PI Interfaces for Batch and Manufacturing Execution Systems

User Guide

OSIsoft, LLC
777 Davis St., Suite 250
San Leandro, CA 94577 USA
Tel: (01) 510-297-5800
Fax: (01) 510-357-8136
Web: http://www.osisoft.com

PI Interfaces for Batch and Manufacturing Execution Systems User Guide

Version: 3.0.X

Published: 25 Jun 2014

Contents

Introduction to PI interfaces for batch and manufacturing execution systems

PI interfaces for Batch Execution Systems (BES) and Manufacturing Execution Systems (MES) operate with the following batch execution systems:

- ABB 800xA Batch
- Emerson DeltaV Batch, and DeltaV Syncade Batch
- Event-file-based interfaces:
  - GE iBatch
  - Honeywell TotalPlant Batch
  - Rockwell FactoryTalk
- FoxBatch IA Series (not included in this document - see the user guide installed with the interface)
- Siemens SIMATIC
- Werum PAS-X
- Wonderware InBatch

The PI interfaces for these systems are based on a common framework. For vendor-specific information, refer to the corresponding chapter in this guide.

Note:
If you record batch process data directly to PI tags and do not use a BES, you can generate batch data from PI tag data using the PiBaGen or PIEFGEN utilities. For details, refer to the manuals for these applications.

PI interfaces for batch and manufacturing execution systems are scan-based interfaces that populate the PI AF database (with event frames and elements) or the PI Batch Database and PI Module Database (with batches, modules and properties) based on events and data read from a data source.

Note:
To use event frames, your PI batch interface must be version 3.x or higher.

These interfaces can read data from multiple batch data sources, which enables the PI System to handle scenarios in which different overlapping batch recipes can access the same unit in different stages of the production cycle. By acquiring data for the same time frame from multiple sources and collating it into a single time-ordered sequence, a single interface instance can capture the complete history of the batch process.

The interfaces can be configured to create and update PI tags and PI Units or AF elements based on the data received. The interface cannot update the batch data source.

Unlike other OSIsoft interfaces, batch-related interfaces do not use PI buffering. Batch data is persistent in the data source and not in danger of being lost. If the interface loses its connection to the PI Data server, it continues to collect data from the data source, transmitting it to the server when the connection is reestablished.

Note:
These interfaces are designed for recipes that constrain a unit to run only one unit procedure at a time.

Two different models are used to describe batch processes. The equipment model describes the physical equipment necessary to create a batch while the recipe model describes the procedures that are performed during the execution of a recipe. There is no intrinsic or direct relationship between the models. With the exception of arbitration events, journal files contain only recipe model event information.

The S88 process model is composed of the following hierarchy:

- Procedure (recipe)
- Unit procedures
- Operations
- Phases
- Phase steps
- Phase states

Note:
According to the ISA S88.01 standard, procedures and unit procedures are optional. A recipe can be composed solely of operations and phases.

The physical model is composed of the following equipment-oriented hierarchy:

- Enterprise
- Site
- Area
- Process cell
- Unit
- Equipment module
- Control module

The PI Batch Database does not use a strict S88 approach to describe or record batch data. Unit procedures from the data source are mapped to PI UnitBatches. Only a single unit procedure can be active in a unit at any given time, which restricts the configuration of recipes that can be run by the batch execution system if batch data is to be captured by the interface in a reliable and meaningful way. By contrast, event frames support parallel unit procedures natively.

You can configure interfaces to create PI tags and equipment (elements in PI AF or PI Units in the PI Module Database) by defining templates, which specify the events that trigger creation and configure how the tag or equipment item is to be created.

Related manuals:

- *PI Data Archive Reference Guide*
- *PI Data Archive System Management Guide*
- *PI SDK User Manual*
- *AF SDK User Manual*

For details about the format and contents of source data, refer to the documentation provided by your batch execution system vendor.

Prerequisites for installing PI interfaces for batch and manufacturing execution systems

Before installing and configuring, ensure that the following prerequisites are met:

- Verify that the PI Data Archive is running and that the PI Data server is accessible from the computer where you intend to run the interface (the interface node).
- If you intended to generate event frames, make sure that the PI Asset Framework is running and that the PI Asset server is accessible from the interface node.
- Ensure that the system time on all these computers is correct.
- Verify that your batch execution system is up and running and that the data source is accessible from the interface node.

To install the interface, download and run its setup kit. By default, the interface is installed in an interface-specific folder in the following location: `%PIHOME%\Interfaces\`

The interface installation directory contains all the files and folders required to configure and run the interface, and includes example configurations.

The interface can run on the same computer as the BES or on a dedicated node. To avoid affecting the performance of the PI Data Archive, do not install the interface on the PI Data server. When installing the interface, reserve the C: drive for the operating system, and install the interface on another drive.

If the data source is Microsoft SQL Server, you must install the Microsoft SQL Native Client on the interface node. You can download the client from the MSDN web site. If the data source is an Oracle database, you must install the corresponding version of Oracle Provider for OLE DB.

Security overview for PI interfaces for batch and manufacturing execution systems

To configure batch interfaces, the user account under which the PI Event Frames Interface Manager runs must be in the local Administrators group. Set the following permissions for the user that runs the interface and all users who need to run the PI Event Frames Interface Manager:

- PI Data Archive permissions (PI SMT: Browse to **Security > Database Security**)
  - Database security: Enable read/write access for the PIPOINT table and read access for PIBACKUP
  - Point Database security: Set both PtSecurity and DataSecurity to read/write
  - Enable read access to the active points
- PI Asset Framework permissions
  - Database: read/write
  - Categories: read
  - Element: read/write

- ◦ Element templates: read
- ◦ Event frames: read/write

If you are running PI Data Archive 3.4.380.36 or later, you can take advantage of its support for Windows Integrated Security by running the interface service using a Windows account that has the required permissions on the PI Data server. To configure Windows Integrated Security, use PI SMT to define a mapping that assigns a PI identity that has the required permissions to the user or user's group.

For pre-3.4.380.36 versions of the PI Data Archive, you must create a PI trust for the user that runs the interface and configuration tool. Limit the trust to the hostname or IP address of the interface node and the application name (BIFConfig.exe for the PI Event Frames Interface Manager).

Create and configure the interface instance

For each instance you create, settings are stored in a separate Windows command (.bat) file and an initialization (.ini) file in the interface installation folder. The batch file launches the interface, specifying settings as command line parameters. The initialization file also contains settings, and it defines templates that determine how data from the data source is stored in the PI System.

To configure settings for interfaces, use the PI Event Frames Interface Manager. Use this tool even if you are configuring the interface to store data in the PI Batch Database rather than event frames.

To launch the PI Event Frames Interface Manager, click **Start** > **All Programs** > **PI System** > **PI Event Frames Interface Manager.** For detailed information about the settings on each tab, see PI Event Frames Interface Manager reference.

To create an instance of the interface, perform the following steps using PI Event Frames Interface Manager:

Procedure

1. On the **Interface Selection** tab, click **Add Interface.** The Interface Selection window opens.
2. Browse to the directory where the interface is installed and choose its executable file. For example: `C:\Program Files (x86)\PIPC\Interfaces\FTBInt\PIFTBInt.exe`.
3. Save your entry, then click **OK**.
4. On the **File Selection** tab, browse to the INI file where you want to store templates.

   The installer installs a sample INI file, named with a "_new" extension, which you can copy and rename.
5. On the **Server Information** tab, specify settings for your PI servers.

   If you intend to create event frames, check **Create event frames** and specify the PI Asset server and AF database.
6. On the **Source** tab, configure the settings for the data source (the BES).

   Note that you can configure multiple data sources for the same interface instance.
7. On the **Templates** tab, define templates for creating and updating PI tags, overriding incoming batch data, and creating and updating batch properties or event frame attributes.

For details, see Templates for mapping data source events.

8. On the **Filters** tab, specify any batch levels from the data source that you do not want the interface to process.
9. On the **Time Settings** tab, configure retry and timeout settings.
10. On the **Batch Setup** tab, configure settings according to the requirements of your batch execution system.
11. On the **Operational Settings** tab, configure settings to determine how batch data is recorded.
12. To save your changes, click **Save Settings**.

PI Data server connectivity

If a connection is lost during processing, the interface suspends all actions until it reconnects to the PI Data server or the data source. If the data source connection is down, the interface tries to reconnect on every scan until it succeeds. If the PI Data server connection is lost, the interface attempts to reconnect periodically until it times out.

You can configure both the retry interval and the timeout period on the **Time Settings** tab of the PI Event Frames Interface Manager. The interface logs any connection errors that occur.

Logging and error messages

The interface logs operational messages during interface startup, data collection and recovery. Additional messages are logged if you enable debugging. The log also contains messages from the interface framework and the PI API (on which this interface is based), and the buffering program. To view messages, open PI System Management Tools, and click **Operation** > **Message Logs**.

For detailed information about interface logging, see the related article in the OSIsoft Knowledge Base.

For details about managing the error logging process, see the PI API Installation Instructions manual.

To enable debug output for troubleshooting, launch PI Event Frames Interface Manager, select your interface instance, and go to the **Operational Settings** tab.

How PI interfaces for batch and manufacturing execution systems work

The PI interface for Batch Execution Systems (BES) and Manufacturing Execution Systems (MES) scans a data source for data that indicates the start or end of a level in the batch hierarchy. Based on the events detected in the data source, the interface generates entries in the PI Batch Database or event frames. You can configure the interface to create and update PI tags and equipment assets in PI AF or the PI Module Database.

You can configure a single interface instance to read multiple data sources, to handle situations in which multiple batch execution systems manage related batch processes that you want to collate.

How interfaces process batch event data

The interface processes start and end events for each level. The level at which a recipe executes depends on the equipment it requires. For example, a batch-level recipe is most likely composed of unit procedures and procedures executed on multiple different units. By contrast, an operation-level recipe might execute a set of phases in a single unit. The interface automatically creates PIBatches (or level 1 events) and PIUnitBatches (or level 2 events) for operation- and lower-level recipes, even though the events in the data source do not include these levels.

Note that the BES events that trigger the start and end of each level are vendor-specific. For details, refer to the vendor-specific information in this document.

- **PIBatch/Procedure**

  The PIBatch is the highest level recorded in the PI Batch Database (or as an event frame). Its properties record general data such as the batch ID, recipe name and type, and so on. If a recipe is composed solely of levels below the unit batch (for example, an operation- or phase-level recipe), the interface generates parent batches and unit batches. The best way for the interface to determine the precise start and end time for a unit batch is to use equipment arbitration events, which record the time a unit was acquired or released. If the BES does not support arbitration events, the interface uses unit batch start and end events, which can be affected by the start and end of lower levels and are inherently less precise than equipment acquisition events.

- **PIUnitBatch/Unit Procedure**

  For each unit procedure that it reads from the data source, the interface creates a PIUnitBatch or an equivalent event frame. The start and end times record the duration of physical processing within a unit.

  The PIUnitBatch or event frame properties contain the batch ID and procedure name as recorded by the data source unless you override it by configuring a batch ID mask using PI Event Frames Interface Manager (on the **Batch Setup** tab). When operation- or phase-level recipes are run, the interface creates a parent unit batch, using the operation or phase name as the procedure name.

- **Operation/PISubBatch**

  For each operation that it reads from the data source, the interface creates a PISubBatch or an equivalent event frame as a child of the parent PIUnitBatch object. For recipes that occur below the unit batch level, the interface generates parent data (unit batches and batches).

- **Phase/PISubBatch**

  For each phase that it reads from the data source, the interface creates a PISubBatch or an equivalent event frame as a child of the parent operation-level object. For recipes that occur below the unit batch level, the interface generates parent events.

- **Phase State/PISubBatch**

  For each phase state that it reads from the data source, the interface creates a PISubBatch or an equivalent event frame as a child of the parent phase-level object. The start of a new phase state ends the previous phase state, except for the terminating COMPLETE, ABORTED and STOPPED phase states.

  Note:
  No phase state data is available from an Emerson DeltaV Alarms & Events data source.

- **Phase Step/PISubBatch**

  Phase steps are not S88-compliant and are implemented differently by BES vendors (some vendors do not support them). By default, phase steps are not enabled. To enable phase steps using PI Event Frames Interface Manager, go to the **Batch Setup** tab, select the **Report as step** check box, and specify the strings recorded by the BES to indicate the start and end of a phase step.

  For each phase step that it reads from the data source, the interface creates a PISubBatch or corresponding event frame as a child of the parent phase-state-level object. Phase steps are always created beneath the first RUNNING phase state, regardless of whether the parent phase state has ended. The first start event starts a phase step, and subsequent start events for the same phase step are ignored. Likewise, the first end event ends a phase step, and subsequent end events for the same phase step are ignored.

  If the parent phase is not found, phase steps do not create higher-level procedures, unit procedures, phase states or operations. If the phase step is not closed by an appropriate closing event, it is closed by the end of the parent phase. Zero-duration phase steps are ignored.

- **Merging Multiple Source Batches**

  To merge multiple source batches that have identical batch IDs, enable the **Merge multiple source batches with same batch ID into one PI batch** option on the **Batch Setup** tab of the PI Event Frame Interface Manager. To ensure that related batch IDs match, you can configure a batch ID mask that extracts a common substring from the incoming ID.

  The interface caches batches and, when it reads a new batch from the source, it checks its cache for a batch with a matching ID. If a match is found, the interface merges the batches. If no match is found in the cache, the interface creates a new batch.

  By default, the interface caches batches for one day. To configure the cache duration, go to the **Time Settings** tab and set the **Cache time** value.

- **Configuring the Batch ID for Merging Multiple Batches**

  To enable the interface to merge multiple incoming source batches into a single batch, you must ensure that the batch IDs are identical. To override the default batch ID that the interface reads from the data source, you can configure a batch ID mask that extracts a common substring of the incoming ID to be used as the batch ID. The mask must match a contiguous substring from the incoming batch ID (that is, you cannot define a mask that skips characters).

To configure the mask using PI Event Frame Interface Manager as follows: on the **Batch Setup** tab, set the **Batch ID mask** field. Specify the mask using fixed text and the following wildcards:

Wildcard	Description
#	Single digit numerical value, 0-9
@	Single alpha character, a-z, A-Z
?	Any single symbol
!	Repeat the previous mask symbol
*	Any set of symbols

You can specify multiple masks as a comma-separated list. The interface uses the result from the first mask that successfully generates a substring. If the interface cannot match the mask to the data in the incoming field, the entire field is used.

Example: For the data source BatchID column that contains the text `lot30112 / 90dev123 / 12345stp / ld567`, the following table lists masks and results.

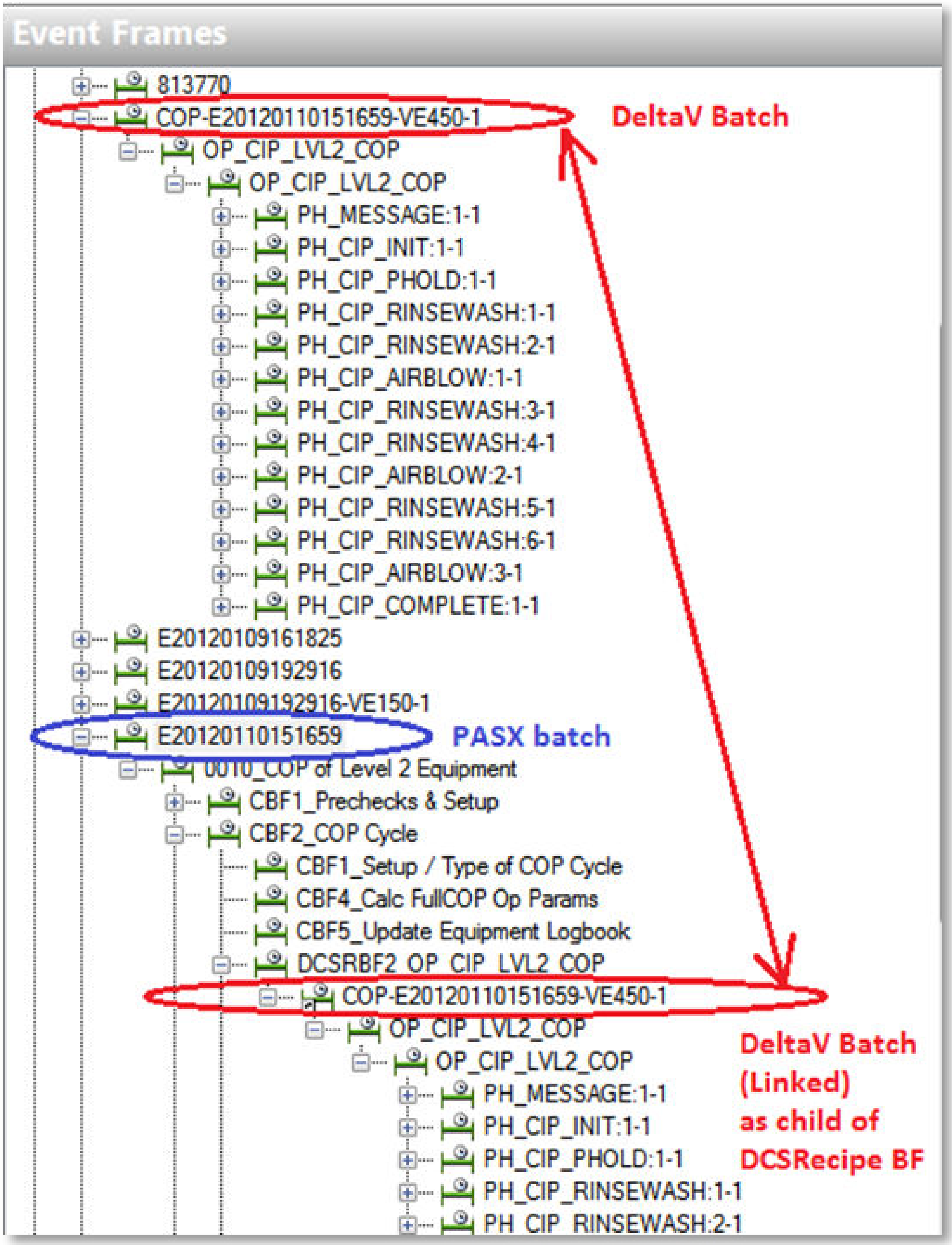
Mask	Result	Matches...
#####	30112	The first five contiguous digits. The first matching substring is used.
###	301	The first three contiguous digits.
@@@#####	lot30112	Five contiguous digits with three contiguous characters and the characters are placed before the sequence of digits.
##@@@###	90dev123	Five contiguous digits with three contiguous characters and the characters are placed before the third digit.
#####@@@	12345stp	Five contiguous digits with three contiguous characters and the characters are followed the digits.
?????	lot30	The first five characters, regardless of type.

- **Linking BES to MES Batches Using Event Frames**

  To consolidate the output of multiple batch execution systems under a common PI AF element, specify the linking element as follows: using PI Event Frames Interface Manager, go to the **Batch Setup** tab, enable the **Link BES event frames to MES** option, and specify the path to the AF element.

  This configuration is typical in an environment where a manufacturing execution system supervises multiple batch execution systems to manufacture a single product. For each batch execution system, configure a dedicated interface instance and specify the same PI AF linking element. The interface creates a consolidated set of references under the linking element, which provides a consolidated view of all the related events from the different batch execution systems.

For example, the following figure illustrates an event frame created based on a batch from an Emerson DeltaV batch execution system. This event frame is referenced under a linking element to an event frame created by a Werum PAS-X manufacturing execution system. Other batch interface instances that are configured to use the same linking element create references to their batches under the same link.



Event Frames

```
813770
COP-E20120110151659-VE450-1
    OP_CIP_LVL2_COP
        OP_CIP_LVL2_COP
            PH_MESSAGE:1-1
            PH_CIP_INIT:1-1
            PH_CIP_PHOLD:1-1
            PH_CIP_RINSEWASH:1-1
            PH_CIP_RINSEWASH:2-1
            PH_CIP_AIRBLOW:1-1
            PH_CIP_RINSEWASH:3-1
            PH_CIP_RINSEWASH:4-1
            PH_CIP_AIRBLOW:2-1
            PH_CIP_RINSEWASH:5-1
            PH_CIP_RINSEWASH:6-1
            PH_CIP_AIRBLOW:3-1
            PH_CIP_COMPLETE:1-1
E20120109161825
E20120109192916
E20120109192916-VE150-1
E20120110151659
    0010_COP of Level 2 Equipment
        CBF1_Prechecks & Setup
        CBF2_COP Cycle
            CBF1_Setup / Type of COP Cycle
            CBF4_Calc FullCOP Op Params
            CBF5_Update Equipment Logbook
            DCSRBF2_OP_CIP_LVL2_COP
                COP-E20120110151659-VE450-1
                    OP_CIP_LVL2_COP
                        OP_CIP_LVL2_COP
                            PH_MESSAGE:1-1
                            PH_CIP_INIT:1-1
                            PH_CIP_PHOLD:1-1
                            PH_CIP_RINSEWASH:1-1
                            PH_CIP_RINSEWASH:2-1
```

DeltaV Batch

PASX batch

DeltaV Batch (Linked) as child of DCSRecipe BF

Templates for mapping data source events

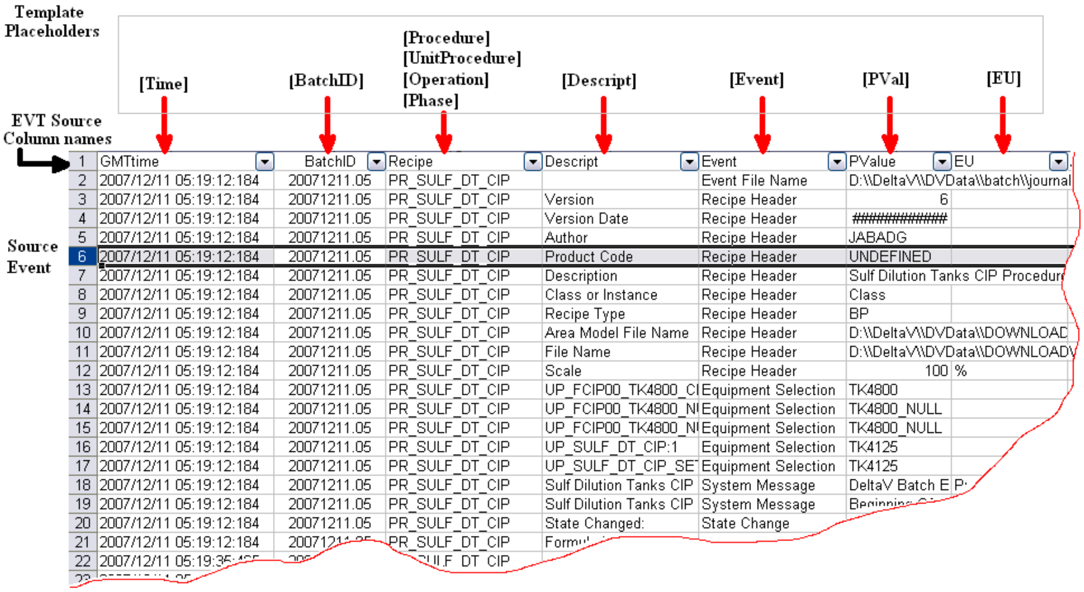
Templates enable you to capture data from the data source, specifying the format for the desired data and the events that cause the interface to capture the data. To configure templates, go to the PI Event Frame Interface Manager **Templates** tab. You can define the following types of templates:

Template Type	Description
Tag	Create and update PI tags using data from data source.
Attribute	Create and update event frame attributes using data from data source.
Property	Create and update PI batch properties using data from data source.
Recipe	Override the data source recipe level names.
Alarm tag	Create and update PI tags using data from an Emerson DeltaV alarms and events data source.

You can configure templates that map the source data to string, integer or float data types and configure the format of the information to be written to the target. The precise format of the events coming from the data source depends on the BES vendor. For detailed information, refer to the vendor-specific topic.

Placeholders and advanced parsing

Placeholders enable you to incorporate data from incoming events into tag names and data. Placeholders can be used in all types of templates. The precise set of placeholders supported by an interface depends on the data source. The following example illustrates how placeholders correspond to columns in a data source.



