AVEVA™

Analytics and Notifications for PI System Explorer (PI Server 2024)

2024

aveva.com

Contents

Welcome to Asset Analytics and Notifications for PI System Explorer

The AVEVA™ PI System™ offers many ways to transform simple process data into decision-making information. Its analytics or calculation tools allow you to produce values from simple to complex calculations. You can use the results of these calculations to create new data streams, make business decisions such as inputs to other calculations, and trigger notifications.

Asset analytics are based on Asset Framework (PI AF) and enable users to perform the following tasks:

- Create expressions based on performance equation syntax
- Roll up attributes for aggregation
- Generate event frames based on user-defined triggering conditions
- Perform SQC analyses based on the Western Electric SQC Handbook

PI AF also allows you to configure notifications and send alerts about events of interest. Notification messages can be customized to include information about an event, and sent via email to individuals, groups, or a web service. These options are available if your installation of PI System Explorer includes the Management plug-in.

Note: If PI Analysis Service and PI Notification Service are configured to use claims, users who sign on to PI System Explorer can use OpenID Connect (OIDC) to access other claims-enabled PI System resources. If OIDC is not enabled, Windows authentication is used.

Asset analytics

Asset analytics allows you to create and manage analyses in Asset Framework (AF). An analysis reads values of PI AF attributes, performs calculations, and writes results to other attributes. Within PI System Explorer, you can access asset analytics when you work with elements and element templates.

Introduction to analyses

An analysis performs calculations on a specified schedule. An analysis takes existing PI AF attribute values as inputs and produces new outputs, either new calculated values or new event frames. Every analysis has an associated element or element template.

The following types of analyses are available:

- **Expression**

  Calculates one or more output values from specified functions, operators, and input values.
- **Rollup**

  Calculates standard statistical functions for a group of selected attributes. This group is selected from attributes on an element or from the set of all attributes on its sub-elements.

- **Event frame generation**

  Starts or ends event frames based on specified conditions.
- **SQC**

  Uses Statistical Quality Control (SQC) methods to monitor that attribute values lie within predetermined boundaries.

All analyses can be created from analysis templates with either Enabled or Disabled status; by default, analyses will be enabled when created. To create an analysis with Disabled status, make sure to clear the **Enable analyses when created from template** check box in the analysis template.

For event frame generation analyses and SQC analyses with event frame outputs, you can create a notification rule template by clicking **Create a new notification rule template for Analysis Template Name**.

For expression analyses and event frame generation analyses, you specify inputs in expressions.

For event frame generation analyses (running PI Analysis Service 2017 R2 or later versions), expression and rollup analyses, you can map analysis outputs to PI AF attributes. If you map outputs to attributes configured as PI point data references, the analysis saves the history of the output to a PI point. We recommend you save analysis outputs to PI points. You can use the PI point in visualization tools to view the trend of the analysis. Saving outputs to PI points also results in better PI AF performance than saving outputs to attributes without data references.

Note: PI Analysis Service does not use exception reporting while writing to output configured as PI point data references. Although exception settings for the output PI point are not used, compression settings for the output PI point will be used to determine the outputs that are written to Data Archive.

Beginning with PI AF 2018 SP2, analyses will honor the display digits configured on an attribute or attribute template. Values you see when you evaluate an analysis or preview results will display the same number of digits as your output attribute so long as you have mapped your output attribute. For more information, see Control the display of attribute and attribute template values.

Configure and manage analyses on an element

A newly-created analysis is listed in the table in **Analyses** tab. You can see its current status, whether or not it is associated with a template, its analysis type, name, and backfilling/recalculation status. Hover over items or icons in the list to see descriptive tooltips. To manage multiple analyses in the **Management** pane, see Management of analyses.

To view the expression and scheduling for an analysis, select it from the **Analyses** viewer.

You can perform these operations from the **Analyses** viewer:

- **Create a new analysis**

  You can click (New Analysis) at the top of the viewer, or right-click anywhere in the viewer and click **New**.
- **Enable or disable a selected analysis**

  You can select an analysis and click ▶ (Enable Analysis) or ■ (Disable Analysis) at the top of the viewer.
- **Choose columns to display**

  You can right-click any column heading and select headings to display.
- **Preview results**

  You can preview results for an analysis to see how it operates over a specified time range by right-clicking an analysis and selecting **Preview Results**. You can export the results table to a spreadsheet or you can copy selected rows from the results table into other applications.

For information on viewing trend charts for output times that are specified relative to trigger times, see Trend charts in Preview Results.

- **Backfill or recalculate data for an analysis**

  You can execute an analysis over an earlier time period to backfill or recalculate data, as described in Backfill or recalculate data for an analysis.

  An expression, rollup, or SQC analysis can backfill or recalculate data if the analyses outputs are mapped to PI point attributes. You can backfill the missing data in a time range and retain existing data, or you can recalculate, which deletes and recalculates data in the time range.

  An event frame generation analysis can also recalculate event frames over a specified time period. The recalculation process, however, automatically deletes all existing event frames for that time period, as well as annotations on affected event frames.

  In order to backfill/recalculate, you need Execute permission on the analyses. Proper permission can be obtained by mapping your account to Asset Analytics Recalculation identity. Note that recalculation is only available for PI Analysis Service 2016 R2 or later. In addition, Data Archive that stores PI points must be running version 2016 R2 or later.

- **Enable automatic recalculation for an analysis**

  From PI AF 2017 R2, you can automatically recalculate analyses. Although automatic recalculation is globally enabled by default, you need to opt in at individual analysis (template) level if you expect late-arriving or out-of-order data that may trigger recalculation.

  An analysis listed with means automatic recalculation is enabled. See Configure automatic recalculation of analyses and analyses templates for more information.

- **Go to template**

  An analysis listed with alongside is based on an analysis template. You can open the template directly from the analysis by right-clicking the analysis and selecting **Go to Template**.

  Note: You can go to the analysis directly from the attribute by right-clicking the attribute and clicking **Go to Analysis**. If there is no analysis configured on the attribute, you will see **New Analysis** instead.

- **Audit Trail Events**

  You can open the audit log directly from the analysis by right-clicking the analysis and selecting **Audit Trail Events**. For more information on audit trail, see Track PI AF changes with Audit Trail.

- **Reset to template**

  Only scheduling changes can be made to an analysis that is based on an analysis template. To revert an analysis' scheduling to template scheduling, right-click the analysis and click **Reset to Template**.

- **Convert to template**

  If the element for an analysis is derived from an element template, you can convert the analysis to an analysis template on that element template by right-clicking the analysis and selecting **Convert to Template**. This feature enables you to develop and debug an analysis against a specific element before generalizing it as a template.

  Note: If any of the analysis outputs write to a specific PI point, you will be prompted to choose how to specify the PI point in the analysis template to ensure the outputs of derived analyses write to unique PI points.

To create specific types of analyses, see:

- Create an expression analysis
- Create a rollup analysis
- Create an event frame generation analysis
- Create an SQC analysis

Analysis templates

To apply an analysis to an element, you specify the element directly as you create the analysis. However, to apply an analysis to a group of elements (created from the same element template), it is much more efficient to use an *analysis template* than to apply the analysis individually to each element.

An analysis template defines the form of an analysis. Analyses derived from it are all similar but have specific input and output attributes. Analysis templates let you take advantage of your PI AF hierarchy. Every element derived from an element template automatically acquires analyses from its analysis templates.

You create an analysis template on an element template in much the same way you create an analysis on an element. When you add or change an analysis template on an element template, those changes propagate to every element derived from the element template. Deleting an analysis template deletes all analyses derived from it, except for analyses tied to notifications.

With PI Server 2015 R2 and later versions, while creating an analysis template on an element template that is derived from a base element template, you can choose input attributes from attribute templates that are defined on derived as well as base element templates. Similarly, you can map analysis outputs to derived as well as base template attributes.

Note: An analysis derived from an analysis template cannot be removed from an element directly.

To create specific types of analysis templates, see:

- Create an expression analysis template
- Create a rollup analysis template
- Create an event frame generation analysis template
- Create an SQC analysis template

Updates of analyses and analysis templates

As soon as a new analysis is checked in, it is automatically enabled and available to run. Any change that is checked in is immediately picked up by all the analyses affected by it, even if they are running. Changes that can affect an analysis include:

- Edits made directly to the analysis
- Changes to an element that impact its analyses, including adding or deleting an analysis
- Changes to an element template, which propagate to all elements that come from the template, including adding or deleting an analysis template
- Changes to an analysis template, which propagate to all analyses that come from the template
- Changes to an event frame template used by an analysis; new event frames are based on the updated template

Excluded attributes in analyses

Beginning with PI AF 2018, behavior and handling of excluded attributes have changed in expression, event frame generation and SQC analyses. For more information on excluded attributes, see Excluded attribute property.

Prior to 2018, analyses configured with excluded attributes would return **Calc Failed** when evaluated and go into error. Beginning with PI AF 2018, such analyses will be suspended and marked by an icon (⏸) in the Analyses Viewer. If you hover over the icon, you will see a tooltip with more information.

If you do not want the analyses with excluded attributes to go into a suspended state, you can validate the excluded attribute with the BadVal expression function.

To prevent analyses with excluded attributes from being suspended, use the **BadVal** expression function to directly validate the excluded attribute. For example, the following expression syntax validates the att1 attribute, allowing the analysis to continue to run:

```
If BadVal('att1') Then ('att2') Else ('att1' + 'att2')
```

In the example above, validation is only successful if the exact syntax is used; there cannot be a nested expression within **BadVal**.

Note: Any change in the configuration of the excluded attribute in the analysis is automatically picked up by PI Analysis Service. For example, if you decide to change the property of 'att1' to be no longer excluded and check it in, the service will detect it and run the analysis with 'att1.'

When you select **Event-Triggered** as a scheduling option, you are required to select a triggering input. Note that if the excluded attribute is the only available triggering input, the analysis will be suspended whether or not you validate it with **BadVal**. It is because analyses do not trigger on excluded attributes. Make sure that you have other inputs to trigger on in your analyses. For more information, see Analysis scheduling.

Analysis scheduling

Each analysis requires you to specify scheduling. Scheduling indicates when to evaluate an analysis automatically. There are two types of scheduling:

- **Periodic**

  With periodic scheduling, evaluation occurs based on clock time (DateTime object expressed as Coordinated Universal Time (UTC)). You can specify a specific time for evaluation each day or the time between evaluations. With PI Server 2015 R2 and later versions, the maximum frequency you can set for analyses is 120 times per second.

  Note: If you have analyses that are previously configured to run more than 120 times per second, you must reduce the period to less than the allowed maximum, else they will generate errors while running with PI Server 2015 R2. Similarly, if you are setting periodic scheduling outside of the PI System Explorer user interface, make sure that the period is set to less than the allowed maximum.

  If the period is not divisible by the 24, 7 minutes for example, the evaluation time will reset at the beginning of each day. Let's say you have scheduled the analysis to run every 7 minutes and the last evaluation was at 11:56pm. The next evaluation will be at midnight (4 minutes later), not at 12:03am the next day.

- **Event-Triggered**

  With event-triggered scheduling, evaluation occurs based on events. You can specify one or more input attributes that trigger an evaluation whenever the attribute value changes.

Note: When auto-backfilling is enabled, triggering events with time stamps before analysis service start-time are ignored for the purpose of real-time evaluation. For more discussion on real-time evaluation and auto-backfilling, including the setting of the *AutoBackfillingEnabled* configuration parameter, see Analysis service configuration.

By default, analyses use event-triggered scheduling, triggered on changes to any input.

Many factors affect the speed that an analysis runs and writes output values:

- System architecture, including inputs and outputs
- Network configuration
- Load and performance of different computers

Monitor and test your analyses, especially those scheduled at high frequencies, to ensure the system resources support the configuration.

Output time stamps for analyses

For any scheduled analysis, the default time stamp for output values is the trigger time. For periodic scheduling, the trigger time is the scheduled evaluation time, and for event-triggered scheduling, the trigger time is the time that a specified attribute changes values.

You can specify the time stamp of analysis output values. By specifying an output time stamp, you can:

- Produce results that are time-stamped at a specified offset from the trigger time.
- Produce results that are time-stamped at a particular time each day, regardless of when the analysis is actually evaluated. For example, you can ensure a 6:00 p.m. time stamp for the output of an analysis that calculates results for a daily production shift ending at 6:00 p.m., even if the analysis is not actually evaluated until 10:00 p.m.
- Repeatedly evaluate an analysis to calculate a forecast value for a particular date and time, as input conditions change. The results of each evaluation have the same timestamp: the analysis output values reflect results from the most recent evaluation.

Specify time stamp for analysis outputs

Use the Advanced options window in an analysis to specify the time stamp of analysis output values.

1. In the Elements browser, select an element, and in the viewer, click the **Analyses** tab.
2. From the list of analyses, select an analysis, and then click **Advanced** in the scheduling area near the bottom of the window.
3. In the Advanced options window, specify the output time stamp:
    - **Trigger Time**

      Default value. The clock time that a schedule specifies or that an input value changes.
    - **Execution Time**

      The clock time at which the analysis calculates the value. This differs from the trigger time when the system has a lag time or when there is a configured calculation wait time (see *CalculationWaitTimeInSeconds* in Analysis service configuration).

- **Relative to Trigger Time**

  A time specified by a PI time expression. Enter a valid time expression, such as a time relative to the trigger time or a fixed time. In the expression, the current time and implied reference time is the trigger time (that is, the clock time that the schedule specifies or that an input value changes). If you enter a fixed time, all output values from the analysis will have the entered time stamp. When not specified, the default unit of measure for relative time is hours.

  For information on viewing trend charts for output times that are specified relative to trigger times, see Trend charts in Preview Results.

- **Variable**

  Use the value of a variable as an output timestamp. **Variable** must be a timestamp of value type DateTime or, a String that can be parsed as *DateTime*. If this value is calculated from an expression, PI Analysis Service uses the calculated value for each trigger events as the output timestamp. This option is only available for expression analyses.

  You can preview your results by right-clicking the analysis and selecting **Preview Results**. For more information, see Trend charts in Preview Results.

Trend charts in Preview Results

When you right-click an expression or rollup analysis and select Preview Results, you see the trend chart with input and output attribute values at different trigger times.

If you select **Relative to Trigger Time** and specify '*-8h' as output time stamp for example, you see two trend charts displayed, one for input and the other for output. The input chart shows the same time range as the configured start and end time; however, the output chart shows the time that is calculated using the output time stamp override to the trigger time. Each trend chart shows a maximum of four input and four output attributes; currently, the input or output attributes that are displayed on the chart are automatically selected by PI Analysis Service.

In the table, the column **Output Time** indicates the output time calculated with the time stamp override.

Note: The Preview Results charts may depict data that differs from the actual results of the PI Analysis Service. For example, if the same timestamp data is overwritten several times, the Preview Results table and charts show each written value of the data point, but the PI Analysis Service only uses the last written value.

Expressions for analyses

For expression analyses and event frame generation analyses, you specify inputs and calculations in expressions. In PI System Explorer, you enter each expression in a row. Each row has the following columns:

- **Name**

  A variable name for the value computed by the row. This variable is internal to the analysis. You can use this variable as an input to expressions specified by subsequent rows in the analysis.

- **Expression**

  The specification of a calculation performed. The expression can include PI AF attributes, variables from earlier rows in the analysis, and functions.

- **Value at Evaluation**

  The current computed value of the expression. Click **Evaluate** to compute again.

- **Value at Last Trigger**

  The value when the last trigger was computed.
- **Output Attribute**

  For expression analyses, the attribute that stores the computed value. If the analysis contains multiple expressions, you can map the value from any expression to an output attribute. However, at least one expression must be mapped to an output attribute.

For example, an expression analysis might contain one expression, specified in a single row.

Name	Expression	Value at Evaluation	Output Attribute
`Variable1`	`PrevVal('Att1') - 100`	*505*	`Analysis1_Output`

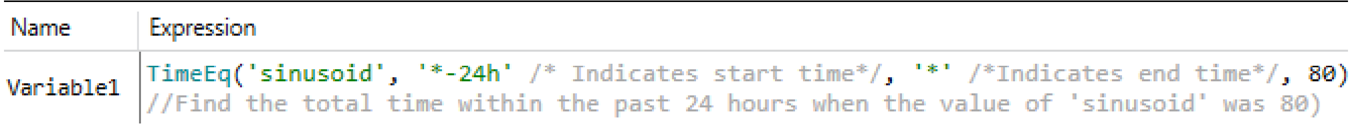
You can also specify the calculation of an analysis with multiple expressions across several rows. For example, an analysis might contain three expressions: the first two calculate variables V1 and V2, which are used as inputs to the last expression.

Name	Expression
`V1`	`'Att1'/2`
`V2`	`'Att2' - 'Att3'`
`Result`	`V1 + 3*V2`

Analyses with multiple expressions evaluate the expressions in order: the top row first, the one below it next, and so on. Because expressions in lower rows may depend on results from higher rows, you can reorder the rows to evaluate them in the proper order.

You can click **Add a new variable** to add a row and the ⊗ icon to delete it.

Tip: You can add a line break within the expression by clicking shift+enter. You can also make your expressions easier to read by adding comments next to or within an expression. Use double slash (//) to introduce a comment or place the comment between slash asterisk (/*) and asterisk slash (*/): `// Add comments here IF x THEN y ELSE /*Do this if x is false*/ z`



```
Name       Expression
Variable1  TimeEq('sinusoid', '*-24h' /* Indicates start time*/, '*' /*Indicates end time*/, 80)
           //Find the total time within the past 24 hours when the value of 'sinusoid' was 80)
```

Expression specification

To create an expression, click in the **Expression** column of a row and then specify the input attributes, variables, and functions that define a calculation. You can do the following:

- Select and insert attributes and functions into the expression from the **Functions** and **Attributes** panels to the right of the expression.
- Enter the text directly into the expression. As you type, functions, attributes, and variables that match what you entered appear in a selection list at the cursor: you can select an item you want to insert.
- Attributes and time expressions are enclosed in single quotes.

- Strings are enclosed in double quotes.

Expression simplification with refactoring

In real-world applications, the expression in an analysis can be extremely complex; lengthy attribute names can add to the complexity. To simplify a complicated expression, refactor it into a group of smaller expressions assigned to variables.

Refactored analyses are easier to understand. Refactored analyses are also easier to test and debug: you can evaluate the smaller expressions one at a time and isolate any errors.

To refactor an analysis, define components of the analysis as separate expressions and reference by variable name. For example, you might define a separate expression that defines a long attribute name as a separate variable or that defines a particular calculation as a separate variable. To create these separate expressions, simply add new expression rows to the analysis and enter text in the **Name** and **Expression** columns.

To reduce typing, you can select a term in an expression, right-click the selection, and then click **Assign to variable**. In the row above the expression, PI System Explorer inserts a new row that maps the selected term to a new variable name. In the original expression, PI System Explorer replaces the term with the new variable name.

You can edit the name of any row. However, if you change the name of an expression row already used as a variable in other expressions, you break the connection between the expressions.

Example

Suppose you have a complex expression defined in a single row:

Name	**Expression**
`Variable1`	`2*'LongAttributeName' + Avg('Att2', 'Att3', 'Att4')`

You can refactor the expression into the following three rows that contain simpler expressions. Note that the third row uses the names of the earlier rows as variables.

Name	**Expression**
`Variable1`	`'LongAttributeName'`
`Variable2`	`Avg('Att2', 'Att3', 'Att4')`
`Variable3`	`2*Variable1 + Variable2`

Future data and analyses

You can use future data as input to an analysis. You can also use an analysis to produce future data by specifying a future time stamp for the output from an analysis.

Expressions can use values from future events as input to an analysis. However, if an analysis uses event-triggered scheduling, inputs with future data can generate analysis evaluations. If a trigger attribute has future events, an evaluation occurs as each event becomes current, when the clock time coincides with the time of that event.

The following sections show two uses of future data in analyses. The first example shows how to use future data as an input to an analysis. The second example shows how to use an analysis to produce future data.

Example: Input future data to evaluate forecast values

Suppose you need to evaluate forecast values for the outside temperature provided by a third-party service. To do that, create an analysis that finds the difference between forecast and actual values.

The analysis needs two input attributes:

- **ActualTemp**

  Stores near real-time temperature readings from an outside thermometer.
- **ForecastTemp**

  Stores future data, in this case forecast values for the outside temperature each day at 6 a.m., 2 p.m., and 10 p.m.

The analysis calculates the difference between the *ActualTemp* and *ForecastTemp* values and writes results to an output attribute named DeltaTemp.

Choose event-triggered scheduling for the analysis with the *ForecastTemp* attribute as the only triggering input attribute. With event-triggered scheduling, an evaluation occurs when the analysis detects a new event for a trigger attribute. Because this trigger attribute stores future data, the analysis detects a new event three times a day, when the clock time reaches 6 a.m., 2 p.m., and 10 p.m.; this triggers an evaluation and a new result for the *DeltaTemp* attribute.

Example: Output future data to calculate weekly emissions goals

Suppose you need to set weekly goals for emissions at a cement plant that operates under a voluntary annual CO_2 cap. The weekly goal will likely change every week in response to the actual emission levels of the past week and to the amount remaining under the voluntary annual cap.

At the end of every week, you know actual emissions since the beginning of the year and can calculate a new weekly emissions goal. You want the time stamp for the new goal to be exactly one week from the calculation time, which coincides with the calculation time of the actual emissions for the next week.

In PI AF, create attributes for the parameters the calculations require, such as the annual cap.

Next, create a new expression analysis that calculates an output attribute named *WeeklyGoal*. The analysis completes the following steps:

1. Find the actual emissions since the beginning of the year.
2. Find the difference between the year-to-date total and the annual cap.
3. Divide the difference by the number of days remaining under the annual cap and multiply by seven to calculate the *WeeklyGoal* attribute.
4. Save the *WeeklyGoal* attribute as a future PI point, which preserves history and allows for trending and other uses.

When creating the analysis, choose periodic scheduling for every day at 12:00 a.m. The syntax of the analysis ensures the calculations only run on Sunday.

Finally, specify the time stamp for the analysis outputs. Enter the expression ** + 7d* to set the time stamp to exactly seven days from the trigger time. The *WeeklyGoal* attribute will have a time stamp that is one week in the future (that is, the following Sunday at 12:00 a.m.).

Types of analyses

There are four categories of analyses you can create in Asset Analytics:

- Expression
- Rollup
- Event frame generation
- Statistical Quality Control (SQC)

Expression analyses

Expression analyses calculate one or more output values from specified functions, operators, and input values. You specify the inputs and calculations for these analyses in expressions. You can map any output that an expression calculates to an attribute. You must map at least one output to an attribute.

You can map the output to an existing attribute or you can create a new attribute when you map the output. If you create a new attribute when you map the output (from the **Analyses** tab in PI System Explorer), you must choose whether or not to save the output history; the system creates the appropriate type of attribute:

- If you choose to save output history, the system creates an attribute with a PI point data reference and creates a PI point in the specified Data Archive. With the output saved in a PI point, you can use visualization tools to see the trend associated with the calculation, and you can retrieve the value for any time that the analysis performed the calculation. The analysis calculates and stores a value as specified by the analysis schedule.

- If you choose not to save output history, PI System Explorer creates an attribute with an analysis data reference. An analysis only calculates the value for expressions mapped to attributes with an analysis data reference when requested. For best performance, map output attributes to attributes with analysis data references if you only need to know the most recent value and do not need previous values.

Create an expression analysis template

In PI System Explorer, you can create an expression analysis template for an element template. In the template, you can create an expression for each calculation. To record the value that an expression calculates, map the output to an attribute template. You can create new attribute templates from the expression-analysis template.

1. In the navigator, click **Library**.
2. In the Library browser, select the element template where you want to create the analysis template.
3. In the viewer, click the **Analysis Templates** tab.

   The tab lists any analysis templates already defined for the element template.
4. Create a new analysis template for the element template:
   - If no analysis templates exist, click **Create a new analysis template** to create the first one.

- Otherwise, click **New Analysis Template** to create a new one.

5. Enter information to identify the analysis:
   - **Name**

     The name that uniquely identifies this analysis.
   - **Description**

     Optional text that describes the analysis.
   - **Categories**

     Optional category that you can assign to the analysis. Click the list and select the check box next to categories you want to assign or click **New** to create a new category.
   - **Analysis Type**

     Click **Expression**.
6. Optional. Clear the **Enable analyses when created from template** check box so that the analysis is created in *Disabled* status.

   By default, a new analysis will be created with *Enabled* status. You must clear the option if you do not want to start the analysis when created.
7. Optional. To temporarily evaluate expressions in the analysis template based on values of a particular element, click the **Select an example element** link and select an element based on the current element template.

   If no elements have been created from the current element template, you must create an element first.

   Templates have no concrete data associated with their attributes. You cannot evaluate an expression in an analysis template, unless you borrow attribute values from a specific element.

   PI System Explorer sets **Example Element** to the element you selected.
8. Specify one or more expressions for the analysis.

   Enter each expression in a row. An expression specifies inputs and calculations. See Expressions for analyses.
9. Map at least one expression to an output attribute template.
   a. In the **Output Attribute** column for the expression, click **Map**.
   b. Specify the attribute template.

To ...	**Do this ...**
Map the output to an existing attribute template	Click the name of the attribute template. Select an attribute template with a PI point data reference to store the history of the calculation. For best performance when you do not require history, use an attribute template with an analysis data reference.
Map the output to a new attribute template	a. Click **New Attribute Template** to open the Attribute Template Properties window. b. Indicate whether the attribute should save the history of the output: a. Click **Yes** to create an attribute with a PI point data reference. This attribute saves

To ...	Do this ...
	the history of the output. b. Click **No** to create an attribute with an analysis data reference. This attribute calculates a new value when requested. c. Configure the attribute template to create: a. **Name** Name of created attribute. b. **Description** Optional text that describes the attribute. c. **Data Server** For attributes that save output history, the Data Archive server that stores the PI point data reference. By default, this is set to *%Server%,* which sets to the default Data Archive server. d. **Value Type** The data type the attribute stores. After creating the attribute template, you can refine its definition from the **Attribute Templates** tab. d. Click **OK** to create the attribute template.

10. Optional. If you specified an example element, click Evaluate to verify that output values of individual expressions.

    Specify scheduling for the analysis. See Analysis scheduling.
11. Optional. If you specified an example element, verify the analysis by reviewing the results produced with historical data:
    a. In the analysis list, right-click the analysis and then click **Preview Results**.
    b. Enter a start time and end time not later than the current time, and click **Generate Results** to see a list of results.
12. To apply changes and save your work locally, click ✓ on the toolbar.
13. Click **Check In** on the toolbar to save the analysis template to the PI AF server and to create the analysis for any element based on the element template.

Create an expression analysis

In PI System Explorer, you can create a unique expression analysis for an individual element. In the analysis, you can create an expression for each calculation. To record the value that an expression calculates, map the output to an attribute. You can create new attributes as you configure an analysis.

1. In the navigator, click **Elements**.

2. In the Elements browser, select the element where you want to create the analysis.
3. In the viewer, click the **Analyses** tab.

   The tab lists any analyses already defined for the element.
4. Create a new analysis for the element:
   - If no analyses exist, click **Create** a new analysis to create the first one.
   - Otherwise, click **New Analysis** to create a new one.
5. Enter information to identify the analysis:
   - **Name**

     The name that uniquely identifies this analysis.
   - **Description**

     Optional text that describes the analysis.
   - **Categories**

     Optional category that you can assign to the analysis. Click the list and select the check box next to categories you want to assign or click **New** to create a new category.
   - **Analysis Type**

     Click **Expression**.
6. Specify one or more expressions for the analysis.

   Enter each expression in a row. An expression specifies inputs and calculations. See Expressions for analyses.
7. Map at least one expression to an output attribute:

   a. In the **Output Attribute** column for the expression, click **Map**.

   b. Specify the attribute.

To ...	Do this ...
Map the output to an existing attribute	Click the name of the attribute. Select an attribute with a PI point data reference to store the history of the calculation. For best performance when you do not require history, use an attribute with an analysis data reference.
Map the output to a new attribute	a. Click **New Attribute** to open the Attribute Properties window. b. Indicate whether the attribute should save the history of the output: a. Click **Yes** to create an attribute with a PI point data reference. This attribute saves the history of the output. b. Click **No** to create an attribute with an analysis data reference. This attribute calculates a new value when requested. c. Configure the attribute: a. **Name**

	Name of created attribute. b. **Description** Optional text that describes the attribute. c. **Data Server** For attributes that save output history, the Data Archive server that stores the PI point data reference. By default, this is set to *%Server%*, which sets to the default Data Archive server. d. **Value Type** The data type the attribute stores. After creating the attribute, you can refine its definition from the **Attributes** tab. d. Click **OK** to create the attribute.

8. Optional. Click Evaluate to verify the output values at both Value at Last Trigger and Value at Evaluation times.

   Specify scheduling for the analysis. See Analysis scheduling.
9. Optional. To verify the analysis, review the results produced with historical data:
   a. In the analysis list, right-click the analysis and then click **Preview Results**.
   b. Enter a start time and end time not later than the current time, and click **Generate Results** to see a list of results.
10. To apply changes and save your work locally, click ✓ on the toolbar.

    This does not run the analysis.
11. Click **Check In** on the toolbar.

PI System Explorer saves and checks in your analysis, and then starts running the analysis.

Rollup analyses

A rollup analysis calculates statistical values such as sum and average for selected attributes associated with an element. For example, a rollup analysis on a factory element might use temperature attribute values for all pumps in the factory to calculate their average temperature.

You can choose either of two sources for attributes to include in a rollup analysis:

- **Attributes of the element**

  For example, you want to verify that the level of a tank is constant by checking that all inflows and outflow sum to zero. From the list of the tank element's attributes, select its inflow and outflow attributes for a summation calculation. A rollup using an element's own attributes is also known as an aggregation.
- **Attributes of the element's child elements**

  For example, you want to calculate average energy consumption for a group of pumps in a refinery. To do that, you create a roll-up analysis on the parent element (the refinery) that averages energy consumption attributes from its child elements (pumps).

Use selection criteria to select attributes for rollup. You can select attributes that match attribute name text you enter or that are associated with a category or element template you specify.

With PI Server 2015 R2 and later versions, you can select child attributes for rollup analyses. For more information, see Create a rollup analysis.

You must specify one or more functions to calculate statistics on the selected attributes in the rollup (see Rollup analysis functions). Be sure each output is mapped to an output attribute.

Be aware that:

- You should select attributes carefully to obtain sensible results. Avoid selecting a set with mixed units of measure (UOM; such as temperature and mass) or value types (such as numeric values and time stamps). As long as the inputs and outputs have different units within the same UOM class, the system makes reasonable effort to convert the input units to that of the output for calculation.
- You can add or delete elements or attributes in your hierarchy without the need to update roll-up analyses. Because a rollup identifies input attributes each time it is executed, it automatically includes any new attributes that meet its selection criteria.
- Some selected attributes (such as those with string values) will not participate in rollup calculations, which operate only on numeric or DateTime values.

Rollup analysis functions

You can specify one or more of the functions listed below to calculate statistics on the input attributes in a rollup analysis.

- Bad input attributes are omitted from rollup analysis. For example, if you have 5 attributes that match the criteria but 1 is a bad value, *Count* function will return 4.
- Input value types numeric, *DateTime* and *Enumeration* value are supported for functions *Minimum*, *Maximum*, and *Count*. Function *Average* takes numeric and *DateTime* value types. Input values for all other functions must be of numeric value type.

Function	**Description**	**Supported data types**
Sum	The sum of values from selected attributes.	Numeric
Average	The average (mean) of values from two or more selected attributes.	Numeric, *DateTime*
Minimum	The minimum value from selected attributes.	Numeric, *DateTime*, *Enumeration* value
Maximum	The maximum value from selected attributes.	Numeric, *DateTime*, *Enumeration* value
Count	The number of attributes actually used in rollup calculations, which can differ from the total number of selected attributes. For	Numeric, *DateTime*, *Enumeration* value

Function	Description	Supported data types
	example, an attribute containing a string value (such as a part name) might meet the selection criteria but would not participate in rollup calculations and would be excluded for *Count*.	
Median	The middle value of three or more values from selected attributes; for an even-numbered group of values, the average of the two middle values. An equal number of values fall above and below the median.	Numeric
Population standard deviation	The population standard deviation, which is calculated using the entire population. Useful when all values in a population are available to include as inputs for the calculation.	Numeric
Sample standard deviation	The sample standard deviation, which is calculated using values from a sample of a population. Useful to provide an estimate of the population standard deviation when it is impractical or impossible to include all population values in the calculation.	Numeric

Create a rollup analysis template

In PI System Explorer, you can create a rollup analysis template for a selected element template. You select input values for the rollup either from attributes of the selected element template or from the attributes of a child element template (whose parent element is derived from the element template). You can choose an example child element and use its attributes to help you develop a rollup analysis template.

Prerequisite: Identify the statistical functions you want the rollup to calculate and where you want to save the results. You can map results from a function to an existing attribute of the element template, or you can create a new output attribute when you configure an analysis template.

1. In the Library browser, select the element template where you want to create the analysis template.
2. Under **Rollup attributes from**, indicate the source of input values:
   - Click **Child elements of Element Template Name** to roll up attributes of child elements of an element

based on the current element template.

- Click **This element - Element Template Name** to roll up attributes of an element based on the current element template.

3. In the viewer, click the **Analysis Templates** tab.

   The tab lists any analysis templates already defined for the element template.

4. Create a new analysis template for the element template:
   - If no analysis templates exist, click **Create** a new analysis template to create the first one.
   - Otherwise, click **New Analysis Template** to create a new one.

5. Enter information to identify the analysis:
   - **Name**

     The name that uniquely identifies this analysis.
   - **Description**

     Optional text that describes the analysis.
   - **Categories**

     Optional category that you can assign to the analysis. Click the list and select the check box next to categories you want to assign or click **New** to create a new category.
   - **Analysis Type**

     Click **Rollup**.

6. Optional. Clear the **Enable analyses when created from template** check box so that the analysis is created in *Disabled* status.

   By default, a new analysis will be created with *Enabled* status. You must clear the option if you do not want to start the analysis when created.

7. Under **Rollup attributes from**, indicate the source of input values:
   - Click **Child elements of Element Template Name** to roll up attributes of child elements of an element based on the current element template.
   - Click **This element - Element Template Name** to roll up attributes of an element based on the current element template.

8. Select an example element to provide candidate attributes:
   - Click the **Select an example element** link to open a window that shows elements derived from the selected element template.
   - Select an element and click **OK**.

9. Select the attributes to include in the rollup:

If rollup attributes come from:	**Then:**
Child elements	a. From the **Sample Child Element** list, select the element from which you want to view attributes. The table shows attributes of the selected element. To improve performance, the table only shows a limited number of attributes. You can click the **Show more attributes** link below

If rollup attributes come from:	Then:
	the table to show additional attributes from the selected element. b. To the left of the table, specify criteria that select attributes to roll up: a. **Attribute Name** Type an entire attribute name or part of a name with wildcard characters. For example, type `temp*` to select all attributes that begin with "temp", such as Temperature and Temp1. **Note:** The selection of attributes depends on whether *Attribute Level* is set to "Root Level" or "Child Level". b. **Attribute Level** Select **Root Level** to choose from top-level attributes or **Child Level** for child attributes. You cannot choose both child attributes and top-level attributes in a single roll-up analysis. c. **Attribute Category** Select a category to limit selected attributes to those in that category. d. **Element Category** Select an element category to limit selected attributes to attributes that have parent elements in that category. e. **Element Template** Select an element template to limit selected attributes to attributes that have parent elements based on that template. **Note:** The criteria apply to all attributes of any child element, not just those shown in the table. You can specify criteria that select attributes not shown in the table. For example, if you set **Sample Child Element** to an element from category CatX but set **Element Category** to CatY, you will select none of the attributes in the table but the analysis will still include the attributes in the rollup.
This element	To the left of the table, specify criteria that select attributes to roll up:

If rollup attributes come from:	Then:
	• **Attribute Name** **Type an entire** attribute name or part of a name with wildcard characters. For example, type `temp*` to select all attributes that begin with "temp", such as Temperature and Temp1. **Note:** The selection of attributes depends on whether *Attribute Level* is set to "Root Level" or "Child Level". • **Attribute Level** Select **Root Level** to choose from top-level attributes or **Child Level** for child attributes. You cannot choose both child attributes and top-level attributes in a single roll-up analysis. • **Attribute Category** Select a category to limit selected attributes to that category.

The table shows a check mark next to any displayed attribute included in the rollup.

10. In the functions table, specify the rollup calculations and output:
    a. Select the check boxes next to any statistical function you want the rollup to calculate.
    b. Click **Map** to store results from a calculation to an output attribute.
       You can map the output to an existing attribute or map the output to a new output attribute.
    c. Optional. At the top of the table, click **Evaluate** to verify the output values of the selected functions.
11. Optional. To verify the analysis, examine the results produced with historical data:
    a. In the analysis list, right-click the analysis and then click **Preview Results**.
    b. Enter a start time and end time not later than the current time, and click **Generate Results** to see a list of results.
12. To apply changes and save your work locally, click ✓ on the toolbar.
13. Click **Check In** on the toolbar to save the analysis template to the PI AF server and to create the analysis for any element based on the element template.

Create a rollup analysis

In PI System Explorer, you create a rollup analysis for a selected element. You select input attributes for the rollup either from attributes of the selected element or from attributes of child elements.

Prerequisite: Identify the statistical functions you want the rollup to calculate and where you want to save the results. You can map results from a function to an existing attribute of the element, or you can create a new output attribute when you configure the analysis.

1. In the navigator, click **Elements**.
2. In the Elements browser, select the element where you want to create the analysis.

3. In the viewer, click the **Analyses** tab.

   The tab lists any analyses already defined for the element.
4. Create a new analysis for the element:
5. If no analyses exist, click **Create a new analysis** to create the first one.
6. Otherwise, click **New Analysis** to create a new one.
7. Enter information to identify the analysis:
   - **Name**

     The name that uniquely identifies this analysis.
   - **Description**

     Optional text that describes the analysis.
   - **Categories**

     Optional category that you can assign to the analysis. Click the list and select the check box next to categories you want to assign or click **New** to create a new category.
   - **Analysis Type**

     Click **Rollup**
8. Under **Rollup attributes from**, indicate the source of input values:
   - Click **Child elements of Element Name** to roll up attributes of a child element.
   - Click **This element - Element Name** to roll up attributes of the current element.
9. Select the attributes to include in the rollup:

If rollup attributes come from ...	**Do this ...**
Child elements	a. From the **Sample Child Element** list, select the element from which you want to view attributes. The table shows attributes of the selected element. To improve performance, the table only shows a limited number of attributes. You can click the **Show more attributes** link below the table to show additional attributes from the selected element. b. To the left of the table, specify criteria that select attributes to roll up: a. **Attribute Name** Type an entire attribute name or part of a name with wildcard characters. For example, type `temp*` to select all attributes that begin with "temp," such as Temperature and Temp1. **Note:** The selection of attributes depends on whether *Attribute Level* is set to "Root Level" or "Child Level". b. **Attribute Level**

If rollup attributes come from ...	Do this ...
	Select **Root Level** to choose from top-level attributes or **Child Level** for child attributes. You cannot choose both child attributes and top-level attributes in a single roll-up analysis. c. **Attribute Category** Select a category to limit selected attributes to those in that category. d. **Element Category** Select an element category to limit selected attributes to attributes that have parent elements in that category. e. **Element Template** Select an element template to limit selected attributes to attributes that have parent elements based on that template. **Note:** The criteria apply to all attributes of the selected child element, not just those shown in the table. You can specify criteria that select attributes not shown in the table. For example, if you set **Sample Child Element** to an element from category CatX but set **Element Category** to CatY, you will select none of the attributes in the table but the analysis will still include the attributes in the rollup.
This element	To the left of the table, specify criteria that select attributes to roll up: • **Attribute Name** Type an entire attribute name or part of a name with wildcard characters. For example, type `temp*` to select all attributes that begin with "temp," such as Temperature and Temp1. **Note:** The selection of attributes depends on whether *Attribute Level* is set to "Root Level" or "Child Level". • **Attribute Level** Select **Root Level** to choose from top-level attributes or **Child Level** for child attributes. You cannot choose both child attributes and top-level attributes in a single roll-up analysis. • **Attribute Category**

If rollup attributes come from ...	Do this ...
	Select a category to limit selected attributes to that category.

The table shows a check mark next to any displayed attribute included in the rollup.

10. In the functions table, specify the rollup calculations and output:
    a. Select the check boxes next to any statistical function you want the rollup to calculate.
    b. Click **Map** to store results from a calculation to an output attribute.

       You can map the output to an existing attribute or map the output to a new output attribute.
    c. Optional. At the top of the table, click **Evaluate** to verify the output values of the selected functions.
11. Specify scheduling for the analysis. See Analysis scheduling.
12. Optional. To verify the analysis, review the results produced with historical data:
    a. In the analysis list, right-click the analysis and then click **Preview Results**.
    b. Enter a start time and end time not later than the current time, and click Generate Results to see a list of results.
13. To apply changes and save your work locally, click ✓ on the toolbar.

    This does not run the analysis.
14. Click **Check In** on the toolbar.

    PI System Explorer saves and checks in your analysis, and then starts running the analysis.

Event frame generation analyses

An event frame generation analysis specifies the conditions to start and end event frames automatically.

This type of analysis includes either one or two expressions. When a single condition triggers both the start and the end of an event frame, only a start trigger expression is needed, depicted by the **StartTrigger1** field. For example, a temperature value rising above a threshold might start an event frame and end it when the value returns below the threshold. When the start and end conditions are different, an **EndTrigger** expression is also needed. Because they test start and end conditions, expressions in event frame generation analyses must evaluate to true or false. For more information, see Start and end trigger conditions.

You can write up expressions to be used as start or end triggers. If your expression gets very complex, you can break it into a group of smaller expressions and assign them to variables. For more information, see Expression simplification with refactoring.

Beginning with the 2018 release, a new way of event frame generation is possible where you do not have to set an explicit trigger. For more information, see Implicit modes of event frame generation in asset analytics.

When configuring event frame generation analyses, you can add expressions to **Outputs at Close** group to write outputs back to event frame attributes. Depending on whether you want to save the history of the output, you can select either attributes configured with a PI point data reference or static event frame attributes. When saving output history to event frame attributes configured as a PI Point data reference, the attribute must be mapped to a corresponding PI Point data reference attribute of the referenced element.

- Outputs from root cause event frames are only written to static attributes.
- Parent event frames with multiple child event frames will also write only to static attributes. For more

information on child event frame generation, see Child event frame generation with multiple start triggers.

In your expressions, you can use inputs that come from assets to write outputs to event frame attributes at the close of an event frame. You can also use the **EventFrame** function to access the start time, end time and duration of the event frame. For more information, see expression function EventFrame.

An event frame naming pattern typically includes substitution parameters. For more information on naming pattern, see Naming patterns.

- Parameters *%Endtime%* and *%Duration%* in an event frame naming pattern will be evaluated at the end of an event frame when the event frame closes.
- An attribute value may update at any given time. To avoid confusion, substitution parameter *%@Attribute%* (which resolves to a value of an attribute) in an event frame naming pattern will always be (re-)evaluated in the context of a start of an event frame.
- Default naming pattern (*%ANALYSIS% %STARTTIME:yyyy-MM-dd HH:mm:ss.fff%*) will be used in the absence of a name or if the event frame name evaluates to a null value.
- In a rare case when the duration of an event frame with *%@Attribute%* in its naming pattern matches the configured **True for**, the event frame name will revert to the default naming pattern.

You can create multiple start triggers for your analysis and specify different Boolean conditions to trigger each event frame. For more information on multiple start triggers and child event frame generation, see Child event frame generation with multiple start triggers.

Note: If you change the name of a start trigger from the default name, you must change all the start triggers and use unique names for each trigger.

A spike in input data can trigger the start of an unwanted event frame. To counter the effects of data spikes, you can require that the start trigger remain true for a set time interval before creating an event frame. Enter a value in the **True for** field to specify the time interval.

You can also specify a Severity level for the trigger. The choices for severity are: None, Information, Warning, Minor, Major, and Critical.

An event frame generation analysis is based on an event frame template, which specifies the attributes for the generated event frames. Before you create an event frame generation analysis, be sure an appropriate event frame template is available.

Event frames usually include a referenced element collection. The element associated with an event frame generation analysis becomes the primary reference element for its generated event frames. For more information, see Primary referenced element. Other elements can be referenced using relative paths based on the primary referenced element, as described in Data references to attributes in other elements.

For some events, such as a forced shutdown, you may want to evaluate the conditions leading up to the event. To do that, you can have PI Analysis Service generate root cause event frames. When configured, every generated event frame includes a child root cause event frame that captures attributes for a specified time interval just before the start of the generated event frame. For more information on generating a root cause event frame, see step 11 on Create an event frame generation analysis template.

Video

For information on how to set up event frame generation analyses, watch this video:

Implicit modes of event frame generation in asset analytics

Beginning with the 2018 release, users of asset analytics in PI AF have the option to select different modes of event frame generation. Prior to 2018, users had to explicitly set a start trigger (now called Explicit Trigger). In 2018 or later versions of PI AF, you can choose among the three implicit modes in addition to an explicit trigger: pulse, step and step continuous. After having selected Event Frame Generation as your analysis type, select one of the following as **Generation Mode**:

- Pulse: a transition from zeroth state to any other state starts an event frame, and transition to the zeroth state ends the event frame. You have an option to generate a root cause event frame in **Advanced Event Frame Settings**.
- Step: any change in value ends one event frame and starts the next, except when the value changes to the zeroth state, which ends the current event frame but does not start a new event frame.
- Step Continuous: all transitions would close the open event frame and start a new event frame

Note that you must select a triggering attribute that is either integer, string or enumeration value for these event frames. For pulse and step event frames, you will need to select an attribute as a trigger and set zeroth state. Zeroth state is not supported for step continuous event frames. Zeroth state determines when the event frame should be closed. You can select zeroth states from digital state or enumeration sets. For more information, see PI EFGen topic Event start and end detection.

Create an event frame generation analysis template

In PI System Explorer, you can create an event frame generation analysis template for an element template.

1. In the navigator, click **Library**.
2. In the Library browser, select the element template where you want to create the analysis template.
3. In the viewer, click the **Analysis Templates** tab.

The tab lists any analysis templates already defined for the element template.

4. Create a new analysis template for the element template:
   - If no analysis templates exist, click **Create a new analysis template** to create the first one.
   - Otherwise, click **New Analysis Template** to create a new one.
5. Enter information to identify the analysis:
   - **Name**

     The name that uniquely identifies this analysis.
   - **Description**

     Optional text that describes the analysis.
   - **Categories**

     Optional category that you can assign to the analysis. Click the list and select the check box next to categories you want to assign or click New to create a new category.
   - **Analysis Type**

     Click **Event Frame Generation**.
6. Optional. Clear the **Enable analyses when created from template** check box so that the analysis is created in `Disabled` status.

   By default, a new analysis will be created with `Enabled` status. You must clear the option if you do not want to start the analysis when created.
7. Optional. To temporarily evaluate expressions in the analysis template based on values of a particular element, click the **Select an example element** link and select an element based on the current element template.

   If no elements have been created from the current element template, you must create an element first.

   Templates have no concrete data associated with their attributes. You cannot evaluate an expression in an analysis template, unless you borrow attribute values from a specific element.

   PI System Explorer sets **Example Element** to the element you selected.
8. From the **Event Frame Template** list, select the template for the event frame that this analysis generates.

   To create an event frame template, see Create event frame templates. An event frame naming pattern typically includes substitution parameters. For more information on naming pattern, see Naming patterns.

   Note: Parameters *%Endtime%* and *%Duration%* in an event frame naming pattern will be evaluated at the end of an event frame when the event frame closes.An attribute value may update at any given time. To avoid confusion, substitution parameter *%@Attribute%* (which resolves to a value of an attribute) in an event frame naming pattern will always be (re-)evaluated in the context of a start of an event frame. Default naming pattern (*%ANALYSIS% %STARTTIME:yyyy-MM-dd HH:mm:ss.fff%*) will be used in the absence of a name or if the event frame name evaluates to a null value.
9. Specify the necessary expressions for the analysis:
   a. In the **StartTrigger1** row, enter the Boolean expression that specifies the condition that starts an event frame.

   b. Optional. Change the default name of the start trigger.

      Note: If you change the name of a start trigger from the default name, you must change the names of all start triggers and use unique names for each trigger.

   c. Optional. In the **True for** field, enter the amount of time that the start-trigger condition must be true before the analysis starts an event frame.

Note: Specify a value in the **True for** field to reduce unwanted event frames caused by momentary fluctuations in input data.

d. Optional. In the **Severity** field, enter the severity of the trigger.

   The choices for severity are: None, Information, Warning, Minor, Major, and Critical. If you have defined multiple start triggers, the one with the highest severity will be the effective start trigger; if all start triggers have the same severity, the first start trigger in the list will be the effective start trigger.

e. Add multiple start triggers by clicking on **Add a new start trigger**.

   For each start trigger, enter a Boolean expression, specify a value for **True for** and select a value for **Severity**.

   Note: If you have changed the default name of a start trigger, make sure to assign unique names to all the start triggers.

f. Optional. If a different condition ends the event frame, enter the expression in the **EndTrigger** row. Otherwise, the event frame will close when the start trigger condition is no longer true.

g. Optional. Click **Add a new variable** to insert a new expression row where you can specify a variable and expression for use in one of the trigger expressions.

h. Optional. Click **Add a new output expression** to insert a new expression to write outputs to event frame attributes.

   You can map the output as static (doesn't save history) or PI point (saves history) attribute. When mapping to PI point attribute, make sure that it is mapped back to the attribute on the asset.

10. Optional. If you specified an example element, click **Evaluate** to verify that output values of individual expressions.

11. Optional. Generate a root cause event frame for every event frame that the analysis generates.

    A root cause event frame captures asset data, which can help you analyze conditions just before the start of the event frame.

    Note: Outputs from root cause event frames are only written to static attributes.

    a. From the Advanced Event Frame Settings window, select **Generate child root cause event frame before parent event frame starts**.

    b. In the **Duration** field, enter the length of the root cause event frame.

       The root cause event frame starts this long before the generated parent event.

