



# Plant Applications 8.2

Result Sets



**Proprietary Notice**

The information contained in this publication is believed to be accurate and reliable. However, General Electric Company assumes no responsibilities for any errors, omissions or inaccuracies. Information contained in the publication is subject to change without notice.

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

© 2021, General Electric Company. All rights reserved.

**Trademark Notices**

GE, the GE Monogram, and Predix are either registered trademarks or trademarks of General Electric Company.

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

All other trademarks are the property of their respective owners.

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

[doc@ge.com](mailto:doc@ge.com)

# Table of Contents

|  |    |
|--|----|
| Description of Result Sets.....                                | 1  |
| General Methodology used for Result Sets:.....                 | 2  |
| Techniques used in creating Result Set Stored Procedures:..... | 2  |
| Testing Result Set Stored Procedures .....                     | 2  |
| Tables and Stored Procedures .....                             | 3  |
| Using Result Sets with Stored Procedures .....                 | 4  |
| Example for a Production Event Result Set.....                 | 4  |
| Result Set 1 - Production Events .....                         | 8  |
| Partial Example Code for Using Result Set 1 .....              | 9  |
| Result Set 2 - Variable Values .....                           | 11 |
| Partial Example Code for Using Result Set 2.....               | 11 |
| Result Set 3 - Product Changes .....                           | 13 |
| Partial Example Code for Using Result Set 3.....               | 13 |
| Result Set 4 - Topic Messages .....                            | 15 |
| Result Set 5 - Downtime Events .....                           | 15 |
| Result Set 6 - Alarms .....                                    | 16 |
| Result Set 7 - Sheet Columns.....                              | 17 |
| Result Set 8 - User-Defined Event.....                         | 19 |
| Result Set 9 - Waste Event .....                               | 20 |
| Result Set 10 - Production Event Details.....                  | 22 |
| Result Set 11 - Genealogy Event Components .....               | 23 |
| Result Set 12 - Genealogy Input Events.....                    | 24 |
| Input Event Result Sets Tips .....                             | 24 |
| Result Set 13 - Defect Detail.....                             | 24 |
| Result Set 14 - Historian Write.....                           | 26 |
| Result Set 15 - Production Plan.....                           | 26 |
| Partial Example Code for Using Result Set 15.....              | 27 |
| Result Set 16 - Production Setup.....                          | 32 |
| Result Set 17 - Production Plan Starts .....                   | 32 |
| Result Set 18 - Production Execution Path Unit Starts.....     | 33 |
| Result Set 19 - Production Stats.....                          | 33 |
| Result Set 20 - HistorianRead Result Set.....                  | 34 |

Partial Sample Code for Using Result Set 20..... 35

Result Set 21 - Non-Productive Time ..... 38

Result Set 50 - Output File Creation ..... 39

    Configuration..... 39

    Startup and Shutdown ..... 39

    Communication Protocol..... 39

    Log Files ..... 39

    Result Set 50 Description ..... 39

    Result Set 50 Output File Parameters ..... 40

    Key Parts of a Result Set 50 Example..... 40

Result Set 51 - Return Parameters ..... 42

## Description of Result Sets

Result Sets allow you to add real-time messages to model-based custom SQL stored procedures and to Plant Applications' variable stored procedure calculations. These result sets will allow you to safely update the Plant Applications database and notify the Plant Applications client of the changes made.

---

**Note:** Run the *spServer\_CmnShowResultSets* stored procedure to see what fields are in the result sets.

---

Plant Applications employs a messaging system and result sets allow you to use this messaging system. The Proficy Router service (called Message Bus in earlier versions) acts as the postal service. The Plant Applications Client and Proficy Server both send and receive messages. The Database Manager service handles all database updates. Instead of directly updating the Plant Applications database with SQL **Insert** and **Update** statements in a custom stored procedure, result sets use the Plant Applications messaging infrastructure to make the database changes and allows the clients that are currently running to see these database changes in real time. This method of issuing a message is necessary to notify the client that a change has been made to the database.

The following table lists the result set types available and the appropriate number to use when invoking a result set. All result sets require that the first column is one of the following values in the Type column.

| Type               | Name  | Description  |
|--------------------|---|--|
| <a href="#">1</a>  | <a href="#">Production Events</a>                     | <a href="#">Create, update, and delete event records.</a>                      |
| <a href="#">2</a>  | <a href="#">Variable Values</a>                       | <a href="#">Create, update, and delete Variable data records.</a>              |
| <a href="#">3</a>  | <a href="#">Product Change</a>                        | <a href="#">Create, update, and delete Product Changes.</a>                    |
| <a href="#">5</a>  | <a href="#">Downtime Events</a>                       | <a href="#">Create, update, and delete Downtime records.</a>                   |
| <a href="#">7</a>  | <a href="#">Sheet Columns</a>                         | <a href="#">Create, update, and delete Sheet Column records.</a>               |
| <a href="#">8</a>  | <a href="#">User Defined Events</a>                   | <a href="#">Create, update, and delete User Defined Event records.</a>         |
| <a href="#">9</a>  | <a href="#">Waste Events</a>                          | <a href="#">Create, update, and delete Waste records.</a>                      |
| <a href="#">10</a> | <a href="#">Production Event Details</a>              | <a href="#">Create, update, and delete Production Event Detail records.</a>    |
| <a href="#">11</a> | <a href="#">Production Event Components</a>           | <a href="#">Create, update, and delete Production Event Component records.</a> |
| <a href="#">12</a> | <a href="#">Genealogy Input Events</a>                | <a href="#">Create, update, and delete Genealogy Input Events.</a>             |
| <a href="#">13</a> | <a href="#">Defect Details</a>                        | <a href="#">Create, update, and delete Defect Detail records.</a>              |
| <a href="#">14</a> | <a href="#">Historian Write</a>                       |  |
| <a href="#">15</a> | <a href="#">Production Plan</a>                       |  |
| <a href="#">16</a> | <a href="#">Production Setup</a>                      |  |
| <a href="#">17</a> | <a href="#">Production Plan Starts</a>                |  |
| <a href="#">18</a> | <a href="#">Production Execution Path Unit Starts</a> |  |
| <a href="#">19</a> | <a href="#">Production Stats</a>                      |  |
| <a href="#">20</a> | <a href="#">HistorianRead Result Set</a>              |  |
| <a href="#">21</a> | <a href="#">Non-Productive Time</a>                   | <a href="#">Create, update and delete non-productive time records.</a>         |
| <a href="#">50</a> | <a href="#">Output File Data Export</a>               | <a href="#">Create output files for exporting data.</a>                        |
| <a href="#">51</a> | <a href="#">Return Parameters</a>                     | <a href="#">Allows special parameters to be returned from spLocals</a>         |

All Generic Models that used to trigger a Stored Procedure that can be used for creating Result Sets start with the same 2 parameters as outputs at the beginning of the stored procedure. The first parameter is the Status indicating if the Result Sets should be executed. If this is set to a 1, Result

Sets are executed by the Event Manager Service and the Message Output is logged in the Event Manager log file. If no Message Output is desired, set the message parameter to a blank value. If the first parameter is set to 0, result sets are not performed and the Message Output is logged in the Event Manager log file.

#### General Methodology used for Result Sets:

1. Gather data for the Result Set.
2. Check to see if the record already exists.
3. Select the Result Set data.
4. Set the Return Status Value to a 1 to execute Result Sets or set the Return Status Value to a 0 to ignore Result Sets and to report Errors to the Event Manager Service for Model based Result Sets.

#### Techniques used in creating Result Set Stored Procedures:

- Use SQL temporary tables to hold data.
- Use the right number of columns for the Result Set.
- Set NoCount On to turn off count messages.
- Use Date formatting
- Use a Custom Plant Applications User to aid in troubleshooting problems.

#### Testing Result Set Stored Procedures

- No Stored Procedure should be attached to a model or variable without first running it manually to see if anything comes to the screen unexpectedly. Anything that comes to the screen will appear to be a result set to the caller.
- Test by running the Stored Procedure manually from SQL Server Management Studio. Supply a set of parameter values that match the stored procedure input parameters. Then use the SQL Execute statement to run the stored procedure.
- It is important that you don't forget the @ symbols in front of variables within your stored procedures
- An example would be, `Select X = 7` instead of, `Select @X = 7`. This looks like a result set to the calling code in the Event Manager or Calculation Manager. This `Select X = 7` the calling code would think it has a SheetColumn result set coming and would start accessing columns that are not present.
- Confirm the output in the Management Studio is correct. Make sure there are no extra messages being sent out that may be interpreted as result sets. Sometimes extra `Select` messages may be used to check on specific values while troubleshooting a Result Set Stored Procedure such as:  
`Select @X`  
 By testing it, you may notice that there are `Select` messages that still need to be removed prior to using this Stored Procedure in the production environment.
- When first activating the Stored Procedure in the production environment, check the Event Manager log file for messages if this is a Result Set Stored Procedure driven by a model. If it is driven by a Plant Applications Variable, then check the Calculation Manager log file for messages.
- When first activating the Stored Procedure in the production environment you can put the Database Manager service in Debug Mode and check the log file for additional error messages.

## Tables and Stored Procedures

Most result sets are associated with a system stored procedure and a particular table in the Plant Applications database. When a result set is issued, the end result is a record added, updated, or deleted from its associated table.

The following table lists the result set and the associated stored procedure and table.

| ID | Result Set                 | Database Table                      | Stored Procedure  |
|----|----------------------------|-------------------------------------|---|
| 1  | Production Events          | Events                              | spServer_DBMgrUpdEvent  |
| 2  | Variable Test              | Tests                               | spServer_DBMgrUpdTest2 or<br>spServer_DBMgrUpdTest (legacy)                         |
| 3  | Product Change             | Production_Starts                   | spServer_DBMgrUpdGrade2 or<br>spServer_DBMgrUpdGrade (legacy)                       |
| 5  | Downtime Event             | Timed_Event_Details                 | spServer_DBMgrUpdTimedEvent   |
| 7  | Sheet Columns              | Sheet_Columns                       | spServer_DBMgrUpdColumn2 or<br>spServer_DBMgrUpdColumn (legacy)                     |
| 8  | User Defined Events        | User_Defined_Events                 | spServer_DBMgrUpdUserEvent  |
| 9  | Waste Events               | Waste_Event_Details                 | spServer_DBMgrUpdWasteEvent   |
| 10 | Production Event Details   | Event_Details                       | spServer_DBMgrUpdEventDet   |
| 11 | Production Event Component | Event_Components                    | spServer_DBMgrUpdEventComp  |
| 12 | Genealogy Input Events     | PrdExec_Input_Events                | spServer_DBMgrUpdPEInputEvent   |
| 13 | Defect Details             | Defect_Details                      | spServer_DBMgrUpdDefectDetail   |
| 14 | Historian Write            | none                                | None. Must use a result set.  |
| 15 | Production Plan            | Production_Plan                     | spServer_DBMgrUpdProdPlan   |
| 16 | Production Setup           | Production_Setup                    | spServer_DBMgrUpdProdSetup  |
| 17 | Production Plan Starts     | Production_Plan_Starts              | spServer_DBMgrUpdProdPlanStarts and<br>spServer_DBMgrUpdProdPlanStartsFromPlan      |
| 18 | PrdExec Path Unit Starts   | PrdExec_Path_Unit_Starts            | spServer_DBMgrUpdPrdExecPathUnitStarts<br>and<br>spServer_DBMgrUpdPrdExecPathStarts |
| 19 | Production Stats           | Production_Plan<br>Production_Setup | spServer_DBMgrUpdProdStats  |
| 20 | Historian Read             | none                                | None. Must use a result set.  |
| 21 | Non-Productive Time        |                                     |   |
| 50 | Output File Data Export    | none                                | None. Must use a result set.  |

## Using Result Sets with Stored Procedures

When creating a stored procedure that uses result sets, you should use a standard approach. From a cursory point of view, you should use the following steps:

- Create a table variable for the result set that contains a field for each parameter in the result set. Some fields in the table variable can have default values assigned to them.
- Create a local variable for each field in the table that the result set will update.
- Create additional local variables that will be used for program flow and/or error checking.
- Determine if the result set will add, update, or delete a record.