1	GMTtime	BatchID	Recipe	Descript	Event	PValue	EU
2	2007/12/11 05:19:12:184	20071211.05	PR_SULF_DT_CIP		Event File Name	D:\\DeltaV\\DVData\\batch\\journal	
3	2007/12/11 05:19:12:184	20071211.05	PR_SULF_DT_CIP	Version	Recipe Header	6	
4	2007/12/11 05:19:12:184	20071211.05	PR_SULF_DT_CIP	Version Date	Recipe Header	############	
5	2007/12/11 05:19:12:184	20071211.05	PR_SULF_DT_CIP	Author	Recipe Header	JABADG	
6	2007/12/11 05:19:12:184	20071211.05	PR_SULF_DT_CIP	Product Code	Recipe Header	UNDEFINED	
7	2007/12/11 05:19:12:184	20071211.05	PR_SULF_DT_CIP	Description	Recipe Header	Sulf Dilution Tanks CIP Procedure	
8	2007/12/11 05:19:12:184	20071211.05	PR_SULF_DT_CIP	Class or Instance	Recipe Header	Class	
9	2007/12/11 05:19:12:184	20071211.05	PR_SULF_DT_CIP	Recipe Type	Recipe Header	BP	
10	2007/12/11 05:19:12:184	20071211.05	PR_SULF_DT_CIP	Area Model File Name	Recipe Header	D:\\DeltaV\\DVData\\DOWNLOAD	
11	2007/12/11 05:19:12:184	20071211.05	PR_SULF_DT_CIP	File Name	Recipe Header	D:\\DeltaV\\DVData\\DOWNLOAD\	
12	2007/12/11 05:19:12:184	20071211.05	PR_SULF_DT_CIP	Scale	Recipe Header	100	%
13	2007/12/11 05:19:12:184	20071211.05	PR_SULF_DT_CIP	UP_FCIP00_TK4800_CI	Equipment Selection	TK4800	
14	2007/12/11 05:19:12:184	20071211.05	PR_SULF_DT_CIP	UP_FCIP00_TK4800_N	Equipment Selection	TK4800_NULL	
15	2007/12/11 05:19:12:184	20071211.05	PR_SULF_DT_CIP	UP_FCIP00_TK4800_N	Equipment Selection	TK4800_NULL	
16	2007/12/11 05:19:12:184	20071211.05	PR_SULF_DT_CIP	UP_SULF_DT_CIP:1	Equipment Selection	TK4125	
17	2007/12/11 05:19:12:184	20071211.05	PR_SULF_DT_CIP	UP_SULF_DT_CIP_SE	Equipment Selection	TK4125	
18	2007/12/11 05:19:12:184	20071211.05	PR_SULF_DT_CIP	Sulf Dilution Tanks CIP	System Message	DeltaV Batch E	P
19	2007/12/11 05:19:12:184	20071211.05	PR_SULF_DT_CIP	Sulf Dilution Tanks CIP	System Message	Beginning	
20	2007/12/11 05:19:12:184	20071211.05	PR_SULF_DT_CIP	State Changed:	State Change		
21	2007/12/11 05:19:12:184	20071211.05	PR_SULF_DT_CIP	Formu			
22	2007/12/11 05:19:35		ULF_DT_CIP				

When you define templates using the PI Event Frame Interface Manager, you can choose from a list of supported placeholders. To use a placeholder in a field when editing a template, click the **Add Placeholder…** button and choose the desired placeholder.

You can define placeholders that read data from PI tags when triggered by batch events. To specify a tag-based placeholder, use the following syntax:

```
[Tag, Name="PI Tag Name", <comma-delimited list of parameters>]
```

For example:

```
total:[Tag, name="sinusoid", range="10d", func="TOTAL"]
```

```
min:[Tag, name="test_data_1", range="10d", func="MIN"]
```

The following table describes the components of a tag-based placeholder.

Parameter	Description
`Name="tagname"`	(Required) Defines the exact name of the PI tag which should be used for data retrieval.
`Range="data_range"`	(Optional) Defines the time frame for which the data is queried. It can be number of events, time frame or "PIOBJECT". "PIOBJECT" instructs the interface to use the time frame of the related PI batch/unitbatch/subbatch object. Examples: **`Range="10"`**: Retrieve the ten events that precede the triggered batch event timestamp. **`Range="10d"`**: Retrieve the events for ten days from the triggered batch event timestamp. **`Range="PIOBJECT"`**: Retrieve the events between the start and end times of the related batch object .
`Func="option"`	(Optional) Used with **Range** to aggregate retrieved data. Options: • "MIN": Minimum value in the time frame. • "MAX": Maximum value in the time frame. • "TOTAL": Sum of values in the time frame. • "MID": Average of values in the time frame.

Following are wildcards, which you can use to define a mask for template settings and placeholders, to match incoming data and format data to be written to the PI System.

Wildcard	Description
#	Single digit numerical value, 0-9
@	Single alpha character, a-z, A-Z
?	Any single symbol
!	Repeat the previous mask symbol
*	Any set of symbols

For example, to match any event that starts with "Repo", use the * wildcard as follows: `[EVENT, VALUE="Repo*"]`.

The interface supports a set of parameters that provide fine control over how incoming data is parsed. These advanced parsing parameters can be used in all types of templates. To enable you to incorporate these parsing parameters into a placeholder expression, the **Build a Placeholder** dialog provides an **Add Substring Parsing** option.

If you specify parsing parameters for one or more placeholders and no matching data is found, the entry is set to blank. The following table lists the parameters for parsing incoming data. The names of parameters, placeholders, and value substrings are not case-sensitive.

Parameter	Description
`LBE=”substring”`	Defines the left bound of the target substring value. The resulting substring does not include the specified substring.
`LBI=”substring”`	Defines the left bound of the target substring value. The resulting substring includes the specified substring.

Parameter	Description
RBE=”substring”	Defines the right bound of the target substring value. The resulting substring does not include the specified substring.
RBI=”substring”	Defines the right bound of the target substring value. The resulting substring includes the specified substring.
DELIM=”substring”	Specifies the field separator character or substring. Must be used in conjunction with the **COUNT** parameter. It narrows the results to the substring contained in delimiters, where the starting delimiter index is specified by the **COUNT** parameter. To parse the delimited substring, you can specify right and left boundary substrings.
COUNT=#	Index (position) of the delimiter from which to start parsing. Must be used in conjunction with the **DELIM** parameter.

To search all fields of an incoming event, specify a wildcard for the placeholder name (for example **`[*,lbe="u:"]`**).

The following table lists examples of parsing a field that contains the following data: `|U:browntod|C:SP_CHARGE_AMOUNT|O:1200|N:1123|E:kg|M:Local`

Example	Resulting Data
[value, lbe="N:"]	1123|E:kg|M:Local
[value, lbi="N:"]	N:1123|E:kg|M:Local
[value, rbe="tod"]	|U:brown
[value, rbi="tod"]	|U:browntod
[value, lbe="U:", rbe="|"]	Browntod
[value, lbi="U:", rbe="|"]	U:browntod
[value, lbe="O:", rbi="kg"]	1200|N:1123|E:kg
[value, delim="|",count=3]	O:1200
[value, delim="|",count=3,lbe="O:"]	1200
[value, delim="|",count=3,lbe="C:SP",rbe="UNT"]	_CHARGE_AMO
[value, delim="|",count=6,lbe="M:"]	Local

Create and update PI tags using tag templates

For details about specific tag template settings, see Tag templates.

Procedure

1. To create or update PI tags when specified events are read, you configure tag templates.
2. To create or update PI tags based on alarms read from an Emerson DeltaV Alarms & Events data source, you configure alarm tag templates.
3. To define tag templates using PI Event Frame Interface Manager, go to the **Templates** page and click the **Tag** tab.

4. To configure the name of the tag to be created or updated, you specify the **Name** field. To assign tag names based on incoming data, use placeholders.

   For example, to track phase module report events on a per-unit basis, you might configure the name as follows:

   ```
   [Unit] [phasemodule] Report
   ```

   With the preceding template, when the interface reads a report event for the NORTON phase module on unit XUNIT_52003, it replaces the placeholders with data from the specified fields and creates or updates a PI tag with the following name:

   ```
   XUNIT_52003 NORTON Report
   ```

   If the name structure contains placeholders, the tag template is triggered only if all the corresponding fields from the incoming event contain data (that is, are not blank).

   Different templates can update the same PI tag, if the templates' name structure resolves to the same tag. This capability enables you to write different values to the tag depending on the nature of the triggering event. For example, a value of 1 can be written to the tag when a unit procedure starts and a value of 0 can be written to the same tag when the unit procedure ends.

5. To specify the data to be written to the tag, you configure the **Value** field. To include data read from the data source in the tag value, use placeholders.

   For example, to simply record the incoming value without transforming it, specify the [PVAL] placeholder. A more complex example: to configure a value that concatenates phase module, event, description, incoming value and engineering units, specify the following:

   ```
   [PHASEMODULE].[EVENT].[DESCRIPT]: [PVAL] [EU]
   ```

   The preceding expression generates data like the following:

   ```
   CHARGE_DIW.Recipe Value.CPP_HIGH_LIMIT: 2535 kg
   ```

   Unlike placeholders in tag names, value placeholders can be replaced with empty fields from the incoming event, unless you use advanced field parsing to configure the value.

6. To update a tag when a particular event is read from the data source, specify the EVENT keyword in the **Name** field, as follows:

   ```
   [EVENT, VALUE="event_text"]
   ```

   This approach enables you to write different values to the tag depending on the text in the EVENT column.

   If you require a more refined approach, specify the incoming data that causes the template to be evaluated by configuring one or more triggers on the **Trigger** tab of the tag template.

   To configure the template to handle multiple different events, specify separate triggers ("OR" logic). To ensure that the template is triggered only when a set of multiple conditions are all detected ("AND" logic), specify a single trigger containing all the conditions. For example, to trigger the template only for system message events that are phase logic failures, specify the trigger as follows:

   ```
   [EVENT, value="System Message"] [DESCRIPT, value="Phase Logic Failure"]
   ```

   To ignore specified incoming values, use "!=" (not equal) . For example, to ignore undefined values, specify the following expression:

   ```
   [PVAL, VALUE!="UNDEFINED"]
   ```

   You can use wildcards to specify pattern-matching expressions in triggers.

7. To configure the tag template settings, specify settings as described in the following table:

Setting	Description
NAME	(Required) Name of PI tag to be created or updated.
VALUE	(Required) Value to be assigned to PI tag (text)
TRIGGER	Event text from data source (can be specified using wildcards)
TYPE	String/integer/float/auto. "Auto" directs the interface to automatically detect the data type.
UNITALIAS	Configure how unit alias (AF: PI point reference) is created. By default, the alias is created in the unit. To override the default, specify the path where you want the alias created. Example: `UNITALIAS = \Building1\Unit2|[PHASE]` The alias is created under the Unit2 module, named using the value of the [PHASE] column.
PHASEALIAS	Configure how the phase alias is created. By default, the alias is created in the phase module. To override the default, specify the path where you want the alias created.
DESCRIPTOR	Value for PI point descriptor attribute.
ENGUNITS	Engineering units
TRANSLATE	To enable translation, set to TRUE (default: FALSE)
ANNOTATION	Simple annotation to be written to the tag when the interface updates it.
ANNOTATION2	Structured annotation to be written to the tag when the interface updates it. For details about structured annotations, refer to the PI Data Archive System Management Guide.

8. To configure tag templates that catch events raised by the interface when it updates the PI Batch Database, specify the following placeholders in the TRIGGER setting of the tag template:

Placeholder	Values	Description
EVENT	**PIEVENT**	Specify [EVENT, value="PIEVENT"]
DESCRIPT	**BATCH** **UNITBATCH** **OPERATION** **PHASE** **PHASESTATE** **PHASESTEP**	Specify the batch level you want to trigger on. For example: [DESCRIPT, value="UNITBATCH"] [DESCRIPT, value="PHASE"]

Placeholder	Values	Description
PVAL	**START** **END**	Specifies whether to catch the start or ending event of the specified level: [PVAL, value="START"] [PVAL, value="END"]

For example, to detect the start of a batch, specify the following expression:

```
[EVENT, VALUE="PIEVENT"][DESCRIPT, VALUE="BATCH"][PVAL, VALUE="START"]
```

The following placeholders are supported when the triggering expression contains `[Parameter, value="PIEVENT"]`.

Placeholder	Batch Database	Event Frames
[BATCHID]	PIBatch and PIUnitBatch: BatchID property.	For a top-level event frame, "Name" property. For second-level event frame, "BatchID" attribute
[PROCEDURE]	PIBatch "Recipe" property	Event frame "Recipe" attribute
[UNITPROCEDURE]	PIUnitBatch "Procedure" property	Event frame "Name" property
[OPERATION]	PISubBatch "Name" property	
[PHASE]	Level 4 PISubBatch "Name" property	
[PHASESTATE]	Level 5 PISubBatch "Name" property	
[PHASESTEP]	Level 6 PISubBatch "Name" property	
[UNIT]	PIUnit "Name" property	AF element "Name" property

Update event frame attributes and batch properties using templates

To update event frame attributes or batch properties based on events, you define property or attribute templates using the PI Event Frame Interface Manager, on the **Templates** page. As in tag templates, you can use placeholder, advanced parsing and wildcards to specify the setting.

In event frames, attributes can be set at any level of the batch hierarchy. In the batch database, properties can be stored only at the root level of the batch hierarchy, in its properties collection. Each node in the collection is the name of the recipe level, (procedure, unit procedure, operation, or phase). Data is stored in name-value lists under each node.

Note:
The collection can store a maximum of 1Mb per PIBatch (no such limit for event frames). To conserve space, do not store all incoming events in a collection.

Every event name under the same node must be unique. If the template does not define a name, the interface assigns the name as "Event_<*event count*>", where <*event count*> is the number of events already stored under the node.

The following settings are supported for attribute and property templates:

Setting	Description
Index	Unique numeric ID (integer)
Name	Property name
Value	Value to assign to property
Trigger	Data source event that triggers updates
Translate	true/false (default: false)
EngUnits	Engineering units (AF only)
Type	`INTEGER`, `FLOAT`, `STRING` or `FLOATSTRING` (Default is string)
Category	Asset category (AF only)
UOM	Unit of measure (AF only)
Descriptor	Description (AF only)

If **Type** is set to `AUTO`, the interface assigns the data type for the attribute or property based on the first item of data that it receives. If the first value received is an integer but subsequent values are floating point, the interface creates an integer attribute or property and truncates any subsequent floating point values before storing them. To avoid truncation numeric input that includes a mix of integer and floating point values, specify `FLOATSTRING` rather than `AUTO`.

Procedure

1. To map the units of measure in the data source to the correct PI AF engineering units, configure an attribute template using PI Event Frames Interface Manager and select the UOM check box, then configure the mapping between the source units of measure and the UoM available in PI AF.
2. To write properties under the root UniqueID PIProperty node or root AF element, regardless of the level from which the triggering event originated, specify the $ symbol as the first element in name path, as follows: `$\[Parameter]`
3. To write properties under the root node, regardless of the level from which the triggering event originated, specify the @ symbol as the first element in name path as follows: `@\[Parameter]`

   The property/attribute is updated when a batch starts. For example, to create or update a property or attribute named TestTagCalc that contains a ten-day total for the "sinusoid" tag and a ten-day minimum for the "test_data_1" tag, configure the template with the following settings:

Setting	Set to
Name	TestTagCalc
Value	total:[Tag, name="sinusoid", range="10d", func="TOTAL"]
Trigger	[Event,value="PIEVENT"] [Descript,value="BATCH"] [Pval,value="START"]

   To populate the product property/attribute with the formula name when a "Recipe header" event arrives, ignoring undefined product codes, the template would use the following settings:

Setting	Set to
Product	[PVAL]

Setting	Set to
ProductTrigger	[EVENT,VALUE="Recipe Header"] [DESCRIPT,VALUE="Product Code"] [PVAL, VALUE!="UNDEFINED"]
ProductTrigger	[EVENT,VALUE="Formula Header"] [DESCRIPT,VALUE="Formula Name"]

Handle Emerson DeltaV alarms and events using templates

You can configure the interface to update PI points when specified events from an Emerson DeltaV Alarms & Events data source are read.

Procedure

1. To configure the triggering events and the data to be written, you define an **alarm tag template.**
2. To define an alarm tag template using PI Event Frame Interface Manager, go to the **Templates** tab and click **Alarm Tags**.

   You specify alarm tag templates the same way as tag templates.

   To handle alarm events you must configure at least one alarm tag template. For details, see Templates.

   The following table lists the placeholders that you can use in alarm tag templates.

Setting	Allowed Placeholders	Description
VALUE	Same as for Name setting, plus **[TIME]**, **[TAG]**	(Required) Specifies the event value structure for the specific PI Point. Allowed placeholders are not case sensitive. The event timestamp is taken from the incoming event [TIME] field.
TRIGGER	Same as for Name setting	(Optional) Specifies an event that triggers evaluation of the template. If you omit column name, the interface checks the incoming EVENT field for the specified text. To check a different field, specify the field name. For example: TAG[1].TRIGGER=[Pval, value="CREATED"]

Setting	Allowed Placeholders	Description
ALIAS	Same as for Name setting	(Optional) Defines how the unit-level alias name is assigned for the specific template tag. Alias names must be unique. By default, the interface uses the NAME property for the module-level alias name, and the alias is created in the phase module as a submodule of the unit. To configure a different location, specify the path of the desired location. Separate parent and child modules using backslashes, and separate the module path from the alias name using the pipe (|) symbol as follows: [Unit]\Phase|AliasName
DESCRIPTOR	Same as for Name setting	(Optional) Define how the tag's Descriptor attribute is to be set.
ENGUNITS	Same as for Name setting	(Optional) Defines the EngUnits for the specific PI Point.
TRANSLATE	Must be `true` or `false`	(Optional) To enable any translations configured for Name, Value, Alias, Descriptor and EngUnits, set to `true`.

For example, to store all alarm and events coming from an Emerson DeltaV Alarms & Events data source and create the module/AF element and alias automatically, use the following settings:

Setting	Set to
NAME	Alarm Test [MODULE]
VALUE	[EVENT] | [CATEGORY] | [AREA] | [PROCESSCELL] | [UNIT] | [MODULE] | [MODULEDESC] | [ATTRIBUTE] | [STATE] | [LEVEL] | [DESC1] | [DESC2]
TRIGGER	[EVENT]

To store alarms and events with level 5 or 10 to 19 in a single global alarm tag, the template can use the following settings:

Setting	Set to
NAME	Alarm Test Global
VALUE	[EVENT] | [CATEGORY] | [AREA] | [PROCESSCELL] | [UNIT] | [MODULE] | [MODULEDESC] | [ATTRIBUTE] | [STATE] | [LEVEL] | [DESC1] | [DESC2]
TRIGGER	[EVENT] [level, value="1#*"]
TRIGGER	[EVENT] [level, value="5*"]

To create an integer tag for each combination of module and category using advanced parsing to store only numerical values embedded in the Desc2 column for Event type "CHANGE" when the Attribute column contains an "ALM_COUNTER" substring, the template can use the following settings:

Setting	Set to
NAME	Test [MODULE] [CATEGORY]
VALUE	[DESC2,LBE="VALUE = "]
TYPE	INTEGER
TRIGGER	[EVENT, VALUE="CHANGE"] [ATTRIBUTE, VALUE="ALM_COUNTER*"]

Override incoming recipe names using recipe templates

To override the recipe data read from the data source, you define **recipe templates.**

Procedure

1. To create a recipe template using PI Event Frames Interface Manager, go to the **Templates** tab and click **Recipe.**

   The following recipe template settings are supported:

Setting	Description
NAME	(Required) Defines the convention used by the interface to assign names to procedures, unit procedures, etc. You can use the advanced parsing parameters to define this field. Example: **abc_[PROCEDURE]** If the procedure field of the incoming event contains "Test", the corresponding Recipe field is set to "abc_Test"
BATCHID	(Optional) Specifies the batch ID for the procedure or unit procedure.
MODULEPATH	(Optional) For unit procedures (level 2) or phase (level 4), specifies where the recipe resides in the PI Module Database or PI AF element hierarchy. In the Module Database, the path specifies the location of the PIUnit for the unit batch.
PRODUCT	(Optional) Set the product for the recipe. Sets the Product field for procedures and unit procedures. To set the product field to the value read from the data source, specify the following placeholder: **[PRODUCT]**

Setting	Description
PRODUCTTRIGGER	(Optional) Sets the product for the recipe after the recipe object is created. Intended for use when the product is defined in a separate event. If a product trigger is defined, the product is defined by the event that satisfies the trigger. If no product trigger is defined, the product gets its value from the event that created the recipe, and the template is populated by the event's placeholder data. Example: **[Parameter, Value="Recipe Header"] [Descript, value="Product Name"]**
TRANSLATE	(Optional) To enable translation, set to TRUE. Default: FALSE
MERGE	(Optional) To merge identically-named objects under the same parent, set to TRUE. Default: FALSE

2. To specify the level of hierarchy for the template, set the **Index** field on the **Configuration** page as follows:

Batch Database Field	Default Name	Index
PIBatch Recipe	Procedure	1
PIUnitBatch Procedure	UnitProcedure	2
PISubBatch Name	Operation	3
PISubBatch Name	Phase	4
PISubBatch Name	PhaseState	5
PISubBatch Name	PhaseStep	6

3. To configure additional settings for event frames, the following settings support the placeholders for the NAME setting. These settings are valid only for event frames.

Template Name	Description
Descriptor	(Optional) Specifies the Event frame descriptor property for the particular source Recipe object.
Category	(Optional) For each recipe level, defines the event frame category. If the event that creates an event frame contains insufficient information, no category is assigned. To assign a category to an event frame after its creation, use `Category[x]`.

Template Name	Description
Category[x].Name	(Optional) For each recipe level, define the event frame category based on an event that is related to the particular recipe item. This setting can create as many categories as desired. The index is a positive integer that associate the Name and Trigger subproperties for the specific `Category[x]` property. If the AF category does not exist, the interface creates it. To use this setting, you must also specify the triggering event using the Recipe[#].Category[x].Trigger setting. Example: When the specified trigger event arrives, create an event frame, assigning it the SCR category. `Category[10].Name = SCRCategory[10].Trigger = [Descript, value="Formula Name"] [Pval, value="SCR 20051"]`

Category[x].Trigger	(Optional) Defines the expression that triggers assignment of a category by the Recipe[#].Category[x] setting. There can be multiple triggers for a single Recipe[#].Category[x].Name. To use this setting, you must also specify the category to be assigned, using the Recipe[#].Category[x].Name setting.
	Example: When any of the specified trigger event arrives, create an event frame, assigning it the SCR category.
	`Category[10].Name = SCR`
	`Category[10].Trigger = [Descript, value="Formula Name"] [Pval, value="SCR 20051"]`
	`Category[10].Trigger = [Descript, value="Formula Name"] [Pval, value="SCR 20051_01"]`
	`Category[10].Trigger = [Descript, value="Formula Name"] [Pval, value="SCR 20051_02"]`

Template	(Optional) For each recipe level, specify the event frame template. If the interface cannot find a matching event frame template, the template is left blank. To assign a template to an event frame after its creation, use the `Template[x]` property

Template Name	Description

Recipe[#].Template[x].Name	(Optional) For each recipe level, this dynamic property enables you to define the event frame template. based on any event that is related to particular recipe item. This property can assign only one AF template to a particular event frame. The interface uses the first matching Recipe[#].Template[x] property to be assigned to an event frame. The index is a positive integer that associates the Name and Trigger subproperties for the Template[x] property. If you specify this property, you must specify the Trigger property.
	Example:
	`Template[10].Name = BATCH_A`
	`Template[10].Trigger = [Descript, value="Formula Name"] [Pval, value="SCR 20051"]`

Recipe[#].Template[x].Trigger	(Optional) This property defines the triggering expression for the event frame template. There can be multiple triggers for a single recipe template. If you specify this property, you must specify the Name property.
	Example:
	`Template[10].Name = BATCH_A`
	`Template[10].Trigger = [Descript, value="Formula Name"] [Pval, value="SCR 20051"]`
	`Template[10].Trigger = [Descript, value="Formula Name"] [Pval, value="SCR 20051_01"]`
	`Template[10].Trigger = [Descript, value="Formula Name"] [Pval, value="SCR 20051_02"]`

The following placeholders are supported:

- AREA
- BATCHID
- DESCRIPT
- EU
- EVENT
- OPERATION
- PHASE
- PHASEMODULE
- PHASESTATE

- PHASESTEP
- PROCEDURE
- PROCESSCELL
- PVAL
- UNIQUEID
- UNIT
- UNITPROCEDURE
- USERID
- [*,value= "Field"]
- [*,value= "Mask"]

Filter incoming events

To exclude events from being processed by the interface, you can configure **filters**.

You can filter out data based on its level in the batch hierarchy and on its unit. The interface examines fields in incoming events for the specified data and, if it matches, the event is not processed.

Procedure

1. To exclude incoming data from processing using PI Event Frames Interface Manager, go to the **Filters** tab.
2. Expand the setting for the type of data that you want to filter (phases, units, recipes or phase states).
3. Specify the data that you want to exclude.

   You can use wildcards to define masks for matching.

Translate foreign language data

To process data from non-native language data sources, you can translate strings from the source language to the target language.

Procedure

1. To configure a translation using PI Event Frame Interface Manager, open the desired template.
2. Select the **Translate** check box and click **Translations**.

   The Translations window opens.
3. Specify the desired source and target text.

   Translations are not case-sensitive.

How batch data is stored

PI Interfaces for Batch Execution Systems (BES) and Manufacturing Execution Systems (MES) provide two options for storing batch data: the PI Batch Database or PI Event frames and assets. The batch database is a special-purpose database, optimized for (and constrained by) S88 and similar models, whereas AF event frames and elements provide a highly flexible, general-purpose structure that enables you to define your asset model and event hierarchy as you wish. Most notably, the batch database imposes a one-unit-batch-per-unit restriction, whereas event frames do not restrict the number of unit batches that can be concurrently active in the same unit.

The PI Batch Database

The PI Module and Batch Databases are used to organize and store batch data. Each object in the PI Batch Database represents a specific level of the S88 Recipe Model. To represent recipes, unit procedures, operations, phases, phase states and phase steps, the interface creates PIBatches, PIUnitBatches and a hierarchy of PISubBatches in the PI Batch Database.

In the PI Batch Database, each level of the hierarchy is recorded as a module. All levels record their start and end time. PIBatch and PIUnitBatch record the batch ID. In a PIBatch, the Recipe property contains the recipe name. In a PIUnitBatch, the Procedure property contains the name of the corresponding recipe level. If the highest level that a recipe contains is an operation or phase (that is, neither the procedure nor unit procedure levels are defined), the interface creates a parent PIBatch and PIUnitBatch.

To view batches in the PI Batch Database, use the PI System Management Tools **Batch>Batch Database** option. The following sections describe how the different levels of the batch hierarchy are stored in the Batch Database.

- **PIBatch**

  The interface creates a PIBatch for each batch found in the data source. The PIBatch corresponds to the procedure in the recipe. PIBatches are not associated with a specific piece of equipment. Each PIBatch contains one or more PI UnitBatches that correspond to the unit procedures in the recipe.