12. Optional. Map a start trigger name and start trigger expression to an event frame attribute.

    a. From the Advanced Event Frame Settings window, select **Save start trigger name to event frame attribute** and then select an attribute (template) or create a new attribute template on the event frame template to which the start trigger name is saved.

    b. From the Advanced Event Frame Settings window, select **Save start trigger expression to event frame attribute** and then select an attribute (template) or create a new attribute template on the event frame template to which the start trigger expression is saved.

       The start trigger name and start trigger expression can now be seen on generated event frames, on the selected attributes.

13. Specify scheduling for the analysis.See Analysis scheduling.

14. Optional. If you specified an example element, verify the analysis by reviewing the results produced with historical data:

    a. In the analysis list, right-click the analysis and then click **Preview Results**.

   b. Enter a start time and end time not later than the current time, and click **Generate Results** to see a list of results.

15. To apply changes and save your work locally, click ✓ on the toolbar.
16. Click **Check In** on the toolbar to save the analysis template to the PI AF server and to create the analysis for any element based on the element template.

Create an event frame generation analysis

In PI System Explorer, you can create an event frame generation analysis for an individual element.

Prerequisites: Check that an event-frame template is available for the analysis.

1. In the navigator, click **Elements**.
2. In the browser, select the element where you want to create the analysis.
3. In the viewer, click the **Analyses** tab.

   The tab lists any analyses already defined for the element.
4. Create a new analysis for the element:
5. If no analyses exist, click **Create a new analysis** to create the first one.
   - Otherwise, click **New Analysis** to create a new one.
6. Enter information to identify the analysis:
   - **Name**

     The name that uniquely identifies this analysis.
   - **Description**

     Optional text that describes the analysis.
   - **Categories**

     Optional category that you can assign to the analysis. Click the list and select the check box next to categories you want to assign or click New to create a new category.
   - **Analysis Type**

     Click **Event Frame Generation**.
7. From the **Event Frame Template** list, select the template for the event frame that this analysis generates.

   To create an event frame template, see Create event frame templates. An event frame naming pattern typically includes substitution parameters. For more information on naming pattern, see Naming patterns.

   Note: Parameters *%Endtime%* and *%Duration%* in an event frame naming pattern will be evaluated at the end of an event frame when the event frame closes. An attribute value may update at any given time. To avoid confusion, substitution parameter *%@Attribute%* (which resolves to a value of an attribute) in an event frame naming pattern will always be (re-)evaluated in the context of a start of an event frame. Default naming pattern (*%ANALYSIS% %STARTTIME:yyyy-MM-dd HH:mm:ss.fff%*) will be used in the absence of a name or if the event frame name evaluates to a null value.
8. Specify the necessary expressions for the analysis:

   a. In the **StartTrigger1** row, enter the Boolean expression that specifies the condition that starts an event frame.

   b. Optional. Change the default name of the start trigger.

Note: If you change the name of a start trigger from the default name, you must change all the start triggers and use unique names for each trigger.

c. Optional. In the **True for** field, enter the amount of time that the start-trigger condition must be true before the analysis starts an event frame.

   Note: Specify a value in the **True for** field to reduce unwanted event frames caused by momentary fluctuations in input data.

d. Optional. In the **Severity** field, enter the severity of the trigger.

   The choices for severity are: None, Information, Warning, Minor, Major, and Critical. If you have defined multiple start triggers, the one with the highest severity will be the effective start trigger; if all start triggers have the same severity, the first start trigger in the list will be the effective start trigger.

e. Add multiple start triggers by clicking on **Add a new start trigger**.

   For each start trigger, enter a Boolean expression, specify a value for **True for** and select a value for **Severity**.

   Note: If you have changed the default name of a start trigger, make sure to assign unique names to all the start triggers.

f. Optional. If a different condition ends the event frame, enter the expression in the **EndTrigger** row. Otherwise, the event frame will close when the start trigger condition is no longer true.

g. Optional. Click **Add a new variable** to insert a new expression row where you can specify a variable and expression for use in one of the trigger expressions.

h. Optional. Click **Add a new output expression** to insert a new expression to write outputs to event frame attributes.

   You can map the output as static (doesn't save history) or PI point (saves history) attribute. When mapping to PI point attribute, make sure that it is mapped back to the attribute on the asset. See Expressions for analyses for more information.

9. Optional. Generate a root cause event frame for every event frame that the analysis generates.

   A root cause event frame captures asset data, which can help you analyze conditions just before the start of the event frame.

   Note: Outputs from root cause event frames are only written to static attributes.

   a. From the Advanced Event Frame Settings window, select **Generate child root cause event frame before parent event frame starts**.

   b. In the **Duration** field, enter the length of the root cause event frame.

      The root cause event frame starts this long before the generated parent event.

      Note: The root cause event frame is created only for the first instance, not for each time that the trigger is `True`.

10. Optional. Map a start trigger name and start trigger expression to an event frame attribute.

    Note: We recommend that you provide meaningful names to your analysis start triggers while saving them to an event frame attribute.

    a. From the Advanced Event Frame Settings window, select **Save start trigger name to event frame attribute** and then select an attribute (template) or create a new attribute template on the event frame template to which the start trigger name is saved.

    b. From the Advanced Event Frame Settings window, select **Save start trigger expression to event frame attribute** and then select an attribute (template) or create a new attribute template on the event frame template to which the start trigger expression is saved.

The start trigger name and start trigger expression can now be seen on generated event frames, on the selected attributes.

11. Specify scheduling for the analysis. See Analysis scheduling.
12. Optional. To verify the analysis, review the results produced with historical data:
    a. In the analysis list, right-click the analysis and then click **Preview Results**.
    b. Enter a start time and end time not later than the current time, and click **Generate Results** to see a list of results.
13. To apply changes and save your work locally, click ✓ on the toolbar.

    This does not run the analysis.
14. Click **Check In** on the toolbar.

    PI System Explorer saves and checks in your analysis, and then starts running the analysis.

Start and end trigger conditions

An event frame generation analysis starts an event frame when the start trigger condition evaluates to true. Because a random spike in an input value can make the condition true, noisy data can lead to the creation of many unwanted event frames. To counter the effects of data spikes, you can set a time interval before starting an event frame by entering a value in the **True for** field.

When a start trigger condition changes to true, the **True for** value determines how long to wait to evaluate it again; if it is still true, an event frame is started.

With *Periodic* scheduling, the analysis will only evaluate at the time specified in the schedule irrespective of the **True for** setting. **True for** evaluation in this case will be the first periodic evaluation following the end of **True for** period.

Depending on how you configure the *Periodic* schedule, there may not be any evaluation within the **True for** period. If you want to ensure that the analysis evaluates within that time frame, you may do so by setting the periodic schedule to be shorter than the **True for** value (for example, evaluate every minute and **True for** 5 minutes) or, opt for an *Event-Triggered* scheduling.

If you want the analysis to evaluate right at the end of **True for** period, align the **True for** to fall on the periodic schedule (evaluate every 3 minutes with **True for** value of 9 minutes, for example) or, choose the *Event-Triggered* scheduling.

With *Event-Triggered* scheduling, the analysis will evaluate every time the value of the triggering input attribute updates. Configuring the analysis to be *Event-Triggered* and having it triggered off of every input attribute in the start trigger condition can prevent the analysis from missing evaluations.

The start time for the event frame is the time that the start trigger condition first evaluated to true. The **True for** value has no effect on the start or end time of the event frame.

The event frame will close when the start trigger condition is false. Configuring an end trigger is optional. If both the start and end trigger conditions are configured, the event frame will end when the end trigger condition is met. This is regardless of whether the start trigger condition still holds true.

Tip: If using both start and end triggers, make sure the expressions never evaluate to TRUE at the same time since this may lead to event frames with zero second durations. Try to configure your event frames to use only a start trigger expression.

Child event frame generation with multiple start triggers

PI Analysis Service can capture child event frames. Child event frames are useful when you need to send escalating notifications or perform an in-depth analysis of the past events. This topic discusses the following:

- Video that shows you how to configure event frames with multiple start triggers
- Effective start triggers
- Example of child event frame generation

Note: This topic is only applicable to the explicit trigger.

Video

For information on setting up event frames with multiple start triggers and severities, watch this video:

Effective start triggers

Event frame generation analysis supports multiple start triggers. When triggering an analysis, the effective start trigger is determined by taking the following three things into account: trigger evaluation, severity and order of the triggers.

- **Evaluation**

  If only one start trigger is *True*, then that is the effective start trigger. If multiple start triggers evaluate to *True*, then the analysis must take severity into account.
- **Severity**

  If the multiple start triggers evaluate to *True*, the one with the highest severity level is the effective start trigger. For example, if there are two start triggers, *StartTrigger1* with severity *Major* and *StartTrigger2* with severity *Critical*, and they both evaluate to *True* at the same point in time, then *StartTrigger2* is considered the effective trigger as it has the higher severity.

Note: Severity levels in decreasing order of magnitude are: *Critical*, *Major*, *Minor*, *Warning*, *Information* and *None* (default).

- **Order**

  If multiple start triggers with the same severity evaluate to *True*, then the start trigger defined first is the effective start trigger.

In the end, there is only one effective start trigger.

Child event frame generation due to change in the effective start trigger

An analysis with multiple start triggers may automatically create child event frames. For example, an event frame generation analysis would complete the following steps:

1. A start trigger evaluates to *True* and there is no open event frame. The analysis starts an event frame.
2. In a subsequent evaluation, while the event frame is still open, the effective start trigger does not change and continues to evaluate to *True*. Then the event frame will remain open.
3. Another evaluation causes the effective start trigger to be different than step 1. This causes the analysis to do the following:
   a. Start a new parent event frame with the same start time as the original event frame.
   b. Make the original event frame a child of this new parent event frame.
   c. Close the original event frame (now a child event frame).
   d. Create a new child event frame with the new effective start trigger.
4. Another evaluation does not cause the effective start trigger to change. The analysis keeps both the child and the parent event frames open.
5. Another evaluation causes the effective start trigger to change again. Then:
   a. The child event frame created in 3d closes.
   b. Yet another child event frame is created by this new effective start trigger.
6. Another evaluation causes all of the start triggers to be *False*. Then:
   a. Child event frame created in 5b closes.
   b. Parent event frame created in 3a closes.

SQC analyses

Using PI System Explorer, you can create SQC analyses that use standard Statistical Quality Control (SQC) tests to determine if the value of a PI AF attribute lies within user-specified margins of error. Further, you can specify either attributes or event frames as outputs of your SQC analysis.

The following SQC pattern tests may be selected:

- **Outside Control:**

  Counts the number of samples outside the control limit on one side of the center line.
- **Outside 2 Sigma:**

  Evaluates the sample against a limit drawn 2/3 of the way between the center line and the control limit.

- **Outside 1 Sigma:**

  Evaluates the sample against a limit drawn 1/3 of the way between the center line and the control limit.
- **One Side of Center Line:**

  Counts the number of samples on one side of the center line.
- **Stratification:**

  Counts the number of samples that fall within the upper and lower One Sigma limits on both sides of the center line.
- **Mixture:**

  Counts the number of samples that fall outside the upper and the lower One Sigma limits on both sides of the center line.
- **Trend:**

  Counts the number of samples which are increasing or decreasing in a monotonic sequence.

Create an SQC analysis template

In PI System Explorer, you can create an SQC analysis template for an element template.

1. In the navigator, click **Library**.
2. In the Library browser, select the element template where you want to create the analysis template.
3. In the viewer, click the **Analysis Templates** tab.

   The tab lists any analysis templates already defined for the element template.
4. Create a new analysis template for the element template:
   - If no analysis templates exist, click **Create a new analysis template** to create the first one.
   - Otherwise, click **New Analysis Template** to create a new one.
5. Enter information to identify the analysis:
   - **Name**

     The name that uniquely identifies this analysis.
   - **Description**

     Optional text that describes the analysis.
   - **Categories**

     Optional category that you can assign to the analysis. Click the list and select the check box next to categories you want to assign or click **New** to create a new category.
   - **Analysis Type**

     Click **SQC**.
6. Optional. Clear the **Enable analyses when created from template** check box so that the analysis is created in *Disabled* status.

   By default, a new analysis will be created with *Enabled* status. You must clear the option if you do not want to start the analysis when created.
7. Optional. Select an example element.
8. Provide the inputs for the SQC pattern test. All these inputs should be attribute templates previously defined in your PI System.

   a. In the **Source** field, select the PI AF attribute template that provides the data for the pattern test.
   b. In the **Upper Control Limit** and **Lower Control Limit** fields, select the PI AF attribute template that provides the data for the largest and smallest limits on the dimension of the data being tested.
   c. In the **Center Line** field, select the PI AF attribute template that provides the normal or ideal value of the data.

9. Choose the output of your analysis as either **Event Frame** or **AF attribute**.

   If you choose **Event Frame**, select an event frame template. If you choose **AF attribute**, you can either choose from a list, or create a new attribute template. SQC Analysis rule automatically creates an enumeration set named **SQC Enumeration**. This enumeration set has the same values used in Data Archive SQC DigitalState set so that continuity with the existing Data Archive is preserved. **SQC Enumeration** is created when: 1) you map the output to a new **AF attribute**, 2) this or another enumeration set with the required name and values does not already exist and 3) you have the right to create enumeration sets.

   AF attribute output defaults to a string attribute if the aforementioned conditions are not met.

10. Optional. Generate a root-cause event frame for every event frame that the analysis generates.

    A root-cause event frame captures asset data, which can help you analyze conditions just before the start of the event frame.

    a. From the **Advanced Event Frame Settings** menu, select **Generate Child Root Cause Event Frame Before Parent Event Frame Starts**.
    b. In the **Duration** field, enter the length of the root-cause event frame.

       The root-cause event frame starts this long before the generated parent event.

       Note: The root-cause event frame is created only for the first instance, not for each time that the trigger is *True*.

11. Optional. Map a start trigger name and start trigger expression to an event frame attribute.

    a. From the Advanced Event Frame Settings window, select **Save start trigger name to event frame attribute** and then select an attribute template on the event frame template to which the start trigger name is saved.
    b. From the Advanced Event Frame Settings window, select **Save start trigger expression to event frame attribute** and then select an attribute template on the event frame template to which the start trigger expression is saved.

    The start trigger name and start trigger expression can now be seen on generated event frames, on the selected attributes.

12. Enter SQC pattern test information.

    a. Select the check box for each SQC pattern test that you wish to apply to your data.
    b. In the **X of Y** column, select appropriate numerical values for X and Y for each selected pattern test that requires this input.

       Specify the number of values X out of a number of sampled values Y that can be above or below the corresponding pattern test limits.

       Patterns (X and Y values) suggested by Western Electric are shown by default.

       Note: X may not be greater than Y.

    c. Select the **Limit** as **Above**, **Below** or **Both**, for each selected pattern test.

       By default, tests are applied on both sides of the center line.

13. Optional. Select the **Clear on Control Limit Change** check box.

With **Clear on Control Limit Change** selected, if any of the control limits changes, asset analytics resets the state of a running SQC calculation, and the SQC calculation is restarted with the changed values. In addition, associated open event frames are closed before the calculation is resumed.

With **Clear on Control Limit Change** not selected, the calculation simply proceeds using the changed values for the control limits.

If all selected SQC tests pass, the **Output Value** field displays "Normal". If some tests fail, the **Output Value** field shows the highest priority test that failed. Changing any of the SQC inputs will clear the **Output Value** field.

14. Optional. Click **Evaluate** to verify the output values at both **Value at Last Trigger** and **Value at Evaluation times**.

    If all selected SQC tests pass, the **Output Value** field displays "Normal". If some tests fail, the **Output Value** field shows the highest priority test that failed. Changing any of the SQC inputs will clear the **Output Value** field.
15. Specify scheduling for the analysis. See Analysis scheduling.
16. Click **Check In** on the toolbar to save the analysis template to the PI AF server and to create the analysis for any element based on the element template.

Create an SQC analysis

Identify the Statistical Quality Control (SQC) pattern tests that you wish to use. See SQC analyses for more information.

1. In the navigator, click **Elements**.
2. In the Elements browser, select the element where you want to create the analysis.
3. In the viewer, click the **Analyses** tab.

   The tab lists any analyses already defined for the element.
4. Create a new analysis for the element:
   - If no analyses exist, click **Create a new analysis** to create the first one.
   - Otherwise, click **New Analysis** to create a new one.
5. Enter information to identify the analysis:
   - **Name**

     The name that uniquely identifies this analysis.
   - **Description**

     Optional text that describes the analysis.
   - **Categories**

     Optional category that you can assign to the analysis. Click the list and select the check box next to categories you want to assign or click **New** to create a new category.
   - **Analysis Type**

     Click **SQC**.
6. Provide the inputs for the SQC pattern test. All these inputs should be attributes or attribute templates previously defined in your PI System.
   a. In the **Source** field, select the PI AF attribute or attribute template that provides the data for the pattern

test.

   b. In the **Upper Control Limit** and **Lower Control Limit** fields, select the PI AF attribute or attribute template that provides the largest and smallest limits on the dimension of the data being tested.
   c. In the **Center Line** field, select the PI AF attribute or attribute template that provides the normal or ideal value of the data.

7. Choose the output of your analysis as either **Event Frame** or **AF attribute**.

   If you choose **Event Frame**, select an event frame template. If you choose **AF attribute**, you can either choose from a list, or create a new attribute. SQC Analysis rule automatically creates an enumeration set named **SQC Enumeration**. This enumeration set has the same values used in Data Archive SQC DigitalState set so that continuity with existing Data Archive is preserved. **SQC Enumeration** is created when: 1) you map the output to a new **AF attribute**, 2) this or another enumeration set with the required name and values does not already exist and 3) you have the right to create enumeration sets.

   AF attribute output defaults to a string attribute if the aforementioned conditions are not met.

8. Optional. Generate a root-cause event frame for every event frame that the analysis generates.

   A root-cause event frame captures asset data, which can help you analyze conditions just before the start of the event frame.

   a. From the **Advanced Event Frame Settings** menu, select **Generate Child Root Cause Event Frame Before Parent Event Frame Starts** .
   b. In the **Duration** field, enter the length of the root-cause event frame.

      The root-cause event frame starts this long before the generated parent event.

      Note: The root-cause event frame is created only for the first instance, not for each time that the trigger is *True*.

9. Optional. Map a start trigger name and start trigger expression to an event frame attribute.

   a. From the Advanced Event Frame Settings window, select **Save start trigger name to event frame attribute** and then select an attribute or attribute template on the event frame template to which the start trigger name is saved.
   b. From the Advanced Event Frame Settings window, select **Save start trigger expression to event frame attribute** and then select an attribute or attribute template on the event frame template to which the start trigger expression is saved.

      The start trigger name and start trigger expression can now be seen on generated event frames, on the selected attributes.

10. Enter SQC pattern test information.

    a. Select the check box for each SQC pattern test that you wish to apply to your data.
    b. In the **X of Y** column, select appropriate numerical values for X and Y for each selected pattern test that requires this input.

       Specify the number of values X out of a number of sampled values Y that can be above or below the corresponding pattern test limits.

       Patterns (X and Y values) suggested by Western Electric are shown by default.

       Note: X may not be greater than Y.

    c. Select the **Limit** as **Above**, **Below** or **Both**, for each selected pattern test.

       By default, tests are applied on both sides of the center line.

11. Optional. Select the **ClearOnControlLimitChange** check box.

With **ClearOnControlLimitChange** selected, if any of the control limits changes, asset analytics resets the state of a running SQC calculation, and the SQC calculation is restarted with the changed values. In addition, associated open event frames are closed before the calculation is resumed.

With **ClearOnControlLimitChange** not selected, the calculation simply proceeds using the changed values for the control limits.

If all selected SQC tests pass, the **Output Value** field displays "Normal". If some tests fail, the **Output Value** field shows the highest priority test that failed. Changing any of the SQC inputs will clear the **Output Value** field.

12. Optional. Click **Evaluate** to verify the output values at both **Value at Last Trigger** and **Value at Evaluation** times. If all selected SQC tests pass, the Output Value field displays "Normal". If some tests fail, the Output Value field shows the highest priority test that failed. Changing any of the SQC inputs will clear the Output Value field.
13. Click **Check In** on the toolbar.

    PI System Explorer saves and checks in your analysis, and then starts running the analysis.

Sample analyses

The sample analyses in this section perform the following calculations in the same PI AF database:

1. Uses an expression type analysis to track the deviation from target efficiency for the asset and processes represented by the element, *Ethylene*.
2. Creates a template to generate a set of rollup analyses that aggregate the total power draw of all processes at a refinery. The element template from which the element for each refinery was created is called *Refinery*.
3. Creates event frames to capture data any time the 15-minute running-average efficiency for the element, *Ethylene*, drops below 90%.

Build an expression analysis to study efficiency deviation

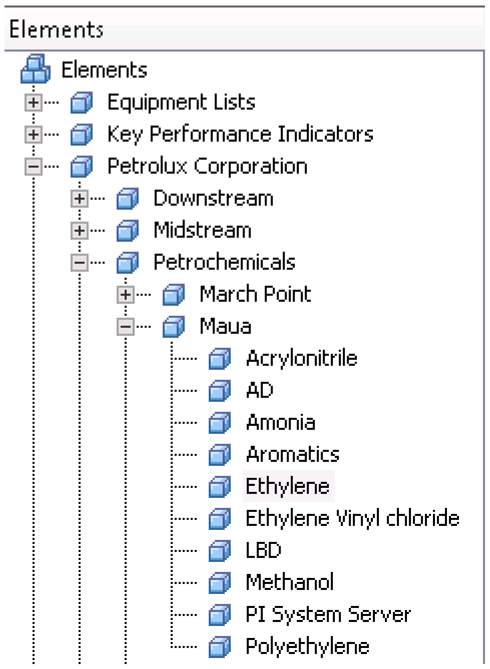
This example demonstrates how to build an expression type analysis. The analysis tracks how the efficiency of a process deviates from the target efficiency.

First, we'll calculate the hourly average efficiency, then determine how much it varies from the target efficiency.

This example creates a new PI point to store the output data (the deviation from target efficiency, as it varies over time). When you create your own analyses, it is also recommended that you create the PI points required for output data before you start to create the analysis.

Prerequisite: Create an element named *Ethylene* that has an attribute named *Efficiency* that contains a PI point data reference.

1. Under **Elements**, select the element for which you want to build the analysis, such as **Ethylene**.

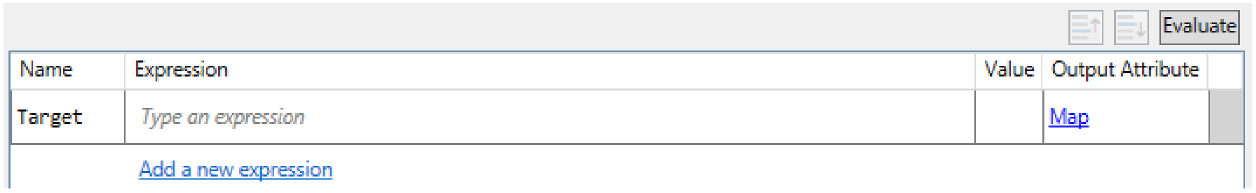


2. Click the **Analyses** tab.
3. Create a new analysis for the element:
   - If no analyses exist, click **Create a new analysis** to create the first one.
   - Otherwise, click **New Analysis** to create a new one.
4. Enter information to identify the analysis:
   - **Name**

     The name that uniquely identifies this analysis. For example, `EfficiencyCalc`.
   - **Analysis Type**

     Click **Expression**. The expression analysis is the simplest type of analysis; it contains one or more expressions plus scheduling information. For more information, see Expression analyses.
5. In the expressions table, create a variable to hold the constant value of the target efficiency:

   a. In the **Name** column, type `Target`.



Name	Expression	Value	Output Attribute
Target	Type an expression		Map

   b. In the **Expression** column, enter `90`.

   This sets this variable to the constant value of 90.
6. Create a second variable to calculate the hourly average efficiency:

   a. Click **Add a new expression**.

   b. Name the variable `HourlyEfficiency`.

   c. In the **Expression** column, enter the following performance equation syntax:

```
    TagAvg('Efficiency', '*-1h', '*')
```

This expression calculates the average value of the attribute called Efficiency over the past hour.

7. Create a third variable to calculate how much the hourly efficiency deviates from the target of 90:
   a. Click **Add a new expression**.
   b. Name the variable *Deviation*.
   c. In the **Expression** column, enter the following expression:
      ```
              HourlyEfficiency - Target
      ```
8. Save the output from the Deviation variable to a PI point:
   a. In the **Output Attribute** column, click **Map**.
   b. Click **New Attribute** to create a new PI AF attribute.
   c. To define the attribute as a PI point data reference, click **Yes** to save output history.
   d. In the **Name** field, enter the attribute name, such as `EfficiencyDifference`.
   e. Click **OK** to create the attribute.
9. Specify scheduling for the analysis:
   a. Set the **Scheduling** option to **Periodic**.
   b. Click **Configure** and specify a period of 30 seconds.

   The calculation will run every 30 seconds.
10. Verify that the results look reasonable:
    a. In the analyses table, right-click the analysis and then click **Preview Results**.
    b. Enter the time range of the preview.

       The default time range starts at the preceding midnight and ends at the current time, but you can enter different start and end times, if you prefer.
    c. Click **Generate Results**.
11. When you are satisfied with your new analysis, click **Check In** on the toolbar to check in your work.

    PI System Explorer saves and checks in your analysis, and then starts running the analysis.

    A green circle in the **Status** column of the analyses table indicates that PI Analysis Service is running the analysis.
12. Optional. Backfill the PI point with results from the calculation using past data.

    PI Analysis Service can back fill calculation results that have output attributes mapped to PI point data references. However, the PI point cannot contain data at the specified times. If desired, you can delete pre-existing data and then back fill that time range.
    a. Right-click the analysis in the analyses table, and click **Backfill/Recalculate**.
    b. Specify a start and end time for back filling.
    c. Select **Permanently delete existing data and recalculate**.
    d. Click **Start**.

Create a template for power rollup analyses

This example demonstrates how to create a template that you can use to generate a set of rollup analyses. The analysis template aggregates the total power draw of all processes at a refinery.

Prerequisites: Create the following necessary objects in your PI AF database:

- Element template named *Refinery*.
- Element named *Rotterdam Refinery*, derived from the *Refinery* template
- Child element named *Catalytic Cracking* under the *Rotterdam Refinery* element
- Attribute named *Power Draw* under the *Catalytic Cracking* child element

1. In the Library browser, select the element template for which you want to create the analysis template.

   In this example, the element template is called Refinery. This template is used to create refinery elements.
2. Click the **Analysis Templates** tab.
   - Name

     The name that uniquely identifies this analysis. For example, `PowerRollup`.
   - Analysis Type

     Click **Rollup**. A rollup analysis calculates statistical data for specified attributes, such as a total or an average value. For more information, see Rollup analyses.
3. Enter information to identify the analysis:
   - Name

     The name that uniquely identifies this analysis. For example, `PowerRollup`.
   - Analysis Type

     Click **Rollup**. A rollup analysis calculates statistical data for specified attributes, such as a total or an average value. For more information, see Rollup analyses.
4. Select an example element to provide candidate attributes:
5. Click the **Select an example element** link to open a window that shows elements derived from the selected element template.
6. Select an element and click **OK**.

   For example, in the list of elements derived from the Refinery template, click **Rotterdam Refinery**.
7. For the **Rollup attributes from** option, click **Child elements of Rotterdam Refinery** to indicate that the analysis rolls up attributes of child elements of the refinery element.
8. Select the attributes to include in the rollup:
9. From the **Sample Child Element** list, select the element from which you want to view attributes.

   For example, select **Catalytic Cracking** to see attributes from the Catalytic Cracking element in the attributes table.

   The table only shows attributes from the selected element. The rollup include attributes from all child elements that match the criteria that you specify.
10. To the left of the attributes table, specify criteria that select attributes to roll up.

    For example, in the **Attribute Name** field, type `Power*` to specify all attributes that begin with Power. The attributes table shows a check mark next to any attributes in the selected child element that match this criteria. However, the rollup includes attributes from any child element that match the criteria, not just those in the table.
11. In the functions table, specify the rollup calculations and output:
12. Select the check boxes next to any statistical function you want the rollup to calculate.

    For example, click the **Sum** check box to total values.
13. In the **Output(s)** column, click **Map** to store results from the calculation to an output attribute.

14. Click **New Attribute Template** to create a new attribute template with a PI point data reference.
15. Specify values that define the created attribute and point, and then click **OK**.

    For example, you can use the default values, which name the point and attribute PowerRollup_Sum and write the point to the default Data Archive, indicated by the substitution parameter.
16. Click **Evaluate** and check whether the result returned in the **Value** column is reasonable.
17. Set the **Scheduling** option to **Event-Triggered**.
18. To verify the analysis, examine the results produced with historical data:
19. In the analyses list, right-click the analysis and then click **Preview Results**.
20. Enter a start time and end time not later than the current time, and click **Generate Results** to see a list of results.
21. Click **Check In** on the toolbar to save the analysis template to the PI AF server and to create the analysis for any element based on the element template.

Create event frames automatically to track inefficiency

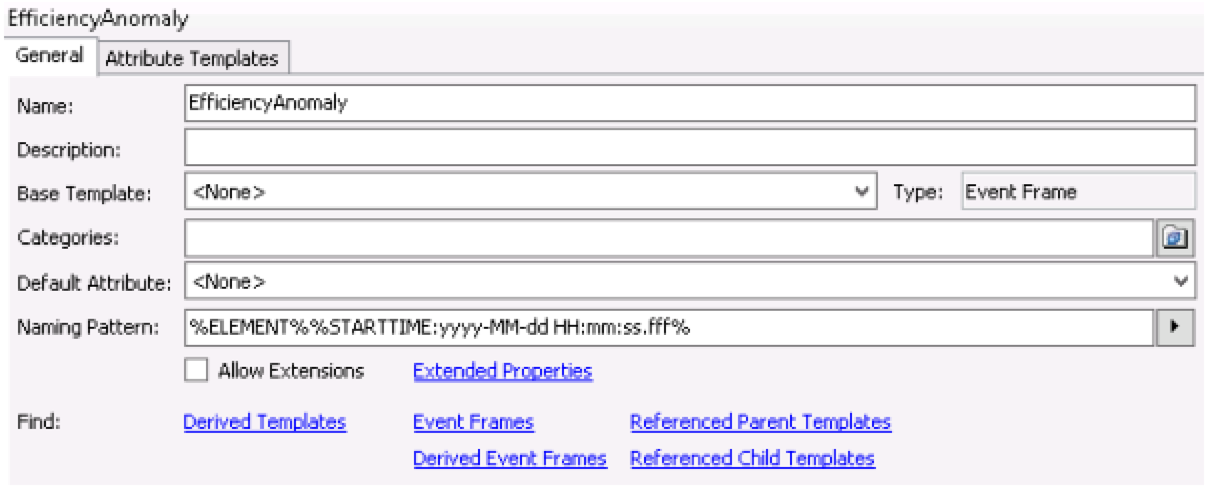
We created a sample analysis to track how the efficiency of a process deviates from the target efficiency in Build an expression analysis to study efficiency deviation. In this example, we create event frames to capture data any time the 15-minute running-average efficiency drops below 90%.

Prerequisite: Complete the steps in Build an expression analysis to study efficiency deviation.

1. In the Library browser, expand **Templates**.
2. Right-click **Event Frame Templates** and then click **New Template** to create a new event frame template.
3. In the **Name** field, enter `EfficiencyAnomaly`.
4. In the **Naming Pattern** field, specify the name of the event frames produced from the template.

   You can use substitution parameters to specify a varying name. PI AF resolves the parameters when creating the event frames. Then, each event frame will have a unique, identifiable name.
   a. Click the arrow next to the field to see valid substitution parameters that you can add to the name.
   b. Select **%ELEMENT%**.
   c. Select **%STARTTIME:yyyy-MM-dd HH:mm:ss.fff%**.

      The *STARTTIME* substitution parameter includes the date-time formatting pattern to use.

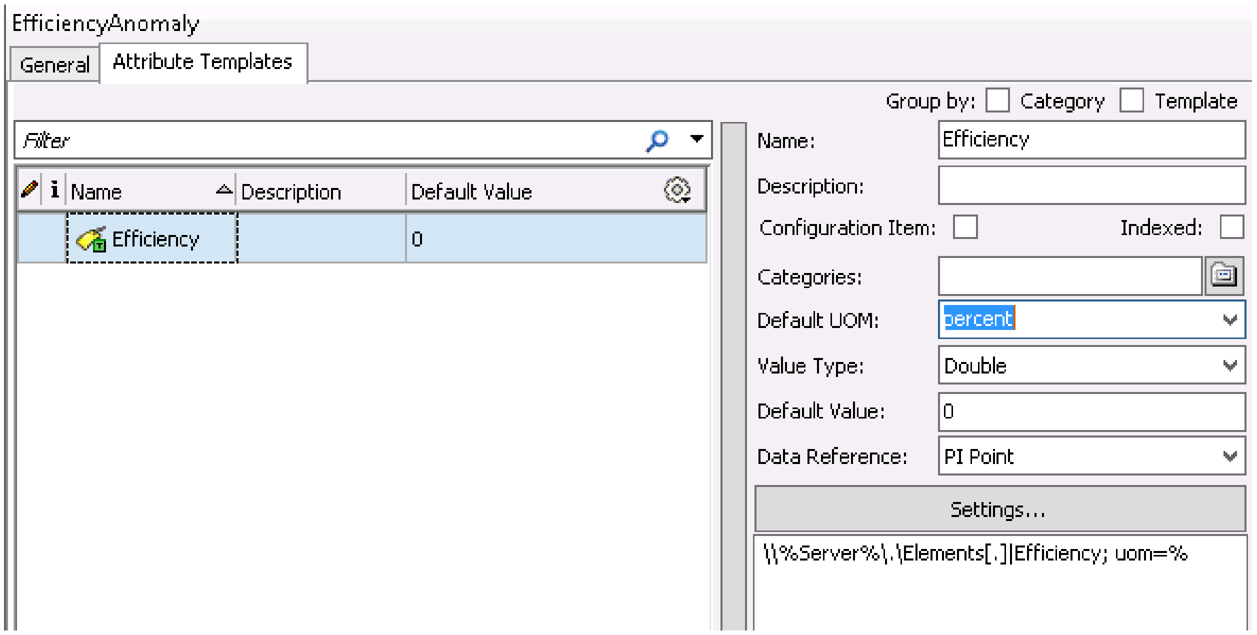

EfficiencyAnomaly

Name:	EfficiencyAnomaly
Description:	
Base Template:	<None>
Type:	Event Frame
Categories:	
Default Attribute:	<None>
Naming Pattern:	%ELEMENT%%STARTTIME:yyyy-MM-dd HH:mm:ss.fff%

5. Click the **Attribute Templates** tab.
6. Create a new PI point data reference attribute using an attribute template.
   a. Click **New Attribute Template**.
   b. In the **Name** field, enter `Efficiency`.
   c. Click **Settings** and enter the following configuration string in the **Tag name** field:

   ```
           .\Elements[.]|Efficiency; uom=%
   ```

   d. From the **Default UOM** list, select **percent**.



Name	Description	Default Value
Efficiency		0

Name: Efficiency
Description:
Configuration Item: ☐ Indexed: ☐
Categories:
Default UOM: percent
Value Type: Double
Default Value: 0
Data Reference: PI Point
\\%Server%\.\Elements[.]|Efficiency; uom=%

7. In the Elements pane, click **Ethylene**, the element for which you built an analysis in Build an expression analysis to study efficiency deviation.
8. Click the **Analyses** tab.
9. Click **New Analysis**.
   - **Name**

     Enter `EfficiencyAnomalyEvent`.
   - **Analysis Type**

     Click **Event Frame Generation**.
10. From the **Event Frame Template** list, select **EfficiencyAnomaly**.

    This is the event frame template you created in the earlier steps.
11. Enter expressions that specify the start and end of the event frame:
    - **StartTrigger1**

    ```
            TagAvg('Efficiency', '*-15m', '*') <90
    ```

    Creates an event frame whenever the 15-minute average efficiency drops below 90%.
    - **EndTrigger**

    ```
            TagAvg('Efficiency', '*-15m', '*') >92
    ```

    Ends the event frame whenever the 15-minute average efficiency has recovered and exceeds 92%.
12. Click **Evaluate** to check the current values of the expressions.

The **Value** column is populated with either *True* or *False* for each condition.

a. For the **Scheduling** option, click **Periodic**.

b. Click **Configure** and set the analysis to run every minute.

13. Specify scheduling for the analysis.

    a. For the **Scheduling** option, click **Periodic**.

    b. Click **Configure** and set the analysis to run every minute.

14. Check in your work.

Manage analyses and the PI Analysis Service

Asset Analytics provides tools and metrics to help you gain insights into your analyses and PI Analysis Service performance.

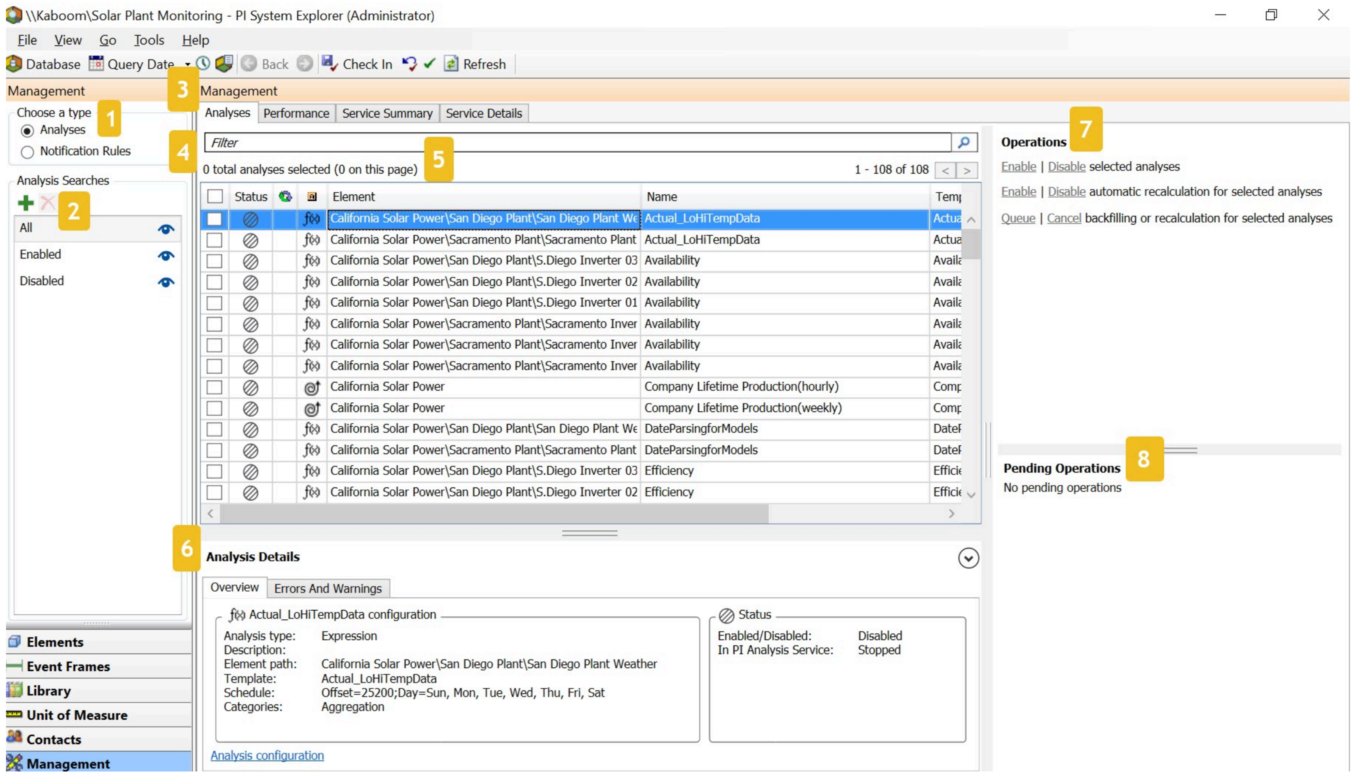
Note: The Management feature is only available if your installation of PI System Explorer includes the Management plug-in. To configure the PI Analysis Service, you must be signed in using an account with appropriate permissions.

For information on how to manage notification rules, see Management of notification rules.

Important: If the PI Analysis Service is configured to use claims, users who sign on to PI System Explorer can use OpenID Connect (OIDC) to access other claims-enabled PI System resources. If OIDC is not enabled, Windows authentication is used.

Bulk management of analyses

The Management pane is where you view, search for, manage, and perform bulk operations on analyses.



0 total analyses selected (0 on this page)

1 - 108 of 108

	Status			Element	Name	Temp
☐	⊘		f(x)	California Solar Power\San Diego Plant\San Diego Plant We	Actual_LoHiTempData	Actua
☐	⊘		f(x)	California Solar Power\Sacramento Plant\Sacramento Plant	Actual_LoHiTempData	Actua
☐	⊘		f(x)	California Solar Power\San Diego Plant\S.Diego Inverter 03	Availability	Availa
☐	⊘		f(x)	California Solar Power\San Diego Plant\S.Diego Inverter 02	Availability	Availa
☐	⊘		f(x)	California Solar Power\San Diego Plant\S.Diego Inverter 01	Availability	Availa
☐	⊘		f(x)	California Solar Power\Sacramento Plant\Sacramento Inver	Availability	Availa
☐	⊘		f(x)	California Solar Power\Sacramento Plant\Sacramento Inver	Availability	Availa
☐	⊘		f(x)	California Solar Power\Sacramento Plant\Sacramento Inver	Availability	Availa
☐	⊘			California Solar Power	Company Lifetime Production(hourly)	Comp
☐	⊘			California Solar Power	Company Lifetime Production(weekly)	Comp
☐	⊘		f(x)	California Solar Power\San Diego Plant\San Diego Plant We	DateParsingforModels	DateP
☐	⊘		f(x)	California Solar Power\Sacramento Plant\Sacramento Plant	DateParsingforModels	DateP
☐	⊘		f(x)	California Solar Power\San Diego Plant\S.Diego Inverter 03	Efficiency	Efficie
☐	⊘		f(x)	California Solar Power\San Diego Plant\S.Diego Inverter 02	Efficiency	Efficie

Operations

Enable | Disable selected analyses

Enable | Disable automatic recalculation for selected analyses

Queue | Cancel backfilling or recalculation for selected analyses

Pending Operations

No pending operations

Analysis Details

Overview | Errors And Warnings

Actual_LoHiTempData configuration

Analysis type:	Expression
Description:	
Element path:	California Solar Power\San Diego Plant\San Diego Plant Weather
Template:	Actual_LoHiTempData
Schedule:	Offset=25200;Day=Sun, Mon, Tue, Wed, Thu, Fri, Sat
Categories:	Aggregation

Status

Enabled/Disabled:	Disabled
In PI Analysis Service:	Stopped

Analysis configuration

Overview of Management pane

The table below lists the sections in the Management pane when the **Analyses** option is selected.

Numbered section	Description
1. **Choose a type** option	Select **Analyses** to display and search for analyses in the current AF database
2. **Analysis Searches** section	Select the plus (+) button to search for specific analyses. See Search for analyses.
3. **Management** pane (center)	Provides access to information about performance for both analyses and the PI Analysis Service. For more information, see Overview of the Analyses tab, Overview of the Performance tab, View analytic performance data, and View analysis service statistics.
4. Filter text box	Enter search criteria to locate specific analyses in the list.
5. Analyses list	Contains a list of all the analyses in the current AF database.
6. **Analysis Details** section	Lists configuration details and error and warning messages for the selected analysis.
7. **Operations** section	Provides access to enable/disable, automatic calculation, and other operations when the **Analyses** tab is open.
8. **Pending Operations** section	Shows the status of bulk operations for analyses when the **Analyses** tab is open.

Note: The **Analyses** tab lists details for analyses in the currently selected AF database only. The other three tabs -- **Performance**, **Service Summary**, and **Service Details** -- provide details on analytic performance across the entire AF server.

Overview of the Analyses tab

The **Analyses** tab provides information on analyses, and contains these columns:

- Status
- (Automatic recalculation)
- (Analysis Type)
- Element
- Name (of analysis)
- Template

- Backfilling (status)

Note: Click one or more column headers to sort analyses by element, name or template.

On the **Analyses** tab, the "~" character or word "about" (for example: "~1000" or "1-500 of about 1000") indicates an approximate number of analyses. In certain cases, the exact number of analyses is not displayed until all analyses have been loaded.

Analysis selection

Each page of results can contain up to 500 individual analyses. Use the following techniques to quickly select analyses on the **Analyses** tab:

- To select one or more analyses: Select the check box to the left of an analysis.
- To select all analyses on the current page: Click the checkbox on the top-left corner.
- To select all analyses in the results (when more than 500 analyses exist): Click the checkbox on the top-left corner then click the **Select all analyses on all pages** link.

Note: Clearing specific analyses automatically makes your subsequent operation page-based. For example, if you first select all 1000 analyses, and then clear two, the operation will be performed on 498 (500 minus 2) and not 998 (1000-2) analyses.

Backfill and recalculate operations

You can enable, disable, or backfill/recalculate the analyses you have selected. For specific selection criteria, see the "Analysis selection" section above. For PI AF version 2017 R2 and later, you can enable or disable automatic recalculation of the analyses you selected. For more information, see Enable or disable automatic recalculation for multiple analyses.

Obtain adequate permissions

You must have Execute permission on any analyses you want to backfill or recalculate data on. Proper permission can be obtained by mapping your account to the Asset Analytics Recalculation identity. Manual recalculation is only available for PI Analysis Service 2016 R2 or later. In addition, the Data Archive that stores PI points must run version 2016 R2 or later. For more information, see Backfill or recalculate data for multiple analyses.

Expression, rollup, and SQC analyses

Expression, rollup, or Statistical Quality Control (SQC) analyses can backfill or recalculate data if the analysis outputs are mapped to PI point attributes. If PI points already have data for the backfill time period, you can retain existing data and fill in missing data, or delete the data and recalculate.

Event frame generation analyses

Event frame generation analyses can also recalculate event frames over a specified time period. The recalculation process, regardless of the selected option, automatically deletes all existing event frames for that time period, as well as annotations on affected event frames.

Pending operations

The status of these bulk operations is shown: Enable, disable, backfill or recalculate a group of analyses.

View analysis details

Select an analysis and take one of the following actions to view specific information:

- To view a summary of the analysis configuration and status, click the **Overview** tab.
- To view error details, click the **Errors and Warnings** tab.
- To view or edit the selected analysis, click **Analysis Configuration** to go to the **Analyses** tab of its associated element.

View, sort, and edit analyses

Follow the procedure below to view, sort, filter, and edit analysis details. See Overview of the Analyses tab for a detailed description of the operations you can perform on this tab.

1. Navigate to the Management plug-in.
2. Select the **Analyses** option.

   The **Analyses** tab opens.
3. Optional. On the center pane, select the **Analyses** tab.

   The **Analyses** tab opens and lists all analyses.

   Note: One page of results can contain up to 500 records.
4. Optional. Use the back and forward arrows to navigate through analyses page results.
5. Perform any of the optional, following steps to sort, filter, view details, and edit analyses:

To...	Do this...
Sort analyses by element, name or template.	Select a column header to sort analyses data by the column header in ascending order (A-Z or highest to lowest value). Select the column header again to sort data in descending order.
Filter all analyses on the current page by keyword(s).	Enter search criteria in the **Filter** text box, then press Enter. **Note:** Use the **Filter** text box to search only analyses on the current page of the **Analyses** tab.
Remove a search filter from the current page.	Delete the search criteria in the **Filter** text box.
Filter all analyses in the current AF database.	See Search for analyses.
View an analysis' type, description, name, element path, template, schedule, and any assigned categories.	Select the checkbox to the left of an analysis in the table to view this information. **Note:** Information is displayed on the **Overview** tab of the **Analysis Details** section.
View error or warning messages for an analysis.	Select the checkbox to the left of an analysis in the table, then select the **Errors and Warnings** tab in the **Analysis Details** section.
Enable or disable an analysis.	Select the checkbox to the left of an analysis in the

To...	Do this...
	table, then select the **Enable or Disable for analyses** link under **Operations** in the left pane.
Enable or disable an automatic recalculation for an analysis.	Select the checkbox to the left of an analysis in the table, then select the **Enable or Disable automatic recalculation for analyses** link under **Operations** in the left pane.
View or edit a selected analysis.	Select an analysis in the table, then select the **Analysis configuration** link on the **Overview** tab. Opens the **Analyses** tab of the associated element.

Note: To enable or disable a group of analyses, select the checkbox to the left of the first analysis, then navigate to the last analysis and [SHIFT+Click] the checkbox next to it. You can then select and apply the desired operation from the right pane under the Operations heading.

See Backfilling and recalculation of analyses and Backfill or recalculate data for multiple analyses to learn more about these tasks.

Search for analyses

You can create a customized search to locate analyses in the current AF database that meet specific criteria.

1. Navigate to the Management plug-in.
2. Select the **Analyses** option.
3. Use the provided search or create your own.

Task	Action
Use a provided search	Select **All**, **Enabled** or **Disabled**. These are view-only and cannot be modified, as indicated by the (**View selected search**) icon.
Create a custom search	**Note:** Customized search is only visible to the user who created it on the computer where it was created. Creating your own search is a new feature in PI AF 2017. Refer to the previous versions of PI System Explorer User Guide if you are using an earlier version of PI AF. a. Click the (**Add new search**) button. b. Select a search criteria. Choices are: • Name • Description • Element Name

Task	Action

	• Template
	• Category
	• Enabled/Disabled: enabled or disabled
	• Service Status: PI Analysis/Notifications Service status
	  ◦ Error
	  ◦ Running
	  ◦ Stopped
	  ◦ Suspended
	  ◦ Warning
	Note: Service Status is based on the PI Analysis Service connection and its running status. This cannot be combined with other search criteria.

View/edit a search	Click the ✎ (**View/edit selected search**) button to view or edit your search.
Delete a search	To delete, click the ✕ (**Delete selected search**) button.

PI Analysis Service management

Asset Analytics provides tools and metrics to help you gain insights into PI Analysis Service performance. You can use the information on these tabs to diagnose and troubleshoot issues related to analysis service performance:

- Service Summary tab: Contains a summary of your PI Analysis Service's performance.
- Service Details tab: Provides detailed statistics related to PI Analysis Service performance.