**NOTE:** In each case you should always check to see if the record exists and handle the situation accordingly. Don't assume a record exists for an update or delete or that the record you will insert is not already in the database.

- Create logic that will assign values to each of the local variables that are associated with the table you are updating.
- Insert a record into the result set table variable using the local variables that are associated with the table being updated.
- Ensure that the message type is a **pre-message** so that the messaging system will inform the Database Manager to update the database.
- Issue the result set using a **Select** statement that selects all of the records from the result set table variable.

### Example for a Production Event Result Set

```
-----  
----  
-- Declare Table Variable for Production Event Result Set(1).  
-----  
----
```

```
DECLARE @rs1ProductionEvent TABLE(  
rs1ResultSetType int DEFAULT 1, -- 00 - Result Set Type (1)  
rs1NotUsed int, -- 01 - Not Used  
rs1TransactionType int, -- 02 - TransactionType  
rs1EventId int, -- 03 - EventId  
rs1EventNum varchar(25), -- 04 - EventNum  
rs1PUId int, -- 05 - PUId  
rs1TimeStamp datetime, -- 06 - TimeStamp  
rs1AppliedProduct int, -- 07 - AppliedProduct  
rs1SourceEvent int, -- 08 - SourceEvent  
rs1EventStatus tinyint, -- 09 - EventStatus  
rs1Confirmed bit, -- 10 - Confirmed  
rs1UserId int DEFAULT 6, -- 11 - UserId  
rs1PostDB int DEFAULT 0, -- 12 - PostDB  
rs1Conformance bit, -- 13 - Conformance  
rs1TestPctComplete tinyint, -- 14 - TestPctComplete  
rs1StartTime datetime, -- 15 - Start Time
```



## Result Sets

```
rs1TransNum int DEFAULT 0, -- 16 - TransNum
rs1TestingStatus int, -- 17 - TestingStatus
rs1CommentId int, -- 18 - CommentId
rs1EventSubTypeId int, -- 19 - EventSubTypeId
rs1EntryOn datetime, -- 20 - EntryOn
rs1ApprovedUserId int, -- 21 - ApprovedUserId
rs1SecondUserId int, -- 22 - SecondUserId
rs1ApprovedReasonId int, -- 23 - ApprovedReasonId
rs1UserReasonId int, -- 24 - UserReasonId
rs1UserSignOffId int, -- 25 - UserSignOffId
rs1ExtendedInfo int) -- 26 - ExtendedInfo
```

```
-----
----
-- Declare Local Variables for Events Table
-----
----
```

```
DECLARE @eEvent_Id int,
@eConfirmed bit,
@eConsumed_Timestamp datetime,
@eTimeStamp datetime,
@eEntry_On datetime,
@eStart_Time datetime,
@eUser_Signoff_Id int,
@eApprover_User_Id int,
@eUser_Reason_Id int,
@eApprover_Reason_Id int,
@eTesting_Status int,
@eEvent_Subtype_Id int,
@eSecond_User_Id int,
@eSource_Event int,
@eUser_Id int,
@eComment_Id int,
@ePU_Id int,
@eApplied_Product int,
@eEvent_Status tinyint,
@eConformance tinyint,
@eTesting_Prct_Complete tinyint,
@eExtended_Info varchar(255),
@eEvent_Num varchar(25)
```

```
-----
----
-- Declare Additional Local Variables.
-----
----
```

```
DECLARE @TransactionType int,
```

```

@PostDB int,
@TransNum int
-----
----
-- Create Logic that determines whether new Event Result is needed or
-- if an existing record needs to be updated or deleted.  Set values of
local
-- variables.
-----
----
-- Add record to @rs1ProudctionEvent Table Variable.
--
-- NOTE: Some fields in following insert statement are commented and
default
-- values for Table Variable are used.  If other values required then
remove
-- comments and assign values using additional local variables.  All fields
-- are included for ease of readability.
-----
----
INSERT INTO @rs1ProudctionEvent (
-- rs1ResultSetType, -- 00 - Result Set Type (1)
-- rs1NotUsed, -- 01 - Not Used
rs1TransactionType, -- 02 - TransactionType
rs1EventId, -- 03 - EventId
rs1EventNum, -- 04 - EventNum
rs1PUIId, -- 05 - PUIId
rs1TimeStamp, -- 06 - TimeStamp
rs1AppliedProduct, -- 07 - AppliedProduct
rs1SourceEvent, -- 08 - SourceEvent
rs1EventStatus, -- 09 - EventStatus
rs1Confirmed, -- 10 - Confirmed
-- rs1UserId, -- 11 - UserId
-- rs1PostDB, -- 12 - PostDB
rs1Conformance, -- 13 - Conformance
rs1TestPctComplete, -- 14 - TestPctComplete
rs1StartTime, -- 15 - Start Time
-- rs1TransNum, -- 16 - TransNum
rs1TestingStatus, -- 17 - TestingStatus
rs1CommentId, -- 18 - CommentId
rs1EventSubTypeId, -- 19 - EventSubTypeId
-- rs1EntryOn, -- 20 - EntryOn
rs1ApprovedUserId, -- 21 - ApprovedUserId
rs1SecondUserId, -- 22 - SecondUserId
rs1ApprovedReasonId, -- 23 - ApprovedReasonId
rs1UserReasonId, -- 24 - UserReasonId

```

## Result Sets

```
rs1UserSignOffId, -- 25 - UserSignOffId
rs1ExtendedInfo) -- 26 - ExtendedInfo
VALUES (
-- rs1ResultSetType, -- 00 - Result Set Type (1)
-- rs1NotUsed, -- 01 - Not Used
@TransactionType, -- 02 - TransactionType
@eEvent_Id, -- 03 - EventId
@eEvent_Num, -- 04 - EventNum
@ePU_Id, -- 05 - PUIId
@eTimeStamp, -- 06 - TimeStamp
@eApplied_Product, -- 07 - AppliedProduct
@eSource_Event, -- 08 - SourceEvent
@eEvent_Status, -- 09 - EventStatus
@eConfirmed, -- 10 - Confirmed
-- rs1UserId, -- 11 - UserId
-- rs1PostDB, -- 12 - PostDB
@eConformance, -- 13 - Conformance
@eTesting_Prct_Complete, -- 14 - TestPctComplete
@eStart_Time, -- 15 - Start Time
-- rs1TransNum, -- 16 - TransNum
@eTesting_Status, -- 17 - TestingStatus
@eComment_Id, -- 18 - CommentId
@eEvent_Subtype_Id, -- 19 - EventSubTypeId
-- rs1EntryOn, -- 20 - EntryOn
@eApprover_User_Id, -- 21 - ApprovedUserId
@eSecond_User_Id, -- 22 - SecondUserId
@eApprover_Reason_Id, -- 23 - ApprovedReasonId
@eUser_Reason_Id, -- 24 - UserReasonId
@eUser_Signoff_Id, -- 25 - UserSignOffId
@eExtended_Info) -- 26 - ExtendedInfo
-----
----
-- Issue Result Set.
-----
----
IF (SELECT Count(*)
FROM @rs1ProudctionEvent
WHERE rs1EventNum IS NOT NULL) > 0
BEGIN
SELECT * FROM @rs1ProudctionEvent
END
```

## Result Set 1 - Production Events

The following table lists the parameters used in a Production Event Result Set.

| Order | Field Name         | Req | Values/Table Reference  |
|-------|--------------------|-----|---|
| 0     | Result Set Type    | X   | 1   |
| 1     | Not Used           |     | Not used.   |
| 2     | Transaction Type   | X   | 1 Add, 2 Update, 3 - Delete   |
| 3     | Event Id           | X   | Events (Event_Id). Not required on updates.   |
| 4     | Event Number       | X   | Events (Event_Num)  |
| 5     | Production Unit Id | X   | Events (PU_Id), Prod_Units (PU_Id)  |
| 6     | Timestamp          | X   | Events.(TimeStamp)  |
| 7     | Applied Product    |     | Events (Applied_Product)  |
| 8     | Source Event       |     | Events (Source_Event)   |
| 9     | Event Status       |     | Events (Event_Status)   |
| 10    | Confirmed          |     |   |
| 11    | User Id            | X   | Events (User_Id), Users (User_Id). Allows a specific User_Id to be used and tracked. If not using a specific User_Id account then set it to 0 and the appropriate Service User_Id will be used. |
| 12    | Update Type        |     | 0 = Pre Update. Database Manager writes to database and sends to client.<br>1 = Post Update. Data already in database and sent to clients only.   |
| 13    | Conformance        |     | 0 - Target<br>1 - Upper/Lower User Limit Exceeded<br>2 - Upper/Lower Warning Limit Exceeded<br>3 - Upper/Lower Reject Exceeded  |
| 14    | TestPctComplete    |     | Events (Testing_Prct_Complete)  |
| 15    | StartTime          |     | Events (Start_Time)   |
| 16    | TransNum           |     | 0 Update fields that are not null to the new values.<br>2 Update all fields of Events to the values in Result Set.  |
| 17    | Testing Status     |     | Events (Testing_Status)<br>1- Normal<br>2 - Skip<br>3 - Reset   |
| 18    | Comment Id         |     | Events (Comment_Id)   |
| 19    | EventSubTypepId    |     | Events (Event_Subtype_Id),<br>Event_Subtypes (Event_Subtype_Id)   |
| 20    | EntryOn            |     | Events (Entry_On)   |
| 21    | ApprovedUserID     |     | User ID of eSig approver  |
| 22    | SecondUserID       |     | User ID of eSig second approver   |
| 23    | ApprovedReasonID   |     | Reason ID of reason code for approval   |
| 24    | UserReasonID       |     | Reason ID for user sign-off   |
| 25    | UserSignOffID      |     | User ID for user sign-off   |
| 26    | ExtendedInfo       |     | Extended information field  |
| 27    | ESignature         |     | Signature ID from the ESignature table  |