  The root property node is named using the batch ID, which is assigned by the BES. The interface stores the following batch data in properties under the root node:

  - BatchID
  - Product
  - Formula Name
  - Recipe
  - Recipe Type
  - Start Time UTC
  - End Time UTC
  - Interface Name
  - Interface ID

- Data Source
- Data from events captured by configuring properties templates

The properties collection is organized according to the hierarchy of the recipe model from the data source. Events of interest are stored in lists under the appropriate recipe node. Each property under the same node must have a unique name. By default, each property is named "Event_<event count>", where <event count> is the current number of events already stored under a specific node. The PIProperty value can be configured using property templates.

Multiple source batches that have identical IDs or a common subset of characters in their ID can be merged into a single PIBatch. The product and recipe properties contain data associated with the first source batch that started the merged PIBatch. The original source batch properties are stored in PIProperties under a node named using the unique ID of the source batch. For each merged source batch, the interface creates a node that is named using the unique ID of the source batch containing the original properties.

Because a source batch can terminate unexpectedly without proper unloading by the operator, the interface caches batches in local memory for a timeout period, after which the batch is considered abandoned. The interface closes abandoned batches, assigning end time using the latest known timestamp for the batch. The default timeout is 100 days. To change the default timeout, edit the interface settings using PI Event Frame Interface Manager.

- **PIUnitBatch**

  A PIUnitBatch is created for each unit procedure recorded in the data source. The start and end times of a PIUnitBatch are intended to reflect the start and completion of physical processing in a unit, so they require both a start event and an "equipment acquired" event. The unit batch ends when an "end" event or an "equipment released" event is detected.

  By default, PIUnitBatches contain the batch ID and procedure name from the data source. To override the default, use PI Event Frame Interface Manager to edit the settings. When operation or phase-level recipes are run, the interface uses the operation or phase name as the PIUnitBatch procedure name.

  If unit procedures on the same unit overlap, the interface closes the conflicting PI UnitBatches. The actual end time for a truncated unit batch is stored in its product property, and the product is appended with the text "_TrueEndUTC=", followed by the actual end time of the unit batch, specified as UTC in seconds.

- **PISubBatches (Operation, Phase, Phase State and Phase Step)**

  In the batch database, all levels below unit batches are recorded using a hierarchy of subbatches, as follows:

  - **Operation**: The interface creates a PISubBatch for each child of a unit batch encountered in the data source.
  - **Phase**: The interface creates a PISubBatch for each child of an operation encountered in the data source.
  - **Phase State**: The interface creates a PISubBatch for each child of a phase encountered in the data source. The start of new phase state ends the previous phase state, except for the COMPLETE, ABORTED and STOPPED states, which set only the end time.
  - **Phase Step**: If you enable this features, the interface creates a PISubBatch for each child of a phase state encountered in the data source.

Subbatches populate parent levels of the hierarchy with the name of the source operation, phase, phase state and phase step as required.

Phase steps are not S88-compliant and support for phase steps varies among BES vendors. (ABB800xA, Foxboro IA Series, Siemens SIMATIC, and WonderWare InBatch do not support phase steps.) Phase steps are created beneath the first phase state subbatch named "RUNNING", regardless of whether the parent phase state is ended. The name of phase step start and stop events is read from the data source [DESCRIPT] column. The triggering event is "Report". If the parent phase is not found, phase steps do not trigger the creation of PIBatches, UnitBatches and SubBatches. If the phase step was not closed by an appropriate closing event, the interface closes it by the end of its parent operation-level PI SubBatch. Zero-duration phase steps are ignored.

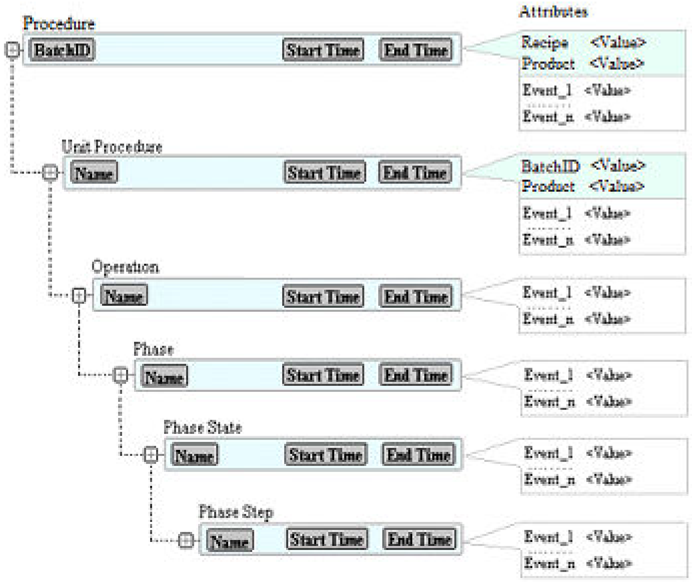
Event frames

As an alternative to the PI Batch Database, you can use event frames to record batch data. Unlike the PI Batch Database, which records a predefined and fixed set of data at each level, you can define the attributes to be stored in event frames at every level of the hierarchy.

Each event frame contains the following fields:

- Name
- Description
- Start time
- End time
- Template
- Category (Default is "OSIBatch")
- Event-specific attributes
- Referenced elements such as unit or phase module

To view batch event frames and AF elements generated by the interface, use PI System Explorer. The following figure illustrates the event frame hierarchy that the interface creates.



In the root-level event frame, the Name field contains the batch ID from the data source. In lower-level event frames, the Name field contains the recipe name. In event frames for a procedure-level recipe, the product and recipe properties from the data source are stored as attributes. For unit procedure-level recipes, the batch ID and product from the data source are stored as attributes.

If the highest recipe level is an operation or phase (that is, neither procedure nor unit procedure levels are defined), the interface creates event frames that correspond to the procedure and unit procedure level.

The following sections explain how data is stored in event frames for each level of the batch hierarchy.

- **Procedure**

  For each batch in the data source, the interface creates a root-level event frame that represents the procedure in the recipe. Each root-level event frame contains a collection of child event frames that correspond to unit procedures. Data from source batch events can be recorded in the attributes of the event frame and in PI points. Source batches can have identical IDs and recipe names in the same time frame. To match the source batch with an event frame, the interface stores additional information in the extended properties of the root event frame. The Name property contains the ID of the source batch, and the Value property contains an XML structure composed of the following batch data:

  - Batch ID
  - Product (searchable)
  - Formula name
  - Recipe (searchable)
  - Recipe type
  - Start time UTC
  - End time UTC
  - Interface name

- Interface ID
- Data source

The following table shows how the source batch properties map to event frame attributes.

Source Procedure Properties	Event Frame Fields	Event Frame Attributes
BatchID	Name	
Procedure Name		Recipe
Product		Product
Start Time	Start Time	
End Time	End Time	
	Template="Procedure"	Recipe, Product

By default, the interface captures the following batch-associated events and stores them in procedure-level event frame attributes:

- Recipe header
- Formula header
- Recipe value
- Report

For the preceding attributes, the name is assigned from the source [DESCRIPTION] column, the value from the source [PVALUE] columns, and the unit of measure attribute from the [EU] column.

The procedure-level event frame can represent merged source batches. The product and recipe attributes contain data associated with the source batch that started the merged event frame. For each merged source batch, the interface creates an entry node in the event frame’s extended properties, named using the unique ID of the source batch, with its value containing an XML structure composed of the original source batch properties.

- **Unit Procedure**

  The interface creates a unit procedure-level event frame for each unit procedure read from the data source. Each unit procedure-level event frame is a child of the procedure-level event frame and contains the subset of event frames that represent the source batch operation-level recipe. The start and end times of an event frame record the start and end of physical processing in a unit.

  The name field of the unit procedure-level event frames contains the unit procedure name as read from the data source. The batch ID and product properties are searchable attributes of the event frame. The following table shows how the source batch properties map to event frame attributes:

Source UnitProcedure Properties	Event Frame Fields	Event frame Attributes	Referenced Elements
BatchID		BatchID	
UnitProcedure Name	Name	Procedure	
Product		Product	
Start Time	Start Time		
End Time	End Time		

Source UnitProcedure Properties	Event Frame Fields	Event frame Attributes	Referenced Elements
Unit			Unit
	Template="UnitProcedure"	default Attributes: BatchID, Procedure, Product	

In addition to batch ID, procedure and product attributes, the interface records recipe and report events in event frame attributes. For these events, the attribute name is assigned from the [DESCRIPTION] column, the value from the source [PVALUE] column, and units of measure from the [EU] column.

Unit procedure-level event frame properties do not change if the parent object is a merged event frame. By default, unit procedure event frames contain the batch ID and procedure name read from the data source. (To override the default, use PI Event Frame Interface Manager to configure the **Batch ID mask** field on the **Batch Settings** page.)

When operation or phase-level recipes are run, the interface uses the operation or phase name as the name of the unit procedure-level event frame.

- **Operation**

  The interface creates an operation-level event frame for each operation read from the data source. Each operation-level event frame is a child of the unit procedure-level event frame and contains the subset of event frames that represent the source batch phase-level recipe.

  The name field of the operation-level event frames is the operation name read from the data source. The following table shows how the source batch properties map to event frame attributes:

Source Operation Properties	Event Frame Fields	Referenced Elements
Operation Name	Name	
Start Time	Start Time	
End Time	End Time	
Unit		Unit
	Template="Operation"	

  By default, the interface records recipe value and report events in event frame attributes. For these events, the attribute name is assigned from the [DESCRIPTION] column, the value from the source [PVALUE] column, and units of measure from the [EU] column.

- **Phase**

  The interface creates a phase-level event frame for each phase read from the data source. Each phase-level event frame is created as a child of an operation-level event frame and contains the subset of event frames that represent the source batch phase states-level recipe.

  The name field of the phase-level event frame contains the phase name read from the data source. The following table shows how the source batch properties map to event frame attributes:

Source Phase Properties	Event Frame Fields	Referenced Elements
Phase Name	Name	

Source Phase Properties	Event Frame Fields	Referenced Elements
Start Time	Start Time	
End Time	End Time	
Unit		Unit
Phase Module		Phase Module
	Template="Phase"	

By default, the interface records recipe value and report events in event frame attributes. For these events, the attribute name is assigned from the [DESCRIPTION] column, the value from the source [PVALUE] column, and units of measure from the [EU] column.

- **Phase State**

  The interface creates a phase state-level event frame for each phase state read from the data source. Each phase state-level event frame is created as a child of a phase-level event frame and, if so configured, can contain the subset of event frames that represent the phase steps. The start of new phase state ends the previous one, unless the new state is COMPLETE, ABORTED or STOPPED, which end the current phase state without beginning a new one. These phase states have a zero-duration time frame.

  The name field of the phase state event frames reflects an actual source recipe phase state name. Below is the mapping of source phase state to event frame fields and attributes:

Source Phase State Properties	Event Frame Fields	Referenced Elements
Phase State	Name	
Start Time	Start Time	
End Time	End Time	
Unit		Unit
Operation Module		Operation Module
Phase Module		Phase Module
	Template="Phase State"	

- **Phase Step**

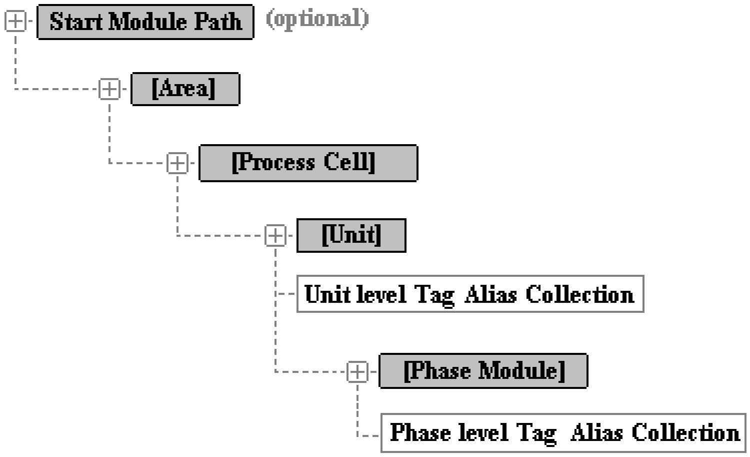
  Phase steps are not S88-compliant and support varies among BES vendors. (ABB800xA, Foxboro IA Series , Siemens Simatic , and WonderWareInBatch do not support phase step.) Phase steps are created beneath the first phase state sub-batch named "RUNNING", regardless of whether the parent phase state is ended. The name of phase step start and stop events is read from the data source [DESCRIPT] column. The triggering event is "Report". Phase steps do not trigger the creation of parent-level events if the parent phase is not found. If the phase step was not closed by an appropriate closing event, the interface closes it by the end of its parent operation-level event frame. Zero-duration phase steps are ignored.

Equipment hierarchy

When recording batch data from the data source in the PI Batch Database, the interface creates PIUnits (modules representing equipment) as required. If you configure the interface to create Event frames, it creates AF elements representing the equipment hierarchy. The following levels of the equipment hierarchy are created:

- Area
- Process Cell
- Unit
- Phase Module

The following diagram shows the default equipment hierarchy generated by the interface.



By default, the hierarchy is located at the root level of the PI Module Database or the PI AF database, depending on whether you are creating PIBatches or event frames. To specify a different root, use PI Event Frame Interface Manager to configure the start module path.

Unit aliases are created for a tag if the template includes the [UNIT] placeholder, phase aliases are created if it contains the [PHASEMODULE] placeholder. Unit aliases are also created if the tag name contains the unit name (for example, if the unit name is UNIT202 and the tag name is "UNIT202 report tag", a unit alias is created; likewise for phase modules.

Interface modes

PI Interfaces for Batch and Manufacturing Execution Systems can be run in five different modes:

- RealTime (default)
- Recovery
- Preprocess
- Statistics
- Delete

RealTime Mode

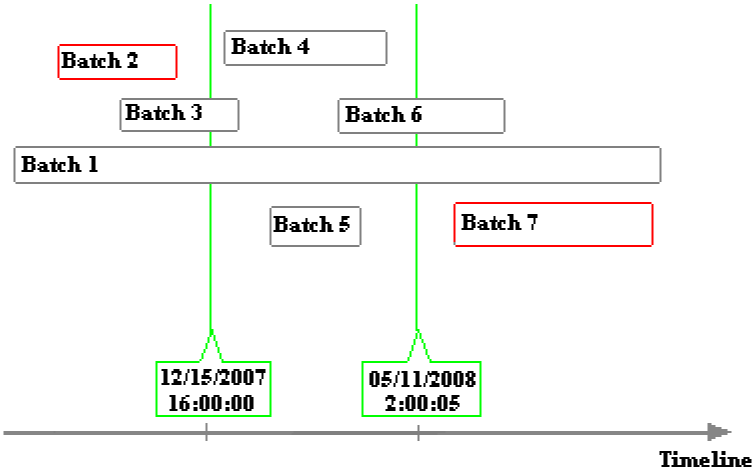
In **RealTime** mode, the interface monitors the data source for events that indicate the start or end of a batch or any child level thereof, recording these events in the PI System according to your batch configuration. The interface records newly-acquired data at the end of each scan regardless of whether batches are completed on the source. At startup, before it begins real-time data collection, the interfaces attempts to recover any data written after it was shut down.

Recovery Mode

To recover events that occurred during interface downtime, the interface scans the data source for a specified period. If you omit an end time or specify "*" (current time), the interface recovers data and then starts collecting data in **RealTime** mode. If you specify an end time, the interface recovers data for the specified period and then exits.

In recovery mode, the interface reads batch data from the data source for the specified time period. This mode can be used to initialize the PI System with historical data from the data source. If batch data for the specified period already exists in the PI System and the interface detects discrepancies, it attempts to correct the PI System data, logging any errors.

For example, the following figure shows a data source that contains batch data for seven batches.



If you recover data for the period from 12/15/2007 16:00:00 through 05/11/2008 2:00:05, the interface recovers contained batches (Batch 4 and 5) as well as border batches (Batch 1, Batch 3 and Batch 6). Batches outside the time frame (Batch 2 and 7) are not recovered.

Preprocess Mode

If your data source contains data with timestamps that are earlier than the period covered by the primary archive, you can recover events by running the interface in **Preprocess** mode, which scans the data source and creates the required tags, modules and units in the PI System. After running the interface in **Preprocess** mode, you must reprocess older archives to create entries for the tags, modules and units, then run the interface in recovery mode. This process is also referred to as "backfilling." (See the PI Data Archive System Management Guide for details on reprocessing archives.)

Statistics Mode

In **Statistics** mode, the interface scans a specified period, comparing data from the data source with the corresponding data in the PI Batch database. After comparing data, the interface reports the results and exits. If you omit an end time, the interface scans from the specified start time until the current time. To run the interface in Statistics mode using PI Event Frames Interface Manager, go to the **Operation Settings** tab, choose **Statistics** mode and specify the start and end times for the period to be analyzed. To analyze data through the current time, omit the end time. The interface logs results to the specified output file and exits.

Delete Mode

To delete batch data from the PI Batch Database or from event frames, run the interface in **Delete** mode, specifying the time period to be deleted. To run the interface in **Delete** mode using PI Event Frames Interface Manager, go to the **Operation Settings** tab, choose **Delete** mode and specify the start and end times for the period to be deleted. To delete data through the current time, omit the end time. If you intend to recover events from the data source for a specified time period, consider deleting existing batch data for that time period first.

ABB 800xA interface

This interface is intended for use with ABB 800xA and later systems that use Oracle 9.2.0.7.0 and above as the batch historian.

Current version number of this interface is 3.0.16.470. The OSIsoft part number for this interface is PI-IN-ABB-800BAT-NTI.

ABB 800xA data source

To enable the interface to access the Oracle server where ABB 800xA stores batch data, you must install Oracle OLEDB Provider on the interface node prior to starting the interface. Be sure to install the version of OLEDB Provider that corresponds to the version of Oracle you are running.

For each Oracle data source, you must configure the Net Service Name (set using Oracle Net Manager), Oracle database, username and password required to connect. You configure these settings using PI Event Frame Interface Manager, on the **Source** page.

The interface reads batch event data from the following tables in the data source.

View/Table	Description
Task	Contains UniqueID (TaskID), BatchID, start time (UTC), end time (UTC) Procedure Path.
Resource_Associations	Contains Units associated with all batch recipes for Unit Procedure level (based on TaskID).
Task_Variables_Occurrences	Contains batch-associated data for all batches, such as Variable Name, Variable Value, Occurrence Time (UTC).
PDLMSGLOG	Contains batch-associated alarm data.

This batch interface enables you to recover batches from restored archives in the ABB 800xA data source. To enable recovery from restored archives, launch PI Event Frames Interface Manager. On the **Source** settings page, check **Collect batches from restored archives.** Note that, if you have configured multiple data sources, the interface recovers batches from restored archives in all the data sources that you configured.

ABB 800xA batch start and stop events

The ABB system does not provide phase state changes or phase steps.

PIBatch/Procedure

The start and end times for the batch is taken from the Task event in which the [LevelNumber] field contains "1". The event is retrieved using a complex query against the following tables:

- task
- task_variables_occurrences (for recipe name)

PIUnitBatch/Unit Procedure

The start and end times for a unit procedure are taken from the Task event in which the [LevelNumber] field contains "1" or "2" (depending on the unit associations). The event is retrieved using a complex query performed against the following tables:

- task
- task_variables_occurrences (for recipe name)
- resource_associations (for allocated equipment).

Operation

The start and end times for an operation are taken from the Task event in which the [LevelNumber] field contains "2" or "3" (depending on the unit procedure level). The event is retrieved using a complex query performed against the following tables:

- task
- task_variables_occurrences (for recipe name)
- resource_associations (for allocated equipment)

Phase

The start and end times for a phase are taken from the Task event in which the [LevelNumber] field contains "4" or "5" (depending on the unit procedure level). The event is retrieved using a complex query performed against the following tables:

- task
- task_variables_occurrences (for recipe name)
- resource_associations (for allocated equipment)

ABB 800xA template placeholders

The interface supports the following placeholders for defining template settings.

- [BATCHID]
- [OPERATION]
- [PARAMETER]
- [PHASE]
- [PROCEDURE]
- [TIME]
- [UNIQUEID]
- [UNIT]
- [UNITPROCEDURE]
- [VALUE]

Emerson DeltaV interfaces

The Emerson Delta V interface is designed to be used with DeltaV batch execution systems, including an Emerson DeltaV Alarms & Events data source. This interface can use the DeltaV Batch Historian (Microsoft SQL Server) or DeltaV event journals as a data source. For DeltaV 8.4 systems, this interface can use only DeltaV event journals as the primary data source. The use of DeltaV event journals as a public interface for the DeltaV System is not recommended by Emerson.

Currently shipping version of the DeltaV interface is 3.0.15.418. The OSIsoft part number for this interface is PI-IN-EM-DVB-NTI.

This interface is primarily designed to be used for DeltaV 10.3 and later systems utilizing the DeltaV OPC A&E Server and the DeltaV Batch Historian. For DeltaV 9.3 systems this interface can utilize the DeltaV Batch Historian or DeltaV event files as the primary data source. For DeltaV 8.4 systems this interface can only use DeltaV event files as the primary data source.

Emerson data sources

The interface can collect data from event journal files and from SQL Server databases. A single interface instance can collect data from multiple data sources, provided the data sources are all the same type (all event journals or SQL Server database).

Note:
In version 1.0.0.0, the interface reads from the Descript column of the batcheventview view. This column is a concatenation of a description, a parameter value, and the engineering units. In version 1.0.1.0, the interface has been revised to read the fields from the underlying source tables rather than the view, which is a simpler, more reliable approach. If you require backward compatibility with version 1.0.0.0 of the interface to generate your batch data, enable the **Use original batch event view** option.

Event journals

Event journals are text files in which the BES logs batch events. For the Emerson DeltaV systems, the interface expects each line in the event journal to be composed of the following tab-delimited fields, in the order specified:

- [TIMESTAMP] (local or GMT)
- [BATCHID]
- [RECIPE]
- [DESCRIPT]
- [EVENT]
- [PVALUE]
- [EU] (engineering units)
- [AREA]
- [PROCCELL]

- [UNIT]
- [PHASE]
- [PHASEDESC]
- [USERID]
- [UNIQUEID]
- [COMMENT]

In event journals, the product ID information is stored in the [PVALUE] field, in the row that contains the description "Product Code". Typically this is a recipe header event.

If the BES indicates the product ID using a value other than "Product Code," you must translate the value to "Product Code" to ensure that the interface detects it. To configure the translation using PI Event Frame Interface Manager, check the **Translation** field on the template's **Configuration** tab and specify the source and target string on the **Translations** dialog.

DeltaV Batch Historian (SQL Server)

The default name of the database where the DeltaV Batch Historian records events is "DVHisDB". The following table lists the tables and views that the interface reads and the data that it reads from each table.

View/Table	Data Read by Interface
batchview	UniqueID, BatchID, start time, end time, Product, UniqueID and archived flag with new archive database name for all batches.
brecipestatechangeview	State-change events that are used by default for PI batch generation.
batchrecipeview	Recipe data for all batches, such as Procedure, Unit Procedure, Operation, Phase, equipment linkage, start and end time for each object.
batchequipmentview	Equipment arbitration for all batches.
batcheventview	Batch-associated data for all batches. Read if the **Use original event batch view** option is enabled. This view does not provide separate [DESCRIPT], [PVAL] and [EU] fields: the [DESCRIPT] field contains all three fields combined.
localevars	Read to determine the time zone offset.

View/Table	Data Read by Interface
Union of tables: • bactivestepchangeevent • bequipmentselectionevent • bmaterialchargeevent • bmaterialchargerequestevent • bpausestatusevent • bphasebatchrequestevent • bphaselinkpermissiveevent • brecipecomment • brecipemodechangeevent • brecipemodecommandevent • brecipestatechangeevent • brecipestatecommandevent • brecipevaluechangeevent • brecipevaluerequestevent • breportevent • btextmessageevent • unhandledbatchmsg	Batch-associated data for all batches. These tables are used by default to retrieve batch-associated events. The union of the tables provides the batch-associated events with the [DESCRIPT], [PVAL] and [EU] fields.

Alarms & events

The interface can be configured to read alarms and events data from an Emerson DeltaV Alarms & Events data source and an OPC alarms and events server. To read from an Emerson DeltaV Alarms & Events data source, you must install the Microsoft SQL Native Client on the interface node. If the interface loses its connection to an Emerson DeltaV Alarms & Events data source, the interface collects all batch data from the batch historian until it succeeds in reconnecting to the Emerson DeltaV Alarms & Events data source.

The OPC alarms and events server provides real-time data. The Delta V Alarms & Events Historian stores its data using Microsoft SQL Server. The default alarms and events database server is named "DELTAV_CHRONICLE" and the default database is named "EJournal". The interface reads events from the Journal table.

To configure an alarms and events data source using PI Event Frames Interface Manager, perform the following steps:

1. On the **Data Source** tab, right-click the **Sources** node and choose **Add SQL Source.** The **Configuration** page is displayed.
2. Check **Use Alarm and Events Historian.**
3. To specify mappings that ensure that process cell data is recorded correctly (DeltaV does not emit process cell information), click **Area to Process Cell…** or **Unit to Process Cell…** and specify the process cell to be recorded with its events.
4. To configure the interface to use the equipment hierarchy XML file that is generated by DeltaV, specify the path to the XML file in the **DeltaV equipment hierarchy** field.
5. To configure an OPCAE server as a data source to be read in conjunction with an Emerson DeltaV Alarms & Events data source, click the DeltaV OPCAE disclosure and specify the host

node and server name. When you configure an OPCAE server, the DeltaV Batch Historian serves as a backup source and the source for additional batch-associated events.

The interface starts a batch when it reads the BATCH-EVENT event with its event attribute [6] = "LOAD" (using a 0-based index).