PI Analysis Service runs all analyses in the current PI AF server. Two features help you manage the service:

- The Service Details tab enables you to explore statistics about the service to monitor its operations. These statistics are particularly important in diagnosing and identifying a solution for performance issues. You can save these statistics as a text file, which can provide a detailed snapshot of your analysis service to share with support staff helping you troubleshoot a problem. See View analysis service statistics.
- The Configuration window enables you to view or modify settings for analysis service properties to adjust performance. See View or modify analysis service configuration and Analysis service configuration.

These options are available if your installation of PI System Explorer includes the Management plug-in and you are signed in to a user account with permissions to configure PI Analysis Service.

Important: If the PI Analysis Service is configured to use claims, users who sign on to PI System Explorer can use OpenID Connect (OIDC) to access other claims-enabled PI System resources. If OIDC is not enabled, Windows authentication is used.

Overview of the Performance tab

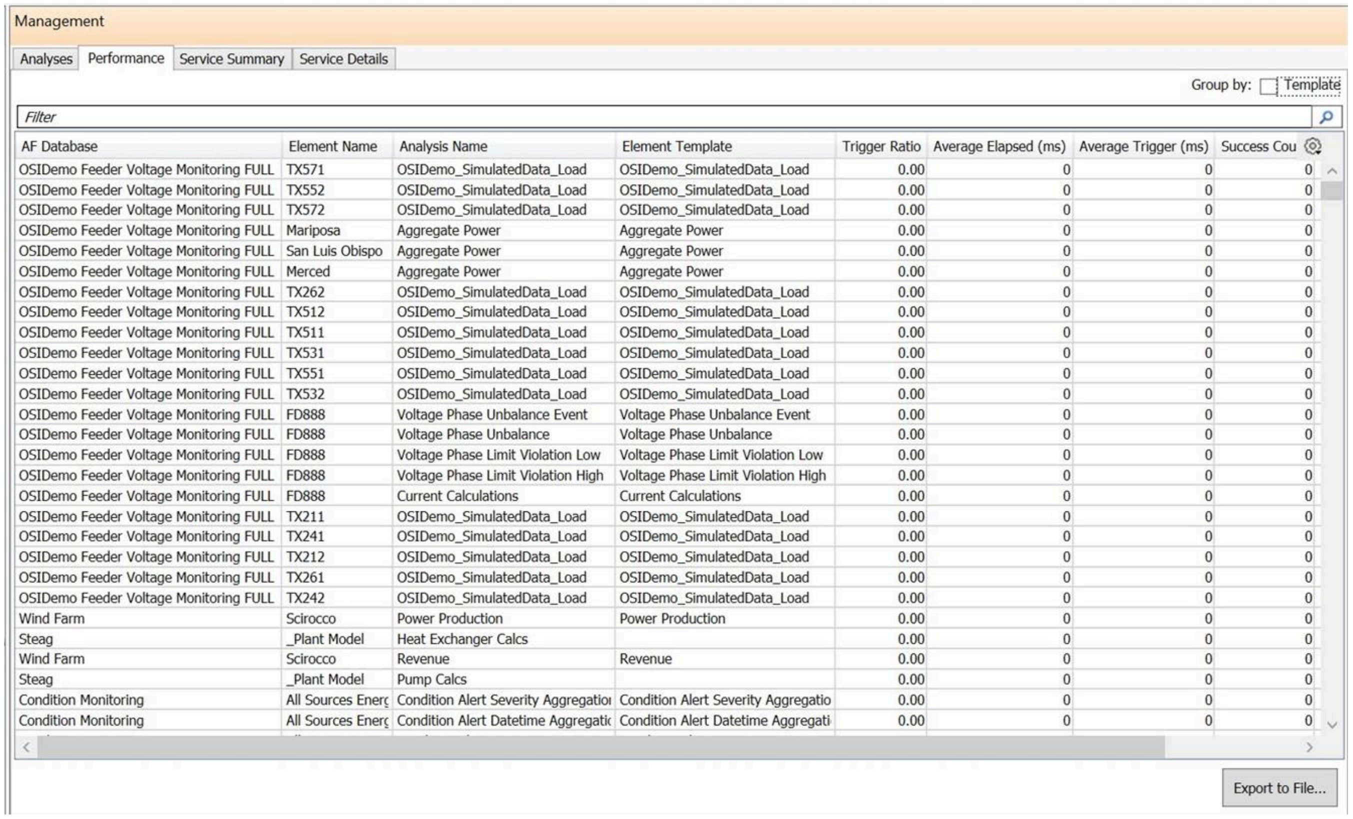
The Performance tab provides helpful insights into individual analytic and group analyses performance. The metrics on this tab can help identify potential issues and problematic analyses.

By default, analyses are grouped by template. To view a full list of analyses on the PI AF Server and their statistics, deselect the **Group by: Template** option.

Note: The analyses on the Performance tab are retrieved from the default PI AF Server, not just the current AF database.

From this tab, you can perform the following tasks:

- View analysis performance
- Sort analyses by column headers
- Add and remove column headers
- Filter analyses
- Export table data to a file



AF Database	Element Name	Analysis Name	Element Template	Trigger Ratio	Average Elapsed (ms)	Average Trigger (ms)	Success Cou
OSIDemo Feeder Voltage Monitoring FULL	TX571	OSIDemo_SimulatedData_Load	OSIDemo_SimulatedData_Load	0.00	0	0	0
OSIDemo Feeder Voltage Monitoring FULL	TX552	OSIDemo_SimulatedData_Load	OSIDemo_SimulatedData_Load	0.00	0	0	0
OSIDemo Feeder Voltage Monitoring FULL	TX572	OSIDemo_SimulatedData_Load	OSIDemo_SimulatedData_Load	0.00	0	0	0
OSIDemo Feeder Voltage Monitoring FULL	Mariposa	Aggregate Power	Aggregate Power	0.00	0	0	0
OSIDemo Feeder Voltage Monitoring FULL	San Luis Obispo	Aggregate Power	Aggregate Power	0.00	0	0	0
OSIDemo Feeder Voltage Monitoring FULL	Merced	Aggregate Power	Aggregate Power	0.00	0	0	0
OSIDemo Feeder Voltage Monitoring FULL	TX262	OSIDemo_SimulatedData_Load	OSIDemo_SimulatedData_Load	0.00	0	0	0
OSIDemo Feeder Voltage Monitoring FULL	TX512	OSIDemo_SimulatedData_Load	OSIDemo_SimulatedData_Load	0.00	0	0	0
OSIDemo Feeder Voltage Monitoring FULL	TX511	OSIDemo_SimulatedData_Load	OSIDemo_SimulatedData_Load	0.00	0	0	0
OSIDemo Feeder Voltage Monitoring FULL	TX531	OSIDemo_SimulatedData_Load	OSIDemo_SimulatedData_Load	0.00	0	0	0
OSIDemo Feeder Voltage Monitoring FULL	TX551	OSIDemo_SimulatedData_Load	OSIDemo_SimulatedData_Load	0.00	0	0	0
OSIDemo Feeder Voltage Monitoring FULL	TX532	OSIDemo_SimulatedData_Load	OSIDemo_SimulatedData_Load	0.00	0	0	0
OSIDemo Feeder Voltage Monitoring FULL	FD888	Voltage Phase Unbalance Event	Voltage Phase Unbalance Event	0.00	0	0	0
OSIDemo Feeder Voltage Monitoring FULL	FD888	Voltage Phase Unbalance	Voltage Phase Unbalance	0.00	0	0	0
OSIDemo Feeder Voltage Monitoring FULL	FD888	Voltage Phase Limit Violation Low	Voltage Phase Limit Violation Low	0.00	0	0	0
OSIDemo Feeder Voltage Monitoring FULL	FD888	Voltage Phase Limit Violation High	Voltage Phase Limit Violation High	0.00	0	0	0
OSIDemo Feeder Voltage Monitoring FULL	FD888	Current Calculations	Current Calculations	0.00	0	0	0
OSIDemo Feeder Voltage Monitoring FULL	TX211	OSIDemo_SimulatedData_Load	OSIDemo_SimulatedData_Load	0.00	0	0	0
OSIDemo Feeder Voltage Monitoring FULL	TX241	OSIDemo_SimulatedData_Load	OSIDemo_SimulatedData_Load	0.00	0	0	0
OSIDemo Feeder Voltage Monitoring FULL	TX212	OSIDemo_SimulatedData_Load	OSIDemo_SimulatedData_Load	0.00	0	0	0
OSIDemo Feeder Voltage Monitoring FULL	TX261	OSIDemo_SimulatedData_Load	OSIDemo_SimulatedData_Load	0.00	0	0	0
OSIDemo Feeder Voltage Monitoring FULL	TX242	OSIDemo_SimulatedData_Load	OSIDemo_SimulatedData_Load	0.00	0	0	0
Wind Farm	Scirocco	Power Production	Power Production	0.00	0	0	0
Steag	_Plant Model	Heat Exchanger Calcs		0.00	0	0	0
Wind Farm	Scirocco	Revenue	Revenue	0.00	0	0	0
Steag	_Plant Model	Pump Calcs		0.00	0	0	0
Condition Monitoring	All Sources Enerç	Condition Alert Severity Aggregatio	Condition Alert Severity Aggregatio	0.00	0	0	0
Condition Monitoring	All Sources Enerç	Condition Alert Datetime Aggregatic	Condition Alert Datetime Aggregati	0.00	0	0	0

Description of column headings

The following table lists the column headers you can add to the Performance tab. Column headings can be added or removed depending on the amount of information you wish to view.

Column heading	Description
AF Database	The name of the AF database where an analysis is stored.
Analysis Name	The name of the analysis
Average Analysis Count	Indicates the average number of analyses running in the group since the service started up. Generally, this will stay constant at the number of analyses that belong to the group, but can fluctuate if analyses are enabled or disabled over time.
Average Elapsed (ms)	The average amount of time in milliseconds that the analysis took to execute.
Average Error Ratio	The ratio of the average number of times an analysis failed due to an error.
Average Lag (ms)	The average amount of lag time in milliseconds between when the analysis should have run and when it actually ran.
Average Success Ratio	The ratio of the average number of times an analysis successfully ran.
Average Trigger (ms)	The average interval in milliseconds between analysis executions.
Current Evaluation Lag (ms)	The lag time in milliseconds between how long it took the Analysis Service to start and finish the last evaluation.
Current Lag (ms)	The amount of time in milliseconds between the expected time an analysis was supposed to run due to a trigger or schedule and the actual time it executed.
Current Scheduling Lag (ms)	The amount of lag time in milliseconds that an analysis execution was delayed due to waiting in the evaluation queue.
Duplicate Ignored Count	This indicates that the Analysis or group received a triggering event that matched exactly with the latest trigger time. This may be harmless and simply indicate that one or more inputs received triggering events at the same time with the same timestamp, or it could indicate that one of the triggering inputs received late data.
Element Template	Name of the AF element template where an analysis is configured.

Column heading	Description
Error Count	The number of times an analysis tried to run but failed due to an error.
First Trigger Time	The time that the first trigger executed an analysis for evaluation.
Group Name	The name of the analysis or analyses group.
Group Trigger Ratio	A ratio of the average interval between analyses group executions.
Impact score	A rating that indicates an analysis's impact on system performance. This score is calculated for each analysis group using the following formula: Average Elapsed / Average Trigger x Average Analysis Count.
Last Trigger Time	The time that the last trigger executed an analysis for evaluation.
Out of Order Ignored Count	When an analysis is triggered at a timestamp that is prior to the latest trigger time, that trigger is considered out of order and discarded. This indicates that one or more input attributes received triggering events late or out-of-order with respect to one or more other triggering input attributes.
Rank	Indicates the depth of the analysis or analysis group in a dependency chain. Analyses whose input attributes do not come from any other analyses are rank 0. For example, an analysis with 1 or more input attributes that are outputs of rank 0 analyses is rank 1. An analysis with 1 or more input attributes that are outputs of rank 1 analyses is rank 2. The rank of a given analysis corresponds to the deepest input attribute. For a given dependency chain, no rank N analyses will be evaluated until every related analysis of rank N-1 or lower has been evaluated for that same timestamp.
Schedule	How often an analysis is scheduled to run. There are two types of scheduling: Periodic and event-triggered. See Analysis scheduling.
Skip Count	The number of times the execution of an analysis was skipped.
Skipped Evaluation Count	The count of the number of times an analysis evaluation was skipped.

Column heading	Description
Skipped Evaluation Percentage	The percentage of skipped evaluations since the Analysis Service started.
Success Count	The number of times an analysis was evaluated successfully.
Template path	The path of the AF element template where an analysis is stored.
Total Evaluation Count	The total number of times an analysis was evaluated.

View analytic performance data

Follow this procedure to view group or individual analytic performance data. You can also sort analyses by column headers, add or remove columns, group and ungroup analyses by template, search for analyses, and export data to a CSV file. See Performance tab for a description of the column headers on this tab.

1. Navigate to the Management plug-in.
2. Select the **Analyses** option.

   The **Analyses** tab opens.
3. On the center pane, click the **Performance** tab.

   The **Performance** tab opens and displays analytic performance data.
4. Optional: To view a list of all analyses in the current AF Server and their statistics, deselect the **Group by: Templates** option.
5. Perform any of the optional, following steps to modify what analytic performance data is shown on the **Performance** tab:

To...	Do this...
Sort analyses by a column header	Select a column header to sort analyses data in ascending order (A-Z or highest to lowest value). Select the column header again to sort analysis data in descending order.
Reorder columns	Drag-and-drop a column header to a new position.
Filter analyses by keyword or character string	Enter search criteria in the **Filter** text box, then press Enter.
Remove a search filter	Delete the search criteria in the **Filter** text box.
Show or hide column headers	Select the icon, then select a column header to add or remove it from the table.
Display all column headers	Select the icon, then select **Show All**

To...	Do this...
	Columns.
Display only default column headers	Select the ⚙ icon, then select **Show Default Columns**.
Ungroup or group analyses by AF element template	Select the **Group by: Template** checkbox to select or clear this option.
Export analytic data to a CSV file	Select the **Export** button to open the **Save As** dialog, navigate to the desired folder location, enter a file name in the **File name** text box, then select **Save**.

Overview of the Service Summary tab

You can view a summary of your PI Analysis Service's performance on the **Service Summary** tab. Information such as startup time, number of calculations performed, and other details are listed on this tab.

1. Navigate to the Management plug-in, then select the Analyses option in the left pane

   The **Analyses** tab opens.
2. Select the **Service Summary** tab.

   The **Service Summary** tab opens and displays current information about your PI Analysis Service operations and performance.
3. Hover over a tile title to view a pop-up description of the information shown on the tile.

Tile title	Description
Service Startup Time	The amount of time between the initial startup of the AF Analysis Service and when the calculation of analytics began.
Calculations per Second	The average number of analytics calculations evaluated per second.
Skipped Calculations	The total number of calculations skipped by the AF Analysis Service since its startup.
Max. Calculation Lag	The latency of the worst-performing analysis since the startup of the AF Analysis Service. This metric includes a built-in wait time of 5 seconds.
Cache Hit to Miss Ratio	The ratio of the number of AF analysis inputs retrieved from the cache to the number of analysis inputs not in the cache.
Max. Update Duration	The time taken to update the cache for any AF analysis input.

Tile title	Description
Highest Trigger Ratio	The AF analysis template with the highest average trigger ratio. This ratio should be less than 1. This ratio is defined as the template's average elapsed time divided by its average trigger time.

Overview of the Service Details tab

The **Service Details** tab provides statistical insights on PI Analysis Service operations and any PI AF plug-ins. The amount of information shown on this tab depends on which version of the PI Analysis Service is installed. To ensure access to the newest statistics and improvements, upgrade to the latest version.

The following table provides a description of the main sections shown on this tab.

Note: To troubleshoot performance issues, use the **Performance** and **Service Summary** tabs. See View analytic performance data.

Group	Subheading	Description
Plug-Ins	EventFrame	Provides count information on event frame analyses.
	Natural	Provides count information on event-triggered analyses.
	PerformanceEquation	Provides count information on expression analyses.
	PerformanceEquation_EventFrame	Provides count information on expression analyses and the expressions used within event frame generation analyses.
	Rollup	Provides count information on rollup analyses.
Service summary	ManagerStatistics	Provides statistics for the Analysis Manager process, such as the amount of RAM on the PI Analysis Service machine.
	ProcessorStatistics	Provides statistics on PI Analysis Service real-time processing performance.
	RecalculationProcessorStatistics	Provides statistics on PI Analysis Service recalculation processing performance.
Service details	ServiceStartupStatistics	Provides statistics on the steps

Group	Subheading	Description
		performed during service startup and how long each takes before calculations begin.
	AnalysisConfigurationStatistics	Provides counts of different analysis configuration settings (such as analysis type, status, and rank; template-based or not; output types; schedule types).
	EvaluationStatistics	Provides statistics for real-time calculations by calculation group (including the 5 most expensive analyses per group) and for data cache updates.
	RecalculationStatistics	Provides statistics for manual and automatic backfilling and recalculation.

View analysis service statistics

The Analysis Service Statistics window enables you to explore statistics about PI Analysis Service to monitor its operations. These statistics are particularly important in diagnosing and identifying a solution for performance issues. You can save these statistics as a text file, which can provide a detailed snapshot of your analysis service to share with support staff helping you troubleshoot a problem. See the AVEVA Knowledge Base article Asset Analytics Best Practices for more information.

Note: The Management plug-in is required to view analysis service statistics in PI System Explorer.

1. Navigate to the **Management** window, then select the **Analysis** option.

   The **Management** window opens.
2. In the center pane, select the **Service Details** tab.

   The **Service Details** tab opens.

   Note: You can also right-click anywhere in the Operations panel and select View Analysis Service Statistics to open the Service Details pane.
3. Use the arrow to expand the Plug-ins, Service summary or Service details headings and reveal subheadings.
4. Select a subheading to display details on a particular plug-in or set of service statistics.

   A table that contains information about the selected service or plug-in displays below the headings. See Overview of the Service Details tab for section descriptions.
5. To filter analysis service statistics by a particular value or keyword, enter filter criteria in the **Filter** text box, then select [Enter].

   Search results display based on filter criteria.
6. Optional. Select **Refresh** to update statistics with the most current data.

7. Optional. Click Save to open the Save As dialog, then enter a new filename and navigate to the desired folder and click Save.

   The current statistics are saved as a text file.

   Note: Share this text file with customer support staff to help troubleshoot problems.

Analysis service configuration

You can modify these settings in the Configuration window. For information on how to configure the parameters below, see View or modify analysis service configuration.

Note: Changes to the *AutoBackfillingEnabled*, *MaximumAllowedAutoBackfillingSpanInHours* or *NumberParallelDataPipes* settings require a restart of PI Analysis Service to take effect.

- **AutoBackfillingEnabled**

  This property enables or disables automatic backfilling of gaps in data that result from periods when PI Analysis Service is not active. Automatic backfilling is enabled by default. In some cases, it may not be needed (in a test environment, for example). Enter False to disable this property. If you update this setting, you must restart PI Analysis Service.

- **AutoRecalculationEnabled**

  This property enables or disables automatic recalculation of analyses. Automatic recalculation executes analyses when a new input with a time stamp before the latest evaluated trigger is received. This useful feature automatically recalculates late-arriving inputs. Automatic recalculation is enabled by default. Enter False to disable this property.

  Note: Although automatic recalculation is enabled globally by setting this parameter, you need to opt in for an automatic recalculation at individual analysis (template) level. For more information, see Configure automatic recalculation of analyses and analyses templates.

- **AutoRecalculationIgnoreTimeInSeconds**

  This setting allows users to specify a duration whereby any input events with time stamps within this duration and older than the latest trigger time are ignored for automatic recalculation. The default value for this property is 30 seconds. Setting this parameter can be useful for filtering out noises in some analysis inputs that are not desirable for automatic recalculation, for example, if they are known to be delayed.

- **AutoRecalculationMinWaitTimeInSeconds**

  This setting controls how long analysis service will wait before queuing an analysis for automatic recalculation upon receiving an out-of-order triggering event. A valid range for this setting is 30 to 900 seconds, with a default of 60 seconds. The wait time is reset with the arrival of new out-of-order triggering events. Regardless of this setting, any analyses that have had an out-of-order triggering event will be queued for recalculation after 900 seconds.

- **CacheTimeSpanInMinutes**

  PI Analysis Service dynamically determines data cache time span for AF attributes used as inputs in calculations. For example, for analyses calculating hourly summary, PI Analysis Service would automatically cache an hour worth of data for required input attributes. This ensures that data required for calculations is available in the data cache, while unnecessary data can be trimmed. Manually set *CacheTimeSpanInMinutes* only if the time span cannot be determined by calculation configuration. A longer time setting means that any input data needed for calculations is available in the data cache, but it uses more system resources. For calculations that require earlier data (for example, from an hour earlier), you can specify a longer period for this setting to keep data cached, thus avoiding calls to Data Archive. The maximum time span that would be cached is 7 days and any data in the time span beyond 7 days would be trimmed. Note that the *MaxCacheEventsPerAttribute* setting takes precedence over this setting, so both settings may have to be adjusted simultaneously.

- **CalculationWaitTimeInSeconds**

  A period of time to wait before evaluating analyses that are running in real-time. This wait time can be used to compensate for late-arriving inputs. For example, if a calculation depends on an interface with a known latency of up to 30 seconds, you might adjust this setting to at least 30 seconds to make sure you get the most current event from the interface. Longer settings reduce the chance of missing calculation inputs caused by latency but increase the delay in getting calculation results. Note that calculations use input values at the trigger time stamp when they are evaluated. The default value for this property is 5 seconds.

- **CreateFuturePIPointsForAmbiguousOutputTimes**

  A flag that determines the type of PI point to create for newly mapped output attributes that save output history when the time stamp of the output is uncertain. PI Analysis Service attempts to create the PI point type that matches the time stamp of the output, future PI points for output with future time stamps or historical PI points for output with present or past time stamps. If set to False, PI Analysis Service creates historical PI points in cases when the time stamp of the output is ambiguous. If set to True, PI Analysis Service creates future PI points in cases when the time stamp of the output is ambiguous.

- **Default to Auto Create PI Points for Template**

  When this check box is selected, PI Analysis Service automatically creates PI points for output attributes in analyses derived from analysis templates.

- **EvaluationPartitionSize**

  Calculations are grouped for efficient evaluation. Analyses based on an analysis template with periodic scheduling, for example, are evaluated together as one calculation group. To speed up evaluation time, very large calculation groups are partitioned into subgroups based on the *EvaluationPartitionSize* setting; the subgroups can then be evaluated in parallel. The default value for this property is 10,000. For a group of analyses with long individual evaluation times, you may want to specify a smaller partition size. A group of 100 analyses, each performing summary calculations for a day's worth of data, may evaluate faster as 4 subgroups with a partition size of 25, depending on the CPU resources available on the machine.

- **EvaluationsToQueueBeforeSkipping**

  If it takes too long to execute a group of calculations, the queue of waiting calculations may become unmanageable. This setting determines when to skip calculations to avoid running out of resources. At the default setting, if more than 50 evaluations are found, PI Analysis Service starts skipping the oldest evaluations to try to return to its normal processing.

- **IsLoadSheddingEnabled**

  For some applications, accuracy and completeness are critical, and no calculations can be skipped regardless of the delay in completing them. To disable skipping calculations, enter False for this setting.

- **IsTelemetryAllowed**

  Enter False to stop sending usage data from PI Analysis Service for the Customer Experience Improvement Program. (This does not affect the corresponding setting for program participation you can configure in PI System Explorer.)

- **MaxAllowedAutoRecalculationSpanInDays**

  The maximum time span in days for which an analysis will be automatically recalculated on receiving an out of order event. Any recalculation time span exceeding this limit will not be considered for automatic recalculation. The default value for this parameter is 180 days.

- **MaxCacheEventsPerAttribute**

  Once this limit is reached, the oldest events are trimmed from the data cache. Note that this setting takes precedence over the *CacheTimeSpanInMinutes* setting.
- **MaxConcurrentRecalculationRequests**

  The maximum number of backfilling or recalculation operations that PI Analysis Service can perform concurrently. PI Analysis Service dynamically scales the number of threads used for backfilling or recalculation between 1 and the number specified, depending on available resources (CPU and memory) of the machine that is running the service. You can set this parameter to any integer value between 1 and 256. The default value for this parameter is (1/2)**P*, where *P* represents the number of logical processors on the machine that PI Analysis Service is running.
- **MaximumAllowedAutoBackfillingSpanInHours**

  Specify the maximum downtime (in hours) for the PI Analysis Service to initiate automatic backfilling. Setting the parameter to "0" or a negative number will restore the previously recorded setting for this option. If you update this setting, you must restart PI Analysis Service.

  Note: To disable automatic backfilling, you must disable the *AutoBackfillingEnabled* parameter.
- **MinCacheEventsPerAttribute**

  The default setting of 1 ensures that at least one event per attribute remains in the data cache.
- **NumberDataWriterThreads**

  The number of threads PI Analysis Service uses to write analysis outputs. More threads are useful for high frequency calculations that write values at high rates and for writing to Data Archive on a computer with high network latency (other threads can write values while waiting for a response from Data Archive). Consider increasing the number of threads when analyses skip many evaluations or analyses have high evaluation lag. More threads can help if data writes cause a bottleneck. You might set the number of threads to the number of processors on the computer where PI Analysis Service runs.
- **NumberEvaluationThreads**

  The number of threads PI Analysis Service uses for evaluating analyses concurrently. You can set this parameter to any integer value between 1 and 256. The default value for this parameter is *P*, where *P* represents the number of logical processors on the machine that PI Analysis Service is running. Consider increasing the number of evaluation threads when analyses are evaluating with high lag or skipping evaluations, but PI Analysis Service is not fully utilizing available CPU resources. This may happen when you have a lot of analyses with summary calculations that are triggered frequently. Having a larger number of evaluation threads is helpful for analyses that require frequent access to massive amount of input attribute data from outside the data cache. Note that increasing the number of threads for reducing lag or skipped evaluations may not help in cases where PI Analysis Service is already using high CPU.
- **NumberParallelDataPipes**

  Attributes receive updates from data sources, including Data Archive, through data pipes. Using a single data pipe to update a large number of attributes may take an unacceptably long time. You can increase the number of data pipes which can operate simultaneously to decrease update time. This can be especially useful for calculations with high-frequency data. Keep in mind, however, that increasing the number of data pipes increases the load on Data Archive. If you update this setting, you must restart PI Analysis Service.
- **RuntimeStorageFolderPath**

  Specify the location of the directory that contains PI Analysis Service run-time information such as user backfilling and calculation groups . By default, PI Analysis Service stores run-time information in **\\ProgramData\OSIsoft\PIAnalysisNotifications**.

- **Suggested PI Point Name**

  You can modify the naming pattern here with a substitution parameter. For a complete list, see List of PI AF substitution parameters. Note, however, that some parameters may not be meaningful in the context of a PI point name.

  You can specify PI Point attributes as well as PI Point names. For more information on the attribute syntax, see the examples in Configuration strings for PI point data references. Do not enter the *pointtype* and *future* attributes as they are automatically set by asset analytics.

View or modify analysis service configuration

You can change PI Analysis Service configuration settings to help improve performance. See Analysis service configuration for details about these settings.

Note: Changing the settings can greatly impact system performance. If you have any questions about changing the default settings, contact customer support staff for help. You need to have administrative rights to view or change these settings.

1. In the navigator, click **Management**, and then select **Analysis** to view the analysis management area.
2. Right-click anywhere in the **Operations** panel, and then click **Edit Analysis Service Configuration**.

   The Configuration window opens.
3. To modify a property, edit its setting in the **Configuration** column.
4. Click **OK** to apply any changes and close the window.

Buffering PI point outputs for PI Analysis Service

You can set up buffering for PI point attributes that write outputs to Data Archive from PI Analysis Service. This way, you can optimize throughput and create data resiliency by preventing data loss. You can access Buffering manager from PI System Explorer under **Tools** > **Buffering Manager**. See Configure buffering for analysis outputs to PI points for more information.

Backfilling and recalculation of analyses

Backfill or recalculate

When you backfill an analysis for a specified time range, data that are missing are calculated to fill in the gaps for the outputs. Recalculation, on the other hand, deletes all of the data in the time range and calculate results for the analysis outputs again. Currently, there are four different options of backfill and recalculation. For more information, see the knowledge base article Asset analytics backfilling and recalculation milestones and requirements.

Expression, rollup or SQC analyses can be backfilled or recalculated over an earlier time period if the analyses outputs are mapped to PI point attributes. You can backfill the missing data in a time range and retain existing data, or you can recalculate, which deletes and recalculates data in the time range. Event frame generation analyses can recalculate event frames over a specified time period, but automatically delete all existing event frames in that time period, as well as annotations on affected event frames.

In order to backfill/recalculate, you need Execute permission on the analyses. Proper permission can be obtained by mapping your account to Asset Analytics Recalculation identity. For more information on identity mapping, see PI AF identities and mappings.

Note: For analyses writing outputs to attributes configured as PI point, backfilling or recalculation may result in writing out-of-order events. Such events are written to the Data Archive without compression.

Enabling automatic backfilling is useful as it fills the gaps in case a short analysis service downtime occurs. For more information, see parameters *AutoBackfillingEnabled* and *MaximumAllowedAutoBackfillingSpanInHours* in Analysis service configuration.

From PI AF 2017 R2, you can enable automatic recalculation of analyses. PI AF Server and PI Analysis Service 2017 R2 or later are required to make use of automatic recalculation. Although automatic recalculation is globally enabled by default (*AutoRecalculationEnabled*), you need to opt in at individual analysis (template) level if you expect late-arriving or out-of-order data that may trigger recalculation. For more information, see Configure automatic recalculation of analyses and analyses templates, Enable or disable automatic recalculation for multiple analyses and parameters *AutoRecalculationEnabled*, *AutoRecalculationIgnoreTimeInSeconds*, *AutoRecalculationMinWaitTimeInSeconds*, and *MaxAllowedAutoRecalculationSpanInDays* in Analysis service configuration. To examine a log file with a list of queued automatic and manual backfill/recalculation requests for troubleshooting, see Open recalculation log folder.

Note: Event frame generation and SQC analyses that are configured to generate event frames are excluded from automatic recalculation.

Backfill or recalculate data for an analysis

PI Analysis Service 2016 R2 introduced recalculation capabilities for expression, rollup and SQC analyses. Data Archive storing the output PI points must be running version 2016 R2 or later. Beginning with PI Analysis Service 2018, you can opt for recalculation of all dependent analyses. An analysis whose input comes from an output of another analysis is considered dependent.

1. In the Elements browser, select an element, and click the **Analyses** tab to see the analyses defined for the element.
2. From the list of analyses, right-click the running analysis that you want to use to backfill or recalculate data, and click **Backfill/Recalculate**.
3. In the **Backfilling or recalculation for analysis name** window, perform the following actions.
    a. In the **Start Time** and **End Time** fields, specify the time period for which you want to backfill missing data while retaining existing data, or delete and recalculate data. You can use standard PI time expressions or click 📅 to select a specific date period.
    b. Decide how existing data should be treated. For an expression, rollup or SQC analysis that outputs to an attribute:
        a. To ensure that existing data is retained and only missing data is backfilled, select **Leave existing data and fill in gaps**.
        b. If input data or analysis algorithms have changed and you want to delete existing output data, select **Permanently delete existing data and recalculate**.
        c. Optional. Check the box next to **Recalculate dependent analyses** to queue all dependent analyses for manual recalculation.

        Note: Selecting this option may cause recalculation to occur multiple times for dependent analyses. The recalculation time range for dependent analyses may be different than the time range you selected when queuing recalculation. For more information, see note at the end of step 7 in Backfill or recalculate data for multiple analyses.

Note: Event frames are automatically deleted and recalculated for event frame generation analyses. Be aware that information such as annotations, acknowledgments, reason codes on those event frames will be lost.

4. Click **Start** to start the backfill or recalculation.
5. For details on the backfill or recalculation status or to cancel, right-click the analysis and click **Backfill/ Recalculate Status**.

Video

For information on how to backfill or recalculate data for an analysis, watch this video:

Backfill or recalculate data for multiple analyses

PI Analysis Service 2016 R2 introduced recalculation capabilities for expression, rollup and SQC analyses. Data Archive storing the output PI points must be running version 2016 R2 or later. Beginning with PI Analysis Service 2018, you can opt for recalculation of all dependent analyses. An analysis whose input comes from an output of another analysis is considered dependent.

1. In the navigator, select **Management** at the bottom of the list.
2. Select **Analyses** radio button in the **Management** pane.
3. Optional. From **Analysis Searches**, select a search query and then select the result you want to operate upon.

   For example, select **Enabled**.

   Only the analyses that meet the criterion (✓) are displayed.
4. In the Analyses pane, select the analyses you want to backfill or recalculate.
5. In the Operations pane, select **Queue backfilling or recalculation for selected analyses**.

Note: From PI AF 2018 SP2, you can also cancel backfilling on multiple analyses with **Cancel backfilling or recalculation for selected analyses**.

6. In the **Start Time** and **End Time** fields, specify the time period for which you want to backfill missing data while retaining existing data, or delete and recalculate data. You can use standard PI time expressions, or click 📅 to select a specific date period.
7. Choose from the following actions:

To backfill or recalculate ...	Do this ...
Expression, rollup, or SQC analyses only	Decide how existing data should be treated: • To ensure that existing data is retained and only missing data is backfilled, select **Leave existing data and fill in gaps**. • If input data or analysis algorithms have changed and you want to delete existing output data, select **Permanently delete existing data and recalculate**.
A mix of analyses that includes event frame generation analyses or More than a page of analyses selections	Perform one of the following actions: • Keep **Leave existing data and fill in gaps** selected and acknowledge that event frames in the time range will be permanently deleted along with all annotations on those event frames. • Select **Permanently delete existing data and recalculate**. All annotations on event frames will be lost. • Optional. Check the box next to **Recalculate dependent analyses** to queue all dependent analyses for manual recalculation. • Selecting this option may cause recalculation to occur multiple times for dependent analyses • The recalculation time range for dependent analyses may be different than the time range you selected when queuing recalculation

There are three factors affecting the recalculation time range of dependent analyses:

- Time range of the original analysis
- How the output of the original analysis is used in the dependent analyses
- Whether or not the output time override is configured in the original analysis

8. Click **Queue** to start multiple backfills or recalculations.
9. Review the **Pending Operations** pane for details on the backfill or recalculation status.

Reconciliation of event frames during backfilling

When PI Analysis Service is restarted after short down-times, it initiates automatic backfilling concurrently with real-time analysis.

Note: With PI Server 2015 and PI Server 2015 R2, PI Analysis Service does not initiate the automatic backfilling process if down-time is greater than 72 hours. With PI Server 2016, you can configure the *MaximumAllowedAutoBackfillingSpanInHours* configuration parameter in the PI Analysis Service to specify the desired time.

When PI Analysis Service restarts, it will reconcile the open event frames. When automatic backfilling for analyses is completed, asset analytics determines if there are event frames to be reconciled and, accordingly, starts the reconciliation phase. Event frame reconciliation may be necessary for any in-progress event frames that cannot be closed within the auto-backfilling time range because no closing event can be found. These event frames may be generated in either of these ways:

- The event frame may be already present when the analysis service was stopped
- The event frame may be newly created during auto-backfilling

Such in-progress event frames may have to be reconciled with those generated during real-time evaluation.

When you select an analysis using the **Analysis** tab or the **Management** plug-in, an orange icon (✓) in the analysis window indicates that the event frame reconciliation is pending completion. Hover over the icon to see details such as the number of events being processed, start and end times, and the time of last evaluation. When reconciliation is complete, the icon turns from orange to green.

Configure automatic recalculation of analyses and analyses templates

Automatic analysis recalculation is available with PI AF 2017 R2 and later. Event frame generation and SQC analyses that are configured to generate event frames are excluded from automatic recalculation. An event is considered late if its timestamp is earlier than the last execution time of the analysis. The last execution time is the same as the timestamp of the latest triggering event in an analysis (event-triggered) or the last time an analysis ran (periodic).

Prerequisites: Although automatic recalculation is globally enabled by default, you need to opt in at individual analysis (template) level if you expect late-arriving or out-of-order data that may trigger recalculation. PI AF Server and PI Analysis Service 2017 R2 or later are required.

To configure automatic recalculation for a particular analysis (template), follow this procedure.

1. Go to the analysis (template) for which you want to enable automatic recalculation.
2. Click **Advanced** button at the bottom of the pane.

   Advanced options window opens.
3. Check the box next to **Recalculate analysis for out-of-order input events** and click **OK**.

   Note: You cannot opt in or out of automatic recalculation by accessing the analysis if it is based on a template. Go to the template to make changes.

Enable or disable automatic recalculation for multiple analyses

For analyses writing to attributes (expression, rollup and SQC), automatic recalculation is available for PI Analysis Service 2017 R2 or later. In addition, PI AF server 2017 R2 or later is required.

Note: Event frame generation and SQC analyses that are configured to generate event frames are excluded from automatic recalculation.

1. In the navigator, select **Management** at the bottom of the list.
2. Select **Analyses** radio button in the **Management** pane.
3. Optional. From **Analysis Searches**, select a search query and then select the result you want to operate upon.

   For example, select **Enabled**.

   Only the analyses that meet the criterion (✓) are displayed.
4. In the Analyses pane, select the analyses you want to enable or disable automatic recalculation.
5. In the Operations pane, click either **Enable** or **Disable automatic recalculation for selected analyses**.
6. Check the box next to acknowledgment and click **Enable** or **Disable**.
7. Review the **Pending Operations** pane to track the progress of enabled or disabled operation.

Open recalculation log folder

To access a comprehensive log file of queued automatic/manual backfilling and recalculation requests for troubleshooting, follow this procedure.

1. In the navigator, select **Management** at the bottom of the list.
2. Select **Analyses** radio button in the **Management** pane.
3. Right-click anywhere in the **Operations** panel and click **Open Recalculation Log folder**.

   A folder containing a csv file with queued backfill/recalculation opens.

Asset analytics expression functions by type

The following list contains links to asset analytics expression functions by type.

Array operations

- ArrayLength
- FilterData
- FirstValue
- LastValue
- MapData

Date and time

- Bod
- Bom
- Bonm
- Day
- DaySec
- Hour
- Minute
- Month
- Noon
- ParseTime
- Second
- Weekday
- Year
- Yearday

Event frame properties

- EventFrame

Logical

- AND
- ELSE
- Exit
- IF
- IN
- NOT
- OR
- THEN

Math

- Abs
- Acos
- Asin
- Atn
- Atn2
- Ceiling

- Cos
- Cosh
- Cot
- Coth
- Csc
- Csch
- Curve
- E
- Exp
- Float
- Floor
- Frac
- Int
- Log
- Log10
- Logbase
- Mod
- Pi
- Poly
- Pow
- Rate
- Remainder
- Round
- Roundfrac
- Sec
- Sech
- Sgn
- Sin
- Sinh
- Sqr
- Tan
- Tanh
- Trunc

Operators

- Mod

PI Data Archive digital states

- DigState
- DigText
- StateNo

Point attributes

- TagDesc
- TagEU
- TagExDesc
- TagName
- TagNum
- TagSource
- TagSpan
- TagType
- TagTypVal
- TagZero

Search and retrieval

- Convert
- DeltaValue
- FindEq
- FindGE
- FindGT
- FindLE
- FindLT
- FindNE
- InterpolatedValues
- NextEvent
- NextVal
- NumOfChanges
- PrevEvent
- PrevVal
- RecordedValues
- RecordedValuesByCount
- SecSinceChange
- TagVal
- TimeEq

- TimeGE
- TimeGT
- TimeLE
- TimeLT
- TimeNE
- TimeStamp

Statistical

- Avg
- Compare
- Max
- Median
- Min
- Normalrnd
- Poisson
- PStDev
- Rand
- SStDev
- Total

Status

- BadVal
- HasChanged
- HasValueChanged
- IsSet
- NoOutput
- TagBad

String

- Ascii
- Char
- Compare
- Concat
- Contains
- Format
- InStr
- LCase

- Left
- Len
- LTrim
- Mid
- Right
- RTrim
- Split
- String
- Text
- Trim
- UCase

Time series value statistics

- Cov
- EventCount
- LinRegr
- PctGood
- Range
- StDev
- TagAvg
- TagMax
- TagMean
- TagMin
- TagTot

Expression functions reference

Expression functions available for use in expression and event frame generation analyses are listed alphabetically. These functions follow performance equation (PE) syntax.

Note: Expression functions (ArrayLength, FilterData, FirstValue, InterpolatedValues, LastValue, MapData, RecordedValues and RecordedValuesByCount) that enable you to retrieve and manage multiple values as an array may not work very well with attributes configured as PI point array data reference.

Abs

Return the absolute value of an integer or real number.

Syntax

```
Abs(x)
```

Arguments

- x
  An integer or real number

Returns

The absolute value of *x*

Exceptions

Return an error value if *x* is not an integer or real number

Notes

Unit of measure of the argument, if it exists, is carried over to the result

Example

- `Abs(-2.2)`
  [Returns 2.2]
- Abs('att1')
  [Return the absolute value of 'att1' at trigger time]

Acos

Return the inverse cosine (arccos) of an integer or real number. The inverse cosine of *x* is the angle in radians whose cosine is equal to *x*.

Syntax

```
Acos(x)
```

Arguments

- x
  Must be a real number between -1.0 and 1.0, inclusive

Returns

The inverse cosine of *x*, in radians

Exceptions

If *x* is not a number, or is less than -1.0 or greater than 1.0, returns an error value

Example

- `Acos(-0.5)`
  [Returns 2.0944]
- `Acos(0.75)`
  [Returns 0.72273]

AND

The logical conjunction of two expressions that returns *True* if both expressions are true and *False* otherwise.

Syntax

`expression1 AND expression2`

Arguments

- **expression1, expression2**
  Any expression that evaluates to true or false

Returns

True if both expressions are true (non-zero) and *False* (zero) otherwise

Exceptions

None

Notes

If the inputs are numeric, **AND** can do bitwise operations.

Example

- `('att1' > 50) AND ('att2' = "good")`

ArrayLength

Get the length of an array.

Syntax

```
ArrayLength(a1)
```

Arguments

- **a1**
  a variable representing the array to operate on

Returns

The number of values in the array

Exceptions

None

Notes

None

Example

- `ArrayLength(Data)`
  [Return the number of values in an array named *Data*]

Ascii

Return the ASCII character code of the first character of a string.

Syntax

```
Ascii(s1)
```

Arguments

- **s1**
  Any expression evaluating to a string

Returns

The character code of the first character of the string

Exceptions

If the argument is not a string, an error value is returned

Example

- `Ascii("D")`
  [Returns 68, the ASCII character code for D]
- `Ascii("Program")`
  [Returns 80, the ASCII character code for the first letter of the string]

Asin

Return the inverse sine (arcsin) of a number. The inverse sine of *x* is the angle in radians whose sine is equal to *x*.

Syntax

```
Asin(x)
```

Arguments

- **x**
  Must be a real number between -1.0 and 1.0, inclusive

Returns

The inverse sine of *x*, in radians

Exceptions

If *x* is not a number, or is less than -1.0 or greater than 1.0, an error value is returned

Example

- `Asin(-0.5)`
  [Returns -0.5236]
- `Asin(TagVal('att1','y+8h'))`
  [Return the inverse sine of the value of 'att1' at 8am yesterday]
- `Asin('att1')`
  [Return the inverse sine of the value of 'att1' at trigger time]

Atn

Return the inverse (or arc) tangent of an integer or real number. The inverse tangent of *x* is the angle in radians whose tangent is equal to *x*.

Syntax

```
Atn(x)
```

Arguments

- **x**
  Must be an integer or real number

Returns

The inverse tangent of *x*, in radians

Exceptions

Returns an error if *x* is not an integer or real number

Example

- `Atn(1)`
  [Returns 0.7854]
- `Atn(-2.2)`
  [Returns -1.1442]
- `Atn('att1')`
  [Return the inverse tangent of the value of 'att1' at trigger time]
- `Atn('att1' - 'att2')`
  [Return the inverse tangent of the difference of 'att1' and 'att2' at trigger time]

Atn2

Return the inverse tangent (arctan) of a tangent value a/b. The inverse tangent is the angle measured in radians from the positive x-axis to a line whose endpoints are the origin and the Cartesian coordinates (b, a).

Syntax

```
    Atn2(x, y)
```

Arguments

- **x**

  An integer or real number
- **y**

  A non-zero integer or real number

Returns

The inverse tangent in radians of the tangent value x/y

Exceptions

Returns an error if *x* or *y* is not an integer or real number

Example

```
Atn2(1,2)
```

- [Returns 0.46365]

  ```
  Atn2('att1', 'att2')
  ```
- [Return the inverse tangent in radians of the tangent value 'att1'/'att2' at trigger time]

  ```
  Atn2(TagVal('att1','y+8h'), TagVal('att2', 'y+8h'))
  ```
- [Return the inverse tangent in radians of the tangent value 'att1'/'att2' at 8am yesterday]

Avg

Return the time-weighted or event-weighted average from a set of one or more values.

Syntax

```
Avg(x1 [, ... xn])
Avg(array [, calculationBasis])
```

Arguments

- **x1, ... xn**

  Arguments or a single array of same value type (integers and real numbers, time expressions, or time intervals)
- **calculationBasis**

  Optional. The type of calculation to be performed enclosed in double quotes. Choose between "EventWeighted" or "TimeWeighted." If omitted, the default is event-weighted. If you select "TimeWeighted," the earliest timestamp of the value marks the start time and the latest timestamp the end of a time range

Returns

The average of one or more values. The result is of the same data type as arguments

Exceptions

Arguments whose run-time values are character strings or digital states are not included in the average. If all values are character strings or digital states, Avg returns an error value. Returns an error if the array does not consist of same value type

Notes

- Unit of Measure (UOM) of the arguments is carried over to the result when at least one argument has a defined UOM while others don't. The returned value lacks a UOM if arguments have different UOMs

Example

- `Avg(14, 'att1', 14.5, TagVal('att2', '14-Dec-16'))`

  [Find the average of these values: 14, the current value of 'att1', 14.5, and the value for 'att2' at the start of day (12:00am) on Dec 14, 2016]
- `Avg('*', 'y+8h', 'Saturday')`

  [Find the average of these time stamps: now, 8:00am yesterday, and start of day (12:00am) last Saturday]
- `Avg('*'-'*-1h', 't+4h'-'y+8h', 'y+2h'-'20')`

  [Find the average of these time periods: from 1 hour ago to now, from 8:00am yesterday to 4:00am today, and 2:00am yesterday to the start of day (12:00am) on the 20th of this month]
- `Avg(Variable1)`

Name	Expression	Value at Evaluation
Variable1	`RecordedValues('att1', '*-10m', '*')`	[1, 3, 5, 7, 9]
Variable 2	`Avg(Variable1)`	5

  [Find the average of values in an array named Variable1]
- `Avg(Data)`

  [Return the event-weighted average of values for an array named *Data*]
- `Avg(Data, "EventWeighted")`

  [Return the event-weighted average of values for an array named *Data*]
- `Avg(Data, "TimeWeighted")`

  [Return the time-weighted average of values for an array named *Data* using the earliest timestamp of the value as the start time and the latest timestamp as the end time]

BadVal

Test a value to see if it is bad. For an attribute associated with a PI point, any system digital state value is considered bad (*No Data* or *Calc Failed*, for example).

Syntax

Badval(x)

Arguments

- **x**
  any expression evaluating to a value

Returns

True if the value is bad

False if the value is not bad

Exceptions

Returns *True* for blob PI points. Returns *False* for string PI points

Example

- `BadVal(1/0)`
  [Returns True]
- `BadVal(10)`
  [Returns False]
- `BadVal('att1')`
  [Returns True if the value of 'att1' is bad]
- `BadVal(FindEq('att1', 't', '*', 10))`
  [Returns True if the embedded function (**FindEq** in this example) has no result or has error for any reason]

Bod

Return a timestamp for beginning of the day from a time expression.

Syntax

`Bod(t1)`

Arguments

- **t1**
  A time expression, enclosed in single quotes

Returns

Timestamp for the start of the day

Exceptions

None

Notes

This function is useful for establishing a time at a unique clock time independent of the length of particular days.

Example

- `Bod('*')`
  [Return a timestamp for beginning of the day today]
- `Bod('y')`
  [Return a timestamp for beginning of the day yesterday]
- `Bod(FindEq('att1', '-3d', '*', 50))`
  [Return a timestamp for beginning of the day when 'att1' value was first equal to 50 in the past 72 hours]

Bom

Return a timestamp for midnight on the first day of the month from a given time expression.

Syntax

```
Bom(t1)
```

Arguments

- **t1**
  A time expression

Returns

Timestamp for the start of the month

Exceptions

None

Notes

This function is useful for establishing a time at a unique clock time independent of the length of particular days.

Example

- `Bom('*')`
  [Return a timestamp for midnight on the first day of this month]
- `Bom(PrevEvent('att1', '*'))`
  [Return a timestamp for midnight on the first day of the month when 'att1' had a value before the current one]
- `Bom(FindEq('att1', '-60d', '*', 50))`
  [Return a timestamp for midnight on the first day of the month when the value of 'att1' was first equal to 50 in the past 60 days]

Bonm

Return a timestamp for midnight on the first day of a following month from a given time expression.

Syntax

```
Bonm(t1)
```

Arguments

- **t1**
  Time expression, enclosed in single quotes

Returns

Timestamp for the start of the next month

Exceptions

None

Notes

This function is useful for establishing a time at a unique clock time independent of the length of particular days.

Example

- `Bonm('*')`
  [Return a timestamp for midnight on the first day of next month]
- `Bonm('y')`
  [Return a timestamp for midnight on the first day of the following month from yesterday's date]
- `Bonm(FindEq('att1', '-60d', '*', 50))`
  [Return a timestamp for midnight on the first day of the following month when the value of 'att1' was first equal to 50 in the past 60 days]

Ceiling

Return the nearest integer greater than or equal to *x*.

Syntax

```
Ceiling(x)
```

Arguments

- **x**
  integer or a real number, an attribute whose value evaluates to a number

Returns

The nearest integer greater than or equal to *x*

Exceptions

Returns an error if *x* is a digital state, time expression, time interval or a string

Notes

Unit of measure of the argument, if it exists, is carried over to the result

Example

- `Ceiling(2.3)`
  [Returns 3]
- `Ceiling(-2.3)`
  [Returns -2]
- `Ceiling(TagVal('att1', '12/30/16'))`

[Return the nearest integer to the value of 'att1' at the start of day (12:00am) on December 30, 2016]

Char

Build a string from ASCII character codes.