## Result Sets

### Partial Example Code for Using Result Set 1

```
--Declare the SQL variables used
Declare
@EUOrderId int,
@EUTransaction_Type int,
@EUEvent_Id int,
@EUEvent_Num varchar(25),
@EUPU_Id int,
@EUTimestamp datetime,
@EUApplied_Product int,
@EUSource_Event int,
@EUEvent_Status int,
@EUConfirmed int,
@EUUser_Id int,
@EUPostUpdate int,
@EUConformance int,
@EUTestPctComplete int,
@EUStartTime datetime,
@EUTransactionNumber int,
@EUTestingStatus int,
@EUComment_Id int,
@EUEvent_Subtype_Id int,
@EUEntry_On datetime
--Create a temporary table to hold all event result set rows
CREATE TABLE #EventUpdates (
    EUId int,
    EUTransaction_Type int,
    EUEvent_Id int NULL,
    EUEvent_Num Varchar(25),
    EUPU_Id int,
    EUTimeStamp varchar(25),
    EUApplied_Product int Null,
    EUSource_Event int Null,
    EUEvent_Status int Null,
    EUConfirmed int Null,
    EUUser_Id int,
    EUPostUpdate int,
    EUConformance int,
    EUTestPctComplete int,
    EUStartTime datetime,
    EUTransactionNumber int,
    EUTestingStatus int,
    EUComment_Id int,
    EUEvent_Subtype_Id int,
```

## Result Set 1 - Production Events

```
    EUEEntry_On datetime)
--Set Result Set Variables to values retrieved in your custom code.  No
specific custom code is being show.
Select @EUOrderId
Select @EUTransaction_Type = 1
Select @EUEvent_Id = Null
Select @EUEvent_Num = @NewEvent_Num
Select @EUPU_Id = @PU_Id
Select @EUTimestamp = @EventTimestamp
Select @EUApplied_Product = NULL
Select @EUSource_Event = NULL
Select @EUEvent_Status = ProdStatus_Id from Production_Status
    Where ProdStatus_Desc = 'Active'
Select @EUConfirmed = 1
Select @EUUser_Id = Null
Select @EUPostUpdate = 0
Select @EUConformance = Null
Select @EUTestPctComplete = Null
Select @EUStartTime = @EventTimestamp
Select @EUTransactionNumber = Null
Select @EUTestingStatus = Null
Select @EUComment_Id = Null
Select @EUEvent_Subtype_Id = Null
Select @EUEEntry_On = Getdate()

Insert into #EventUpdates
(EUId,EUTransaction_Type,EUEvent_Id,EUEvent_Num,EUPU_Id,EUTimeStamp,EUAppli
ed_Product,EUSource_Event,EUEvent_Status,EUConfirmed,EUUser_Id,EUPostUpdate
,EUConformance,
EUTestPctComplete,EUStartTime,EUTransactionNumber,EUTestingStatus,
EUComment_Id, EUEvent_Subtype_Id,EUEEntry_On)
Values(@EUOrderId,@EUTransaction_Type,@EUId,@NewEvent_Num,@PU_Id,@EventTime
stamp,@EUApplied_Product,@EUSource_Event,@EUEvent_Status,@EUConfirmed,@EUUs
er_Id,@EUPostUpdate,@EUConformance,@
EUTestPctComplete,@EUStartTime,@EUTransactionNumber,@EUTestingStatus,@EUCom
ment_Id,@EUEvent_Subtype_Id,@EUEEntry_On)
--Send out Result Set Message back to the calling Service
If (Select Count(*) From #EventUpdates) > 0
    Select 1,* From #EventUpdates
Select @ReturnStatus = 1
Return
```

## Result Set 2 - Variable Values

The following table lists the parameters used in a Variable Result Set.

| Order | Field Name         | Req | Values/Table Reference   |
|-------|--------------------|-----|--|
| 0     | Result Set Type    | X   | 2  |
| 1     | Variable Id        | X   | Tests (Var_Id)   |
| 2     | Production Unit Id | X   | Variables (PU_Id)<br>Prod_Units (PU_Id)  |
| 3     | User Id            | X   | Tests (User_Id)<br>Users (User_Id). Allows a specific User_Id to be used and tracked. If not using a specific User_Id account then set it to 0 and the appropriate Service User_Id will be used. |
| 4     | Cancelled          | X   | 0 – False<br>1 – True<br>If set to True then it is treated like a delete.  |
| 5     | Value              | X   | Tests (Result)   |
| 6     | Timestamp          | X   | Tests (Result_On)  |
| 7     | Transaction Type   |     | 1 – Add, 2 – Update, 3 - Delete  |
| 8     | Update Type        |     | 0 – Pre Update. Database Manager writes to db and sends to client.<br>1 – Post Update. Data already in db and sent to clients only.<br><b>NOTE:</b> PostDB messages do not support comment IDs.  |
| 9     | SecondUserId       |     |  |
| 10    | TransNum           |     | 0 = Coalesce all input values with values from database before updating<br>2 = Update database using only input values to this stored procedure  |
| 11    | EventId            |     | Events (Event_Id)  |
| 12    | ArrayId            |     | Array_Data (Array_Id)  |
| 13    | CommentId          |     | Tests (Comment_Id)   |
| 14    | ESignature         |     | Signature ID from the ESignature table   |

### Partial Example Code for Using Result Set 2

```
--Declare the SQL variables used
Declare
@VUVar_Id int,
@VUPU_Id int,
@VUUser_Id int,
@VUCanceled int,
@VUResult Varchar(25),
@VUResult_On varchar(25),
@VUTransaction_Type int,
@VUPostUpdate int
--Create a temporary table to hold all variable result set rows
CREATE TABLE #VariableUpdates (
    VUVar_Id int,
    VUPU_Id int,
```

## Result Set 2 - Variable Values

```
VUUser_Id int,  
VUCanceled int,  
VUResult Varchar(25),  
VUResult_On datetime,  
VUTransaction_Type int,  
VUPostUpdate int)  
--Set Result Set Variables to values retrieved in your custom code. No  
specific custom code is being show.  
Select @VUVar_Id = Var_Id  
    from Variables where PU_Id = @PU_Id And Var_Desc Like '%Remaining%'  
Select @VUPU_Id = @PU_Id  
Select @VUUser_Id = 0  
Select @VUCanceled = 0  
Select @VUResult = @Value  
Select @VUResult_On = @Timestamp  
Select @VUTransaction_Type = 1  
Select @VUPostUpdate = 0  
  
Insert into #VariableUpdates  
(VUVar_Id,VUPU_Id,VUUser_Id,VUCanceled,VUResult,VUResult_On,VUTransaction_T  
ype,VUPostUpdate)  
Values(@VUVar_Id,@VUPU_Id,@VUUser_Id,@VUCanceled,@VUResult,@VUResult_On,@VU  
Transaction_Type,@VUPostUpdate)  
  
If (Select Count(*) From #VariableUpdates) > 0  
    Select 2,* from #VariableUpdates
```



## Result Set 3 - Product Changes

The following table lists the parameters used in a Product Change Result Set.

| Order | Field Name         | Req | Values/Table Reference  |
|-------|--------------------|-----|---|
| 0     | Result Set Type    | X   | 3   |
| 1     | Start Id           | X   | Production_Starts (Start_Id)  |
| 2     | Production Unit Id | X   | Production_Starts (PU_Id)<br>Prod_Units (PU_Id)   |
| 3     | Product Id         | X   | Production_Starts (Prod_Id)<br>Products (Prod_Id)   |
| 4     | Start Time         | X   | Production_Starts (Start_Time)  |
| 5     | Update Type        | X   | 0 – Pre Update. Database Manager writes to database and sends to client.<br>1 – Post Update. Data already in database and sent to clients only.   |
| 6     | User Id            |     | Production_Starts (User_Id)<br>Users (User_Id) Allows a specific User_Id to be used and tracked. If not using a specific User_Id account then set it to 0 and the appropriate Service User_Id will be used. |
| 7     | Second User Id     |     |   |
| 8     | Transaction Type   |     |   |
| 9     | ESignature         |     | Signature ID from the ESignature table  |

### Partial Example Code for Using Result Set 3

```
--Declare the SQL variables used
Declare
@CurrentProd_Id int,
@NewProd_Id int,
@NewPCTime datetime,
@PCStart_Id int,
@PCPU_Id int,
@PCProd_Id int,
@PCStart_Time datetime,
@PCPostUpdate int
-----Product Change Logic-----
--Retrieve Current Product being run on a Unit
Select @CurrentProd_Id = Prod_Id From Production_Starts
    Where (PU_Id = @PU_Id) And
        (Start_Time < @EventTimestamp) And
        ((End_Time >= @EventTimestamp) Or (End_Time Is Null))

--Check to see if the Product is new
If @NewProd_Id <> @CurrentProd_Id
Begin
    Select @PrevEventTimeStamp = Timestamp
```

## Result Set 3 - Product Changes

```
    From Events Where (PU_Id = @PU_Id) And (Timestamp = (Select
Max(Timestamp) From Events
    Where (PU_Id = @PU_Id) and (Timestamp < @EventTimestamp)))
--Set the Product Change Time to 1 Minute after the Previous Event Time
Select @NewPCTime = DateAdd(mi,1,@PrevEventTimeStamp)

Create Table #ProductChange(
    PCStart_Id Int Null,
    PCPU_Id      Int,
    PCProd_Id   Int,
    PCStart_Time DateTime,
    PCPostUpdate Int Null)

Select @PCStart_Id = Null
Select @PCPU_Id = @PU_Id
Select @PCProd_Id = @NewProd_Id
Select @PCStart_Time = @NewPCTime
Select @PCPostUpdate = 0

Insert into #ProductChange
(PCStart_Id,PCPU_Id,PCProd_Id,PCStart_Time,PCPostUpdate)
    Values (@PCStart_Id,@PCPU_Id,@PCProd_Id,@PCStart_Time,@PCPostUpdate)

Select 3,* From #ProductChange

Drop Table #ProductChange
End
```

- 
- *Full code samples can be downloaded from the [support site](#).*
-

## Result Set 4 - Topic Messages

This result set is used for sending out topic messages. The following table lists the parameters used in a Topic Result Set.

| Order | Field Name      | Req | Values/Table Reference |
|-------|-----------------|-----|------------------------|
| 0     | Result Set Type | X   | 4                      |
| 1     | Topic           | X   |                        |
| 2     | Key             | X   |                        |

## Result Set 5 - Downtime Events

The following table lists the parameters used in a Downtime Event Result Set.