The interface creates the batch when it identifies the recipe type of the batch, which normally occurs when the recipe is loaded and started. The interface starts a batch when it reads the BATCH-EVENT event with its event attribute [6] = "REMOVED". The recipe structure is read from the events that trigger the start and end of levels, such as Procedure Started/Finished, UnitProcedure Started/Finished, etc.

For all levels below batch, the interface checks event attribute [6] for the value "State Change" and parses event attribute [8] to determine whether the event was a start or end event and to find out what level of the batch hierarchy is affected.

The interface starts a level when it reads the BATCH-EVENT event containing the following data:

- Event Attribute [6] = "State Changed"
- Event Attribute [8] = <batch recipe hierarchy> RUNNING

The interface end a level when it reads the BATCH-EVENT event containing the following data:

- Event Attribute [6] = "State Changed"
- Event Attribute [8] = <batch recipe hierarchy> COMPLETED or ABORTED or STOPPED

No phase step data is available from the OPC alarms and events server.

Syncade

Emerson Syncade creates and executes recipes on DeltaV batch execution systems. The interface requires Syncade version 4.0.1 or higher.

Emerson Syncade

For each level, Syncade records the start time in the [StartUtcDateTime] timestamp and end time in the [EndUtcDateTime] timestamp of the object in which the event is recorded. For phase- and operation-level recipes, the interface creates parent procedures and unit procedures, using the start and end times of the operation or phase to start and end the parent levels. Syncade does not provide phase state or phase step data.

The batch ID comes from the batch object's [OrderNumber] property, and the batch recipe type is provided for each object by the data source.

Emerson batch start and stop events

The following sections describe how the interface determines the start and end times of batch events based on the data read from the data source. The start and end time for the batch event are taken from the data read from the data source. The precise logic that the interface uses depends on the data source and on whether you enable the **Use batch recipe** option.

Default

To set the start time for a unit batch, the interface requires events indicating that the unit procedure has started and the required unit has been acquired. Batches and unit batches are also created when the interface processes operation- and phase-level recipes. In both cases, the unit batch start time is taken from the start event or the arbitration event in which the unit was acquired, whichever is later. The end time is taken from the end event or the event that releases the unit, whichever is earlier.

Phase steps are disabled by default. To process them, you must enable processing and specify the strings that indicate start and end. The interface parses the description field to determine the name of the phase step.

The fields that indicate the start and end of levels in the batch hierarchy depend on whether you are using event journals or SQL Server as your data source, as follows:

Data source	Trigger Field	Start/End Field	Level Field
Event journals	[EVENT]	[PVALUE]	[RECIPE]
SQL Server	[EVENTTYPE]	[EVENTDESCRIPT]	[ACTION]

The following table lists the fields and values that trigger the start and end of batch events.

Level	Trigger Field Text	Start/End Text	
PIBatch/Procedure	"State change"	CREATED	REMOVED
PIUnitBatch/Unit Procedure		RUNNING	COMPLETE STOPPED ABORTED
Operation			
Phase			
Phase State	"Report"		

Phase steps are disabled by default. To detect them, you must enable processing and specify the strings that indicate start and end. The interface parses the description field to determine if the start or end string is present.

"Use batch recipe" (/UBR) Enabled

By default, to detect the beginning of a batch, the interface scans the Event column for "State Change" events. To configure the interface to scan for "System Message" events instead, enable the **Use batch recipe** option (on the **Batch Setup** page of Event Frame Interface Manager). This logic is the approach used by the PI Event File interface. Be advised that, when this option is enabled, the batch start time is taken from the event that loads the batch, not the event that begins execution, and there can be a significant delay between loading and execution.

The following sections describe how the interface determines whether a particular level has started or ended when the **Use batch recipe** option is enabled. The precise logic depends on your data source.

Default start and stop events (Emerson)

To set the start time for a unit batch, the interface requires events indicating that the unit procedure has started and the required unit has been acquired. Batches and unit batches are also created when the interface processes operation- and phase-level recipes. In both cases, the unit batch start time is taken from the start event or the arbitration event in which the unit was acquired, whichever is later. The end time is taken from the end event or the event that releases the unit, whichever is earlier.

Phase steps are disabled by default. To process them, you must enable processing and specify the strings that indicate start and end. The interface parses the description field to determine the name of the phase step.

The fields that indicate the start and end of levels in the batch hierarchy depend on whether you are using event journals or SQL Server as your data source, as follows:

Data source	Trigger Field	Start/End Field	Level Field
Event journals	[EVENT]	[PVALUE]	[RECIPE]
SQL Server	[EVENTTYPE]	[EVENTDESCRIPT]	[ACTION]

The following table lists the fields and values that trigger the start and end of batch events.

Level	Trigger Field Text	Start/End Text	
PIBatch/Procedure	"State change"	CREATED	REMOVED
PIUnitBatch/Unit Procedure		RUNNING	COMPLETE STOPPED ABORTED
Operation			
Phase			
Phase State	"Report"		

Phase steps are disabled by default. To detect them, you must enable processing and specify the strings that indicate start and end. The interface parses the description field to determine if the start or end string is present.

UBR start and stop events (Emerson)

By default, to detect the beginning of a batch, the interface scans the Event column for "State Change" events. To configure the interface to scan for "System Message" events instead, enable the **Use batch recipe** option (on the **Batch Setup** page of Event Frame Interface Manager). This logic is the approach used by the PI Event File interface. Be advised that, when this option is enabled, the batch start time is taken from the event that loads the batch, not the event that begins execution, and there can be a significant delay between loading and execution.

The following sections describe how the interface determines whether a particular level has started or ended when the **Use batch recipe** option is enabled. The precise logic depends on your data source.

Emerson PIBatch/procedure

Event journal start/end events

- **Start**

  [EVENT] = "System Message" and [PVALUE] = "Beginning Of BATCH"

- **End**

  Either of these events end a batch:

  - [EVENT] = "System Message" and [PVALUE] = "End Of BATCH"
  - [EVENT] = "State Change" and [PVALUE] field = "REMOVED", "COMPLETE" or "ABORTED"

SQL server start/end events

- **Start**

  Batch recipe event with [DEACTIVATETIME] set, read from the batchview view.

- **End**

  The interface ends a batch when it reads an event containing a timestamp in the [DEACTIVATETIME] field, which is retrieved from the batchview table.

Emerson PIUnitBatch/Unit procedure

Event journal start/end events

Start

For procedure- and unit procedure-level recipes, the interface requires both a start event and an "equipment acquired" event, as follows:

- [EVENT] = "System Message" and [DESCRIPT] field = "Unit Procedure Started".

And

- [EVENT] field = "Recipe Arbitration", [DESCRIPT] field = "Resource Acquired by recipe" and [EU] field = "Unit". The [PVALUE] field contains the actual unit name.

The latest timestamp is used as the start time for the unit batch.

Operation and phase-level recipes create parent batches and unit batches automatically. Operation-level recipes require a start event and an "equipment acquired" event. Phase-level recipes set the unit batch start time from the event that starts the phase; no equipment acquisition event is required.

End

For procedure- and unit procedure-level recipes, the interface requires both an end event and an "equipment released" event, as follows:

- [EVENT] = "System Message" and [DESCRIPT] field = "Unit Procedure Finished".

And

- [EVENT] field = "Recipe Arbitration", [DESCRIPT] field = "Resource Released by recipe" and [EU] field = "Unit". The [PVALUE] field contains the actual unit name.

The earliest timestamp is used as the end time.

Operation-level recipes require an end event and an "equipment released" event. Phase-level recipes set the unit-batch end time from the event that ends the phase; no equipment release event is required.

SQL server start/end events

Start

To start a unit batch, the interface requires both a "start" event and an "equipment acquired" event. It uses the latest timestamp as the start time for the unit batch. Specifically, the interface scans for the following events:

- The batch recipe event containing the [STARTTIME] timestamp associated with the specific "unitprocedure" object retrieved from the batchrecipeview view.
- The arbitration event containing the [ACQUIRETIME] timestamp associated with the specific unit arbitration object retrieved from the batchequipmentview view.

Operation- and phase-level recipes create parent batches and unit batches automatically. Operation-level recipes require a "start" event and an "equipment acquired" event. Phase-level recipes set the unit batch start time from the event that starts the phase; no equipment acquisition event is required.

For operation-level recipes, the following two events are required to start a unit batch:

- The batch recipe event containing the [STARTTIME] timestamp associated with the operation retrieved from the batchrecipeview view.
- The arbitration event containing the [ACQUIRETIME] timestamp associated with the unit arbitration event retrieved from the batchequipmentview view.

The latest timestamp is used as the start time.

For phase-level recipes, the start time is set using the [STARTTIME] field of the batch recipe associated with the phase object.

End

To end a unit batch, the interface requires both an "end" event and an "equipment released" event. It uses the earliest timestamp as the end time for the unit batch. Specifically, the interface scans for the following events:

- The batch recipe event containing the [ENDTIME] timestamp associated with the specific "unitprocedure" object retrieved from the "batchrecipeview" view.
- The arbitration event containing the [RELEASETIME] timestamp associated with the specific unit arbitration object retrieved from the batchequipmentview view.

For operation-level recipes, the following two events are required to end a unit batch:

- The batch recipe event containing the [STARTTIME] timestamp associated with the operation retrieved from the batchrecipeview view.
- The arbitration event containing the [ACQUIRETIME] timestamp associated with the unit arbitration event retrieved from the batchequipmentview view.

The earliest timestamp is used as the end time.

For phase-level recipes, the end time is set using the [ENDTIME] field of the batch recipe associated with the phase object.

Emerson batch operation

Event journal start/end events

Start

For recipes that run at or above the operation level, the event containing [EVENT] = "System Message" and [DESCRIPT] = "Operation Started" and [RECIPE] = "Operation" indicates the start of an operation.

For phase-level recipes, the event containing [EVENT] = "State Change" and [PVALUE] = "RUNNING" indicates the start of an operation.

End

For recipes that run at or above the operation level, the earlier of the following events ends the operation:

- The event containing [EVENT] = "System Message" and [DESCRIPT] field = "Operation Finished"

Or

- The event containing [EVENT] = "State Change" and [PVALUE] field = "REMOVED" and [RECIPE] = "Operation".

For phase-level recipes, the interface assigns the operation start time from the batch recipe event containing [EVENT] = "State Change" and [PVALUE] = "RUNNING".

SQL server start/end events

Start

For operation- and higher-level recipes, the interface sets the start time for an operation using the [STARTTIME] timestamp from the operation start event retrieved from the batchrecipeview view. For phase-level recipes, the interface starts a parent operation using the timestamp from the first phase start event ([EVENTTYPE] = "State Change" and [EVENTDESCRIPT] contains "RUNNING"). The event is retrieved from the brecipestatechangeview or batcheventview view.

End

For operation- and higher-level recipes, the interface sets the end time for an operation using [ENDTIME] timestamp from the operation end event retrieved from the batchrecipeview view. For phase-level recipes, the interface ends a parent operation using the timestamp from the first phase end event ([EVENTTYPE] = "State Change" and [EVENTDESCRIPT] contains "COMPLETE", "ABORTED" or "STOPPED". The event is retrieved from the brecipestatechangeview or batcheventview views.

Emerson batch phase

Event journal start/end events

Start

For recipes that run at or above the phase level, the event containing [EVENT] = "State Change" and [PVALUE] field = "RUNNING".

End

For recipes that run at or above the phase level, the event containing [EVENT] = "State Change" and [PVALUE] field = "COMPLETE" / "STOPPED" / "ABORTED" ends a phase.

SQL server start/end events

Start

For operation- and higher-level recipes, the interface sets the start time for a phase using [STARTTIME] timestamp from the phase start event retrieved from the batchrecipeview view. For phase-level recipes, the interface starts a parent operation using the timestamp from the first phase start event ([EVENTTYPE] = "State Change" and [EVENTDESCRIPT] contains "RUNNING").

End

For operation- and higher-level recipes, the interface sets the end time for a phase using [ENDTIME] timestamp from the phase end event retrieved from the batchrecipeview view. For phase-level recipes, the interface ends a parent operation using the timestamp from the first phase end event ([EVENTTYPE] = "State Change" and [EVENTDESCRIPT] = "COMPLETE", "STOPPED" or "ABORTED").

Emerson batch phase state

Event journal start/end events

The event containing [EVENT] = "State Change" and [PVALUE] field = <State Name> indicates a new phase state.

SQL server start/End events

The interface reads the name of the new phase state from the [ACTION] column in the brecipestatechangeview or batcheventview view.

Emerson batch phase step

By default, phase steps are not detected. You must enable phase steps and configure the strings that indicate state changes. Unlike other levels, no end events are recorded for phase steps: the start of a phase step ends the preceding one.

Event journal start/end events

Start

The event containing [EVENT] = "Report" and [DESCRIPT] or [PVALUE] field = <Start Substring> starts a phase step. The name of the phase step is parsed from the text in the [DESCRIPT] or [PVALUE] field.

End

The event containing [EVENT] = "Report" and [DESCRIPT] or [PVALUE] field = <End Substring> ends a phase step. The name of the phase step is parsed from the text in the [DESCRIPT] or [PVALUE] field.

SQL server start/end events

Start

The event containing [EVENTTYPE] = "Report" and [EVENTDESCR] field = <Start Substring> starts a phase step. The name of the phase step is parsed from the text in the [EVENTDESCR] field.

End

The event containing [EVENTTYPE] = "Report" and [EVENTDESCR] field = <End Substring> ends a phase step. The name of the phase step is parsed from the text in the [EVENTDESCR] field.

Emerson template placeholders

The following sections list placeholders provided by the interface for defining template settings.

DeltaV Batch Historian

- [AREA]
- [BATCHID]
- [COMMENT]
- [DESCRIPT]
- [EU]
- [EVENT] or [PARAMETER]
- [OPERATION]
- [PHASE]
- [PHASEMODULE]
- [PROCEDURE]
- [PROCESSCELL]
- [PVAL] or [VALUE]
- [UNIQUEID]
- [UNIT]
- [UNITPROCEDURE]
- [USERID] or [USER]

For the VALUE setting and for NAME in property and attribute templates, the additional placeholders [TAG] and [TIME] are supported.

DeltaV Alarms & Events Historian Placeholders

- [AREA]
- [ATTRIBUTE]
- [CATEGORY]
- [DESC1]
- [DESC2]
- [EVENT]
- [LEVEL]
- [MODULE]
- [MODULEDESC]
- [NODE]
- [PROCESSCELL]
- [STATE]
- [TIME]
- [UNIT]: For DeltaV Alarms & Events Historian, if [MODULEDESC] is "Unit Module", the [UNIT] placeholder contains the module name. Otherwise it contains the unit name.

DeltaV Syncade Placeholders

- [AREA]
- [BATCHID]
- [DESCRIPT]
- [HIGH]
- [LOW]
- [OPERATION]
- [PARAMETER]
- [PHASE]
- [PROCEDURE]
- [PROCESSCELL]
- [SET]
- [TIME]
- [UNIQUEID]
- [UNIT]
- [UNITPROCEDURE]

- [USER]
- [VALUE]

Event-file-based interfaces (Rockwell, GE and Honeywell)

The following interfaces collect batch events from event journal files created by batch execution systems that are based on Sequencia OpenBatch technology. Because they use the same underlying technology, they contain the same set of start and end events.

This section describes interfaces for the following batch execution systems.

Batch Execution System	Interface Version	OSIsoft interface part number
Rockwell FactoryTalk	3.0.6.301	PI-IN-RW-FTB-NTI
General Electric iBatch	3.0.12.372	PI-IN-GE-IB-NTI
Honeywell TotalPlant	3.0.11.371	PI-IN-HW-TPB-NTI

Event file data sources

The interface collects data from event journal files that are generated directly by the batch execution system. Each file records the execution of particular recipe and contains a log of batch events as well as batch-related data. The interface requires each record (row) in the event file to consists of tab-delimited columns that contain the following information in the following order:

- TIMESTAMP (LOCAL OR GMT)
- BATCHID
- RECIPE
- DESCRIPT
- EVENT
- PVALUE
- EU
- AREA
- PROCCELL
- UNIT
- PHASE
- PHASEDESC
- USERID
- UNIQUEID
- MATERIALNAME
- CONTAINER

Event file batch start and stop events

The following sections describe how the interface determines the start and end times of batch events based on the data read from the data source. The start and end time for the batch event are taken from the data read from the data source. The precise logic that the interface uses depends on the data source and on whether you enable the **Use batch recipe** option.

"Use batch recipe" Disabled (Default)

To set the start time for a unit batch, the interface requires events indicating that the unit procedure has started and the required unit has been acquired. Batches and unit batches are also created when the interface processes operation- and phase-level recipes. In both cases, the unit batch start time is taken from the start event or the arbitration event in which the unit was acquired, whichever is later. The end time is taken from the end event or the event that releases the unit, whichever is earlier.

Phase steps are disabled by default. To process them, you must enable processing and specify the strings that indicate start and end. The interface parses the description field to determine the name of the phase step.

The fields that the interface checks to detect events depend on which data source you are using, as follows:

Data source	Trigger Field	Start/End Field	Level Field
Event journals	[EVENT]	[PVALUE]	[RECIPE]
SQL Server	[EVENTTYPE]	[EVENTDESCRIPT]	[ACTION]

The following table lists the fields and values that trigger the start and end of batch events.

Level	Trigger Field Text	Start/End Text	
PIBatch/Procedure	"State change"	CREATED	REMOVED
PIUnitBatch/Unit Procedure		RUNNING	COMPLETE STOPPED ABORTED
Operation			
Phase			
Phase State	"Report"		

Phase steps are disabled by default. To detect them, you must enable processing and specify the strings that indicate start and end. The interface parses the description field to determine if the start or end string is present.

"Use batch recipe" Enabled

By default, to detect the beginning of a batch, the interface scans the Event column for "State Change" events. To configure the interface to scan for "System Message" events instead, enable the **Use original batch event view** option. This logic is the approach used by the PI Event File interface. Be advised that, when this option is enabled, the batch start time is taken from the event that loads the batch, not the event that begins execution. There can be a significant delay between loading and execution.

The following sections describe how the interface determines whether a particular level has started or ended when the **Use batch recipe** option is enabled. The precise logic depends on your data source.

Event file PIBatch/procedure

Event journal start/end events

Start

[EVENT] = "System Message" and [PVALUE] = "Beginning Of BATCH"

End

Either of these events end a batch:

- [EVENT] = "System Message" and [PVALUE] = "End Of BATCH"

Or

- [EVENT] = "State Change" and [PVALUE] field = "REMOVED", "COMPLETE" or "ABORTED"

Event file PIUnitBatch/unit procedure

Event journal start/end events

Start

For procedure- and unit procedure-level recipes, the interface requires both a start event and an "equipment acquired" event, as follows:

- [EVENT] = "System Message" and [DESCRIPT] field = "Unit Procedure Started"

And

- [EVENT] field = "Recipe Arbitration", [DESCRIPT] field = "Resource Acquired by recipe" and [EU] field = "Unit". The [PVALUE] field contains the actual unit name.

The latest timestamp is used as the start time for the unit batch.

Operation and phase-level recipes create parent batches and unit batches automatically. Operation-level recipes require a start event and an "equipment acquired" event. Phase-level recipes set the unit batch start time from the event that starts the phase; no equipment acquisition event is required.

End

For procedure- and unit procedure-level recipes, the interface requires both an end event and an "equipment released" event, as follows:

- [EVENT] = "System Message" and [DESCRIPT] field = "Unit Procedure Finished".

And

- [EVENT] = "System Message" and [DESCRIPT] field = "Unit Procedure Finished".

And

- [EVENT] field = "Recipe Arbitration", [DESCRIPT] field = "Resource Released by recipe" and [EU] field = "Unit". The [PVALUE] field contains the actual unit name.

The earliest timestamp is used as the end time.

Operation-level recipes require an end event and an "equipment released" event. Phase-level recipes set the unit-batch end time from the event that ends the phase; no equipment release event is required.

SQL server start/end events

Start

To start a unit batch, the interface requires both a "start" event and an "equipment acquired" event. It uses the latest timestamp as the start time for the unit batch. Specifically, the interface scans for the following events:

- The batch recipe event containing the [STARTTIME] timestamp associated with the specific "unitprocedure" object retrieved from the batchrecipeview view.
- The arbitration event containing the [ACQUIRETIME] timestamp associated with the specific unit arbitration object retrieved from the batchequipmentview view.

Operation- and phase-level recipes create parent batches and unit batches automatically. Operation-level recipes require a "start" event and an "equipment acquired" event. Phase-level recipes set the unit batch start time from the event that starts the phase; no equipment acquisition event is required.

For operation-level recipes, the following two events are required to start a unit batch:

- The batch recipe event containing the [STARTTIME] timestamp associated with the operation retrieved from the batchrecipeview view.
- The arbitration event containing the [ACQUIRETIME] timestamp associated with the unit arbitration event retrieved from the batchequipmentview view.

The latest timestamp is used as the start time.

For phase-level recipes, the start time is set using the [STARTTIME] field of the batch recipe associated with the phase object.

End

To end a unit batch, the interface requires both an "end" event and an "equipment released" event. It uses the earliest timestamp as the end time for the unit batch. Specifically, the interface scans for the following events:

- The batch recipe event containing the [ENDTIME] timestamp associated with the specific "unitprocedure" object retrieved from the "batchrecipeview" view.
- The arbitration event containing the [RELEASETIME] timestamp associated with the specific unit arbitration object retrieved from the batchequipmentview view.

For operation-level recipes, the following two events are required to end a unit batch:

- The batch recipe event containing the [STARTTIME] timestamp associated with the operation retrieved from the batchrecipeview view.
- The arbitration event containing the [ACQUIRETIME] timestamp associated with the unit arbitration event retrieved from the batchequipmentview view.

The earliest timestamp is used as the end time.

For operation-level recipes, the following two events are required to end a unit batch:

- The batch recipe event containing the [STARTTIME] timestamp associated with the operation retrieved from the batchrecipeview view.
- The arbitration event containing the [ACQUIRETIME] timestamp associated with the unit arbitration event retrieved from the batchequipmentview view.

The earliest timestamp is used as the end time.

For phase-level recipes, the end time is set using the [ENDTIME] field of the batch recipe associated with the phase object.

Event file operation

Event journal start/end events

Start

For recipes that run at or above the operation level, the event containing [EVENT] = "System Message" and [DESCRIPT] = "Operation Started" and [RECIPE] = "Operation" indicates the start of an operation.

For phase-level recipes, the event containing [EVENT] = "State Change" and [PVALUE] = "RUNNING" indicates the start of an operation.

End

For recipes that run at or above the operation level, the earlier of the following events ends the operation:

- The event containing [EVENT] = "System Message" and [DESCRIPT] field = "Operation Finished"

Or

- The event containing [EVENT] = "State Change" and [PVALUE] field = "REMOVED" and [RECIPE] = "Operation".

For phase-level recipes, the interface assigns the operation start time from the batch recipe event containing [EVENT] = "State Change" and [PVALUE] = "RUNNING".

SQL server start/end events

Start

For operation- and higher-level recipes, the interface sets the start time for an operation using the [STARTTIME] timestamp from the operation start event retrieved from the batchrecipeview view. For phase-level recipes, the interface starts a parent operation using the timestamp from the first phase start event ([EVENTTYPE] = "State Change" and [EVENTDESCRIPT] contains "RUNNING"). The event is retrieved from the brecipestatechangeview or batcheventview view.

End

For operation- and higher-level recipes, the interface sets the end time for an operation using [ENDTIME] timestamp from the operation end event retrieved from the batchrecipeview view. For phase-level recipes, the interface ends a parent operation using the timestamp from the first phase end event ([EVENTTYPE] = "State Change" and [EVENTDESCRIPT] contains "COMPLETE", "ABORTED" or "STOPPED". The event is retrieved from the brecipestatechangeview or batcheventview views.

Event file phase

Event journal start/end events

Start

For recipes that run at or above the phase level, the event containing [EVENT] = "State Change" and [PVALUE] field = "RUNNING".

End

For recipes that run at or above the phase level, the event containing [EVENT] = "State Change" and [PVALUE] field = "COMPLETE" / "STOPPED" / "ABORTED" ends a phase.

SQL server start/end events

Start

For operation- and higher-level recipes, the interface sets the start time for a phase using [STARTTIME] timestamp from the phase start event retrieved from the batchrecipeview view. For phase-level recipes, the interface starts a parent operation using the timestamp from the first phase start event ([EVENTTYPE] = "State Change" and [EVENTDESCRIPT] contains "RUNNING").

End

For operation- and higher-level recipes, the interface sets the end time for a phase using [ENDTIME] timestamp from the phase end event retrieved from the batchrecipeview view. For phase-level recipes, the interface ends a parent operation using the timestamp from the first phase end event ([EVENTTYPE] = "State Change" and [EVENTDESCRIPT] = "COMPLETE", "STOPPED" or "ABORTED").

Event file phase state

Event journal start/end events

The event containing [EVENT] = "State Change" and [PVALUE] field = <State Name> indicates a new phase state.

SQL server start/end events

The interface reads the name of the new phase state from the [ACTION] column in the brecipestatechangeview or batcheventview view.

Event file phase step

By default, phase steps are not detected. You must enable phase steps and configure the strings that indicate state changes. Unlike other levels, no end events are recorded for phase steps: the start of a phase step ends the preceding one.

Event journal start/end events

Start

The event containing [EVENT] = "Report" and [DESCRIPT] or [PVALUE] field = <Start Substring> starts a phase step. The name of the phase step is parsed from the text in the [DESCRIPT] or [PVALUE] field.

End

The event containing [EVENT] = "Report" and [DESCRIPT] or [PVALUE] field = <End Substring> ends a phase step. The name of the phase step is parsed from the text in the [DESCRIPT] or [PVALUE] field.