Syntax

```
Char(x1, ... xn)
```

Arguments

- **x1, ... xn**
  Integers

Returns

A string built from the 80 character codes

Exceptions

Returns an error if an argument is not a number

Example

- `Char(80, 73)`
  [Returns "PI"]
- `Char(65)`
  [Returns "A"]
- `Char(5 * 10)`
  [Returns "2"]

Compare

Compare two strings using wildcard characters ("*" and "?") or compare two operands using logical operators and deadband value.

Syntax

```
Compare {[(s1, s2 [, caseSensitive])]|[x1, x2, op, deadband]}
```

Arguments

- **s1, s2**
  string (s2 can contain wildcard characters)
- **caseSensitive**
  Optional flag indicating if the comparison is case sensitive. If False (the default), the comparison is not case-sensitive. If True, the comparison is case-sensitive
- **x1, x2**
  Numeric value (integer or real number) or time expression
- **op**
  An operator used for comparison. Must be one of the following operators, or a variable defined as one of these: <, >, <=, >=
- **deadband**
  Numeric value (integer or real number) or timespan. *Deadband* specifies a buffer (tolerance) around *x2* and prevents repeated alerts when the value fluctuates within the *deadband* range

Returns

- True if *s1* = *s2*
- True if *x1* <op> *x2* is true, and uses *deadband* value for subsequent evaluations

False otherwise

Exceptions

Wildcard characters in *s1* are treated literally and not as wildcards

The deadband value is applied to *x2*, not *x1*

Example

- `Compare('att1', 100, "<=", 5)`
  [Returns True if 'att1' is less than or equal to 100, and uses the deadband value of 5 for subsequent evaluations]
- `Compare("What", "what", True)`
  [Returns False]
- `Compare("b", "a")`
  [Returns False]
- `Compare("What", "wha?")`
  [Returns True]
- `Compare("What", "wh")`
  [Returns False]

Concat

Concatenate two or more strings.

Syntax

```
Concat(s1, s2 [, ... sn])
```

Arguments

- **s1, ... sn**
  Must be character strings, or expressions yielding character strings.

Returns

The character strings, concatenated together. This function does not insert blanks between its arguments. To include a space in the concatenated string, add an argument consisting of a string that has a single space enclosed in double quotes.

Example

- `Concat("shut", "down")`
  [Returns "shutdown"]
- `Concat("shut ", "down")`
  [Returns "shut down"]

Contains

Determine if first string contains second string and return True or False.

Syntax

```
Contains(s1, s2 [,caseSensitive])
```

Arguments

- **s1**
  Primary string
- **s2**
  Substring searched for in primary string
- **caseSensitive**
  Optional: Boolean that indicates whether the search is case sensitive. If False (or omitted), the search is not case sensitive. If True, the search is case sensitive

Returns

True if the primary string (*s1*) contains the substring (*s2*) and False otherwise

Exceptions

If either of the first two arguments (*s1* or *s2*) is not a string, the function returns an error value

If the optional third argument (*caseSensitive*) is not a Boolean, the function returns an error value

Example

- `Contains("What","Hat",True)`

  [Returns False]
- `Contains("What","Hat")`

  [Returns True]
- `Contains("What","at")`

  [Returns True]
- `Contains("hat","what")`

  [Returns False]

Convert

Convert a value from its current unit of measure (UOM) to a specified UOM; for a value with no UOM, assign the specified UOM.

Syntax

```
Convert(x, toUnit)
```

Arguments

- **x**

  Any expression that resolves to a numeric value, including the name of a numeric attribute enclosed in single quotation marks, a variable name, or a constant value
- **toUnit**

  The output UOM enclosed in double quotation marks

Returns

The numeric value converted to the specified UOM

Exceptions

Returns an error if *toUnit* and the initial UOM for *x* are not in the same UOM class

Example

- `Convert('MetricWeight', "lb")`
- `Convert('Ambient Temperature', "degC")`

  Note: PI AF substitutes *deg* with °, if not otherwise found in the lookup.

Cos

Return the trigonometric cosine of an integer or real number.

Syntax

`Cos(x)`

Arguments

- **x**

  Must be an integer or real number, which represents an angle in radians

Returns

The cosine of *x*

Exceptions

If *x* is not an integer or real number, returns an error value

Example

- `Cos(1.1)`

  [Returns 0.4536]
- `Cos(1)`

  [Returns 0.5403]
- `Cos('att1')`

  [Return the trigonometric cosine of the value of 'att1' at trigger time]

Cosh

Return the hyperbolic cosine of a number.

Syntax

```
Cosh(x)
```

Arguments

- **x**
  Must be an integer or real number

Returns

The hyperbolic cosine of *x*

Exceptions

If *x* is not an integer or real number, returns an error

Example

- `Cosh(1)`
  [Returns 1.5431]
- `Cosh(-1)`
  [Returns 1.5431]
- `Cosh('att1')`
  [Return the hyperbolic cosine of the value of 'att1' at trigger time]

Cot

Return the trigonometric cotangent of a number.

Syntax

```
Cot(x)
```

Arguments

- **x**
  Must be an integer or real number, which represents an angle in radians

Returns

The cotangent of *x*

Exceptions

If *x* is not a number, returns an error

Example

- `Cot(1)`
  [Returns 0.64209]
- `Cot(1.1)`
  [Returns 0.50897]
- `Cot('att1')`
  [Return the trigonometric cotangent of the value of 'att1' at trigger time]

Coth

Return the hyperbolic cotangent of a number.

Syntax

`Coth(x)`

Arguments

- **x**
  Must be an integer or real number

Returns

The hyperbolic cotangent of *x*

Exceptions

If *x* is not a number, returns an error value

Example

- `Coth(1)`
  [Returns 1.313]
- `Coth(1.1)`

[Returns 1.2492]

- `Coth('att1')`

  [Return the hyperbolic cotangent of the value of 'att1' at trigger time]

Cov

Determine the covariance between two sets of values given by attributes over a specified time range.

Syntax

```
Cov(x, y, mode, starttime, endtime, type [, pctgood])
```

Arguments

- **x**

  an attribute with the first set of time series data (such as PI point data reference) enclosed in single quotes
- **y**

  an attribute with the second set of time series data (such as PI point data reference) enclosed in single quotes
- **mode**

  a number that specifies how to align time-stamped values

  choose from 0, 1 or 2 where:

  0 represents the combination of time-stamped values from *x* and *y*

  1 represents values from both attributes according to *x*'s time stamps

  2 represents values from both attributes according to *y*'s time stamps
- **starttime**

  a time expression representing the beginning of a time range enclosed in single quotes; can be a relative time (such as '-3h') in reference to an absolute *endtime*
- **endtime**

  a time expression representing the end of a time range enclosed in single quotes; can be a relative time (such as '+1h') in reference to an absolute *starttime*
- **type**

  a number specifying the covariance type: use 0 for "population" and 1 for "sample"
- **pctgood**

  Optional. Minimum percentage of time during the time range that attribute values must be good.

  You can set *pctgood* as a threshold to ensure that there are sufficient good values to calculate **Cov**.

Returns

Covariance of *x* and *y*

Exceptions

If the attribute has no good values or the *pctgood* minimum is not reached for the given time range, returns an error value

Notes

Bad values are excluded from **Cov** calculation

Note: If the attribute has very few good values during the time range, this function's result may not be trustworthy. Use the **PctGood** function to find out what percentage of the values is good

Example

- `Cov('att1', 'att2', 0, 't', '+1h', 1, 80)`
  [Return the sample covariance of 'att1' and 'att2' between 12:00 and 1:00am today when at least 80% of the values were good. Return an error when minimum *pctgood* is not reached]
- `Cov('att1', 'att2', 1, 't', '+1h', 1)`
  [Return the population covariance of 'att1' and 'att2' based on time stamps of 'att1' between 12:00 and 1:00am today]

Csc

Return the trigonometric cosecant of a number.

Syntax

```
Csc(x)
```

Arguments

- **x**
  Must be an integer or real number, which represents an angle in radians

Returns

The cosecant of *x*

Exceptions

If *x* is not a number, returns an error

Example

- `Csc(1)`

[Returns 1.1884]
- `Csc(1.1)`
  [Returns 0.7487]
- `Csc('att1')`
  [Return the trigonometric cosecant of the value of 'att1' at trigger time]

Csch

Return the hyperbolic cosecant of a number.

Syntax

```
Csch(x)
```

Arguments

- **x**
  Must be an integer or real number

Returns

The hyperbolic cosecant of *x*

Exceptions

If *x* is not a number, returns an error value

Example

- `Csch(1)`
  [Returns 0.85092]
- `Csch(1.1)`
  [Returns 0.748]
- `Csch('att1')`
  [Return the hyperbolic cosecant of the value of 'att1' at trigger time]

Curve

For a given *x*, returns the value of a piecewise linear interpolation function defined by the set of n points (x_1,y_1)...(x_n,y_n).

Syntax

```
Curve(x, (x1,y1) (x2,y2) … (xn,yn))
```

Arguments

- **x**
  Expression evaluating to a number
- **x1, y1**
  The first point on the curve. The x_i's and y_i's are numeric constants evaluated at compile time. The values set for x_i's must be in ascending order.

Returns

For a given *x*, returns the value of a piecewise linear interpolation function defined by the set of n points (x_1,y_1)...(x_n,y_n). If the value of *x* is less than x_1 then y_1 is returned and if it is greater than x_n, y_n is returned. The points are assumed to be ordered in the *x* direction from smallest to largest.

Exceptions

If the value of *x* is not an integer or real number, an error value is returned

Example

- `Curve(3, (0,100) (100,0))`
  [Returns 97]
- `Curve('att1', (25,25) (75,75))`

Day

Extract the day of the month from a time expression.

Syntax

```
Day(t1)
```

Arguments

- **t1**
  time expression enclosed in single quotes

Returns

The day of the month of a time expression in the range 1 through 31

Exceptions

None

Example

- `Day('*')`
  [Return what day of the month today is]
- `Day('y')`
  [Return what day of the month yesterday was]
- `Day(FindEq('att1', '-28d', '*', 50))`
  [Return what day of the month it was when the value of 'att1' was first equal to 50 in the past 28 days]

DaySec

Return the total time in seconds between the start of day (midnight) and the time denoted in the argument.

Syntax

```
DaySec(t1)
```

Arguments

- **t1**
  A time expression, enclosed in single quotes

Returns

Total seconds since the start of day (midnight) till *t1*, in the range 0-86399

Exceptions

None

Example

- `DaySec('*')`
  [Return the number of seconds from the start of day (midnight) until now]

DeltaValue

Return the difference between the current and immediately previous values or *evaluations* of an attribute with a numeric or DateTime value type.

Syntax

```
DeltaValue(x [, prevEval])
```

Arguments

- **x**

  An attribute or expression of a numeric or DateTime value type
- **prevEval**

  (Boolean) If False or not specified, *x* must be an attribute, and **DeltaValue** returns the difference between its current value and immediately previous value.

  If True, **DeltaValue**returns the difference between the current and immediately previous *evaluations* of *x*, which can be any expression with a numeric or DateTime value type

Returns

Returns the difference between the current value and the immediately previous value of an attribute with a numeric or `DateTime` value type, or the difference between the current and immediately previous *evaluations* for any expression with a numeric or `DateTime` value type

Notes

When *x* is of DateTime data type, **DeltaValue** returns the time interval (in seconds) between two DateTime values. If the time interval value is output to an attribute, make sure the attribute data type is set to double

This function may use input values from PI points. If the compression is on for a PI point, all of its values may not be preserved in PI Data Archive. Before you try to validate the results of this function using a client tool such as PI Datalink, make sure the compression is turned off. This is to ensure that all associated PI point values are included for validation. When validation is complete, be sure to turn the compression back on again as a best practice

Unit of measure of the argument, if it exists, is carried over to the result

Example

- `DeltaValue('att1' [, FALSE])`

  [Return the difference between the current and immediately preceding value of 'att1']
- `DeltaValue('att1', TRUE)`

  [Return the difference between the current and last *evaluated* value of 'att1']

DigState

Convert a string into a corresponding Data Archive digital state object, either based on the attribute's digital state enumeration or on a system enumeration set values (for example, *Calc Failed*). Use **DigState** embedded in other functions when Data Archive digital state object is required for input (such as with **StateNo**) or in conjunction with other functions for comparison with digital state attributes.

Syntax

```
DigState(s1 [, x])
```

Arguments

- **s1**
  A string representing a digital state in double quotes
- **x**
  Optional: An attribute with PI point digital state reference or of value type enumeration set. If omitted, only the system digital set is searched for the given string.

Returns

An enumeration value

Exceptions

If the string does not represent a digital state of the specified attribute, the function returns *Calc Failed*. If the attribute is omitted and string does not represent a system digital state, *Calc Failed* is returned.

Example

- `'att1' > DigState("Program", 'att1')`
  [Returns True if current digital state of 'att1' is greater than the digital state represented by "Program" value]
- `DigState("No Result")`
  [Construct a value with system digital state *No Result*]

DigText

Obtain the text corresponding to the digital state.

Syntax

```
DigText(digstate)
```

Arguments

- **digstate**
  An argument that evaluates to a digital state

Returns

The text for the digital state

Exceptions

Returns an error if the argument is not a digital state

Example

- `DigText(TagVal('enum_att1', 'y'))`
  [Returns the digital state value in string]
- `DigText('enum_att1')`
  [Returns digital state value in string. Returns an error message if 'enum_att1' is not a PI point attribute with digital state value]

E

Return the value for *e* (the base of the natural logarithm).

Syntax

```
E()
```

Arguments

None

Returns

The value for *e* (the base of the natural logarithm)

Exceptions

None

Example

- `E()`

ELSE

Operator that returns the second of two specified values when the conditional expression in IF-THEN-ELSE statement is True.

Syntax

```
IF (expression) THEN (x) ELSE (y)
```

Arguments

- **expression**
  Any expression that evaluates to true or false
- **x, y**
  An expression that evaluates to an output value

Returns

x when the conditional expression is true and *y* when the expression is false

Exceptions

None

Example

- `IF ('att1' > 50) THEN ('att2') ELSE ('att3')`
  [If the value of 'att1' is greater than 50, then return the value of 'att2'; otherwise return the value of 'att3']

EventCount

Find the number of events for an attribute during a specified time range.

Syntax

```
EventCount(attname, starttime, endtime [, pctgood])
```

Arguments

- **attname**
  attribute with time series data (such as PI point data reference) enclosed in single quotes
- **starttime**
  time expression representing the beginning of a time range enclosed in single quotes; can be a relative time

(such as '-3h') in reference to an absolute *endtime*

- **endtime**

  time expression representing the end of a time range enclosed in single quotes; can be a relative time (such as '+1h') in reference to an absolute *starttime*
- **pctgood**

  Optional: Minimum percentage of good values for events within the specified time interval.

Returns

Number of events for the attribute during a specified time range.

Exceptions

Returns an error if the *pctgood* minimum is not reached for the given time interval.

Notes

If the attribute has very few good values during the time range, this function's result may not be trustworthy. Use the **PctGood** function to find out what percentage of the values is good.

Example

- `EventCount('att1', 't', '+1h')`

  [Return the count of events between 12:00 and 1:00am today.]
- `EventCount('att1', 't', '+1h', 80)`

  [Return the count of events between 12:00 and 1:00am today when at least 80% of the values were good.]

EventFrame

Return the value of event frame properties.

Syntax

```
EventFrame(parameter)
```

Arguments

- **parameter**

  parameter string *StartTime*, *EndTime* or *Duration* enclosed in double quotes; not case-sensitive

Returns

The value of the parameter

Exceptions

If the event frame is not active or still open, returns *No Results*

Notes

- This function is only to be used when configuring output expressions in an event frame generation analysis
- The output can be mapped to an attribute with value type DateTime for *StartTime* and *EndTime*. For *Duration*, select double (time returned in seconds)
- You can preview results by right-clicking the analysis

Example

- `EventFrame("Duration")`

  [Return the duration of an event frame. Time is returned in seconds]
- `TagAvg('att1', EventFrame("StartTime"), EventFrame("EndTime"))`

  [Find the time-weighted average of the values of 'att1' during an event frame]

Exit

Stop the analysis from running further evaluation.

Syntax

```
Exit()
```

Arguments

None

Notes

- It is important to include the parentheses after this function. Use Exit() instead of Exit.
- Suppose you have multiple expressions within a single analysis or a sequence of analyses that forms a dependency chain. The calculation ends when it hits Exit.
- Unlike Exit, which immediately exits out of the analysis, `NoOutput` will continue until it goes through every expression in an analysis.

Example

- `IF 'att1' < 100 OR 'att1' > 200 THEN 'att1' ELSE Exit()`

Exp

Return the exponential of an integer or real number. This is the number e^x, where e = 2.7182818...

Syntax

```
Exp(x)
```

Arguments

- **x**
  Must be an integer or real number

Returns

The exponential of *x*

Exceptions

If *x* is not an integer or real number, returns an error value

Example

- `Exp(11)`
  [Returns 59874]
- `Exp('att1')`
  [Return the exponential of the value of 'att1' at trigger time]
- `Exp(TagVal('att1','t'))`
  [Return the exponential of the value of 'att1' at the start of day (12am) today]

FilterData

Get all the values in an array that satisfy a given condition.

Syntax

```
FilterData(a1, condition($val))
```

Arguments

- **a1**
  a variable representing the array to operate on

- **condition($val)**
  any supported PE statement, as a function of *$val*, that returns a true or false. *$val* is a placeholder for each value in the array

Returns

An array of all the values that satisfy the given condition

Exceptions

None

Notes

None

Example

- `FilterData(Data, (NOT BadVal($val) AND $val > 50))`
  [From an array named *Data*, return a new array of values where none are bad values and all are greater than 50]

FindEq

Find the timestamp closest to *starttime* within a specified time interval when the attribute is equal to a specified value.

Syntax

```
FindEq(attname, starttime, endtime, x)
```

Arguments

- **attname**
  name of an attribute enclosed in single quotes
- **starttime**
  time expression representing the beginning of a time range enclosed in single quotes; can be a relative time (such as "-3h") in reference to an absolute *endtime*
- **endtime**
  time expression representing the end of a time range enclosed in single quotes; can be a relative time (such as "+1h") in reference to an absolute *starttime*
- **x**
  The value to search for; must be an integer or real number or digital state (character string)

Returns

The timestamp closest to *starttime* within the specified interval when the attribute was equal to the specified value

Exceptions

If the attribute was never equal to the specified value, an error value is returned

Notes

- If needed, interpolation is done between attribute values to determine when the attribute was equal to the specified value
- Interpolation occurs when:
  - the underlying PI point has step attribute turned off
  - there is no recorded value with the specified timestamp
- The interpolated value will be based on the values before and after the timestamp
- No value will be interpolated if the step attribute is turned on. In this case, the value of record will be that at the time of the event (if it exists) or the previously-recorded value
- If *endtime* is earlier than *starttime*, the time interval is searched backwards

Example

- `FindEq('att1', '-1h', '*', 40)`

  [Return the timestamp of the first time since an hour ago (*starttime*) when ' att1' was equal to 40]
- `FindEq('enum_att1', '30-March-17', '*', "On")`

  [Return the timestamp of the first time since the start of day (12am; *starttime*) on March 30th, 2017 when enumeration value 'enum_att1' was equal to "On"]

FindGE

Find the first time within a time interval when an attribute is greater than or equal to a specified value.

Syntax

```
FindGE(attname, starttime, endtime, x)
```

Arguments

- **attname**

  The name of an attribute enclosed in single quotation marks
- **starttime**

  time expression representing the beginning of a time range enclosed in single quotes; can be a relative time

(such as "-3h") in reference to an absolute *endtime*

- **endtime**

  time expression representing the end of a time range enclosed in single quotes; can be a relative time (such as "+1h") in reference to an absolute *starttime*
- **x**

  The value to search for; must be an integer or real number or digital state (character string)

Returns

The timestamp closest to *starttime* within the specified interval when the attribute was greater than or equal to the specified value. Returns the corresponding system digital state (*No Result*, *No Data*, and the like) if the attribute was always less than the specified value.

Exceptions

None

Notes

- If needed, interpolation is done between attribute values to determine when the attribute was greater than or equal to the specified value
- Interpolation occurs when:
  - the underlying PI point has step attribute turned off
  - there is no recorded value with the specified timestamp
- The interpolated value will be based on the values before and after the timestamp
- No value will be interpolated if the step attribute is turned on. In this case, the value of record will be that at the time of the event (if it exists) or the previously-recorded value
- If *endtime* is earlier than *starttime*, the time interval is searched backwards

Example

- `FindGE('att1', 'y', '+2h', 40)`

  [Return the timestamp between midnight and 2:00 am yesterday when 'att1' was first greater than or equal to 40]

FindGT

Find the first time within a time interval when an attribute is greater than a specified value.

Syntax

```
FindGT(attname, starttime, endtime, x)
```

Arguments

- **attname**

  The name of an attribute enclosed in single quotation marks
- **starttime**

  time expression representing the beginning of a time range enclosed in single quotes; can be a relative time (such as "-3h") in reference to an absolute *endtime*
- **endtime**

  time expression representing the end of a time range enclosed in single quotes; can be a relative time (such as "+1h") in reference to an absolute *starttime*
- **x**

  The value to search for; must be an integer or real number or digital state (character string)

Returns

The timestamp closest to *starttime* within the specified interval when the attribute was greater than the specified value. Returns the corresponding system digital state (*No Result*, *No Data*, and the like) if the attribute was never greater than the specified value.

Exceptions

None

Notes

- If needed, interpolation is done between attribute values to determine when the attribute was greater than the specified value
- Interpolation occurs when:
  - the underlying PI point has step attribute turned off
  - there is no recorded value with the specified timestamp
- The interpolated value will be based on the values before and after the timestamp
- No value will be interpolated if the step attribute is turned on. In this case, the value of record will be that at the time of the event (if it exists) or the previously-recorded value
- If *endtime* is earlier than *starttime*, the time interval is searched backwards

Example

- `FindGT('att1', 't', '*', 40)`

  [Return the timestamp of the first time after midnight today when 'att1' was greater than 40]
- `FindGT('att1', '-1h', '*', TagVal('att1', 'y+18h'))`

  [Return the timestamp of the first time since an hour ago when the value of 'att1' was greater than its value at 6pm yesterday]

FindLE

Find the first time within a time interval when an attribute is less than or equal to a specified value.

Syntax

```
FindLE(attname, starttime, endtime, x)
```

Arguments

- **attname**

  The name of an attribute enclosed in single quotation marks
- **starttime**

  time expression representing the beginning of a time range enclosed in single quotes; can be a relative time (such as "-3h") in reference to an absolute *endtime*
- **endtime**

  time expression representing the end of a time range enclosed in single quotes; can be a relative time (such as "+1h") in reference to an absolute *starttime*
- **x**

  The value to search for; must be an integer or real number or digital state (character string)

Returns

The timestamp closest to *starttime* within the specified interval when the attribute was less than or equal to the specified value. Returns the corresponding system digital state (*No Result*, *No Data*, and the like) if the attribute was always greater than the specified value.

Exceptions

None

Notes

- If needed, interpolation is done between attribute values to determine when the attribute was equal to the specified value
- Interpolation occurs when:
  - the underlying PI point has step attribute turned off
  - there is no recorded value with the specified timestamp
- The interpolated value will be based on the values before and after the timestamp
- No value will be interpolated if the step attribute is turned on. In this case, the value of record will be that at the time of the event (if it exists) or the previously-recorded value
- If *endtime* is earlier than *starttime*, the time interval is searched backwards

Example

- `FindLE('att1', 't', '*', 40)`
  [Return the timestamp of the first time after midnight today when *att1* was less than or equal to 40]

FindLT

Find the first time within a time interval when an attribute is less than a specified value.

Syntax

```
FindLT(attname, starttime, endtime, x)
```

Arguments

- **attname**
  The name of an attribute with a PI point data reference, enclosed in single quotation marks
- **starttime**
  time expression representing the beginning of a time range enclosed in single quotes; can be a relative time (such as "-3h") in reference to an absolute *endtime*
- **endtime**
  time expression representing the end of a time range enclosed in single quotes; can be a relative time (such as "+1h") in reference to an absolute *starttime*
- **x**
  Must be an integer or real number or digital state (character string), the value to search for

Returns

The timestamp closest to *starttime* within the specified interval when the attribute was less than the specified value. Returns the corresponding system digital state (*No Result*, *No Data*, and the like) if the attribute was never less than the specified value.

Exceptions

None

Notes

- If needed, interpolation is done between attribute values to determine when the attribute was lower than the specified value
- Interpolation occurs when:
  - the underlying PI point has step attribute turned off
  - there is no recorded value with the specified timestamp

- The interpolated value will be based on the values before and after the timestamp
- No value will be interpolated if the step attribute is turned on. In this case, the value of record will be that at the time of the event (if it exists) or the previously-recorded value
- If *endtime* is earlier than *starttime*, the time interval is searched backwards

Example

- `FindLT('att1', 'y', 't', 40)`

  [Return the timestamp of the first time between midnight yesterday and midnight today that 'att1' was less than 40]
- `FindLT('att1', '-1h', '*', TagVal('att1', 'y+18h'))`

  [Return the timestamp of the first time since an hour ago when the value of 'att1' was less than its value at 6pm yesterday]

FindNE

Find the first time within a time interval when an attribute is not equal to a specified value. "No Data" events will be skipped.

Syntax

```
FindNE(attname, starttime, endtime, x)
```

Arguments

- **attname**

  The name of an attribute enclosed in single quotation marks
- **starttime**

  time expression representing the beginning of a time range enclosed in single quotes; can be a relative time (such as "-3h") in reference to an absolute *endtime*
- **endtime**

  time expression representing the end of a time range enclosed in single quotes; can be a relative time (such as "+1h") in reference to an absolute *starttime*
- **x**

  The value to search for; must be an integer or real number or digital state (character string)

Returns

The timestamp closest to *starttime*, within the specified interval, for which the attribute was not equal to the specified value. "No Data" events will be skipped

Exceptions

If the attribute was always equal to the specified value, an error value is returned

Notes

- If needed, interpolation is done between attribute values to determine when the attribute was not equal to the specified value
- Interpolation occurs when:
  - the underlying PI point has step attribute turned off
  - there is no recorded value with the specified timestamp
- The interpolated value will be based on the values before and after the timestamp
- No value will be interpolated if the step attribute is turned on. In this case, the value of record will be that at the time of the event (if it exists) or the previously-recorded value
- If *endtime* is earlier than *starttime*, the time interval is searched backwards

Example

- `FindNE('att1', '-1h', '*', 40)`

  [Return the timestamp of the first time since an hour ago when ' att1' was not equal to 40]
- `FindNE('enum_att1', '30-March-17', '*', "On")`

  [Return the timestamp of the first time since the start of day (12am) on March 30th, 2017 when enumeration value 'enum_att1' was not equal to "On"]

FirstValue

Get the first value in an array that satisfies a given condition.

Syntax

```
FirstValue(a1 [, condition($val)])
```

Arguments

- **a1**

  a variable representing the array to operate on
- **condition($val)**

  any supported PE statement, as a function of *$val*, that returns a true or false. *$val* is a placeholder for each value in the array

Returns

The first value that satisfies the given condition

Exceptions

None

Notes

None

Example

- FirstValue(Data)
  [Return the first value from an array named *Data*]
- `FirstVal(Data, $val < 90)`
  [From the array named *Data*, return the first value in the array that is less than 90]

Float

Convert a string to a number.

Syntax

```
Float(x)
```

Arguments

- **x**
  A string or a number, or an attribute whose value evaluates to a number at time of evaluation

Returns

A number for a numeric string. If *x* is already a number, *x* is returned

Exceptions

If *x* is not a number or a numeric string, returns *Calc Failed*

Notes

Unit of measure of the argument, if it exists, is carried over to the result. **Float** also takes *timespan* and *boolean* as argument. Note, however, that **Float** only converts *timespan* format to the number of seconds from 12:00am Jan 1, 1970.

Example

- `Float("12.3")`
  [Converts string to a number and returns 12.3]
- `Float(12.3)`
  [Returns 12.3]
- `Float('*')`
  [Return the total number of seconds passed since Jan 1, 1970]

Floor

Return the nearest integer less than or equal to *x*.

Syntax

```
Floor(x)
```

Arguments

- **x**
  Integer or a real number, or an attribute whose value evaluates to a number at time of evaluation

Returns

The nearest integer less than or equal to *x*

Exceptions

Returns an error if *x* is a digital state, time expression, time interval or a string

Notes

Unit of measure of the argument, if it exists, is carried over to the result

Example

- `Floor(3.14)`
  [Returns 3]
- `Floor(-3.14)`
  [Returns -4]
- `Floor(TagVal('att1', '*'))`
  [Return the nearest integer less than or equal to the current value of 'att1']

Format

Convert a number to string according to a format expression.

Syntax

```
Format(x, format [,culture])
```

Arguments

- **x**
  An integer or a real number
- **format**
  Format-control string, similar to that used by the C language function **Sprintf**
- **culture**
  Optional. Culture name string, enclosed in double quotation marks. Example values include "de-DE", "ja-jp", "pt-br", "zh-cn". Use "" to specify the culture-invariant setting (the default setting).

Returns

A formatted string

Example

- `Format(1234.567, "%10.2f", "de-DE")`
  [Returns "1234,57", a formatted string]
- `Format(45, "%3.3d")`
  [Returns "045"]

Frac

Return the fractional part of a real number. Returns 0 for integers.

Syntax

```
Frac(x)
```

Arguments

- **x**
  Must be an integer or real number

Returns

The fractional part of *x*

Exceptions

If *x* is not an integer or real number, returns an error value

Notes

By definition: *Int(x) + Frac(x) = x*

Unit of measure of the argument, if it exists, is carried over to the result

Example

- `Frac(1)`
  [Returns 0]
- `Frac(1.3)`
  [Returns 0.3]
- `Frac(TagVal('att1', '*'))`
  [Return the fractional part of the value of 'att1' at current time.]

HasChanged

Return True if an attribute has had any event in the specified time period; otherwise return False.

Syntax

```
HasChanged(attname, t1)
```

Arguments

- **attname**
  The name of an attribute enclosed in single quotation marks
- **t1**
  The start of a time period ending at execution time; can be any relative time expression in single quotation marks (such as 't - 4h')

Returns

True if *attname* has any event in the specified time period; otherwise returns False

Example

- `HasChanged('att1', 't-4h')`
  [Returns True if 'att1' received any updates since 8 p.m. last night, regardless of whether the update changed the actual attribute value]

HasValueChanged

Determine if the value of an attribute or expression has changed since it was last evaluated during an analysis.

Syntax

```
HasValueChanged(x)
```

Arguments

- **x**
  attribute or expression

Returns

True if the value of *x* has changed since last evaluated during the analysis; otherwise returns False.

If *x* is an expression that this function has not yet evaluated during the analysis, then the function returns False.

If *x* is an attribute that this function has not yet evaluated during the analysis, then the function finds the previous recorded value and compares the current value to that previous value. If there is no previous value, then the function returns False.

Notes

`HasValueChanged` is a "stateful" function. Previous evaluations are stored in memory and compared against the new value to see if there has been a change in state. If PI Analysis Service restarts during an analysis, memory is flushed and any previous evaluations are lost.

Example

- `HasValueChanged('att1')`
- [Returns True if the value of *att1* has changed since last evaluated during the analysis]
- `HasValueChanged('att1'+'att2')`
  [Returns True if the value of sum of *att1* and *att2* has changed since last evaluated during the analysis]

Hour

Extract the hour from a time expression.

Syntax

```
Hour(t1)
```

Arguments

- **t1**
  A time expression, enclosed in single quotes

Returns

The hour of time, in the range 0-23

Exceptions

None

Example

- `Hour('*')`
  [Return the hour portion of current time]
- `Hour('Saturday')`
  [Returns 0]
- `Hour(FindEq('att1', '-1d', '*', 50))`
  [Return the hour of the time when the value of 'att1' was first equal to 50 in the past 24 hours]

IF

Operator that introduces the condition in *IF-THEN-ELSE* statement. Return the first of two specified values if the conditional expression is true. Otherwise, return the second value.

Syntax

```
IF (expression) THEN (x) ELSE (y)
```

Arguments

- **expression**
  Any expression that evaluates to true or false
- **x, y**
  An expression that evaluates to an output value

Returns

x when the conditional expression is true and *y* when the expression is false

Exceptions

None

Example

- `IF ('att1' > 50) THEN ('att2') ELSE ('att3')`
  [If the value of 'att1' is greater than 50, then return 'the value of 'att2'; otherwise return the value of 'att3']

IN

Return *True* if a value is included in a set or a range of inclusive values.

Syntax

```
x In (y1 [, ... yn])
x In (y1...yn)
```

Arguments

- **x**
  numeric value or string
- **y1 [, ... yn]**
  a set of numeric or string values
- **y1 ... yn**
  a range of numeric values

Returns

True when *x* is included in a set or a range of values

Exceptions

None

Notes

If y_1 is a decimal number that defines a range (as in the second syntax), use 2 dots instead of 3 between y_1 and y_n.

Example

- `1 In (9, 10)`
  [Returns False since 1 is not included in a set of 9 and 10]
- `2 In (1...10)`
  [Returns True since 2 is included in a range of 1 to 10]
- `2 In (1.5..10)`
- [Returns True since 2 is included in a range of 1.5 to 10]

InStr

Return the location within a string where a sub-string match is first found.

Syntax

```
InStr([start,] s1, s2 [,caseSensitive])
```

Arguments

- **start**
  Optional: An integer specifying which character in *s1* to start the comparison. Must be larger than or equal to 1.
- **s1, s2**
  Two strings to be compared.
- **caseSensitive**
  Optional: A flag indicating if the comparison is case-sensitive. If 0 (the default) the comparison is case-insensitive, if 1, the comparison is case-sensitive.

Returns

0 if *s2* is not a sub-string of *s1* starting from the start position; otherwise, the location of character where *s2* first matches the characters in *s1* from the start position.

Exceptions

Wildcard characters are not treated as wildcards.

Example

- `InStr("What", "At")`
  [Returns 3]
- `InStr("What What What", "What")`

[Returns 1]

- `InStr("what", "At", 1)`

  [Returns 0]
- `InStr(4, "what", "At")`

  [Returns 0]
- `InStr('att1', "Error")`

  [Returns 1 if the value of *att1* is "Error"]

Int

Return the integer part of an integer or real number.

Syntax

```
Int(x)
```

Arguments

- **x**

  Number, Boolean, numeric string, timespan, or an attribute whose value evaluates to a number at time of evaluation

Returns

If *x* is a number, the integer part of *x* is returned. If *x* is a string, it is first converted into a number. If *x* is a *timespan*, the number of seconds from 12:00am January 1, 1970 is returned

Exceptions

Returns error if the variable is not assigned

Notes

Int('')* returns UTC time in seconds from 12:00am January 1, 1970

Example

- `Int('att1')`

  [Return the integer part of the value of 'att1' at current time]
- `Int('*'-'t')`

  [Return how many seconds have passed since the start of day today]
- `Int(2.1)`

[Returns 2]

- `Int("2.1")`

  [First converts the string to a number and returns 2]
- `Int(true)`

  [Returns 1]

InterpolatedValues

Obtain interpolated values over a specified time range using the specified time step.

Syntax

```
InterpolatedValues(attname, starttime, endtime, timestep)
```

Arguments

- **attname**

  attribute of time series data (such as PI point data reference) enclosed in single quotes
- **starttime**

  a time expression representing the beginning of a time range enclosed in single quotes; can be a relative time (such as '-3h') in reference to an absolute endtime
- **endtime**

  a time expression representing the end of a time range enclosed in single quotes; can be a relative time (such as '+1h') in reference to an absolute starttime
- **timestep**

  a time interval enclosed in single quotes (such as '+1m' or '-10s') representing the incremental change in time between the specified range to obtain the interpolated values

Returns

An array of the values obtained within the specified range in intervals of the time step

Exceptions

If the attribute does not support range calls or interpolated values of range calls, the function returns an error indicating as such

Notes

- If the *starttime* is earlier than the *endtime*, the resulting values will be in time-ascending order, otherwise they will be in time-descending order.
- When a positive timestep is specified, the interval calculation begins at the earliest bounding time in the time range and applies the interval repeatedly in the time-ascending direction to generate the calculation

intervals.

- If a negative timestep is specified, the interval calculation begins at the latest bounding time in the time range and applies the interval repeatedly in the time-descending direction to generate the calculation intervals.
- Note that the order of values returned will still be reflected by the time range, regardless of the timestep sign.

Example

- `InterpolatedValues('att1', 't', '+1h', '+10m')`

  [Return the values of 'att1' between 12:00 and 1:00am today in 10-minute intervals]
- `InterpolatedValues('att1', 't', '+1h', '-14m')`

  [Return the values of 'att1' between 12:00 and 1:00am today in 14-minute intervals, anchoring from 1:00am (that is, 12:04,12:18,12:32,12:46,1:00)]

IsSet

For an attribute associated with a PI point, determine if the point is annotated, substituted, or questionable.

Syntax

```
IsSet(attname, select)
```

Arguments

- **attname**

  An attribute associated with a PI point of value type integer, real number, enumeration value, or string
- **select**

  A string but only the first character is considered: "a" for annotate, "s" for substituted and "q" for questionable. Not case-sensitive.

Returns

True or False

Exceptions

None

Example

- `IsSet('att1', "a")`

  [Returns True if the value of 'att1' is annotated]

- `IsSet('att1', "s")`
  [Returns True if the value of 'att1' is substituted]]
- `IsSet('att1', "q")`
  [Returns True if the value of 'att1' is questionable]

LastValue

Get the last value in an array that satisfies a given condition.

Syntax

```
LastValue(a1 [, condition($val)])
```

Arguments

- **a1**
  a variable representing the array to operate on
- **condition($val)**
  any supported PE statement, as a function of *$val*, that returns a true or false. *$val* is a placeholder for each value in the array

Returns

The last value that satisfies the given condition

Exceptions

None

Notes

None

Example

- `LastValue(Data)`
  [Return the last value from an array named *Data*]
- `LastValue(Data, $val > 90 and $val < 105)`
  [From the array named *Data*, return the last value in the array that is greater than 90 and less than 105]

LCase

Convert a string to a lowercase string.

Syntax

```
LCase(s1)
```

Arguments

- **s1**
  string

Returns

A string that has been converted to lowercase

Exceptions

If the argument is not a string, returns an error value

Example

- `LCase("String")`
  [Returns "string"]

Left

Return a specified number of characters of a string from the left.

Syntax

```
Left(s1, len)
```

Arguments

- **s1**
  String
- **len**
  Integer

Returns

The first **len** characters of the string, starting from the left

Exceptions

If the arguments are not of the required types, returns an error

Example

- `Left("String_att", 3)`
  [Returns "Str"]

Len

Return the length of a string.

Syntax

```
Len(s1)
```

Arguments

- **s1**
  string

Returns

The length of a string

Exceptions

If the argument is not a string, returns an error value

Example

- `Len("String")`
  [Returns 6]
- `Len('sinusoid')`
  [Returns *Calc Failed*]

LinRegr

Apply linear least squares fitting to two sets of values given by attributes over a specified time range. The output is a one-based array with the values of the slope, intercept and R^2.

Syntax

```
LinRegr(x, y, mode, starttime, endtime [, pctgood])
LinRegr(y, starttime, endtime [, pctgood])
```

Arguments

- **x**

  an attribute with the first set of time series data (such as PI point data reference) enclosed in single quotes
- **y**

  an attribute with the second set of time series data (such as PI point data reference) enclosed in single quotes
- **mode**

  a number that specifies how to align time-stamped values
- **choose from 0, 1 or 2 where:**

  0 represents the combination of time-stamped values from *x* and *y*

  1 represents values from both attributes according to *x*'s time stamps

  2 represents values from both attributes according to *y*'s time stamps
- **starttime**

  a time expression representing the beginning of a time range enclosed in single quotes; can be a relative time (such as '-3h') in reference to an absolute *endtime*
- **endtime**

  a time expression representing the end of a time range enclosed in single quotes; can be a relative time (such as '+1h') in reference to an absolute *starttime*
- **pctgood**

  Optional. Minimum percentage of attribute values during the time range that must be good.

  You can set *pctgood* as a threshold to ensure that there are sufficient good values to calculate **LinRegr**.

Returns

One-based array with the values of slope, intercept and R^2.

Exceptions

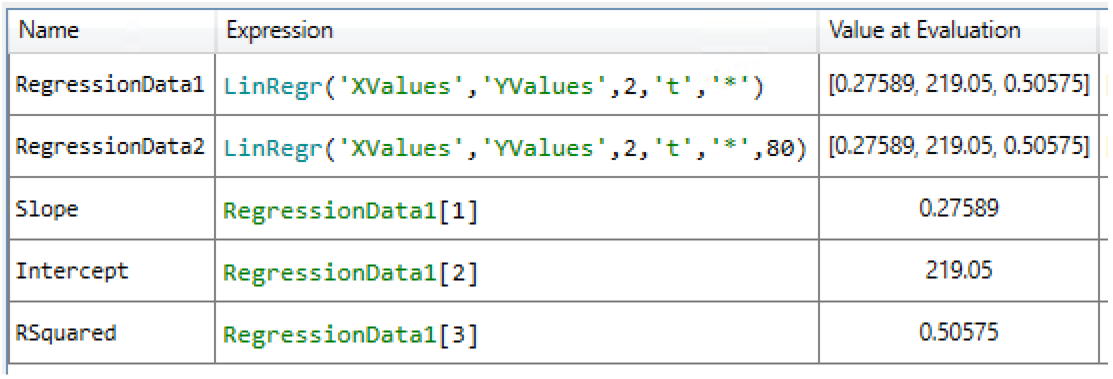
If the attribute has no good values or the *pctgood* minimum is not reached for the given time range, returns an error value

Notes

Bad values are excluded from **LinRegr** calculation.

For *LinRegr(y, starttime, endtime [, pctgood])*, the unit of the output slope is *unit of y/second*.

Note: If the attribute has very few good values during the time range, this function's result may not be trustworthy. Use the **PctGood** function to find out what percentage of the values is good.



Name	Expression	Value at Evaluation
RegressionData1	LinRegr('XValues','YValues',2,'t','*')	[0.27589, 219.05, 0.50575]
RegressionData2	LinRegr('XValues','YValues',2,'t','*',80)	[0.27589, 219.05, 0.50575]
Slope	RegressionData1[1]	0.27589
Intercept	RegressionData1[2]	219.05
RSquared	RegressionData1[3]	0.50575

Index the slope, intercept and R^2 outputs from [1].

Example

- `LinRegr('att1', 'att2', 0, 't', '+1h', 80)`

  [Return a one-based array of slope, intercept and R^2 based on time-stamped values of 'att1' and 'att2' between 12:00 and 1:00am today when at least 80% of the values were good. Return an error when minimum *pctgood* is not reached]
- `LinRegr('att1', 'att2', 1, 't', '+1h')`

  [Return a one-based array of slope, intercept and R^2 based on time-stamped values of 'att1' between 12:00 and 1:00am today]
- `LinRegr('att1', 't', '+1h', 80)`

  [Return a one-based array of slope, intercept and R^2 of 'att1' between 12:00 and 1:00am today when at least 80% of the values were good. Return an error when minimum *pctgood* is not reached]
- `LinRegr('att1', 't', '+1h')`

  [Return a one-based array of slope, intercept and R^2 of 'att1' between 12:00 and 1:00am today]

Log

Return the natural logarithm (base e = 2.7182818...) of an integer or real number.

Syntax

```
Log(x)
```

Arguments

- **x**

  Must be an integer or real number greater than zero

Returns

The natural logarithm of *x*

Exceptions

If *x* is zero or negative, or not a number, returns an error value

Example

- `Log(14)`
  [Returns 2.6391]
- `Log(TagVal('att1', '14-Dec-16'))`
  [Return the natural log of the value of 'att1' at 12:00am on Dec 14, 2016]

Log10

Return the base 10 logarithm of an integer or real number.

Syntax

```
Log10(x)
```

Arguments

- **x**
  Must be an integer or real number greater than zero

Returns

The base 10 logarithm of *x*

Exceptions/Errors

If *x* is zero or negative, or not a number, returns an error value

Example

- `Log10(100)`
  [Returns 2]
- `Log10(TagVal('att1', '14-Dec-16'))`
  [Return the base 10 logarithm of the value of 'att1' at 12:00am on Dec 14, 2016]

Logbase

Return the logarithm of positive numeric value *x* to a specified base *y*.

Syntax

```
Logbase(x,y)
```

Arguments

- **x**
  An integer or real number greater than zero
- **y**
  A positive integer indicating the base of the logarithm

Returns

The logarithm of *x* to base *y*

Exceptions

If *x* is zero or negative, or not a number, returns an error value

Example

- `Logbase(256, 2)`
  [Returns 8]
- `Logbase(1000, 10)`
  [Returns 3]
- `Logbase(TagVal('att1', '14-Dec-16'), 16)`
  [Return the logarithm of the value of 'att1' at 12:00am on Dec 14, 2016 to base 16]

LTrim

Remove the leading blanks from a string.

Syntax

```
LTrim(s1)
```

Arguments

- **s1**

string

Returns

A string with leading blanks removed

Exceptions

If *s1* is not a string, an error value is returned.

Example

- `LTrim(" String")`
  [Returns "String"]
- `LTrim("String ")`
  [Returns "String "]

MapData

Apply a function to each value in an array.

Syntax

```
MapData(a1, function($val))
```

Arguments

- **a1**
  a variable representing the array to operate on
- **function($val)**
  any supported PE statement, as a function of *$val*, that performs an operation. *$val* is a placeholder for each value in the array

Returns

The transformed array of all the values

Exceptions

None

Notes

The output array has the same timestamps as the input array.

Example

- `MapData(Data, (IF BadVal($val) THEN 0 ELSE $val))`
  [From an array named *Data*, return a new array with bad values set to 0]

Max

Return the time-weighted or event-weighted maximum from a set of values.

Syntax

```
Max(x1, ... xn)
Max(array [, calculationBasis])
```

Arguments

- **x1, ... xn**
  Arguments or a single array of same value type (integers and real numbers, enumeration values, time expressions, or time intervals)
- **calculationBasis**
  Optional. The type of calculation to be performed enclosed in double quotes. Choose between "EventWeighted" or "TimeWeighted." If omitted, the default is event-weighted. If you select "TimeWeighted," the earliest timestamp of the value marks the start time and the latest timestamp the end of a time range

Returns

The maximum from a set of arguments. The result is of the same data type as the arguments

Exceptions

Arguments whose run-time values are digital states are ignored. If all values are digital states, **Max** returns an error. Returns an error if the array does not consist of same value type

Notes

- Unit of Measure (UOM) of the arguments is carried over to the result when at least one argument has a defined UOM while others don't. The returned value lacks a UOM if arguments have different UOMs

Example

- `Max(14, 'att1', 14.5, TagVal('att2','14-Dec-16'))`

  [Find the maximum from these values: 14, value of 'att1' at trigger time, 14.5, and the value for 'att2' at start of day (12:00am) on Dec 14, 2016]
- `Max('enum_att1', 'enum_att2', 'enum_att3')`

  [At trigger time, find the highest value from an enumeration set and return its name]
- `Max('*', 'y', 'Saturday')`

  [Find the most recent timestamp. Returns current time from the example above]
- `Max('*'-'*-1h', 't+8h'-'y+4h', '*'-'t')`

  [Find the longest interval from these time periods: from 1 hour ago to now, from 4:00am yesterday to today at 8am, from 12:00am yesterday to 12:00am on 20th of this month, and from the beginning of day today (12:00am) till now]
- `Max(Variable1)`

Name	Expression	Value at Evaluation
Variable1	RecordedValues('att1', '*-10m', '*')	[1, 3, 5, 7, 9]
Variable 2	Max(Variable1)	9

  [Find the maximum of values in an array named Variable1]
- `Max(Data)`

  [Return the event-weighted maximum of values from an array named *Data*]
- `Max(Data, "EventWeighted")`

  [Return the event-weighted maximum of values from an array named *Data*]
- `Max(Data, "TimeWeighted")`

  [Return the time-weighted maximum of values from an array named *Data* using the timestamps of the earliest and latest values of the array as the start time and end time respectively]

Median

Return the time-weighted or event-weighted median (middle value) of one or more values.

Syntax

```
Median(x1 [, ... xn])
Median(array)
```

Arguments

- **x1 [, ... xn]**

  Arguments and single array of same value type (integers and real numbers, time expressions, or time intervals)

Returns

The median value of the input argument(s). If there are even number of arguments, the average of the two middle values is returned. Returns an error if the array does not consist of same value type

Exceptions

Arguments whose run-time values are digital states are ignored. The function must have one or more arguments that evaluate to non-digital states; otherwise, `Median` returns an error value. Also returns an error if the array does not consist of same value type

Notes

Arguments must be of same value type. The data type of the first argument sets the tone for the rest

Unit of Measure (UOM) of the arguments is carried over to the result when at least one argument has a defined UOM while others don't. The returned value lacks a UOM if arguments have different UOMs

Examples

- `Median(14, 'att1', 14.5, TagVal('att2', '14-Dec-16'))`

  [Find the median of these values: 14, the current value of *att1*, 14.5, and the value for *att2* at midnight on Dec 14, 2016]
- `Median('*', 'y', 'Saturday')`

  [Find the median of these timestamps: now, 12:00am yesterday, and 12:00am last Saturday]
- `Median('*'-'*-1h', 't+4h'-'y+8h', 'y+2h'-'20')`

  [Find the median of these time periods: from 1 hour ago to now, from 8:00am yesterday to 4:00am today, and 2:00am yesterday to the start of day (12am) on the 20th of this month]
- `Median(Variable1)`

Name	Expression	Value at Evaluation
Variable1	RecordedValues('att1', '*-10m', '*')	[1, 3, 5, 7, 9]
Variable 2	Median(Variable1)	5

  [Find the median of values in an array named Variable1]
- `Median(Data)`

  [Return the event-weighted median of values for an array named *Data*]

Mid

Return a sub-string within a string.