| Order | Field Name         | Req | Values/Table Reference  |
|-------|--------------------|-----|---|
| 0     | Result Set Type    | X   | 5   |
| 1     | Production Unit Id | X   | Timed_Event_Details.PU_Id,<br>Prod_Units.PU_Id  |
| 2     | Location           | X   | Timed_Event_Details.Source_PU_Id,<br>Prod_Units.PU_Id   |
| 3     | Status Id          | X   | Timed_Event_Details.TEStatus_Id,<br>Timed_Event_Status.TEStatus_Id  |
| 4     | Fault Id           |     | Timed_Event_Details.TEFault_Id,<br>Timed_Event_Faults.TEFault_Id  |
| 5     | Reason 1 Id        |     | Timed_Event_Details.Reason_Level1,<br>Event_Reasons.Reason_Id   |
| 6     | Reason 2 Id        |     | Timed_Event_Details.Reason_Level2,<br>Event_Reasons.Reason_Id   |
| 7     | Reason 3 Id        |     | Timed_Event_Details.Reason_Level3,<br>Event_Reasons.Reason_Id   |
| 8     | Reason 4 Id        |     | Timed_Event_Details.Reason_Level4,<br>Event_Reasons.Reason_Id   |
| 9     | Production Rate    |     | Timed_Event_Details (Production_Rate)   |
| 10    | Duration           |     | Timed_Event_Details (Duration) This is a<br>calculated (end_time – start_time) column in<br>the database. It is not used by the SP.                   |
| 11    | Transaction Type   | X   | 1 = Add<br>2 = Update<br>3 = Delete<br>4 = Close  |
| 12    | Start Time         | X   | Timed_Event_Details.Start_Time  |
| 13    | End time           |     | Timed_Event_Details.End_Time  |
| 14    | Downtime Event Id  |     | Timed_Event_Details.TEDet_Id  |
| 15    | Update Type        |     | 0 – Pre Update. Database Manager writes to<br>db and sends to client.<br>1 – Post Update. Data already in db and sent<br>to clients only.             |
| 16    | TransNum           |     | 0 = Coalesce all input values with values from<br>database before updating<br>2 = Update database using only input values to<br>this stored procedure |
| 17    | Action1            |     | Timed_Event_Details.Action_Level1,<br>Event_Reasons.Reason_Id   |

|    |                   |  |  |
|----|-------------------|--|--|
| 18 | Action2           |  | Timed_Event_Details.Action_Level2,<br>Event_Reasons.Reason_Id  |
| 19 | Action3           |  | Timed_Event_Details.Action_Level3,<br>Event_Reasons.Reason_Id  |
| 20 | Action4           |  | Timed_Event_Details.Action_Level4,<br>Event_Reasons.Reason_Id  |
| 21 | ActionCommentId   |  | Timed_Event_Details.Action_Comment_Id,<br>Comments.Comment_Id  |
| 22 | ResearchStatusId  |  | Timed_Event_Details.Research_Status_Id   |
| 23 | ResearchCommentId |  | Timed_Event_Details.Research_Comment_Id,<br>Comments.Comment_Id                                      |
| 24 | ResearchOpenDate  |  | Timed_Event_Details.Research_Open_Date   |
| 25 | ResearchCloseDate |  | Timed_Event_Details.Research_Close_Date  |
| 26 | CommentId         |  | Timed_Event_Details.Cause_Comment_Id,<br>Comments.Comment_Id   |
| 27 | TargetProdRate    |  | Not Used   |
| 28 | DimensionX1       |  | Not Used   |
| 29 | DimensionX2       |  | Not Used   |
| 30 | DimensionY1       |  | Not Used   |
| 31 | DimensionY2       |  | Not Used   |
| 32 | DimensionZ1       |  | Not Used   |
| 33 | DimensionZ2       |  | Not Used   |
| 34 | ResearchUserId    |  | Timed_Event_Details.Research_User_Id   |
| 35 | RsnTreeDataId     |  | Identifies what Node on the Reason Tree this<br>Downtime event corresponds to. Used in<br>reporting. |
| 36 | ESignature        |  | Signature ID from the ESignature table   |

## Result Set 6 - Alarms

This result set is no longer used.

The following table lists the parameters used in the Alarm Result Set.

| Order | Field Name                | Req | Values/Table Reference  |
|-------|---------------------------|-----|---|
| 0     | Result Set Type           | X   | 6   |
| 1     | Update Type               | X   | 0 - Post-Update. Database Manager writes to<br>database and sends to client.<br>1 - Pre-Update. Data already in database and<br>sent to clients only. |
| 2     | Transaction Number        | X   | 0 – Update fields that are not null to the new<br>values.<br>2 – Update all fields of Alarms to the values in<br>Result Set.                          |
| 3     | Alarm Id                  | X   | Alarms (Alarm_Id)   |
| 4     | Alarm Template<br>Data Id | X   | Alarm_Template_Var_Data (ATD_Id)  |
| 5     | Start Time                | X   | Alarms (Start_Time)   |
| 6     | End Time                  |     | Alarms (End_Time)   |
| 7     | Duration                  |     | Alarms (Duration)   |
| 8     | Acknowledged              |     | Alarms (Ack)  |
| 9     | Ack Timestamp             |     | Alarms (Ack_On)   |
| 10    | Acknowledged By           |     | Alarms (Ack_By)   |
| 11    | Starting Value            |     | Alarms (Start_Result)   |
| 12    | Ending Value              |     | Alarms (End_Result)   |

## Result Sets

|    |                         |  |   |
|----|-------------------------|--|---|
| 13 | Minimum Value           |  | Alarms (Min_Result)                                 |
| 14 | Maximum Value           |  | Alarms (Max_Result)                                 |
| 15 | Cause 1                 |  | Alarms (Cause1), Event_Reasons (Reason_Id)          |
| 16 | Cause 2                 |  | Alarms (Cause2), Event_Reasons (Reason_Id)          |
| 17 | Cause 3                 |  | Alarms (Cause3), Event_Reasons (Reason_Id)          |
| 18 | Cause 4                 |  | Alarms (Cause4), Event_Reasons (Reason_Id)          |
| 19 | Cause Comment Id        |  | Alarms (Cause_Comment_Id), Comments (Comment_Id)    |
| 20 | Action 1                |  | Alarms (Action1), Event_Reasons (Reason_Id)         |
| 21 | Action 2                |  | Alarms (Action1), Event_Reasons (Reason_Id)         |
| 22 | Action 3                |  | Alarms (Action1), Event_Reasons (Reason_Id)         |
| 23 | Action 4                |  | Alarms (Action1), Event_Reasons (Reason_Id)         |
| 24 | Action Comment Id       |  | Alarms (Action_Comment_Id), Comments (Comment_Id)   |
| 25 | Research User Id        |  | Alarms (Research_User_Id), Users (User_Id)          |
| 26 | Research Status Id      |  | Alarms (Research_Status_Id)                         |
| 27 | Research Open Date      |  | Alarms (Research_Open_Date)                         |
| 28 | Research Close Date     |  | Alarms (Research_Close_Date)                        |
| 29 | Research Comment Id     |  | Alarms (Research_Comment_Id), Comments (Comment_Id) |
| 30 | Source PU Id            |  | Alarms (Source_PU_Id), Prod_Units (PU_Id)           |
| 31 | Alarm Type Id           |  | Alarms (Alarm_Type_Id), Alarm_Types (Alarm_Type_Id) |
| 32 | Key Id                  |  |   |
| 33 | Alarm Description       |  | Alarms (Alarm_Desc)                                 |
| 34 | Transaction Type        |  | 1 = Add<br>2 = Update<br>3 = Delete                 |
| 35 | Template Var Comment Id |  | Alarm_Templates (Comment_Id), Comments (Comment_Id) |
| 36 | Alarm Priority Id       |  | Alarm_Templates (AP_Id), Alarm_Priorities (AP_Id)   |
| 37 | Alarm Template Id       |  | Alarm_Templates (AT_Id)                             |
| 38 | Variable Comment Id     |  | Variables (Comment_Id), Comments (Comment_Id)       |
| 39 | Cutoff                  |  | Alarms (Cutoff)                                     |
| 40 | ESignature              |  | Signature ID from the ESignature table              |

## Result Set 7 - Sheet Columns

The following table lists the parameters used in a Sheet Columns Result Set.

| Order | Field Name       | Req | Values/Table Reference   |
|-------|------------------|-----|--|
| 0     | Result Set Type  | X   | 7  |
| 1     | Sheet Id         | X   | Sheet_Columns (Sheet_Id)<br>Sheets (Sheet_Id)  |
| 2     | User Id          | X   | Sheet_Columns (User_Id)<br>Users (User_Id). Allows a specific User_Id to be used and tracked. If not using a specific User_Id account then set it to 0 and the appropriate Service User_Id will be used. |
| 3     | Transaction Type | X   | 1 = Add<br>2 = Update<br>3 = Delete  |

|    |                    |   |   |
|----|--------------------|---|---|
| 4  | Timestamp          | X | Sheet_Columns (Result On)   |
| 5  | Update Type        | X | 0 – Pre Update. Database Manager writes to db and sends to client.<br>1 – Post Update. Data already in db and sent to clients only. |
| 6  | Approved User Id   |   | Sheet_Columns.Approved_User_Id,<br>Users.User_Id  |
| 7  | Approved Reason Id |   | Sheet_Columns.Approved_Reason_Id,<br>Reasons.Reason_Id  |
| 8  | User Reason Id     |   | Sheet_Columns.User_Reason_Id,<br>Reasons.Reason_Id  |
| 9  | User Sign Off Id   |   | Sheet_Columns.User_Signoff_Id,<br>Users.User_Id   |
| 10 | ESignature         |   | Signature ID from the ESignature table  |

## Result Set 8 - User-Defined Event

The following table lists the parameters used in a User-Defined Event Result Set.

| Order | Field Name            | Req | Values/Table Reference  |
|-------|-----------------------|-----|---|
| 0     | Result Set Type       | X   | 8   |
| 1     | Update Type           | X   | 0 - Post-Update. Data already in db and sent to clients only.<br>1 - Pre-Update. Database Manager writes to db and sends to client. |
| 2     | User Defined Event Id | X   | User_Defined_Events (UDE_Id)  |
| 3     | UD Event Number       | X   | User_Defined_Events (UDE_Desc)  |
| 4     | Unit Id               | X   | User_Defined_Events (PU_Id), Prod_Units (PU_Id)   |
| 5     | Event Subtype Id      | X   | User_Defined_Events (Event_Subtype_Id)  |
| 6     | Start Time            | X   | User_Defined_Events (Start_Time)  |
| 7     | End Time              |     | User_Defined_Events (End_Time)  |
| 8     | Duration              |     | User_Defined_Events (Duration)  |
| 9     | Acknowledged          |     | User_Defined_Events (Ack)   |
| 10    | Ack Timestamp         |     | User_Defined_Events (Ack_On)  |
| 11    | Acknowledged By       |     | User_Defined_Events (Ack_By)  |
| 12    | Cause 1               |     | User_Defined_Events (Cause1),<br>Event_Reasons (Reason_Id)  |
| 13    | Cause 2               |     | User_Defined_Events (Cause2),<br>Event_Reasons (Reason_Id)  |
| 14    | Cause 3               |     | User_Defined_Events (Cause3),<br>Event_Reasons (Reason_Id)  |
| 15    | Cause 4               |     | User_Defined_Events (Cause4),<br>Event_Reasons (Reason_Id)  |
| 16    | Cause Comment Id      |     | User_Defined_Events (Cause_Comment_Id),<br>Comments (Comment_Id)  |
| 17    | Action 1              |     | User_Defined_Events (Action1),<br>Event_Reasons (Reason_Id)   |
| 18    | Action 2              |     | User_Defined_Events (Action1),<br>Event_Reasons (Reason_Id)   |
| 19    | Action 3              |     | User_Defined_Events (Action1),<br>Event_Reasons (Reason_Id)   |
| 20    | Action 4              |     | User_Defined_Events (Action1),<br>Event_Reasons (Reason_Id)   |
| 21    | Action Comment Id     |     | User_Defined_Events (Action_Comment_Id),<br>Comments (Comment_Id)   |
| 22    | Research User Id      |     | User_Defined_Events (Research_User_Id),<br>Users (User_Id)  |
| 23    | Research Status Id    |     | User_Defined_Events (Research_Status_Id)  |
| 24    | Research Open Date    |     | User_Defined_Events (Research_Open_Date)  |
| 25    | Research Close Date   |     | User_Defined_Events (Research_Close_Date)   |
| 26    | Research Comment Id   |     | User_Defined_Events<br>(Research_Comment_Id),<br>Comments (Comment_Id)  |
| 27    | UD Event Comment Id   |     | User_Defined_Events (Comment_Id)  |

|    |                     |  |   |
|----|---------------------|--|---|
| 28 | Transaction Type    |  | 1 = Add<br>2 = Update<br>3 = Delete   |
| 29 | Event Sub Type Desc |  | Event_Subtypes (Event_Subtype_Desc)   |
| 30 | Transaction Number  |  | 0 – Update fields that are not null to the new values.<br>2 – Update all fields of User_Defined_Events to the values in Result Set. |
| 31 | User Id             |  | User_Defined_Events (User_Id)<br>Users (User_Id)  |
| 32 | ESignature          |  | Signature ID from the ESignature table  |