SQL server start/end events

Start

The event containing [EVENTTYPE] = "Report" and [EVENTDESCR] field = <Start Substring> starts a phase step. The name of the phase step is parsed from the text in the [EVENTDESCR] field.

End

The event containing [EVENTTYPE] = "Report" and [EVENTDESCR] field = <End Substring> ends a phase step. The name of the phase step is parsed from the text in the [EVENTDESCR] field.

Template placeholders for event-file-based interfaces

This interface supports the following placeholders:

- [AREA]
- [BATCHID]
- [DESCRIPT]

- [EU]
- [EVENT]
- [LABEL]
- [LOTNAME]
- [MATERIALID]
- [MATERIALNAME]
- [OPERATION]
- [PHASE]
- [PHASEMODULE]
- [PROCEDURE]
- [PROCESSCELL]
- [PVAL]
- [TIME]
- [UNIQUEID]
- [UNIT]
- [UNITPROCEDURE]
- [USERID] or [USER]

Siemens SIMATIC interface

This interface collects batch data from the Siemens SIMATIC system using the Siemens PI CONNECT SIMATIC BATCH API (PIConnectSimatic.dll), which can be obtained from Seimens. The interface supports the SIMATIC extended data model.

Current version is 3.0.14.387. The OSIsoft part number for this interface is PI-IN-SI-SBAT-NTI.

SIMATIC batch start and stop events

For any level, start is indicated by events in which the Event field contains "System" and the Value field contains "Start". End is indicated by events in which the Event field contains "System" and the Value field contains "End". Phase states start when the Event field contains "State Change" and Level = 4 (phase). Phase states stop when the next phase state begins or when the parent phase ends.

Recipe data is read from events in which the Event field contains "Recipe Header", and the process data is read from events where Event field contains "Act_Val" or "SetPoint". The following table describes the data that accompanies events read from the data source.

Event Field	Meaning	Optional or Required
Level (hidden)	1. Batch 2. Unit Procedure 3. Operation 4. Phase	Required, hidden
Time	GMT Timestamp of the event in "yyyy-MM-dd hh:mm:ss.000" format	Required
UniqueID	Unique ID of the PI Batch (= Simatic Batch "Batch ID")	Required
BatchID	Batch ID of the PI Batch (=Simatic Batch "Batch Name")	Required
Procedure	Name of the recipe	Required
UnitProcedure	Name of the unit procedure with activation counter in [ ] as suffix	Level 1: Empty Level 2, 3, 4: Required
Operation	Name of the Operation with activation counter in [ ] as suffix	Level 1, 2: Empty Level 3, 4: Required
Phase	Name of the Phase, with activation counter in [ ] as suffix	Level 1, 2, 3: Empty Level 4: Required
Category	Available Category field values: • ProcessInput • ProcessOutput • ProcessParameter	Required for parameter event (in Event field) Act_Val SetPoint

Event Field	Meaning	Optional or Required
Descript	Depends on contents of Event field as follows: • StateChange: new state • Act_Val or SetPoint: process parameter name • Recipe Header: recipe parameter name	Required for parameter event (in Event field): • Act_Val • SetPoint • Recipe Header • StateChange
Event	Indicates nature of event as follows: • Batch start / end data: System • Batch / recipe header data: Recipe Header • State change events: StateChange • Process parameter data: Act_Val or SetPoint	Required
Value	Depends on contents of Event field as follows: • StateChange: integer value of the new state • Act_Val or SetPoint: process parameter value • Recipe Header: recipe parameter value • System: Start or End	Required
Eu	Engineering units. Depends on contents of Event field as follows: • Act_Val or SetPoint: process parameter engineering units • Recipe Header: recipe parameter engineering units	Optional
Area	First level of equipment structure in PI Batch	Optional
ProcessCell	Second level of equipment structure in PI Batch	Optional
Unit	Name of the unit module for PI Batch equipment structure (Simatic Batch unit name)	Level 1: empty Level 2, 3, 4: required
Operation Module	Name of the operation module for PI Batch equipment structure (Simatic Batch operation name)	Level 1, 2: empty Level 3, 4: required
PhaseModule	Name of the phase module for PI Batch equipment structure (Simatic Batch phase name)	Level 1, 2, 3: empty Level 4: required

Event Field	Meaning	Optional or Required
MaterialName	Name of the material	Required for Parameter Event: Act_Val SetPoint All others empty
MaterialID	ID of the material	Required for Parameter Event: Act_Val SetPoint All others empty

SIMATIC template placeholders

This interface supports the following placeholders:

- [AREA]
- [BATCHID]
- [CATEGORY]
- [DESCRIPT]
- [EU]
- [EVENT] or [PARAMETER]
- [EXTVALUE]
- [MATERIALID]
- [MATERIALNAME]
- [OPERATION]
- [OPERATIONACTIVATION]
- [OPERATIONCONTID]
- [PARAMETERID]
- [PHASE]
- [PHASEACTIVATION]
- [PHASECONTID]
- [PHASEMODULE]
- [PHASESTATE]
- [PHASETERMID]
- [PROCEDURE]
- [PROCESSCELL]
- [TAG]

- [TIME]
- [UNIQUEID]
- [UNIT],[OPERATIONMODULE]
- [UNITACTIVATION]
- [UNITCONTID]
- [UNITID]
- [UNITPROCEDURE]
- [VALUE]

Siemens XFP interface

This interface gathers information from the Siemens XFP manufacturing execution system. Current version is 3.0.16. The OSIsoft part number for this interface is PI-IN-SI-SXFP-NTI.

Siemens XFP data source

The interface collects data from the SQL Server databases used by the Siemens XFP MES. A single interface instance can collect data from multiple data sources. The following tables describe the XFP database tables and columns from which the interface derives the data that it stores in the PI System. You use these fields when you compose templates to configure how you want the interface to store batch data. In the tables, note that *level* is a value from 1 to 8 and *equipment number* is a value from 1 to 4.

STATE CHANGE Event

Indicates Start or End of a MES Task

Field	Contents
EVENT	"STATE CHANGE"
BATCHID	Recipe template-dependent
PRODUCT	Recipe template-dependent
PROCEDURE	Recipe template-dependent
UNITPROCEDURE	Recipe template-dependent
OPERATION	Recipe template-dependent
PHASE	Recipe template-dependent
PHASESTATE	Recipe template-dependent
PHASESTEP	Recipe template-dependent
PHASESTEP2	Recipe template-dependent
PHASESTEP3	Recipe template-dependent
EQPCODE_{level}_{equipment number}	MDSEQP_EQPEXE_TRACES. EQPCODE
EQPNAME_{level}_{equipment number}	MDSEQP_EQPEXE_TRACES. EQPNAME
LOCATION_{level}_{equipment number}	MDSEQP_EQPEXE_TRACES. LOCATION
AREA_{level}_{equipment number}	MDS_SITE. SITE_NAME_ENG
LOTID	MDSXFP_WOHEADER. PHARMALOTNUMBER
TITLE	MDSE2S_PFC_TRAIL_MAN.TITLE
SITE	MDS_SITE. SITE_NAME_ENG
XFPBATCHID	MDSE2S_PFC_TRAIL_MAN.BATCHID
XFPTASKID	MDSE2S_PFC_TRAIL_MAN.TASKID
WORKCENTER	MDSE2S_PFC_TRAIL_MAN.WORKCENTER
WORKSTATION	MDSE2S_PFC_TRAIL_MAN.WORKSTATION
UNIQUEID	MDSE2S_PFC_TRAIL_MAN.WONUMBER
TIMESTAMP	MDSE2S_PFC_TRAIL_MAN.DATETIME

WONUMBER	MDSE2S_PFC_TRAIL_MAN.WONUMBER
PICODE	MDSE2S_PFC_TRAIL_MAN.PICODE
PIVERSION	MDSE2S_PFC_TRAIL_MAN.PIVERSION

REPORT PARAMETER Event

Field	Contents
EVENT	"REPORT PARAMETER"
BATCHID	Recipe template-dependent
PRODUCT	Recipe template-dependent
PROCEDURE	Recipe template-dependent
UNITPROCEDURE	Recipe template-dependent
OPERATION	Recipe template-dependent
PHASE	Recipe template-dependent
PHASESTATE	Recipe template-dependent
PHASESTEP	Recipe template-dependent
PHASESTEP2	Recipe template-dependent
PHASESTEP3	Recipe template-dependent
EQPCODE_{level}_{equipment number}	MDSEQP_EQPEXE_TRACES. EQPCODE
EQPNAME_{level}_{equipment number}	MDSEQP_EQPEXE_TRACES. EQPNAME
LOCATION_{level}_{equipment number}	MDSEQP_EQPEXE_TRACES. LOCATION
AREA_{level}_{equipment number}	MDS_SITE. SITE_NAME_ENG
LOTID	MDSXFP_WOHEADER.PHARMALOTNUMBER
TITLE	MDSE2S_PFC_TRAIL_MAN.TITLE
UNIQUEID	MDSE2S_PFC_TRAIL_MAN.WONUMBER
WONUMBER	MDSE2S_PFC_TRAIL_MAN.WONUMBER
PICODE	MDSE2S_PFC_TRAIL_MAN.PICODE
PIVERSION	MDSE2S_PFC_TRAIL_MAN.PIVERSION
TIMESTAMP	MDSE2S_PIDATA_MAN. INPUTDATE
TAGNUMBER	MDSE2S_PIDATA_MAN. TAGNUMBER
PARAMETERCODE	MDSE2S_PIDATA_MAN. PARAMETERCODE
PARAMETERVERSION	MDSE2S_PIDATA_MAN. PARAMETERVERSION
PARAMETERDESCRIPTION	MDSE2S_PIDATA_MAN. PARAMETERDESCRIPTION
VALUE	MDSE2S_PIDATA_MAN. VALUE
INPUTINDEX	MDSE2S_PIDATA_MAN. INPUTINDEX
TASKDESCRIPTION	MDSE2S_PIDATA_MAN. TASKDESCRIPTION
LOGIN	MDSE2S_PIDATA_MAN. LOGIN
EU	MDSE2S_PIDATA_MAN.UNIT

Report PARAMETER METADATA Event

Field	Contents

EVENT	"PARAMETER METADATA"
BATCHID	Recipe template-dependent
PRODUCT	Recipe template-dependent
PROCEDURE	Recipe template-dependent
UNITPROCEDURE	Recipe template-dependent
OPERATION	Recipe template-dependent
PHASE	Recipe template-dependent
PHASESTATE	Recipe template-dependent
PHASESTEP	Recipe template-dependent
PHASESTEP2	Recipe template-dependent
PHASESTEP3	Recipe template-dependent
EQPCODE_{level}_{equipment number}	MDSEQP_EQPEXE_TRACES. EQPCODE
EQPNAME_{level}_{equipment number}	MDSEQP_EQPEXE_TRACES. EQPNAME
LOCATION_{level}_{equipment number}	MDSEQP_EQPEXE_TRACES. LOCATION
AREA_{level}_{equipment number}	MDS_SITE. SITE_NAME_ENG
LOTID	MDSXFP_WOHEADER.PHARMALOTNUMBER
TITLE	MDSE2S_PFC_TRAIL_MAN.TITLE
UNIQUEID	MDSE2S_PFC_TRAIL_MAN.WONUMBER
WONUMBER	MDSE2S_PFC_TRAIL_MAN.WONUMBER
PICODE	MDSE2S_PFC_TRAIL_MAN.PICODE
PIVERSION	MDSE2S_PFC_TRAIL_MAN.PIVERSION
TIMESTAMP	MDSE2S_PIDATA_MAN. INPUTDATE
TAGNUMBER	MDSE2S_PIDATA_MAN. TAGNUMBER
PARAMETERCODE	MDSE2S_PIDATA_MAN. PARAMETERCODE
PARAMETERVERSION	MDSE2S_PIDATA_MAN. PARAMETERVERSION
PARAMETERDESCRIPTION	MDSE2S_PIDATA_MAN. PARAMETERDESCRIPTION
PARAMETERVALUE	MDSE2S_PIDATA_MAN. PARAMETERVALUE
EU	MDSE2S_PIDATA_MAN.UNIT
METADATANAME	MDSE2S_METADATAVALUE_MAN.NAME
METADATASETID	MDSE2S_PIDATA_MAN. METADATASET_ID
METADATADESC	MDSE2S_METADATAVALUE_MAN. DESCRIPTION

WORKORDER HEADER Event

Field	Contents
EVENT	"WORKORDER HEADER"
BATCHID	Recipe template-dependent
PRODUCT	Recipe template-dependent
PROCEDURE	Recipe template-dependent
UNITPROCEDURE	Recipe template-dependent
OPERATION	Recipe template-dependent

PHASE	Recipe template-dependent
PHASESTATE	Recipe template-dependent
PHASESTEP	Recipe template-dependent
PHASESTEP2	Recipe template-dependent
PHASESTEP3	Recipe template-dependent
EQPCODE_{level}_{equipment number}	MDSEQP_EQPEXE_TRACES. EQPCODE
EQPNAME_{level}_{equipment number}	MDSEQP_EQPEXE_TRACES. EQPNAME
LOCATION_{level}_{equipment number}	MDSEQP_EQPEXE_TRACES. LOCATION
AREA_{level}_{equipment number}	MDS_SITE. SITE_NAME_ENG
LOTID	MDSXFP_WOHEADER.PHARMALOTNUMBER
TITLE	MDSE2S_PFC_TRAIL_MAN.TITLE
UNIQUEID	MDSE2S_PFC_TRAIL_MAN.WONUMBER
WONUMBER	MDSE2S_PFC_TRAIL_MAN.WONUMBER
PICODE	MDSE2S_PFC_TRAIL_MAN.PICODE
PIVERSION	MDSE2S_PFC_TRAIL_MAN.PIVERSION
TIMESTAMP	MDSE2S_PFC_TRAIL_MAN.DATETIME
DESCRIPT	MDSXFP_WOHEADER source column (see below)
VALUE	Value from source column
EU	MDSXFP_WOHEADER.WEIGHTEQUUNIT

Source columns for DESCRIPT field

- "WOQUANITITY "
- "WEIGHTEQUQUANTITY "
- "WEIGHTEQUUNIT "
- "PRODUCTCODE "
- "PHARMALOTNUMBER "
- "WONUMBER "
- "WOINDEX "
- "STATE "
- "PRODUCTDESCRIPTION "
- "WOUNIT "
- "PRIORITY "
- "REFERENCECODE "
- "STABILITY "
- "CUSTOMERCODEREF "
- "ALLOCATIONPART "
- "ORIGINE "
- "DTCREATIONDATEBYSYSTEM "

- "DTPLANNEDLAUNCHINGDATE "
- "DTLASTCHANGESTATEDATE "
- "TRANSACTIONCODE "
- "FUNCTIONCODE "
- "PICODE "
- "PIVERSION "
- "COMBINATIONEDITNUMBER "
- "XFIELD_00 "
- "XFIELD_01 "
- "XFIELD_02 "
- "XFIELD_03 "
- "XFIELD_04 "
- "XFIELD_05 "
- "XFIELD_06 "
- "XFIELD_07 "
- "XFIELD_08 "
- "MANUALWEIGHTSTARTDATE "
- "MANUALWEIGHTENDDATE "

WORKORDER CONSUMPTION Event

Field	Contents
EVENT	"WORKORDER CONSUMPTION"
BATCHID	Recipe template-dependent
PRODUCT	Recipe template-dependent
PROCEDURE	Recipe template-dependent
UNITPROCEDURE	Recipe template-dependent
OPERATION	Recipe template-dependent
PHASE	Recipe template-dependent
PHASESTATE	Recipe template-dependent
PHASESTEP	Recipe template-dependent
PHASESTEP2	Recipe template-dependent
PHASESTEP3	Recipe template-dependent
EQPCODE_{level}_{equipment number} Level 1 to 8 Equipment number 1 to 4	MDSEQP_EQPEXE_TRACES. EQPCODE
EQPNAME_{level}_{equipment number}	MDSEQP_EQPEXE_TRACES. EQPNAME
LOCATION_{level}_{equipment number}	MDSEQP_EQPEXE_TRACES. LOCATION
AREA_{level}_{equipment number}	MDS_SITE. SITE_NAME_ENG
LOTID	MDSXFP_WOHEADER.PHARMALOTNUMBER

TITLE	MDSE2S_PFC_TRAIL_MAN.TITLE
UNIQUEID	MDSE2S_PFC_TRAIL_MAN.WONUMBER
WONUMBER	MDSE2S_PFC_TRAIL_MAN.WONUMBER
PICODE	MDSE2S_PFC_TRAIL_MAN.PICODE
PIVERSION	MDSE2S_PFC_TRAIL_MAN.PIVERSION
TIMESTAMP	MDSXFP_WOCONSUMPTION. DTWEIGHENDDATE
DESCRIPT	Source column from MDSXFP_WOCONSUMPTION table (see below)
VALUE	Value from source column
EU	MDSXFP_WOHEADER. WEIGHTEUUNIT
ITEMCODE	MDSXFP_WOCONSUMPTION. ITEMCODE
ITEMDESC	MDSXFP_WOCONSUMPTION. ITEMDESC
WEIGHTINGINDEX	MDSXFP_WOCONSUMPTION. WEIGHTINGINDEX
LABELNUM	MDSXFP_WOCONSUMPTION. LABELNUM

Values for DESCRIPT field

- "WONUMBER"
- "WOINDEX"
- "PHASE"
- "PHASESEQNUM"
- "DOSE"
- "ACTION"
- "ITEMCODE"
- "WEIGHINGINDEX"
- "LABELNUM"
- "OPERATOR"
- "WORKSTATION"
- "WORKCENTER"
- "WDUCLEANTYPE"
- "LOTNUMBER"
- "LOCATION"
- "CONTAINER"
- "LOTPOTENCY"
- "LOTPOTENCY2"
- "LOTDENSITY"
- "LOTSTATUS"

- "ITEMUNIT"
- "FLAGPOTENCY"
- "FLAGPOTENCY2"
- "ITEMPOTENCY"
- "ITEMPOTENCY2"
- "ITEMDENSITY"
- "TOLERANCE"
- "FLAGDENSITYTAG"
- "CONCORDANCENUM"
- "STARTGROSS"
- "ENDGROSS"
- "CALCULATEDNET"
- "TARE"
- "NET"
- "GROSS"
- "WEIGHUNIT"
- "WEIGHGAP"
- "FORCEDWEIGHING"
- "FLAGSAMECONTAINER"
- "PALLETCODE"
- "NUMCONTAINERS"
- "WEIGHINGMODE"
- "WEIGHINGTYPE"
- "WEIGHINGSTATUS"
- "BALANCENUMBER"
- "BALANCETYPE"
- "BALANCEDESCRIPTION"
- "FLAGFORCEDPRECISION"
- "FLAGFORCEDRELEASE"
- "ENDOFCONTAINER"
- "FLAGMESSAGE"
- "FLAGFORCEDLOT"
- "FLAGCORRECTLOT"
- "FLAGKEYBOARDINPUT"
- "FLAGPRINTEDLABEL"

- "BARCODELABEL"
- "CHECK"
- "STOCKUNIT"
- "NOTICKET"
- "RFCHECK"
- "COMBINATION"
- "INITIALEQTCODE"
- "TARGETEQTCODE"
- "COUNTINGUNIT"
- "DTWDUCLEANDATE"
- "DTWEIGHSTARTDATE"
- "DTWEIGHENDDATE"
- "ACTIONID"
- "XFIELD_00"
- "XFIELD_01"
- "XFIELD_02"
- "XFIELD_03"
- "XFIELD_04"
- "XFIELD_05"
- "XFIELD_06"
- "XFIELD_07"
- "XFIELD_08"
- "XFIELD_09"
- "CONS_XFIELD_00"
- "CONS_XFIELD_01"
- "CONS_XFIELD_02"
- "CONS_XFIELD_03"
- "CONS_XFIELD_04"
- "CONS_XFIELD_05"
- "CONS_XFIELD_06"
- "CONS_XFIELD_07"
- "CONS_XFIELD_08"
- "CONS_XFIELD_09"
- "IDENTIFIER"
- "LOCATIONDESCR"

- "ITEMDESCR"
- "ITEMLONGDESCR"
- "CONTSTATUS"
- "LOTEXPIRYDATE"
- "REMAININGQTY"
- "SHORTNAME"
- "FULLNAME"
- "MATLINESTATUS"
- "ITEMPOTENCYUNIT1"
- "ITEMPOTENCYUNIT2"

COMMENT Event

Data sent when there is a comment for the deviation in the PICODE.

Field	Contents
EVENT	"COMMENT"
BATCHID	Recipe template-dependent
PRODUCT	Recipe template-dependent
PROCEDURE	Recipe template-dependent
UNITPROCEDURE	Recipe template-dependent
OPERATION	Recipe template-dependent
PHASE	Recipe template-dependent
PHASESTATE	Recipe template-dependent
PHASESTEP	Recipe template-dependent
PHASESTEP2	Recipe template-dependent
PHASESTEP3	Recipe template-dependent
EQPCODE_{level}_{equipment number}	MDSEQP_EQPEXE_TRACES. EQPCODE
EQPNAME_{level}_{equipment number}	MDSEQP_EQPEXE_TRACES. EQPNAME
LOCATION_{level}_{equipment number}	MDSEQP_EQPEXE_TRACES. LOCATION
AREA_{level}_{equipment number}	MDS_SITE. SITE_NAME_ENG
LOTID	MDSXFP_WOHEADER.PHARMALOTNUMBER
TITLE	MDSE2S_PFC_TRAIL_MAN.TITLE
UNIQUEID	MDSE2S_PFC_TRAIL_MAN.WONUMBER
WONUMBER	MDSE2S_PFC_TRAIL_MAN.WONUMBER
PICODE	MDSE2S_PFC_TRAIL_MAN.PICODE
PIVERSION	MDSE2S_PFC_TRAIL_MAN.PIVERSION
TIMESTAMP	MDSE2S_PICOMMENTS_MAN.DTCOMMENTDATE
COMMENTNUMBER	MDSE2S_PICOMMENTS_MAN. COMMENTNUMBER
COMMENT	MDSE2S_PICOMMENTS_MAN. COMMENT

DESCRIPT	The source column from the MDSE2S_PICOMMENTS_MAN table (see below)
VALUE	Value from source column

Values for DESCRIPT field

- "SITE_NAME_ENG"
- "PICODE"
- "PIVERSION"
- "STATUS"
- "LOGIN"
- "SHORTNAME"
- "FULLNAME"
- "TYPE"
- "COMMENTTYPE"
- "WORKSTATION"
- "WORKCENTER"
- "ORIGIN"
- "TASKTITLE"
- "PRODUCTCODE"
- "PHARMALOTNUMBER"

Siemens XFP batch start and stop events

The interface uses proprietary logic rather than a fixed set of events to determine the start and stop of the various levels of a procedure (unlike other batch interfaces, which rely on a fixed set of events).

Siemens XFP template placeholders

This interface supports the following placeholders:

- AREA_1_1, AREA_1_2, AREA_1_3, AREA_1_4
- AREA_2_1, AREA_2_2, AREA_2_3, AREA_2_4
- AREA_3_1, AREA_3_2, AREA_3_3, AREA_3_4
- AREA_4_1, AREA_4_2, AREA_4_3, AREA_4_4
- AREA_5_1, AREA_5_2, AREA_5_3, AREA_5_4
- CATEGORY
- COMMENT, COMMENTNUMBER

- EQPCODE_1_1, EQPCODE_1_2, EQPCODE_1_3, EQPCODE_1_4
- EQPCODE_2_1, EQPCODE_2_2, EQPCODE_2_3, EQPCODE_2_4
- EQPCODE_3_1, EQPCODE_3_2, EQPCODE_3_3, EQPCODE_3_4
- EQPCODE_4_1, EQPCODE_4_2, EQPCODE_4_3, EQPCODE_4_4
- EQPCODE_5_1, EQPCODE_5_2, EQPCODE_5_3, EQPCODE_5_4
- EQPNAME_1_1, EQPNAME_1_2, EQPNAME_1_3, EQPNAME_1_4
- EQPNAME_2_1, EQPNAME_2_2, EQPNAME_2_3, EQPNAME_2_4
- EQPNAME_3_1, EQPNAME_3_2, EQPNAME_3_3, EQPNAME_3_4
- EQPNAME_4_1, EQPNAME_4_2, EQPNAME_4_3, EQPNAME_4_4
- EQPNAME_5_1, EQPNAME_5_2, EQPNAME_5_3, EQPNAME_5_4
- EU
- FULLNAME
- INPUTINDEX
- ITEMCODE
- ITEMDESCR
- LABELNUM
- LOCATION_1_1, LOCATION_1_2, LOCATION_1_3, LOCATION_1_4
- LOCATION_2_1, LOCATION_2_2, LOCATION_2_3, LOCATION_2_4
- LOCATION_3_1, LOCATION_3_2, LOCATION_3_3, LOCATION_3_4
- LOCATION_4_1, LOCATION_4_2, LOCATION_4_3, LOCATION_4_4
- LOCATION_5_1, LOCATION_5_2, LOCATION_5_3, LOCATION_5_4
- LOGIN
- LOTID
- METADATADESC, METADATANAME, METADATASETID
- PARAMETERCODE, PARAMETERDESCRIPTION, PARAMETERVALUE, PARAMETERVERSION
- PHASESTEP2, PHASESTEP3
- PICODE
- PIVERSION
- SHORTNAME
- SITE
- TAGNUMBER
- TASKDESCRIPTION
- TIME
- TITLE
- USERID

- WEIGHINGINDEX
- WOINDEX
- WONUMBER
- WONUMBER
- WORKCENTER
- WORKSTATION
- XFPBATCHID
- XFPTASKID

Werum PAS-X interface

This interface is designed to be used for Werum PAS-X systems using Oracle 11g and later as the batch historian. Current version is 3.0.5.240. The OSIsoft part number for this interface is PI-IN-WRM-PASX-NTI.