Syntax

```
Mid(s1, start [,len])
```

Arguments

- **s1**
  string
- **start**
  An integer specifying the position of the first character within the string. The first character in the string is number 1
  len
  Optional: The maximum length of the returned string. The default is the length of the string

Returns

`len` characters of the string to the right of (and including) the first character whose position is specified by start

Exceptions

If the arguments are not of the required types, an error value is returned. The maximum number of characters that can be returned is 999

Example

- `Mid("String", 3)`
  [Returns "ring"]
- `Mid("String", 3, 2)`
  [Returns "ri"]

Min

Return the time-weighted or event-weighted minimum from a set of values.

Syntax

```
Min(x1, ... xn)
Min(array [, calculationBasis])
```

Arguments

- **x1, ... xn**
  Arguments or a single array of same value type (integers and real numbers, enumeration values, time

expressions, or time intervals)

- **calculationBasis**

  Optional. The type of calculation to be performed enclosed in double quotes. Choose between "EventWeighted" or "TimeWeighted." If omitted, the default is event-weighted. If you select "TimeWeighted," the earliest timestamp of the value marks the start time and the latest timestamp the end of a time range

Returns

The minimum from a set of arguments. The result is of the same data type as arguments

Exceptions

Arguments whose run-time values are digital states are ignored. If all values are digital states, **Min** returns an error. Returns an error if the array does not consist of same value type

Notes

- Unit of Measure (UOM) of the arguments is carried over to the result when at least one argument has a defined UOM while others don't. The returned value lacks a UOM if arguments have different UOMs

Example

- `Min(14, 'att1', 14.5, TagVal('att2','14-Dec-16'))`
- [Find the minimum from these values: 14, value of 'att1' at trigger time, 14.5, and the value for 'att2' at start of day (12:00am) on Dec 14, 2016]
- `Min('enum_att1', 'enum_att2', 'enum_att3')`
- [At trigger time, find the lowest value from an enumeration set and return its name]
- `Min('*', 'y', 'Saturday')`

  [Find the oldest timestamp from a set]
- `Min('*'-'*-1h', 't+8h'-'y+4h', 'y'-'20', '*'-'t')`

  [Find the shortest interval from these time periods: from 1 hour ago to now, from 4:00am yesterday to today at 8am, from 12:00am yesterday to 12:00am on 20th of this month, and from the beginning of day today (12:00am) till now]
- `Min(Variable1)`

Name	Expression	Value at Evaluation
Variable1	RecordedValues('att1', '*-10m', '*')	[1, 3, 5, 7, 9]
Variable 2	Min(Variable1)	1

  [Find the minimum of values in an array Variable1]
- `Min(Data)`

[Return the event-weighted minimum of values from an array named *Data*]

- `Min(Data, "EventWeighted")`

  [Return the event-weighted minimum of values from an array named *Data*]
- `Min(Data, "TimeWeighted")`

  [Return the time-weighted minimum of values from an array named *Data* using the timestamps of the earliest and latest values of the array as the start time and end time respectively]

Minute

Extract the minute from a time expression.

Syntax

```
Minute(t1)
```

Arguments

- **t1**

  A time expression, enclosed in single quotes

Returns

The minute of time, in the range 0-59

Exceptions

None

Example

- `Minute('*')`

  [Extract the minute from current time]
- `Minute(FindGT('att1', '-1h', '*', 5)`

  [Extract the minute from a timestamp when the value of 'att1' was first greater than 5 in the past hour. Return error if it was never over 5]

Mod

The modulus operator (mod) returns the remainder from the quotient of two numeric values. For *x* **Mod** *y*, this is the remainder from *x*/*y*.

Syntax

```
x Mod y
```

Arguments

- **x**
  Any expression that evaluates to a numeric value
- **y**
  Any non-zero numeric value

Returns

Remainder from *x*/*y*

Exceptions

None

Notes

UOM of *x*, if it exists, is carried over to the result

Example

- `11 Mod 3`
  [Returns 2]

Month

Extract the month from a given time expression.

Syntax

```
Month(t1)
```

Arguments

- **t1**
  A time expression, enclosed in single quotes

Returns

The month of time, in the range 1-12

Exceptions

None

Example

- `Month('*')`
  [Return the current month]
- `Month(FindEq('att1', '-10d', '*', 5'))`
  [Return the month from a timestamp when the value of 'att1' was first equal to 5 in the past 10 days]

NextEvent

Find the timestamp of the next recorded value after a specified time.

Syntax

```
NextEvent(attname, t1)
```

Arguments

- **attname**
  The name of an attribute, enclosed in single quotation marks
- **t1**
  A time expression

Returns

The timestamp of the next recorded value after the specified time for the specified attribute

Exceptions

If there is no recorded value after the specified time, then the function returns *No Data*

Example

- `NextEvent('att1', 't')`
  [Find and return the timestamp of the next recorded value of 'att1' since the start of day (12am) today]

NextVal

Find the next recorded value after a specified time.

Syntax

```
NextVal(attname, t1)
```

Arguments

- **attname**
  The name of an attribute, enclosed in single quotation marks
- **t1**
  A time expression

Returns

The next recorded value after the specified time for the specified attribute

Exceptions

If there is no recorded value after the specified time, the function returns an error.

Example

- `NextVal('att1', 't')`
  [Find and return the next recorded value of 'att1' since the start of day (12am) today]

Noon

Return a timestamp for noon on the day of a given time expression.

Syntax

```
Noon(t1)
```

Arguments

- **t1**
  A time expression enclosed in single quotes

Returns

A timestamp corresponding to noon of the day of the input time

Exceptions

None

Notes

This function is useful for establishing a unique clock time independent of the length of particular days.

Example

- `Noon('*')`
  [Return the timestamp for noon of current day]
- `Noon(FindEq('att1', '-3d', '*', 50))`
  [Return the timestamp for noon of the day when 'att1' was first equal to 50 in the past 3 days]

NoOutput

Do not write current calculation result.

Syntax

```
NoOutput()
```

Arguments

None

Notes

It is important to include the parentheses after this function (use *NoOutput()* instead of *NoOutput* as *NoOutput* is an invalid syntax). This function applies only to the current calculation.

Example

- `If 'att1' < 100 or 'att1' > 200 then 'att1' else NoOutput()`

Normalrnd

Return a random number that maps the normal distribution curve using a specified mean and standard deviation.

Syntax

```
Normalrnd(x, y)
```

Arguments

- **x**
  A real number specifying the mean of the normal distribution curve
- **y**
  A real number specifying the standard deviation of the normal distribution curve

Returns

A random number that maps the normal distribution curve with a specified mean and standard deviation

Exceptions

None

Example

- `Normalrnd(300, 2.5)`

NOT

Return the negation of an expression. Return *True* if the truth value of an expression is false and *False* if the truth value of an expression is true.

Syntax

```
NOT expression
```

Arguments

- **expression**
  Any expression that evaluates to true or false

Returns

True when the expression is false and *False* when the expression is true

Exceptions

None

Example

- `NOT (1 In (1...10))`
  [Returns False since (1 In (1...10)) is true]

NumOfChanges

Return the number of changes in value for an attribute within a specified time range.

Syntax

```
NumOfChanges(attname, starttime, endtime)
```

Arguments

- **attname**
  The name of an attribute
- **starttime**
  A time expression that specifies the start of a time range, or a time span (such as '-3h') that specifies the start time relative to *endtime*; entries must be enclosed in single quotation marks
- **endtime**
  A time expression that specifies the end of a time range, or a time span (such as '+3h') that specifies the end time relative to *startime*; entries must be enclosed in single quotation marks

Returns

The count of changes in value for *attname* in the specified time range excluding bad values

Example

- `NumOfChanges('att1', 't', '*')`
  [Return the number of times the value of 'att1' changed since midnight until now]

OR

The logical disjunction of two expressions that returns *True* if either expression is true and *False* if both are false.

Syntax

```
expression1 OR expression2
```

Arguments

- **expression1**
  Any expression that evaluates to true or false
- **expression2**
  Any expression that evaluates to true or false

Returns

True if either expression is true and *False* if both are false

Exceptions

None

Notes

If the inputs are numeric, **OR** can do bitwise operations.

Example

- `('att1' > 50) OR ('att2' = "good")`

ParseTime

Translate a PI time expression to a timestamp. Use regular time expression inside single quotes for better performance.

Syntax

```
ParseTime(s1)
```

Arguments

- **s1**
  A character string or an array of strings in PI time format, enclosed in double quotes

Returns

The timestamp corresponding to *s1*

Exceptions

If *s1* is not a character string, or if there is a syntax error, returns an error value.

If the array does not consist of strings in a time format, returns an error.

Notes

Use of the time expression '01-Jan-2018' over the string "01-Jan-2018" is strongly recommended.

Example

- `ParseTime(Concat("12", "-31", "-16"))`
  [Returns 12/31/2016 12:00:00 AM, which is the same as '12/31/16']
- `ParseTime("14-Dec-16")`
  [Renders the same result as '14-Dec-16'. Use only when string operations are necessary]
- `ParseTime("*")`
  [Renders the same result as '*'. Use only when string operations are necessary]
- `ParseTime(Variable1)`

Name	Expression	Value at Evaluation
Variable1	'TimeStringArray'	[t, t+3h, t+14h]
Variable 2	ParseTime(Variable1)	[3/14/2018 12:00:00 AM, 3/14/2018 3:00:00 AM, 3/14/2018 2:00:00 PM]

[Renders the array results in a proper PI Time format, assuming today is 3/14/18]

PctGood

Find the time-weighted or event-weighted percentage, over a specified time range, for which the attribute had good values.

Syntax

```
PctGood(attname, starttime, endtime [, calculationBasis])
```

Arguments

- **attname**
  attribute with time series data (such as PI point data reference) enclosed in single quotes
- **starttime**
  time expression representing the beginning of a time range enclosed in single quotes; can be a relative time

(such as'-3h') in reference to an absolute *endtime*

- **endtime**

  time expression representing the end of a time range enclosed in single quotes; can be a relative time (such as '+1h') in reference to an absolute *starttime*
- **calculationBasis**

  Optional. A string indicating the type of calculation to be performed enclosed in double quotes. Choose between "TimeWeighted" or "EventWeighted". If omitted, the default is time-weighted

Returns

An integer or real number from 0 to 100: time-weighted or event-weighted minimum time percentage during a specified time range for which the attribute had good values

Example

- `PctGood('att1', 't', '+1h')`

  [Return the time-weighted percentage between 12:00 and 1:00am today for which 'att1' had good values]
- `PctGood('att1', '-1h', '*')`

  [Return the time-weighted percentage between now and hour ago today for which 'att1' had good values]
- `PctGood('att1', '-1h', '*', "EventWeighted")`

  [Return the event-weighted percentage between now and hour ago today for which 'att1' had good values]

Pi

Returns the value for pi.

Syntax

```
Pi()
```

Arguments

Returns

The value for pi

Exceptions

None

Example

- `Pi()`

Poisson

Return a random number that maps a Poisson distribution with a specific mean.

Syntax

```
Poisson(x)
```

Arguments

- **x**
  Must be an integer or real number

Returns

A random number that maps a Poisson distribution with a specified mean

Exceptions

None

Example

- `Poisson(15)`

Poly

Return the polynomial $c_0 + c_1x + c_2x^2 + \ldots +c_nx^n$.

Syntax

```
Poly(x, c0 [, ... cn])
```

Arguments

- **x**
  Variable. An integer or real number
- **c0 [, ... cn]**
  Coefficients. There must be at least one coefficient. All must be numbers.

Returns

The value of the polynomial

Exceptions

If *x* or any coefficient is not an integer or real number, **Poly** returns an error value

Example

- `Poly(3, 4, 5)`
  [Returns 19]
- `Poly('att1', 2, 3)`

Pow

Return *x* raised to the power *y*.

Syntax

```
Pow(x, y)
```

Arguments

- **x**
  numeric value
- **y**
  numeric value

Returns

The value of *x* raised to the power *y*

Exceptions

None

Notes

- The **Pow** function only works in PI Asset Framework (PI AF) asset analytics expression functions. Use caret (^) operator for same effect in Data Archive performance equations
- Both *Pow(2, 3)* and 2^3 are acceptable in PI AF asset analytics expression functions

Example

- `Pow(2, 3)`
  [Returns 8]

- `Pow(2.5, 3)`
  [Returns 15.625]
- `Pow(TagVal('att1', '*'), 8)`
  [Return the current value of 'att1' raised to the power 8]

PrevEvent

Find the timestamp of the last recorded value before a specified time.

Syntax

```
PrevEvent(attname, t1)
```

Arguments

- **attname**
  The name of an attribute, enclosed in single quotation marks
- **t1**
  A time expression

Returns

The timestamp of the last recorded value before the specified time for the specified attribute

Exceptions

If there is no recorded value before the specified time, the function returns an error.

Example

- `PrevEvent('att1', '*')`
  [Find the timestamp of the last recorded value of 'att1' before current time]

PrevVal

Find the last recorded value before a specified time.

Syntax

```
PrevVal(attname, t1)
```

Arguments

- **attname**
  The name of an attribute, enclosed in single quotation marks
- **t1**
  A time expression

Returns

The last recorded value before the specified time for the specified attribute

Exceptions

If there is no recorded value before the specified time, the function returns an error.

Example

- `PrevVal('att1', '*')`
  [Find and return the last recorded value of 'att1' before current time]
- `PrevVal('att1', '15-Mar-17')`
  [Find and return the last recorded value of 'att1' before start of day (12am) March 15th, 2017]

PStDev

Return the time-weighted or event-weighted population standard deviation for a population of one or more values.

Syntax

```
PStDev(x1 [, ... xn])
PStDev(array [, calculationBasis])
```

Arguments

- **x1, ... xn**
  Arguments and a single array of same value type (integers and real numbers, time expressions, or time intervals)
- **calculationBasis**
  Optional. The type of calculation to be performed enclosed in double quotes. Choose between "EventWeighted" or "TimeWeighted." If omitted, the default is event-weighted. If you select "TimeWeighted," the earliest timestamp of the value marks the start time and the latest timestamp the end of a time range

Returns

The *population standard deviation* for the arguments. Returns a numeric value if the arguments are numbers. For arguments that are time expressions (time or time period), a number indicating a time period expressed in seconds is returned

The population standard deviation of a population x1, ..., xn is

$$\sqrt{\frac{\sum (x_i - \mu)^2}{n}}$$

where μ is the mean of the arguments; that is,

$$\frac{\sum x_i}{n}$$

Exceptions

Arguments whose run-time values are digital states are ignored. If all values are digital states, an error value is returned. Returns an error if the array does not consist of same value type

Notes

- **PStDev** uses every value in a population to calculate the population standard deviation. However, it is common, especially for a large population, to estimate standard deviation from a sample of the population. **SStDev** uses a set of sample values to calculate sample standard deviation, which approximates the population standard deviation
- Unit of Measure (UOM) of the arguments is carried over to the result when at least one argument has a defined UOM while others don't. The returned value lacks a UOM if arguments have different UOMs

Example

- `PStDev(14, 'att1', 14.5, TagVal('att2','14-Dec-16'))`

  [Return the "population standard deviation" of these values: 14, the current value of 'att1', 14.5, and the value for 'att2' at start of day (12:00am) on Dec 14, 2016]
- `PStDev('*', 'y', 'Saturday')`

  [Return the "population standard deviation" of these timestamps]
- `PStDev('*'-'*-1h', 't+8h'-'y+4h', 'y'-'20', '*'-'t')`

  [Return the "population standard deviation" of these time periods: from 1 hour ago to now, from 4:00am yesterday to today at 8am, from 12:00am yesterday to 12:00am on 20th of this month, and from the beginning of day today (12:00am) till now]
- `PStDev(Variable1)`

Name	Expression	Value at Evaluation
Variable1	RecordedValues('att1', '*-10m', '*')	[1, 3, 5, 7, 9]
Variable 2	PStDev(Variable1)	2.8284

[Find the "population standard deviation" of values in an array named Variable1]

- `PStDev(Data)`

  [Return the event-weighted "population standard deviation" of values in an array named *Data*]
- `PStDev(Data, "EventWeighted")`

  [Return the event-weighted "population standard deviation" of values in an array named *Data*]
- `PStDev(Data, "TimeWeighted")`

  [Return the time-weighted "population standard deviation" of values in an array named *Data* using the timestamps of the earliest and the latest values of the array as the start time and end time respectively]

Rand

Return a random number between 0 and 1. For specified *x* and *y* values, return a random number between x - y/2 and x + y/2.

Syntax

```
Rand(x, y)
```

Arguments

- **x**

  A real number specifying the center point of the range
- **y**

  A real number specifying the size of the range.

  If no arguments are specified, the default range is from 0 to 1.

Returns

A random number between 0 and 1. For specified *x* and *y* values, returns a random number between x- y/2 and x + y/2.

Exceptions

None

Example

- `Rand()`
- `Rand(500, 250)`

Range

Find the time-weighted or event-weighted difference between the maximum and minimum values for an attribute during a specified time range. Time-weighted values are interpolated at time boundaries when possible and extrapolated otherwise.

Syntax

```
Range(attname, starttime, endtime [, pctgood, calculationBasis])
```

Arguments

- **attname**

  attribute with time series data (such as PI point data reference) enclosed in single quotes
- **starttime**

  time expression representing the beginning of a time range enclosed in single quotes; can be a relative time (such as'-3h') in reference to an absolute *endtime*
- **endtime**

  time expression representing the end of a time range enclosed in single quotes; can be a relative time (such as '+1h') in reference to an absolute *starttime*
- **pctgood**

  Optional. Time-weighted or event-weighted minimum time percentage over a specified time range for which the attribute had good values
- **calculationBasis**

  Optional. A string indicating the type of calculation to be performed enclosed in double quotes. Choose between "TimeWeighted" or "EventWeighted". If omitted, the default is time-weighted

Returns

The time-weighted or event-weighted difference between the attribute's maximum and minimum values during the specified time. Time-weighted values are interpolated at time boundaries when possible and extrapolated otherwise

Exceptions

If the attribute has no good values or the *pctgood* minimum is not reached in the given time range, an error is returned

Notes

- Bad values are excluded from **Range** calculation
- In order to configure an optional parameter, any previous optional parameter must be specified

  Note: If the attribute has very few good values during the time range, this function's result may not be trustworthy. Use the **PctGood** function to find out what percentage of the values is good

Example

- `Range('att1', 't', '+1h')`

  [Return the time-weighted difference between the maximum and minimum values for 'att1' between 12:00 and 1:00am today]
- `Range('att1', 't', '+1h', 80)`

  [Return the time-weighted difference between the maximum and minimum values for 'att1' between 12:00 and 1:00am today when at least 80% of the values were good]
- `Range('att1', 't', '+1h', 80, "EventWeighted")`

  [Return the event-weighted difference between the maximum and minimum values for 'att1' between 12:00 and 1:00am today when at least 80% of the values were good]

Rate

Return the difference over time between the current value and a previously evaluated value of an attribute or (time) expression.

Syntax

```
Rate(x)
```

Arguments

- **x**

  An attribute or (time) expression that resolves to a numeric value

Returns

- The calculated difference, using the value of the attribute, numeric, or timestamp value of the expression parameter, according to this formula: (current value – previously evaluated value) / (current timestamp – timestamp from previous evaluation)

Example

- `Rate(‘att1’)`
- `Rate('sinusoid')`

- `Rate(DaySec(‘*’) + 1)`

RecordedValues

Obtain values within a particular time range based on user-specified boundary type.

Syntax

```
RecordedValues(attname, starttime, endtime [, boundarytype])
```

Arguments

- **attname**

  attribute of time series data (such as PI point data reference) enclosed in single quotes
- **starttime**

  a time expression representing the beginning of a time range enclosed in single quotes; can be a relative time (such as '-3h') in reference to an absolute endtime
- **endtime**

  a time expression representing the end of a time range enclosed in single quotes; can be a relative time (such as '+1h') in reference to an absolute starttime
- **boundarytype**

  a string indicating the data retrieval behavior at the end points of a specified time in double quotes:

  "Inside" - return the nearest recorded values inside the requested time range as the first and last values

  "Outside" - return the nearest recorded values on the outside of the requested time range as the first and last values

  "Interpolated" - create an interpolated value at the end points of the requested time range if a recorded value does not exist at that time

  Note: If unspecified, the default is the "Inside" mode.

Returns

An array of the values obtained within the specified range

Exceptions

If the attribute does not support range calls, the function returns an error with indication

Notes

- If the *starttime* is earlier than the *endtime*, the resulting values will be in time-ascending order, otherwise they will be in time-descending order.
- When no values are available but you specified "Outside" or "Interpolated" boundary types, a special value (**NoData**) is returned.

- This function retrieves up to 500,000 values at a time.

Example

- `RecordedValues('att1', 't+2h', 't+8h')`

  [Return the values of 'att1' between 2:00 and 8:00am today using the default "Inside" retrieval mode]
- `RecordedValues('att1', 't', '+1h', "Outside")`

  [Return the values of 'att1' between 12:00 and 1:00am today using the "Outside" retrieval mode]

RecordedValuesByCount

Obtain a specified number of values beginning at the requested start time in the direction specified.

Syntax

```
RecordedValuesByCount(attname, starttime, count [, timedirection, boundarytype])
```

Arguments

- **attname**

  attribute of time series data (such as PI point data reference) enclosed in single quotes
- **starttime**

  a time expression representing the beginning of request
- **count**

  the number of stored (recorded) values to return. The value must be greater than zero
- **timedirection**

  a string indicating the time direction in data retrieval enclosed in double quotes:

  "Backward" - begin at the start time and move backward in time. Values will be returned in time-descending order

  "Forward" - begin at the start time and move forward in time. Values will be returned in time-ascending order

  Note: If unspecified, the default is the "Backward" mode.
- **boundarytype**

  a string indicating the data retrieval behavior at the end point of a specified time in double quotes:

  "Inside" - return the nearest recorded values inside the requested time boundary

  "Outside" - return the nearest recorded values on the outside of the requested time boundary

  "Interpolated" - create an interpolated value at the requested time boundary if a recorded value does not exist at that time

  Note: If unspecified, the default is the "Inside" mode.

Returns

An array of the specified number of values (if available)

Exceptions

If the attribute does not support the calls, the function returns an error with indication

Notes

- When no values are available but you specified "Outside" or "Interpolated" boundary types, a special value (**NoData**) is returned.

Example

- `RecordedValuesByCount('att1', 't+2h', 5)`

  [Return 5 values of 'att1' in time-descending order starting from 2:00am today using the default "Inside" retrieval mode moving backwards in time]
- `RecordedValuesByCount('att1', 't+8h', 3, "Forward", "Outside")`

  [Return the first 3 values of 'att1' since 8am today using the "Outside" retrieval mode]

Remainder

Return the remainder resulting from the division of x by y using the IEEERemainder method. This remainder is equal to x - (y Q), where Q is the quotient of x / y rounded to the nearest integer (if x / y falls halfway between two integers, the even integer is returned). If x - (y Q) is zero, the value +0 is returned if x is positive, or -0 if x is negative. If y = 0, NaN (not a number) is returned.

Syntax

```
Remainder(x, y)
```

Arguments

- **x**

  Any numeric value
- **y**

  Any non-zero numeric value

Returns

Return the remainder resulting from the division of x by y using the IEEERemainder method

Note: This is not the same as the remainder returned using the Mod (modulus) operator

Exceptions

None

Notes

Unit of measure of *x*, if it exists, is carried over to the result

Example

- `Remainder(11, 3)`
  [Returns -1]
- `Remainder(10, 3)`
  [Returns 1]

Right

Return a specified number of characters of a string from the right.

Syntax

```
Right(s1, len)
```

Arguments

- **s1**
  string
- **len**
  integer

Returns

len characters of the string from the right

Exceptions

If the arguments are not of the required types, an error value is returned.

Example

- `Right("String", 3)`
  [Returns "ing"]
- `Right("String", 20)`

[Returns "String"]

Round

Round a number or time to the nearest unit.

Syntax

```
Round(x [, unit] [, roundingMode])
```

Arguments

- **x**

  Integer, real number, time expression, or time interval
- **unit**

  Optional. The size of the unit to round to. If *x* is a number, unit must be a number. If *x* is a time expression or time period, unit must be a time expression. If unit is omitted, **Round** rounds to the nearest even integer (for numbers) or second (for time expressions).
- **roundingMode**

  Optional. The type of rounding to apply. Using the string "AwayFromZero" applies rounding of midpoint values away from zero. Omission of this parameter, or the use of "ToEven", applies the default banker's rounding.

Returns

The nearest value to *x* which is an integer multiple of unit. Returns the same data type as *x*. For more details, see the following examples.

Exceptions

If *x* is a string, or if unit is of the wrong data type, an error value is returned.

Notes

The system uses a banker's rounding which rounds a number to the nearest even number.

If *x* is time and unit is omitted, this routine has no effect: times are accurate only to a second.

Unit of measure of the argument, if it exists, is carried over to the result.

Example

- `Round(12.499)`

  [Returns 12 (rounded to the nearest integer)]
- `Round(12.5)`

[Returns 12 (rounded to the nearest even integer)]

- `Round(12.5, "AwayFromZero")`

  [Returns 13 (rounded up to the nearest integer)]
- `Round (13.5)`

  [Returns 14 (rounded to the nearest integer)]
- `Round(13.45, 0.1)`

  [Returns 13.4 (rounded to the nearest even number in one decimal place)]
- `Round(13.45, 0.1, "AwayFromZero")`

  [Returns 13.5 (rounded up to the nearest number in one decimal place)]
- `Round(1285, 10)`

  [Returns 1280 (rounded to the nearest even 10)]
- `Round(1285, 10, "AwayFromZero")`

  [Returns 1290]
- `Round(1285, 10, "ToEven")`

  [Returns 1280 (rounded to the nearest even 10)]
- `Round('14-Dec-16 11:47:16')`

  [Returns 14-Dec-16 11:47:16 (rounded to the nearest second)]
- `Round('14-Dec-16 12:30', '+1h')`

  [Returns 14-Dec-97 12:00:00 (rounded to the nearest even hour)]
- `Round('18:47:15'-'15:00:09')`

  [Returns 03:47:06 (rounded to the nearest second)]
- `Round('18:45:40'-'15:15:10','+1m')`

  [Returns 03:30:00 (rounded to the nearest minute)]

  Note: Round to the nearest day results in a timestamp of the closest day in UTC time and not local time.

Roundfrac

Round a numeric value x to y decimal places.

Syntax

```
Roundfrac(x, y [,roundingMode] )
```

Arguments

- **x**

  Must be an integer, real number or time expression
- **y**

  Integer that specifies the number of decimal places to round *x* to
- **roundingMode**

Optional. The type of rounding to apply. Using the string "AwayFromZero" applies rounding of midpoint values away from zero. Omission of this parameter, or the use of "ToEven", applies the default banker's rounding

Returns

The value of x rounded to y decimal places. Returns the same data type as *x*. For more details, see the examples

Exceptions

If *x* is a string, neither a numeric value nor a time expression, returns an error

Notes

The system uses a banker's rounding which rounds a number to the nearest even number.

Unit of measure of the argument, if it exists, is carried over to the result.

If *x* is time and unit is omitted, this routine has no effect: times are accurate only to 1 second.

Example

- `Roundfrac (Pi(), 4)`
  [Returns 3.1416]
- `Roundfrac (0.005, 2)`
  [Returns 0]
- `Roundfrac (0.005, 2, "ToEven")`
  [Returns 0]
- `Roundfrac (0.005, 2, "AwayFromZero")`
  [Returns 0.01]
- `Roundfrac (TagVal('att1'), 2)`
  [Return the value of 'att1' rounded to 2 decimal places]

RTrim

Trim trailing blanks from a string.

Syntax

```
RTrim(s1)
```

Arguments

- **s1**

string

Returns

The source string with trailing blanks removed

Exceptions

If *s1* is not a string, an error value is returned

Example

- `RTrim("String ")`
  [Returns "String"]
- `RTrim(" String")`
  [Returns " String"]

Sec

Return the trigonometric secant of a number.

Syntax

`Sec(x)`

Arguments

- **x**
  Must be an integer or real number, which represents an angle in radians

Returns

The secant of *x*

Exceptions

If *x* is not a number, returns an error value

Example

- `Sec(1)`
  [Returns 1.8508]
- `Sec(1.1)`

[Returns 2.2046]
- `Sec('att1')`
  [Return the trigonometric secant of the value of 'att1' at time of evaluation]

Sech

Return the hyperbolic secant of a number.

Syntax

`Sech(x)`

Arguments

- **x**
  Must be an integer or real number

Returns

The hyperbolic secant of *x*

Exceptions

If *x* is not a number, returns an error value

Example

- `Sech(1)`
  [Returns 0.64805]
- `Sech(1.1)`
  [Returns 0.59933]
- `Sech('att1')`
  [Return the hyperbolic secant of the value of 'att1' at trigger time]

Second

Extract the second from a time expression.

Syntax

`Second(t1)`

Arguments

- **t1**
  A time expression enclosed in single quotes.

Returns

The second of time, in the range 0-59

Exceptions

None

Example

- `Second('*')`
  [Return the second from current time]
- `Second(FindEq('att1', '-1m', '*', 2)`
  [Return the second from a timestamp when the value of 'att1' was first equal to 2 in the past minute]

SecSinceChange

Return the time in seconds since an attribute value or the timestamp was updated.

Syntax

```
SecSinceChange(attname)
```

Arguments

- **attname**
  The name of an attribute, enclosed in single quotation marks

Returns

The number of seconds since the last update of the value or a timestamp for an attribute

Example

- `SecSinceChange('att1')`
  [Return the number of seconds since the value of 'att1' or the timestamp was updated]

Sgn

Return an indicator of the numerical sign of a number.

Syntax

```
Sgn(x)
```

Arguments

- **x**
  Must be an integer or real number

Returns

-1 if x < 0; 0 if x = 0; 1 if x > 0

Exceptions

If *x* is not an integer or real number, returns an error value

Example

- `Sgn(1.1)`
  [Returns 1]
- `Sgn(0)`
  [Returns 0]
- `Sgn(-0.1)`
  [Returns -1]
- `Sgn('att1')`
  [Returns 1 if the value of 'att1' is positive, 0 if it equals 0, and -1 if it is negative]

Sin

Return the trigonometric sine of a number.

Syntax

```
Sin(x)
```

Arguments

- **x**

Must be an integer or real number, which represents an angle in radians

Returns

The sine of *x*

Exceptions

If *x* is not a number, returns an error value

Example

- `Sin(1)`
  [Returns 0.84147]
- `Sin(1.1)`
  [Returns 0.89121]
- `Sin('att1')`
  [Return the trigonometric sine of the value of 'att1' at trigger time]

Sinh

Return the hyperbolic sine of a number.

Syntax

```
Sinh(x)
```

Arguments

- **x**
  Must be an integer or real number

Returns

The hyperbolic sine of *x*

Exceptions

If *x* is not a number, returns an error value

Example

- `Sinh(1)`

  [Returns 1.1752]
- `Sinh(1.1)`
  [Returns 1.3356]
- `Sinh('att1')`
  [Return the hyperbolic sine of the value of 'att1' at trigger time]

Split

Split a string into substrings based on a specified delimiter and return an array.

Syntax

```
Split(s1, delimiter)
```

Arguments

- **s1**
  Input string in double quotes
- **delimiter**
  Space or character separating the input string such as "|" or "," in double quotes

Returns

Returns an array of substrings obtained by the separation of the input string based on the delimiter

Exceptions

- If the delimiter doesn’t exist, an array is returned that contains one item, the original string
- If either of the two arguments (s1 or delimiter) is not a string, the function returns an error

Example

- `Split("Value1|Value2|Value3", "|")`
  [Returns [Value1, Value2, Value3]]

Sqr

Return the square root of a number.

Syntax

```
Sqr(x)
```

Arguments

- **x**
  Must be an integer or real number equal to or greater than zero

Returns

The square root of *x*

Exceptions

If *x* is negative, or is not a number, returns an error value

Example

- `Sqr(2)`
  [Returns 1.4142]
- `Sqr(2.1)`
  [Returns 1.4491]
- `Sqr('att1')`
  [Return square root of the value of 'att1' at trigger time]

StateNo

Translate a digital state into its corresponding enumeration value.

Syntax

```
StateNo(digstate)
```

Arguments

- **digstate**
  An enumeration value

Returns

The offset into the Digital State Set corresponding to *digstate*

Exceptions

If a point is passed as *digstate* that is not a digital point, returns an error value

Notes

A digital state may appear more than once in the digital state table. In this case, the value that **StateNo** returns may vary. If *digstate* is the value of a digital point, **StateNo** returns a code number appropriate for that point.

Example

- `StateNo(DigState("No Data"))`
  [Returns 248]
- `StateNo(TagVal('enum_att1', '*-1h'))`
  [Return the digital state number corresponding to the value of 'enum_att1' an hour ago]

SStDev

Return the time-weighted or event-weighted sample standard deviation for two or more arguments that are a sample of a larger population.

Syntax

```
SStDev(x1, x2 [, ... xn])
SStDev(array [, calculationBasis])
```

Arguments

x1, ... xn

- Arguments and a single array of same value type (integers and real numbers, time expressions, or time intervals)
- **calculationBasis**
  Optional. The type of calculation to be performed enclosed in double quotes. Choose between "EventWeighted" or "TimeWeighted." If omitted, the default is event-weighted. If you select "TimeWeighted," the earliest timestamp of the value marks the start time and the latest timestamp the end of a time range

Returns

The *sample standard deviation* of the arguments. Returns a numeric value if the arguments are numbers. For arguments that are time expressions (time or time period), a number indicating a time period expressed in seconds is returned

The standard deviation of a sample x1, ... xn is equal to

$$\sqrt{\frac{\sum(x_i-\mu)^2}{n-1}}$$

Where μ is the sample mean; that is,

$$\frac{\sum x_i}{n}$$

Exceptions

Arguments whose run-time values are digital states are ignored. If there are not at least two numeric values, **SStDev** returns a zero. Returns an error if the array does not consist of same value type

Notes

- In the rare case where you have the entire population, rather than a sample, you might use the function **PstDev**, rather than **SStDev**
- Unit of Measure (UOM) of the arguments is carried over to the result when at least one argument has a defined UOM while others don't. The returned value lacks a UOM if arguments have different UOMs

Example

- `SStDev(14, 'att1', 14.5, TagVal('att2','14-Dec-16'))`

  [Return the "sample standard deviation" of these values: 14, the current value of 'att1', 14.5, and the value for 'att2' at start of day (12:00am) on Dec 14, 2016]
- `SStDev('*', 'y', 'Saturday')`

  [Return the "sample standard deviation" of these timestamps]
- `SStDev('*'-'*-1h', 't+8h'-'y+4h', 'y'-'20', '*'-'t')`

  [Return the "sample standard deviation" of these time periods: from 1 hour ago to now, from 4:00am yesterday to today at 8am, from 12:00am yesterday to 12:00am on 20th of this month, and from the beginning of day today (12:00am) till now]
- `SStDev(Variable1)`

Name	Expression	Value at Evaluation
Variable1	RecordedValues('att1', '*-10m', '*')	[1, 3, 5, 7, 9]
Variable 2	SStDev(Variable1)	3.1623

  [Find the "sample standard deviation" of values in an array named Variable1]
- `SStDev(Data)`

[Return the event-weighted "sample standard deviation" of values in an array named *Data*]

- `SStDev(Data, "EventWeighted")`

  [Return the event-weighted "sample standard deviation" of values in an array named *Data*]
- `SStDev(Data, "TimeWeighted")`

  [Return the time-weighted "sample standard deviation" of values in an array named *Data* using the timestamps of the earliest and latest values of the array as the start time and end time respectively]

StDev

Find the time-weighted or event-weighted standard deviation of values for an attribute from a specified time range.

Syntax

```
StDev(attname, starttime, endtime [, pctgood, calculationBasis])
```

Arguments

- **attname**

  attribute with time series data (such as PI point data reference) enclosed in single quotes
- **starttime**

  time expression representing the beginning of a time range enclosed in single quotes; can be a relative time (such as'-3h') in reference to an absolute *endtime*
- **endtime**

  time expression representing the end of a time range enclosed in single quotes; can be a relative time (such as '+1h') in reference to an absolute *starttime*
- **pctgood**

  Optional. Time-weighted or event-weighted minimum time percentage during a specified time range for which the attribute had good values
- **calculationBasis**

  Optional. A string indicating the type of calculation to be performed enclosed in double quotes. Choose between "TimeWeighted" or "EventWeighted". If omitted, the default is time-weighted

Returns

The time-weighted or event-weighted standard deviation for attribute values from the specified time range

Exceptions

If the attribute has no good values or the *pctgood* minimum is not reached for the given time range, an error is returned

Notes

- Bad values are excluded from **StDev** calculation
- In order to configure an optional parameter, any previous optional parameter must be specified

  Note: If the attribute has very few good values during the time range, this function's result may not be trustworthy. Use the **PctGood** function to find out what percentage of the values is good

Example

- `StDev('att1', 't', '+1h')`

  [Return the time-weighted standard deviation of values for 'att1' between 12:00 and 1:00am today]
- `StDev('att1', 't', '+1h', 80)`

  [Return the time-weighted standard deviation of values for 'att1' between 12:00 and 1:00am today when at least 80% of the values were good]
- `StDev('att1', 't', '+1h', 80, "EventWeighted")`

  [Return the event-weighted standard deviation of values for 'att1' between 12:00 and 1:00am today when at least 80% of the values were good]

String

Convert any value to a string.

Syntax

```
String(x)
```

Arguments

- **x**

  Any expression that is of normal PI value type

Returns

The string representing the value argument

Exceptions

None

Example

- `String("Hello, PI user!")`

  [Returns "Hello, PI user!"]

- `String(12.23)`
  [Returns "12.23"]
- `String('sinusoid')`
  [Return current value of 'sinusoid' in string]
- `String('*')`
  [Return the current time in string]

TagAvg

Find the time-weighted or event-weighted average of values for an attribute during a specified time range.

Syntax

```
TagAvg(attname, starttime, endtime [, pctgood, calculationBasis])
```

Arguments

- **attname**
  attribute with time series data (such as PI point data reference) enclosed in single quotes
- **starttime**
  time expression representing the beginning of a time range enclosed in single quotes; can be a relative time (such as '-3h') in reference to an absolute *endtime*
- **endtime**
  time expression representing the end of a time range enclosed in single quotes; can be a relative time (such as '+1h') in reference to an absolute *starttime*
- **pctgood**
  Optional. Time-weighted or event-weighted minimum time percentage in a specified time range for which the attribute had good values

  calculationBasis

  Optional. A string indicating the type of calculation to be performed enclosed in double quotes. Choose between "TimeWeighted" or "EventWeighted". If omitted, the default is time-weighted

Returns

The time-weighted or event-weighted average of attribute values during a specified time range

Exceptions

If the point has no good values or the *pctgood* minimum is not reached for the given time range, returns an error value

Notes

- Bad values are excluded from **TagAvg** calculation.
- In order to configure an optional parameter, any previous optional parameter must be specified.
- For more information, see the knowledge base article How Time-Weighted and Event-Weighted Totals and Averages Work.

  Note: If the attribute has very few good values during the time range, this function's result may not be trustworthy. Use the **PctGood** function to find out what percentage of the values is good

Example

- `TagAvg('att1', 't', '+1h')`

  [Return the time-weighted average of values for 'att1' between 12:00 and 1:00am today]
- `TagAvg('att1', 't', '+1h', 80)`

  [Return the time-weighted average of values for 'att1' between 12:00 and 1:00am today when at least 80% of the values were good]
- `TagAvg('att1', 't', '+1h', 80, "EventWeighted")`

  [Return the event-weighted average of values for 'att1' between 12:00 and 1:00am today when at least 80% of the values were good]

TagBad

For an attribute associated with a PI point, test the point for an abnormal state at a specified time. If the point is an integer or real number, any digital state is considered abnormal. For digital points, the states that are defined for that point are normal; all others are abnormal.

Syntax

```
TagBad(attname [, t1])
```

Arguments

- **attname**

  An attribute associated with a PI point
- **time**

  Optional: A time expression. If omitted, current time ('*') is used.

Returns

False if the point's state at time is normal, True if it is abnormal.

Exceptions

None

Notes

BadVal can test any value or expression; **TagBad** can only test a PI point.

Example

- `TagBad('att1')`
  [Returns True if PI point for 'att1' does not exist or its value corresponds to system digital state (No Data, for example)]
- `TagBad('enum_att1', '14-Dec-16')`
  [Returns True if PI point for 'enum_att1' does not exist or its value corresponds to system digital state (No Data, for example) at 12:00am on December 14th, 2016]

TagDesc

For an attribute associated with a PI point, retrieve that point's descriptor from the point database.

Syntax

```
TagDesc(attname)
```

Arguments

- **attname**
  The name of an attribute with a PI point data reference, enclosed in single quotes

Returns

The point's descriptor

Exceptions

If the PI point does not exist, an error value is returned

Example

- `TagDesc('sinusoid')`
  [Returns **12 Hour Sine Wave**]
- `TagDesc('cdt158')`

[Returns **Atmospheric Tower OH Vapor**]

TagEU

For an attribute associated with a PI point, retrieve the point's engineering unit string from the point database.

Syntax

```
TagEU(attname)
```

Arguments

- **attname**
  The name of an attribute with a PI point data reference, enclosed in single quotes

Returns

The PI point's engineering units

Exceptions

If the PI point does not exist, an error values is returned

Notes

Returns blank when PI point lacks an engineering unit

Example

- `TagEU('cdt158')`
  [Returns **DEG.C**]

TagExDesc

For an attribute associated with a PI point, retrieve the point's extended descriptor from the point database.

Syntax

```
TagExDesc(attname)
```

Arguments

- **attname**

An attribute with a PI point data reference, enclosed in single quotes

Returns

The PI point's extended descriptor

Exceptions

If the PI point does not exist, an error values is returned

Notes

Returns blank when PI point lacks extended descriptor

Example

- `TagExDesc('cdt158')`
  [Returns blank since the point lacks extended descriptor]

TagMax

Find the time-weighted or event-weighted maximum value of an attribute during a specified time range. Time-weighted values are interpolated at time boundaries when possible and extrapolated otherwise.

Syntax

```
TagMax(attname, starttime, endtime [, pctgood, calculationBasis])
```

Arguments

- **attname**
  attribute with time series data (such as PI point data reference) enclosed in single quotes
- **starttime**
  time expression representing the beginning of a time range enclosed in single quotes; can be a relative time (such as'-3h') in reference to an absolute *endtime*
- **endtime**
  time expression representing the end of a time range enclosed in single quotes; can be a relative time (such as '+1h') in reference to an absolute *starttime*
- **pctgood**
  Optional. Time-weighted or event-weighted minimum time percentage over a specified time range for which the attribute had good values

  calculationBasis

  Optional. A string indicating the type of calculation to be performed enclosed in double quotes. Choose between "TimeWeighted" or "EventWeighted". If omitted, the default is time-weighted

Returns

The attribute's time-weighted or event-weighted maximum value during the specified time range. Time-weighted values are interpolated at time boundaries when possible and extrapolated otherwise

Exceptions

If the attribute has no good values or the *pctgood* minimum is not reached for the given time range, an error value is returned

Notes

- Bad values are excluded from **TagMax** calculation
- In order to configure an optional parameter, any previous optional parameter must be specified

Note: If the attribute has very few good values during the time range, this function's result may not be trustworthy. Use the **PctGood** function to find out what percentage of the values is good

Example

- `TagMax('att1', 't', '+1h')`

  [Return the time-weighted maximum value of 'att1' between 12:00 and 1:00am today]
- `TagMax('att1', '14-Dec-16', '15-Dec-16')`

  [Return the time-weighted maximum value of 'att1' between 12:00am 12/14/2016 and 12:00am 12/15/2016]
- `TagMax('att1', 't', '+1h', 80)`

  [Return the time-weighted maximum value of 'att1' between 12:00 and 1:00am today when at least 80% of the values were good]
- `TagMax('att1', 't', '+1h', 80, "EventWeighted")`

  [Return the event-weighted maximum value of 'att1' between 12:00 and 1:00am today when at least 80% of the values were good]

TagMean

Find the average (mean) of values for an attribute during a specified time range. **TagAvg** with event-weighted option provides the same result as **TagMean**. Consider using **TagAvg** rather than **TagMean** especially if time-weighted **TagAvg** is used in other analyses.

Syntax

```
TagMean(attname, starttime, endtime [, pctgood])
```

Arguments

- **attname**

  attribute with time series data (such as PI point data reference) enclosed in single quotes
- **starttime**

  time expression representing the beginning of a time range enclosed in single quotes; can be a relative time (such as'-3h') in reference to an absolute *endtime*
- **endtime**

  time expression representing the end of a time range enclosed in single quotes; can be a relative time (such as '+1h') in reference to an absolute *starttime*
- **pctgood**

  Optional: Minimum percentage of the specified time range that attribute values must be good.

Returns

The attribute's average value over the given time. Notice that the average is not time-weighted. For time-weighted average, use **TagAvg** instead

Exceptions

If the attribute has no good values or the *pctgood* minimum is not reached for the specified time range, an error value is returned. Unlike some other summary functions, **TagMean** does not interpolate any value on the boundary. Thus, if there is no value in the specified interval, an error value is returned

Notes

Bad values are excluded from **TagMean** calculation

Note: If the attribute has very few good values during the time range, this function's result may not be trustworthy. Use the **PctGood** function to find out what percentage of the values is good

Example

- `TagMean('att1', 't', '+1h')`

  [Return the average (mean) value of 'att1' between 12:00 and 1:00am today]
- `TagMean('att1', 't', '+1h', 80)`

  [Return the average (mean) value of 'att1' between 12:00 and 1:00am today when at least 80% of the values were good]
- `TagMean('att1', '14-Dec-16', '15-Dec-16')`

  [Return the average (mean) value of 'att1' between 12:00am 12/14/2016 and 12:00am 12/15/2016]

TagMin

Find the time-weighted or event-weighted minimum value of an attribute during a specified time range. Time-weighted values are interpolated at time boundaries when possible and extrapolated otherwise.

Syntax

```
TagMin(attname, starttime, endtime [, pctgood, calculationBasis])
```

Arguments

- **attname**

  attribute with time series data (such as PI point data reference) enclosed in single quotes
- **starttime**

  time expression representing the beginning of a time range enclosed in single quotes; can be a relative time (such as'-3h') in reference to an absolute *endtime*
- **endtime**

  time expression representing the end of a time range enclosed in single quotes; can be a relative time (such as '+1h') in reference to an absolute *starttime*
- **pctgood**

  Optional. Time-weighted or event-weighted minimum time percentage over a specified time range for which the attribute had good values

  calculationBasis

  Optional. A string indicating the type of calculation to be performed enclosed in double quotes. Choose between "TimeWeighted" or "EventWeighted". If omitted, the default is time-weighted

Returns

The time-weighted or event-weighted minimum value for the attribute during the specified time range. Time-weighted values are interpolated at time boundaries when possible and extrapolated otherwise

Exceptions

If no good attribute values exist or if the *pctgood* minimum is not reached for the specified time range, an error value is returned

Notes

- Bad values are excluded from **TagMin** calculation
- In order to configure an optional parameter, any previous optional parameter must be specified

  Note: If the attribute has very few good values during the time range, this function's result may not be trustworthy. Use the **PctGood** function to find out what percentage of the values is good

Example

- `TagMin('att1', 't', '+1h')`

[Return the time-weighted minimum value of 'att1' between 12:00 and 1:00am today]

- `TagMin('att1', '14-Dec-16', '15-Dec-16')`

  [Return the time-weighted minimum value of 'att1' between 12:00am 12/14/2016 and 12:00am 12/15/2016]
- `TagMin('att1', 't', '+1h', 80)`

  [Return the time-weighted minimum value of 'att1' between 12:00 and 1:00am today when at least 80% of the values were good. Return error when minimum pctgood is not reached]
- `TagMin('att1', 't', '+1h', 80, "EventWeighted")`

  [Return the event-weighted minimum value of 'att1' between 12:00 and 1:00am today when at least 80% of the values were good. Return error when minimum pctgood is not reached.]

TagName

For an attribute associated with a PI point, get a point's name from the point database.

Syntax

`TagName(attname)`

Arguments

- **attname**

  The name of an attribute with a PI point data reference, enclosed in single quotes

Returns

The name of the PI point associated with *attname*

Exceptions

If the PI point does not exist, an error value is returned

Example

- `TagName('sinusoid')`

  [Returns **SINUSOID**]
- `TagName('cdt158')`

  [Returns **CDT158**]

TagNum

For an attribute associated with a PI point, get a point's number from the point database.

Syntax

```
TagNum(s1)
```

Arguments

- **string**
  The name of an attribute with a PI point data reference, enclosed in double quotes

Returns

The point's number

Exceptions

If the point does not exist, returns an error value

Example

- `TagNum("sinusoid")`
  [Returns 1]
- `TagNum("cdt158")`
  [Returns 3]

TagSource

For an attribute associated with a PI point, get a point's **point source** string from the point database.

Syntax

```
TagSource(attname)
```

Arguments

- **attname**
  The name of an attribute with a PI point data reference, enclosed in single quotes

Returns

The point's **point source** string

Exceptions

If the point does not exist, an error value is returned

Example

- `TagSource('sinusoid')`
  [Returns R]
- `TagSource('cdt158')`
  [Returns R]

TagSpan

For an attribute associated with a PI point, get a point's span from the point database.

Syntax

```
TagSpan(attname)
```

Arguments

- **attname**
  The name of an attribute with a PI point data reference, enclosed in single quotes

Returns

The point's span. If the point's type is *digital*, returns an integer whose value is the number of digital state defined for the point.

Example

- `TagSpan('sinusoid')`
  [Returns 100]
- `TagSpan('cdt158')`
  [Returns 200]

TagTot

Find the time-weighted (time integral) or event-weighted totalized value of an attribute's values over a specified time range, optionally converting the totalized value from its current unit of measure (UOM) to a specified UOM in same class. Time-weighted values are interpolated at time boundaries when possible and extrapolated otherwise.