## Result Set 9 - Waste Event

The following table lists the parameters used in a Waste Event Result Set.

| Order | Field Name      | Req | Values/Table Reference  |
|-------|-----------------|-----|---|
| 0     | Result Set Type | X   | 9   |
| 1     | PreDB           | X   | The update type.<br>0 - Post-Update. Data already in database and sent to clients only.<br>1 - Pre-Update. Database Manager writes to database and sends to client. |
| 2     | TransNum        | X   | The transaction number.<br>0 - Update fields that are not null to the new values.<br>2 - Update all fields of Event_Details to the values in Result Set.            |
| 3     | UserId          | X   | The user ID.<br>Waste_Event_Details (User_Id)<br>Users (User_Id)  |
| 4     | TransType       | X   | The transaction type.<br>1 = Add<br>2 = Update<br>3 = Delete  |
| 5     | WasteEventId    | X   | Waste_Event_Details (WED_Id)  |
| 6     | PUId            | X   | Waste_Event_Details (PU_Id)<br>Prod_Units (PU_Id)   |
| 7     | SourcePUId      | X   | The location.<br>Waste_Event_Details (Source_PU_Id)<br>Prod_Units (PU_Id)   |
| 8     | TypeId          |     | Waste_Event_Details (WET_Id)<br>Waste_Event_Types (WET_Id)  |
| 9     | MeasId          |     | Waste_Event_Details (WEMT_Id)<br>Waste_Event_Meas (WEMT_Id)   |
| 10    | Reason1         |     | Waste_Event_Details (Reason1)<br>Event_Reasons (Reason_Id)  |
| 11    | Reason2         |     | Waste_Event_Details (Reason2)<br>Event_Reasons (Reason_Id)  |
| 12    | Reason3         |     | Waste_Event_Details (Reason3)<br>Event_Reasons (Reason_Id)  |
| 13    | Reason4         | X   | Waste_Event_Details (Reason4)<br>Event_Reasons (Reason_Id)  |



## Result Sets

|    |                   |   |   |
|----|-------------------|---|---|
| 14 | EventId           | X | Waste_Event_Details (Event_Id)<br>Events (Event_Id)   |
| 15 | Amount            |   | Waste_Event_Details (Amount)  |
| 16 | Marker1           |   | Waste_Event_Details (Marker1)   |
| 17 | Marker2           |   | Waste_Event_Details (Marker2)   |
| 18 | Timestamp         |   | Waste_Event_Details (TimeStamp)   |
| 19 | Action1           |   | Waste_Event_Details (Action1)<br>Event_Reasons (Reason_Id)  |
| 20 | Action2           |   | Waste_Event_Details (Action1)<br>Event_Reasons (Reason_Id)  |
| 21 | Action3           |   | Waste_Event_Details (Action1)<br>Event_Reasons (Reason_Id)  |
| 22 | Action4           |   | Waste_Event_Details (Action1)<br>Event_Reasons (Reason_Id)  |
| 23 | ActionCommentId   |   | Waste_Event_Details (Action_Comment_Id)<br>Comments (Comment_Id)  |
| 24 | ResearchCommentId |   | Waste_Event_Details (Research_Comment_Id)<br>Comments (Comment_Id)  |
| 25 | ResearchStatusId  |   | Waste_Event_Details (Research_Status_Id)  |
| 26 | ResearchOpenDate  |   | Waste_Event_Details (Research_Open_Date)  |
| 27 | ResearchCloseDate |   | Waste_Event_Details (Research_Close_Date)   |
| 28 | CommentId         |   | Waste_n_Timed_Comments (WTC_Id)   |
| 29 | TargetProdRate    |   | Waste_Event_Details (Target_Prod_Rate)  |
| 30 | ResearchUserId    |   | Waste_Event_Details (Research_User_Id)<br>Users (User_Id)   |
| 31 | FaultId           |   | Waste_Event_Details (Fault_Id), Waste_Event_Fault<br>(Fault_Id)   |
| 32 | RsnTreeDataId     |   | Waste_Event_Details (Event_Reason_Tree_Data_Id),<br>Event_Reason_Tree_Data<br>(Event_Reason_Tree_Data_Id) |
| 33 | DimensionX        |   |   |
| 34 | DimensionY        |   |   |
| 35 | DimensionZ        |   |   |
| 36 | DimensionA        |   |   |
| 37 | StartCoordinateX  |   |   |
| 38 | StartCoordinateY  |   |   |
| 39 | StartCoordinateZ  |   |   |
| 40 | StartCoordinateA  |   |   |
| 41 | General1          |   |   |
| 42 | General2          |   |   |
| 43 | General3          |   |   |
| 44 | General4          |   |   |
| 45 | General5          |   |   |
| 46 | OrderNum          |   |   |
| 47 | ESignature        |   | Signature ID from the ESignature table  |

## Result Set 10 - Production Event Details

The following table lists the parameters used in an Event Details Result Set.

| Order | Field Name                      | Req | Values/Table Reference   |
|-------|---------------------------------|-----|--|
| 0     | Result Set Type                 | X   | 10   |
| 1     | Update Type                     | X   | 0 - Post-Update. Data already in database and sent to clients only.<br>1 - Pre-Update. Database Manager writes to database and sends to client.  |
| 2     | User Id                         | X   | Event_Details (User_Id), Users (User_Id). Allows a specific User_Id to be used and tracked. If not using a specific User_Id account then set it to 0 and the appropriate Service User_Id will be used. |
| 3     | Transaction Type                | X   | 1 Add, 2 Update, 3 - Delete  |
| 4     | Transaction Number              | X   | 0 Update fields that are not null to the new values.<br>2 Update all fields of Event_Details to the values in Result Set. Values of 0 for dimensions are set to null.                                  |
| 5     | Event Id                        | X   | Event_Details (Event_Id)<br>Events (Event_Id)  |
| 6     | Unit Id                         | X   | Prod_Units (PU_Id)   |
| 7     | Primary Event Number            | X   | Event_Details (Primary_Event_Num)  |
| 8     | Alternate Event Number          |     | Event_Details (Alternate_Event_Num)  |
| 9     | Comment Id                      |     | Event_Details (Comment_Id)<br>Comments (Comment_Id).   |
| 10    | Event Sub Type Id               |     | Event_Subtypes (Event_Subtype_Id)  |
| 11    | Original Product                |     | Event_Details (Original_Product)<br>Products (Prod_Id)   |
| 12    | Applied Product                 |     | Event_Details (Applied_Product)<br>Products (Prod_Id)  |
| 13    | Event Status                    | X   | Event_Details (Event_Status)<br>Production_Status (ProdStatus_Id)  |
| 14    | Timestamp                       | X   | Event_Details (TimeStamp)  |
| 15    | Entry On                        |     | Event_Details (Entry_On)   |
| 16    | Production Plan Setup Detail Id |     | Event_Details (PP_Setup_Detail_Id)<br>Production_Setup_Detail (PP_Setup_Detail_Id)   |
| 17    | Shipment Item Id                |     | Event_Details (Shipment_Item_Id)<br>Shipment_Line_Items (Shipment_Item_Id)   |
| 18    | Order Id                        |     | Event_Details (Order_Id)<br>Customer_Orders (Order_Id)   |
| 19    | Order_Line_Id                   |     | Event_Details (Order_Line_Id)<br>Customer_Order_Line_Items (Order_Line_Id)   |
| 20    | Production Plan Id              |     | Event_Details.PP_Id<br>Production_Plan (PP_Id)   |
| 21    | Initial_Dimension X             |     | Event_Details (Initial_Dimension_X)  |
| 22    | Initial_Dimension Y             |     | Event_Details (Initial_Dimension_Y)  |
| 23    | Initial_Dimension Z             |     | Event_Details (Initial_Dimension_Z)  |
| 24    | Initial_Dimension A             |     | Event_Details (Initial_Dimension_A)  |
| 25    | Final_Dimension X               |     | Event_Details (Final_Dimension_X)  |

## Result Sets

|    |                   |  |  |
|----|-------------------|--|--|
| 26 | Final_Dimension Y |  | Event_Details (Final_Dimension_Y)      |
| 27 | Final_Dimension Z |  | Event_Details (Final_Dimension_Z)      |
| 28 | Final_Dimension A |  | Event_Details (Final_Dimension_A)      |
| 29 | Orientation X     |  | Event_Details (Orientation_X)          |
| 30 | Orientation Y     |  | Event_Details. (Orientation_Y)         |
| 31 | Orientation Z     |  | Event_Details (Orientation_Z)          |
| 32 | ESignature        |  | Signature ID from the ESignature table |

## Result Set 11 - Genealogy Event Components

The following table lists the parameters used in a Genealogy Event Components Result Set.

| Order | Field Name                     | Req | Values/Table Reference  |
|-------|--------------------------------|-----|---|
| 0     | Result Set Type                | X   | 11  |
| 1     | Update Type                    | X   | 0 - Post-Update. Data already in database and sent to clients only.<br>1 - Pre-Update. Database Manager writes to database and sends to client.   |
| 2     | User Id                        | X   | Event_Components (User_Id)<br>Users (User_Id). Allows a specific User_Id to be used and tracked. If not using a specific User_Id account then set it to 0 and the appropriate Service User_Id will be used. |
| 3     | Transaction Type               | X   | 1 = Add<br>2 = Update<br>3 = Delete   |
| 4     | Transaction Number             | X   | 0 Update fields that are not null to the new values.<br>2 Update all fields of Event_Details to the values in Result Set. Values of 0 for dimensions are set to null.                                       |
| 5     | Component Id                   | X*  | Event_Components (Component_Id)   |
| 6     | Event Id                       | X   | Event_Components (Event_Id)<br>Events (Event_Id)  |
| 7     | Source Event Id                | X   | Event_Components (Source_Event_Id)<br>Events (Event_Id)   |
| 8     | Dimension X                    |     | Event_Components (Dimension_X)  |
| 9     | Dimension Y                    |     | Event_Components (Dimension_Y)  |
| 10    | Dimension Z                    |     | Event_Components (Dimension_Z)  |
| 11    | Dimension A                    |     | Event_Components (Dimension_A)  |
| 12    | Start Coordinate X             |     | Event_Components (Start_Coordinate_X)   |
| 13    | Start Coordinate Y             |     | Event_Components (Start_Coordinate_Y)   |
| 14    | Start Coordinate Z             |     | Event_Components (Start_Coordinate_Z)   |
| 15    | Start Coordinate A             |     | Event_Components (Start_Coordinate_A)   |
| 16    | Start Time                     |     | Event_Components (Start_Time)   |
| 17    | Timestamp                      |     | Event_Components (Timestamp)  |
| 18    | Parent Component Id            |     | Event_Components (Parent_Component_Id)  |
| 19    | Entry On                       |     | Event_Components (Entry_On)   |
| 20    | Extended Info                  |     | Event_Components (Extended_Info)  |
| 21    | Production Excecution Input ID |     | Event_Components (PEI_Id)   |
| 22    | ReportAsConsumption            |     | Event_Components (Report_As_Consumption)  |
| 23    | Child Unit ID                  |     |   |
| 24    | ESignature                     |     | Signature ID from the ESignature table  |

\*Component Id is a required field only when performing an Update or Delete (Transaction Type 2 or 3). When performing a Transaction Type of 1 to add a new record, the Component Id is not required and is generated by the system.