Werum PAS-X data sources

There are two methods the interface can use to retrieve data from Oracle Server: querying a single view that contains data in the form of XML events, or querying a combination of tables from the PASX and WLTUSER name spaces. This second method requires version 3.1.4 of the Werum PAS-X system. The user that runs the interface requires read access to only the IFMESTOPIBATCH_C table, which contains batch-associated data.

Data retrieval from the Oracle server is scan-based. The interface uses the Oracle OLEDB driver (part of Oracle Provider for OLE DB package) to communicate with the Oracle server. After installing the Oracle Provider for OLE DB, verify that the `ORACLE_BASE\ORACLE_HOME\bin` directory contains the OraOLEDB.dll file.

To reduce the CPU load spike imposed by interface queries issued against the PAS-X database server, you can configure a delay between queries (**`/ibfd`**).

Werum PAS-X batch start and stop events

The interface creates events based on the contents of the data source as follows:

Batch event in PI System	Data source
PIBatch/Procedure	Manufacturing Order
PIUnitBatch/Unit Procedure	Basic Operation
Operation	Basic Function
Phase	
Phase State	

Werum PAS-X template placeholders

Generic placeholders

Placeholder	Description
[TIMESTAMP]	Timestamp of the event. For PI EVENTS, this placeholder refers to the start or end of the destination time interval. For all other events, this placeholder contains the timestamp of the event..
[TAG]	Refers to a PI point.

Data sources for parameter data placeholders

The placeholders in the following table are set when the [PARAMETER] placeholder contains one of the values listed in the right-hand column.

Placeholder	Available in Version	Description	Set when [PARAMETER]=
[ACTUALQTY]	3.1.4+	Actual Quantity of material produced or consumed	MATERIAL, TAKEOUT, YIELDDETERMINATION, STOCKCREATION
[CHARGEBATCHID]	3.1.4+	Charge BatchID. Applicable to output materials.	MATERIAL
[CONTAINERID]	3.1.4+	Material's Container ID.	TAKEOUT
[CRITICALITY]	3.1.4+	Provides Parameter Criticality string value.	
[CRITICALITYID]	3.1.4+	Provides Parameter Criticality numeric value. Available only for PASX 3.1.4	
[DATATYPE]	All	The data type of the events, possible values: NUMERIC, STRING, DATE (supported in versions prior to 3.1.4)	
[DEFINITIONLWB]	All	Parameter's definition lower bound. Applicable to numeric events only.	
[DEFINITIONUPB]	All	Parameter's definition upper bound. Applicable to numeric events only.	
[DESC]	All	Parameter description.	
[INTERVENTIONLWB]	All	Parameter's intervention lower bound. Applicable to numeric events only.	
[INTERVENTIONUPB]	All	Parameter's intervention upper bound. Applicable to numeric events only.	
[LOTID]	3.1.4+	Material Lot ID	MATERIAL, TAKEOUT, YIELDDETERMINATION, STOCKCREATION
[MATERIALID]	All	Material ID	MATERIAL, TAKEOUT, YIELDDETERMINATION, STOCKCREATION
[MATERIALNAME]	All	Material Description	MATERIAL, TAKEOUT, YIELDDETERMINATION, STOCKCREATION

Placeholder	Available in Version	Description	Set when [PARAMETER]=
[PARAMETER]	All	Parameter name. Mapped to data source CustomId column.	
[SETVALUE]	All	The Set Value of the event. Applicable to numeric events only.	
[TARGETQTY]	31.4+	Target Quantity of material to be produced or consumed	MATERIAL, TAKEOUT, YIELDDETERMINATION , STOCKCREATION
[TOLERANCELWB]	All	Parameter's tolerance lower bound. Applicable to numeric events only.	
[TOLERANCEUPB]	All	Parameter's tolerance upper bound. Applicable to numeric events only.	
[UOM]	All	The Units of Measure	
[USERID]	Pre-3.1.4 only	The UserYou ID	
[VALUE]	All	The Value of the event.	
[WARNINGLWB]	All	Parameter's warning lower bound. Applicable to numeric events only.	
[WARNINGUPB]	All	Parameter's warning upper bound. Applicable to numeric events only.	

Data sources for batch-specific placeholders

Placeholder	Availability	Description
[AREADESC]	3.1.4+	Area Description. Available at and below Basic Operation level
[AREAID]	3.1.4+	Area ID. Available at and below Basic Operation level
[BATCHID]	3.1.4+	BatchID of Manufacturing order. In version 3.1.4+, mapped to the WLTUSER.VFERTIGUNGSAUFTRAG.chargennr column
[BATCHTIMESTAMP]	All	Manufacturing Order Start Time.
[BFDESC]	All	Basic Function Description (Level = 3)
[BFDESC{n}]	All	Basic Function Description (Level = 3+{n}, where {n} >=1 and {n} <=5))
[BFID]	All	Basic Function ID (Level = 3)
[BFID{n}]	All	Basic Function ID (Level = 3+{n}, where {n} >= 1 and {n} <=5))
[BFINDEX]	3.1.4+	Basic Function Repeat Index (Level = 3)

Placeholder	Availability	Description
[BFINDEX{n}]	3.1.4+	Basic Function Repeat Index (Level = 3+{n}, where {n} >=1 and {n} <=5)
[BFTYPE]	All	Basic Function Type (Level = 3)
[BFTYPE{n}]	All	Basic Function Type (Level = 3+{n}, where {n} >=1 and {n} <=5)
[BODESC]	All	Basic Operation Description
[BOID]	All	Basic Operation ID
[BOLONGDESC]	< 3.1.4 only	Basic Operation Long Description.
[DCSBATCHID]	3.1.4+	The batchI D assigned to the BES batch by DCS Basic Function with type: [BFTYPE] = DCSRECIPE
[MATERIALQTY]	All	Output material quantity.
[MATERIALUOM]	All	Output material unit of measurements.
[MBRAREA]	All	MBR Area.
[MBRDESC]	All	MBR Description
[MBRID]	All	MBR ID
[MBRLONGDESC]	All	MBR Long Description
[MBRSTATUS]	All	MBR Status
[MBRVERSION]	All	MBR Version
[MBRVERSIONID]	All	MBR Version ID
[MODESC]	3.1.4+	Manufacturing Order Description. Mapped to WLTUSER.VFERTIGUNGSAUFTRAG.HVVARIANTE column
[MOID]	3.1.4+	Manufacturing Order ID. Mapped to WLTUSER.VFERTIGUNGSAUFTRAG.FANR column
[PRODUCTID]	All	Product ID. Synonym for [MATERIALID]
[PRODUCTNAME]	All	Product Name. Synonym for [MATERIALNAME]
[SFOID]	3.1.4+	Shopfloor Order ID. Mapped to WLTUSER.vproduktionsauftrag.pakey column
[STATUS]	All	MO, BO and BF execution status MO and BO: string status representation, BF: numeric status representation.

Placeholder	Availability	Description
[UNIQUEID]	3.1.4+	Refers to root unique key of the time interval object. For PASX version 3.1.4, this is mapped to the PASX.manufacturingorder.entity key data source property.
[UNITID]	All	Workroom ID
[UNITNAME]	All	Workroom Name.
[UNITTYPE]	3.1.4+	Workroom Type.

PIEVENT placeholders

These placeholders are supported for expression triggered by PI batch events (Parameter, value="PIEVENT"]).

Place Holder	Batch Level	How Stored	
		Batch Database	Event Frames
[BATCHID]	1 or 2	String value that is stored as PIBatch BatchID and PIUnitBatch BatchID property	Top-level event frame: Name property. Second-level event frame: BatchID attribute
[PROCEDURE]	1	PIBatch Recipe property	Top level event frame "Recipe" Attribute.
[UNITPROCEDURE]	2	PIUnitBatch Procedure property	Name property
[OPERATION]	3	PISubBatch Name property	
[PHASE]	4		
[PHASESTATE]	5		
[PHASESTEP]	6		
[UNIT]	n/a	PIModule Name property	AF element Name property.

Wonderware InBatch interface

This interface enables you to capture history from Invensys' Wonderware InBatch batch management system. Current version is 3.0.10.316. The OSIsoft part number for this interface is PI-IN-WW-IB-NTI.

Wonderware data sources

The Wonderware InBatch Batch Execution System (BES) version 8.1 SP1 and above stores batch data in Microsoft SQL Server. The interface reads data from the following tables:

View/Table	Data read by interface
BatchIDLog	UniqueID, BatchID, start time, end time
BatchDetail	Batch recipe data, product, equipment used for recipe execution, Start/End batch events.
Union of tables: • ProcessVar • ProcessVarChange • OperatorComment • PhaseInstruction • MaterialInput • MaterialInputChange • MaterialOutput	BatchDetail data and data for all batches, including Variable Name, Variable Actual Value, Variable Target or Old Value, and Event Timestamp.

To communicate with SQL Server databases, the interface requires Microsoft ADO driver for Microsoft SQL, part of the SQL Native client package, to be installed on the interface node. To download the SQL Native client package, go to the MSDN web site.

Wonderware batch start and stop events

Events are read from the BatchIDLog and BatchDetail tables. The InBatch system does not provide phase state changes or phase steps.

PIBatch/Procedure

The start time for a batch is taken from the event in which the [Action_CD] field contains "201". The end time for a batch is taken from the event in which the [Action_CD] field contains "205", "206" or "209".

PIUnitBatch/Unit Procedure

The start time for a batch is taken from the event in which the [Action_CD] field contains "500". The end time for a batch is taken from the event in which the [Action_CD] field contains "501".

Operation

The start time for a batch is taken from the event in which the [Action_CD] field contains "502". The end time for a batch is taken from the event in which the [Action_CD] field contains "503".

Phase

The start time for a batch is taken from the event in which the [Action_CD] field contains a value from "227" to "246" and "415", excluding "236" and "240".

The end time for a batch is taken from the event in which the [Action_CD] field contains one of the following values:

- 233
- 234
- 239
- 415

Wonderware template placeholders

- [BATCHID]
- [DESCRIPT]
- [EU]
- [OLDVALUE]
- [OPERATION]
- [PARAMETER]
- [PHASE]
- [PROCEDURE]
- [TARGETVALUE]
- [TIME]
- [UNIQUEID]
- [UNIT]
- [UNITPROCEDURE]
- [USERID]
- [VALUE]

Wonderware placeholder data sources

The following tables map placeholders to the data source table from which they are derived.

Combined - BatchDetail + BatchIDLog

Template Placeholder	SQL Column Name	Description
[ActionCD]	Action_CD	Defined in BatchDetail table.
[TimeStamp]	DateTime	Defined in BatchDetail table.

Template Placeholder	SQL Column Name	Description
[UniqueID]	Batch_Log_ID	Used to bind both BatchDetail and BatchIDLog tables. Serves as UniqueID of each batch
[BatchID]	Batch_ID	Defined in BatchIDLog table.
[Product]	Product_Name or Product_ID	Defined in BatchIDLog table. This placeholder maps to Product_Name if non-empty, otherwise it maps to Product_ID
[Procedure]	Recipe_ID	Defined in BatchIDLog table.
[Unit]	UnitOrConnection	Defined in BatchDetail table.
[UnitProcedure]	UnitProcedure_ID	Defined in BatchDetail table.
[Operation]	Operation_ID	Defined in BatchDetail table.
[Phase]	Phase_ID	Defined in BatchDetail table.
[PhaseInstance]	Index derived from Phase_Instance_ID	Defined in BatchDetail table. Provides phase instance index.
[UserID]	DoneBy_User_ID	Defined in BatchDetail table.
[Campaign]	Campaign_ID	Defined in BatchIDLog table.
[Lot]	Lot_ID	Defined in BatchIDLog table.
[Train]	Train_ID	Defined in BatchIDLog table.
[BatchSize]	Batch_Size	Defined in BatchIDLog table.
[ProductID]	Product_ID	Defined in BatchIDLog table.
[ProductName]	Product_Name	Defined in BatchIDLog table.
[RecipeID]	Recipe_ID	Defined in BatchIDLog table.
[RecipeName]	Recipe_Name	Defined in BatchIDLog table.
[RecipeVersion]	Recipe_Version	Defined in BatchIDLog table.
[RecipeState]	Recipe_State	Defined in BatchIDLog table.
[RecipeType]	Recipe_Type	Defined in BatchIDLog table.

Combined ProcessVar + BatchIDLog

Template Placeholder	SQL Column Name	Description
[TimeStamp]	DateTime	Defined in ProcessVar table.
[BatchID]	Batch_ID	Defined in BatchIDLog table.
[UniqueID]	Batch_Log_ID	Used to bind both ProcessVar and BatchIDLog tables. Serves as UniqueID of each batch
[Product]	Product_Name or Product_ID	Defined in BatchIDLog table. This placeholder maps to Product_Name if non-empty, otherwise it maps to Product_ID
[Procedure]	Recipe_ID	Defined in BatchIDLog table.
[Unit]	UnitOrConnection	Defined in ProcessVar table.
[UnitProcedure]	UnitProcedure_ID	Defined in ProcessVar table.
[Operation]	Operation_ID	Defined in ProcessVar table.

Template Placeholder	SQL Column Name	Description
[Phase]	Phase_ID	Defined in ProcessVar table.
[PhaseInstance]	Index derived from Phase_Instance_ID	Defined in ProcessVar table. Provides phase instance index.
[Parameter]	Parameter_ID	Defined in ProcessVar table.
[Descript]	N/A	Contains "Actual_Value" if column Actual_Value is NOT empty, otherwise contains text "Target_Value"
[Value]	Actual_Value or Target_Value	Contains the data from Actual_Value column if it is not empty, otherwise contains data from Target_Value column.
[TargetValue]	Target_Value	Defined in ProcessVar table.
[ActualValue]	Actual_Value	Defined in ProcessVar table.
[EU]	UnitOfMeasure	Defined in ProcessVar table.
[Campaign]	Campaign_ID	Defined in BatchIDLog table.
[Lot]	Lot_ID	Defined in BatchIDLog table.
[Train]	Train_ID	Defined in BatchIDLog table.
[BatchSize]	Batch_Size	Defined in BatchIDLog table.
[ProductID]	Product_ID	Defined in BatchIDLog table.
[ProductName]	Product_Name	Defined in BatchIDLog table.
[RecipeID]	Recipe_ID	Defined in BatchIDLog table.
[RecipeName]	Recipe_Name	Defined in BatchIDLog table.
[RecipeVersion]	Recipe_Version	Defined in BatchIDLog table.
[RecipeState]	Recipe_State	Defined in BatchIDLog table.
[RecipeType]	Recipe_Type	Defined in BatchIDLog table.

Combined ProcessVarChange + BatchIDLog

Template Placeholder	SQL Column Name	Description
[TimeStamp]	DateTime	Defined in ProcessVarChange table.
[BatchID]	Batch_ID	Defined in BatchIDLog table.
[UniqueID]	Batch_Log_ID	Used to join the ProcessVarChange and BatchIDLog tables. Serves as UniqueID of each batch
[Product]	Product_Name or Product_ID	Defined in BatchIDLog table. This placeholder maps to Product_Name if non-empty, otherwise it maps to Product_ID
[Procedure]	Recipe_ID	Defined in BatchIDLog table.
[Unit]	UnitOrConnection	Defined in ProcessVarChange table.

Template Placeholder	SQL Column Name	Description
[UnitProcedure]	UnitProcedure_ID	Defined in ProcessVarChange table.
[Operation]	Operation_ID	Defined in ProcessVarChange table.
[Phase]	Phase_ID	Defined in ProcessVarChange table.
[PhaseInstance]	Index derived from Phase_Instance_ID	Defined in ProcessVarChange table. Provides phase instance index.
[Parameter]	Parameter_ID	Defined in ProcessVarChange table.
[Descript]	N/A	Contains text "ProcessVarChange"
[Value]	New_Target_Value	Defined in ProcessVarChange table.
[TargetValue]	New_Target_Value	Defined in ProcessVarChange table.
[OldValue]	Old_Target_Value	Defined in ProcessVarChange table.
[EU]	UnitOfMeasure	Defined in ProcessVarChange table.
[UserID]	DoneBy_User_ID	Defined in ProcessVarChange table.
[Campaign]	Campaign_ID	Defined in BatchIDLog table.
[Lot]	Lot_ID	Defined in BatchIDLog table.
[Train]	Train_ID	Defined in BatchIDLog table.
[BatchSize]	Batch_Size	Defined in BatchIDLog table.
[ProductID]	Product_ID	Defined in BatchIDLog table.
[ProductName]	Product_Name	Defined in BatchIDLog table.
[RecipeID]	Recipe_ID	Defined in BatchIDLog table.
[RecipeName]	Recipe_Name	Defined in BatchIDLog table.
[RecipeVersion]	Recipe_Version	Defined in BatchIDLog table.
[RecipeState]	Recipe_State	Defined in BatchIDLog table.
[RecipeType]	Recipe_Type	Defined in BatchIDLog table.

Combined PhaseInstruction + BatchIDLog

Template Placeholder	SQL Column Name	Description
[TimeStamp]	DateTime	Defined in PhaseInstruction table.
[BatchID]	Batch_ID	Defined in BatchIDLog table.
[UniqueID]	Batch_Log_ID	Used to bind both PhaseInstruction and BatchIDLog tables. Serves as UniqueID of each batch

Template Placeholder	SQL Column Name	Description
[Product]	Product_Name or Product_ID	Defined in BatchIDLog table. This placeholder maps to Product_Name if non-empty, otherwise it maps to Product_ID
[Procedure]	Recipe_ID	Defined in BatchIDLog table.
[Unit]	UnitOrConnection	Defined in PhaseInstruction table.
[UnitProcedure]	UnitProcedure_ID	Defined in PhaseInstruction table.
[Operation]	Operation_ID	Defined in PhaseInstruction table.
[Phase]	Phase_ID	Defined in PhaseInstruction table.
[PhaseInstance]	Index derived from Phase_Instance_ID	Defined in PhaseInstruction table. Provides phase instance index.
[Parameter]	N/A	Contains text: "PhaseInstruction"
[Value]	Instruction (complex)	Defined in PhaseInstruction table
[Campaign]	Campaign_ID	Defined in BatchIDLog table.
[Lot]	Lot_ID	Defined in BatchIDLog table.
[Train]	Train_ID	Defined in BatchIDLog table.
[BatchSize]	Batch_Size	Defined in BatchIDLog table.
[ProductID]	Product_ID	Defined in BatchIDLog table.
[ProductName]	Product_Name	Defined in BatchIDLog table.
[RecipeID]	Recipe_ID	Defined in BatchIDLog table.
[RecipeName]	Recipe_Name	Defined in BatchIDLog table.
[RecipeVersion]	Recipe_Version	Defined in BatchIDLog table.
[RecipeState]	Recipe_State	Defined in BatchIDLog table.
[RecipeType]	Recipe_Type	Defined in BatchIDLog table.

Combined OperatorComment + BatchIDLog

Template Placeholder	SQL Column Name	Description
[TimeStamp]	DateTime	Defined in OperatorComment table.
[BatchID]	Batch_ID	Defined in BatchIDLog table.
[UniqueID]	Batch_Log_ID	Used to bind both OperatorComment and BatchIDLog tables. Serves as UniqueID of each batch
[Product]	Product_Name or Product_ID	Defined in BatchIDLog table. This placeholder maps to Product_Name if non-empty, otherwise it maps to Product_ID
[Procedure]	Recipe_ID	Defined in BatchIDLog table.

Template Placeholder	SQL Column Name	Description
[Unit]	UnitOrConnection	Defined in OperatorComment table.
[UnitProcedure]	UnitProcedure_ID	Defined in OperatorComment table.
[Operation]	Operation_ID	Defined in OperatorComment table.
[Phase]	Phase_ID	Defined in OperatorComment table.
[PhaseInstance]	Index derived from Phase_Instance_ID	Defined in OperatorComment table. Provides phase instance index.
[Parameter]	N/A	Contains text: "OperatorComment"
[Value]	Operator_Comment (complex)	Defined in OperatorComment table.
[UserID]	DoneBy_User_ID	Defined in OperatorComment table.
[Campaign]	Campaign_ID	Defined in BatchIDLog table.
[Lot]	Lot_ID	Defined in BatchIDLog table.
[Train]	Train_ID	Defined in BatchIDLog table.
[BatchSize]	Batch_Size	Defined in BatchIDLog table.
[ProductID]	Product_ID	Defined in BatchIDLog table.
[ProductName]	Product_Name	Defined in BatchIDLog table.
[RecipeID]	Recipe_ID	Defined in BatchIDLog table.
[RecipeName]	Recipe_Name	Defined in BatchIDLog table.
[RecipeVersion]	Recipe_Version	Defined in BatchIDLog table.
[RecipeState]	Recipe_State	Defined in BatchIDLog table.
[RecipeType]	Recipe_Type	Defined in BatchIDLog table.

Combined MaterialInput + BatchIDLog+BatchDetail

Template Placeholder	SQL Column Name	Description
[TimeStamp]	DateTime	Defined in MaterialInput table.
[BatchID]	Batch_ID	Defined in BatchIDLog table.
[UniqueID]	Batch_Log_ID	Used to bind both MaterialInput and BatchIDLog tables. Serves as UniqueID of each batch
[Product]	Product_Name or Product_ID	Defined in BatchIDLog table. This placeholder maps to Product_Name if non-empty, otherwise it maps to Product_ID
[Procedure]	Recipe_ID	Defined in BatchIDLog table.
[Unit]	UnitOrConnection	Defined in MaterialInput table.
[UnitProcedure]	UnitProcedure_ID	Defined in MaterialInput table.
[Operation]	Operation_ID	Defined in MaterialInput table.

Template Placeholder	SQL Column Name	Description
[Phase]	Phase_ID	Defined in MaterialInput table.
[PhaseInstance]	Index derived from Phase_Instance_ID	Defined in MaterialInput table. Provides phase instance index.
[Parameter]	N/A	Contains text "MaterialInput"
[Descript]	Material_Parameter	Defined in MaterialInput table.
[Value]	Actual_Qty	Defined in MaterialInput table.
[ActualValue]	Actual_Qty	Defined in MaterialInput table.
[TargetValue]	Target_Qty	Defined in MaterialInput table.
[EU]	UnitOfMeasure	Defined in MaterialInput table.
[UserID]	DoneBy_User_ID	Defined in BatchDetail table.
[ReviewerID]	CheckBy_User_ID	Defined in BatchDetail table.
[MaterialID]	Material_ID	Defined in MaterialInput table.
[MaterialName]	Material_Name	Defined in MaterialInput table.
[MaterialLotID]	Mtrl_Lot_ID	Defined in MaterialInput table.
[Campaign]	Campaign_ID	Defined in BatchIDLog table.
[Lot]	Lot_ID	Defined in BatchIDLog table.
[Train]	Train_ID	Defined in BatchIDLog table.
[BatchSize]	Batch_Size	Defined in BatchIDLog table.
[ProductID]	Product_ID	Defined in BatchIDLog table.
[ProductName]	Product_Name	Defined in BatchIDLog table.
[RecipeID]	Recipe_ID	Defined in BatchIDLog table.
[RecipeName]	Recipe_Name	Defined in BatchIDLog table.
[RecipeVersion]	Recipe_Version	Defined in BatchIDLog table.
[RecipeState]	Recipe_State	Defined in BatchIDLog table.
[RecipeType]	Recipe_Type	Defined in BatchIDLog table.

Combined MaterialInputChange + BatchIDLog

Template Placeholder	SQL Column Name	Description
[TimeStamp]	DateTime	Defined in MaterialInputChange table.
[BatchID]	Batch_ID	Defined in BatchIDLog table.
[UniqueID]	Batch_Log_ID	Used to bind both MaterialInputChange and BatchIDLog tables. Serves as UniqueID of each batch
[Product]	Product_Name or Product_ID	Defined in BatchIDLog table. This placeholder maps to Product_Name if non-empty, otherwise it maps to Product_ID
[Procedure]	Recipe_ID	Defined in BatchIDLog table.
[Unit]	UnitOrConnection	Defined in MaterialInputChange table.

Template Placeholder	SQL Column Name	Description
[UnitProcedure]	UnitProcedure_ID	Defined in MaterialInputChange table.
[Operation]	Operation_ID	Defined in MaterialInputChange table.
[Phase]	Phase_ID	Defined in MaterialInputChange table.
[PhaseInstance]	Index derived from Phase_Instance_ID	Defined in MaterialInputChange table. Provides phase instance index.
[Parameter]	N/A	Contains text: "MaterialInputChange"
[Descript]	Material_Parameter	Defined in MaterialInputChange table.
[Value]	New_Target_Qty	Defined in MaterialInputChange table.
[TargetValue]	New_Target_Qty	Defined in MaterialInputChange table.
[OldValue]	Old_Target_Qty	Defined in MaterialInputChange table.
[MaterialID]	Material_ID	Defined in MaterialInput table.
[UserID]	DoneBy_User_ID	Defined in MaterialInputChange table.
[ReviewerID]	CheckBy_User_ID	Defined in MaterialInputChange table.
[Campaign]	Campaign_ID	Defined in BatchIDLog table.
[Lot]	Lot_ID	Defined in BatchIDLog table.
[Train]	Train_ID	Defined in BatchIDLog table.
[BatchSize]	Batch_Size	Defined in BatchIDLog table.
[ProductID]	Product_ID	Defined in BatchIDLog table.
[ProductName]	Product_Name	Defined in BatchIDLog table.
[RecipeID]	Recipe_ID	Defined in BatchIDLog table.
[RecipeName]	Recipe_Name	Defined in BatchIDLog table.
[RecipeVersion]	Recipe_Version	Defined in BatchIDLog table.
[RecipeState]	Recipe_State	Defined in BatchIDLog table.
[RecipeType]	Recipe_Type	Defined in BatchIDLog table.