Syntax

```
TagTot(attname, starttime, endtime [, toUnit, pctgood, calculationBasis])
```

Arguments

- **attname**

  attribute with time series data (such as PI point data reference) enclosed in single quotes
- **starttime**

  time expression representing the beginning of a time range enclosed in single quotes; can be a relative time (such as '-3h') in reference to an absolute *endtime*
- **endtime**

  time expression representing the end of a time range enclosed in single quotes; can be a relative time (such as '+1h') in reference to an absolute *starttime*
- **toUnit**

  Optional. The UOM to which the totalized value is converted. The specified UOM must be compatible with the UOM of the input attribute. Specifying an empty string ("") produces the same result as if no UOM were specified.

  pctgood

  Optional. Time-weighted or event-weighted minimum time percentage over a specified time range for which the attribute had good values

  calculationBasis

  Optional. A string indicating the type of calculation to be performed enclosed in double quotes. Choose between "TimeWeighted" or "EventWeighted". If omitted, the default is time-weighted.

Returns

The time-weighted (time integral) or event-weighted totalized value of an attribute's values over a specified time range. Time-weighted values are interpolated at time boundaries when possible and extrapolated otherwise.

Exceptions

- If the attribute has no good values or the *pctgood* minimum is not reached for the given time range, an error value is returned.
- If the specified UOM for conversion is incompatible with the UOM of the input attribute, an error value is returned. For example, if the UOM of the input attribute is gallons/minute, and the UOM for conversion is "m" (meters), then an error is returned.

Notes

- In case *toUnit* is not specified, the system chooses a scale factor such that the integral is correct only if the flow is expressed in units per day. If the flow is expressed in units per hour, or per some other time unit, you must multiply this result by a conversion factor. The conversion factor equals the number of actual flow time units in a day.

- For instance, if you totalize values measured in gallons per minute, multiply the result of **TagTot** by 1440 to get the answer in gallons. This conversion factor is not related to the time period you are totalizing over; it is strictly a function of the attribute's engineering units.
- Bad values are excluded from **TagTot** calculation.
- When the percentage of good data is less than 100%, **TagTot** determines the total based on good data and divides the fraction of good data in the interval. The time period during which the data is bad is ignored when calculating the total.
- In order to configure an optional parameter, any previous optional parameter must be specified.

  Note: If the attribute has very few good values during the time range, this function's result may not be trustworthy. Use the **PctGood** function to find out what percentage of the value is good.

Example

- `TagTot('att1', 't', '+1h')`

  [Return the time-weighted totalized value of 'att1' between 12:00 and 1:00am today.]
- `TagTot('att1', 't', '+1h', "m")`

  [Return the time-weighted totalized value of 'att1' calculated based on its unit between 12:00 and 1:00am today and convert to meters.]
- `TagTot('att1', 't', '+1h', "", 80)`

  [Return the time-weighted totalized value of 'att1' calculated based on its unit between 12:00 and 1:00am today when at least 80% of the values were good. Return error when minimum pctgood is not reached.]
- `TagTot('att1', 't', '+1h', "", 80, "TimeWeighted")`

  [Return the time-weighted totalized value of 'att1' calculated based on its unit between 12:00 and 1:00am today when at least 80% of the values were good. Return error when minimum pctgood is not reached.]
- `TagTot('att1', 't', '+1h', "m", 80)`

  [Return the time-weighted totalized value of 'att1' calculated based on its unit between 12:00 and 1:00am today and convert to meters when at least 80% of the values were good. Return error when minimum pctgood is not reached.]
- `TagTot('att1', 't', '+1h', "m", 80, "EventWeighted")`

  [Return the event-weighted totalized value of 'att1' calculated based on its unit between 12:00 and 1:00am today and convert to meters when at least 80% of the values were good. Return error when minimum pctgood is not reached.]

TagType

For an attribute associated with a PI point, get a point's **point type** from the point database.

Syntax

```
TagType(attname)
```

Arguments

attname

- The name of an attribute with a PI point data reference, enclosed in single quotes

Returns

Point type

Exceptions

If the point does not exist, returns an error

Example

- `TagType('sinusoid')`
  [Returns **Float32**]
- `TagType('cdm158')`
  [Returns **Digital**]

TagTypVal

For an attribute associated with a PI point, get a point's typical value from the point database.

Syntax

```
TagTypVal(attname)
```

Arguments

- **attname**
  The name of an attribute with a PI point data reference, enclosed in single quotes

Returns

The point's typical value. If the point is an integer or real number, a number is returned; if the point's value type is digital, this indicates a digital state

Exceptions

If the point does not exist, returns an error

Example

- `TagTypVal('sinusoid')`
  [Returns 50]
- `TagTypVal('cdm158')`
  [Returns 3, a digital state]

TagVal

Return the value of an attribute at a specified time.

Syntax

```
TagVal(attname [, t1])
```

Arguments

- **attname**
  The name of an attribute with a PI point data reference, enclosed in single quotation marks
- **t1**
  Optional: A time expression. If you omit this argument, '*' is used

Returns

The value of an attribute at a specified time. This value is interpolated unless the associated PI Point has the Step property set to True or the value type of the attribute cannot be interpolated (for example, if it is a string)

Exceptions

If the associated point does not exist or has no value at the specified time, an error is returned

Example

- `TagVal('att1')`
  [Return the value of 'att1' at current time]
- `TagVal('att1', 't+8h')`
  [Return the value of 'att1' at 8am today]
- `TagVal('enum_att1')`
  [Return the value of 'enum_att1', an attribute from an enumeration set at current time]

TagZero

For an attribute with a PI point data reference, get a PI point's "zero" value from the point database.

Syntax

```
TagZero(attname)
```

Arguments

- **attname**
  The name of an attribute with a PI point data reference, enclosed in single quotes

Returns

The point's zero value. If the point type is integer or real number, this is a number; if the point's type is digital, this is a digital state (character string)

Exceptions

If the point does not exist, returns an error

Example

- `TagZero('sinusoid')`
  [Returns 0]
- `TagZero('cdt158')`
  [Returns 50]

Tan

Return the trigonometric tangent of a number.

Syntax

```
Tan(x)
```

Arguments

- **x**
  Must be an integer or real number, which represents an angle in radians

Returns

The tangent of *x*.

Exceptions

If *x* is not a number, returns an error

Example

- `Tan(1)`
  [Returns 1.5574]
- `Tan(1.1)`
  [Returns 1.9648]
- `Tan('att1')`
  [Return the trigonometric tangent of the value of 'att1' at trigger time]

Tanh

Return the hyperbolic tangent of a number.

Syntax

```
Tanh(x)
```

Arguments

- **x**
  Must be an integer or real number

Returns

The hyperbolic tangent of *x*

Exceptions

If *x* is not a number, returns an error value

Example

- `Tanh(1)`
  [Returns 0.76159]
- `Tanh(1.1)`

[Returns 0.8005]

- `Tanh('att1')`

  [Return the hyperbolic tangent of the value of 'att1' at trigger time]

Text

Concatenate strings representing argument values.

Syntax

```
Text(x1 [, x2, … xn])
```

Arguments

- **x1 … xn**

  Any expression of normal PI value type

Returns

A string that is the concatenation of strings representing the argument values.

Example

- `Text(Round('sinusoid', 1), " is the current value of 'sinusoid' at ", '*')`

  [Returns rounded current value of 'sinusoid' at current time]

THEN

Operator that returns the first of two specified values when the conditional expression in IF-THEN-ELSE statement is True.

Syntax

```
IF(expression) THEN (x) ELSE (y)
```

Arguments

- **expression**

  Any expression that evaluates to true or false
- **x , y**

  An expression that evaluates to an output value

Returns

x when the conditional expression is true and *y* when the expression is false

Exceptions

None

Example

- `IF ('att1' > 50) THEN ('att2') ELSE ('att3')`
  [If the value of 'att1' is greater than 50, then return the value of 'att2'; otherwise return the value of 'att3']

TimeEq

Find the total time, within a range, when an attribute value is equal to a specified value. Time returned is in seconds.

Syntax

```
TimeEq(attname, starttime, endtime, x)
```

Arguments

- **attname**
  The name of an attribute with a PI point data reference, enclosed in single quotation marks
- **starttime**
  time expression representing the beginning of a time range enclosed in single quotes; can be a relative time (such as '-3h') in reference to an absolute *endtime*
- **endtime**
  time expression representing the end of a time range enclosed in single quotes; can be a relative time (such as '+1h') in reference to an absolute *starttime*
- **x**
  The reference value of the search; must be an enumeration set (string), integer or real number

Returns

The total time (in seconds), within a range, when an attribute value is equal to a given value

Notes

Bad values are excluded from **TimeEq** calculation

Example

- `TimeEq('att1', 't', '+1h', 80)`

  [Find the total time between 12:00 and 1:00am today when 'att1' was equal to 80. Return the result in seconds]
- `TimeEq('att1', '-1h', '*', TagVal('att1', 't+8h'))`

  [Find the total time in the past hour when the value of 'att1' was equal to the value at 8am today. Return the result in seconds]
- `TimeEq('Temperature Status', '-1h', '*', "Normal")`

Expression	**Output Attribute (attribute mapped to a PI point)**
If 'Temp' > 500 THEN "HiLimit" ELSE "Normal"	Temperature Status
TimeEq('Temperature Status', '-1h', '*', "Normal")	

  [Find the total time in the past hour when the value of the attribute 'Temperature Status' was equal to "Normal". Return the result in seconds]
- `TimeEq('enum_att1', '*-1d', '*', DigState("Normal", 'enum_att1'))`

  [Find the total time in the past 24 hours when the value of 'enum_att1' was equal to "Normal". Return the result in seconds]

TimeGE

Find the total time, within a range, when an attribute value is greater than or equal to a specified value. Time returned is in seconds.

Syntax

```
TimeGE(attname, starttime, endtime, x)
```

Arguments

- **attname**

  The name of an attribute with a PI point data reference, enclosed in single quotation marks
- **starttime**

  time expression representing the beginning of a time range enclosed in single quotes; can be a relative time (such as '-3h') in reference to an absolute *endtime*
- **endtime**

  time expression representing the end of a time range enclosed in single quotes; can be a relative time (such as '+1h') in reference to an absolute *starttime*
- **x**

  The reference value of the search; must be an enumeration set (string), integer or real number

Returns

The total time (in seconds), within a range, when an attribute value is greater than or equal to a given value

Exceptions

None

Notes

TimeGE interpolates between events, if necessary, to find the times when the attribute value crossed the given value

Bad values are excluded from **TimeGE** calculation

Example

- `TimeGE('att1', 't', '+1h', 80)`

  [Find the total time between 12:00 and 1:00am today when 'att1' was greater than or equal to 80]
- `TimeGE('att1', '-1h', '*', TagVal('att1', 't+8h'))`

  [Find the total time in the past hour when the value of 'att1' was greater than or equal to the value at 8am today. Result is in seconds]
- `TimeGE('enum_att1', '*-1d', '*', DigState("Normal", 'enum_att1'))`

TimeGT

Find the total time, within a range, when an attribute value is greater than a specified value. Time returned is in seconds.

Syntax

```
TimeGT(attname, starttime, endtime, x)
```

Arguments

- **attname**

  The name of an attribute with a PI point data reference, enclosed in single quotation marks
- **starttime**

  time expression representing the beginning of a time range enclosed in single quotes; can be a relative time (such as '-3h') in reference to an absolute *endtime*
- **endtime**

  time expression representing the end of a time range enclosed in single quotes; can be a relative time (such as '+1h') in reference to an absolute *starttime*
- **x**

The reference value of the search; must be an enumeration set (string), integer or real number

Returns

The total time (in seconds), within a range, when an attribute value is greater than a given value

Exceptions

None

Notes

TimeGT interpolates between events, if necessary, to find the times when the attribute value crossed the specified value

Bad values are excluded from **TimeGT** calculation

Example

- `TimeGT('att1', 't', '+1h', 80)`
  [Find the total time between 12:00 and 1:00am today when 'att1' was greater than 80]
- `TimeGT('att1', '-1h', '*', TagVal('att1', 't+8h'))`
  [Find the total time in the past hour when the value of 'att1' was greater than the value at 8am today. Result is in seconds]
- `TimeGT('enum_att1', '*-1d', '*', DigState("Normal", 'enum_att1'))`

TimeLE

Find the total time, within a range, when an attribute value is less than or equal to a given value. Time returned is in seconds.

Syntax

```
TimeLE(attname, starttime, endtime, x)
```

Arguments

- **attname**
  The name of an attribute with a PI point data reference, enclosed in single quotation marks
- **starttime**
  time expression representing the beginning of a time range enclosed in single quotes; can be a relative time (such as '-3h') in reference to an absolute *endtime*
- **endtime**
  time expression representing the end of a time range enclosed in single quotes; can be a relative time (such

as '+1h') in reference to an absolute *starttime*

- **x**

  The reference value of the search; must be an enumeration set (string), integer or real number

Returns

The total time (in seconds), within a range, when an attribute value is less than or equal to a given value

Exceptions

None

Notes

TimeLE interpolates between archive events, if necessary, to find the times when the point crossed the given value

Bad values are excluded from **TimeLE** calculation

Examples

- `TimeLE('att1', 't', '+1h', 80)`

  [Find the total time between 12:00 and 1:00am today when 'att1' was less than or equal to 80]
- `TimeLE('att1', '-1h', '*', TagVal('att1', 't+8h'))`

  [Find the total time in the past hour when the value of 'att1' was less than or equal its value at 8am today. Result is in seconds]
- `TimeLE('enum_att1', '*-1d', '*', DigState("Normal", 'enum_att1'))`

TimeLT

Find the total time, within a range, when an attribute value is less than a specified value. Time returned is in seconds.

Syntax

```
TimeLT(attname, starttime, endtime, x)
```

Arguments

- **attname**

  The name of an attribute with a PI point data reference, enclosed in single quotation marks
- **starttime**

  time expression representing the beginning of a time range enclosed in single quotes; can be a relative time (such as '-3h') in reference to an absolute *endtime*

- **endtime**

  time expression representing the end of a time range enclosed in single quotes; can be a relative time (such as '+1h') in reference to an absolute *starttime*
- **x**

  The reference value of the search; must be an enumeration set (string), integer or real number

Returns

The total time (in seconds), within a range, when an attribute value is less than a given value

Exceptions

None

Notes

TimeLT interpolates between Archive events, if necessary, to find the times when the attribute value crossed the specified value

Bad values are excluded from **TimeLT** calculation

Example

- `TimeLT('att1', 't', '+1h', 80)`

  [Find the total time between 12:00 and 1:00am today when 'att1' was less than 80]
- `TimeLT('att1', '-1h', '*', TagVal('att1', 't+8h'))`

  [Find the total time in the past hour when the value of 'att1' was less than its value at 8am today. Result is in seconds]
- `TimeLT('enum_att1', '*-1d', '*', DigState("Normal", 'enum_att1'))`

TimeNE

Find the total time, within a range, when an attribute value is not equal to a specified value. Time returned is in seconds.

Syntax

```
TimeNE(attname, starttime, endtime, x)
```

Arguments

- **attname**

  The name of an attribute with a PI point data reference, enclosed in single quotation marks
- **starttime**

time expression representing the beginning of a time range enclosed in single quotes; can be a relative time (such as '-3h') in reference to an absolute *endtime*

- **endtime**

  time expression representing the end of a time range enclosed in single quotes; can be a relative time (such as '+1h') in reference to an absolute *starttime*
- **x**

  The reference value of the search; must be an enumeration set (string), integer or real number

Returns

The total time (in seconds), within a range, when an attribute value is not equal to a specified value

Exceptions

None

Notes

Bad values are excluded from **TimeNE** calculation

Example

- `TimeNE('att1', 't', '+1h', 80)`

  [Find the total time between 12:00 and 1:00am today when 'att1' was not equal to 80]
- `TimeNE('att1', '-1h', '*', TagVal('att1', 't+8h'))`

  [Find the total time in the past hour when the value of 'att1' was not equal to the value at 8am today. Result is in seconds]
- `TimeNE('enum_att1', '*-1d', '*', DigState("Normal", 'enum_att1'))`

TimeStamp

Return the time stamp for a single time-stamped value or an array of time-stamped values.

Syntax

```
TimeStamp(x)
```

Arguments

- **x**

  A time-stamped value or an array of time-stamped values

Returns

The timestamp of the specified time-stamped value

Exceptions

If the value of *x* is *No Data*, then *Calc Failed* is returned

Example

- `TimeStamp(PrevVal('sinusoid', '*'))`

  [Return the time stamp of the last recorded value before the current one for 'sinusoid']
- `TimeStamp(Variable1)`

Name	Expression	Value at Evaluation
Variable1	RecordedValues('att1', '*-10m', '*')	[1, 3, 5]
Variable2	TimeStamp(Variable1)	[1/3/2018 8:15:00 AM, 1/3/2018 8:16:17 AM, 1/3/2018 8:20:00 AM]

  [Assuming current time is 8:23 AM on 1/3/2018, returns the timestamps of 3 recorded values in the past 10 minutes]

Total

Return the time-weighted or event-weighted sum of two or more values.

Syntax

```
Total(x1, x2 [, ... xn])
Total(array [, calculationBasis])
```

Arguments

- **x1, ... xn**

  Numbers, time intervals or a single array of numbers or time intervals
- **calculationBasis**

  Optional. The type of calculation to be performed enclosed in double quotes. Choose between "EventWeighted" or "TimeWeighted." If omitted, the default is event-weighted. If you select "TimeWeighted," the earliest timestamp of the value marks the start time and the latest timestamp the end of a time range

Returns

The total of the arguments. The result has the same data type as the arguments

Exceptions

Arguments whose run-time values are digital states are not included in the total. If all values are digital states, **Total** returns an error value. Returns an error if the array does not consist of same value type

Notes

- Unit of Measure (UOM) of the arguments is carried over to the result when at least one argument has a defined UOM while others don't. The returned value lacks a UOM if arguments have different UOMs

Example

- `Total('att1', 'att2', TagVal('att1', 'y+2h'), 40)`

  [Return the sum of following values: values of 'att1' and 'att2' at current time, the value of 'att1' at 2am yesterday and 40]
- `Total('t'-'y', '+1h')`

  [Returns 1.01:00:00]
- `Total(Variable1)`

Name	Expression	Value at Evaluation
Variable1	RecordedValues('att1', '*-10m', '*')	[1, 3, 5, 7, 9]
Variable 2	Total(Variable1)	25

  [Return the sum of values in an array named Variable1]
- `Total(Data)`

  [Return the event-weighted sum of values from an array named *Data*]
- `Total(Data, "EventWeighted")`

  [Return the event-weighted sum of values from an array named *Data*]
- `Total(Data, "TimeWeighted")`

  [Return the time-weighted sum of values from an array named *Data* using the timestamps of the earliest and latest values of the array as the start time and end time respectively]

Trim

Trim blanks on both sides of a string.

Syntax

```
Trim(s1)
```

Arguments

- **s1**
  string

Returns

The source string with leading and trailing blanks removed.

Exceptions

If *s1* is not a string, an error value is returned.

Example

- `Trim(" String ")`
  [Returns "String"]
- `Trim(" String is a string attribute. ")`
  [Returns "String is a string attribute."]

Year

Extract the year from a time expression.

Syntax

```
Year(t1)
```

Arguments

- **t1**
  A time expression, enclosed in single quotes

Returns

The year of time, in the range 1970-present

Exceptions

None

Example

- `Year('*')`
  [Return what year it is now]
- `Year(FindGT('att1', '1/1/1970', '*', 50))`
  [Return what year it was when the value of 'att1' was first greater than 50 since 1/1/1970]

Trunc

Truncate a number or time to the next lower unit.

Syntax

```
Trunc(x [, unit])
```

Arguments

- **x**
  An integer or real number, time expression, or time period.
- **unit**
  Optional. The size of the unit to truncate to; *x* will be truncated to a multiple of *unit*. If *x* is a number, *unit* must be a number. If *x* is a time expression or time period, *unit* must be a time period. If *unit* is omitted, **Trunc** truncates to the next lower integer (for a number) or second (for a time period).

Returns

The largest multiple of unit that is less than *x*. For a negative *x*, it returns the lowest multiple of *unit* larger than *x*. The return is the same data type as *x*.

Exceptions

If *x* is a string, or if unit is of the wrong data type, an error is returned.

Notes

If *x* is a time, and unit is omitted, this routine has no effect, as times are only accurate to one second.

When |x| < |unit|, 0 is returned.

Example

- `Trunc(12.999)`
  [Returns 12, truncated to the next lower integer]
- `Trunc(28.75, 10)`
  [Returns 20, truncated to next lower multiple of 10]
- `Trunc('14-Dec-16 11:47', '+1h')`
  [Returns 12/14/2016 11:00:00 AM, truncated to next lower hour]
- `Trunc('18:47'-'15:00','+1h')`
  [Returns 03:00:00, truncated period to next lower hour]

  Note: Truncating to the next lower day results in a timestamp of the next lower day in UTC time, not local time.

UCase

Convert a string to an uppercase string.

Syntax

```
UCase(s1)
```

Arguments

- **s1**
  String in double quotes

Returns

s1 in uppercase

Exceptions

If the argument is not a string, returns an error value.

Example

- `UCase("String")`
  [Returns "STRING"]

Weekday

Extract the day of the week from a given time expression.

Syntax

```
Weekday(t1)
```

Arguments

- **t1**
  A time expression, enclosed in single quotes

Returns

The day of the week, in the range 1-7, where 1 represents a Sunday

Exceptions

None

Example

- `Weekday('*')`
  [Return what day of the week today is]
- `Weekday(FindEq('att1', '-7d', '*', 50))`
  [Return what day of the week it was when the value of 'att1' was 50 for the first time in the past 7 days]

Yearday

Extract the day of the year from a time expression. The day of the year (also known as a Julian day) is an integer ranging from 1 to 366, where 1 represents January 1.

Syntax

```
Yearday(t1)
```

Arguments

- **t1**
  A time expression, enclosed in single quotes

Returns

The day of the year of time, in the range 1-366, where 1 represents January 1

Exceptions

None

Example

- `Yearday('*')`
  [Return what day of the year today is, where 1 represents January 1]
- `Yearday('y')`
  [Return what day of the year yesterday was, where 1 represents January 1]

Steam functions for analysis expressions

You can include steam functions in analysis expressions to calculate the thermodynamic properties of steam. These functions are based on the 1997 IAPWS Industrial Formulation. The implementation is based on the combination of officially published forward and backward equations for all applicable regions.

Note: Avoid confusing these steam functions with a similar set of steam functions used for PI Performance Equations in tag-based analytics. For more information on steam functions in asset and tag-based analytics, see Steam functions in asset and tag-based analytics.

The table below lists supported functions and descriptions. Ranges and units of measure for steam function properties are detailed in Ranges for steam function inputs. Reference states that apply to steam are listed in Steam property reference states.

Steam function descriptions

Function	Description
`Steam_TsatP`	Saturation temperature as a function of pressure.
`Steam_HsatP`	Saturation vapor specific enthalpy as a function of pressure.
`Steam_SsatP`	Saturation vapor specific entropy as a function of pressure.
`Steam_VsatP`	Saturation vapor specific volume as a function of pressure.
`Steam_PsatT`	Saturation pressure as a function of temperature.
`Steam_HsatT`	Saturation vapor specific enthalpy as a function of temperature.
`Steam_SsatT`	Saturation vapor specific entropy as a function of temperature.
`Steam_VsatT`	Saturation vapor specific volume as a function of temperature.

Function	Description
`Steam_VPT`	Vapor specific volume as a function of pressure and temperature.
`Steam_VPTL`	Liquid specific volume as a function of pressure and temperature. (For water.)
`Steam_VPH`	Vapor specific volume as a function of pressure and enthalpy.
`Steam_VPS`	Vapor specific volume as a function of pressure and entropy.
`Steam_HPT`	Vapor specific enthalpy as a function of pressure and temperature.
`Steam_HPTL`	Liquid specific enthalpy as a function of pressure and temperature. (For water.)
`Steam_HPS`	Vapor specific enthalpy as a function of pressure and entropy.
`Steam_SPT`	Vapor specific entropy as a function of pressure and temperature.
`Steam_SPTL`	Liquid specific entropy as a function of pressure and temperature. (For water.)
`Steam_SPH`	Vapor specific entropy as a function of pressure and enthalpy.
`Steam_TPH`	Temperature as a function of enthalpy and pressure.
`Steam_TPS`	Temperature as a function of entropy and pressure.
`Steam_XPH`	Steam quality (vapor fraction) as a function of pressure and enthalpy. (For wet steam.)
`Steam_XPS`	Steam quality (vapor fraction) as a function of entropy and pressure. (For wet steam.)
`Steam_VPX`	Steam specific volume as a function of pressure and steam quality. (For wet steam.)
`Steam_HPX`	Specific enthalpy as a function of pressure and steam quality. (For wet steam.)
`Steam_SPX`	Steam specific entropy as a function of pressure and steam quality. (For wet steam.)

Steam functions in asset and tag-based analytics

Asset and tag-based analytics tools both include steam functions that calculate thermodynamic properties of steam and water. Both sets of tools provide a complete set of steam functions but are based on different standards and have different implementations.

- Steam functions in asset analytics

  Steam functions in asset analytics are based on the 1997 IAPWS Industrial Formulation. The implementation is based on the combination of officially published forward and backward equations for all applicable regions.

  Note: The steam function names in asset analytics begin with "Steam_".

- Steam functions in tag-based analytics

  Steam functions in tag-based analytics refer to the PI PE Steam Functions Module (PI Steam), which is an extension to the PI Performance Equations Scheduler. These functions are based on the ASME Steam Tables, 6th Ed., and are also available in a COM library, **PISteamTableFunctions.dll**, which is distributed with our other products.

  Note: Tag-based steam functions support both English and SI units; the function names begin with "StmSI_" or "StmEng_" for SI units and English units respectively.

Ranges for steam function inputs

The table below shows ranges and default units of measure for each of the five properties involved in steam function calculations.

The default units of measure are SI units. For information about using steam functions with other units of measure, see Unit of measure conversion for steam functions.

Pressure affects the state of steam. Several things to keep in mind for functions with pressure as an input are described in Considerations for steam functions that have pressure as an input.

Range of inputs for steam function properties

Function	Temp (°C)	Pressure (kPa)	Enthalpy(J/g)	Entropy(J/(g K))	Quality
`Steam_TsatP`		0.611212678 to 22064			
`Steam_HsatP`		0.611212678 to 22064			
`Steam_SsatP`		0.611212678 to 22064			
`Steam_VsatP`		0.611212678 to 22064			
`Steam_PsatT`	0.0 to 373.946				
`Steam_HsatT`	0.0 to 373.946				

Function	Temp (°C)	Pressure (kPa)	Enthalpy(J/g)	Entropy(J/(g K))	Quality
`Steam_SsatT`	0.0 to 373.946				
`Steam_VsatT`	0.0 to 373.946				
`Steam_VPT`	0.0 to 800	0.611212678 to 100000			
`Steam_VPT (high temperature and pressure)`	800 to 2000	0.611212678 to 50000			
`Steam_VPTL`	0.0 to 800	0.611212678 to 100000			
`Steam_VPH`		0.611212678 to 100000	-0.04159 to 4160.7		
`Steam_VPS`		0.611212678 to 100000		-0.000155 to 11.921	
`Steam_HPT`	0.0 to 800	0.611212678 to 100000			
`Steam_HPT (high temperature and pressure)`	800 to 2000	0.611212678 to 50000			
`Steam_HPTL`	0.0 to 800	0.611212678 to 100000			
`Steam_HPS`		0.611212678 to 100000		-0.000155 to 11.921	
`Steam_SPT`	0.0 to 800	0.611212678 to 100000			
`Steam_SPT (high temperature and pressure)`	800 to 2000	0.611212678 to 50000			
`Steam_SPTL`	0.0 to 800	0.611212678 to 100000			
`Steam_SPH`		0.611212678 to 100000	-0.04159 to 4160.7		

Function	Temp (°C)	Pressure (kPa)	Enthalpy(J/g)	Entropy(J/(g K))	Quality
`Steam_TPH`		0.611212678 to 100000	-0.04159 to 4160.7		
`Steam_TPS`		0.611212678 to 100000		-0.000155 to 11.921	
`Steam_XPH`		0.611212678 to 22064	-0.04159 to 4160.7		
`Steam_XPS`		0.611212678 to 22064		-0.000155 to 11.921	
`Steam_VPX`		0.611212678 to 22064			0.0 to 1.0
`Steam_HPX`		0.611212678 to 22064			0.0 to 1.0
`Steam_SPX`		0.611212678 to 22064			0.0 to 1.0

Considerations for steam functions that have pressure as an input

Because pressure affects the state of steam, be aware of these considerations for functions that have pressure as an input.

Functions with pressure and temperature as inputs

Steam_HPT, **Steam_SPT** and **Steam_VPT** can be used for compressed water, superheated or saturated dry steam. For input temperature and pressure on the saturated curve, the calculated entropy, enthalpy or volume is a property of saturated vapor. These three functions can also accept high-range input temperatures (800 to 2000 °C) with pressures from 0.611212678 to 50000 kPa.

Steam_HPTL, **Steam_SPTL** and **Steam_VPTL** are used only for compressed water.

Functions with pressure and enthalpy or entropy as inputs

Steam_VPH, **Steam_VPS**, **Steam_HPS**, **Steam_SPH**, **Steam_TPH** and **Steam_TPS** can be used for compressed water, superheated or wet steam. For these functions, the range for enthalpy (H) is -0.04159 to 4160.7 J/g and for entropy (S) is -0.000155 to 11.921 J/(g K).

For **Steam_XPH** and **Steam_XPS**, input enthalpy or entropy greater than saturated vapor properties or less than saturated liquid properties returns an error.

Unit of measure conversion for steam functions

The default units of measure (UOM) for the steam functions are SI units. Any inputs to the steam functions defined in English units are automatically converted to the corresponding SI units. Similarly, if a steam function is configured with an output attribute, its output is automatically converted to the UOM configured for that attribute.

Note: Automatic conversion requires that all UOM have been defined properly in the UOM database. If the definition of conversion from one unit to another of the same type is missing from the UOM database, conversion will fail.

In the expression editor, when you evaluate a steam function expression, results are displayed in the default SI units. To display the results in other UOM, you can use the **Convert** function.

The following example expressions use numeric values, variables, and attributes to show how steam functions handle UOM for input and output quantities.

Numeric input values

Numeric input values are recognized as quantities in the default UOM. In the following example for **Steam_VPT**, the numeric inputs are recognized as 300 kPa and 200 °C, and the output value is in cm^3/g, the default UOM for specific volume.

Name	Expression	Value	Output Attribute
Variable1	*Steam_VPT(300, 200)*	*716.44 cm3/g*	*Click to map*

Input and output attributes

In the following expression, **Steam_VsatT** takes as its only input the attribute 'TempSat', whose current value is 300 °F. The temperature is automatically converted to °C to evaluate the expression, and the result is displayed in the default SI UOM for **Steam_VsatT**.

Name	Expression	Value	Output Attribute
Variable1	*Steam_VsatT('TempSat')*	*403.7 cm3/g*	*OutputVolume*

The result is also mapped to the 'OutputVolume' output attribute. This attribute is configured in English UOM (ft^3/lb), so the result is automatically converted to that UOM. (On the Attributes tab in PI System Explorer, 'OutputVolume' would be listed with a value of 6.466627 ft^3/lb.)

Variables and the Convert function

This example uses variables and the **Convert** function to specify quantities in English UOM as steam function inputs.

1. In the P and T rows of the expression below, **Convert** assigns numeric values to English UOM for pressure and temperature.
2. In the B row, the variables P and T are automatically converted to SI units for the **Steam_VPT** calculation; the result is displayed in default SI UOM. The result is also mapped to the output attribute 'OutputVolume'. That attribute is configured in English UOM (ft^3/lb), so the result is automatically converted to that UOM.
3. In the last row, **Convert** is used to convert the result (B) to English UOM and display it in the expression.

Name	Expression	Value	Output Attribute
P	*Convert(2000, "psia")*	*2000 psia*	*Click to map*
T	*Convert(800, "°F")*	*800 °F*	*Click to map*
B	*Steam_VPT(P, T)*	*19.206 cm3/g*	*OutputVolume*
Volume	*Convert(B, "ft3/lb")*	*0.30762 ft3/lb*	*DisplayedVolume*

Steam property reference states

The American Society of Mechanical Engineers (ASME) establishes the following reference states:

- **triple point**

  Triple point of water is at 273.16 K or 0.01 °C or 32.018 °F.
- **Celsius scale**

  The Celsius temperature is exactly T_k - 273.15.
- **critical point**

  Critical point of steam is at 647.096 K and 22,064 kPa, or 705.103 °F and 3200.1 psia.
- **reference state**

  The specific internal energy and specific entropy of the liquid phase were fixed at zero at the triple point of water.

Steam functions reference (asset analytics)

Steam functions available for use in expression analyses are listed alphabetically.

Steam_HPS

Calculate the vapor specific enthalpy as a function of pressure (*P* in kPa) and entropy (*S* in J/(g K)). Use for both superheated and wet steam as well as compressed water. By default, input arguments and returned values are in SI units. You can change the units of measure, for example, by using the `Convert` function.

Syntax

```
Steam_HPS(P, S)
```

Arguments

- **P**

  Pressure of the steam. *P* can range from 0.611212678 to 100000 kPa.
- **S**

Specific entropy of the steam. *S* can range from -0.000155 to 11.921J/(g K).

Returns

Computed vapor specific enthalpy in J/g.

Example

- `Steam_HPS(2500, 7.5956)`
  [Returns 3684.9 J/g]

Steam_HPT

Calculate the vapor specific enthalpy as a function of pressure (*P* in kPa) and temperature (*T* in °C). Use for an entire range from compressed water to superheated steam. By default, input arguments and returned values are in SI units. You can change the units of measure, for example, by using the `Convert` function.

Syntax

```
Steam_HPT(P, T)
```

Arguments

- **P**
  Pressure of the steam. *P* can range from 0.611212678 to 100000 kPa.
- **T**
  Temperature of the steam. *T* can range from 0.0 to 800 °C.

Returns

Computed vapor specific enthalpy in J/g.

Example

- `Steam_HPT(40000, 600)`
  [Returns 3350.4 J/g]

Steam_HPTL

Calculate the liquid specific enthalpy as a function of pressure (*P* in kPa) and temperature (*T* in °C). Only use for compressed water. For any *T* greater than the saturation temperature for *P*, the function returns an out-of-range error. The function accepts any *T* in the temperature range as long as *P* is greater than the critical pressure. By default, input arguments and returned values are in SI units. You can change the units of measure, for example, by using the `Convert` function.

Syntax

```
Steam_HPTL(P, T)
```

Arguments

- **P**
  Pressure of the steam. *P* can range from 0.611212678 to 100000 kPa.
- **T**
  Temperature of the steam. *T* can range from 0.0 to 800 °C.

Returns

Computed liquid specific enthalpy in J/g.

Example

- `Steam_HPTL(40000, 100)`
  [Returns 449.27 J/g]

Steam_HPX

Calculate the specific enthalpy as a function of pressure (*P* in kPa) and quality (vapor fraction). Use only for wet steam. By default, input arguments and returned values are in SI units. You can change the units of measure, for example, by using the `Convert` function.

Syntax

```
Steam_HPX(P, X)
```

Arguments

- **P**
  Pressure of the steam. *P* can range from 0.611212678 to 22064 kPa.
- **X**
  Steam quality (vapor fraction). *X* can range from 0.0 to 1.0.

Returns

Computed specific enthalpy in J/g.

Example

- `Steam_HPX(10000, 0.9)`
  [Returns 2593.7 J/g]

Steam_HsatP

Calculate the saturated vapor specific enthalpy as a function of pressure (*P* in kPa). By default, input arguments and returned values are in SI units. You can change the units of measure, for example, by using the `Convert` function.

Syntax

```
Steam_HsatP(P)
```

Arguments

- **P**
  Pressure of the steam. *P* can range from 0.611212678 to 22064 kPa.

Returns

Computed specific enthalpy for saturated vapor in J/g.

Example

- `Steam_HsatP(10000)`
  [Returns 2725.5 J/g]

Steam_HsatT

Calculate the saturated vapor specific enthalpy as a function of temperature (*T* in °C). By default, input arguments and returned values are in SI units. You can change the units of measure, for example, by using the `Convert` function.

Syntax

```
Steam_HsatT(T)
```

Arguments

- **T**
  Temperature of the steam. *T* can range from 0.0 to 373.946 °C.

Returns

Computed saturated vapor specific enthalpy in J/g.

Example

- `Steam_HsatT(250)`
  [Returns 2801 J/g]

Steam_PsatT

Calculate the saturation pressure as a function of temperature (*T* in °C). By default, input arguments and returned values are in SI units. You can change the units of measure, for example, by using the `Convert` function.

Syntax

```
Steam_PsatT(T)
```

Arguments

- **T**
  Temperature of the steam. *T* can range from 0.0 to 373.946 °C.

Returns

Computed saturation pressure of the steam in kPa.

Example

- `Steam_PsatT(250)`
  [Returns 3975.9 kPa]

Steam_SPH

Calculate the vapor-specific entropy as a function of pressure (*P* in kPa) and enthalpy (*H* in J/g). Use for both superheated and wet steam as well as compressed water. By default, input arguments and returned values are in SI units. You can change the units of measure, for example, by using the `Convert` function.

Syntax

```
Steam_SPH(P, H)
```

Arguments

- **P**
  Pressure of the steam. *P* can range from 0.611212678 to 100000 kPa.
- **H**
  Enthalpy of the steam. *H* can range from -0.04159 to 4160.7 J/g.

Returns

Computed vapor specific entropy in J/(g K).

Example

- `Steam_SPH(10000, 3500)`
  [Returns 6.756 J/(g K)]

Steam_SPT

Calculate the vapor specific entropy as a function of pressure (*P* in kPa) and temperature (*T* in °C). Use for an entire range from compressed water to superheated steam. By default, input arguments and returned values are in SI units. You can change the units of measure, for example, by using the `Convert` function.

Syntax

```
Steam_SPT(P, T)
```

Arguments

- **P**
  Pressure of the steam. *P* can range from 0.611212678 to 100000 kPa.
- **T**
  Temperature of the steam. *T* can range from 0.0 to 800 °C.

Returns

Computed vapor specific entropy in J/(g K).

Example

- `Steam_SPT(40000, 600)`
  [Returns 6.017 J/(g K)]

Steam_SPTL

Calculate the liquid specific entropy as a function of pressure (*P* in kPa) and temperature (*T* in °C). Only use for compressed water. For any *T* higher than the saturation temperature for *P*, the function returns an out-of-range error. The function accepts any *T* in the temperature range as long as *P* is greater than the critical pressure. By default, input arguments and returned values are in SI units. You can change the units of measure, for example, by using the `Convert` function.

Syntax

```
Steam_SPTL(P, T)
```

Arguments

- **P**
  Pressure of the steam. *P* can range from 0.611212678 to 100000 kPa.
- **T**
  Temperature of the steam. *T* can range from 0.0 to 800 °C.

Returns

Computed liquid specific entropy in J/(g K).

Example

- `Steam_SPTL(30000, 500)`
  [Returns 5.7956 J/(g K)]

Steam_SPX

Calculate the steam specific entropy as a function of pressure (*P* in kPa) and quality (vapor fraction). Use for only wet steam. By default, input arguments and returned values are in SI units. You can change the units of measure, for example, by using the `Convert` function.

Syntax

```
Steam_SPX(P, X)
```

Arguments

- **P**
  Pressure of the steam. *P* can range from 0.611212678 to 22064 kPa.
- **X**
  Steam quality (vapor fraction). *X* can range from 0.0 to 1.0.

Returns

Computed specific entropy of the steam in J/(g K)

Example

- `Steam_SPX(15000, 0.7)`
  [Returns 4.8229 J/(g K)]

Steam_SsatP

Calculate the saturated vapor specific entropy as a function of pressure (*P* in kPa). By default, input arguments and returned values are in SI units. You can change the units of measure, for example, by using the `Convert` function.

Syntax

```
Steam_SsatP(P)
```

Arguments

- **P**
  Pressure of the steam. *P* can range from 0.611212678 to 22064 kPa.

Returns

Computed saturated vapor specific entropy in J/(g K)

Example

- `Steam_SsatP(10000)`
  [Returns 5.6159 J/(g K)]

Steam_SsatT

Calculate the saturated vapor specific entropy as a function of temperature (*T* in °C). By default, input arguments and returned values are in SI units. You can change the units of measure, for example, by using the `Convert` function.

Syntax

```
Steam_SsatT(T)
```

Arguments

- **T**
  Temperature of the steam. *T* can range from 0.0 to 373.946 °C.

Returns

Computed saturated vapor specific entropy in J/(g K).

Example

- `Steam_SsatT(250)`
  [Returns 6.0722 J/(g K)]

Steam_TPH

Calculate the steam temperature as a function of pressure (*P* in kPa) and enthalpy (*H* in J/g). Use for both superheated and wet steam as well as compressed water. By default, input arguments and returned values are in SI units. You can change the units of measure, for example, by using the `Convert` function.

Syntax

```
Steam_TPH(P, H)
```

Arguments

- **P**
  Pressure of the steam. *P* can range from 0.611212678 to 100000 kPa.
- **H**
  Enthalpy of the steam. *H* can range from -0.04159 to 4160.7 J/g.

Returns

Computed steam temperature in °C.

Example

- `Steam_TPH(40000, 3500)`
  [Returns 643.48 °C]

Steam_TPS

Calculate the steam temperature as a function of pressure (*P* in kPa) and entropy (*S* in J/(g K)). Use for both superheated and wet steam as well as compressed water. By default, input arguments and returned values are in

SI units. You can change the units of measure, for example, by using the `Convert` function.

Syntax

```
Steam_TPS(P, S)
```

Arguments

- **P**
  Pressure of the steam. *P* can range from 0.611212678 to 100000 kPa.
- **S**
  Specific entropy of the steam. *S* can range from -0.000155 to 11.921J/(g K).

Returns

Computed steam temperature in °C.

Example

- `Steam_TPS(40000, 6)`
  [Returns 595.93 °C]

Steam_TsatP

Calculate the saturation temperature as a function of pressure (*P* in kPa). By default, input arguments and returned values are in SI units. You can change the units of measure, for example, by using the `Convert` function.

Syntax

```
Steam_TsatP(P)
```

Arguments

- **P**
  Pressure of the steam. *P* can range from 0.611212678 to 22064 kPa.

Returns

Computed saturation temperature in °C.

Example

- `Steam_TsatP(10000)`
  [Returns 311 °C]

Steam_VPH

Calculate the vapor specific volume as a function of pressure (*P* in kPa) and enthalpy (*S* in J/g). Use for both superheated and wet steam as well as compressed water. By default, input arguments and returned values are in SI units. You can change the units of measure, for example, by using the `Convert` function.

Syntax

```
Steam_VPH(P, H)
```

Arguments

- **P**

  Pressure of the steam. *P* can range from 0.611212678 to 100000 kPa.
- **H**

  Enthalpy of the steam. *H* can range from -0.04159 to 4160.7 J/g.

Returns

Computed vapor specific volume in cm^3/g.

Example

- `Steam_VPH(25000, 3500)`

  [Returns 14.197 cm^3/g]

Steam_VPS

Calculate the vapor specific volume as a function of pressure (*P* in kPa) and entropy (*S* in J/(g K)). Use for both superheated and wet steam as well as compressed water. By default, input arguments and returned values are in SI units. You can change the units of measure, for example, by using the `Convert` function.

Syntax

```
Steam_VPS(P, S)
```

Arguments

- **P**

  Pressure of the steam. *P* can range from 0.611212678 to 100000 kPa.
- **S**

  Specific entropy of the steam. *S* can range from -0.000155 to 11.921J/(g K).

Returns

Computed vapor specific volume in cm^3/g

Example

- `Steam_VPS(40000, 6)`

  [Returns 8.0055 cm^3/g]

Steam_VPT

Calculate the vapor specific volume as a function of pressure (*P* in kPa) and temperature (*T* in °C). Use for an entire range from compressed water to superheated steam. You can change the units of measure, for example, by using the `Convert` function.

Syntax

```
Steam_VPT(P, T)
```

Arguments

- **P**

  Pressure of the steam. *P* can range from 0.611212678 to 100000 kPa.
- **T**

  Temperature of the steam. *T* can range from 0.0 to 800 °C.

Returns

Computed vapor specific volume in cm^3/g.

Example

- `Steam_VPT(50000, 500)`

  [Returns 3.8894 cm^3/g]

Steam_VPTL

Calculate the liquid specific volume as a function of pressure (*P* in kPa) and temperature (*T* in °C). For any *T* higher than the saturation temperature for *P*, the function returns an out-of-range error. The function accepts any *T* in the temperature range as long as *P* is greater than the critical pressure. By default, input arguments and returned values are in SI units. You can change the units of measure, for example, by using the `Convert` function.

Syntax

```
Steam_VPTL(P, T)
```

Arguments

- **P**

  Pressure of the steam. *P* can range from 0.611212678 to 100000 kPa.
- **T**

  Temperature of the steam. *T* can range from 0.0 to 800 °C.

Returns

Computed liquid specific volume in cm^3/g

Example

- `Steam_VPTL(75000, 500)`

  [Returns 2.308 cm^3/g]

Steam_VPX

Calculate the steam specific volume as a function of pressure (*P* in kPa) and quality (vapor fraction). Use for only wet steam. By default, input arguments and returned values are in SI units. You can change the units of measure, for example, by using the `Convert` function.

Syntax

```
Steam_VPX(P, X)
```

Arguments

- **P**

  Pressure of the steam. *P* can range from 0.611212678 to 22064 kPa.
- **X**

  Steam quality (vapor fraction). *X* can range from 0.0 to 1.0.

Returns

Computed steam specific volume in cm^3/g

Example

- `Steam_VPX(15000, 0.7)`
  [Returns 7.7352cm^3/g]

Steam_VsatP

Calculate the saturated vapor specific volume as a function of pressure (*P* in kPa). By default, input arguments and returned values are in SI units. You can change the units of measure, for example, by using the `Convert` function.

Syntax

```
Steam_VsatP(P)
```

Arguments

- **P**
  Pressure of the steam. *P* can range from 0.611212678 to 22064 kPa.

Returns

Computed vapor specific volume in cm^3/g

Example

- `Steam_VsatP(10000)`
  [Returns 18.034 cm^3/g]

Steam_VsatT

Calculate the saturated vapor specific volume as a function of temperature (*T* in °C). By default, input arguments and returned values are in SI units. You can change the units of measure, for example, by using the `Convert` function.

Syntax

```
Steam_VsatT(T)
```

Arguments

- **T**
  Temperature of the steam. *T* can range from 0.0 to 373.946 °C.

Returns

Computed saturated vapor specific volume in cm^3/g.

Example

- `Steam_VsatT(250)`

  [Returns 50.087 cm^3/g]

Steam_XPH

Calculate the steam quality (vapor fraction) as a function of pressure (*P* in kPa) and enthalpy (*H* in J/g). Use only for wet steam. By default, input arguments and returned values are in SI units. You can change the units of measure, for example, by using the `Convert` function.

Syntax

```
Steam_XPH(P, H)
```

Arguments

- **P**

  Pressure of the steam. *P* can range from 0.611212678 to 22064 kPa.
- **H**

  Enthalpy of the steam. *H* can range from -0.04159 to 4160.7 J/g.

Returns

Computed steam quality (vapor fraction)

Example

- `Steam_XPH(10000, 2500)`

  [Returns 0.82888]

Steam_XPS

Calculate the steam quality (vapor fraction) as a function of pressure (*P* in kPa) and entropy (*S* in J/(g K)). Use only for wet steam. By default, input arguments and returned values are in SI units. You can change the units of measure, for example, by using the `Convert` function.

Syntax

```
Steam_XPS(P, S)
```

Arguments

- **P**
  Pressure of the steam. *P* can range from 0.611212678 to 22064 kPa.
- **S**
  Specific entropy of the steam. *S* can range from -0.000155 to 11.921J/(g K).

Returns

Computed steam quality (vapor fraction)

Example

- `Steam_XPS(10000, 5)`
  [Returns 0.72695]

Notifications

Notifications is the feature in Asset Framework (PI AF) that you use to create and manage notification rules. Notification rules are the mechanism by which users are alerted for response in real time, for conditions that signal events of interest. Notification messages may be customized to include information that is specifically relevant to the event, and may be sent via email to individuals, groups, or to a web service. Further, escalations are also configurable in the notification system.

You can use PI System Explorer to configure and manage notification rules from both the Notification Rules tab (visible after selecting an element), and from the Management plug-in.

Note: The audit log is accessed by right-clicking a notification rule and then selecting Audit Trail Events. For more information on the audit trail feature, see Track PI AF changes with Audit Trail.

Introduction to notifications

Notification rules: Definition and scope

Notifications is included as a feature of PI Server, starting with the 2016 R2 version, and is not installed as a separate product. The notifications feature allows you to configure "notification rules" to send email messages or to call a web service; you can also configure message content and set up escalations.

Note: Notifications created using PI Notifications 2012 or earlier are henceforth referred to as "legacy notifications".

Notifications are alerts that are generated when an event of interest is detected in the PI Server; such events are detected by examining and comparing real-time and static data with user-defined conditions configured in the PI Server. User-specified information can be configured in notifications that are sent to users or groups in real time to alert users of conditions to which they may need to respond. Notifications integrate with, and leverage, PI AF and other PI System services, and can be shared, managed, and maintained enterprise-wide. All notification actions, such as notification send times, acknowledgments, entry of comments, and escalations, are stored for later retrieval and examination.

Notification rules are based on event frames and notification messages can include information from event frame templates. For more information, see Notifications and event frames.

PI Notifications Service

PI Notifications Service is a Windows service that evaluates notification rules, processes incoming real-time event frames, and sends notifications. The PI Notifications Service need not run on the same machine as the Data Archive, PI AF server, or the client applications.

For information on event frames, see Event frames in PI AF.