## Result Set 12 - Genealogy Input Events

The following table lists the parameters used in a Genealogy Input Events Result Set.

| Order | Field Name                         | Req | Values/Table Reference  |
|-------|------------------------------------|-----|---|
| 0     | Result Set Type                    | X   | 12  |
| 1     | Update Type                        | X   | 0 - Post-Update. Data already in database and sent to clients only.<br>1 - Pre-Update. Database Manager writes to database and sends to client.                           |
| 2     | User Id                            | X   | PrdExec_Input_Event (User_Id)<br>Users (User_Id)  |
| 3     | Transaction Type                   | X   | 1 = Add<br>2 = Update<br>3 = Delete   |
| 4     | Transaction Number                 | X   | 0 – Update fields that are not null to the new values.<br>2 – Update all fields of Event_Details to the values in Result Set. Values of 0 for dimensions are set to null. |
| 5     | Timestamp                          | X   | PrdExec_Input_Event (TimeStamp)   |
| 6     | Entry On                           |     | PrdExec_Input_Event (Entry_On)  |
| 7     | Comment Id                         |     | PrdExec_Input_Event (Comment_Id)<br>Comments (Comment_Id)   |
| 8     | Production Event Input Id          |     | PrdExec_Input_Event (PEI_Id)<br>PrdExec_Inputs (PEI_Id)   |
| 9     | Production Event Input Position Id |     | PrdExec_Input_Event (PEIP_Id)<br>PrdExec_Input_Positions (PEIP_Id)  |
| 10    | Event Id                           |     | PrdExec_Input_Event (Event_Id)<br>Events (Event_Id)   |
| 11    | Dimension X                        |     | PrdExec_Input_Event (Dimension_X)   |
| 12    | Dimension Y                        |     | PrdExec_Input_Event (Dimension_Y)   |
| 13    | Dimension Z                        |     | PrdExec_Input_Event (Dimension_Z)   |
| 14    | Dimension A                        |     | PrdExec_Input_Event (Dimension_A)   |
| 15    | Unloaded                           |     | PrdExec_Input_Event (Unloaded)  |
| 16    | ESignature                         |     | Signature ID from the ESignature table  |

### Input Event Result Sets Tips

- Load Staged Transaction\_Type=2, PEIP\_Id=2,Event\_Id needed
- Unload Staged Transaction\_Type=2, PEIP\_Id=2,Event\_Id Null
- Load Running Transaction\_Type=2, PEIP\_Id=1,Event\_Id needed
- Unload Running Transaction\_Type=2, PEIP\_Id=1,Event\_Id Null
- Complete Transaction\_Type=1, PEIP\_Id=1,Event\_Id Null

## Result Set 13 - Defect Detail

The following table lists the parameters used in an Defect Details Result Set.

## Result Sets

| Order | Field Name        | Req | Values/Table Reference  |
|-------|-------------------|-----|---|
| 0     | Result Set Type   | X   | 13  |
| 1     | PreDB             | X   | 0 - Post-Update. Database Manager writes to db and sends to client.<br>1 - Pre-Update. Data already in database and sent to clients only.                                 |
| 2     | TransType         | X   | 1 = Add<br>2 = Update<br>3 = Delete   |
| 3     | TransNum          | X   | 0 – Update fields that are not null to the new values.<br>2 – Update all fields of Event_Details to the values in Result Set. Values of 0 for dimensions are set to null. |
| 4     | DefectDetailId    | X   | Defect_Details (Defect_Detail_Id)   |
| 5     | DefectTypeId      | X   | Defect_Details (Defect_Type_Id)<br>Defect_Types (Defect_Type_Id)  |
| 6     | Cause1            |     | Defect_Details (Cause1), Event_Reasons (Reason_Id)  |
| 7     | Cause2            |     | Defect_Details (Cause2), Event_Reasons (Reason_Id)  |
| 8     | Cause3            |     | Defect_Details (Cause3), Event_Reasons (Reason_Id)  |
| 9     | Cause4            |     | Defect_Details (Cause4), Event_Reasons (Reason_Id)  |
| 10    | CauseCommentId    |     | Defect_Details (Cause_Comment_Id),<br>Comments (Comment_Id)   |
| 11    | Action1           |     | Defect_Details (Action1), Event_Reasons (Reason_Id)   |
| 12    | Action2           |     | Defect_Details (Action1), Event_Reasons (Reason_Id)   |
| 13    | Action3           |     | Defect_Details (Action1), Event_Reasons (Reason_Id)   |
| 14    | Action4           |     | Defect_Details (Action1), Event_Reasons (Reason_Id)   |
| 15    | ActionCommentId   |     | Defect_Details (Action_Comment_Id),<br>Comments (Comment_Id)  |
| 16    | ResearchStatusId  |     | Defect_Details (Research_Status_Id)   |
| 17    | ResearchCommentId |     | Defect_Details (Research_Comment_Id),<br>Comments (Comment_Id)  |
| 18    | ResearchUserId    |     | Defect_Details (Research_User_Id), Users (User_Id)  |
| 19    | EventId           |     | Defect_Details (Event_Id), Events (Event_Id)  |
| 20    | SourcePUId        |     | Defect_Details (Source_PU_Id), Events (Event_Id)  |
| 21    | PUId              |     | Defect_Details (PU_Id), Prod_Units (PU_Id)  |
| 22    | EventSubtypeId    |     | Defect_Details (Event_Subtype_Id)<br>Event_Subtypes (Event_Subtype_Id)  |
| 23    | UserId            |     | Defect_Details (User_Id), Users (User_Id)   |
| 24    | Severity          |     | Defect_Details (Severity)   |
| 25    | Repeat            |     | Defect_Details (Repeat)   |
| 26    | DimensionX        |     | Defect_Details (Dimension_X)  |
| 27    | DimensionY        |     | Defect_Details (Dimension_Y)  |
| 28    | DimensionZ        |     | Defect_Details (Dimension_Z)  |
| 29    | DimensionA        |     | Defect_Details (Dimension_A)  |
| 30    | Amount            |     | Defect_Details (Amount)   |

|    |                   |  |  |
|----|-------------------|--|--|
| 31 | StartCoordinateX  |  | Defect_Details (Start_Coordinate_X)    |
| 32 | StartCoordinateY  |  | Defect_Details (Start_Coordinate_Y)    |
| 33 | StartCoordinateZ  |  | Defect_Details (Start_Coordinate_Z)    |
| 34 | StartCoordinateA  |  | Defect_Details (Start_Coordinate_A)    |
| 35 | ResearchOpenDate  |  | Defect_Details (Research_Open_Date)    |
| 36 | ResearchCloseDate |  | Defect_Details (Research_Close_Date)   |
| 37 | StartTime         |  | Defect_Details (Start_Time)            |
| 38 | EndTime           |  | Defect_Details (End_Time)              |
| 39 | EntryOn           |  | Defect_Details (Entry_On)              |
| 40 | ESignature        |  | Signature ID from the ESignature table |

## Result Set 14 - Historian Write

The following table lists the parameters used in an Historian Write Result Set.

| Order | Field Name      | Req | Values/Table Reference   |
|-------|-----------------|-----|--|
| 0     | Result Set Type | X   | 14   |
| 1     | NodeAlias       | X   | The Alias Name of the Historian (Historian Name)                             |
| 2     | Tag             | X   | The Tag to write to  |
| 3     | Value           | X   | The Value to be written  |
| 4     | Timestamp       | X   | The time the tag should be written to. Not applicable for OPC DA historians. |

## Result Set 15 - Production Plan

The following table lists the parameters used in a Production Plan Result Set.

| Order | Field Name      | Req | Values/Table Reference   |
|-------|-----------------|-----|--|
| 0     | Result Set Type | X   | 15   |
| 1     | Pre DB          | X   | 1 - Pre-update. Database Manager writes to database and sends to client.<br>0 - Post-update. Data already in database and sent to clients only.  |
| 2     | TransType       | X   | 1 - Add, 2 - Update, 3 - Delete  |
| 3     | TransNum        | X   | 0 - Coalesce: replace any null values in the result set with values for the record being updated from the Production Plan table.<br>1 - Comment update only<br>2 - Do not Coalesce: use the null values sent in with the result set to replace values of the specified record in the Production Plan table.<br>91 - Return the specified Child process order to Parent process order<br>92 - Create Child process order based on Start Time. Important: set the Misc1 value for rows with this transaction number to the appropriate Parent_PP_Setup_Id.<br>93 - Create Child process order before Process Order. Important: set the Misc1 value for rows with this transaction number to the appropriate Parent_PP_Setup_Id.<br>94 - Create Child process order after Process Order Important: set the Misc1 value for rows with this |

## Result Sets

|    |                   |   |   |
|----|-------------------|---|---|
|    |                   |   | transaction number to the appropriate Parent_PP_Setup_Id.<br>95 - Rework process order<br>96 - Either bind or unbind process order based on the current value of the specified record.<br>97 - Process order status transition<br>98 - Move process order back<br>99 - Move process order forward |
| 4  | PathId            | X | Prdexec_Paths (Path_Id)<br>Production_Plan (Path_Id)<br>Production_Plan_Status (Path_Id)  |
| 5  | PPIId             |   | Production_Plan (PP_Id)   |
| 6  | CommentId         |   | Production_Plan (Comment_Id)<br>Comments (Comment_Id)   |
| 7  | ProdId            | X | Production_Plan (Prod_Id)<br>Products (Prod_Id)   |
| 8  | ImpliedSequence   |   | Production_Plan (Implied_Sequence)  |
| 9  | PPStatusId        | X | Production_Plan (PP_Status_Id)<br>Production_Plan_Statuses (PP_Status_Id)   |
| 10 | PPTypeld          |   | Production_Plan (PP_Type_Id)<br>Production_Plan_Types (PP_Type_Id)  |
| 11 | SourcePPIId       |   | Production_Plan (Source_PP_Id)  |
| 12 | UserId            | X | Production_Plan (User_id)<br>Users (User_Id)  |
| 13 | ParentPPIId       |   | Production_Plan (Parent_PP_Id)  |
| 14 | ControlType       |   | Production_Plan (Control_Type)<br>Control_Types (Control_Type_Id)   |
| 15 | ForecastStartTime | X | Production_Plan (Forecast_Start_Date)   |
| 16 | ForecastEndTime   | X | Production_Plan (Forecast_End_Date)   |
| 17 | EntryOn           |   | Production_Plan (Entry On)  |
| 18 | ForecastQuantity  | X | Production_Plan (Forecast_Quantity)   |
| 19 | ProductionRate    |   | Production_Plan (Production_Rate)   |
| 20 | AdjustedQuantity  |   | Production_Plan (Adjusted_Quantity)   |
| 21 | BlockNumber       |   | Production_Plan (Block_Number)  |
| 22 | ProcessOrder      |   | Production_Plan (Process_Order)   |
| 23 | TransactionTime   |   |   |
| 24 | Misc1             |   | Use depends on other values in the row.<br>If transaction number is 92, 93, or 94 the Parent_PP_Setup_Id should be placed in this value.  |
| 25 | Misc2             |   | Currently unused.   |
| 26 | Misc3             |   | Currently unused.   |
| 27 | Misc4             |   | Currently unused.   |
| 28 | BOMFormulationId  |   | Production_Plan (BOM_Formulation_ID)  |

### Partial Example Code for Using Result Set 15

```
--Declare the SQL variables used
Declare
@PreDB tinyint,
@TransType int,
@TransNum int,
@PathId int,
```

```

@PPIId int,
@CommentId int,
@ProdId int,
@ImpliedSequence int,
@PPStatusId int,
@PPTypeId int,
@SourcePPIId int,
@UserId int,
@ParentPPIId int,
@ControlType tinyint,
@ForecastStartTime datetime,
@ForecastEndTime datetime,
@EntryOn datetime,
@ForecastQuantity float,
@ProductionRate float,
@AdjustedQuantity float,
@BlockNumber varchar(50),
@ProcessOrder varchar(50),
@TransactionTime datetime,
@Misc1 varchar(50),
@Misc2 varchar(50),
@Misc3 varchar(50),
@Misc4 varchar(50)

```

--Create a temporary table to hold all variable result set rows

```

CREATE TABLE #ProductionPlanUpdates (
    PreDB tinyint,
    TransType int,
    TransNum int,
    PathId int,
    PPIId int,
    CommentId int,
    ProdId int,
    ImpliedSequence int,
    PPStatusId int,
    PPTypeId int,
    SourcePPIId int,
    UserId int,
    ParentPPIId int,
    ControlType tinyint,
    ForecastStartTime datetime,
    ForecastEndTime datetime,
    EntryOn datetime,
    ForecastQuantity float,

```



## Result Sets

```
    ProductionRate float,  
    AdjustedQuantity float,  
    BlockNumber varchar(50),  
    ProcessOrder varchar(50),  
    TransactionTime datetime,  
    Misc1 varchar(50),  
    Misc2 varchar(50),  
    Misc3 varchar(50),  
    Misc4 varchar(50))  
  
--Set Result Set Variables to values retrieved in your custom code. No  
--specific custom code is being show.  
Select @Var_Id = Var_Id  
    from Variables where PU_Id = @PU_Id And Var_Desc Like '%Remaining%'  
Select @PU_Id = @PU_Id  
Select @User_Id = 0  
Select @Canceled = 0  
Select @Result = @Value  
Select @Result_On = @Timestamp  
Select @Transaction_Type = 1  
Select @PostUpdate = 0  
Select @PreDB = 0  
Select @TransType = 1  
Select @TransNum = 0  
Select @PathId = @IncomingPathId  
Select @PPIId = @IncomingPPIId  
Select @ProdId = @IncomingProdId  
Select @PPStatusId = PP_Status_Id from Production_Plan_Sstatuses  
Where PP_Status_Desc = 'pending'  
Select @PPTypeId = 1 --Scheduled  
Select @SourcePPIId = NULL  
Select @UserId = 0  
Select @ParentPPIId = NULL  
Select @ControlType = 1 --Duration  
Select @ForecastStartTime = getdate()  
Select @ForecastEndTime = dateadd(mi,15,getdate())  
Select @EntryOn = getdate()  
Select @ForecastQuantity = 5000  
Select @ProcessOrder = '12345678'  
Select @TransactionTime = getdate()  
Select @Misc1 = NULL  
Select @Misc2 = NULL  
Select @Misc3 = NULL  
Select @Misc4 = NULL
```

```
Insert into #ProductionPlanUpdates (PreDB,  
    TransType,  
    TransNum,  
    PathId,  
    PPIId,  
    CommentId,  
    ProdId,  
    ImpliedSequence,  
    PPStatusId,  
    PPTypeId,  
    SourcePPIId,  
    UserId,  
    ParentPPIId,  
    ControlType,  
    ForecastStartTime,  
    ForecastEndTime,  
    EntryOn,  
    ForecastQuantity,  
    ProductionRate,  
    AdjustedQuantity,  
    BlockNumber,  
    ProcessOrder,  
    TransactionTime,  
    Misc1,  
    Misc2,  
    Misc3,  
    Misc4)  
Values (@PreDB,  
    @TransType,  
    @TransNum,  
    @PathId,  
    @PPIId,  
    @CommentId,  
    @ProdId,  
    @ImpliedSequence,  
    @PPStatusId,  
    @PPTypeId,  
    @SourcePPIId,  
    @UserId,  
    @ParentPPIId,  
    @ControlType,  
    @ForecastStartTime,
```

## Result Sets

```
@ForecastEndTime,  
@EntryOn,  
@ForecastQuantity,  
@ProductionRate,  
@AdjustedQuantity,  
@BlockNumber,  
@ProcessOrder,  
@TransactionTime,  
@Misc1,  
@Misc2,  
@Misc3,  
@Misc4)
```

```
-- Since the table was created in the same order as the result set  
-- we can just place the 15 in front of the output of a "select *"  
If (Select Count(*) From #ProductionPlanUpdates) > 0  
    Select 15,* from # ProductionPlanUpdates
```

## Result Set 16 - Production Setup

The following table lists the parameters used in a Production Setup Result Set.

| Order | Field Name         | Req | Values/Table Reference   |
|-------|--------------------|-----|--|
| 0     | Result Set Type    | X   | 16   |
| 1     | Update Type        | X   | 0 - Post-Update. Data already in db and sent to clients only.<br>1 - Pre-Update. Database Manager writes to db and sends to client.  |
| 2     | Transaction Type   | X   | 1 = Add<br>2 = Update<br>3 = Delete  |
| 3     | Transaction Number | X   | Transaction Numbers<br>00 - Coalesce<br>01 - Comment Update<br>02 - No Coalesce<br>91 - Return To Parent Sequence<br>92 - Create Child Sequence<br>97 - Status Transition<br>98 - Move Sequence Back<br>99 - Move Sequence Forward |
| 4     | PathId             | X   |  |
| 5     | PPSetupId          |     |  |
| 6     | PPId               | X   |  |
| 7     | ImpliedSequence    |     |  |
| 8     | PPStatusId         |     |  |
| 9     | PatternRepetitions |     |  |
| 10    | CommentId          |     |  |
| 11    | ForecastQuantity   |     |  |
| 12    | BaseDimensionX     |     |  |
| 13    | BaseDimensionY     |     |  |
| 14    | BaseDimensionZ     |     |  |
| 15    | BaseDimensionA     |     |  |
| 16    | BaseGeneral1       |     |  |
| 17    | BaseGeneral2       |     |  |
| 18    | BaseGeneral3       |     |  |
| 19    | BaseGeneral4       |     |  |
| 20    | Shrinkage          |     |  |
| 21    | PatternCode        |     |  |
| 22    | UserId             | X   |  |
| 23    | EntryOn            |     |  |
| 24    | TransactionTime    |     |  |
| 25    | ParentPPSetupId    |     |  |

## Result Set 17 - Production Plan Starts

The following table lists the parameters used in a Production Plan Starts Result Set.

| Order | Field Name      | Req | Values/Table Reference |
|-------|-----------------|-----|------------------------|
| 0     | Result Set Type | X   | 17                     |

## Result Sets

|    |                  |   |   |
|----|------------------|---|---|
| 1  | Update Type      | X | 0 - Post-Update. Data already in db and sent to clients only.<br>1 - Pre-Update. Database Manager writes to db and sends to client. |
| 2  | Transaction Type | X | 1 = Add<br>2 = Update<br>3 = Delete   |
| 3  | Transaction Num  | X |   |
| 4  | PU Id            | X |   |
| 5  | PP Start Id      |   |   |
| 6  | Start Time       | X |   |
| 7  | End Time         |   |   |
| 8  | PP Id            |   |   |
| 9  | Comment Id       |   |   |
| 10 | PP Setup Id      |   |   |
| 11 | User Id          |   |   |

## Result Set 18 - Production Execution Path Unit Starts

The following table lists the parameters used in a Production Execution Path Unit Starts Result Set.

| Order | Field Name           | Req | Values/Table Reference  |
|-------|----------------------|-----|---|
| 0     | Result Set Type (18) | X   | 18  |
| 1     | Update Type          | X   | 0 - Post-Update. Database Manager writes to db and sends to client.<br>1 - Pre-Update. Data already in db and sent to clients only. |
| 2     | Transaction Type     | X   | 1 = Add<br>2 = Update<br>3 = Delete   |
| 3     | Transaction Num      | X   |   |
| 4     | PU Id                | X   |   |
| 5     | PEPUS Id             |     |   |
| 6     | Path Id              | X   |   |
| 7     | Comment Id           |     |   |
| 8     | Start time           |     |   |
| 9     | End Time             |     |   |

## Result Set 19 - Production Stats

The following table lists the parameters used in a Production Stats Result Set.

| Order | Field Name           | Req | Values/Table Reference  |
|-------|----------------------|-----|---|
| 0     | Result Set Type (19) | X   | 19  |
| 1     | PreDB                | X   | 0 - Post-Update. Database Manager writes to db and sends to client.<br>1 - Pre-Update. Data already in db and sent to clients only. |
| 2     | Transaction Type     | X   | 1 = Add<br>2 = Update<br>3 = Delete   |
| 3     | Transaction Number   | X   |   |
| 4     | Path Id              | X   |   |
| 5     | Statistics Type      | X   |   |

|    |                              |   |  |
|----|------------------------------|---|--|
| 6  | Id                           | X |  |
| 7  | Actual Start Time            | X |  |
| 8  | Actual End Time              | X |  |
| 9  | Actual Good Items            | X |  |
| 10 | Actual Bad Items             | X |  |
| 11 | Actual Running Time          | X |  |
| 12 | Actual Down Time             | X |  |
| 13 | Actual Good Quantity         | X |  |
| 14 | Actual Bad Quantity          | X |  |
| 15 | Predicted Total Duration     | X |  |
| 16 | Predicted Remaining Duration | X |  |
| 17 | Predicted Remaining Quantity | X |  |
| 18 | Alarm Count                  | X |  |
| 19 | Late Items                   | X |  |
| 20 | Repetitions                  | X |  |

## Result Set 20 - HistorianRead Result Set

The following table lists the parameters used in a HistorianRead Result Set.

| Order | Field Name      | Req | Values/Table Reference                           |
|-------|-----------------|-----|--|
| 0     | Result Set Type | X   | 20   |
| 1     | Start Time      | X   | Start Time of Sampling Period                    |
| 2     | End Time        | X   | End Time of Sampling Period                      |
| 3     | Tag             | X   | Tag to Read                                      |
| 4     | Node Alias      | X   | The Alias Name of the Historian (Historian Name) |

## Result Sets

|   |               |   |   |
|---|---------------|---|---|
| 5 | Sampling Type | X | 1 = Average<br>2 = Interpolated<br>4 = Minimum<br>5 = Maximum<br>6 = Standard Deviation<br>7 = Total<br>8 = Cpk<br>9 = % In Warning<br>10 = % In Reject<br>12 = Last Good Value<br>13 = Count<br>14 = Next Good Value<br>15 = Closest Good Value<br>16 = Increase<br>17 = Minimum TimeStamp<br>18 = Maximum TimeStamp<br>19 = Event TimeStamp<br>20 = Raw Total<br>21 = Raw Count<br>22 = Raw Unique Count<br>23 = Base TimeStamp - Interpolated<br>24 = Raw Average<br>25 = Raw Minimum<br>26 = Raw Maximum<br>27 = Raw Standard Deviation<br>28 = Raw Values<br>29 = Base TimeStamp - Last Good Value<br>30 = Base TimeStamp - Next Good Value<br>31 = Raw Increase<br>32 = Cp<br>33 = Pp<br>34 = Ppk |
| 6 | Variable Id   | X | The Destination Variable where the result should be placed.   |
| 7 | Lag Time      | X | Time to wait before reading (in seconds)  |

### Partial Sample Code for Using Result Set 20

```
--Declare the SQL variables used

Declare

@StartTime datetime,

@EndTime datetime,

@Tag varchar(255),

@NodeAlias varchar(50),

@SamplingType int,

@VariableId int,

@LagTime int

--Create a temporary table to hold all variable result set rows

CREATE TABLE #HistorianRead (

StartTime datetime,
```

```
EndTime datetime,  
Tag varchar(255),  
NodeAlias varchar(50),  
SamplingType int,  
VariableId int,  
LagTime int  
)  
--Set Result Set Variables to values retrieved in your custom code. No  
--specific custom code is being show.  
Select @StartTime = getdate()  
Select @EndTime = dateadd(mi,15,getdate())  
Select @Tag = 'MyTagName'  
Select @NodeAlias = 'MyHistorianName'  
Select @SamplingType = 2  
Select @VariableId = 1302  
Select @LagTime = 5  
  
Insert into # HistorianRead(  
StartTime,  
EndTime,  
Tag,  
NodeAlias,  
SamplingType,  
VariableId,  
LagTime  
)  
Values(  
@StartTime,  
@EndTime,  
@Tag,  
@NodeAlias,  
@SamplingType,  
@VariableId,
```



## Result Sets

@LagTime)

-- Since the table was created in the same order as the result set

-- we can just place the 20 in front of the output of a "select \*"

If (Select Count(\*) From #HistorianRead) > 0

Select 20,\* from #HistorianRead

## Result Set 21 - Non-Productive Time

The following table lists the parameters used in a Non-Productive Time Result Set.

| Order | Field Name      | Req | Values/Table Reference |
|-------|-----------------|-----|------------------------|
| 0     | Result Set Type | X   | 21                     |
| 1     | PreDB           |     |                        |
| 2     | TransactionType |     |                        |
| 3     | TransNum        |     |                        |
| 4     | PUId            |     |                        |
| 5     | StartTime       |     |                        |
| 6     | EndTime         |     |                        |
| 7     | Reason1         |     |                        |
| 8     | Reason2         |     |                        |
| 9     | Reason3         |     |                        |
| 10    | Reason4         |     |                        |
| 11    | UserId          |     |                        |
| 12    | CommentId       |     |                        |
| 13    | RsnTreeDataId   |     |                        |
| 14    | EntryOn         |     |                        |
| 15    | NPDetId         |     |                        |

## Result Set 50 - Output File Creation

The Plant Applications File interface models allow for the export of structured or delimited data from Plant Applications into a file, and for the import of file data into Plant Applications. A Plant Applications calculation that is either time based or event based triggers the export of data, which is accomplished through calling a stored procedure (SP) from the calculation. The SP does much of the work via its select and update queries, which gather up the information, often into temporary tables, and then writes it out to a file using Plant Applications specific Result Sets to generate the file, its name, and its format. Imports are done by having files FTP'd to the Proficy Server and by configuring a specific model to watch a directory for a new file and specifying a stored procedure to be called by the model to handle the importing of the data into Plant Applications.

### Configuration

General configuration and specific configuration for each type of transfer for the Plant Applications File Interfaces will be performed in Plant Applications models and on variables in the Plant Applications Server using the Plant Applications Administrator. (A Plant Applications model performs tasks according to configurable model attributes in order to accomplish a set of defined tasks.)

### Startup and Shutdown

The Plant Applications File Interfaces run as a calculation inside the Calculation Manager Service or as a component or model inside of the Plant Applications Event Manager Service. In order to start or stop the Plant Applications File Interfaces, the various models are activated or deactivated through the Plant Applications Administrator Configure Events option or by activating or deactivating the calculation.

### Communication Protocol

The protocol of communication between the Plant Applications Server and other computers will be file transfer using FTP over TCP/IP. It can be assumed that both the Proficy Server and other computers will have appropriately configured IP addresses and will have FTP services loaded, tested, and fully functional. The actual transfer of the files can be done using the Plant Applications FTP Engine. This is configured using the Plant Applications Administrator. The testing of sending, retrieval, and deletion of files (Gets) using an FTP client is important to verify by using a program such as WS\_FTP prior to commissioning this interface. The Plant Applications FTP Engine is an FTP Client.

### Log Files

All messages logged by the by the Plant Applications File Interfaces are logged in the Plant Applications Calculation or Event Manager logfiles in the Proficy\Logfiles directory named EventMgr-01.Log or CalculationMgr-01.Log with the 01 being the version of the log file. In the "Log" file there will be error messages and in the "Show" file EventMgr.Shw or CalculationMgr.Shw there is a summary of the current configuration. By default, the interface will log only error messages to the Log file including the date and time along with transaction specific data for the error. The Show file will contain details of the current configuration of the interface including which models are activated and the model parameters.

### Result Set 50 Description

Plant Applications data export to a flat file can occur when a calculation, an event, or time based trigger calls a stored procedure. A Plant Applications calculation that is either time based or event based or an event triggers the export of data, which is accomplished through calling a stored procedure (SP) from the calculation. The Stored Procedure does much of the work via its select and update queries, which gather up the information, often into temporary tables, and then writes it out to a file using the Plant Applications Result Set Type 50 to generate the file, its name, and its format. The appropriate Plant Applications Service will interpret the result set and build the

file. Transfer of files to other computers is done by having files FTP'd to the other computer using the Plant Applications FTP Engine.

---

***INFORMATION:*** *There is a maximum column size of 3000 characters.*

---

### Result Set 50 Output File Parameters

| Order | Field Name          | Values/Table Reference  |
|-------|---------------------|---|
| 0     | Result Set Type     | 50  |
| 1     | File Number         | Allows the capability with one select statement. The number of the file to include the Value in..   |
| 2     | File Name           | The name of the file to create for a specific file number.  |
| 3     | Field Number        | Not Utilized  |
| 4     | Field Name          | Not Utilized  |
| 5     | Type                | Alpha – pad the value with spaces up to the Length parameter.<br>Any value but Alpha (Numeric) – Gets treated like a number. The number is formatted to the precision and the value is padded with spaces up to the Length parameter. |
| 6     | Length              | Length of the Value used for padding spaces at the end.   |
| 7     | Precision           | Null  |
| 8     | Value               | The Value to be put in the file.  |
| 9     | Carriage Return     | Carriage Return for that line. Yes = 1, No = 0  |
| 10    | Construction Path   | Drive and Directory for file construction.  |
| 11    | Final Path          | Drive and Directory for completed file.   |
| 12    | Move Mask (NO PATH) | File mask of the file to move from the Construction Path to the Final Path.   |
| 13    | Add Timestamp       | 0-No, 1-Short, 2-Full   |

### Key Parts of a Result Set 50 Example

```

Create Table #FileOutput (
  FileNumber          int,
  FileName            varchar(20) NULL,
  FieldNumber        int,
  FieldName           varchar(20),
  FieldType           varchar(20),
  FieldLength         int,
  FieldPrecision      int NULL,
  FieldValue          varchar(100) NULL,
  FieldCR             int DEFAULT 0,
  FieldBuildPath      varchar(50) NULL,
  FieldFinalPath      varchar(50) NULL,
  FieldMoveMask       varchar(50) NULL,
  AddTimestamp        int
)
Select @FileNumber = 1
Select @FileBuildPath = 'C:\Proficy\FileExport\Construction\'
Select @FileFinalPath = 'C:\Proficy\FileExport\Outgoing\'
Select @FileNumber = @FileNumber + 1

```

## Result Sets

```
Select @FieldNumber = 1
Select @FileName = @Prod_Code + '.SR'
Select @FieldType = 'r;Alpha'
Select @Msg = 'RECIPE ' + @Prod_Code + ';'
Select @MsgLength= len(@msg)
Select @FieldMoveMask = 'r;*.txt'

Insert Into #FileOutput (FileNumber, FileName, FieldNumber, FieldName,
FieldType, FieldLength, FieldPrecision, FieldValue, FieldCR,
FieldBuildPath, FieldFinalPath,FieldMoveMask,AddTimestamp)
    Values
(@FileNumber,@FileName,@FieldNumber,@FieldName,'Alpha',@MsgLength,Null,'Value1',1,@FileBuildPath,@FileFinalPath, @FieldMoveMask,0)

Insert Into #FileOutput (FileNumber, FileName, FieldNumber, FieldName,
FieldType, FieldLength, FieldPrecision, FieldValue, FieldCR,
FieldBuildPath, FieldFinalPath,FieldMoveMask,AddTimestamp)
    Values (@FileNumber,@FileName,@FieldNumber,@FieldName,'Alpha',1,Null,
'Value2',1,@FileBuildPath,@FileFinalPath, @FieldMoveMask,0)

Insert Into #FileOutput (FileNumber, FileName, FieldNumber, FieldName,
FieldType, FieldLength, FieldPrecision, FieldValue, FieldCR,
FieldBuildPath, FieldFinalPath,FieldMoveMask,AddTimestamp)
    Values (@FileNumber,@FileName,@FieldNumber,@FieldName,'Alpha',1,Null,
'Value3',1,@FileBuildPath,@FileFinalPath, @FieldMoveMask,0)

Drop Table #FileOutput
Select 50, * From #FileOutput Order By FileNumber, FieldNumber
```

## Result Set 51 - Return Parameters

Result Set 51 has two fields and allows special parameters to be returned from local stored procedures (spLocals).

| Order | Field Name      | Req | Values/Table Reference                         |
|-------|-----------------|-----|--|
| 0     | Result Set Type | X   | 51   |
| 1     | ParameterName   | X   | The name of the parameter.                     |
| 2     | ParameterValue  | X   | The value of the parameter you want to return. |