Combined MaterialOutput + BatchIDLog

Template Placeholder	SQL Column Name	Description
[TimeStamp]	DateTime	Defined in MaterialOutput table.
[BatchID]	Batch_ID	Defined in BatchIDLog table.
[UniqueID]	Batch_Log_ID	Used to bind both MaterialOutput and BatchIDLog tables. Serves as UniqueID of each batch

Template Placeholder	SQL Column Name	Description
[Product]	Product_Name or Product_ID	Defined in BatchIDLog table. This placeholder maps to Product_Name if non-empty, otherwise it maps to Product_ID
[Procedure]	Recipe_ID	Defined in BatchIDLog table.
[Unit]	UnitOrConnection	Defined in MaterialOutput table.
[UnitProcedure]	UnitProcedure_ID	Defined in MaterialOutput table.
[Operation]	Operation_ID	Defined in MaterialOutput table.
[Phase]	Phase_ID	Defined in MaterialOutput table.
[PhaseInstance]	Index derived from Phase_Instance_ID	Defined in MaterialOutput table. Provides phase instance index.
[Parameter]	N/A	Contains text "MaterialOutput"
[Descript]	Material_Parameter	Defined in MaterialOutput table.
[Value]	Actual_Qty	Defined in MaterialOutput table.
[ActualValue]	Actual_Qty	Defined in MaterialOutput table.
[TargetValue]	Target_Qty	Defined in MaterialOutput table.
[EU]	UnitOfMeasure	Defined in MaterialOutput table.
[MaterialID]	Material_ID	Defined in MaterialOutput table.
[MaterialName]	Material_Name	Defined in MaterialOutput table.
[Campaign]	Campaign_ID	Defined in BatchIDLog table.
[Lot]	Lot_ID	Defined in BatchIDLog table.
[Train]	Train_ID	Defined in BatchIDLog table.
[BatchSize]	Batch_Size	Defined in BatchIDLog table.
[ProductID]	Product_ID	Defined in BatchIDLog table.
[ProductName]	Product_Name	Defined in BatchIDLog table.
[RecipeID]	Recipe_ID	Defined in BatchIDLog table.
[RecipeName]	Recipe_Name	Defined in BatchIDLog table.
[RecipeVersion]	Recipe_Version	Defined in BatchIDLog table.
[RecipeState]	Recipe_State	Defined in BatchIDLog table.
[RecipeType]	Recipe_Type	Defined in BatchIDLog table.

PI Event Frames Interface Manager reference

The PI Event Frames Interface Manager configuration tool enables you to configure settings for batch interfaces. To configure settings, go to the **Interface Selection** tab and select the interface instance you want to configure, or add a new instance. On the **File Selection** tab, specify the interface settings file that stores settings and the configuration for the interface instance. The interface settings (.ini) file contains the interface startup parameter and configuration settings. Be sure to specify the .ini extension.

On the remaining tabs, configure the settings for generating batch data. When you are done, go to the **Save Settings** tab to save your entries.

To start the interface service with PI Event Frames Interface Manager, click **Start Interface Service** on the **Service Configuration** tab. To stop the interface service click **Stop Interface Service.**

The following sections describe the settings on the tabs of the PI Event Frames Interface Manager configuration tool.

Server Information tab

The **Server Information** tab is where you configure the PI Data server and PI Asset server that you intend to use with the interface instance. The interface can generate either batches in the PI Batch Database or event frames in the PI Asset Framework.

- **PI Data server (/HOST)**

  Specifies the PI Data server to which the interface batch data. *Host* is the IP address or fully-qualified domain name of the PI Data server. If the server that you want to use is not listed, add it to the known servers table using the AboutPI-SDK application.

- **User and Password**

  For PI Data Archive version 3.4.380.36 and higher, use Windows Integrated Security for authentication. Omit the user name and password from these fields, and ensure that the Windows account that runs the interface has sufficient permissions on the PI Data server to write data to PI points. For versions prior to version 3.4.380.36, configure a trust that permits access for the user that runs the interface service.

- **[PI Server] Port**

  The port number for TCP/IP communication. The default port (recommended) is 5450.

- **Create event frames**

  Check this box to create event frames in PI Asset Framework, instead of creating batches in the PI Batch Database.

- **Host and Database**

  (/AFHOST and /AFDATABASE): The PI Asset server and database where you want to create event frames.

- **User and Password**

  If you are not using Windows Integrated Security for authentication, enter the user name and password for the Windows user account that you intend to use to connect to PI Asset Framework.

Source tab

On this tab you configure the data sources that you want the interface to read. To add a data source, right-click the top Sources node and choose the Add option. After you add the first data source, all additional data sources must be the same type (event journal or SQL database); the configuration utility no longer offers you the other option.

Active sources are displayed in black text, inactive sources in gray. To activate or deactivate a source, right-click it and choose the desired option.

To configure the SQL Alarm and Event Historian, you must specify mappings that ensure that process cell data is recorded correctly, because DeltaV does not emit process cell information. To map process cells to areas, click Area to Process Cell… and, for each unique area, specify the process cell to be recorded with its events. Alternately, to assign units to process cells, click Unit to Process Cell…

Templates

Templates map data from the data source to PI tags. Depending on your interface, you can define the following types of templates:

- Tag: Maps data to PI tags.
- Attribute: Maps data from the data source to event frame attributes.
- Property: Maps data to batch properties in the PI Batch Database.
- Recipe: Map the data source recipe levels to the levels used by the PI System.
- Alarm tag: Writes data to a PI tag when the data source raises an alarm (Emerson DeltaV only).

The data source emits its data as strings, and you can configure templates that map the source data to string, integer or float data types. For each type of template, you can specify triggers and enter comments to be stored in the initialization file, and you can activate and deactivate templates.

When defining templates, you specify the data to be written and the events that trigger the update. To define templates using PI Event Frame Interface Manager, go to the Templates page and navigate to the desired type of template. The following sections provide details about the specific types of templates.

Tag templates

Tag templates create and update PI tags when events are read from the data source. Alarm tag templates create and update PI tags based on alarms raised by the Emerson DeltaV Alarms & Events Historian. If you define one or more triggers, the target PI tag is updated only when the specified events occur. If you do not define any triggers, every event from the data source triggers an update of the target PI tag.

Tag configuration

- **Index**

  Specify a unique numeric identifier for the entry.

- **Name**

  The name of the field from the data source. The name can include wildcards.

- **Value**

  The value to be written to the PI tag. Composed of placeholders, which are aliases for data fields read from the data source.

- **Type**

  The data type to be used to write the value to the tag. Note that if the incoming value is incompatible with the tag type, the event is not processed, and an error is logged.

- **Translate**

  Maps text from the data source to the text that you want to record in the PI System.

Advanced features (attributes)

- **Descriptor**

  Populates the descriptor field for the target PI tag.

- **Engineering units**

  The engineering units for the data.

- **Unit Alias**

  Specifies an alias to be recorded under the **Aliases** node in the corresponding PIUnit or AF element.

- **Phase Alias**

  Specifies an alias to be recorded under the **Phases** > **Aliases** node in the corresponding PIUnit or element.

- **Annotation**

  Specifies the annotation to be associated with the event. The result is stored in a PI tag as a string.

- **Use the NamedValues collection for annotations**

  Store annotations in a PI tag as a name-value collection. The name is derived from the event sent by the data source.

- **Remove annotations from tag values**

  Store values without annotations.

Attribute and property templates

By default, this page displays a **Property** tab. If you choose to generate event frames, the Templates page displays an **Attribute** tab instead.

These tabs enable you to configure how data from the data source is written to PI AF attributes or batch database properties.

Configuration

- **Index**

  Assigns a unique numeric identifier for the template.

- **Name**

  Specifies how the target attribute or property is named. You can use placeholders to configure values emitted by the data source. The interface adds attributes and properties as required when new equipment is added.

- **Value**

  Specifies the value to be recorded. Use placeholders to derive values from the data emitted by the data source.

- **Data type**

  PI data type of the value. Note that if the incoming value is incompatible with the tag type, the event is not processed, and an error is logged.

- **Translate**

  Maps text from the data source to the text that you want to record in the PI System.

- **UOM**

  Unit of measure to be used to store value, if different from unit provided by data source.

Advanced features (attributes)

- **Descriptor**

  Populates the tag's Descriptor attribute.

- **Engineering units**

  Unit of measurement.

- **Level**

  Specifies the recipe level as follows:

Level value	Corresponds to:
-1	Same level as in data source
1	Unit Procedure
2	Procedure
3	Operation
4	Phase

Level value	Corresponds to:
5	Phase State
6	Phase Step

- **Category**

  Specifies the PI AF category to be associated with the value.

Recipe templates

Recipe templates enable you to override the default recipe name read from the data source.

Configuration

- **Index**

  Specifies the level in recipe hierarchy as follows:

Level	Index	PI object	Default
Procedure	1	PIBatch Recipe	Recipe[1].Name =[Procedure]
Unit Procedure	2	PIUnitBatch Procedure	Recipe[2].Name = [UnitProcedure]
Operation	3	PISubBatch Name field	Recipe[3].Name =[Operation]
Phase	4	PISubBatch Name field	Recipe[4].Name =[Phase]
Phase State	5	PISubBatch Name field	Recipe[5].Name =[PhaseState]
Phase Step	6	PISubBatch Name field	Recipe[6].Name =[PhaseStep]

- **Name**

  Defines the naming convention used by the interface to assign names to batch events. For event frames, this template modifies the Recipe AF attribute. For the Batch database, this template modifies the PIBatch name.

  For example: *abc_[Procedure]* If the incoming event's [Procedure] field contains "Test", the resulting procedure **Recipe** field is "abc_Test".

- **Value**

  The value to be written to the property or attribute. Can be defined using placeholders as well as fixed text.

Advanced features

- **Batch ID**

  Configures the batch ID of the particular recipe object, overriding the incoming (default) batch ID. If you override the batch ID for the procedure, the batch ID is propagated to the child unit batches Batch ID field. For event frames, this template modifies the event frame name.

  > **Note:**
  > If you use a recipe template to set the batch ID, the recipe template overrides any batch ID mask you might have configured to enable merging of batches.

- **Module/Element Path**

  Redirects the entry to a unit or PI AF element other than the one in the event. Specifies the path for the desired module or AF element. Supports only unit procedure (level 2 and 4).

- **Product**

  Specifies the product of the particular recipe object. Supports the procedure and unit procedure [Product] fields, which must be present in the source event that creates the batch. If a product trigger is not defined, this template is populated based on the data in the event that creates the particular Recipe object.

- **Product Trigger**

  Populates the [Product] field of the particular recipe object after the object is created, which is useful if the product is defined in a separate event.

  For example: `[Parameter, Value="Recipe Header"] [Descript, value="Product Name"]`.

- **Merge same named objects under parent**

  Combines identically-named child objects.

- **Event Frame Template**

  Specifies the template to be used to create event frames for this recipe.

Triggers

To specify the events that initiate updates to tags, properties or attributes, you define triggers. You can define triggers for events read from the data source and for batch-related events raised by the PI System itself (PI Events). To define triggers using PI Event Frame Interface Manager, go to the Templates page, navigate to the desired tag, property or attribute template, and click the Triggers tab.

If you omit triggers, the target is updated by every event. If you specify multiple conditions in a single trigger, data is written only when all conditions are met (logical AND). If you define multiple triggers, data is written when any one of the conditions is met (logical OR).

In the following example, the template is triggered when the PI System records the start of a batch: `[Event,value="PIEVENT"] [Descript, value="BATCH"] [Pval, value="START"]`

Filters

You use the **Filters** tab to configure the recipes, units, phases, or phase states to be excluded from processing by the interface.

- **Skip Phases (/SKIPPHASES)**

  The interface ignores any event that contains the specified phases in the [Phase] or [Phasemodule] column.

- **Skip Units (/SKIPUNITS)**

  The interface ignores any event containing the specified units in the [Unit] column.

- **Skip Recipes (/SKIPRECIPES)**

  The interface ignores any event containing the specified recipe in the appropriate column ([Procedure] for a procedure recipe, [UnitProcedure] for a unitprocedure recipe, and so on.)

- **Exclude Phase States (/EXCLUDESTATES)**

  The interface ignores phase state events that contain the specified phase states in the [PhaseState] column.

Time Settings tab

You use the **Time Settings** tab to configure the settings that control how the interface handles loss of connectivity and how it processes data.

Query time settings

- **Scan (/SCAN=<seconds>)**

  Specifies how frequently the interface scans the data source.

- **Cache time (/CACHETIME=<days>)**

  Specifies how long completed events are retained in memory. Default is one day. Specify the maximum duration expected between events that need to be merged, plus any desired margin of safety.

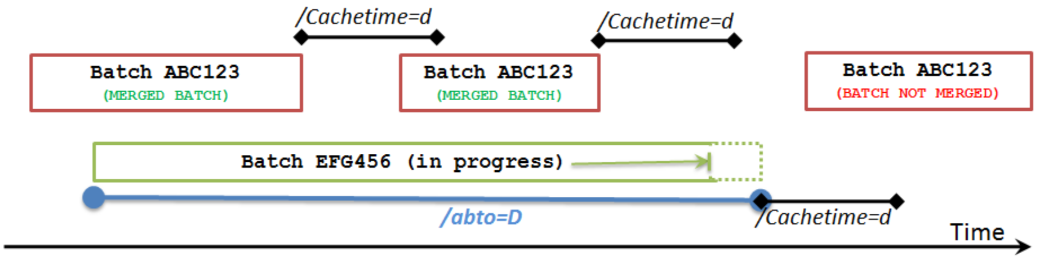
  The value can be specified as whole day or fraction of a day. For example, to release completed batches when their end time is less than 7 days and 12 hours from current time, specify the following cache time setting:

  ```
  CACHETIME=7.5 days
  ```

- **Abandoned batch timeout (/ABTO=<days>)**

  Specifies how long batches can remain open before being considered abandoned. When this period (plus cache time) elapses, the interface writes an end time to the event to close it. Specify the maximum duration expected for events, plus any desired margin of safety.

  For example, if you set abandoned batch timeout to 50.5 days and cache time is seven days, events open for 57.5 days are automatically closed. The following figure illustrates timeout logic.



- **Maximum query time frame (/MAXQTF=<days>)**

  To help manage workload and memory usage, defines the maximum time frame for queries. For example, if you specify 30 days and the interface queries for one year's worth of data, the interface issues 12 one-month queries rather than one (large) one-year query.

- **Maximum stop time (/MAXSTOPTIME=<seconds>)**

  Specifies the maximum time allowed for the interface to properly shutdown. If the shutdown process takes longer than the specified time, the interface is forced to terminate immediately. The default value is 120 seconds.

- **Use local time stamps to process incoming events (/TS)**

  Specifies whether timestamps from an SQL data source are interpreted as local time or GMT. By default, timestamps are interpreted as GMT.

PI connection settings

- **PI connection timeout (/PICONNTO=<seconds>)**

  Override the default PI SDK Connection TimeOut property.

- **PI data access timeout (/PIDATO=<seconds>)**

  Override the default PI SDK Data Access TimeOut property.

- **Retry (/RETRY=<seconds>)**

  Specifies how often the interface retries failed attempts to write data. The default is 60 seconds.

- **Retry timeout (/RETRYTO=<seconds>)**

  Specifies how long the interface continues retrying an attempt to write data. To avoid data loss, set to 0 (default, no timeout).

Operational Settings tab

You configure the settings on the **Operational Settings** tab to configure the interface mode and other related settings.

- **Runtime mode (/MODE=<mode>)**

  Interface modes are as follows:

Mode	Description
Realtime (default)	Scan data source to collect data in realtime

Mode	Description
Recovery	Scan data source and generate or correct events accordingly. always starts in recovery mode, then switches to realtime mode.
Statistics	Compare data source history against events and report results without updating any data.
Delete	Delete events for a specified period.

- **Perform one scan then stop (/SINGLERUN)**

  The interface performs one scan of active points, then exits.

- **Print result of first scan to file (/PRINT=<file name>)**

  Print the results of the first scan to the specified text file. The results include the event frame hierarchy tree, the tag list, and the equipment tree. This parameter is designed primarily for troubleshooting and configuration testing when the interface is run in statistics mode.

- **Debug level (/DB=<#>)**

  Specifies level of detail for logging as follows:

  - 0: Log errors and warnings (default)
  - 1: Log errors, warnings and major successes
  - 2: Verbose logging

- **Numeric settings (/NS=<lang>)**

  Configures how numeric values are formatted by the interface, to enable the interface to properly interpret numeric values based on the machine’s regional setting or a user-specified language. Default is "English_UnitedStates".

- **Interface ID (/ID=x)**

  Specifies the numeric interface instance identifier (maximum nine digits). To detect PI points maintained by the interface instance, the interface matches this setting against the value in the points’ *Location1* attribute.

- **Point source (/PS=x)**

  Point source for the interface instance. Point source is not case sensitive. Corresponds to the *PointSource* attribute of individual PI Points. The interface loads PI points with the same point source.

- **Associate all reference elements with child Event Frames (/DPRETC)**

  When creating event frames in PI AF, by default all reference elements are associated with child event frames.

Failover settings

- **Failover tag (/FAILOVERTAG=x)**

  The PI tag to be used to coordinate failover.

- **Failover ID (/FAILOVERID=x)**

  A unique identifier for this interace instance, used to coordinate which instance is primary.

- **Failover swap time (/SWAPTIME=<seconds>)**

  Specify how long an interface can be inactive before failover to another instance occurs.

Security settings

- **Specify point security (/PTSEC=x)**

  Override the default point security created by the interface.

- **Specify data security (/DATASEC=x)**

  Override the default data security of PI points created by the interface.

Test Configuration tab

You test the configuration settings in the **Test Configuration** tab.

Specify test settings as follows, then click **Run Test**, and check the output file for results.

Field	Description
BAT File	The path to the interface .BAT file to run during the test execution
Output File	The path to the text file where the test results are written
Start Time	Start time for scanning
End Time	End time for scanning

Security overview for PI interfaces for batch and manufacturing execution systems

To configure batch interfaces, the user account under which the PI Event Frames Interface Manager runs must be in the local Administrators group. Set the following permissions for the user that runs the interface and all users who need to run the PI Event Frames Interface Manager:

- PI Data Archive permissions (PI SMT: Browse to **Security > Database Security**)
  - Database security: Enable read/write access for the PIPOINT table and read access for PIBACKUP
  - Point Database security: Set both PtSecurity and DataSecurity to read/write
  - Enable read access to the active points
- PI Asset Framework permissions
  - Database: read/write
  - Categories: read
  - Element: read/write

- Element templates: read
- Event frames: read/write

If you are running PI Data Archive 3.4.380.36 or later, you can take advantage of its support for Windows Integrated Security by running the interface service using a Windows account that has the required permissions on the PI Data server. To configure Windows Integrated Security, use PI SMT to define a mapping that assigns a PI identity that has the required permissions to the user or user's group.

For pre-3.4.380.36 versions of the PI Data Archive, you must create a PI trust for the user that runs the interface and configuration tool. Limit the trust to the hostname or IP address of the interface node and the application name (BIFConfig.exe for the PI Event Frames Interface Manager).

Assign permissions for user accounts and PI points

The user that runs the interface requires read/write access to the PI Data Archive.

To create event frames and write data to elements and attributes, the user must have permission to connect to the PI Asset Framework.

To configure the interface, the user must be in the local Administrators group.

Assign the required permissions to the user that runs the interface and all users who need to run the PI Event Frames Interface Manager configuration tool.

Procedure

1. To assign permissions for the PI Data Archive, launch PI System Management Tools, and click **Security** > **Database Security**.
2. For the user account under which the batch interfaces run, set the PIPOINT table to read and write permission.
3. For the user account under which the batch interfaces run, set the PIBACKUP table to read permission.
4. For PI points maintained by batch interfaces, set security as follows:
   a. Set the PtSecurity to read and write permission for any point that the interface creates.
   b. Set the DataSecurity to read and write permission for any point to which the interface writes data.

   You can set PI point permissions using PI System Management Tools by choosing **Points** > **Point Builder**.

   You can change point settings in bulk using the PI SMT plug-in for Microsoft Excel.

Configure security for the PI Asset Framework

To configure batch interface security settings for the PI Asset Framework, perform the following steps.

Procedure

1. Launch PI System Explorer.
2. Click **Database** on the toolbar.

The Select Database window opens, listing existing databases.

3. Right-click the database where you intend to store the batch event frames, and then click **Security**.
4. Browse to the category and enter the required settings:
   a. Set **Database** to read/write.
   b. Set **Categories** to read.
   c. Set **Element** to read/write.
   d. Set **Element templates** to read.
   e. Set **Event frames** to read/write.

Configure security for the PI Data Archive

If you are running PI Data Archive 3.4.380.36 or later, you can take advantage of its support for Windows Integrated Security by running the interface service using a Windows account that has the required permissions. To use Windows Integrated Security, use PI System Management Tools to define a mapping that assigns the Windows user (or group) a PI identity that has the required permissions.

For pre-3.4.380.36 versions of the PI Data Archive, you must create a PI trust for the user who runs the interface and configuration tool. For tightest security, limit the trust to the hostname or IP address of the interface node and the application name.

If you are installing the interface on a node other than the PI Data server, you must create trusts to ensure that the configuration tool and the interface have access to the server.

Procedure

1. To create a trust, launch PI System Management Tools and connect to the target server.
2. Click **Security**, and then click **Mappings & Trusts**.
3. Right-click within the **Trusts** tab, and click **New Trust...** . The **Add Trust** wizard launches.
4. Enter a meaningful name and description for the trust.
5. Configure the following settings:

Program	Type of Trust	Application Name
PI Event Frame Interface Manager	PI-API application	`BIFConfig.exe`
Interface executable	PI-SDK application	Executable name

Configure interface instances for failover

To ensure that batch data continues to be collected if an instance of the interface fails, you can configure multiple interface instances to run in failover mode. The instances must be configured identically.

To enable failover, perform the following steps before you start the interface instances:

Procedure

1. Configure instances of the interface on different host computers. Assign the same interface ID and point source, and assign unique failover IDs to each instance.

   You can configure more than two instances for failover.

2. In the target PI Data Archive, create a string tag and configure it with the same point source and Location1 (interface ID) as the interfaces.

   The interface instance uses this string tag to coordinate failover.

3. Using PI Event Frame Interface Manager, go to the **Operational Settings** tab and configure the failover settings.

   - **Failover tag**

     The name of the tag that you created in the previous step.

   - **Failover identifier**

     A unique ID for the interface instance.

   - **Failover swap time**

     The amount of time that the current primary interface must be unavailable before failover occurs.

4. Start the interface instances.

   When the primary interface instance is operational, it updates the failover tag with the timestamp of the last value written. The backup instance monitors this tag and, if the failover swap time elapses and the failover tag has not been updated, the backup instance takes over data collection and becomes the primary instance.

Command-line parameter reference

To configure an interface, you use the PI Event Frames Interface Manager, which maintains a Windows command (.bat) file that specifies settings using command line parameters. This appendix describes the command line parameters in the .bat file and is provided for troubleshooting purposes.

Note:
To ensure a valid .bat file, do not edit the file manually. Always use PI Event Frames Interface Manager to configure settings.

The following table is a compilation of the command line parameters from all the OSIsoft batch framework interfaces. Some parameters are specific to an interface. To list the parameters supported by your interface, invoke its executable at the command line, specifying the -? flag.

Parameter	Description
`/abto=<#days>`	(Optional) Specifies how long, in addition to the CACHETIME setting, the interface waits before closing a batch for which no end event has arrived. When an abandoned batch is closed, the interface uses the timestamp of its last event as the end time and logs an "Abandoned batch found" message. Default is 100 days, minimum is .042 days (approximately one hour), and maximum is 365 days.
`/adu=[true | false]`	(Optional) Enable the creation of unit batches for recipes in units that are allocated at the phase level rather than the unit batch level. By default, the interface requires the unit name to be present in the unit batch start event. When you enable / adu, the interface creates the unit batch and defers setting the unit name until the phase-level allocation event arrives.

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td><b>/bidm=<list></b></td><td>
(Optional) Override the incoming batch ID by selecting a substring. Specify one or more masks composed of text and wildcards, to be used to compose the desired batch ID from the contents of the source BatchID field. The resulting batch ID is used for the Batch ID field of the top-level procedure and for the [BATCHID] placeholder. By default, the batch ID field in unit procedures contain the full batch ID from the data source. To use the truncated batch ID instead, configure the <b>TBID</b> setting.
Valid wildcards are as follows:
<table>
<tr><th>Wildcard</th><th>Description</th></tr>
<tr><td>#</td><td>Single digit numerical value, 0-9</td></tr>
<tr><td>@</td><td>Single alpha character, a-z, A-Z</td></tr>
<tr><td>?</td><td>Any single symbol</td></tr>
<tr><td>!</td><td>Repeat the previous mask symbol</td></tr>
<tr><td>*</td><td>Any set of symbols</td></tr>
</table>
For example, if the incoming Batch ID is: <b>lot30112 / 90dev123 / 12345stp / ld567</b>
The following list shows example masks and resulting data.
<table>
<tr><th>Mask</th><th>Result</th></tr>
<tr><td>"#####"</td><td>30112</td></tr>
<tr><td>"##@!"</td><td>90dev</td></tr>
<tr><td>"*##@!"</td><td>lot30112 / 90dev</td></tr>
<tr><td>"@@@@, #8dev4, #!"</td><td>30112</td></tr>
</table>
In the last example, the first and second masks do not match, so the third mask is used.
</td></tr>
<tr><td><b>/cachetime=<days></b></td><td>(Optional) Batches are cached in memory to enable the interface to capture events sent by the BES after the batch has closed. This setting specifies how long (in days) completed batches are cached. To specify fractions of a day, use decimal values. For example, <b>/cachetime=7.5</b> releases completed batches when their end time is more than 7 days and 12 hours from the current time. The default value is 1 day, minimum is .042 days (approximately one hour), and maximum is 60 days.</td></tr>
</table>

Parameter	Description
/dac	(Optional) Disable arbitration counters: directs the interface to release a unit on the first resource release event even if the number of acquire events is higher than number of release events. By default, the interface requires the number of acquire and release events for a unit to be the same.
/damcae	(Optional) Ignore events from Alarms and Events data source when creating or checking PI Module Database objects. If the module path defined for an AlarmTag[#].Alias entry contains $ (root node symbol), the interface checks the module path regardless of whether this option is enabled.
/datasec=<string>	(Optional) Specifies the security settings to be assigned to interface-generated tags. For PI Data Archive 3.4.375.99 or earlier, use owner, group, world format. Example: **/datasec="o:rw g:r w:r"** For PI Data Archive 3.4.380.36 or later, specify an access control list (ACL). Example: **/datasec="piadmin: A(r,w) | PIEngineers: A(r)"**
/db=[#]	(Optional) Enabled debugging output: - **0:** Log only errors and warnings (default) - **1:** Log errors, warnings and major success messages - **2:** Log all messages (most verbose)
/dpretc	(Optional – event frames only) Disable propagation of referenced elements to children. By default, the interface propagates each event frame element reference to its children event frames.
/dumpread=<filename>	(Optional) Read data from a dump file. For troubleshooting only.
/dumpwrite=<filename>	(Optional) Create a dump file containing the data currently being processed by the interface. For troubleshooting only.
/equipmentXML=<filepath>	(Optional) Specifies the location of the DeltaV-generated equipment hierarchy XML file. The EMDVB interface uses this reference data to locate missing ProcessCell field by searching based on the combination of Area and Unit fields. Valid only when a DeltaV AE SQL datasource is defined. Example: **/EquipmentXML="C:\DeltaV\EquipHier.xml"**
/failoverID=<string>	(Optional) Configure a unique failover ID for the interface instance. Must be used with the **/FailOverTag** parameter. Example: **/FailOverID="intf1"**

Parameter	Description
/failovertag=<tag name>	(Optional) Specifies the PI tag to be used to track the primary interface instance. Must be used with the **/FailOverID** parameter. Example: **/FailOverTag="Batch_FailoverTag"**
/host=host:port	(Required) Specifies the PI Data server where data is to be stored. *Host* is the IP address or domain name of the server node. *Port* is the port number for TCP/IP communication. The port is always 5450. Examples: **/host=marvin** **/host=marvin:5450** **/host=206.79.198.30** **/host=206.79.198.30:5450**
/ibfd=<#milliseconds>	**(Werum PAS-X only) Inter Basic Function Delay:** To reduce the CPU load spike imposed on the PAS-X database server, you can specify a delay between the queries that the interface issues. By default, the queries are issued without any intervening delay.
/id=<identifier>	(Required) Specify a one- to nine-number identifier for the interface instance. Assigned to the Location1 attribute of tags that are updated by the interface instance.
/inifile=<path>	(Optional) Override the default location and name for the initialization file. By default, the .ini file resides in the interface installation directory and has the same file name and the .bat file.
/link=<AF element path>	(Optional – event frames only) Combine batches from different interface instances under the specified AF element. Useful when you have an MES controlling multiple BES's. Configure an interface instance for each BES, specifying the same linkage element. The interface instances create event frame references under the linkage element, providing a consolidated view of the activity.
/maxqtf=<days>	(Optional) Sets the maximum number of days for which a query can return data. Used to break a large query into a set of smaller queries, to ensure that the system doesn't run out of memory. The value can be fractional. • Minimum: 0.001 • Maximum: 180 • Default: 30
/maxstoptime=<seconds>	(Optional) Specifies (in seconds) the maximum time allowed for the interface to properly shutdown. If shutdown takes longer than the specified time, the interface is forced to terminate immediately. Default: 120 seconds

Parameter	Description
`/merge`	(Optional) Enable merging of multiple source batches with same ID. The original data for each batch merged is stored in PI properties under a node named using the ID of the original batch. The data includes the original batch ID, start time (UTC), end time (UTC), product and formula name. The interface merges only batches that are cached in local memory. The time frame for merging is configured using the **`/cachetime`** switch. If the IDs of the batches you want to merge are different, use the /bidm flag to override incoming IDs. Example: Given the following five batches running within the cache time frame: • Test12345_1 • Test_12345_2 • CleaningTest • USPO12345_test • CleaningTest With merging enabled, only the Cleaningtest batches are merged. To merge the other three batches, which have IDs containing the string "12345", specify **`/bidm=#####`**.

Parameter	Description
/mode=<mode>	(Optional) Valid modes are as follows: **Realtime:** (Default) Real-time data collection. If a recovery start time is specified (**/rst**), the interface recovers data before starting realtime collection. **Stat:** Statistics mode. Compare source data with the corresponding PI System batch data. The interface does not write to or modify any data PI batch data. On completion, the interface reports results and stops. **Delete:** Delete batch data from PI archives for specified period, leaving data from all other sources intact. Use only if the interface is unable to synchronize source batch data with the PI System. Must be used in conjunction with the recovery mode switches (**/rst** and **/ret**). **NoData:** Newly-added tags, units and modules are indexed (referenced) in the primary PI archive, but older archives do not have entries for these modules, units and tags. In **NoData** mode, the interface creates modules, units, tags and tag aliases without processing batch data and writing events to the tags. To recover batch data for a period prior to the one in the primary archive, you must reprocess older archives with the offline archive utility. Manual archive reprocessing creates indexes for newly-added units, modules, tags. Always run the interface in this mode before writing new batch data to older PI archives (that is, archives other than the primary archive).
/mop	(Optional) Merge identically-named operations under the same parent unit procedure. The start time of the combined operation is the start of the earliest operation and the end time is the end time of the latest or longest operation that was merged.
/mup	(Optional) Merge identically-named sequential unit procedures running on the same unit into a single unit procedure. Unit procedures are not merged if the unit was used by another recipe between candidates for merging. The start time of the resulting merged unit procedure is the start of the earliest unit procedure, and the end time is the end time of the latest or longest unit procedure that was merged.
/noarbitration	Create unit batches based solely on source batch recipe data. For use when the source Batch Executive System (BES) provides batch data without equipment arbitration.

Parameter	Description
`/ns[=lang]`	(Optional) Perform numerical conversions using the specified language's conventions. Useful when the numerical conventions differ from the default settings (for example, comma instead of decimal point). Default is "English_UnitedStates". If you omit the language parameter, the interface uses the "Regional and Language Options" settings in effect for the interface node. Language types and abbreviations: • chinese chinese-simplified (chs) • chinese-traditional (cht) • czech (csy) • danish (dan) • belgian, dutch-belgian (nlb) • dutch (nld) • australian, english-aus (ena) • canadian, english-can (enc) • english english-nz (enz) • english-uk (uk) • american, american-english, english-american, english-us, english-usa, (enu) (us) (usa) • finnish (fin) • french-belgian (frb) • french-canadian (frc) • french (fra) • french-swiss (frs) • german-swiss, swiss (des) • german (deu) • german-austrian (dea) • greek (ell) • hungarian (hun) • icelandic (isl) • italian (ita) • italian-swiss (its) • japanese (jpn) • korean (kor) • norwegian-bokmal (nor) • norwegian norwegian-nynorsk (non) • polish (plk) • portuguese-brazilian (ptb) • portuguese (ptg) • russian (rus) • slovak (sky) • spanish (esp) • spanish-mexican (esm) • spanish-modern (esn) • swedish (sve) • turkish (trk)

Parameter	Description
/piconnto=<seconds>	(Optional) Override the default SDK setting for PI connection timeout.
/pidato=<seconds>	(Optional) Override the default SDK setting for PI data access timeout.
/pipswd=<password>	(Optional) Specify the user password to be used to connect to the PI Data Archive. By default, the interface uses PI trusts for authentication.
/piuser=<name>	(Optional) Specify the user name to be used to connect to the PI Data Archive. By default, the interface uses PI trusts for authentication.
/print=<filename>	(Optional) Prints the results of first scan to specified text file. The results include the batch tree, tag list, and equipment tree. Used for troubleshooting.
/ps=pointsource	Specifies the point source for the points maintained by the interface.
/ptsec=<string>	(Optional) Specifies the access security settings to be assigned to interface-generated tags. For PI Data Archive version 3.4.375.99 or earlier, use owner, group, world format. Example: **/ptsec="o:rw g:r w:r"** For PI Data Archive version 3.4.380.36 or later, specify an access control list (ACL). Example: **/ptsec="piadmin: A(r,w) | PIEngineers: A(r)"**
/ras=<startstr, stopstr>	(Optional) Directs the interface to use the "Report" event to create phase steps under active phase states. The phase step name and start/stop events are obtained from the "Descript" column. The start and stop strings must start in the same position in the data source and must not be the first characters in the "Descript" column.The phase step name is derived from the characters preceding the start/stop text. Specify the strings in a double-quoted comma-separated list, as shown in the example below. Example: **/ras="-STRT, -STOP"** If the Descript Column contains TEST123-STRT-B, the interface generates a phase step named "TEST123" under the currently active phase state. Open phase steps are closed by the end of the parent operation and not by the end of parent phase or phase state.
/restore	For the ABB 800xA interface, enable recovery of batches from restored archives in all configured ABB 800xA data sources.

Parameter	Description
/restef	(Optional) Enables an event frame with references to inherit security settings from its primary reference element.
/ret=<datetime>	(Optional) Specifies the end time for data recovery. The interface recovers batches that start before the specified time, including batches that end after the specified end time. Specify the time using the interface node format and time zone.
/retry=<seconds>	(Optional) Specifies how often the interface retries a failed attempt to write data. The default is 60 seconds.
/retryto=<seconds>	Specifies how long the interface retries a failed attempt to write data before timing out. By default, the interface never times out. If you configure a timeout setting, be advised that you risk losing data.
/rst=<datetime>	(Optional) Specifies recovery start time. The interface recovers batches that start after the specified time, as well as batches that start before the specified time but end after it. Specify the time using the interface node format and time zone.
/rti	Remove trailing index from Recipe fields. Applicable to Procedure, Unit Procedure and Operation Recipe fields. Emerson EVT data source only.
/scan=<seconds>	(Optional) Specifies, in seconds, how often to scan the data source for new data. The default is 60 seconds. A scan that returns a large amount of data can cause the interface to skip the subsequent scan.
/singlerun	(Optional) Perform one scan and stop.
/smp="equipment path"	(Optional) Specifies an alternate PI Module path or PI AF element path for a particular equipment hierarchy. By default, the interface scans starting at the root level. Use the following syntax: `\\<RootModule>\<SubModule>\<…>`
/sqlconnto=<seconds> (DeltaV SQL only)	(Optional) Override the default SQL timeout setting (60 seconds).
/sqldato=<seconds> (DeltaV SQL only)	(Optional) Override the default SQL data access timeout setting (100 seconds).
/swaptime=<seconds>	(Optional) Specifies, in seconds, how long the current primary interface must be unavailable before failover occurs. Default: 300 seconds.
/tbid	(Optional) Use the truncated batch ID in the batch ID field of unit procedures. Incoming batch IDs are reformatted using the mask defined in the **/bidm** parameter.

Parameter	Description
`/tbse`	(Optional) Directs the interface to use top level recipe start/end events for creating batch objects. By default, the interface uses batch load/unload events. Intended for batches with S88 recipe types: Procedure, Unit Procedure, Operation, and Phase.
`/ts=GMT | LCL`	(Optional) Specifies how the interface interprets event timestamps from an SQL data source. Options are local time or GMT. Default is GMT.
`/uobev` (DeltaV SQL 9.3+ ONLY)	(Optional) Directs the interface to use the original batch event view. By default the interface queries 17 tables to retrieve data for batch-associated events. Note that this view does not provide explicit [Descript], [Pval] and [EU] fields. Instead the [Descript] field combines data from all three fields. This option is provided for backward compatibility.
`/ubr` (Emerson DeltaV only)	(Optional) Use DeltaV Batch Recipe View data instead of Recipe State Changes to detect start and end events. The Recipe State Change timestamps differs from Batch Recipe timestamps, as a result, inconsistencies in timestamps might occur.
`/uidlist=<list>` Optional (FTBInt, WPASX and Wonderware only)	(Optional) Recover specified manufacturing orders, then exit. Specify a comma-separated list of unique IDs of the manufacturing orders to be recovered. For WPASX, the IDs can be obtained from the source PASX.ManufacturingOrder.EntityKey column. This parameter overrides any settings specified for the /rst and /ret recovery switches. Example: **`/uidlist=5010350293,5011438395`**

Initialization file reference

The initialization (.ini) file stores configuration information about data sources, translations, product template, equipment template, tag templates and property templates. This file is generated and updated by the PI Event Frame Interface Manager configuration tool.

The information in this appendix is intended for troubleshooting purposes, to enable you to understand the contents of the .ini file.

> **Note:**
> Do not edit the .ini file manually.

The .ini file is divided into the following sections:

- SOURCE TEMPLATE: Specifies the settings required to connect to the data source, events to be skipped (filtered)
- GENERAL: Specifies a prefix to be prepended to the unit name when the interface creates a PI unit.
- TAG TEMPLATE: Contains templates that define how tags are to be created based on incoming data from the data source.
- PROPERTY TEMPLATE: Contains templates that define how PI properties are to be created based on incoming data from the data source.
- TRANSLATIONS: Maps text from the data source language to the target language for the PI System.

The following table describes optional settings that are stored in the .ini file.

Parameter	Description
excludestates=<list>	Specifies a comma-separated list of phase states to ignore. Not case-sensitive. You can use wildcards for matching. Examples: **excludestates=COMPLETED,AB*ING** **excludestates="COMPLETED, ABO*ING"** **excludestates= IDLING, COMPLE***
skipphases=<list>	Specifies a comma-separated list of phases to ignore. Not case-sensitive. You can use wildcards for matching. The interface checks the list against the [Phase] field (batch recipe) or [PhaseModule] field (equipment) Examples: **skipphases=phase_1, ph*2 skipphases = ph_test*, ph*ort***
skiprecipes=< list>	Specifies a comma-separated list of recipes to ignore. Not case-sensitive. You can use wildcards for matching. The interface checks the list against the corresponding event field, depending on recipe type, as follows: Examples: **skiprecipes=recipe_1, rec*2 skiprecipes = PRC_PAINT*, UP_TEST:2**

Parameter	Description
skipunits=<list>	Specifies a comma-separated list of units to ignore. Not case-sensitive. You can use wildcards for matching. The interface checks the list against the corresponding [Unit] field. Example: **skipunits = unit_1, u*2**

Diagnostic tag reference

Performance tags, which are automatically created for each instance of the interface, enable you to monitor the performance of the instance. There are 35 performance tags, categorized as follows:

- Health monitoring
- Object counters
- Timers

Health monitoring tags

There are two tags designed to monitor the health of the interface: the HeartBeat tag and the DeviceStatus tag. The HeartBeat tag is updated with the scan frequency configured for the interface or 60 seconds, whichever is less. The value of the HeartBeat tag is a cycle of integers from 1 to 15. The device status tag is automatically created and configured by the interface on startup. The device tag reflects status as follows:

- "Good": The interface is properly communicating and reading data from the data sources.
- "1 | Starting": The interface is executing its initialization routines.
- "2 | <details>": Indicates successful connection to the data source.
- "3 | <details>": Indicates failure to access the event journal file directory or failure to read data from the event journal file.

The properties of health monitoring tags are provided in the table below, where the *Prefix* represents *<Interface>_<ID>*. Note, if these tags do not exist. the interface creates them automatically during startup.

Tag name	Point Type	Loc1	Loc3	PointSource	ExcDesc
Prefix_DeviceStatus	Int32	Intf ID	0	Intf Pt Src	[UI_DEVSTAT]
Prefix_HeartBeat	Int32	Intf ID	1	Intf Pt Src	[UI_HEARTBEAT]

Object counter tags

The first time it is started, the interface creates 24 tags designed to monitor its performance. Archiving flag for these tags is turned off. The following table contains common point attributes for this group of tags:

Point type	Location1	Point source
Int32	<Interface ID>	<Interface Point Source>

The attributes for each performance counter tag are provided in the table below where the Prefix is defined as `_<Interfaceid>`. All counters are set to 0 when the interface starts up.

Tag Name	Loc3	ExcDesc	Description
<interfaceID>_EventReadCount	2	[UI_EVENTREADCOUNT]	Number of events read from the source since startup.
<interfaceID>_ErrorCount	3	[UI_ERRORCOUNT]	Number of errors since startup.
<interfaceID>_SourceUnitCount	4	[UI_SOURCEUNITCOUNT]	Number of units found on the data source(s) since startup.
<interfaceID>_PIUnitCount	5	[UI_PIUNITCOUNT]	Number of units added since startup.
<interfaceID>_SourcePhaseModCount	6	[UI_SOURCEPHASEMODCOUNT]	Number of phase modules found on the data source(s) since startup.
<interfaceID>_PIPhaseModCount	7	[UI_PIPHASEMODCOUNT]	Number of phase modules added since startup.
<interfaceID>_SourceBatchCount	8	[UI_SOURCEBATCHCOUNT]	Number of batches found on the data source(s) since startup.
<interfaceID>_PIBatchCount	9	[UI_PIBATCHCOUNT]	Number of batches added since startup.
<interfaceID>_SourceUnitBatchCount	10	[UI_SOURCEUNITBATCHCOUNT]	Number of unit batches found on the data sources(s) since startup.
<interfaceID>_PIUnitBatchCount	11	[UI_PIUNITBATCHCOUNT]	Number of unit batches added since startup.
<interfaceID>_SourceSubBatchCount	12	[UI_SOURCESUBBATCHCOUNT]	Total number of operations, phases, and phase states found on the data source since startup.
<interfaceID>_PISubBatchCount	13	[UI_PISUBBATCHCOUNT]	Total number of operations, phases, and phase states added since startup.
<interfaceID>_SourcePropertyNodeCount	14	[UI_SOURCEPROPNODECOUNT]	Number of property nodes found in data source(s) since startup
<interfaceID>_PIPropertyNodeCount	15	[UI_PIPROPNODECOUNT]	Number of PIProperty objects (nodes) added since startup.
<interfaceID>_SourcePropertyEventCount	16	[UI_SOURCEPROPEVENTCOUNT]	Number of events to be written to the batch properties found on the data source(s) since startup.
<interfaceID>_PIPropertyEventCount	17	[UI_PIPROPEVENTCOUNT]	Number of PIProperties (events) added since startup.
<interfaceID>_SourceTagCount	18	[UI_SOURCETAGCOUNT]	Number of tags found on the data source(s) since startup

Tag Name	Loc3	ExcDesc	Description
<interfaceID>_PITagCount	19	[UI_PITAGCOUNT]	Number of PI tags added since startup.
<interfaceID>_SourceTagEventCount	20	[UI_SOURCETAGEVENTCOUNT]	Number of events to be written into tags found on the data sources(s) since startup.
<interfaceID>_PITagEventCount	21	[UI_PITAGEVENTCOUNT]	Number of events written into PI points since startup.
<interfaceID>_SourceTagAliasCount	22	[UI_SOURCETAGALIASCOUNT]	Number of tag aliases to be created based on the data source(s) since startup.
<interfaceID>_PITagAliasCount	23	[UI_PITAGALIASCOUNT]	Number of PI aliases added since startup.
<interfaceID>_CachedBatchCount	24	[UI_CACHEDBATCHCOUNT]	Number of batch objects cached in the local memory.
<interfaceID>_OpenBatchCount	25	[UI_OPENBATCHCOUNT]	Subset of cached objects that have no end time set.
<interfaceID>_WaitingForEquipmentUB	34	[UI_UBWAITFOREQUIP]	Number of unit batches that do not have equipment allocated yet.

Timers

The last performance tag category is composed of timer tags, which are built automatically on first interface startup. Each timer tag reports on how much time per scan it took the interface to perform a particular task. There are three categories of timer tag: data source reading, local data caching and synchronizing cached data with the PI System.

Attributes of performance timer tags

Tag Name	Loc3	ExcDesc	Description
<interfaceID>_SourceReadTime	26	[UI_SOURCEREADTIME]	The time per scan it took the interface to read data from data source(s).
<interfaceID>_TagCacheTime	27	[UI_TAGCACHETIME]	The time per scan it took the interface to populate local tag cache.
<interfaceID>_BatchCacheTime	28	[UI_BATCHCACHETIME]	The time per scan it took the interface to populate the local batch cache.
<interfaceID>_EquipmentCacheTime	29	[UI_EQUIPCACHETIME]	The time per scan it took the interface to populate the local equipment (module) cache.

Tag Name	Loc3	ExcDesc	Description
`<interfaceID>_BatchSyncTime`	30	[UI_BATCHSYNCTIME]	The time per scan it took the interface to synchronize local batch cache with the PI System.
`<interfaceID>_TagSyncTime`	31	[UI_TAGSYNCTIME]	The time per scan it took the interface to synchronize local tag cache with the PI System.
`<interfaceID>_EquipmentSyncTime`	32	[UI_EQUIPSYNCTIME]	The time per scan it took the interface to synchronize local equipment cache with the PI System.
`<interfaceID>_TotalTime`	33	[UI_TOTALTIME]	The total time per scan it took to read data, cache it in local memory and synchronize the local cache with the PI System.

Event file monitor utility

For interfaces that can use event journal files for a data source, OSIsoft provides a utility that automatically copies new event journals from the directory where the BES creates them to the directory where the interface processes them.

The utility is installed in the same directory as the interface, along with a template Windows command file that you can copy and edit to launch the utility with the desired settings. The executable file name ends with "Sync" (for example, `EVTSync.exe`).

To configure the utility, copy and edit the batch file, specifying settings using command line parameters as follows:

Parameter	Description
`/dest=<path>`	Full path to destination directory, where the interface processes incoming event journals.
`/rate=#`	Optional rate in seconds to scan source and destination directory. Default scan rate is 30 seconds. This parameter must be an integer value.
`/src=<path>`	Full path to source directory, where the BES creates event journals.

After specifying settings, invoke the batch file to verify that the utility launches without errors.

To ensure that the utility restarts when the interface node is rebooted, configure a Windows automatic service. To install the service, invoke the executable, specifying the `-install` switch. For example: `EVTSync.exe –install`.

To verify that the service was added successfully, check the Microsoft Windows Services control panel.

Supported features

Platforms: (32-bit or 64-bit in emulation mode)

- Windows Vista
- Windows 2008 and Windows 2008 R2
- Windows 7
- Windows 8
- Windows Server 2012

No native 64-bit builds of the interfaces are available.

Feature	Support
Part Number	*See interface-specific chapter*
Auto-creates PI Points and equipment assets?	Yes
Point Builder Utility	No
ICU Control	No (use PI Event Frame Interface Manager configuration tool)
PI Point Data Types*	Integer/ Float32/ String
Sub-second Timestamps	Yes
Sub-second Scan Classes	No
Automatically Incorporates Changes to PI Point Attributes	No
Exception Reporting	No
Outputs from PI	No
Inputs to PI	Event and Scan-based
Supports Questionable Bit	No
Supports Multi-character PointSource	Yes
Maximum Point Count	No maximum
Uses PI SDK	Yes: version 1.3.4.333 or higher required
Uses AF SDK	Yes: version 2.5.x or higher required
PINet String Support	N/A
Source of Timestamps *	BES (not system time on interface node)
History Recovery	Yes
UniInt-based	No
Disconnected Startup *	No
SetDeviceStatus *	Yes
Failover	Yes
Vendor Software Required on PI Interface Node	No (except for the Siemens SIMATIC interface)
Vendor Hardware Required	No
Additional PI Software Included with Interface	Yes

Feature	Support
Device Point Types	The interface receives data from source as strings and coerces the data into numerical types according to tag templates, if defined.
Serial-Based Interface	No

History recovery

You can stop the interface without losing any data, because the data is persistent in the data source. Data recovery is limited by the history available from the BES, the number of licensed PI tags, and the size and time frame of the PI archives into which data is recovered.

Device status tag

This string tag contains information about communication between the interface and the data source. This tag is evaluated only while the heartbeat tag is updating and is updated on startup, change, and shutdown.

During normal operation, the tag contains the digital state set value "Good," indicating that the interface is communicating properly with the data source. Otherwise, the tag contains a string indicating status. The following table lists standard status strings.

Message	Description
`1 | Starting`	The interface is starting.
`2 | Connected / No Data`	The interface is connected to the data source but is not capable or reading or writing data to the foreign device.
`3 | n device(s) in error`	The interface is not able to communicate with the specified number of devices. Usually includes additional interface-specific details.
`4 | Intf Shutdown`	The interface is shutting down.
`5 | interface_specific_message`	Message specific to the interface.

Technical support and other resources

For technical assistance, contact OSIsoft Technical Support at +1 510-297-5828 or through the OSIsoft Tech Support website (https://techsupport.osisoft.com). The website offers additional contact options for customers outside of the United States.

When you contact OSIsoft Technical Support, be prepared to provide this information:

- Product name, version, and build numbers
- Details about your computer platform (CPU type, operating system, and version number)
- Time that the difficulty started
- Log files at that time
- Details of any environment changes prior to the start of the issue
- Summary of the issue, including any relevant log files during the time the issue occurred

The OSIsoft Virtual Campus (vCampus) website (http://vcampus.osisoft.com) has subscription-based resources to help you with the programming and integration of OSIsoft products.