Training video

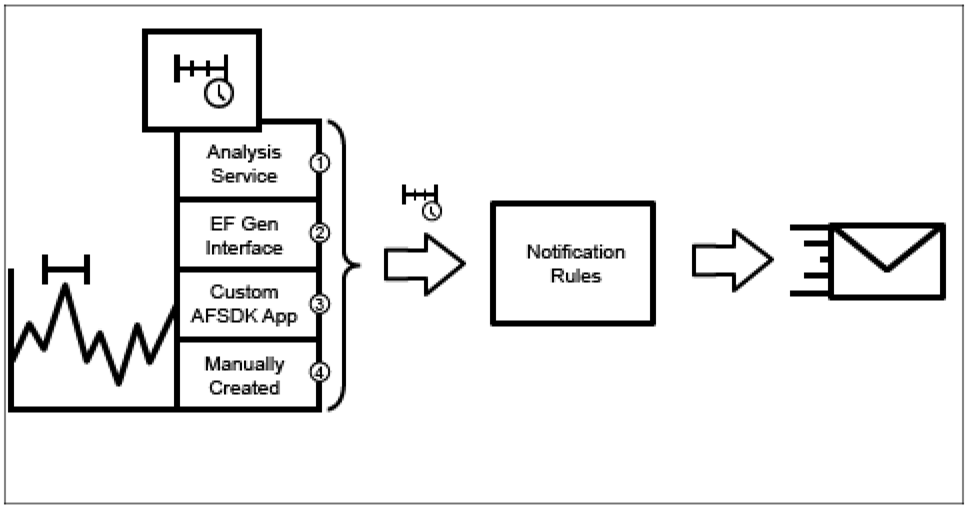
For more information on using notifications in PI Asset Framework, watch the videos on the AVEVA PI System Learning channel playlist:

Notifications and event frames

You can create notification rules based on any generated event frames (regardless of the method used to create them). The notification rules will be triggered when the configured condition or event is detected in the system or process, and a customized notification message is sent to a user, a group, or a web service.

Notification triggers are based on event frames; the event frames may come from different sources like analyses, the Event Frame Generator (EFGen) interface or custom AF SDK applications. Relevant information for events are stored in event frames and compatible tools can retrieve such information. Information contained in elements, event frame templates, and other places may also be added to the notification content. Further, event frame annotations contain historical notification information and may also include attachments; all annotations may be viewed using AVEVA PI Vision or other client tools that support viewing annotations.

Relationship between event frames and notification rules



- 1. Analysis-generated event frames. For more information, see Event frame generation analyses to create analyses from the **Analyses** tab, and Trigger criteria in notification rules to create analyses from the **Notification Rules** tab of PI System Explorer.
- 2. Event Frames Generator (EFGen) interface-generated event frames. See Introduction to PI EFGen.
- 3. Custom AF SDK applications-generated event frames. See the AF SDK documentation.
- 4. Manually created event frames.

Notifications history

Information associated with previous times that a notification was generated is stored as annotations on each event frame that triggered the notification. The annotations can be accessed by selecting the specific event frame of interest from the Event Frames plug-in on PI System Explorer. Each event frame annotation includes such information as the notification rule corresponding to the send event, the subscribers who were notified and when they were notified, and whether the attempt to notify was successful.

If you have migrated from PI Notifications 2012, and elected to migrate both configuration and history, your legacy notifications history is fully migrated; the historical information is migrated as annotations on the event frames created during migration. The migration report includes information on the legacy history instances that were processed during migration.

Note: For information on permissions required during migration, see Migration from PI Notifications 2012.

Event frame annotations may also be viewed using AVEVA PI Vision or other client tools that support viewing annotations.

Acknowledge a notification

In PI AF 2016, you can require that an event frame be acknowledged. The acknowledgment feature is used by notifications and PI AF client applications, such as AVEVA PI Vision. The event-frame acknowledgment feature replaces the acknowledge notification functionality in legacy PI Notifications 2012 and older versions.

For more information on acknowledging event frames in PI AF, see Acknowledgment of event frames and Acknowledge event frames.

For more information on viewing and acknowledging event frames using AVEVA PI Vision, see View event details and annotate events.

Requirements for notifications

If you are connecting to a claims-enabled AVEVA PI Vision server, both PI AF Server 2024 and PI Notifications Service 2024 are required. In addition, TLS certificates and OpenID Connect (OIDC) must also be enabled and configured to implement claims-based authentication.

To use the notifications feature, you must install these PI Server components in the following order:

1. Either PI AF Client 2017 R2 or later or PI AF Client 2023 with OIDC enabled if you want to include screenshots from a claims-enabled AVEVA PI Vision server
2. PI Notifications Service 2017 R2 or later; PI Notifications Service 2024 with OIDC enabled is required to include screenshots from a claims-enabled AVEVA PI Vision server
3. Optional. AVEVA PI Vision.
   - This is required if you want to use the acknowledgment link in the notifications feature, which is a link to the event details page in AVEVA PI Vision.
   - To include screenshots in notifications from an AVEVA PI Vision server that is configured for Windows authentication, NTLM provider for Windows Authentication needs to be enabled in AVEVA PI Vision. For more information, see Delivery formats in notification rules. Also see the AVEVA PI Vision user guide topics Enable Kerebos delegation using a custom PI Vision service account and Enable basic authentication.
4. Management plug-in from PI AF Client 2017 R2 or later, which allows for managing analyses and notification rules in bulk.

Client support for notifications

AVEVA PI Vision and PI DataLink 2016 and later versions support notifications with the ability to display event frames and associated data. Notification messages can include links and screenshots to pre-configured AVEVA PI Vision and thereby provide visual representation of the event that triggered the notification. See AVEVA PI Vision or PI DataLink product documentation for details on notifications feature support.

Notifications security

The notifications feature uses Windows security for communicating internally and with the PI AF server.

Note: See the PI Server topic PI AF objects and local permissions for PI Notifications Service for information on PI Notifications Service account permissions.

You can secure access to notification rules, notification rule templates, contacts, and contact templates. Object permissions are granted according to the PI AF Security model.

If Active Directory (AD) access is configured, information from the AD is accessed to create individual contacts. You cannot delete an AD contact from PI System Explorer, and you cannot modify security settings for the contact. To configure AD access, see the PI Server topic Configure Active Directory objects for delegation.

If you are using a screenshot feature in your email delivery format, see the security guidelines in the PI Server Installation and Upgrade guide topic Trusted sites for screenshots. For more information, see also Add a screenshot in notification email delivery format and Add a domain to the list of trusted sites.

In line with PI Asset Framework, Notifications supports Windows environments that enforce the use of U.S. Federal Information Processing Standards (FIPS) cryptographic algorithms. For more information, see **Support for FIPS** section in PI System Explorer and Microsoft Support article "System cryptography: Use FIPS compliant algorithms for encryption, hashing, and signing" security setting effects in Windows XP and in later versions of Windows.

Security for contacts

Microsoft Windows security is used to control access to groups, escalation teams, and custom, non-Active Directory contacts. Security settings for Active Directory contacts cannot be changed using PI System Explorer.

Newly created contact objects will inherit default permissions from the parent collections; a newly created custom contact will have security defined by the server's contacts collection.

Summary of permissions for contacts

Windows security must be set appropriately for contacts to view, modify, delete, or assign privileges to other contacts.

Permission	Description
Read	Can view information on contacts.
Write	Can modify contacts.
Delete	Can delete contacts.
Admin	Can assign privileges to contacts.

Edit security for existing contacts

Edit security settings from the **Contacts** plug-in to allow desired access to contacts. Administrator access is required to edit contact security. See Summary of permissions for contacts.

Note: You can also edit security configurations for contacts by navigating to **Database** > **Edit Security**, and then selecting the appropriate **Notification Contact Templates** for your PI AF server (from the **Item** column). For example, if your PI AF Server is named MyServer, you must select **MyServer-Notification Contact Templates**.

1. In PI System Explorer, click **Contacts** in the navigator pane.
2. Search for the group, escalation team or delivery endpoint whose permissions you want to change.

   You may need to double-click the contact to see its delivery endpoints.
3. Right-click the contact's delivery endpoint and select **Security**.

   The **Security Configuration** window for the delivery endpoint appears.
4. Change security settings for the delivery endpoint.

   Permissions can be changed for all identities that have been configured. You can also map new identities and configure their permissions.

Security for notification rules and templates

Notifications uses Microsoft Windows security to control access to notification rules and templates. To edit a notification rule or template, you need write access to notification rules.

Summary of permissions for notification rules and templates

Windows security must be set appropriately to perform specific tasks with notifications.

Permission	Description
Read	Can view a notification rule.
Write	Can modify settings in the Trigger Criteria, Subscriptions, and Formats panes.
Subscribe	Can subscribe oneself to a notification and customize the subscription.
SubscribeOthers	Can subscribe others to a notification and customize their subscriptions.
Delete	Can delete a notification rule.
Execute	Can start, stop or reset a notification rule.
Admin	Can assign privileges to notification rules.

Write permission at the PI AF server level is needed for global formats. See Configure security for a PI AF server for more information.

Edit security for a notification rule or template

Edit security settings to allow desired access to notification rules and templates.

Note: Administrator access is required to edit notification rule and template security. See Summary of permissions for notification rules and templates.

1. In PI System Explorer, navigate to **Database** > **Edit Security**.

   You see the **Security Configuration** window.
2. From the **Item** column, select the appropriate notification rule or notification rule template for your PI AF server.

   For example, on a PI AF server named "MyServer", you must select:
   - **MyServer-Database-Notification Rule** to modify notification rule permissions
   - **MyServer-Database-Notification Rule Template** to modify notification rule template permissions.
3. Change security settings for notification rules or templates.

   Permissions can be changed for all identities that have been configured. You can also map new identities and configure their permissions.

Configure authentication for SMTP server connection

Best practice is to use a secure connection when using basic authentication. To use a secure connection with an SMTP server, enable the Use TLS option. The server must present a valid certificate for a secure connection to be established. To configure authentication for SMTP server connection, follow this procedure.

1. To start PI System Explorer, click **Start** > **Programs** > **PI System** > **PI System Explorer**.
2. Click on the PI System Explorer toolbar, or click **File** > **Server Properties**.
3. Click the **Plug-Ins** tab.
4. Scroll down to the **Delivery Channel Plug-Ins** section, then right-click **Email** and select **Settings**.
5. In the Email Delivery Channel Configuration window, click **Authentication Options** link under **SMTP Server**.

   Note: The same can be done for the Backup SMTP Server by clicking the link by the same name under it.

   Primary SMTP Server Authentication Options window opens.
6. Choose among the 3 authentication options.
   - Windows: PI Notifications Service will pass the network credentials of its service account to the SMTP server
   - Anonymous: no credentials will be provided to the SMTP server
   - Basic: enter a username and password the service will provide to the SMTP server

     Note: New username and password you put in will override existing credentials saved in the PI Notifications Service. For basic authentication in a failover cluster setting, see PI Server installation topic Install PI Notifications Service in a failover cluster.
7. Click **Save and Close** or **Cancel**.
8. Optional. Check **Use TLS** box to enable encryption with basic authentication. Otherwise, the username and password will be sent in plain text.

Configure authentication for a web service connection

Best practice is to use a secure connection when using basic authentication. To use a secure connection for a web service, use the *https://* URI scheme. The server must present a valid certificate for a secure connection to be established. To configure authentication for a web service connection, follow this procedure.

1. Go to **Contacts** plug-in.
2. Under **Delivery Endpoints**, select the web service delivery endpoint you want to authenticate.

   Note: Both SOAP and REST delivery methods are supported. Be sure to check in the delivery endpoint if it is newly configured.
3. Click a hyperlink next to **Authentication Option** at the bottom of the window.

   Select Authentication Option window opens.
4. Choose among the 3 authentication options.
   - Windows: PI Notifications Service will pass the network credentials of its service account to the web service
   - Anonymous: no credentials will be provided to the web service
   - Basic: enter a username and password the service will provide to the web service

Note: New username and password you put in will override existing credentials saved in PI Notifications Service. For basic authentication in a failover cluster setting, see Install PI Notifications Service in a failover cluster.

5. Select **Basic** radio button.
6. Enter the user name and password and click **Save and Close**.

Create a notification rule

Configuring a notification rule includes specifying trigger criteria, adding subscribers to the notification rule, and formatting the message to suit the needs of your organization.

For a detailed sample, see Configuration of notification rules for analyses or event frames.

1. In PI System Explorer, select the element on which you want to create notification rules.
2. From either the **Notifications Rules** tab, or from an existing event frame analysis, select **Create a new notification rule**.
3. Enter a name for the new notification rule and (optionally) select a category.
4. In the Trigger Criteria pane, specify the set of conditions that causes a notification to be sent.

   For more information, see Trigger criteria in notification rules.
5. In the Subscriptions pane, select **Manage Formats** and specify the format for notifications.

   For more information, see Delivery formats in notification rules.
6. In the Subscriptions pane, select **View/edit subscriptions** and specify the contacts to which notifications will be sent.

   For more information, see Subscriptions in notification rules.
7. Test that the notification is triggered when an event occurs that satisfies all of the trigger criteria specified.

Create a notification rule template

You can create a new notification rule template for an existing element template in PI System Explorer. You can also convert an existing notification rule into a notification rule template by right-clicking on the notification rule and selecting **Convert to Template**.

For a detailed sample, see Configuration of notification rules for analyses or event frames.

1. In the navigator pane of PI System Explorer, click **Library**.
2. Select **Templates** > **Element Templates**.
3. Select the specific element template where you want to create the notification rule template.
4. Click on the **Notification Rule Templates** tab.

   The tab lists the notification rule templates already defined for the selected element template.
5. Create a new notification rule template for the selected element template.

   If no notification templates exist, click **Create a new notification rule template** to create the first one. Otherwise, click the New Notification Rule Template icon to create a new notification rule template.
6. Enter information to identify the notification rule template.

- **Name**

  The name that uniquely identifies this notification rule template.
- **Description**

  Optional text that describes the notification rule template.
- **Categories**

  Optional category that you can assign to the notification rule template. Click the list and select the check box next to the category you want to assign, or click **New** to create a new category.

7. Configure trigger criteria for the notification rule template.

   For specific instructions, see Trigger criteria in notification rules.
8. Click on **Manage Formats** to customize the message that is sent with the notification.

   For specific instructions, see Delivery formats in notification rules.
9. Click on **View/Edit Subscriptions** to configure subscriptions.

   For specific instructions, see Subscriptions in notification rules.
10. Click **Check In** Check In to save the changes.

Trigger criteria in notification rules

You can configure trigger criteria for an event to determine when a notification will be sent to the appropriate delivery channel, for example, an email address or web service. The trigger conditions are configurable in the **Trigger Criteria** pane, and further options can be configured in the Options pane.

To see the Trigger Criteria and Options panes on PI System Explorer, navigate to the **Notification Rules** tab for a selected element, and then click **View/Edit Trigger** in the Trigger pane.

Trigger Criteria pane

The Trigger Criteria pane allows you to configure either of these trigger criteria modes: **Analysis** or **Event Frame Search**.

Use analysis mode to trigger a notification rule on event frames generated by a particular analysis. In the **Analysis** mode, you can select an existing analysis or configure a new analysis of type **Event Frame Generation** or **SQC**, and optionally configure additional trigger conditions based on event frame attribute values. If you choose to create a new analysis with either the default or a custom name, you see the same analysis creation window as from the **Analyses** tab. For information on configuring an SQC analysis, see Create an SQC analysis. For information on creating an Event Frame Generation analysis, see Create an event frame generation analysis. For information on expression functions to be used as a trigger condition, see Expression functions reference.

Note: For notifications to take effect, you must check in both the analysis and the notification rule. When creating an SQC analysis from the **Trigger Criteria** pane, you can only choose **Event Frame** as the analysis output.

Use event frame search mode to trigger a notification rule based on event frame name, template and category. In the **Event Frame Search** mode, you can select a configured event frame template, from the drop-down list, and then configure the name and category for the event frames that will trigger your notifications. The name may contain wildcard characters that are supported by event frame search. You can optionally configure additional trigger conditions based on event frame attribute values.

For both **Analysis** and **Event Frame Search** modes, you can add additional trigger criteria using event frame attribute values. Criteria can be specified for any event frame template attribute that is specified in the notification rule trigger criteria. The attribute values that you select in the conditions must be specifically those for which you want to be notified of event frames (that match the analysis event frame template). For example, if your event frame template defines an event like "downtime" but you only want an email about "unplanned" downtime, you can configure an attribute value condition where a "reason code" attribute on the "downtime" event frame template has a value indicating "unplanned" downtime.

Note: Multiple conditions are grouped using the AND operator.

Options pane

The Options pane allows you to configure these trigger options:

- **Resend Interval**

  The time interval after which PI Notifications Service will send additional alerts until the event frame matching the notification rule is acknowledged or closed.
- **Non-repetition Interval**

  The time interval during which PI Notifications Service will not send similar alerts associated with the same notification rule.
- **Event Frame Can Be Acknowledged**

  Option to enable event frame to be acknowledged; the event frame template is also modified accordingly. This option is automatically selected if the event frame template has been configured for acknowledgment.
- **Severity option**

  This option applies only to event frame generation analyses. If you have configured multiple start triggers for your analysis, you may choose to be notified in these ways:
  - When the current trigger severity is higher than any trigger severity encountered so far.
  - When the current trigger severity is higher than the previous trigger severity.
  - Every time a trigger condition is true, regardless of its relative severity to other previous triggers.

Non-Repetition Interval in notification rules

You can configure the **Non-Repetition Interval** setting to prevent notifications from sending similar alerts, which are associated with the same condition, within a specified amount of time. The non-repetition interval is not applied for close sends.

Alerts will be sent for triggering events with a higher severity even if a prior alert has been sent for a triggering event of lower severity, and within the time-frame of the non-repetition interval.

The system determines when to resend the notification based on the time the last notification was sent, and not based on the start time of the triggering event. For example, if the event frame started at time t, and the service, after processing, sends the notification at time t1, the system uses t1 as the basis to calculate the time between alerts.

Note: If you specify both a resend interval and a non-repetition interval, the resend interval must be longer.

Resend Interval in notification rules

You can configure the **Resend Interval** setting so that additional alerts are sent when the event frame has not been acknowledged or closed; it is a reminder that the event of interest leading to the notification is yet to be resolved. The resend interval specifies the time between sending successive alerts.

The system determines when to resend the notification based on the time the last notification was sent, and not based on the start time of the triggering event. For example, if the event frame started at time t, and the service, after processing, sends the notification at time t1, the system uses t1 as the basis to calculate the time between alerts.

Note: If you specify both a resend interval and a non-repetition interval, the resend interval must be longer.

Delivery formats in notification rules

You can configure the format of the message that is delivered when a notification rule is triggered in the Message and Content panes. To access the Message and Content panes, navigate to the **Notification Rules** tab for a selected element, and then click **Manage Formats** in the **Subscriptions** pane.

Global default formats

The system provides one global format that is used as the default format when a subscriber is added to a notification rule. You can edit or rename the global format, but you cannot delete it.

You can also create additional global formats. These global formats can be viewed in notification rules or notification rule templates. To create, edit or delete global formats, navigate to **Tools** > **Global Formats** in PI System Explorer. You can edit the fields in the Manage Global Formats window. To create and edit global formats, follow the procedures in the "Custom delivery formats" heading below. For more information on additional global configuration parameters for notifications, see the knowledge base article Manually Managing Global Configuration Parameters in Notifications 2.x.

Custom delivery formats

To create a custom delivery email format, copy the default **Global Default Email** format by clicking on the copy icon in the Message pane. Alternatively, you can create a new delivery format by clicking the new delivery format icon . The default format includes basic notification information such as the notification rule name, trigger time and event frame start time, as well as system information such as system and database names.

To customize your delivery format, you can drag and drop the following types of information from the Content pane to the Message pane:

- PI AF Server properties
- Database properties
- Notification Rule properties
- Event frame properties
- Event frame attributes. If an event frame template is selected, you may add the event frame template attributes to the content of your email notification.
- Referenced element properties

- Referenced element attributes

These properties and attributes can provide a lot of input to analyzing the event of interest; for example, including the event frame start time property can tell you when the event of interest started. Including specific event frame and referenced element attributes can give you supporting information about key parameters.

Starting from PI AF 2018, any numerical attributes in email notifications will display number of digits according to how **Display Digit** is configured on the attribute or attribute template. For more information, see Control the display of attribute and attribute template values.

For AVEVA PI Vision users, you can construct a link that specifies a full path to a display. You can select **Full Path** from **Referenced element properties** or **Referenced element attributes** under the Content pane.

- PI Notifications Service and PI System Explorer 2017 R2 or later are required to use the **Full Path** option.
- **Full Path** can also be configured for global delivery formats.
- The existing **Path** option is relative to the reference element or attributes.

When sending a notification email, you can include a web, AVEVA PI Vision URI, or a file link. That way, you can easily navigate to specific locations for relevant information about attributes, event frames, and so on. To add a link, click in the Message pane, and then click the link 🔗 icon. You will be prompted to enter your base address. When you click the **Continue** button, **Create a link** window will open. There, you can configure your link.

The link feature allows you to include a **Screenshot** (or a static image) of the data that triggered a notification. You can also specify display options if you choose to display the link as a screenshot. See Add a screenshot in notification email delivery format.

You can edit the message if you choose to display your link as a text. **Display link as** option can be found in 2 other places: in **Event Details Hyperlink** (if AVEVA PI Vision is configured) and URI hyperlink (under **Referenced element attributes**; see Create URI Builder data references for more information).

If you are configuring a Web URI, you can specify a **fragment**, which is a string that is appended at the end of the URI. For more information on query string parameters, see the *PI Vision* user guide topic URL parameters reference.

You can add a file attachment to your delivery format by clicking on the **+** symbol in the **Attachments** field. The selected attachment will be uploaded as an attribute on the referenced element. For acceptable types of attachment files, see **/FileExtensions** in AFDiag utility parameters.

Tip: If the Content pane has a message indicating that your selected format is "read-only", you must first add a new format before you can customize it.

Training video

For more information on message formats, watch the AVEVA PI System Learning channel video:

Add a domain to the list of trusted sites

Protect your system from unwanted security incidents by allowing screenshots only from a trusted source. If you want to receive screenshots from a claims-enabled site that uses OpenID Connect (OIDC), such as PI Vision, you will need to enter the domain to the list of trusted sites.

To add a website to the trusted-sites list, follow this procedure.

1. Click **Database** on the PI System Explorer toolbar.
2. Select the **Configuration** database.
3. In the Elements pane, click **OSIsoft** > **PIANO**.
4. Select the **Attributes** tab.
5. Select **PageRender** > **TrustedSites**.
6. Add the sites in the **Value** field of the attribute delimited by a space.

Add a claims-enabled site

1. Optional. To add a claims-enabled site, select **PageRender** > **OpenIDConnectSites**.
2. Optional: Add the sites in the **Value** field of the attribute delimited by a space.

The wildcard (*) character is allowed. Here are some examples:

Note: Whitelisting applies to the website and all of its embedded elements (images, scripts).

If you entered ...	The system allows ...	The system blocks ...
*	All websites	No website
https://*	All websites using the **https://URI** scheme	All other websites
https://*.AVEVA.com	**https://www.aveva.com** **https://www.aveva.com/index.html** **https://www.aveva.com:80/index.html**	**http://www.aveva.com** **www.aveva.com** **https://aveva.com**
www.aveva.com/index.html	**www.aveva.com/index.html** **www.aveva.com/index.html?abc=123**	**www.aveva.com/index** **www.aveva.com/index.htmlabc**
www.aveva.com/abc	**www.aveva.com/abc** **www.aveva.com/abc/def** **www.aveva.com/abc.html** **www.aveva.com/abc?def=123**	**http://www.aveva.com/abcdef** **www.aveva.com/abc.def**
***.aveva.com**	**www.aveva.com** **www.aveva.com:80** **www.aveva.com/index.html** **https://www.aveva.com** **http://www.aveva.com/index.html**	**aveva.com** **notaveva.com** **www.aveva.com.baddomain**
aveva.com	**www.aveva.com** **www.aveva.com:80** **www.aveva.com/index.html** **https://www.aveva.com** **http://www.aveva.com/index.html**	**notaveva.com** **aveva.com.baddomain**
https://localserver/PIVision	**https://localserver/PIVision** **https://localserver/PIVision/#/Displays**	**https://localserver/** **https://localserver/PI** **http://localserver/PIVision** **https://localserver.int/PIVision**
***://*.int**	**http://internal.website.int** **http://internalwebsite.int** **http://internal.website.int:80**	**https://www.aveva.com** **http://www.aveva.com**

If you entered ...	The system allows ...	The system blocks ...
	http://internal.website.int/index.html	
192.168.1.*	**192.168.1.123** **192.168.1.123/abc.html** **http://192.168.1.123:8080/abc.html**	**192.168.22.123**

Add a screenshot in notification email delivery format

You can include a screenshot (or a static image) of the data that triggered the notification.

To include a screenshot in an email from a site that uses OIDC for claims-based authentication, you will need to add the site to the list of trusted sites. See Add a domain to the list of trusted sites for instructions on adding both Windows and claims-enabled sites to this list. Also, see the installation topic Trusted sites for screenshots.

To add a screenshot in notification email delivery format, follow this procedure.

1. Navigate to the element that you want to receive the notification.
2. Click on the **Notification Rules** tab.
3. Select the notification rule that you want to add a screenshot.

   Click **Create a new notification rule** if you are creating a new notification rule.
4. Click **Manage Formats** in the Subscriptions pane.
5. Copy the default **Global Default Email** format by clicking on the copy icon in the Message pane.

   Alternatively, you can create a new delivery format by clicking the new delivery format icon.
6. Click in the Message pane, and then click the link icon.
7. Enter your base address in Paste a Link window and click **Continue**.
8. In the **Display link as** row, click the **Screenshot** radio button.
9. Optional. In the **Image options** row, choose **Format** and **Size** from each drop down list.
10. Click **OK**.

Subscriptions in notification rules

Contacts and Subscriptions panes in PI System Explorer

From the Subscriptions pane, you can configure the subscribers to be alerted when a notification rule is triggered. A subscriber is an individual entity or group of entities that can receive notification messages by subscribing to notification rules.

To see the Contacts and Subscriptions panes on PI System Explorer, navigate to the **Notification Rules** tab for a selected element, and then click **View/Edit Subscriptions** in the Subscriptions pane.

To add a subscriber from the **Notifications Rules** tab of PI System Explorer, simply drag and drop a contact from the Contacts pane to the Subscriptions pane.

Note: To unsubscribe a contact, right-click on the contact and select **Unsubscribe**.

When you install PI AF, information from Active Directory (AD) is imported and individual contacts are automatically created in Contacts. For more information, see Configure Active Directory access for contacts. If a user does not have an AD account, you can create a custom contact.

Subscriptions pane

When a contact is added to the Subscriptions pane, the following details are included for the contact:

- Name

  The name of the contact.
- Configuration

  The default, inherited, or custom delivery format for this subscriber.
- Notify Option

  The notify option may be one of:
  - Event start and end. You can receive a notification when an event frame matching the notification rule is created or closed.
    - Event start.
    - Event end. You can receive a notification only when an event frame is closed.

Contacts pane

Subscribers may be configured from these types of contacts on the Contacts pane:

- Individual contact
- Escalation teams
- Groups
- Delivery endpoints

Use **Contacts Search** to find individual contacts to include in a group or escalation team, or subscribe to a notification rule. You may use wildcards to expand your name searches; this will search both first and last names. For example, searching for "A*" will list both "Adam Smith" and "John Adams".

Note: The search results show the display name as configured in Active Directory.

For more information on configuring contacts using the Contacts plug-in, see Contacts plug-in.

Training video

For information on how to configure subscribers, watch this video:

Setup Contacts, Delivery Endpoints, Groups, & Escalation Teams for Notifications[2016 R2]

Customization of subscription content and delivery

The default delivery format specified on the **Message** tab is automatically applied to all subscriptions in the notification. To send a different message, you can customize subscription formats by applying a different delivery format.

Note: If you customize subscriptions and later want to change delivery formats, you might need to manually change the delivery formats of subscriptions that you previously customized. For example, if you change the default delivery format, or specify a different delivery format as the default, only subscriptions that inherit that format will be changed; customized subscriptions are not affected.

For more information on delivery formats, see Delivery formats in notification rules.

Configuration of notifications delivery endpoints

A delivery endpoint is a single entity to which a notification is delivered, such as an individual email address or web service call. Each delivery endpoint is associated with a delivery channel, the conduit through which notification messages are sent. Notifications provides delivery channels for email and for both SOAP and REST web services.

To configure email delivery endpoints, see Configure an email delivery endpoint. To configure dynamic email delivery endpoints, new in PI AF 2017 R2, see Configure a dynamic email delivery endpoint.

To configure web service delivery endpoints, see Configure a REST web service delivery endpoint.

To configure notifications to be sent as text messages, see the knowledge base article: How do I set up PI Notifications to send text messages to my cellular phone?.

Configure an email delivery endpoint

If you have not already configured the SMTP server settings for the email delivery channel, follow the instructions in: Update SMTP settings for the email delivery channel.

To configure an email delivery endpoint and add an email subscriber to your notification rule, follow this procedure.

1. Add an email delivery endpoint using the PI System Explorer Contacts plug-in.

   For specific information, see Manage contacts and Options for the notifications email delivery channel. See Configure authentication for SMTP server connection for setting up authentication.
2. Navigate to **Elements** > **your_element** > **Notification Rules** and add the configured email delivery endpoint to your notification rule as a subscriber.
   a. Click **View/Edit subscriptions** in the Subscriptions pane of the notification rule
   b. Drag and drop the configured contact from the Contacts to the Subscriptions pane.

Update SMTP settings for the email delivery channel

You need to configure the email delivery channel to be able to receive email notifications. This must be done once for each PI AF server.

1. To start PI System Explorer, click **Start** > **Programs** > **PI System** > **PI System Explorer**.

2. Click on the PI System Explorer toolbar, or click **File** > **Server Properties**.
3. Click the **Plug-Ins** tab.
4. Scroll down to the **Delivery Channel Plug-Ins** section, then right-click **Email** and select **Settings**.
5. In the Email Delivery Channel Configuration window, change the following parameters as needed:
    - **SMTP Server**: The fully-qualified domain name of the machine running the SMTP server.
    - **Port**: The listener port of the SMTP Server.
    - **Authentication options**: Options for a primary SMTP server authentication. See Configure authentication for SMTP server connection.
    - **Use TLS**: Option to enable Transport Layer Security (TLS) encryption for the primary server
    - **Backup SMTP Server**: The fully-qualified domain name of the machine running the backup SMTP Server.
    - **Port**: The listener port of the backup SMTP Server.
    - **Authentication options**: Options for a secondary SMTP server authentication.
    - **Use TLS**: Option to enable Transport Layer Security (TLS) encryption for the secondary server
    - **Sender Email**: The email address from which notification messages will be sent.
    - **Allow contacts to set sender email**: Specifies whether individual contacts can change the email address from which their notification emails are sent.
    - **Send Timeout**: Time allowed for sent emails to be received by the primary SMTP server before failover to the backup SMTP server occurs.
    - **Backup Fail Back Time**: During failover, specifies how long the backup SMTP server sends email before attempting to fail back to the primary server.

Configure a dynamic email delivery endpoint

In PI AF 2017 R2, you can configure an email delivery endpoint as a value of an attribute. This provides an additional flexibility in situations where notification emails can be sent to different recipients without making changes to the notification rule template. To configure a dynamic email delivery endpoint and add an email subscriber to your notification rule, follow this procedure. For a conventional email delivery endpoint configuration, see Configure an email delivery endpoint.

1. Navigate to the element that you want to receive the notifications from and add an attribute with your email address as a value.

   The value of this attribute must be a string. You can enter multiple email addresses delimited by comma.
2. Navigate to **Notification Rules** tab to configure and add an email delivery endpoint as a subscriber to your notification rule.
    a. Click **View/Edit subscriptions** in the Subscriptions pane of the notification rule.
    b. Click **Create a new dynamic endpoint** under **Dynamic Endpoints**.
    c. Select an email attribute configured in step 1.
    d. Click **Create**.
    e. Drag and drop the configured contact from the Contacts to the Subscriptions pane.
    f. Click **OK**.
3. Go to **Contacts** plug-in to edit or delete a configured delivery endpoint.

It will be found in **Delivery Endpoints** folder.

- The email attribute endpoint can be edited from the Subscriptions pane as well.
- **Test** button in the **Contacts** plug-in is not supported for an attribute-based endpoint at this time.

Configure a REST web service delivery endpoint

To configure a web service delivery endpoint and add a REST web service subscriber to your notification rule, follow this procedure. For SOAP web service, see Configure a SOAP web service delivery endpoint.

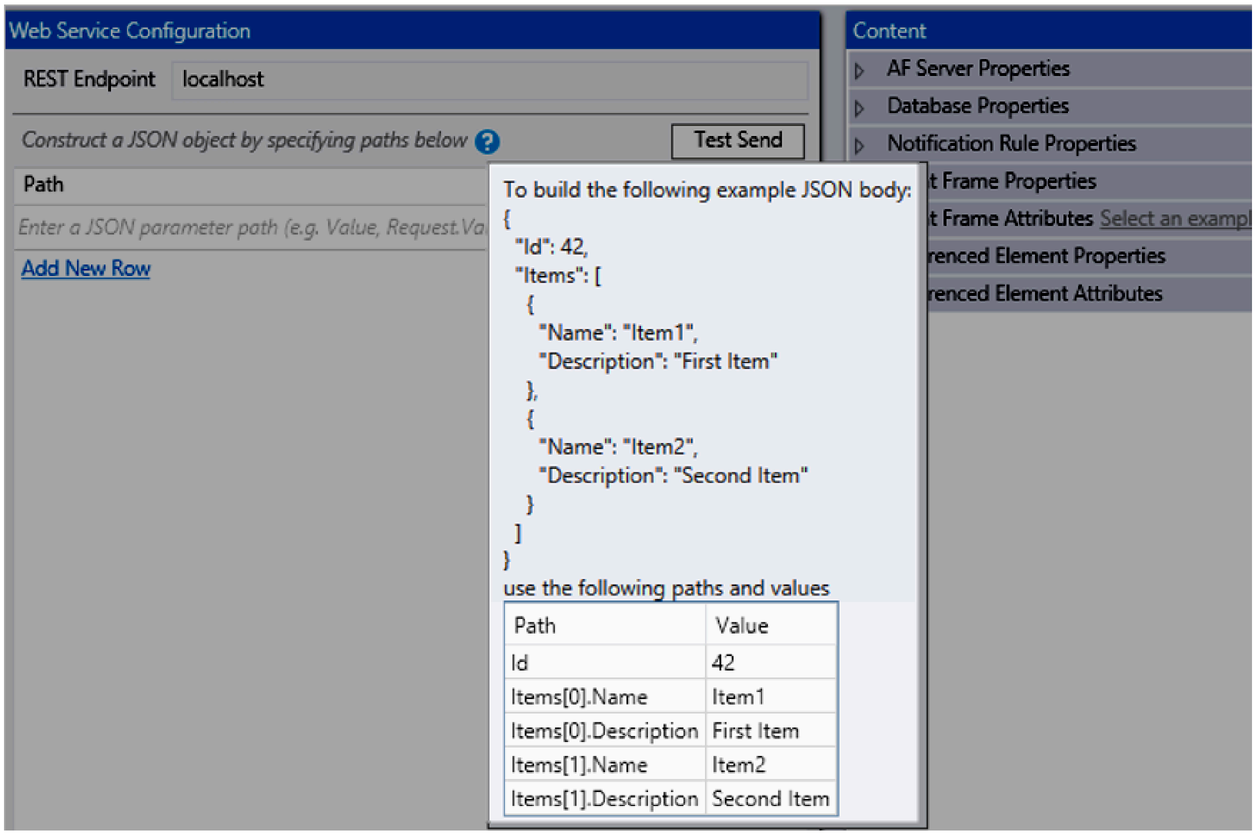
1. Add a web service delivery endpoint using the PI System Explorer Contacts plug-in.
   a. Go to the **Contacts** plug-in of the PI System Explorer.
   b. Right-click the **Delivery Endpoints** folder.
   c. Select **New Delivery Endpoint**.
   d. Optional. **Retry interval**. Select the time interval at which PI Notifications Service will contact the web service. The web service will be contacted as frequently as the system allows if zero is entered.
   e. Optional. **Maximum Retries**. Select the number of times PI Notifications Service will try to connect to the web service. Entering zero means there will be no retries.
   f. Select **WebService** from the drop-down list next to **Delivery channel**.
   g. Select **REST**.
   h. Enter the URL of your web service.
   i. Select **Http Method**, a default web method to be used for the notification. Supported methods are POST, PUT and PATCH.
   j. Select **Authentication Option**.
      a. Windows: the default option. PI Notifications Service will pass the network credentials of its service account to the web service
      b. Anonymous: no credentials will be provided to the web service
      c. Basic: enter a username and password the service will provide to the web service
   k. Check in the changes.

   See **Create a delivery endpoint such as a stand-alone email or a web service** row in Manage contacts and Options for the notifications web service delivery channel for more information. See also Configure authentication for a web service connection for authentication of web service.
2. Add the configured web service delivery endpoint to your notification rule as a subscriber.
   a. Go to the **Notification Rules** tab of the element you want to add the web service as the delivery endpoint.
   b. Click **View/Edit subscriptions** in the Subscriptions pane of the notification rule.
   c. From the Contacts pane on the right side, expand **Delivery Endpoints** and drag the configured REST web service to the Subscriptions pane on the left.
   d. Click the wrench 🔧 icon to open the Web Service Configuration window to configure the JSON object for the REST API method.

      REST can send a JSON object to the REST server in HTTP methods PUT, POST or PATCH.
   e. Click the **Body** tab to enter the path and value on each row.

Currently, only JSON payloads are supported; query string parameters are not supported.

If you hover over the help icon, you see sample JSON parameters (path and value) that are necessary for configuring this method:


Web Service Configuration
REST Endpoint
localhost
Construct a JSON object by specifying paths below
Test Send
Path
Enter a JSON parameter path (e.g. Value, Request.Va
Add New Row
Content
AF Server Properties
Database Properties
Notification Rule Properties
t Frame Properties
t Frame Attributes Select an exampl
renced Element Properties
renced Element Attributes

To build the following example JSON body:

```
{
  "Id": 42,
  "Items": [
    {
      "Name": "Item1",
      "Description": "First Item"
    },
    {
      "Name": "Item2",
      "Description": "Second Item"
    }
  ]
}
```

use the following paths and values

Path	Value
Id	42
Items[0].Name	Item1
Items[0].Description	First Item
Items[1].Name	Item2
Items[1].Description	Second Item

You can validate the method using the **Test Send** button.

f. Click the **Headers** tab to enter HTTP headers for your request.

   You use headers to pass additional information along with your request. You can enter the key and value pair on each row. Common HTTP headers will be available in a drop-down list to enter as a key (for example, Content-Type).

   Note: The default value of the HTTP header Content-Type is application/json; charset=utf-8.

   You can also enter a custom header if your web service requires a specific type of header (for example, a custom API key header.)

   You can validate the request using the **Test Send** button.

Configure a SOAP web service delivery endpoint

To configure a web service delivery endpoint and add a SOAP web service subscriber to your notification rule, follow this procedure.

1. Add a web service delivery endpoint using the PI System Explorer Contacts plug-in.
   a. Go to the **Contacts** plug-in of the PI System Explorer.
   b. Right-click the **Delivery Endpoints** folder.

c. Select **New Delivery Endpoint**.
d. Optional. **Retry interval**. Select the time interval at which PI Notifications Service will contact the web service. The web service will be contacted as frequently as the system allows if zero is entered.
e. Optional. **Maximum Retries**. Select the number of times PI Notifications Service will try to connect to the web service. Entering zero means there will be no retries.
f. Select **WebService** as the **Delivery channel**.
g. Select **SOAP**.
h. Enter the web service address, the URL of your web service.

You can validate the connection using the **Get Web Services** button.

a. Enter the name of the web service to be used for notification in the **Web Service** field.
b. **Default Web Method**: select the default web method to be used for the notification.

This menu displays all of the parameters defined in the web service.

a. Select **Authentication Option**.
    a. Windows: the default option. PI Notifications Service will pass the network credentials of its service account to the web service
    b. Anonymous: no credentials will be provided to the web service
    c. Basic: enter a username and password the service will provide to the web service
b. Check in the changes.

See **Create a delivery endpoint such as a stand-alone email or a web service** row in Manage contacts and Options for the notifications web service delivery channel for more information. See also Configure authentication for a web service connection for authentication of web service.

2. Add the configured web service delivery endpoint to your notification rule as a subscriber.
    a. Go to the **Notification Rules** tab of the element you want to add the web service as the delivery endpoint.
    b. Click **View/Edit subscriptions** in the Subscriptions pane of the notification rule.
    c. From the Contacts pane on the right, expand **Delivery Endpoints** and drag and drop the configured web service to the Subscriptions pane on the left.
    d. Configure the SOAP web service.

    SOAP can call the web service methods. Click the "wrench" 🔧 icon to configure the SOAP API method.

    The "Information" ℹ icon gives you information about the parameters (path and value) that must be configured for this method, and their complexity. Enter the path and value on each row (the path depends on how the web service server is configured); clicking on a row brings up a drop-down list of expected values for that path. The interface also automatically validates the parameters that you enter on each row. You can validate the method using the **Test Send** button.

    You have the option to cancel or retry a connection that is taking a long time to complete; this may happen if you have a high latency (commonly called "laggy") connection where the WSDL information is not quickly retrieved.

    Note: Complex data may be required by the method. To enter the correct information, refer to the Web Service Definition Language (WSDL) documentation for the API method.

Contacts plug-in

In PI System Explorer, the Contacts plug-in provides you with contact options for individual users and groups.

Contacts plug-in versions

PI System Explorer works with different versions of the Contacts plug-in.

- To work with the version that is compatible with the new version of notifications that was released in 2016 R2, execute the 64-bit version of PI System Explorer.
- To work with the legacy version that is compatible with legacy notifications, execute the 32-bit version of PI System Explorer. Note that if legacy notifications is not installed, the new version of the Contacts plug-in will load. For more information on PI System Explorer versions, see PI System Explorer.

Subscriber configuration

You can configure subscribers from these types of contacts:

- **Delivery endpoints**

  A delivery endpoint is an entity to which notifications can be sent, and is typically an individual user or an application.
- **Groups**

  A group contact is an unordered collection of delivery endpoints, other groups or escalation teams. Notification messages are sent to all members of the group simultaneously.

  Note: Group contacts that you create in PI System Explorer are not the same as Active Directory (AD) groups.
- **Escalation teams**

  An escalation team is an ordered collection of delivery endpoints or groups.

Notification messages are sent immediately to the first contact on the list. If the event frame matching the notification rule is not acknowledged or closed within a specified time, notification messages are sent sequentially to the remaining members of the escalation team until the event frame is acknowledged or closed.

The escalation period defines the amount of time to elapse before a notification is sent to the next contact on the list.

Search for contacts

Click the 🔍 icon to open the Search Contacts window in the Contacts pane on the top left of the screen. Currently available search criteria are: Name, Description, Department and Email.

Note: You must be running the 2018 or later version of PI AF server to use email as search criterion. Note that search by email address only works for contacts in Active Directory but does not find contacts created manually.

Configure Active Directory access for contacts

When you use notifications with PI AF server, you may need to specify how to access Microsoft's Active Directory to retrieve contact names for the PI Notifications Service Contacts lists.

Each PI AF server provides the option to specify the domain and contact sub-folder, as well as the account needed to access Active Directory and retrieve contact names. By default, the account under which the PI AF server application service is running is used for Active Directory access. To use a different account or to access an Active Directory in a different domain, configure access from the Configure Active Directory Access for Contacts window.

Note: Beginning with PI AF 2017 R2, Active Directory group is shown by default under **Contacts** in PI System Explorer once it is configured.

Notifications 2016 R2 or later will automatically handle a change in email address of the user or a group in Active Directory by contacting the domain controller before sending an email each time.

1. Open PI System Explorer and connect to a database on the PI AF server for which you want to configure Active Directory access.
2. Click **File** > **Server Properties**.
3. In the PI AF Server Properties window, click the **Configure Active Directory Access for Contacts** link.
4. In the **Active Directory Domain Name** text box, enter the full DNS name of the Active Directory domain from which the contact names will be retrieved for the PI Notifications Service Contacts (for example, `contoso.com`).

   If this field is left blank, the domain in which the PI AF application service resides will be used.
5. In the **Active Directory Contact Sub-Folder** text box, enter the path to the folder containing the list of contacts for this domain.

   In larger Active Directory domains, contacts may be organized within sub-folders. The use of sub-folders can allow for faster retrieval of a list of Active Directory contacts.

   Use the following structure for the sub-folder:

   ```
   DomainUserFolder/SubDomainUserFolder/Sub-SubDomainUserFolder
   ```
6. Choose an option for **Active Directory Access Account**:
   - **Use the account the PI AF Server runs as**

     This is the default option. Select it to access Active Directory using the account under which the PI AF application service runs. By default, the PI AF server is installed using a virtual account, `NT SERVICE\AFService`. However, the PI AF server service account can be changed. If the PI AF server service account does not have the necessary permission to read the Active Directory, no contact names will be retrieved in the Contacts list. If your Active Directory security is configured to allow the PI AF server service account to read the Active Directory, this is the simplest option.
   - **Use the account the AF Client is running as**

     Select this option to use the credentials of the user account under which the connecting client application is running. If the PI AF server service is running under an account (a virtual account, `NT SERVICE\AFService` is the default account) that does not have permission to read the Active Directory, this option can be used. As long as the user account under which the connecting client application is running has permission to read Active Directory, a list of contact names is returned to the Contacts list. The contents of the Contacts list may vary, depending upon the access account used, since the security to read the contact list is determined by Active Directory.

     Note: Specifying this option may require Kerberos configuration if a PI AF SDK application will be using impersonation in a middle tier, such as a Web Service.
   - **Use the specified account**

This option allows you to specify an account to use to read the Active Directory. This can be useful when the Active Directory and PI AF server are in different domains or when the accounts in the first two options have no permission to read the Active Directory. For **Account Name**, use the format `Domain\User`. Make sure the specified account has the appropriate permission to read the target Active Directory.

7. Check **Use Active Directory's locally cached Global Catalog** to use the global catalog for Active Directory domain controller searches. Otherwise searches must go to the owning domain controller.

   Active Directory holds information in a distributed data repository called a global catalog. For installations where there are multiple, distributed domain controllers, each domain controller has a cache of the portions of the global catalog for which it is not responsible, so that Active Directory searches do not have to be referred to the owning domain controller. This improves performance for queries that must otherwise have to access a remote domain controller.

8. Choose a setting for **Return All Persons**.

   Active Directory objects are derived from one another as follows:

   `Top>Persons>OrganizationalPerson>Contact`

   and

   `Top>Persons>OrganizationalPerson>User`

   - Select this check box to return Persons, Organizational Persons, Contacts and Users from the target Active Directory.
   - Clear the check box to return only Users.

Manage contacts

Use the Contacts plug-in to manage individual contact information, as well as manage groups, escalation teams, and delivery endpoints for use with notifications.

Note: You cannot nest escalation teams.

1. In the navigator, click **Contacts**.
2. In the Contacts browser, choose from the following actions.

To ...	Do this ...
Edit an existing contact	a. Choose one of the following actions: a. Enter search criteria in **Search for contacts** and press Enter. Note that * returns all contacts. b. Click 🔍 next to **Search for contacts**, enter search criteria in the Search Contacts window, and click **OK**. c. Expand the **Contacts** folder, click **New search...**, enter search criteria in the Search Contacts window, and click **OK**. Search results are displayed in the Contacts browser.

To ...	Do this ...
	b. Click ⊞ beside the contact you want to update and click the email delivery endpoint. c. In the viewer, you can make the following changes: a. In the **Description** field, add additional contact information. b. Enter notification contact options in the **Retry interval** and **Maximum Retries** fields.
Create a custom contact that is not already in Active Directory	a. Right-click the **Contacts** folder and click **New Contact**. b. In the viewer, enter a name and an email address. All other information is optional. The contact is displayed in the Contacts browser. c. Click ⊞ beside the newly created contact. **Note:** The icon indicates non-AD contacts (as opposed to). d. Click the email delivery endpoint and, in the viewer, enter notification contact options in the **Retry interval** and **Maximum Retries** fields.
Create and edit a group	a. Right-click the **Groups** folder and click **New Group**. b. In the **Name** field, enter a unique name for the group. c. Choose from the following actions: a. Click and, in the Select a Contact window, locate a contact with a new search, or select another group, an escalation team, or a delivery endpoint, and click **OK**. Repeat as needed. b. In the Contacts palette, locate a contact with a new search, or select another group, an escalation team, or a delivery endpoint, and drag it onto the viewer. Repeat as needed. d. Set notification contact options for the group. a. Right-click each subscriber delivery endpoint, click **Options**, and enter notification contact options in the **Retry interval** and **Maximum Retries** fields. b. If escalation teams are included in a group,

To ...	Do this ...
	right-click each escalation team, click **Options**, and enter notification contact options in the **Escalation period** and select an **If not acknowledged** option. e. To remove a subscriber from a group: a. Click the subscriber, then click ✕ or right-click the subscriber and click **Remove**.
Create and edit an escalation team	a. Right-click the **Escalation Teams** folder and click **New Escalation Team**. b. In the **Name** field, enter a unique name for the escalation team. c. In **Escalation period**, specify how long you want an escalation to last. d. In **If not acknowledged**, specify the action to be taken if the notification is not acknowledged after being sent to all contacts in a team: a. To stop the escalation process, select **End escalation**. b. To repeat the escalation process for a specified number of times until the notification closes or is acknowledged, select **Repeat N times**. c. To repeat the escalation process indefinitely until the notification closes or is acknowledged, select **Repeat while active**. e. Choose from the following actions: a. Click and, in the Select a Contact window, locate a contact with a new search, or select another group, or a delivery endpoint, and click **OK**. b. In the Contacts palette, locate a contact with a new search, or select another group, or a delivery endpoint, and drag it onto the viewer. f. Configure the sequence for the escalation chain. a. Click a subscriber and click ⬆ and ⬇ to position as needed. b. To remove a subscriber, click the subscriber to be removed and click ✕.
Create a delivery endpoint such as a stand-alone	a. Right-click the **Delivery Endpoints** folder and

To ...	Do this ...
email or a web service	click **New Delivery Endpoint**. b. In the **Name** field, enter a unique name for the delivery endpoint. c. Enter notification contact options for the delivery endpoint in the **Retry interval** and **Maximum Retries** fields. d. From **Delivery channel**, select a deliver option. a. For **Email**, enter address configuration information, as described in Options for the notifications email delivery channel. b. For **WebService**, enter web service address configuration information, as described in Options for the notifications web service delivery channel.

