrscj                           1/1     Running   0             40h
proficyconfighub-proficyauth-app-service-68f5b59767-hxkn8      1/1     Running   0             40h
proficyconfighub-proficyauth-db-service-77577957bf-sphrr       1/1     Running   0             40h
proficyconfighub-proficyauth-uaa-service-f86dcf667-kp9r7       1/1     Running   0             40h
proficyconfighub-proficyconfighub-container-service-f6946c5s6rk 1/1    Running   0             40h
proficyconfighub-proficyconfighub-webserver-6767998877-tncrx   1/1     Running   0             40h
[cloudshell-user@ip-10-130-18-196 ~]$ 
```

5. To delete any pod, run the following command:

```
Kubectl delete pod <pod-name>
```

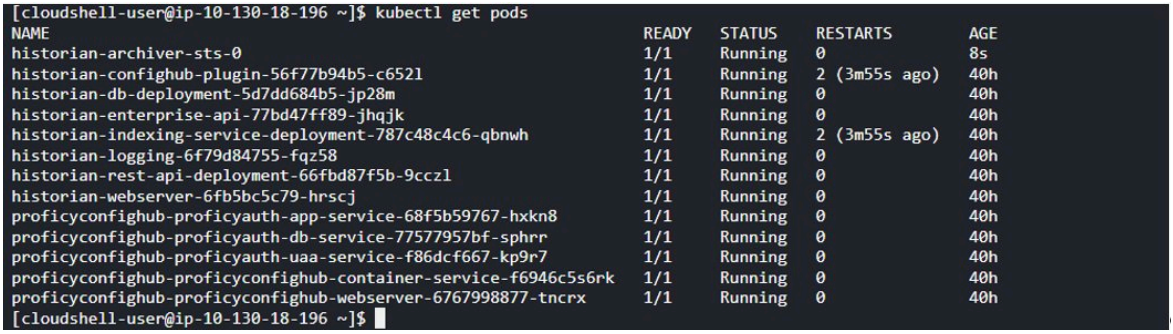
For example:

```
Kubectl delete pod historian-archiver-sts-0
```

6. To verify that pod was deleted, run the following command:

```
Kubectl get pods
```

The output of this command will generate a result similar to the following:



```
[cloudshell-user@ip-10-130-18-196 ~]$ kubectl get pods
NAME                                                              READY   STATUS    RESTARTS        AGE
historian-archiver-sts-0                                          1/1     Running   0               8s
historian-confighub-plugin-56f77b94b5-c652l                       1/1     Running   2 (3m55s ago)   40h
historian-db-deployment-5d7dd684b5-jp28m                          1/1     Running   0               40h
historian-enterprise-api-77bd47ff89-jhqjk                         1/1     Running   0               40h
historian-indexing-service-deployment-787c48c4c6-qbnwh            1/1     Running   2 (3m55s ago)   40h
historian-logging-6f79d84755-fqz58                                1/1     Running   0               40h
historian-rest-api-deployment-66fbd87f5b-9cczl                    1/1     Running   0               40h
historian-webserver-6fb5bc5c79-hrscj                              1/1     Running   0               40h
proficyconfighub-proficyauth-app-service-68f5b59767-hxkn8         1/1     Running   0               40h
proficyconfighub-proficyauth-db-service-77577957bf-sphrr          1/1     Running   0               40h
proficyconfighub-proficyauth-uaa-service-f86dcf667-kp9r7          1/1     Running   0               40h
proficyconfighub-proficyconfighub-container-service-f6946c5s6rk   1/1     Running   0               40h
proficyconfighub-proficyconfighub-webserver-6767998877-tncrx      1/1     Running   0               40h
[cloudshell-user@ip-10-130-18-196 ~]$
```

Note:
The age of the container will be few seconds, as it got restarted after deletion.