3. To save changes you have made to contacts, press Ctrl+S or click **Check In**.

Options for the notifications email delivery channel

An email delivery endpoint can be configured in ways specific to the email delivery channel, such as the address to which notification is sent. The following table shows options that are available for all email delivery endpoints:

Option	Description
To Email	Email address to which the notification is sent. This option cannot be changed in the default email delivery endpoint that was derived from the Active Directory (AD) contact. Multiple email addresses can be entered delimited by comma.
From Email	Email address from which the notification is sent. This option is available solely if your PI System manager has configured the email delivery channel with the setting `Allow contacts to set sender` email.
Use HTML formatting	Specifies if the delivery endpoint receives messages in HTML format. HTML formatting makes messages easier to read and allows messages to contain hyperlinks.

Options for the notifications web service delivery channel

Creating a web service delivery endpoint

Using the Contacts plug-in of the PI System Explorer (PSE), you can configure a web service delivery endpoint for SOAP or REST web services. To map parameters for the API call, create and configure body of the message, see Configure a REST web service delivery endpoint. For authentication of a web service, see Configure authentication for a web service connection.

Configuring the SOAP web service

The following table lists the options you must set for a SOAP web service delivery endpoint:

SOAP web service options

Option	Description
Web Service Address	The URL of your web service. You can validate the connection using the **Get Web Services** button.
Web Service	The name of the web service to be used for notification.
Default Web Method	Default web method to be used for the notification. This menu displays all of the parameters defined in the web service.

The PI System Explorer user interface automatically retrieves the associated Web Service Definition Language (WSDL) methods, provides useful information about the parameters, and guides you through configuring the web service.

Configuring the REST web service

The following table lists the options you must set for a REST web service delivery endpoint:

REST web service options

Option	Description
URL	The URL of your web service.
HTTP Method	Default web method to be used for the notification. Supported methods are POST, PUT AND PATCH.

Notification rules management

To manage your notification rules, use the Management plug-in, accessible on the bottom left of the main screen of the PI System Explorer. When accessed, the plug-in gives you a choice of two radio buttons to manage either your analyses or your notification rules. Select the **Notification Rules** button to view and manage your notification rules. For more information, go to Management of notification rules. To search for notification rules configured in your database, see Search for notification rules.

Management of notification rules

The **Management** plug-in gives you a choice of two radio buttons to manage either your analyses or your notification rules. Select the **Notification Rules** button to view and manage your notification rules.

Notification Rules pane

The **Notification Rules** pane displays information about selected notifications rules in these columns:

- Status
- Element
- Notification rule name
- Template (Notification rule template)
- Categories

You can select or deselect notification rules from the list of configured notification rules, and monitor individual status on the Notification Rule Details pane; you can also enable or disable selected notification rules from the Operations pane.

Operations, Pending Operations, and Notification Rule Details panes

The other screens on the **Management** plug-in include:

- An **Operations** pane where you can enable or disable notification rules.
- A **Pending Operations** pane that provides information on whether an operation is queued or complete.
- A **Notification Rule Details** pane that is displayed when you select a notification rule, and has details, status and error information about the notification rule. Click on **Notification rule configuration** to go back to the element **Notification Rules** tab.

Search for notification rules

1. Navigate to the Management pane.
2. Select the **Notification Rules** option.
3. Use the provided search or create your own.

To	Then
Use a provided search	Select **All**, **Enabled** or **Disabled**. These are view-only and cannot be modified, as indicated by the 👁 (**View selected search**) icon.
Create a custom search	**Note:** Customized search is only visible to the user

To	Then

	who created it on the computer where it was created. Creating your own search is a new feature in PI AF 2017. Refer to the previous versions of PI System Explorer User Guide if you are using an earlier version of PI AF.
	a. Click the ➕ (**Add new search**) button.
	b. Select a search criteria. Choices are:
	  a. Name
	  b. Description
	  c. Element Name
	  d. Template
	  e. Category
	  f. Enabled/Disabled: enabled or disabled
	  g. Service Status: PI Analysis/Notifications Service status
	    a. Error
	    b. Running
	    c. Stopped
	    d. Suspended
	    e. Warning
	Note: Service Status is based on the PI Analysis Service connection and its running status. This cannot be combined with other search criteria.

View/edit a search	Click the ✎ (**View/edit selected search**) button to view or edit your search.
Delete a search	To delete, click the ✗ (**Delete selected search**) button.

Configuration of notification rules for analyses or event frames

The asset analytics and notifications features of the PI AF server together provide effective ways for you to perform condition-based or predictive maintenance. Notification rules leverage the event frame feature of PI AF to notify you of important events that affect your system.

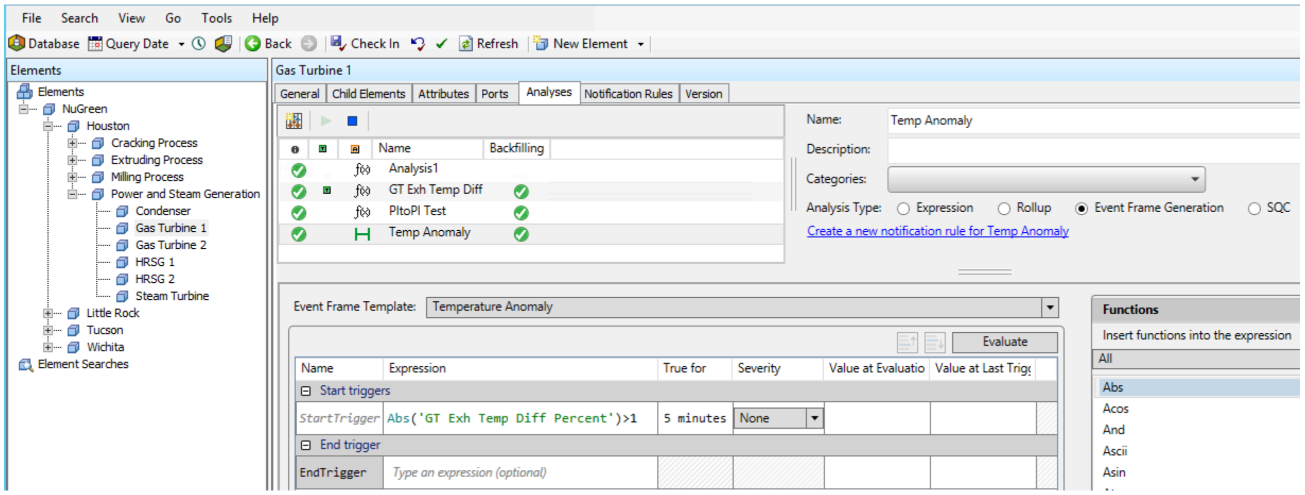
This section guides you in creating notification rules for your analyses or event frames.

Configure notification rules from analyses

If you have already configured analyses to detect anomalies in your PI AF asset data, or you wish to configure such analyses, follow this procedure to configure notification rules on configured analyses.

Note: For information on software and system requirements, see Requirements for notifications. If you want to configure notification rules to trigger on event frames from sources like event frame interfaces or custom applications, see Configure notification rules for user-defined event frames. For information on creating an event frame generation analysis, see Event frame generation analyses.

1. Select the existing event frame generation analysis for which you want to create a notification rule.



Name	Backfilling
Analysis1	
GT Exh Temp Diff	✓
PItoPI Test	✓
Temp Anomaly	✓

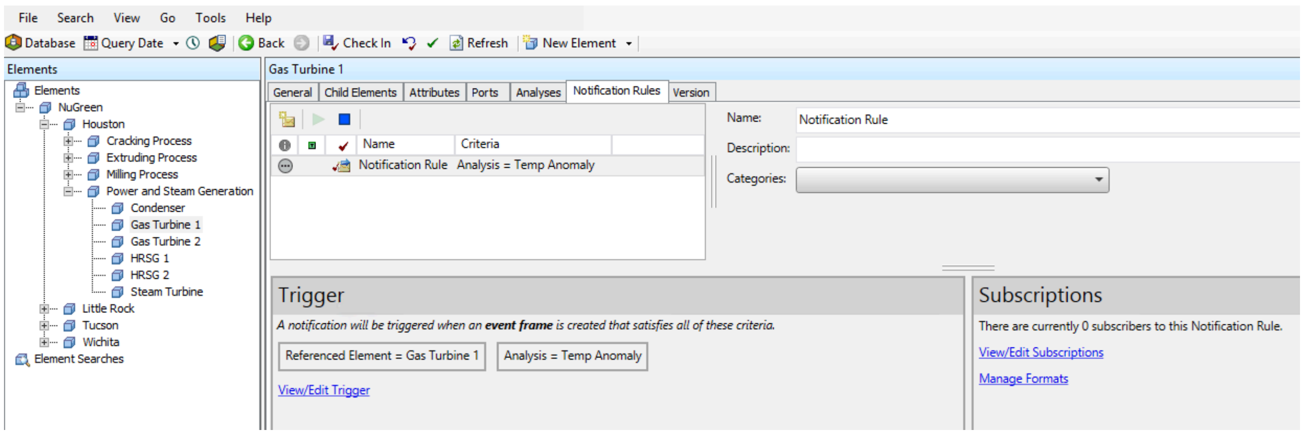
Name: Temp Anomaly
Description:
Categories:
Analysis Type: ○ Expression ○ Rollup [✓ Event Frame Generation] ○ SQC
Create a new notification rule for Temp Anomaly

Event Frame Template: Temperature Anomaly

Name	Expression	True for	Severity	Value at Evaluatio	Value at Last Trig
Start triggers					
StartTrigger	Abs('GT Exh Temp Diff Percent')>1	5 minutes	None		
End trigger					
EndTrigger	Type an expression (optional)				

2. Click **Create a new notification rule for the selected analysis.**

   You see the **Notifications Rules** tab highlighted, as well as the options to configure a notification rule with the default name "Notification Rule".



Name	Criteria
Notification Rule	Analysis = Temp Anomaly

Name: Notification Rule
Description:
Categories:

Trigger
A notification will be triggered when an **event frame** is created that satisfies all of these criteria.
Referenced Element = Gas Turbine 1 | Analysis = Temp Anomaly
View/Edit Trigger

Subscriptions
There are currently 0 subscribers to this Notification Rule.
View/Edit Subscriptions
Manage Formats

3. Edit the created notification rule.

   Provide one or more of the specified criteria; some of your criteria may be pre-selected from your analysis. The notification rule is triggered when all criteria are satisfied. You may change the name of the notification rule from the default, add a suitable description, and select a suitable category.

   Selecting a category aids in organizing and searching for your notification rules.

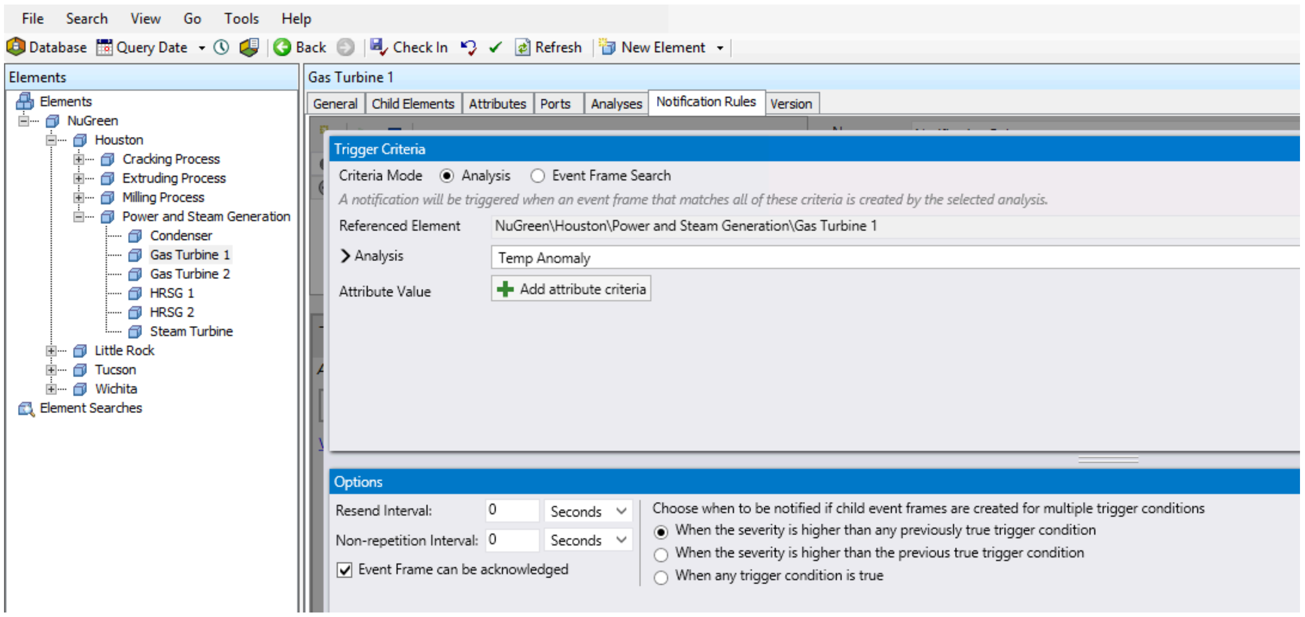
4. Click **View/Edit Trigger**

   The Trigger Criteria pane allows you to configure either of these trigger criteria modes: **Analysis** or **Event Frame Search**. Make sure that the **Analysis** criteria mode is selected.

5. Configure options in the Trigger Criteria pane.

   - You see the details of the selected analysis such as the event frame template name, the start and end triggers, and so on.

- You can configure a new analysis of type **Event Frame Generation** or **SQC**. When you choose to create a new analysis with either the default or a custom name, you see the same analysis creation window as from the **Analyses** tab of PI System Explorer.
- For more information, see Trigger criteria in notification rules.



Criteria Mode [✓ Analysis] Event Frame Search

A notification will be triggered when an event frame that matches all of these criteria is created by the selected analysis.

Referenced Element: NuGreen\Houston\Power and Steam Generation\Gas Turbine 1

Analysis: Temp Anomaly

Attribute Value

Resend Interval: 0 Seconds

Non-repetition Interval: 0 Seconds

[✓] Event Frame can be acknowledged

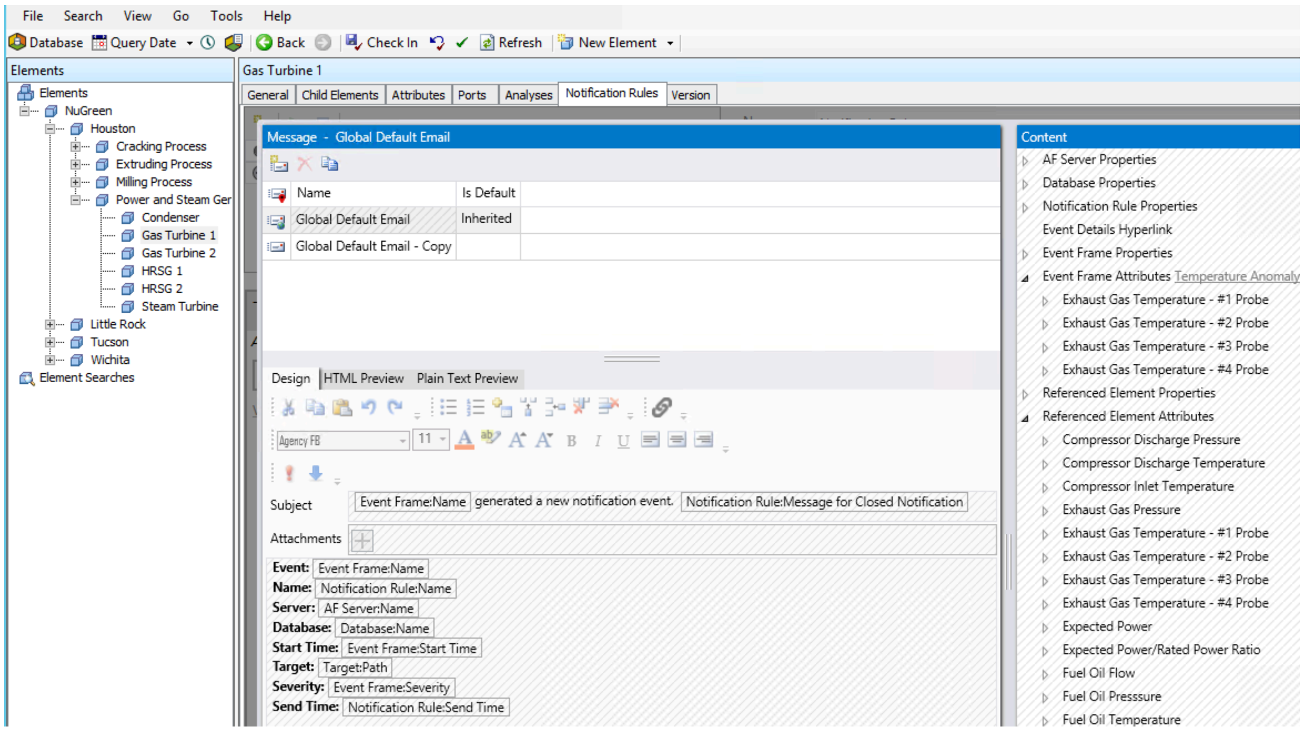
Choose when to be notified if child event frames are created for multiple trigger conditions

[✓ When the severity is higher than any previously true trigger condition]
When the severity is higher than the previous true trigger condition
When any trigger condition is true

6. Configure options in the Options pane.
   - Optional. **Resend Interval**. Select the time interval at which PI Notifications Service will send additional alerts if the event frame matching the notification rule is not acknowledged or closed. For more information, see Resend Interval in notification rules.
   - Optional. **Non-repetition Interval**. Select the time interval for which PI Notifications Service will not send similar alerts associated with the same notification rule. For more information, see Non-Repetition Interval in notification rules.
   - **Event Frame Can Be Acknowledged**. This option is automatically selected if the event frame template has been configured for acknowledgment.
   - Optional. Options based on multiple start triggers. If you have configured multiple start triggers for your analysis, choose between the options of being notified (a) when the current trigger severity is higher than any trigger severity encountered so far, (b) when the current trigger severity is higher than the previous trigger severity, or (c) every time a trigger condition is true, regardless of its relative severity to other previous triggers.
7. Optional. To configure the email format, click **Manage Formats** in the Subscriptions pane.

   You see the Message and Content panes. By default, the notifications feature uses the Global Default Email format for each subscriber. For more information, see Delivery formats in notification rules.

   Note: If the Content pane has a message indicating that your selected format is "read-only", you must first add a new format before you can customize it.



Name	Is Default
Global Default Email	Inherited
Global Default Email - Copy	

Subject: Event Frame:Name generated a new notification event. Notification Rule:Message for Closed Notification

Attachments

Event: Event Frame:Name
Name: Notification Rule:Name
Server: AF Server:Name
Database: Database:Name
Start Time: Event Frame:Start Time
Target: Target:Path
Severity: Event Frame:Severity
Send Time: Notification Rule:Send Time

8. Optional. To create a custom delivery email format, copy the default **Global Default Email** format by clicking on the copy icon in the Message pane. Alternatively, you can create a new delivery format by clicking the new delivery format icon.

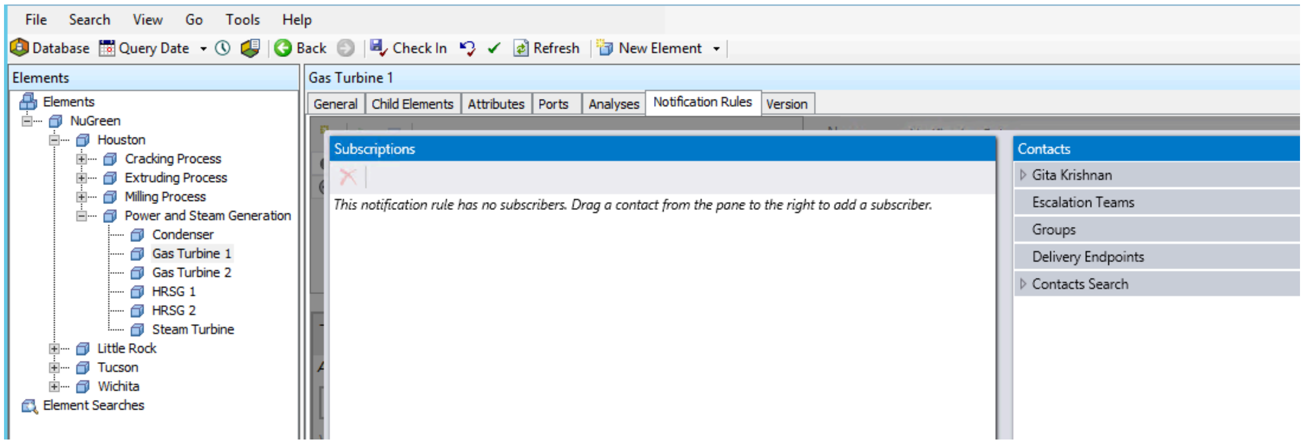
   The default format includes basic notification information such as the notification rule name, trigger time and event frame start time, as well as system information such as system and database names. To add more information to your custom format, drag and drop information from the Content pane to the Message pane.

   Tip: A hyperlink indicates that an event frame template is selected; you may add the attributes defined below the event frame template to the content of your email notification by dragging and dropping the attributes from the Content pane to the Message pane.

   You can designate your custom format as the default by clicking the **Is Default** box in the Message pane.

9. Click **View/edit subscriptions** in the Subscriptions pane.

   You can search for contacts using **Contacts Search** in the Contacts pane. To add a subscriber, drag and drop a specific contact from the Contacts pane to the Subscriptions pane. Make sure to select your customized email format if you have created one, or pick the Global Default Email format. For more information, see Subscriptions in notification rules.



This notification rule has no subscribers. Drag a contact from the pane to the right to add a subscriber.

10. Optional. Configure escalation teams.

    You can configure members of escalation teams from the Contacts plug-in on PI System Explorer, as described in Manage contacts. The escalation team contacts or groups will be notified one by one at the end of time segments specified in **Escalation period**. You can also control the frequency of the escalation notification.

    Note: Escalations are triggered on event frame templates that can be acknowledged. Open the **Library** plug-in on PI System Explorer, select your event frame template and verify that the **Event Frame Can Be Acknowledged** check box is selected.

11. Check in the notification rule.

    When the event of interest occurs, an email will be sent to the subscriber's mailbox. The email shows the event frame that triggered the notification rule with customized details that you have configured.

To manage your notification rules, use the **Management** plug-in on PI System Explorer. See Management of notification rules.

Configure notification rules for user-defined event frames

For information on software and system requirements, see Requirements for notifications.

Note: If you have already configured an analysis on your PI AF asset, or wish to create new analyses to generate the event frames, use the procedure Configure notification rules from analyses.

Use this procedure if you want to configure notification rules to trigger on event frames from sources like event frame interfaces or custom applications.

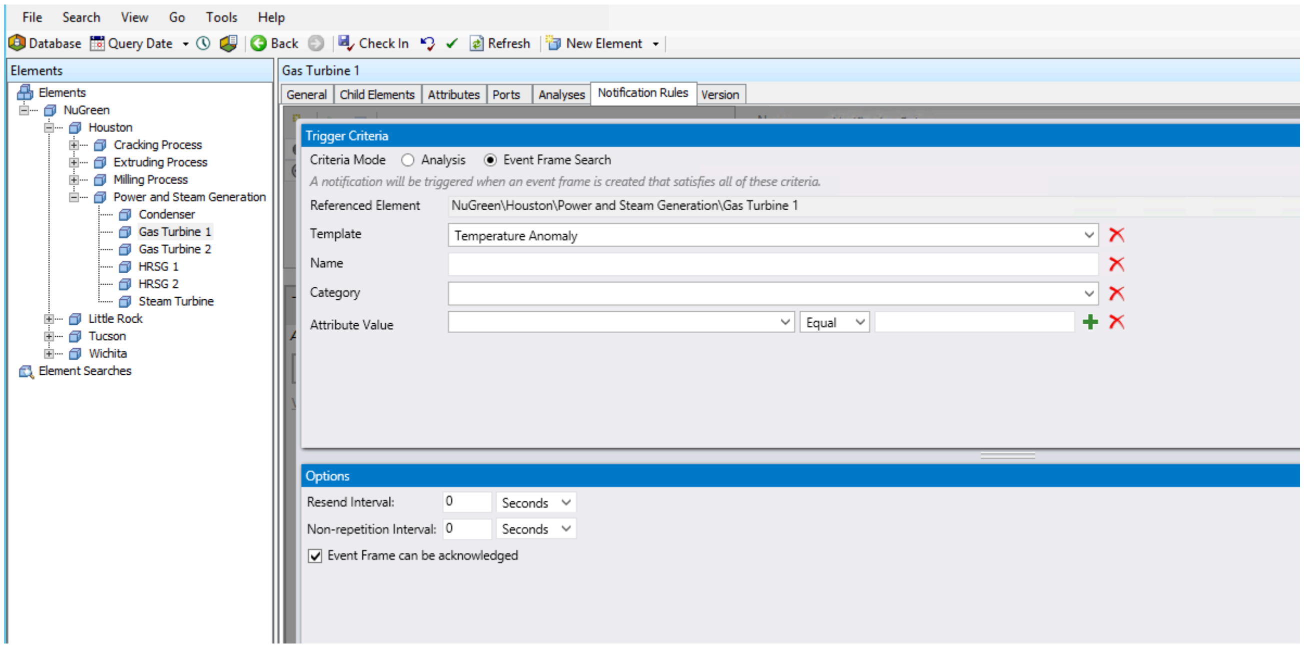
1. In the PI System Explorer navigator, click **Elements** and select the element for which you want to define the notification rule.
2. Select the **Notification Rules** tab.
3. Click **Create a new notification rule**.

   Change the name of the notification rule from the default, add a suitable description, and select a suitable category. Selecting a category aids in organizing and searching for your notification rules.
4. In the Trigger pane, click on the hyperlink **Please configure trigger criteria for this Notification Rule**.

   The Trigger Criteria pane allows you to configure either of these trigger criteria modes: **Analysis** or **Event Frame Search**.

5. Select the **Event Frame Search** trigger criteria mode.
6. Configure options in the Trigger Criteria pane.

   You can select the event frame template from the drop-down list, and add conditions using any of the attribute templates for the event frame template.

   Criteria Mode ( ) Analysis [✓ Event Frame Search]

   A notification will be triggered when an event frame is created that satisfies all of these criteria.

   Referenced Element: NuGreen\Houston\Power and Steam Generation\Gas Turbine 1

   Template: Temperature Anomaly

   Name:

   Category:

   Attribute Value: Equal

   Resend Interval: 0 Seconds

   Non-repetition Interval: 0 Seconds

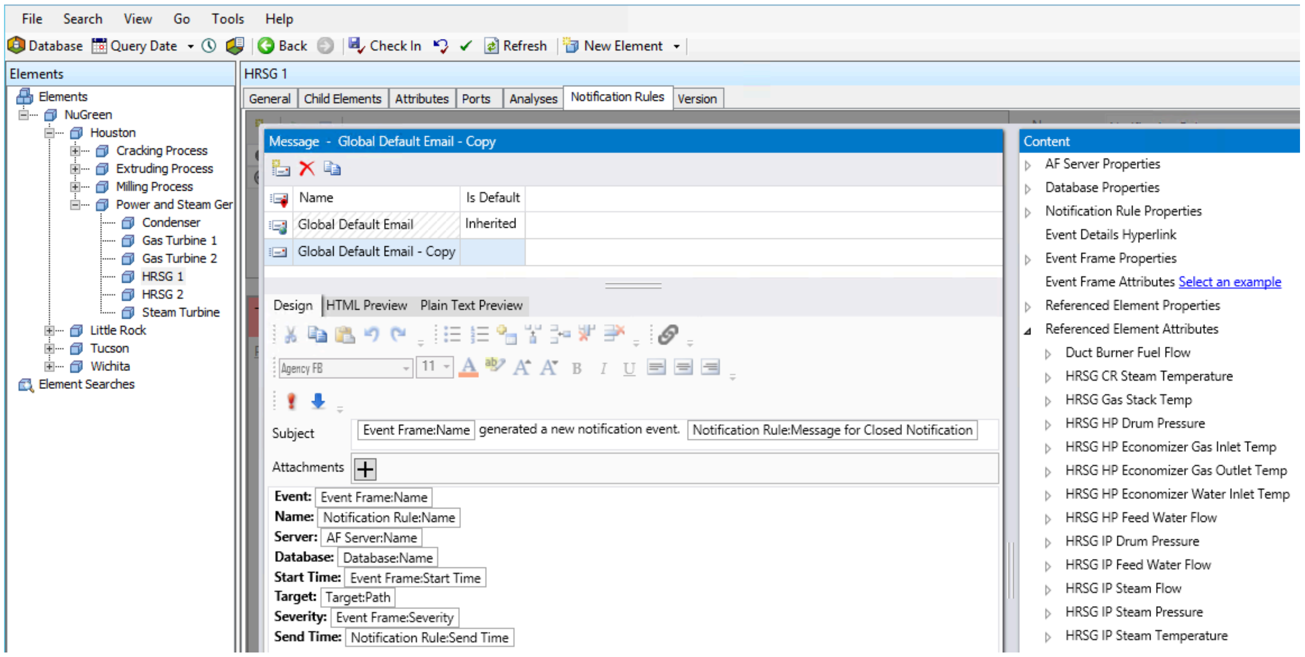
   ☑ Event Frame can be acknowledged

7. Configure options in the Options pane.
   - Optional. **Resend Interval**. Select the time interval at which PI Notifications Service will send additional alerts if the event frame matching the notification rule is not acknowledged or closed.
   - Optional. **Non-repetition Interval**. Select the time interval for which PI Notifications Service will not send similar alerts associated with the same notification rule.

   Note: Verify that the event frame template has the **Event Frame Can Be Acknowledged** check box selected.
8. Optional. To configure the email format, click **Manage format** in the Subscriptions pane.

   You see the Message and Content panes. By default, notifications uses the Global Default email format for each subscriber.

   Note: If the Content pane has a message indicating that your selected format for a subscriber is "read-only", you must first add a new format before you can customize it.


File Search View Go Tools Help
Database Query Date Back Check In Refresh New Element
Elements
Elements
NuGreen
Houston
Cracking Process
Extruding Process
Milling Process
Power and Steam Ger
Condenser
Gas Turbine 1
Gas Turbine 2
HRSG 1
HRSG 2
Steam Turbine
Little Rock
Tucson
Wichita
Element Searches
HRSG 1
General Child Elements Attributes Ports Analyses Notification Rules Version
Message - Global Default Email - Copy
Name
Is Default
Global Default Email
Inherited
Global Default Email - Copy
Design HTML Preview Plain Text Preview
Agency FB 11
Subject
Event Frame:Name generated a new notification event. Notification Rule:Message for Closed Notification
Attachments
Event: Event Frame:Name
Name: Notification Rule:Name
Server: AF Server:Name
Database: Database:Name
Start Time: Event Frame:Start Time
Target: Target:Path
Severity: Event Frame:Severity
Send Time: Notification Rule:Send Time
Content
AF Server Properties
Database Properties
Notification Rule Properties
Event Details Hyperlink
Event Frame Properties
Event Frame Attributes Select an example
Referenced Element Properties
Referenced Element Attributes
Duct Burner Fuel Flow
HRSG CR Steam Temperature
HRSG Gas Stack Temp
HRSG HP Drum Pressure
HRSG HP Economizer Gas Inlet Temp
HRSG HP Economizer Gas Outlet Temp
HRSG HP Economizer Water Inlet Temp
HRSG HP Feed Water Flow
HRSG IP Drum Pressure
HRSG IP Feed Water Flow
HRSG IP Steam Flow
HRSG IP Steam Pressure
HRSG IP Steam Temperature

9. Optional. To create a custom delivery email format, copy the default **Global Default Email** format by clicking on the copy icon in the Message pane. Alternatively, you can create a new delivery format by clicking the new delivery format icon.

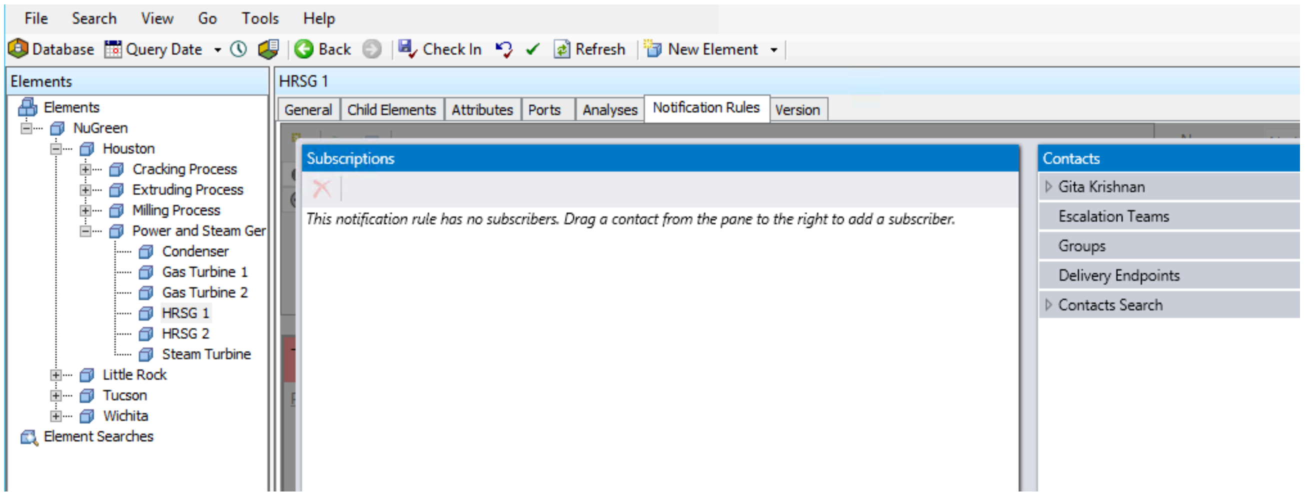
   The default format includes basic notification information such as the notification rule name, trigger time and event frame start time, as well as system information such as system and database names. To add more information to your custom format, drag and drop information from the Content pane to the Message pane.

   Tip: A hyperlink indicates that an event frame template is selected; you may add the attributes defined below the event frame template to the content of your email notification by dragging and dropping the attributes from the Content pane to the Message pane.

   You can designate your custom format as the default by clicking the **Is Default** box in the Message pane.

10. Click **View/edit subscription** in the Subscriptions pane.

    You can search for contacts using **Contacts Search** in the Contacts pane. To add a subscriber, drag and drop a specific contact from the Contacts pane to the Subscriptions pane. Make sure to select your customized email format if you have created one, or pick the Global Default email format.



This notification rule has no subscribers. Drag a contact from the pane to the right to add a subscriber.

11. Optional. Configure escalation teams.

    You can configure members of escalation teams from the Contacts plug-in on PI System Explorer, as described in Manage contacts. The escalation team contacts or groups will be notified one by one at the end of time segments specified in **Escalation period**. You can also control the frequency of the escalation notification.
12. Check in the notification rule.

    When the event of interest is generated, an email is sent to the subscriber's mailbox. The email shows the event frame that triggered the notification rule with the customized details that you have configured.

To manage your notification rules, use the **Management** plug-in on PI System Explorer. See Management of notification rules.

Configure escalations for notification rules

An escalation team is an ordered collection of delivery endpoints or groups. Notification messages are sent sequentially to the members of escalation teams until the event frame matching the notification rule is acknowledged or closed. You can set up an escalation structure to ensure that the right people are notified with the right information, and in the right order.

Subscribe an escalation team to send notification messages sequentially to a group of contacts. For more information on creating an escalation team, see **Create an escalation team** in Manage contacts.

Changes from PI Notifications 2012

The notifications feature is based on event frames and does not have an analysis engine like PI Notifications 2012 and earlier versions (henceforth referred to as "legacy" versions). Notification rules leverage PI Analysis Service, as well as other applications that generate event frames, to identify important events in the PI Server, to use event frames as the container for relevant information for the event, and to use annotations on the event frame to store notification history.

The notifications feature is driven by event frames; therefore, some legacy notification features are either different or rendered obsolete. A migration tool is included to enable migration of notifications from your PI Notifications 2012 legacy version to notification rules, thereby preserving the investment you made in your legacy system.

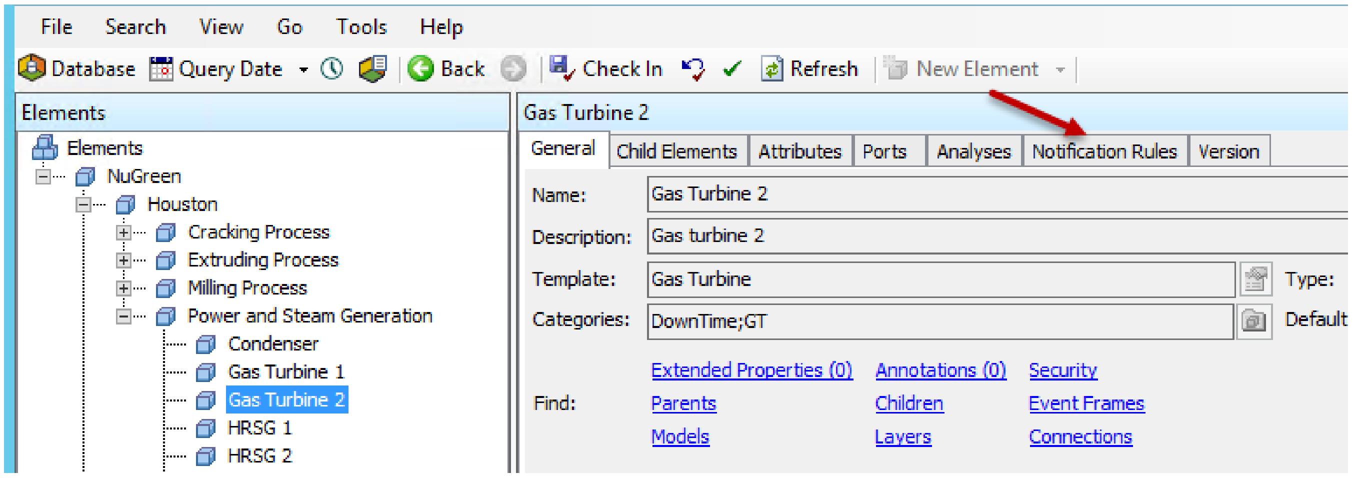
Note: If you choose to migrate, the migration tool converts your legacy notifications into analyses and notification rules.

For more details on the migration tool, see Migration from PI Notifications 2012.

User interface changes from PI Notifications 2012

Notifications in PI Server 2016 R2 (and later versions) is fundamentally different from the legacy PI Notifications 2012, and you cannot share configurations between the two systems.

To configure event-based notifications, you must use the **Notification Rules** tab in the **Elements** view of 32-bit or 64-bit PI System Explorer. To manage legacy notifications, you must use the **Notifications** plug-in on the 32-bit PI System Explorer.


File Search View Go Tools Help
Database Query Date Back Check In Refresh New Element
Elements
Elements
NuGreen
Houston
Cracking Process
Extruding Process
Milling Process
Power and Steam Generation
Condenser
Gas Turbine 1
Gas Turbine 2
HRSG 1
HRSG 2
Gas Turbine 2
General Child Elements Attributes Ports Analyses Notification Rules Version
Name: Gas Turbine 2
Description: Gas turbine 2
Template: Gas Turbine
Type:
Categories: DownTime;GT
Default
Find:
Extended Properties (0) Annotations (0) Security
Parents Children Event Frames
Models Layers Connections

Note: The notifications feature no longer supports the Microsoft Office Communication Service (OCS) or custom delivery channels. However, using REST-based web services, you may be able to integrate notifications with third-party delivery channels.

Migration from PI Notifications 2012

PI Server 2016 R2 and later versions include a migration tool to enable migration of your 2012 notifications to notification rules based on event frames. The migration tool is run at the end of the PI Notifications Service install, and you may choose to migrate your legacy notifications at that time.

You may also run the migration again from PI System Explorer, after the installation is complete. To run the migration tool from PI System Explorer, make sure that all these conditions are true:

- PI Notifications Service is installed on your machine
- The migration tool executable file **PINotificationsMigrationTool.exe** is installed on your machine, typically in the **PIPC\Notification**s folder

The Notifications Migration Tool attempts to migrate all existing notifications by modifying the rule and if necessary prompting you to accept or reject the change if a feature is no longer supported. This section describes the types of notification features that are migrated and those that cannot be migrated.

The migration tool causes these object migrations:

- Legacy notification triggers are migrated as event frame generation analyses and are run in PI Analysis Service.
- Legacy notifications created in previous versions are migrated to notification rules, and they run in the PI Notifications Service.
- Legacy notifications history is fully migrated and created as event frames. History in PI points are migrated as annotations on event frames.
- Legacy notification template name, description and properties are mapped to the notification and target properties; a message is also logged in the migration report.

  Note: Properties of the notification rule template cannot be overridden on a notification rule based on the template with an exception of the description field.

  Due to this, if a legacy notification deviates from the legacy notification template, that information may be lost during migration. For example, if the legacy notification template is named "Notification for Pumps", but a legacy notification based on this template is named "Notification for Pumps - Houston", the migrated notification rule template and all notification rules based on this template will be named "Notification for Pumps". The description field of the migrated notification rule, however, will contain the name of the legacy notification. The migration report will contain the name changes and any other changes that have been made.

The migration tool does not migrate the following legacy notifications:

- Custom delivery channels
- Alarm functions
- Monthly and daily period time rules
- Notifications with the **Notify only on change in status option** unchecked.
- Notifications containing a nested escalation
- Notifications with multiple triggers:
  - If a legacy notification has multiple triggers and you have assigned states from different state groups, the legacy notification is not migrated.
  - If a legacy notification has multiple triggers and you have assigned the same state to more than one trigger, the legacy notification is not migrated.

  Note: For a list of legacy notifications that are not migrated or migrated with some loss of content, see the knowledge base article: Caveats in Migrating from Notifications 2012 to 2.x.

Training video

For more information on migration, watch the AVEVA PI System Learning channel video:

Running the migration tool

If you are running the migration from PI System Explorer, make a backup of the PIFD database before starting the migration. For instructions on how to perform a backup, see the *PI Server Installation and Upgrade* guide topic: Backup job. Note that you can also migrate notifications when you are upgrading PI Server.

Before you begin migration you must verify that you have read permissions on the Data Archive where the legacy notifications history is stored and read/write permissions on the PI AF database where the legacy notifications are configured.

You may need to back up the PIFD (SQL database) if you want to re-run the migration and avoid having duplicate notification rule and analysis objects created. If you re-run your migration and are warned that duplicates may be created during the re-migration, you can first restore your PIFD database from the backup and avoid the creation of duplicates.

You can re-run the migration tool by clicking **Tools** > **Notifications Migration** from PI System Explorer. The migration tool first determines and displays information on your state, which is one of:

- **Not migrated**

  Legacy notifications may be created and managed using the **Notifications** plug-in on the bottom left of the PI System Explorer window. You must use the 32-bit version of PI System Explorer to manage legacy notifications.

  You can choose to be in the "Not migrated" state by choosing the appropriate option while running the migration tool. In this state, you cannot create new notification rules.
- **Migrating**

  Legacy notifications are read-only; only new notification rules may be created. You can stop and start legacy notifications while in this state.

  In the migrating state, you may run and test legacy notifications and new notification rules side-by-side to ensure that the migration is successful and that the new analyses and notification rules are working as intended. You can check the migration report, fix the errors indicated on legacy notifications, and then choose to re-run migration for only those legacy notifications that encountered errors in the previous migration run.

  You may move to the migrated state when you have successfully tested that your migrated notification rules match your legacy notifications. Move to the migrated state only when the following are true:

  - You intend to continue to work only with the new notification rules
  - You will never need to revert to your legacy notifications
  - You have successfully migrated all the legacy notifications that you wanted to migrate
- **Migrated**

  When you move to this state, you can create only new notification rules in the system. Your legacy notifications are deleted and you cannot revert to your legacy notifications or create legacy notifications.

  From the migrated state, you may re-run the migration tool to revert to the "Not migrated" state; however, your legacy notifications cannot be recovered.

After all legacy notifications have been converted, you may want to run both versions of notifications in parallel for a while to ensure that the notification rules are functioning as expected. When you are satisfied with the migrated notifications, run the migration tool again and select **Complete migration and delete all legacy notifications from server**. After running the tool you can also uninstall PI Notifications 2012. It is strongly recommended that you uninstall legacy PI Notifications after any transition period.

If you do choose to run both versions of notifications simultaneously for a period of time, be aware that each notification may be sent twice, once from the PI Notifications Scheduler, which is part of the legacy PI Notifications, and once from the PI Notifications service. If you do not want to receive duplicates during this transitional period, you can disable the PI Notifications Scheduler.

You must manually create notification rules for any unsupported legacy notifications that were not migrated.

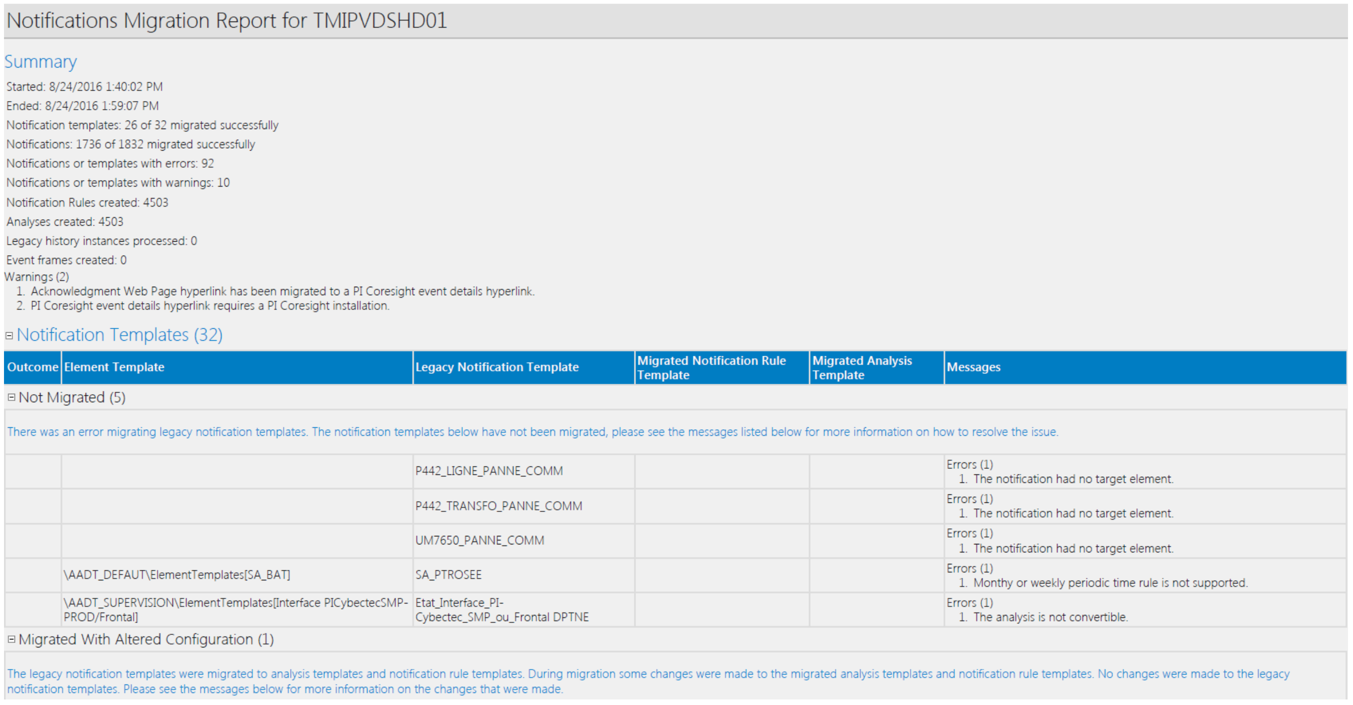
Note: The Notifications Migration Tool only needs to be run once against an AF Collective.

Sample migration report

The migration tool generates a report when it completes the migration process. The report includes information on whether legacy notifications are migrated successfully. The "Errors" section and the "Not Supported" sections indicate notifications that are not migrated. The "Warnings" section indicates notifications with changed functionality, and the report provides more information.

Note: Migration report can be found in **%userprofile%\Documents** folder of the person running the migration.

This is a sample migration report summary:



Notifications Migration Report for TMIPVDSHD01

Summary

Started: 8/24/2016 1:40:02 PM
Ended: 8/24/2016 1:59:07 PM
Notification templates: 26 of 32 migrated successfully
Notifications: 1736 of 1832 migrated successfully
Notifications or templates with errors: 92
Notifications or templates with warnings: 10
Notification Rules created: 4503
Analyses created: 4503
Legacy history instances processed: 0
Event frames created: 0
Warnings (2)
1. Acknowledgment Web Page hyperlink has been migrated to a PI Coresight event details hyperlink.
2. PI Coresight event details hyperlink requires a PI Coresight installation.

⊟ Notification Templates (32)

Outcome	Element Template	Legacy Notification Template	Migrated Notification Rule Template	Migrated Analysis Template	Messages
⊟ Not Migrated (5)					
There was an error migrating legacy notification templates. The notification templates below have not been migrated, please see the messages listed below for more information on how to resolve the issue.					
		P442_LIGNE_PANNE_COMM			Errors (1) 1. The notification had no target element.
		P442_TRANSFO_PANNE_COMM			Errors (1) 1. The notification had no target element.
		UM7650_PANNE_COMM			Errors (1) 1. The notification had no target element.
	\AADT_DEFAUT\ElementTemplates[SA_BAT]	SA_PTROSEE			Errors (1) 1. Monthy or weekly periodic time rule is not supported.
	\AADT_SUPERVISION\ElementTemplates[Interface PICybectecSMP-PROD/Frontal]	Etat_Interface_PI-Cybectec_SMP_ou_Frontal DPTNE			Errors (1) 1. The analysis is not convertible.
⊟ Migrated With Altered Configuration (1)					
The legacy notification templates were migrated to analysis templates and notification rule templates. During migration some changes were made to the migrated analysis templates and notification rule templates. No changes were made to the legacy notification templates. Please see the messages below for more information on the changes that were made.					

In addition to the summary, the report also contains information on legacy and migrated notification rules and templates, and specific information such as notification rules and analysis names, number of migrated event frames, and the number of processed history events.

Additional resources

Training video

For more information on using notifications in PI Asset Framework, watch the videos on the AVEVA PI System Learning channel playlist: