PUBLIC

Document Version: 15.5.3 – 2022-05-20

SAP Plant Connectivity

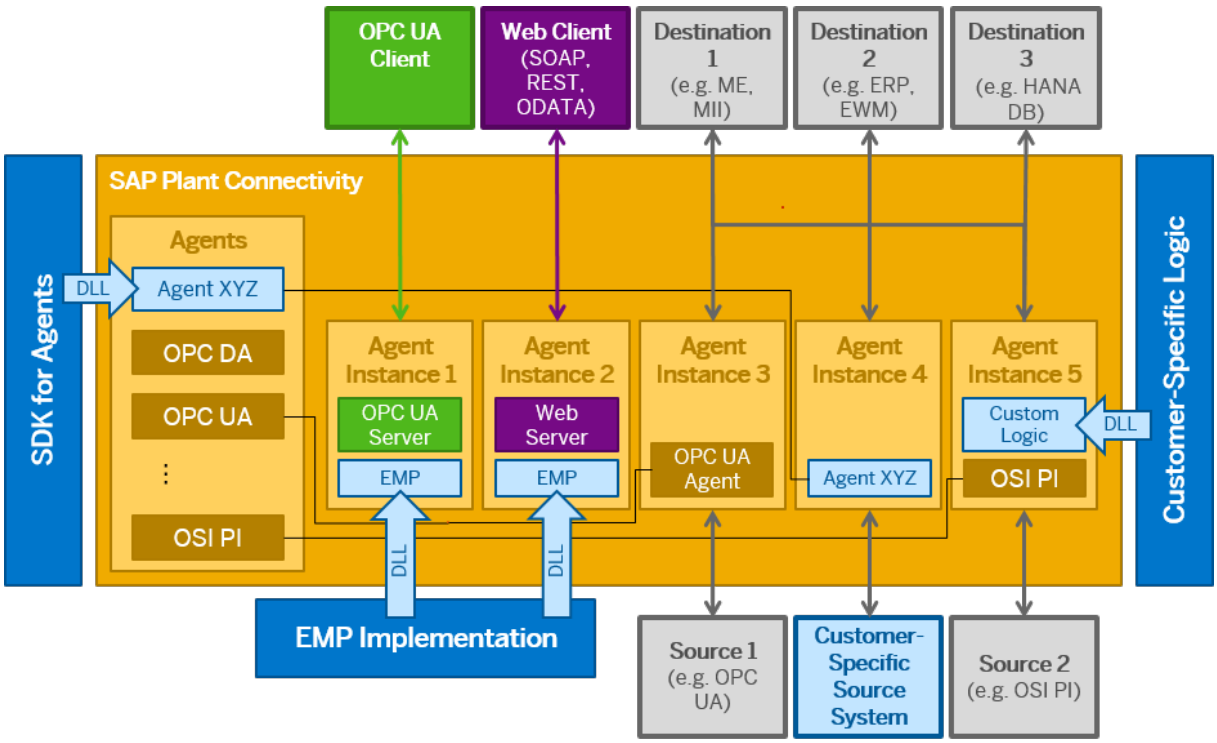
THE BEST RUN SAP

Content

1 SAP Plant Connectivity

Product Information

Product	SAP Plant Connectivity
Release	15.5 (SP03)
Based On	`.Net-Framework` as of version 4.8
Documentation Published	May 2022



Overview of SAP Plant Connectivity

With *SAP Plant Connectivity (PCo)*, SAP provides a software component that enables the exchange of data between an SAP system and the industry-specific standard data sources of different manufacturers, for example, process control systems, plant Historian systems, and SPC systems.

With PCo, you can receive tags and events from the connected source systems in production either automatically or upon request and forward them to the connected SAP systems. Furthermore, using PCo, you can execute, receive, and process OPC UA method calls and thereby also map complex coordination tasks in the programmable logic controllers (PLC) environment. The PCo Web server extends these options and allows for self-defined methods that can be called as Web service operations.

The *SAP Plant Connectivity* component has the following advantages:

- PCo can be configured quickly. No additional developments of your own are necessary in order to connect source systems in production.
- PCo can communicate with different software systems on the shop floor.
- A PCo system can be configured for one or multiple locations.
- The flexible, method-based connection of servers and clients in production is possible using the OPC UA standard or Web service standards.

The *SAP Plant Connectivity* component supports various processes: For more information, see Processes and Integration Scenarios with SAP PCo [page 34].

Implementation Considerations

- PCo is a Windows-based application that builds on the `.NET-Framework`. PCo supports various Web service standards.
  To be able to implement PCo, you must first have installed `.NET Framework 4.8` or a higher release.
- PCo requires *Visual C++ Redistributable* for *Visual Studio 2019* as a *Microsoft* runtime library. This is installed when PCo is installed.
- If PCo is used in the notification process, you can manage queues for the messages using *Microsoft Message Queuing (MSMQ)*. You can install the Windows component automatically when installing PCo. (See also: Storage Method [page 486].)
- If the source system works with OPC DA interfaces, you need to download and install the corresponding OPC software from the manufacturer or from the OPC Foundation. The libraries for connecting OPC UA servers and clients are already integrated in PCo.

Integration

Source Systems

Using agents, PCo supports all current software interfaces to be able to receive data from the source systems. An agent is a .NET DLL assembly component that can establish the connection between the data source and PCo.

PCo agents support the following standardized software interfaces:

- OPC Data Access
- OPC Historical Data Access
- OPC Alarms & Events
- OPC Unified Architecture
- OLE DB
- Open Database Connectivity (ODBC)
- Modbus
- MQTT
- GE Fanuc Proficy Historian

- OSIsoft PI
- Asset Framework
- Socket
- IP21
- Citect
- File Monitor Agent
- File System Agent

> **i Note**
>
> You can use the *Software Developer Kit (SDK)* to develop additional agents of your own. The SDK is a development package that is supplied in addition to PCo.
>
> You can create additional agents that are adapted to the special requirements of particular data sources such as weighing systems and printing machines.

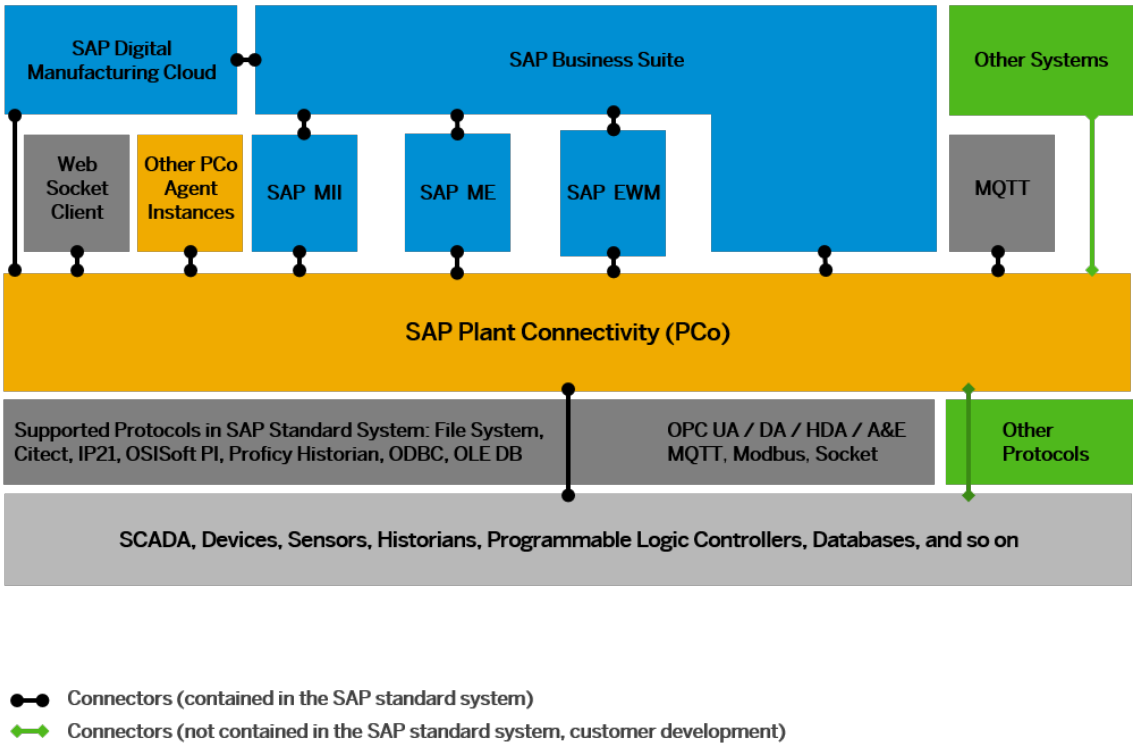
Destination Systems

PCo can send the data received to the following systems:

- SAP Manufacturing Integration and Intelligence (SAP MII)
- SAP Manufacturing Execution (SAP ME)
- Business Suite system, for example, SAP ERP or SAP EWM
- SAP HANA database
- MS SQL Server
- MQTT
- OPC UA client (method-based)
  You can also run PCo as the server (see: PCo as OPC UA Server and as Web Server [page 54]). It then supports the following protocols:
  - WebSocket server
  - OPC UA server (method-based)
  - Web server

The following graphic provides an overview of the integration of the Plant Connectivity system in the system landscape.

Integration of PCo in the System Landscape



Features

Plant Connectivity Management Console

The *Plant Connectivity Management Console* is the central administrative tool for setting up and monitoring Plant Connectivity.

You can create and configure the following elements in the PCo Management Console:

- Source System [page 129]
- Destination System [page 270]
- Agent Instance [page 412]
- Notification [page 496]

Monitoring

- Monitoring and administration
  The *Remote Client* is available for monitoring and administration of your PCo systems in production. The *Remote Client* is a snap-in of the *Microsoft Management Console (MMC 3.0)*, which is provided by SAP together with the *SAP Plant Connectivity* component. With the *Remote Client*, you can monitor all PCo systems - also on remote computers - with their individual agent instances at a glance as well as start and stop the agent instances. See also: Remote Client [page 602]
- Active monitoring

In addition to monitoring with the *Remote Client*, PCo offers active monitoring. During productive operation, all agent instances can be monitored automatically so that outages in the communication connection can be reported immediately.

2 Basics

2.1 Tags

Use

All the data for a source system with which PCo can connect is stored in the form of tags. Each tag is identified by a unique ID and contains the following information:

- Value, for example, measured value
- Specification of the data quality for the value
- Time stamp

The tags are stored in a hierarchy in a node structure. The tags themselves form the lowest level of the hierarchy.

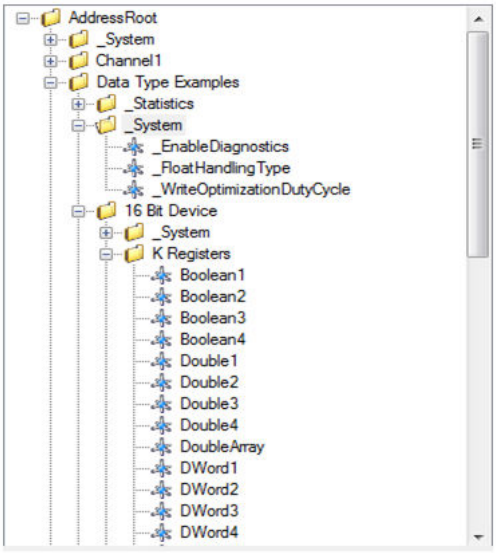
You can subscribe to specific tags in a source system to be able to monitor these tags for possible value changes. (See: Notification Process (with SAP MII) [page 35].)

Queries can read or write individual tag values or determine information about the tag structure. (See: Query Process (with SAP MII) [page 43].)

Example

The following graphic shows the hierarchical structure of the data in a source system. The individual nodes (folder symbols) can represent the sensors of a plant, for example. The individual tags - such as Boolean 1, Boolean 2, and so on in the example here - are on the lowest level of the hierarchy.

Tags



2.2 Methods

Definition

A method is a key element of a service-oriented architecture for industrial applications. Using a method, a server provides an interface with which clients can call specific services or data from the server. The server defines the semantic of the method and which input and output parameters it has. It makes sure that when the method is called, the desired action is executed and that the output parameters are supplied after the action has run.

You can also run an agent instance as a server in PCo (see: PCo as OPC UA Server [page 54] or PCo Web Server [page 464]), and configure methods at this level. However, PCo also supports the calling of methods as a client of a remote server or within your own installation using the OPC UA destination system (see: OPC UA Destination System [page 333]).

If you use PCo as a Web server, you can provide methods for callers in the form of service operations. Service operations of a remote server are accessible via the universal Web service destination system or within your own installation.

Example

Example 1

A robot offers a method `RoboterExecuteJob` with which the execution of a specific robot program can be triggered. A method like this could have the following interface:

Input Parameter	Description
`inHandle`	Unique ID for calling the robot program. This might be the serial number of the product being produced.
`inJobNumber`	Number of the robot program that is to be executed

Output Parameter	Description
`outSuccess`	**True** if execution was successful, **False** if not. An error number is issued at the same time.
`outErrorNumber`	Error code to which there can be a reaction in further processing

You can call such a method in a process-controlled way using PCo and receive and process the return value further after successful execution, for example, in the multiple call destination system.

Example 2

PCo offers a method `CarrierArrived` that is called by the transport system if a carrier has reached a specific position in the production plant:

Input Parameter	
`inHandle`	Unique ID of the product on the carrier
`inPos`	ID of the position reached on the transport system

PCo can then start a configured process if this method is called at a specific position, for example, trigger a robot movement or start a drill.

2.3 Tools for System Settings

- Managing System Settings [page 13]
  Under *Tools* > *Options*, you can make basic global and user-specific settings such as language, date/time format, displaying of logs.
- Registering PCo in the SLD [page 22]
  You can register your PCo system in the *System Landscape Directory system*.

- Authorization Management [page 24]
- Settings for Cloud Integration [page 26]

2.3.1 Options

To call the basic system settings, in the menu choose *Tools* > *Options*. You have the following options:

- Global Settings [page 13]
  The global settings that you make affect **all Plant Connectivity users** on this computer.
- User Settings [page 19]
  The user settings only apply to the user who is currently logged in.

2.3.1.1 Global Settings

> **i** Note
>
> The **global settings** affect all Plant Connectivity users on this computer.

1. To make the global settings, choose the menu entry *Tools* > *Options* in the *Plant Connectivity Management Console*.
   The *Options* dialog box appears.
2. If you click the **system ID** option, you can change the system ID of your PCo instance. See also: Specifying the System ID [page 13]
3. Choose **Main Service** to make the settings for this central Windows service. The main service provides various functions for remote applications and systems as well as for PCo itself. (See: Configuring the Main Service [page 14].)
4. Click on **Compatibility**. In the compatibility settings, you can then allow the creation of deprecated configuration elements or the use of the deprecated expression parser. (See: Compatibility Settings [page 17].)
5. Click *OK*.

> **i** Note
>
> The *Reset All Settings to Defaults* pushbutton resets your changes to the default values, or to the values before the current changes were made.

2.3.1.1.1 Specifying the System ID

Use

You can specify a *system ID* for your PCo Management Console under *Global Properties*. The system ID of the PCo Management Console is required for SLD registration. See also: Registering PCo in the SLD [page 22]

Features

You can specify the system ID for PCo under *Tools* > *Options* > *Global Properties* > *System ID*. **PCO** is automatically the default value in the *System ID* field after an upgrade, a reinstallation, or a migration. You can change this value.

> ⚠ Caution
>
> If you change the system ID, all agents that are running are stopped and the PCo Management Console is restarted automatically.

2.3.1.1.2 Configuring the Main Service

Use

The main service is a Windows service and the host for the services that PCo provides for connecting remote systems or the Digital Manufacturing Cloud. The main service also hosts the configuration services, which are required to access the configuration of PCo.

During the installation, the main service is set up with the unchangeable *Start Mode* **Automatic** and must always be executed in order for PCo to function correctly.

Procedure

1. If you want to adjust the main service settings within the Management Console, choose the menu entry *Tools* > *Options* > *Main Service* in the Plant Connectivity Management Console.
   The dialog for configuring the main service is also started automatically if the system determines that there is no access to the configuration services when the Management Console is started.
2. Activate the services that you require to run PCo:

   Screen Area: Web Services Hosted by Main Service

Services	Description
Configuration Services	The configuration services are essential for running PCo and therefore cannot be deactivated. If there are problems with the port number specified during the installation, you can enter a more suitable free port number here.

Services	Description
Management Services	The management services provide various services for configuring PCo, for starting and stopping agent instances, and for accessing the log. They are required for connecting PCo to SAP MII and for the Remote Client. (See also: Management Services [page 16].)
Cloud Services	The cloud services are needed for integration with the **Digital Manufacturing Cloud**. Here you can specify the *port* that is to be used for the cloud services. Check that the *Active* checkbox is selected. This enables cloud integration. You make the remaining settings in the menu under *Tools* > *Cloud Integration*. (See also: Settings for Cloud Integration [page 26].) **i Note** The configuration options for the cloud services are only displayed if you have selected the *PCo Cloud Management* component when installing PCo.

You can make the changes while the main service is running. If you close the settings dialog with *OK*, the changes are saved and the main service is automatically restarted so that the changed settings take effect. If you have changed the port for the configuration services, the Management Console is then also restarted.
The configuration of the main service is also saved if you start the main service manually via the configuration dialog.

i Note

When the main service is started, it automatically generates certificates that secure the user's logon to the configuration services. These certificates are valid for one year. If the certificates have expired, PCo remains functional, but warnings are logged for the main service. The certificates are automatically renewed when the main service is restarted, for example, when the computer is restarted or when a patch is installed for Plant Connectivity. If required, you can manually renew the certificates by stopping and restarting the main service by choosing *Restart Main Service*. The remaining runtime of the certificates is displayed in the certificate overview. (See also: Certificate Overview [page 592].)

2.3.1.1.2.1 Management Services

Use

The management services provide various services for configuring PCo, for starting and stopping agent instances, and for accessing the log. They are required for connecting PCo to SAP MII and for the Remote Client. (See also: Remote Client [page 602].)

Procedure

1. Choose a suitable free port number for the services and select the *Active*checkbox.
2. In the *Host Name* field, enter the name of the server on which you have installed PCo. We recommend that you use the fully qualified server name here. You can generate this automatically by choosing *Generate Fully Qualified Host Name* (to the right of the *Host Name* field).

   > **i** Note
   >
   > The name that you define here for the server is also used as the host name for the following PCo applications:
   >
   > - As host name for the WebSocket server
   > - As internal host name for the socket connections to SAP MII
   >
   > Usage of the same host name increases the reliability of the data connections and creates the prerequisite for using PCo in high availability data clusters. Furthermore, PCo configurations can be exchanged more easily between installations because the dependency of the configuration on the host name is no longer a factor.

3. If you want to set up a secure connection using **TLS**, select the *Use TLS for Connections* checkbox and select a server certificate with a private key in the *Certificate* field. The server certificate must be suitable for server and client authentication and should ideally have been issued by your organization's intermediate certificate authority.

   > **i** Note
   >
   > When the configuration is saved, PCo assigns the selected certificate to the configured port using the command `netsh http add sslcert`. If you deactivate TLS, PCo executes the command `netsh http delete sslcert` in the background to remove the certificate from the port. The system displays the details of the executed command and the resulting output in an information message.

4. PCo generates the URL required for Web service access from the server name and port. The URL is static and is formatted as follows: `http://<server>:<port>/PCoManagement`. If you have enabled TLS for a secure connection, the URL starts with `https`.
   By choosing the *Copy WSDL URL to Clipboard* pushbutton, you can copy the WSDL URL for Plant Connectivity to the clipboard and from there enter it into the address row of a browser or another configuration dialog.

5. If you choose the *Open WSDL URL in Internet Browser* pushbutton, you can open the WSDL URL, which is provided by the Management Host, in the browser.

2.3.1.1.3 Compatibility Settings

You can make settings here that ensure compatibility with earlier PCo versions.

> **i** Note
>
> However, SAP recommends that you only use this switch in exceptional cases and retain the new default settings, if possible.

Settings

Field	Meaning
Allow Creation of Deprecated Configuration Elements	If you select this checkbox, you can still create the source and destination systems that are flagged as deprecated: • File monitor source system (successor: File system source system) • Web service destination system (successor: Universal Web service destination system) • MII destination system (successor: Universal Web service destination system, multiple call destination system, or OPC UA destination system) The specified source and destination systems are still functional. However, SAP reserves the right to only correct new program errors in the successor system types, or to no longer support the system type in future releases.

Field	Meaning
Use Deprecated Expression Parser	If you select this checkbox, PCo uses the expression parser that was standard before PCo 15.2.2. The purpose of this compatibility checkbox is to ensure that existing PCo configurations, which possibly rely on the faulty behavior of the old parser, still deliver the same calculation results after a migration to PCo 15.2.2 or higher. In the deprecated expression parser, the order of the logical operators `'=='` and `'!='` is too low compared to other operators. This means that a logical expression that is in itself correct, such as `'a' == 0 || 'b' == 0`, can only be evaluated correctly if it is enclosed in brackets as follows: `('a' == 0) || ('b' == 0)` **i Note** If you generate new PCo configurations, you should not select this checkbox.
Allow Verbatim String and Escape Sequences	If you select this checkbox, you can use the verbatim string @ and escape sequences in the expression editor to define string literals and number literals. **i Note** This checkbox is not selected by default in PCo. If you have set up a new PCo instance (version 15.5.3) in **SAP Digital Manufacturing Cloud**, the checkbox is selected automatically by the cloud configuration. In **SAP Digital Manufacturing Cloud**, if you upgrade from an earlier PCo version to PCo 15.5.3, the checkbox is **not** selected automatically. You have to then select the checkbox yourself to be able to use the verbatim string and the escape sequences. **⚠ Caution** Expressions that you created in earlier PCo versions are interpreted as if this checkbox had not been selected. Therefore, if you select this checkbox, you need to manually adjust agent instance and destination system configurations from PCo versions prior to 15.5.3. For more information, see String Literals in Expressions [page 551].

Field	Meaning
OPC UA Server: Allow Usage of Deprecated Security Options	If you select this checkbox, you can continue to use the deprecated security policies **Basic256** and **Basic128Rsa15** for OPC UA. (See also: Select Endpoint [page 151].)

2.3.1.2 User Settings

> **i Note**
>
> The user settings that you make here only apply to the user who is currently logged on.

1. In the *Plant Connectivity Management Console*, choose the menu entry ▶ *Tools* > *Options* > *User Settings* ▶.
2. Click on *Language* and choose the **language** in which you want to display the *PCo console*.
   The *PCo Management Console* is restarted using the new setting as soon as you close the dialog box by choosing OK.
3. To select the language in which you want to display the application help, choose *Online Help*.
4. If you want to adjust the **date and time**, click *Date/Time Format*. Select the *Use Custom Date/Time Format* checkbox and enter the required data.
5. If you click **Browsing**, you can specify the number of tags that you want to be displayed on the *Subscription Items* tab in the agent instance when browsing. The default value is set as *100*, meaning that the search returns 100 tags. You can change the value here.

   > **⚠ Caution**
   >
   > If you set the number too low, the source system will be inundated with calls. If you set the number too high, you will not be able to cancel a browse effectively.

6. To change the **status** of the agent instance, select the *Update Status on Startup* or *Update Status Periodically* checkboxes. In the *Update Status 'Every'* field, you can specify that you want the status to be updated every 3 seconds, for example.

   > **i Note**
   >
   > The default setting for the update of the status is every 3 seconds. If more than one PCo system is running on a server, you must increase this time since the PCo systems become too slow due to the frequent status updates.

Related Information

Configuring the Log Display for the Agent Instance [page 20]
Configuring the Audit Log Display [page 21]
Enhanced Error Handling [page 21]

Settings for the Expression Editor [page 22]

2.3.1.2.1 Configuring the Log Display for the Agent Instance

Use

PCo creates a log for each agent instance and displays this on the *Log* tab of the agent instance. (See Agent Instance: Log Tab [page 415].)

Here you can specify the columns you want to display on the *Log* tab of the agent instance.

Procedure

1. In the *Management Console*, choose *Tools* > *Options* from the menu.
2. Under user settings, choose *Log Display*.
3. Select the checkboxes for the columns you want to display (or deselect those you do not wish to display). Choose *OK*.

   > **i Note**
   >
   > All columns are selected by default and so all columns are shown in the log display.

4. If required, choose the option *Automatically Adjust Column Width* if you want to adjust the column width of the log display automatically to fit its content.
5. If required, select a different separator to use as the column separator when exporting the log to a CSV file. You can choose one of the following options:
   - Semicolon
   - Comma
   - Vertical bar
   - Tab

   > **i Note**
   >
   > During the export, any separators in the message text are replaced with another character to prevent unwanted formatting in the CSV file.

For more information about the meaning of the columns, see Agent Instance: Log Tab [page 415].

2.3.1.2.2 Configuring the Audit Log Display

Use

The audit log documents all changes that the users have made, for example:

- Date/time of change
- User who made the change
- Object changed
- The operation that was carried out, such as *Add* or *Update*
- Previous value and new value

To display the audit log, choose ▶ *Display* ▶ *Audit Log* ▶ from the menu.

Procedure

To specify which columns are displayed in the audit log, proceed as follows:

1. On the *Plant Connectivity Management Console* screen, choose ▶ *Tools* ▶ *Options* ▶ *Audit Log Display* ▶ *Columns* ▶.
2. Select or deselect the checkboxes and choose *OK*.

2.3.1.2.3 Enhanced Error Handling

If errors occur when you are configuring PCo using the Management Console, error icons are displayed within tables at the beginning of the row in which the error occurred. The error message is displayed as a quick info that you can see when you hold the mouse over the error icon. Similar error handling is also often used for input fields. Errors are symbolized by an error icon next to the affected field and the error text is available as a quick info.

To support users with visual or motor impairments, you can also display such visualized errors in a separate dialog box in an overview. You can make the required settings here.

Settings for Error Handling

Field	Description
Enable Log Display for Input Errors	If you select this checkbox, a pushbutton is provided next to the pushbutton for the application help (on the screens on which the function is supported). You can then display the errors in a dialog box. The pushbutton is active as soon as there are error messages on the screen. You can also use the F7 function key to call the dialog box.

Field	Description
Start Numbering of Table Rows with Zero	In the speech output of tables, the first row is called row zero. You can use this option to adjust the row number display in the error dialog so that it also begins with zero.
Show Additional Windows Notification for New Errors	If you also select this checkbox, a Windows notification is issued when a new configuration error occurs. In the Windows system settings, you can define how long the notification is displayed. If you click on the notification, the dialog box also opens to display the error.
Min. Notification Interval	Indicates the minimum time in seconds between consecutive notifications. This helps you avoid having too many notifications issued when a screen is being maintained. The minimum value is 0. In other words, you receive a notification as soon as the error situation changes on the current screen. **i Note** The setting is only effective for the current screen. If you switch to another screen on which enhanced error handling is active, the time interval starts again.

2.3.1.2.4 Settings for the Expression Editor

You can make the following settings here:

- Use simple text editor to support screen readers
- Use enhanced text editor
- Highlight syntax of expressions
  You use this setting if you want to use syntax highlighting for expressions. Syntax highlighting is set by default, but can be deactivated here. (See also: Syntax Highlighting in the Expression Editor [page 554].)

2.3.2 Registering PCo in the SLD

Prerequisites

You have updated the *SAP NetWeaver System Landscape Directory (SLD)* with the current content of the *Component Repository*. For more information, see SAP Note 669669.

Context

You only perform a *System Landscape Directory registration* in the *PCo Management Console* if you want to monitor several distributed PCo systems in the Remote Client and want to manage the PCo instances centrally in the SLD. See also: Remote Client [page 602].

> **i Note**
>
> SLD registration is not mandatory for using the *Remote Client*. In the *Remote Client*, you can also manually add the PCo systems that you want to monitor by entering the host name. In this case, SLD registration is not necessary.

Procedure

1. In the *PCo Management Console*, choose ▶ *Tools* ▶ *SLD Registration* ▶ from the menu.

   The *System Landscape Directory* dialog box appears.
2. Enter the following data:

Field	Description
System Landscape URL	Here you enter the URL of the SLD system in which you want to register your PCo system. The URL is structured as follows: `http://<servername>:<port>/SLD`
User Name	User name for the SLD system
Password	Password for the SLD system
PCo System ID	The system ID that you specified previously under ▶ *Tools* ▶ *Global Properties* ▶ *System ID* ▶ is displayed here. See also: Specifying the System ID [page 13]
PCo Release	Indicates the PCo release that is installed. **Example** `Plant Connectivity 2.202.1066.827.` This means: Plant Connectivity 2.2 (SP02).
PCo Description	Description of the PCo instance; the PCo instance and the server on which the PCo instance is running are given as default values. The description can be changed.
Enable Tracing	Select the checkbox.

3. Choose the *Register* pushbutton to register your PCo instance in the SLD system.

PCo issues a success message.

4. Choose the *Remove* pushbutton if you want to remove the registration from the SLD system.

   PCo issues a relevant success message.

Results

The PCo system is now in the *System Landscape Directory (SLD)*. In the *Remote Client*, you can use the *Use SLD* function key to include the PCo system in the list of systems to be monitored. See Adding a PCo System via SLD [page 606].

2.3.3 Authorization Management

You can use authorization management to define specifically which PCo services you want external callers to be able to access.

Use

Plant Connectivity provides a number of services that can be used by remote computers:

- Management Services
  Using the management services, you can call PCo functions via Web services. You can configure, start, or stop agent instances and query configurations, status information, or protocols.
- Remote Client
  The remote client enables you to monitor PCo from a remote computer, to start or stop agent instances, to export and import configurations, and to query protocols.
- PCo Web Server
  The PCo Web server provides configurable methods in the form of Web service endpoints.

The services simply require that the user can log on to the Windows computer on which PCo is installed. If you want to control in a more restrictive way which PCo services can be accessed by external callers, you can use authorization management in PCo.

Procedure

You can call authorization management in the PCo Management Console under *Tools* > *Authorization Management* or by choosing the relevant icon in the taskbar.

The dialog provides you with a multilevel list of available PCo services. You can define the authorizations for the services at each level and thereby control access in a more detailed way as required.

PCo's authorization management uses **Windows user groups** to control access to PCo services. There are four options in the *Access Mode* column for configuring access to a specific service:

- *No Access*
  External callers cannot access this service.
- *Unrestricted Access*
  Each user who can log on to the Windows computer with the PCo installation has access to the service in question. This setting corresponds to the system behavior until now but is not recommended.
- *Access Depending on User Group*
  Only users who are members of the specified Windows user group can use the service in question. You maintain user groups and the assignment of users in *Computer Management* under *Local Users and Groups*. You can also use the *Active Directory* for this task.
- *Access Inherited from Superordinate Service*
  With this setting the access rights to a specific service are inherited from the superordinate level of the hierarchy. This is the standard setting for newly added agent instances or methods and you only need to change this setting if you want to maintain different access rights.

  > **i** Note
  >
  > This setting is not possible for the services at the topmost level.

In the last column of the dialog (*Effective Authorization*), PCo displays for each service how the settings affect how the service is executed.

Standard Settings

After installing PCo, only users that belong to the *Administrators* user group have access to the services provided by the management services. The PCo Web server can be called by users that belong to the *PCoWebServer* user group provided this user group existed at the time of installation. Otherwise, and for all other services, the default setting is *No Access* for the topmost level and *Access Inherited from Superordinate Service* for the lower levels. After installation, you can revert at any time to these standard settings by choosing the relevant pushbutton.

Notes for the Configuration

If you have configured the endpoint of a PCo Web server so that no authentication is required when services stored there are called, no authorization check can take place when the services are called. Each caller who knows the URL of the service can use the service. The settings in authorization management are not effective for a Web server configured in this way and this is shown accordingly in the *Effective Authorization* column.

If you configure authorizations for the remote client, you need to bear in mind that authorization configuration only has an effect on the functions of the remote client of the current computer. If you are managing remote computers using the remote client, the authorization settings of the remote computer are also effective.

If you change the **authorization settings** for the **management services**, you must restart the **main service after saving** the configuration, in order for these changes to take effect. You can call the management services from the menu in the Management Console under *Tools* > *Options* > *Global Settings* > *Main Service*. The

same applies to the authorization settings for the *Remote Client* and the *Web server*. To take account of the current authorization configuration, you have to exit these two applications and restart them.

2.3.4 Settings for Cloud Integration

Describes the settings for PCo cloud integration

PCo communicates with the **Cloud Connector** using the **cloud services** that are based on RESTful Web services. The cloud services are hosted by the PCo main service. You can specify here the parameters of the connection between the host for the cloud services and the Cloud Connector, in particular the security settings and the authorizations for calling the services provided. The WebSocket protocol, whose parameters you can also maintain here, is used for internal communication between the host and the service providers (agent instances).

To be able to use the cloud services, you need to activate them in the settings for the main service in the menu under *Tools* > *Options* > *Global Settings* > *Main Service*. If required, you can also change the port that the main service uses for the cloud services. (See also: Configuring the Main Service [page 14].)

> **i Note**
>
> If you make changes to the settings for the cloud services, the main service is automatically restarted when you exit the dialog after saving, so that the changed settings take effect.

Maintain the settings for cloud integration on the following tabs:

User Configuration Tab [page 26]

Server Security Settings Tab [page 29]

Internal WebSocket Communication Tab [page 32]

Internal Communication with the Main Service [page 33]

Related Information

Integration with the SAP Digital Manufacturing Cloud [page 50]

2.3.4.1 User Configuration Tab

Use

Here you can define and monitor the rights of the users and user groups that can access the cloud services. Predefined roles are assigned to the individual services. Roles are assigned to the individual user groups using

an application in the **Digital Manufacturing Cloud**. For locally defined users, you can assign the roles here yourself.

Maintain User Groups

The maintenance of user groups, the assignment of PCo roles, and the assignment of user groups to SAP Digital Manufacturing Cloud users **is carried out centrally using applications in the Digital Manufacturing Cloud**. For more information about maintaining user groups in the DMC, see the Machine Model documentation under https://help.sap.com/viewer/76070b83a9954174b76a3411ad31f034/latest/en-US/ffc87516de484cecb2daa918bab5c605.html.

When you are configuring the user groups in the SAP Digital Manufacturing Cloud, you can control whether you want to perform the **authorization checks** for all PCo installations connected to the Digital Manufacturing Cloud in the same way, or whether you want to perform this check more specifically for individual installations or users.

A special default user group is provided for the execution of background tasks. This user group is always used for the authorization check if the service call takes place without user information and user group information. (See also: Process Flow for the Authorization Check [page 29].)

To enable the distribution of user groups from the Digital Manufacturing Cloud, you must maintain at least one local user with the **Administrator** role in each PCo installation. This user runs the cloud services that are used to create and change the user groups.

The master data of user groups, which were created using Cloud services, is protected from changes made in the Management Console. In exceptional cases, you can remove the write protection by choosing *Unlock User Group* and then make the necessary changes.

Maintain Local Users

If you have already worked with the Digital Manufacturing Cloud and there are, therefore, already local users, the users maintained locally in PCo until now are still included with priority in the authorization checks when cloud services are called.

To create a new user, choose the *Add User* pushbutton below the list of users. The system proposes a **name** and an **ID** for the new user. You can change this proposed data on the right side of the screen.

Each user must have a unique name and a unique user ID. The user ID can be identical to the user ID of the Windows user. Select the roles that you want to assign to each user.

User Roles in PCo

You can assign the roles to the local users. Roles are assigned to the individual user groups using an application in the **Digital Manufacturing Cloud**.

Role Descriptions

Role	Description
`Administrator`	This role is for the administration of users and authorizations. The administrator can also trigger the generation and recovery of data backups of the configuration via the Digital Manufacturing Cloud. → Recommendation SAP recommends that users with this role shouldn't be assigned any additional business roles. Users with the **Administrator** role can **only be created locally in PCo**.
`CertificateAdministrator`	A user or user group with this role is allowed to manage certificates. In particular, he or she can establish the trust relationship with communication partners.
`PCoConfigurator`	A user or a user group with this role can create and change PCo configuration elements, in particular service providers and their dependent configuration elements such as source and destination systems.
`ServiceExecutor`	This role is required to execute machine methods of the Machine Model.
`DataReader`	This role is required to read data from a source system using a query.
`DataStorer`	A user or a user group with this role can use a store query to write data back to a source system.
`FileProcessor`	A user or a user group with this role can read and write files. This allows files to be written via the Digital Manufacturing Cloud for handover to a print server to a folder that you have configured for this purpose. For more information about setting up the print function in the DMC, see https://help.sap.com/viewer/97c9e9b9fac74be2a023638cd1700b46/latest/en-US/800486f1fd8e4d97bd062c4694c30772.html.
`Operator`	A user or a user group with this role can start and stop service providers and query the runtime status of service providers (agent instances).
`BackupCreator`	A user or user group with this role can trigger the generation of data backups of the configuration in the Digital Manufacturing Cloud.

Related Information

Process Flow for the Authorization Check [page 29]
Transition to Central User Group Maintenance [page 29]

2.3.4.1.1 Process Flow for the Authorization Check

At the runtime of the main service, which hosts the cloud services, when a cloud service is called, the **Digital Manufacturing Cloud either transfers the user information directly, or transfers a JSON web token (JWT)** that contains information about the identity of the user and the user groups assigned to it. During the authorization check, the PCo system first determines the PCo roles that are assigned to the transferred user, and checks whether they correspond to the roles required by the service call. If no local PCo user is maintained, this check is executed for the roles that are assigned to the transferred user groups.

> **i** Note
>
> For background tasks, the JSON web token does not contain user information or user group information. In this case, the default user group provided for the execution of background tasks is used for the authorization check.

2.3.4.1.2 Transition to Central User Group Maintenance

Since the system treats locally created users with priority during the authorization check for service calls, an uninterrupted transition from decentralized users created locally in PCo, to the user groups created centrally in the Digital Manufacturing Cloud, is readily possible. SAP recommends that you do the following:

1. You first continue to work with the local users in PCo.
2. You define the user groups in the **Digital Manufacturing Cloud** and distribute them to the relevant PCo installations.
3. As soon as the required user groups are available in PCo, you can delete the local users in the PCo installation. You should still define at least one local user with the **Administrator** role so that the user groups can still be maintained.

2.3.4.2 Server Security Settings Tab

Here you define the security settings that apply for communication between the host for the cloud services and the **Cloud Connector**. The main service, which hosts the cloud services, plays the role of the server, and the Cloud Connector plays the role of a client.

Field	Description
Session Timeout	Time interval in minutes after which an inactive session is terminated automatically.
Authentication Mode	In the *basic* authentication mode, the service user is registered using the user name and password. In the authentication mode *JSON Web Tokens and Principal Propagation*, authentication takes place using temporary certificates provided by the Cloud Connector. Alternatively, the Cloud Connector can forward JSON web tokens (JWT). These contain information about the user through which the service call takes place and about the user groups assigned to the user. The use of JSON web tokens is the prerequisite for central authorization management via user groups in the Digital Manufacturing Cloud. In the *certificate authentication mode*, you define that the host for the cloud services allows certificates for authentication. In other words, the client connecting with the host can use a certificate to authenticate himself or herself. This authentication mode is used internally when the agent instance has called, for example, execution in the **Digital Manufacturing Cloud** and is to write the results of the call back to the machine after that. In this case, the agent instance calls the store service of the host for the cloud services to write the data back to the server of the machine.
Displayed Realm	This setting is only relevant in the basic authentication mode and specifies the domain (“realm") that is displayed for the relevant PCo system in the Cloud Connector. This allows a semantic distinction to be made between multiple PCo systems.
Server Certificate	The server certificate of the type `X.509-v3` enables secure communication between PCo and the Cloud Connector. It should be issued for the host name of the computer on which PCo is running and can be self-signed or be embedded in a hierarchy of certificates. When a PCo system is created, the SAP Digital Manufacturing Cloud requests the public key of the server certificate specified here. PCo forwards the public key to the DMC. The DMC needs this public key so that it can transfer the passwords, which you have specified for specific configuration elements in the Machine Model, in encrypted form to PCo. Only the PCo system that has the appropriate private key can decrypt the passwords.

In the screen area for *client certificates*, you specify how the certificates (which are exchanged during **communication between PCo as a server and the Cloud Connector as a client**, or between **PCo as a WebSocket server and the service providers as clients**) are processed.

The client certificate is handled according to the same principles as for communication between an OPC UA server and a UA client:

If the host for the cloud services wants to set up a secure connection to a Cloud Connector as a client, it gets a certificate with a public key from the client. PCo accepts this certificate if it regards it as trustworthy. Otherwise, the certificate is stored in a directory for rejected client certificates. For the cloud services, a directory `certs` located under `%ProgramData%\SAP\PCo\ CertificateStores\CloudServicesHost \Rejected\` is used for this purpose.

If the client is using a self-signed certificate, you can, after an unsuccessful connection attempt, copy the certificate from this store location to the directory for trusted certificates. This establishes the trust relationship between the server and client on the PCo side. This is the `Trusted` directory in the directory tree named above `%ProgramData%\SAP\PCo\ CertificateStores\CloudServicesHost`.

If the application certificate of the client is embedded in a certificate hierarchy, the related root certificate needs to be available to establish a trust relationship to the server. This root certificate must be in the subfolder `certs` of the directory `TrustedIssuer` in the above path.

Field	Description
Store Type	The store type cannot currently be changed because the **file system** is predefined as store type. The certificates used when the connection to the client is set up are stored in fixed storage folders in the file system. You can find these folders as subfolders under `%ProgramData%\SAP\PCo\ CertificateStores\CloudServicesHost`. The pushbutton to the right of the selection opens the default directory `Rejected`.
Revocation Check	You specify here how you want PCo as a server to perform the check for revoked certificates.
Revocation Check Scope	Controls which parts of a certificate chain are involved in the revocation check.
Accept Invalid Host Name	If you choose this option, PCo accepts an invalid host name in the client certificate.
Accept Expired Validity Period	If you choose this option, PCo also accepts expired client certificates.

2.3.4.3 Internal WebSocket Communication Tab

The host for the cloud services communicates with the agent instance wrapped as a service provider through the WebSocket protocol. The host is configured as a server and the individual agent instances are configured as clients. If necessary, you can define parameters that differ from the default values for this communication.

Field	Description
Internal WebSocket Port	You specify a free port number between 1025 and 65535. In the standard system, the same port number is used here as for the cloud services. **i Note** You can maintain the port number for the cloud services in the settings for the main service in the menu under *Tools* > *Options* > *Global Settings* > *Main Service*
Timeout	You use this setting to specify when a timeout should occur during internal communication between the main service, which hosts the cloud services, and the agent instances. It may be necessary to increase this value if processing of notifications triggered by a service call takes longer, for example, because a complex sequence is to be processed in a multiple call destination system.
Authentication Mode	Use the authentication mode *None*, if you are sure that on the PCo computer no WebSocket connections are possible from outside. Otherwise, select the authentication mode *certificate* and create a client certificate with which the individual agent instances can authenticate themselves as clients at the server (in this case, the host for the cloud services).
Client Certificate for Authentication	A client certificate is required if you have selected the authentication mode *certificate*.
Host Name	Host name that is to be used for internal WebSocket communication. It might be necessary, especially if you are using client certificates, to specify the host name given in the certificate.

2.3.4.4 Internal Communication with the Main Service

In some scenarios, cloud services are called within PCo. This is the case, for example, in the notification scenario, if the response from a Web service called in the SAP Digital Manufacturing Cloud is to be written back to the data source. The query Web service, which is part of the cloud services, is then called within PCo.

Here you can enter the **certificate** for PCo-internal communication with the host for the cloud services that is to be used as a client certificate when you authenticate yourself to the host.

The certificate can be self-signed and needs to be issued for the user whose ID is being used to call the service.

> **i** Note
>
> The necessary roles need to be assigned to this user so that the relevant cloud service can be called. In the notification scenario described above, the user needs, for example, the role `DataStorer` to be able to call the query Web service.

3 Processes and Integration Scenarios with SAP PCo

PCo supports the following processes and integration scenarios:

- Notification Process [page 35]
  The notification process enables you to monitor production facilities and record any sudden, undesired events and report them to a destination system. However, you can also use this process to record regularly occurring, desired events such as confirmations.
- Query Process (with SAP MII) [page 43]
  You use the query process if you want to query tags from a query-capable source system, starting from an SAP MII system, for example.
- Integration with the SAP Digital Manufacturing Cloud [page 50]
  You use cloud integration if you want to connect instances of PCo with business applications of the **SAP Digital Manufacturing Cloud**. This enables information to be exchanged between, on the one hand, systems installed on the shop floor or Internet-enabled devices, and on the other hand, **SAP Digital Manufacturing Cloud** applications.
- PCo as OPC UA Server [page 54]
  You can configure agent instances of Plant Connectivity as OPC UA servers. If it is started as an OPC UA server, the agent instance can receive and process method calls from OPC UA clients.
- PCo Web Server [page 464]
  PCo agent instances can be run as Web servers or as OPC UA servers. The Web server provides methods in the form of service operations that can be executed using Web service calls from Web clients. All current Web service standards such as SOAP, OData, and REST are supported.
- Integration with Third-Party Systems Using Web Services (WS Destination System) [page 66]
  You can use PCo directly to transfer data from production to a third-party system, such as an SAP ME system. To do this, you can use the notification process. You can subscribe to specific tags of the data source. When specific events occur, PCo sends notification messages to the destination system by means of a Web service call.
- Integration with Third-Party Systems Using Enhanced Notification Processing (ENP) [page 68]
  Enhanced notification processing (ENP) enables you to flexibly control and document the data flow in production in connection with various destination systems, for example, with Web services. In this way, you can connect a third-party system (such as SAP ME) to PCo and transfer data from machine level to the desired SAP ME activity using Web service calls.
- Connection of Web Applications via WebSocket [page 87]
  The WebSocket network protocol, in accordance with specification RFC6455, allows a high-performance, network-resource-efficient communication between a WebSocket server and a Web application configured as a WebSocket client.
- Connection of External Data Sources to Bus. Suite Applications [page 87]
  SAP Plant Connectivity (PCo) easily facilitates the communication between SAP Business Suite applications and external data sources such as weighing systems, production machines, OPC servers, or process control systems.
- Integration with SAP ODA [page 124]
  If you are using SAP ODA or intend to do so in the future, you can use the Plant Connectivity component to establish the connection to an OPC server instead of using the SAP ODA Connector.

3.1 Notification Process

Use

The notification process enables you to monitor production facilities and record any sudden, undesired events and report them to a destination system. However, you can also use this process to record regularly occurring, desired events such as confirmations.

With this process, you can monitor tags from the source system (from a control system for a production plant, for example) for possible changes. You can specify the trigger conditions in detail that must be satisfied for PCo to send a notification message to one or more destination systems.

You can specify the following as trigger conditions:

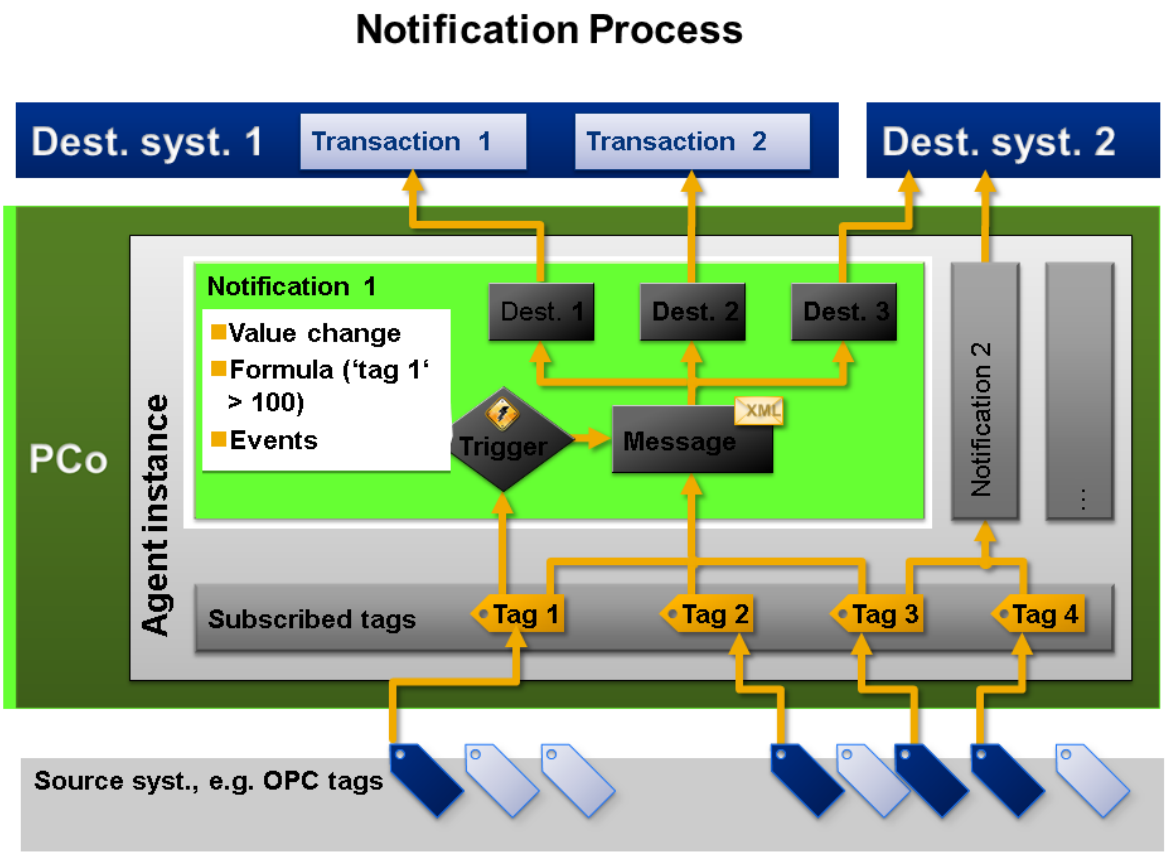
- Changes in values
- Formulas
- Events (such as production standstill)

You can use the following source systems for the notification process:

- All OPC source systems
- MQTT source system
- PI source system
- Proficy Historian source system
- Socket source system
- Asset Framework source system
- File monitor source system
- File system source system
- Modbus source system
- ODBC source system
- OLE DB source system

> **i** Note
>
> For all types of source systems except OPC DA source systems, PCo continually transmits the tags and checks whether the specified conditions are met. If so, PCo sends a notification message to the destination system or systems.
>
> If the source system is an OPC DA system, the source system checks the measurement values itself and sends only the deviating tag to PCo. PCo then sends the notification message to the destination system or systems.

Process



Notification Process

1. The PCo system continually receives measurement values from the source system and checks whether the condition that you defined in the *Trigger* screen area is met.
   For an OPC DA source system, the source system checks whether the condition is met and only sends a *tag* to PCo if the trigger condition is met.
2. If the trigger condition is met, PCo creates a notification message and sends it to one or more destination systems.
3. If the destination system is an SAP MII system, for example, SAP MII processes the notification message.

Example

You are monitoring a boiler with PCo. The boiler control system (source system) is an OPC DA system. You have specified that the boiler temperature *tag* is to be sent to PCo if the measured value falls short of or exceeds the permissible temperature bandwidth.

You have selected the trigger type *OnTrue* in the *Trigger* screen area.

You have specified the formula `TAG_TEMP<57 || TAG_TEMP>63` as the trigger condition.

If the measured value lies below 57°C (134.6°F) or above 63°C (145.4°F), PCo creates a notification message and sends it to the destination system (an SAP MII system, for example). The SAP MII system can then send an e-mail to a previously defined recipient or create a maintenance notification in ERP, for example.

See Also

Settings for the Notification Process with SAP MII [page 37]

3.1.1 Settings for the Notification Process with SAP MII

The following description guides you through the configuration of a typical notification process. It is assumed that when certain tag values of a data server are changed, you want to trigger a transaction in SAP MII to send an e-mail, for example.

To use the notification process with SAP MII, you must create a transaction in SAP MII that you call from Plant Connectivity using a Web service. The Web service is configured as the destination system of a notification.

> **i** Note
>
> Until now, the MII destination system was used to call an MII transaction. However, this destination system is deprecated. SAP therefore recommends that you configure a **universal Web service destination system** for calling MII transactions.

Settings in SAP MII

- Create a new transaction in **SAP MII**.
  For more information, see *SAP Help Portal* under *SAP Manufacturing Integration and Intelligence 15.4* > *SAP Manufacturing Integration and Intelligence* > *Content Development* > *SAP MII Workbench* > *Transactions* > *Creating Transactions*.
- Define the transaction parameters using the properties of the transaction. At runtime, Plant Connectivity supplies these with data from the output expressions of the notification.
  For more information, see the SAP MII documentation under *SAP Manufacturing Integration and Intelligence* > *Content Development* > *SAP MII Workbench* > *Transactions* > *Properties for Transactions*.
- Specify how you want the transaction parameters to be further used in SAP MII. For example, you can configure that an e-mail is sent when a notification message is received. For more information, see the SAP MII documentation under *Send Mail*.

Settings in PCo

- Create the **source system** that is to be used to monitor the changes to the tag values that you want, for example, an OPC UA system. Maintain the entries required for the source system and check the settings on the *Reliable Connection* tab. See also: Source System: Reliable Connection Tab [page 142].
- Create the **universal Web service destination system** with which you want to call the transaction in MII. This destination system is sufficient if you want to implement a **synchronous call of the transaction in SAP MII** without error handling. See also: Calling MII Transactions Using Web Services [page 38].

If you want to use additional functions or an asynchronous transaction call, you must include the universal Web service destination system in one of the following destination system types:

- Multiple call destination system
  You can use this destination system type **in addition** to the previously created universal Web service destination system if you want to perform targeted **error handling** for the transaction call or if you want to execute several other work steps in the context of the transaction call. You achieve this by embedding the universal Web service destination system created previously (using the MII transaction) in a multiple call destination system.
- OPC UA destination system
  You can use this destination system type **in addition** to the universal Web service destination system created previously if you want **asynchronous transaction processing**. To do this, you have to configure the Web service destination system call as the destination of an OPC UA destination system call. See also: Configuring an Asynchronous Transaction Call [page 42].

- Create the *agent instance* for the agent you specified for the source system, for example, *OPC UA*.
- On the *Subscription Items* tab, for the agent instance, select the tags that are relevant to you. You can use the browse function to search for and select source tags.
- Add a *notification* to the *agent instance*. See also: Notification [page 496].
- On the *Output* tab, for the notification, generate an expression for each subscription item. See also: Output Tab [page 512].
  You can use the expression editor to perform additional calculations for the required subscription items. In addition, you can create expressions without reference to subscription items and, for example, enter a message text. See also: Formulating and Calculating Expressions [page 547].
- On the *Destinations* tab, define one or more destinations for the notification to which you want the message to be sent. You can now choose one of the destination systems described above to call the MII transaction. See also: Destinations Tab [page 520].
- In the *Assignment of Notification Output* view, on the *Destinations* tab, maintain the assignments between the output expressions of the notification and the input parameters of the destination system calls. See also: Assignment of Notification Output (Call of MII Transactions) [page 523].

3.1.1.1 Calling MII Transactions Using Web Services

Every MII transaction can be called using Web services.

In Plant Connectivity, you create a *universal Web service destination system* to call the MII transaction. You can use either a **SOAP-based Web service** or a **REST-based Web service** for the call. The SOAP-based service can be configured more easily, but has minor performance disadvantages. The REST-based service also allows the parameterization of the MII transaction with regard to asynchronous execution and persistence behavior.

The following assumptions are made for the following descriptions:

- `<server name>` is the server name of your MII system
- `<port number>` is the port number used to call the system
- `<folder>` is the folder in which the transaction is stored in the catalog view in MII
- `<transaction>` is the name of the MII transaction that is to be called
- `<parameter1>, <parameter2>` are input parameters of the transaction

Related Information

SOAP-Based Web Services [page 39]
REST-Based Web Services [page 41]
Assignment of Web Service Call as Destination of a Notification [page 42]
Configuring an Asynchronous Transaction Call [page 42]

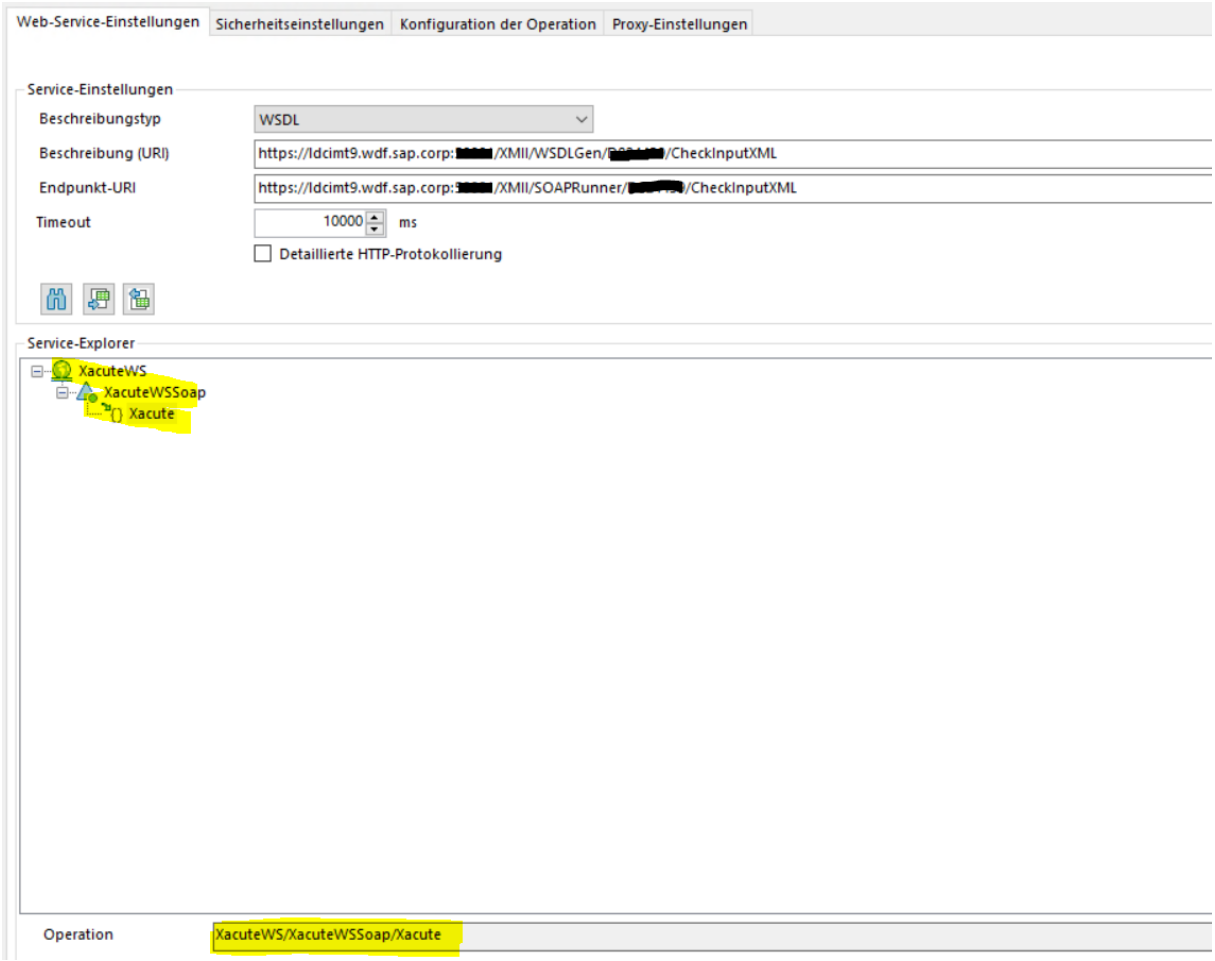
3.1.1.1.1 SOAP-Based Web Services

To be able to call the MII transaction using SOAP-based services, proceed as follows:

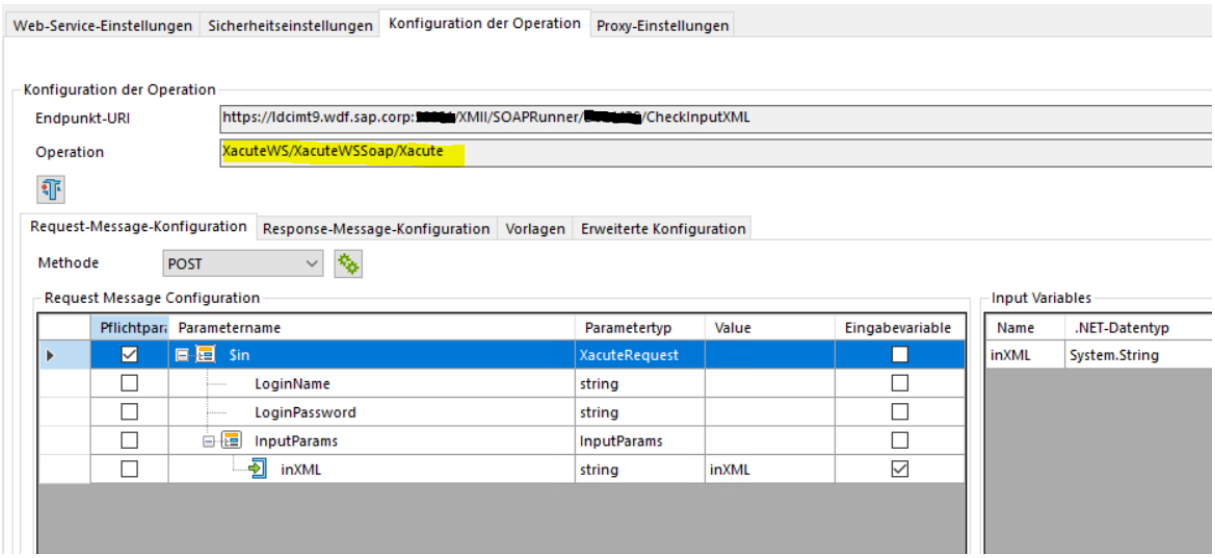
1. When creating the **universal Web service destination system**, select the description type **WSDL** on the *Web Service Settings* tab.
2. In the *Description URI* field, enter a URI with the following schema:
   `https://<server name>:<port number>/XMII/WSDLGen/<folder>/<transaction>`
3. Open the URI in the *service explorer* and select operation `XacuteWS/XacuteWSSoap/Xacute`.
4. On the *Security Settings* tab, select the appropriate authentication parameters.
5. On the *Operation Configuration* tab, maintain the *request and response message configuration* and assign the variables or fixed values to the service parameters.
   The system proposes the parameters in accordance with the service description when you expand the collapsed rows of the parameter list.

Example

The entries on the *Web Service Settings* tab:



The entries on the *Operation Configuration (request message)* tab:



3.1.1.1.2 REST-Based Web Services

To be able to call the MII transaction using REST-based services, proceed as follows:

1. When creating the **universal Web service destination system**, select the description type **No Service Description**.
2. In the *endpoint URI* field, enter the URI as follows:
   `https://<server name>:<port number>/XMII/Runner`
3. On the *Security Settings* tab, select the appropriate authentication parameters.
4. On the *Operation Configuration* tab, change the method to GET and switch to the tab *Templates* > *Request*.
   You enter the template in the *Request* screen area according to the following schema:
   `IsAsync={inIsAsync}&Persistence={inPersistence}&Transaction={inTransaction}&{inParameter1}={inValue1}&{inParameter2}={inValue2}&Session=false&content-type=text/xml`
   The values in the curly brackets are automatically interpreted by the system as input variables and appear in the *variables list* on the *Request Message Configuration* tab. They allow the transaction to be parameterized at runtime. They have the following meaning:

   Input Variables

Input Variable	Meaning
`inIsAsync`	Can have the values **True** or **False** and controls whether the MII transaction is to be processed asynchronously.
`inPersistence`	Can have the values **NEVER**, **ALWAYS**, and **ONERROR** and controls whether the MII transaction is never to be stored, always to be stored, or only to be stored when there are errors.
`inTransaction`	Is the name of the transaction in the format `<folder>/<transaction>`.
`inParameter1`	Denotes the name of the first transaction parameter.
`inValue1`	Denotes the value of the first transaction parameter. This assignment applies accordingly to further parameters.

> **i Note**
>
> If you want to define the properties of the transaction call in the definition of the Web service destination system, you can directly store the corresponding fixed value in the template instead of the input variables in the curly brackets. For example, `Persistence=NEVER` instead of `Persistence={inPersistence}`.

3.1.1.1.3 Configuring an Asynchronous Transaction Call

If you want the transaction call to take place asynchronously, call the Web service destination system as the destination of an asynchronous OPC UA method call:

1. Create an agent instance without a source system and configure an OPC UA server. (See also: OPC UA Server [page 455].)
2. Create a method and select the *asynchronous* checkbox for this method. (See also: Server Method Definitions Tab [page 461].)
3. Create a method notification for your method.
4. Define the previously created universal Web service destination system or a multiple call destination system as the destination of the method call. (See also: Destination Tab [page 542].)
5. Create an OPC UA source system that connects to the local UA server and then an OPC UA destination system that contains the call of the previously created method. (See also: OPC UA Destination System [page 333].)
6. You can now use this OPC UA destination system as the destination of a tag-based notification to call the MII transaction asynchronously. (See also: Destinations Tab [page 520].)

3.1.1.1.4 Assignment of Web Service Call as Destination of a Notification

Using the Web Service Destination System

You can now use the configured universal Web service destination system as the destination of a notification. You have the following options:

- You call the **universal Web service destination system** directly if you want to call the MII transaction synchronously.
- If you embed the Web service call with the MII transaction in a **multiple call destination system**, you can make error handling more flexible and allow the transaction to be called in the context of other process steps.
- By calling an **asynchronous OPC UA method**, you can configure the call of the MII transaction asynchronously. See: Configuring an Asynchronous Transaction Call [page 42].

Appropriate output expressions are now assigned to the variables of the input parameters of the destination system. At runtime, the parameters of the MII transaction are then supplied with the values of the output expressions.

Compatibility with Previous MII Transaction Calls

If you have already developed transactions in SAP MII that you previously called using the MII destination system, you can still call this transaction using PCo, without having to make changes to the transaction in MII.

The XML document sent from the MII destination system then supports inbound processing of the MII transaction. Parsing is performed to extract the transaction parameters from the XML message.

For the transaction call using the universal Web service destination system, you only have to maintain a single input variable in which the XML document is to be received. When assigning the notification output to the inbound parameters of the Web service destination system, you can select the special output value *~NotificationOutputInXMLFormat*. At runtime, it contains the output of the notification in XML format, just as it is also generated by the MII destination system. Assign this expression to the only inbound parameter of the Web service destination system.

The special output expression *~NotificationOutputInXMLFormat* is also available when you configure multiple call destination systems and OPC UA destination systems.

Related Information

Assignment of Notification Output (Call of MII Transactions) [page 523]

3.2 Query Process (with SAP MII)

Use

You use the query process if you want to query tags from a query-capable source system, starting from an **SAP MII system**, for example. With a *tag query* or a *PCo query* in SAP MII, you can read out the tags of your choice via PCo and, if necessary, also write values to SAP MII. You can use all the functions in SAP MII to represent or forward the data determined via query:

- Dashboard
- Alerts
- Forwarding and further processing in SAP ERP, such as creating a maintenance notification in SAP ERP

Prerequisites

Plant Connectivity Console

- You have created a *source system*.

> **i** Note
>
> All agents delivered with PCo are query-capable.

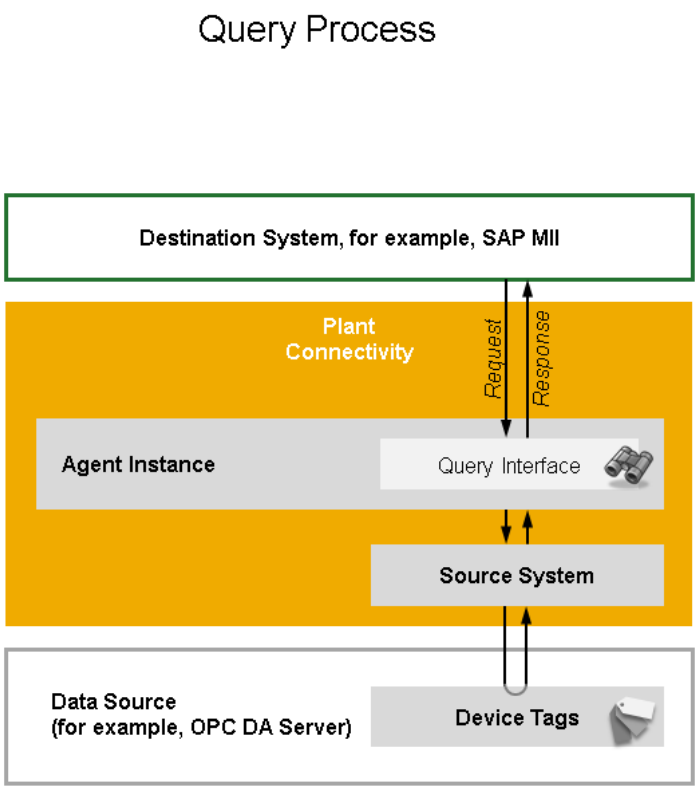
- If necessary, you have defined an alias for the source system. See also: Source System: Aliases Tab [page 138].
- You have created an *agent instance*.
- On the *Servers* tab, you have selected the *SAP MII Query Server* or *SAP MII Query Server (before 12.2)* checkbox and made the settings for the SAP MII system that is to be connected. See also: Servers Tab [page 419].
- You have made the settings on the *Tag Query* tab for the agent instance. See also: Tag Query Tab [page 475].

SAP MII

You have set up the connection data for PCo in the SAP MII system:

- You have created a data server in SAP MII. For more information, see the *SAP Manufacturing Integration and Intelligence* documentation under **Data Servers**.
  - For the data server, you use the *connector type* `UDC` if you are using an SAP MII system up to and including MII 12.1. (See also: *SAP Manufacturing Integration and Intelligence* documentation under **Universal Data Connector**.)
  - For the data server, you use the *connector type* `PCoConnector` if you are using an SAP MII system as of MII 12.2. (See also: *SAP Manufacturing Integration and Intelligence* documentation under **PCo Connector**.)
- You have created a *query* in SAP MII. In the query, you define which tags are to be queried in the source system.
  - You use a **tag query** if you are using an SAP MII system **up to and including MII 12.1**. (See also: *SAP Manufacturing Integration and Intelligence* documentation under **Tag Query**.)
  - You use a **PCo query** if you are using an SAP MII system **as of MII 12.2**. (See also: *SAP Manufacturing Integration and Intelligence* documentation under **PCo Query**.)

Process



1. SAP MII starts a query, for example, to supply a specific dashboard with current data by sending a request to the agent instance.
2. PCo uses a query to query the selected tags in the source system.
3. PCo returns the query results to SAP MII in the form of a response where they are processed further.

Example

Example: Settings for a Query Process with SAP MII 12.1 [page 46]

Example: Settings for a Query Process with SAP MII 14.0 [page 48]

3.2.1 Example: Settings for a Query Process with SAP MII 12.1

Use

The example shows you how to proceed if you want to be able to query tags from an OSISoft PI source system starting from SAP MII.

> **i Note**
>
> In this context, *OSISoft PI* is to be regarded as a data source merely representing every other query-capable data source (for example, an OPC UA server) and is not to be the focus of this example.

The following information, in connection with PCo, refers to:

- *OSISoft PI* (source system)
- *SAP MII 12.1* (destination system)

Prerequisites

- In the PCo Console under ▶ *Tools* > *Options* > *Global Settings* > *Main Service* ◀ you have configured the host name and made additional settings. (See also: Configuring the Main Service [page 14].)

Activities

Settings in PCo

1. Create a source system of the type *PI Agent*. See also: OSISoft PI Source System [page 234]
2. Create an agent instance and choose the *Servers* tab.
3. On the *Servers* tab, select the *SAP MII Query Server (Before 12.2)* checkbox and enter the port, for example, `9001`.
4. Start the agent instance.

Settings in SAP MII

Creating a Data Server

1. In the MII menu, choose ▶ *Data Services* > *Data Servers* ◀.
   The MII system shows a list with all available data servers. For more information, see the *SAP Manufacturing Integration and Intelligence* documentation under **Data Servers**.
2. Choose the *Create* pushbutton.
   The *New SAP MII Data Server* screen appears.
3. Enter the following data:
   - *Server Name*
   - *Connector Type* `UDC`
4. Choose the *Finish* pushbutton.

The data server is created and the *Settings* and *Connection* tabs are displayed.

5. On the *Settings* tab, select the *Enabled* checkbox.
6. Enter the following on the *Connection* tab:

Field	Description
IP	Here you enter the name of the server on which PCo is running. **i Note** You can find the server name in the *PCo Console* under *Tools* > *Options* > *Management Host* > *Settings*.
Port	Specify the port. **i Note** You can copy the port from the *Servers* tab of the agent instance.
Writable	If you select this checkbox, you can write values into the tags of the source systems, starting from MII.

7. Save your entries.
8. Start the agent instance in the *PCo Console*. This establishes the connection between MII and PCo. The status of the data server on the *Status* tab is then green.

Creating a Tag Query

1. Starting from the MII menu, choose *Content Development* > *Workbench*.
   The catalog overview is displayed.
2. Click on the project you created previously.
3. Open the context menu using the right mouse button.
4. In the context menu, choose *New* > *Tag Query*. For more information, see the SAP MII documentation under **Tag Query**.
   The *Data Source* screen appears.
5. Choose the data server you created previously.
6. In the *Mode* field, choose, for example: `Current` (read current value of the tag).
7. In the *Template Categories* screen area, double-click the *Tag Query Details* entry.
8. Choose the desired tag on the *Tag Query Details* screen. (See also: SAP MII documentation under **Tag Query: Details and Values**.)

3.2.2 Example: Settings for a Query Process with SAP MII 14.0

Example Description

The example shows how to proceed if you want to be able to query tags from an OSISoft PI source system starting from SAP MII.

> **i** Note
>
> In this context, *OSISoft PI* is to be regarded as a data source merely representing every other query-capable data source (for example, an OPC UA server) and is not to be the focus of this example.

The following information, in connection with PCo, refers to:

- *OSISoft PI* (source system)
- *SAP MII 14.0* (destination system)

Prerequisites

In the *PCo Management Console* under ▶ *Tools* > *Options* > *Global Settings* > *Main Service* ◀ you have configured the host name and made additional settings. (See also: Configuring the Main Service [page 14].)

Procedure

Settings in PCo

1. Create a source system of the type *PI Agent*. (See also: OSISoft PI Source System [page 234].)
2. Create an agent instance and choose the *Servers* tab.
3. On the *Servers* tab, select the *SAP MII Query Server* checkbox and enter at least the port, for example, **9001**. (See also: SAP MII Query Server [page 421] and Setting Up a Secure Connection for MII Queries [page 422].)

Settings in SAP MII 14.0

Creating a Data Server

1. In the MII menu, choose ▶ *Data Services* > *Data Servers* ◀.
   The MII system shows a list with all available data servers. For more information, see the SAP MII documentation under **Data Servers**.
2. Choose the *Create* pushbutton.
   The *New SAP MII Data Server* dialog box appears.

3. In step 1 *Connector* enter the following data:

Field	Description
Server Name	Enter a name of your choice for the server on which PCo is running.
Connector Type	Choose the `PCoConnector` entry.

4. In step 2 *Connectivity*, enter the URL for the PCo instance manually: `http://<servername>:<port>/PCoManagement`.
   The MII system displays a dialog box for logging on to the computer on which PCo is running.
5. Enter *User Name* and *Password* for the computer on which PCo is running.
   Step 3 *Component* appears. The MII system displays the agent instances created in PCo.
6. Choose the agent instance that you created previously for your PI source system.
7. Choose the *Finish* pushbutton.
   The new data server is created. The name of the data server and the type *PCoConnector* are displayed in the upper screen area. The *Settings* and *Connection* tabs are displayed in the lower screen area.
8. On the *Settings* tab, select the *Enabled* checkbox.
   The data server that you created for PCo is now activated.
9. You need to check or specify the following data on the *Connections* tab:

Field	Description
Agent Name	The name of the selected PCo agent instance is displayed here.
Agent Port	The port of the PCo agent instance is displayed here.
Agent IP	Here you need to enter the IP address of the computer on which PCo is running.
Writable	Select the checkbox if, using the query, you want to change values of specific tags in the PI source system starting from SAP MII.

For more information, see the SAP MII documentation under **PCo Connector**.

Creating a PCo Query

1. Starting from the MII menu, choose *Content Development* > *Workbench*.
   The catalog overview is displayed.
2. If you have not yet created your own project, select the root of the catalog tree and choose the option *New Project* either using the right mouse button or from the menu. Enter a project name.
3. Select the project you created previously. Open the context menu using the right mouse button.
4. In the context menu, choose *New* > *PCo Query*. For more information, see the SAP MII documentation under **PCo Query**.
   The screen for selecting the available data servers appears.

5. Select the data server you created previously and then choose the entry `TagRetrieveQuery` in the *Mode* field. (See also: SAP MII documentation under **Data Source for Queries**.)
6. In the *Template Categories* screen area, double-click the *Tag Retrieve Query* entry.
   The *Hierarchy* tab is displayed. All tags of the PI source system are displayed.
7. On the *Hierarchy* tab, select one or more tags that you want to query using the query. (See also: SAP MII documentation under **Tag Retrieve, Aggregate, and Store Queries**.)

Testing a PCo Query

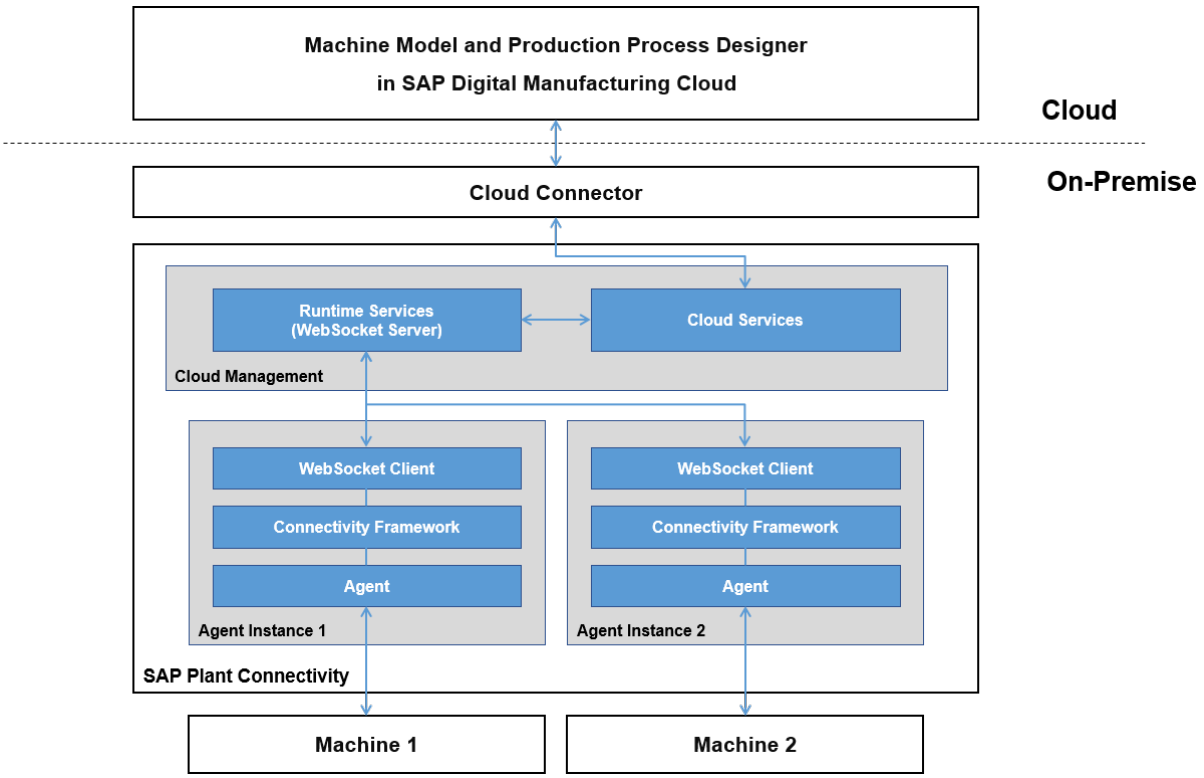
1. To test the newly created query, open the *project* in the workbench.
2. Select the query and choose *Test* in the context menu.
   The MII system displays the *Test Query Template* dialog box.
3. Select *Inner Frame* and choose the *OK* pushbutton to start the query.
   The MII system displays the query results. The selected tags are displayed with the current tag values.

3.3 Integration with the SAP Digital Manufacturing Cloud

Integration with SAP DMC

You can connect instances of Plant Connectivity with the **SAP Digital Manufacturing Cloud (DMC)** applications to enable information exchange between, on the one hand, production related, locally installed systems or Internet-enabled devices, and on the other hand, business applications in the cloud. Communication from the DMC to PCo is through the **Cloud Connector** to satisfy the security requirements for Internet communication. Communication from PCo to the cloud is performed directly, that is, without the Cloud Connector.

The connection of local systems starting from PCo takes place using agent instances that are wrapped as service providers. Service providers are the endpoints of machines in the Machine Model and model the tag information and the services that the machine offers.



In PCo, the connection to the **Cloud Connector** functions using the Windows service **main service** that is started automatically if you have chosen *cloud integration* on installation. The main service hosts the cloud services that can be called for the applications from the SAP Digital Manufacturing Cloud, and with which the service providers can be accessed at machine level.

You can set the parameters of the cloud services in the settings for cloud integration. You can call up these settings in the Management Console by choosing *Tools* > *Cloud Integration* from the menu. (See also: Settings for Cloud Integration [page 26].) You can configure the parameters for the main service in the system settings of PCo, which you can reach from the menu under *Tools* > *Options* > *Global Settings* > *Main Service*. (See also: Configuring the Main Service [page 14].)

Communication between PCo and the Cloud Connector and the **SAP Digital Manufacturing Cloud** takes place using RESTful Web services. Internal communication between the host for the cloud services and the agent instances that function as service providers takes place using the WebSocket protocol for which the agent instances function as clients and the host as server.

The configuration elements that were created in PCo using **SAP Digital Manufacturing Cloud** applications are protected in the Management Console against manual changes. If required, you can enable processing of locked configuration elements in the Management Console by choosing *Edit* > *Unprotect DMC Configuration Element* in the menu. Restore consistency with the cloud application as soon as possible.

Use

You can connect PCo in a first step to the **Machine Model** application of the **SAP Digital Manufacturing Cloud**. The following scenarios are supported:

- Reading and writing tags of machines
- Performing services or methods of machines

Prerequisites

- You have installed SAP Plant Connectivity and selected the **cloud integration** component.
- You have configured the cloud services and started the main service (see Settings for Cloud Integration [page 26] and Configuring the Main Service [page 14]).
- You have installed the Cloud Connector locally on the same network as SAP Plant Connectivity. (See also the *Cloud Integration* section in the Application Operations Guide on the PCo product page https://help.sap.com/viewer/p/SAP_PLANT_CONNECTIVITY.)
- You have configured the Cloud Connector so that it can be used for the Machine Model application. (See also the *Cloud Configuration* section in the Application Operations Guide on the PCo product page https://help.sap.com/viewer/p/SAP_PLANT_CONNECTIVITY.)

More Information

For more information about the **Machine Model**, see the **SAP Digital Manufacturing Cloud** product page under https://help.sap.com/viewer/76070b83a9954174b76a3411ad31f034/latest/en-US.

For more information about shop floor integration, see https://help.sap.com/viewer/c86ca4026fae4cb3ba66ed751866175b/latest/en-US/b39c43da65b94106a01c9d01ec81f23c.html.

For more information about the Cloud Connector, see https://help.sap.com/viewer/product/CP_CONNECTIVITY/Cloud/en-US.

3.3.1 Mapping of Machine Model Entities to Configuration Elements in PCo

The **Machine Model** in the **SAP Digital Manufacturing Cloud** is used to manage information about machines and devices on the shop floor. The digital twin of a machine in the **Machine Model** contains all the information about the machine that is required to call up machine data or plan a machine orchestration.

In the Machine Model, you can model the following objects:

- Shop floor systems

A shop floor system is a server in production that provides tags or services. You can use the shop floor system to model a server as a data source and then connect it as a data source. For example, you can create an OPC UA server as a data source and define all properties of the OPC UA server.

Mapping of Entities (Shop Floor Systems)

Shop Floor System	PCo Configuration Elements
OPC UA server	OPC UA source system and the corresponding agent instance
OPC DA server	OPC DA source system and the corresponding agent instance
OPC HDA server	OPC HDA source system and the corresponding agent instance
IP21 server	IP21 source system and the corresponding agent instance
AF server	Asset Framework source system and corresponding agent instance
Proficy Historian server	Proficy Historian source system and corresponding agent instance

- Service provider
  With the help of a service provider, you can model objects, services, and tag information in the **Machine Model**. When you configure the service provider, you define the service provider type (internal or external) and the usage of the object in PCo.
  The corresponding configuration elements are then generated automatically in PCo. The following table shows which objects (that you configure in the **Machine Model**) correspond to which configuration elements in PCo.

  Mapping of Entities (Service Providers)

Object in Machine Model	Service Provider Type	Usage Type	PCo Configuration Elements
Service provider	External	File system data source	File system source system and corresponding agent instance
Service provider	External	Web server	Service provider is only created in the PCo database; display in the PCo Management Console is not possible
Service provider	Internal	OPC UA server	Service provider in the database and an agent instance for the OPC UA server

Object in Machine Model	Service Provider Type	Usage Type	PCo Configuration Elements
Service provider	Internal	Web server	Service provider in database, an agent instance for Web server
Tag	Internal	-	Tag
Service	External	-	Service, only metadata, display in the PCo Management Console is not possible
Service	Internal	Direct destination call	Service, method definition, method notification with reference to a destination system
Client proxy A client proxy is generated automatically in the Machine Model based on the service providers and service parameters	-	Multi	Destination system of the type multiple call destination system
Client proxy	-	Universal Web server	Destination system of the type Universal Web service destination system (RESTful)
Client proxy	-	OPC UA server	Destination system of type OPC UA destination system

3.4 PCo as OPC UA Server and as Web Server

You can configure agent instances of Plant Connectivity as OPC UA servers or as Web servers.

If it is started as an OPC UA server, the agent instance can receive and process method calls from OPC UA clients. If you run an agent instance as a Web server, the agent instance can provide methods in the form of service operations and make them accessible to Web service callers. In the Management Console, you can configure the actions that are to be executed by a client when a method is called. (See: OPC UA Server [page 455] or Web Server [page 464].)

In principle, there are two ways of executing methods using a PCo server:

- **Method with direct destination system call**

You can configure the method as a method with direct destination system call. To do so, you define the name of the method, the input parameters, and the output parameters manually. Then you assign to the method a method notification using which you can call exactly one destination system. (See also: Method Notification [page 55].)
At runtime, the method notification is executed when the method is called and the destination system defined there is called. If this destination system call delivers return values, these can be written back to the output parameters of the method and thereby be returned to the caller of the method.

- **Enhanced method processing (EMP)**
  You can load the assembly (DLL) of an enhanced method processing (EMP) to the PCo Management Console and activate and configure the methods defined there. The input and output parameters of the methods and the method names are already defined in the EMP assembly and cannot be changed. By creating a method notification you can activate individual methods of the EMP assembly and make them visible and executable for callers. The action that is to be executed when the method is called is defined for EMP methods in the implementation. Depending on the implementation of the EMP method, an additional destination system call needs to be configured.
  At runtime, the method of the EMP implementation is executed by PCo and the results of the execution are written back, if necessary, to the output parameters of the method. If the EMP method requires a destination system call, this call is carried out at the appropriate place using the program logic in the EMP implementation. (See also: Enhanced Method Processing (EMP) [page 57].)

Usage Options

Using the definition of a method **with direct destination system call and usage of the multiple call destination system**, you can run a defined sequence of steps if the method is called by a client. For example, you can use such methods to control processes at production plants for which individual parts of the plant need to be coordinated.

By using the assembly (DLL) *Locking* that is delivered as standard in EMP, you can use an agent instance as a lock server for a part of a plant that is needed by various production steps but can only process one request at a time.

3.4.1 Method Notifications

Use

You can use a method notification to activate a method of a PCo OPC UA server or a PCo Web server and define which action you want to run if the method is called up by a client. See also: Methods [page 11].

The functions offered in the method notification are related to the notification in the notification process but differ in detail, depending on the type of method.

- For methods that you have configured manually (methods of the type *direct destination system call*), you define the trigger condition based on the method input parameters. You then configure the output expressions that are forwarded to the input variables of the destination system call. The output expressions are also based on the method input parameters. You can assign exactly one destination system to the notification. If the destination system delivers return values, such as the OPC UA destination system, Web service destination system, or the multiple call destination system, you can assign the output variables of the destination system call to the output parameters of the method.
- For methods that you have loaded into the PCo configuration using an EMP implementation, a distinction is made between methods with and methods without a destination system call.

- For **EMP methods without a destination system call**, you can only configure the trigger condition based on the method input variables. Processing of the method call is adopted directly from the implementation. The input and output parameters of the method are received directly from the implementation or returned to PCo.
- For **EMP methods with a destination system call**, you can also define a trigger condition. The implementation of method processing provides for a destination system call at the appropriate place in the logic. The EMP implementation provides special input variables for the destination system call that you need to assign to output expressions in the method notification.
  You can **configure exactly one destination system in the notification**. The output variables of the destination system call are assigned to the output variables provided by the EMP implementation so that the implementation can continue to work with the result of the destination system call. The input and output parameters of the method are received directly from the EMP implementation or returned to PCo.

Runtime Behavior of Method Calls

You can create multiple method notifications for a method. However, you need to maintain the trigger conditions so that no more than one of the notifications is executed at runtime. Depending on the input parameters of the method, you can trigger various actions, for example, various destination system calls.

> **i** Note
>
> If no method notification is executed at runtime due to the trigger conditions, process control goes back directly to the caller. Any output parameters are supplied with their initial value.

The method notification is processed directly when the method is called; in other words, in technical terms, in the same thread of the Windows process through which the agent instance is running. If the method is configured as a **synchronous method**, PCo waits until the end of processing of the destination system call or of processing in the EMP implementation. Only then does process control go back to the caller, with the output parameters being supplied according to the configuration. For **asynchronous methods**, process control returns to the caller immediately after processing is triggered while the destination system call runs in an asynchronous thread in the background. Any output parameters of the method are supplied with the initial value.

Configuration Options

The method notification reveals its potential in particular in conjunction with the multiple call destination system and the definition of suitable trigger conditions. The configuration involves the following steps:

- You define a method for the direct destination system call. See also: Server Method Definitions Tab (OPC UA Server) [page 461] or Web Server Method Definitions Tab (Web Server) [page 469].
- You create a method notification for the method.
- You configure a multiple call destination system as a destination system, which you supply with the input parameters of the method.
  In the multiple call destination system, you can program complex processes, such as the multiple call of various destination systems, processing of interim results, branching conditions, and loops using configuration. See also: Multiple Call Destination System [page 353].

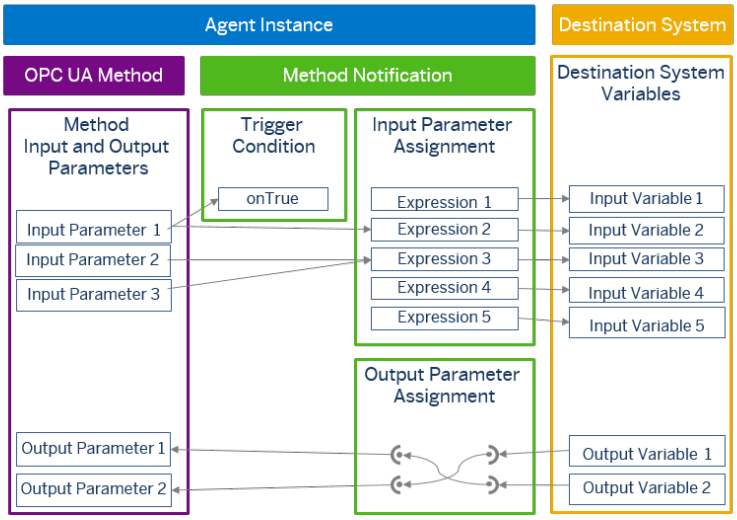
- By creating multiple method notifications for the same method that have different trigger conditions, you can, depending on the input parameters of the method, call different multiple call destination systems that model different processes.

> **i** Note
>
> If the configuration options described are not flexible enough, you can implement an enhanced method processing (EMP) in which you can program the actions when calling a method.

Example

The example shows variable assignment in method processing with a direct destination system call. The example also applies to Web service methods.

Method Notification (Example)



3.4.2 Enhanced Method Processing (EMP)

EMP Concept

Use

Using the concept of enhanced method processing (EMP), you can program yourself the actions that run when a method of a PCo OPC UA server or a PCo Web server is called, if you want to achieve maximum flexibility.

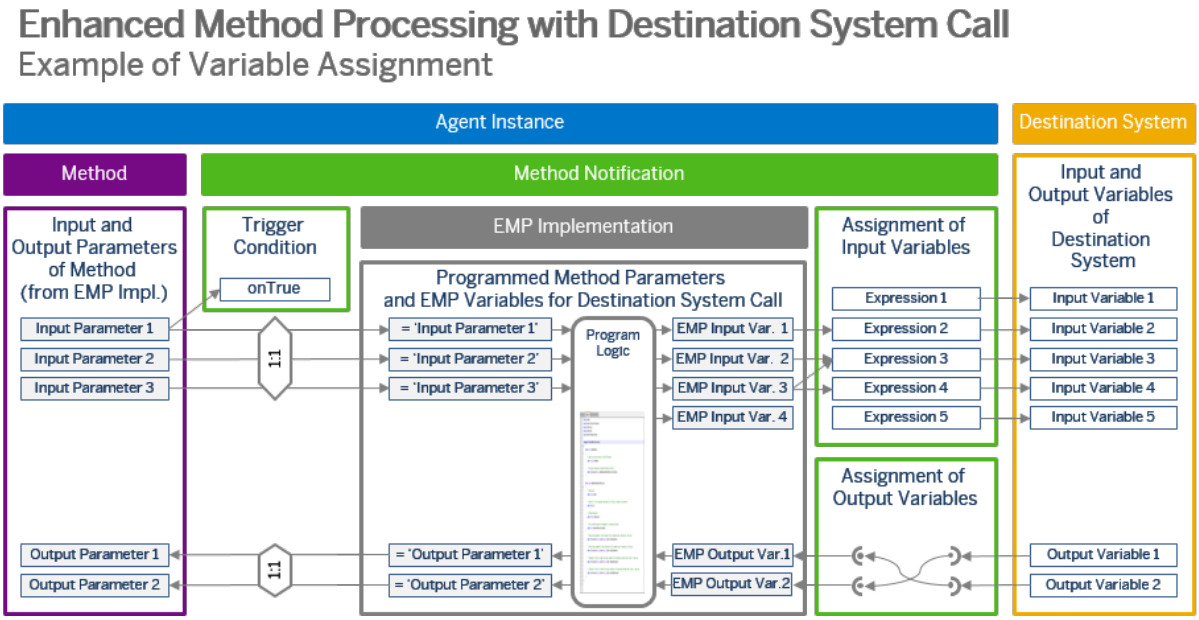
You can use EMP if the configuration options that result from the interaction of methods with a direct system call, method notification and multiple call destination system, are not sufficient to cover your requirements.

Implementation

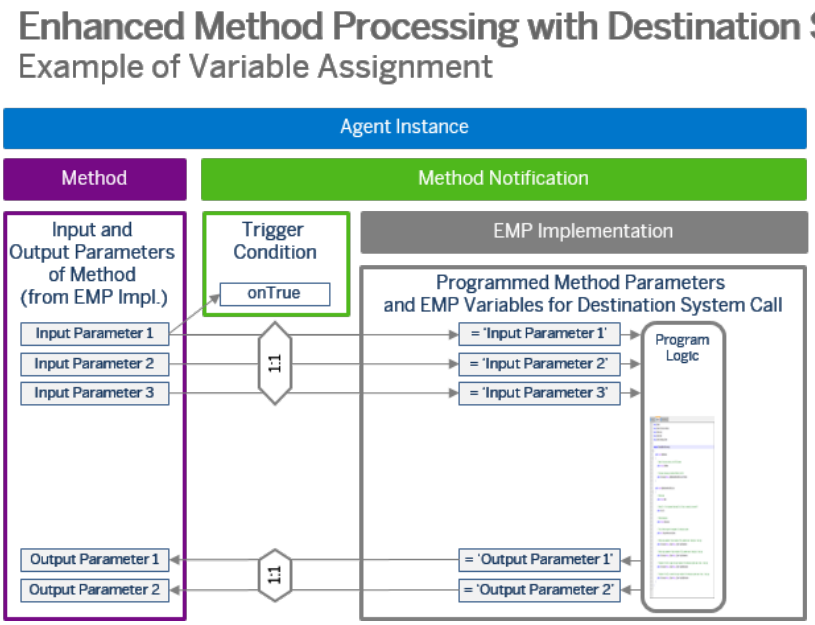
To use EMP with your own programming, you need `Microsoft Visual Studio 2012` or higher as a programming environment. You set up the .NET-based implementation of a class that inherits from the PCo class `MethodProcessingFramework.ApplicationMethodsBase` and that is compiled to an assembly. For details about the procedure during the implementation and for an example, see the Implementation Guide *Enhanced Method Processing (EMP) in Plant Connectivity 15.1*. (See: SAP Note 2354578.)

With the implementation of the EMP, you provide methods that can be made callable using the PCo OPC UA server or Web server. From the programmed flow of a method call, you can, if required, execute a destination system call whose return values can flow into further programming of the method. For this purpose, you provide special input and output variables in the EMP implementation that you can use in the Management Console to parameterize the destination system call. The EMP implementation of the method is called up using the method notification that you assign to the method.

The following overview shows how the EMP implementation is embedded in the configuration of the Management Console if a destination system is to be called from the EMP implementation.



If the EMP implementation handles the program logic completely when the method is called and does not call a destination system, you implement an EMP method without destination system call. The following graphic shows variable assignment for this case:



> **i Note**
>
> In this case, the method notification has only one trigger condition. No other settings are necessary and so they are hidden for configuration using the Management Console.

You can use the same EMP implementations for a PCo UA server and a PCo Web server. However, the different message log of UA method calls and service operation calls requires several adjustments to the context of the server:

- If you create an EMP implementation for a PCo UA server, specific status codes need to be returned to the method caller according to the OPC UA standard. You can return to the caller the status code that is dependent on the method execution result using an additional parameter that is designed as a `TypedObject` (see also SAP Note 2395778). This parameter is not relevant in a Web server environment.
- If you want to use an EMP implementation in the context of a PCo Web server, it might be necessary to adjust the format of the response to the requirements of the caller. For this purpose, PCo provides a separate interface XYZ with which you can, if required, influence the response specific to each method.

The result of the implementation is a compiled assembly (DLL) that you copy to the system directory of the PCo installation. As a rule, the system directory is in the *C:\Program Files (x86)\SAP\Plant Connectivity \System* directory if you have installed PCo using the standard specifications on a Windows 64-bit operating system.

Configuration

To be able to use the methods defined in the EMP implementation, you need to load the method definitions from the assembly into the configuration of the OPC UA server or Web server in the Management Console. To do so, follow the steps that are described under Server Method Definitions Tab (OPC UA-Server) [page 461] or Web Server Method Definitions Tab (Web Server) [page 469].

You need to assign at least one method notification to the methods that are to be visible to a client and that are to be callable, and, if necessary, carry out variable assignment according to the principle described above.

More Information

Standard EMP Implementation Locking [page 60]

Standard EMP Implementation Key-Value Pair Buffer [page 63]

3.4.2.1 Standard EMP Implementation Locking

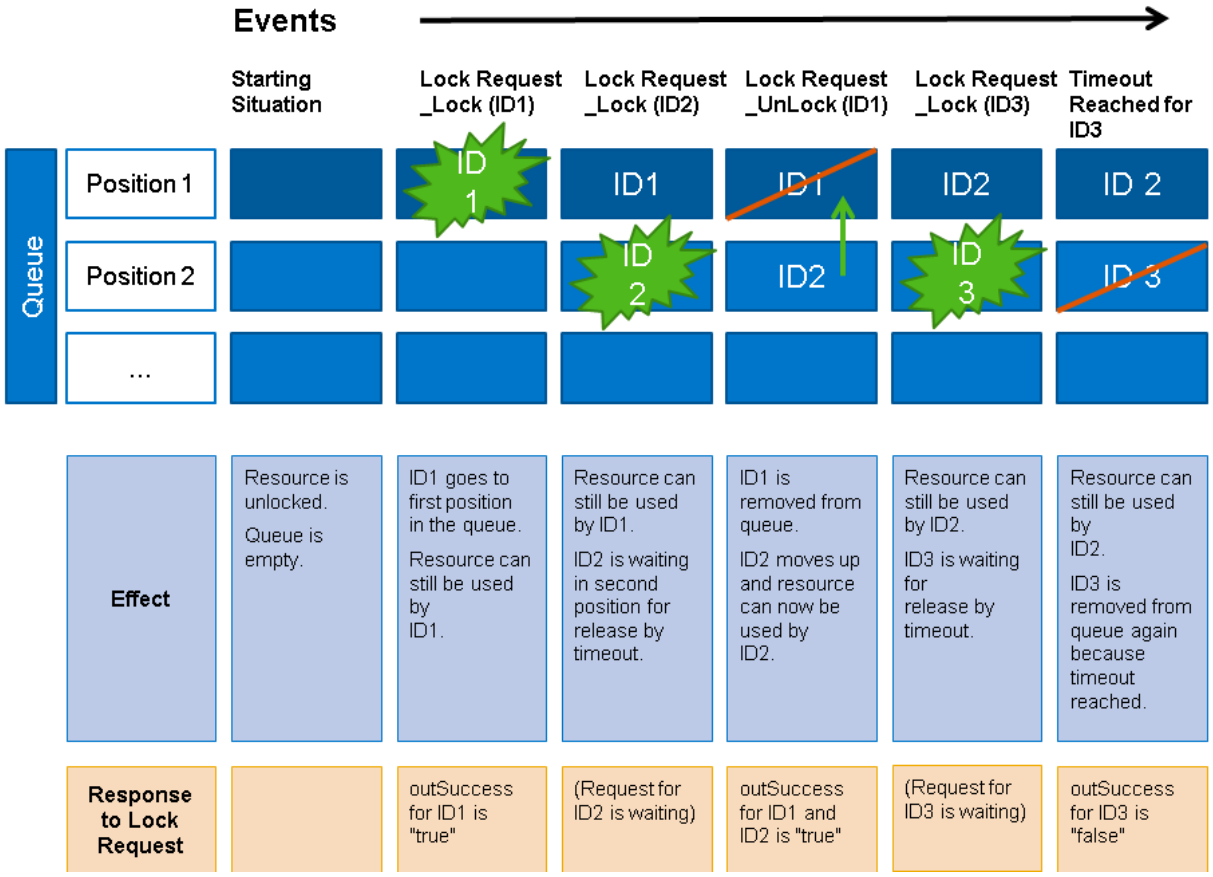
In the standard system, SAP delivers the EMP implementation `Locking` with which you can use an agent instance as a lock server for a resource in production.

The EMP implementation provides a queue for lock requests. Each entry in the queue is given a unique ID, for example, the SFC of the product that is to be produced by the resource. Only the first entry in the queue has access to the resource. The other lock requests are either rejected after a specified time or they move to the first position in the queue during this time and gain access to the resource. The implementation provides the following methods:

Method	Input Parameter	Output Parameter	Function
`_Lock`	`inHandle` `inTimeout`	`outSuccess`	Locks the resource that is modeled by the agent instance. The input parameter `inHandle` needs to contain the unique ID of the lock request. This might be, for example, the SFC of the product that is to be produced by the resource. If the lock cannot be set within the timeframe set by `inTimeout`, `outSuccess` is “false”; otherwise it is “true”. If the lock was able to be set, other lock requests are rejected using the method `_Lock` provided the resource has not been unlocked again using `_Unlock` or `_UnlockAll` .

Method	Input Parameter	Output Parameter	Function
`_Unlock`	`inHandle`	`outSuccess`	Unlocks the resource that is modeled by the agent instance. The input parameter `inHandle` contains the unique ID of the lock request. If there is a lock of the resource for this ID, the lock is removed and the output parameter `outSuccess` is set to “true”. Otherwise it is “false” and any existing lock remains. If the lock for the ID was able to be removed, the next ID in the queue moves up and can use the resource.
`_UnlockAll`		`outSuccess`	The `_UnlockAll` method removes all lock requests that have been sent to the resource. In this case, the output parameter `outSuccess` is “true”. For example, you use the method for initializing the production flow.
`_ListLocks`		`outHandles`	Provides the list of IDs of the lock requests that have currently been sent to the resource and that have not yet been rejected. The first ID in this list is active and can use the resource. The subsequent IDs are in the queue and are rejected if they cannot go to the top of the queue before the timeout.

Using a sequence of lock requests, the following graphic shows the behavior of the lock server in the EMP implementation locking:



None of the methods of the EMP implementation `Locking` need any additional destination system call.

Example

In production, you are using a robot that is used by various parts of the production plant and therefore gets work orders from various clients. Then you can use a PCo agent instance as a central lock server for this robot and make sure that each client needs to request the lock on the resource before it can send a work order to the robot.

3.4.2.2 Standard EMP Implementation Key-Value Pair Buffer

In the standard system, SAP delivers the EMP implementation `KvpBuffer` that represents a key-value pair buffer. You can use the implementation to convert every agent instance into a buffer agent instance.

Function of the Buffer

Using the buffer, you can, using method calls, store data in the form of **key-value pairs** and then enrich notification messages with this data.

The prerequisite for using the implementation is that you are running the agent instance as an OPC UA server or as a PCo Web server.

The buffer entries can be persisted in the local file system or not persisted in the working memory only. Different callers can access the same buffer data.

Example

- A production line has a transport system of carriers. Each carrier has a unique ID. There might be a product on each carrier that corresponds to an SFC in the SAP Manufacturing Execution system. You want the assignment of carrier ID to SFC to be retained and this assignment must not be lost if the PCo agent instance has to be restarted.
- A machine delivers measurement values frequently that are written via an OPC UA agent to an SAP HANA database as a tag-based notification. You want to enrich the measurement values with the name of the worker who is currently on shift. The worker's name is kept in PCo and updated three times a day via an SAP MII user interface.

Implementation Methods

The implementation provides the following methods:

Implementation Methods

Method	Input Parameter	Output Parameter	Function
`_ClearBuffer`			Deletes all keys and values from the buffer.

Method	Input Parameter	Output Parameter	Function
`_GetAllKeys`		`outKeys`	Provides a string array `outKeys` that contains all the keys that are currently in the buffer.
`_GetAllKeyValuePairs`	`inClearBuffer`	`outKeyValuePairs`	Provides a string array `outKeyValuePairs` that contains all the key-value pairs that are currently in the buffer. If the input parameter `inClearBuffer` is set to the value *True*, the entire buffer is emptied after all key-value pairs have been read.
`_GetAllKeyValuePairsJson`	`inClearBuffer`	`outKeyValuePairs`	Provides a JSON-formatted character string `outKeyValuePairs` that contains all the key-value pairs that are currently in the buffer. If the input parameter `inClearBuffer` is set to the value `True`, the entire buffer is emptied after all key-value pairs have been read.
`_GetKeysByValue`	`inValue`	`outKeys`	Returns a string array `outKeys` that contains all keys that have the value `inValue`.
`_GetNumEntries`		`outNumEntries`	In the output parameter `outNumEntries` returns the number of all key-value pairs that are currently in the buffer.
`_GetValueByKey`	`inKey`	`outSuccess` `outValue`	Returns the value `outValue` for the key `inKey`. If the key `inKey` is in the buffer, the output parameter is `outSuccess True`; otherwise it is `False`.

Method	Input Parameter	Output Parameter	Function
`_KeyExists`	`inKey`	`outKeyExists`	Checks if the key `inKey` occurs in the buffer. If the key does occur, the output parameter `outKeyExists` is set to `True`, otherwise it is set to `False`.
`_RemoveKey`	`inKey`	`outSuccess`	Deletes the key-value pair with the key `inKey` from the buffer. If the key `inKey` existed in the buffer before deletion, the output parameter `outSuccess` is set to `True`, otherwise it is set to `False`.

`_SetKeyValue`	`inKey`
	`inValue`
	`inPersistent`
			If the key `inKey` does not yet occur in the buffer, the method adds a new key-value pair with the key `inKey` and the value `inValue`. If the key `inKey` already occurs, the value of the key-value pair is updated. If `inPersistent` is set to `True`, the key-value pair is written to the local data carrier and is still available after the agent instance is restarted. If `inPersistent` is set to `False`, the key-value pair is only kept in the working memory.

If you set the input parameter `inPersistent` to `True` in the `_SetKeyValue` method, the EMP implementation **KvpBuffer** generates the directory `C:\ProgramData\SAP\EMP_KvpBuffer\[ID of agent instance]` in the Windows directory **ProgramData** and generates a file for each key-value pair with the key as file name and the value as file content. For this reason, the key `inKey` must only contain characters that are permitted in a Windows file name if you want to persist the entry. In general, the EMP implementation ignores and cuts off spaces before and after a key.

You can determine the ID of the agent instance from the XML file, for example, after the export of the agent instance.

> ⚠ Caution
>
> If you set the input parameter `inPersistent` to `True` in the `_SetKeyValue` method, you may only use the EMP implementation **key-value pair buffer at most once** for each agent instance. Otherwise, the files

in the `C:\ProgramData\SAP\EMP_KvpBuffer\[ID of agent instance]` directory cannot be assigned correctly to the various buffer instances.

3.5 Integration with Third-Party Systems Using Web Services

Use

You can use PCo directly to transfer data from production to a third-party system, such as an SAP ME system. To do this, you can use the notification process. You can subscribe to specific tags of the data source. When specific events occur, PCo sends notification messages to the destination system by means of a Web service call.

If, for example, you want to connect SAP ME, you need to create a universal Web service destination system in PCo. SAP ME provides its own Web services, such as the Web service `ProductionProcessingInClient`.

PCo retrieves variable values such as the *SFC number (SFC)* from the data source and transfers them to the SAP ME system, for example, to a particular ME activity. PCo can predefine fixed values for specific parameters of the Web service.

Integration

For more information about the Web services provided by SAP ME, see https://help.sap.com/viewer/p/SAP_MANUFACTURING_EXECUTION.

Example

The following example shows which settings you need to make in the *PCo Management Console* for you to be able to transfer the *SFC* to SAP ME using the *ProductionProcessingInClient* Web service provided by SAP ME, for example.

Example

A barcode reader is installed in front of the RES1 machine that records the SFCs of all plant pieces before they are processed. The results from the barcode reader are stored on an OPC DA server and can be retrieved from the OPC DA server in the SCAN1 tag. For each scan operation, you want a message to be sent to SAP ME so that the status of the SFC can be set there to *In Process at Resource RES1*. For this purpose, SAP ME provides the *ProductionProcessingInClient* Web service, which requires the SFC, plant, resource, and the operation. Since the machine is stationary and always executes the same operation, the machine, the plant, and the operation can have constant values. The SFC originates from the OPC server and is therefore variable.

.

Settings in the PCo Management Console

1. In PCo, you create an OPC DA source system, for example, for a KEPWARE OPC server.
2. You create a universal Web service destination system:
   - On the *Web Service Settings* tab, you define the settings for the Web service provided by SAP ME (see also: Web Service Settings Tab [page 274].)
   - On the *Operation Configuration* tab, in the *Operation* field, you select the *Start (StartRequestMessage_sync StartRequest_syn)* entry.
     For more information, see Operation Configuration Tab [page 286].
3. You create the agent instance for the OPC source system to connect the OPC source system to the destination system.
   - Define the settings on the *Host* tab; for example, log level **`Verbose`**.
   - On the *Subscription Items* tab, choose the *Browse* pushbutton. The PCo system then displays the available tags, for example, for `Device 1`. Select the tag `SCAN1`, for example.
4. Create a *notification*. The notification is used to be able to transfer the information from the source system to the destination system.
   - On the *Output* tab, enter the following expressions and values:

Expression Name	Expression Value
SFC	“SCAN1” **i Note** SCAN1 is the subscribed tag.
SITE	“QATEST”
OPERATION	“OP1”
RESOURCE	“RES1”
USER	“SITE_ADMIN”

> **i Note**
>
> The expression values QATEST (name of the plant), OP1 (operation), RES1 (resource used), and SITE_ADMIN (user name) are master data that you created previously in SAP ME. The output value SCAN1 is the subscribed tag. The tag value is transferred here to SAP ME. In this example, the tag value is an SFC that is read and transferred to SAP ME.

   - Click on the *Destinations* tab, where you select the destination system and assign the appropriate output values to the destination variables. You can also have an automatic proposal of the assignment between the Web service variables and the output expressions. The assignment can only be made automatically if the Web service variable and the output expression have the same name.

> **i Note**
>
> You can also call service operations of a PCo Web server configured on a remote computer via the universal Web service destination system.

3.6 Integration with Third-Party Systems Using ENP

Use

Enhanced notification processing (ENP) enables you to flexibly control and document the data flow in production in connection with various destination systems, for example, with Web services. In this way, you can connect a third-party system (such as SAP ME) to PCo and transfer data from machine level to the desired SAP ME activity using Web service calls. This makes it possible, for example, starting from PCo, to call a Web service provided by SAP ME, evaluate the result of the call, and then, depending on the results of the Web service call, call an additional Web service or write data back to a source system.

SAP delivers the standard enhancement **Destination System Calls with Response Processing** for enhanced notification processing with which you can execute one or multiple destination system calls and with which you can write back the results of the calls to the data source of the agent instance or other agent instances. This covers the most common requirements in communication between data sources of production and business systems.

If you want to implement requirements that go beyond the function scope of the SAP standard enhancement, you can implement a **customer-owned enhancement**. PCo provides an interface for this purpose that you can implement. The notification enhancement is mapped in the form of a destination system so that the enhancement is called as part of a notification process.

Enhanced notification processing enables you to do the following:

- You can call one or more destination systems one after the other in any order. This might be, for example, a regular Web service or a RESTful Web service.
- You can call mass-enabled destination systems (Web services) for multiple object instances, for example, for multiple SFCs of the SAP ME system.
- You can assign the output expressions of a notification to the parameters of a Web service statically or dynamically.
- You can perform a program-controlled evaluation of the results of a destination system call and react to the results accordingly.
- After a notification message is received, the destination system can send information to a specific agent instance and update, for example, a tag value in the source system.
- You can send data to a destination system of the type *ODBC destination system*. (See ODBC Destination System [page 338].)

Prerequisites

If you want to create your own implementation, the following prerequisites must be fulfilled:

- You are using **PCo 15.1 or higher**.
- You are using **.NET Common Language Runtime (CLR) 4.0** or higher.
- You are using a .NET development environment, such as **Microsoft Visual Studio 2012 or higher**.
- You have .NET development experience, preferably C#.
- You have experience in using Web services.

Integration

To facilitate the implementation of these functions, the *enhanced notification processing framework* has a library with help methods with which you can call Web services or agents.

If you have implemented a customer-owned implementation and then made the required settings in PCo, this enhancement is called automatically when notification messages are processed. (See also: Process with Enhanced Notification Processing (ENP) [page 69].)

See Also

Process with Enhanced Notification Processing (ENP) [page 69]

Destination System Calls with Response Processing [page 81]

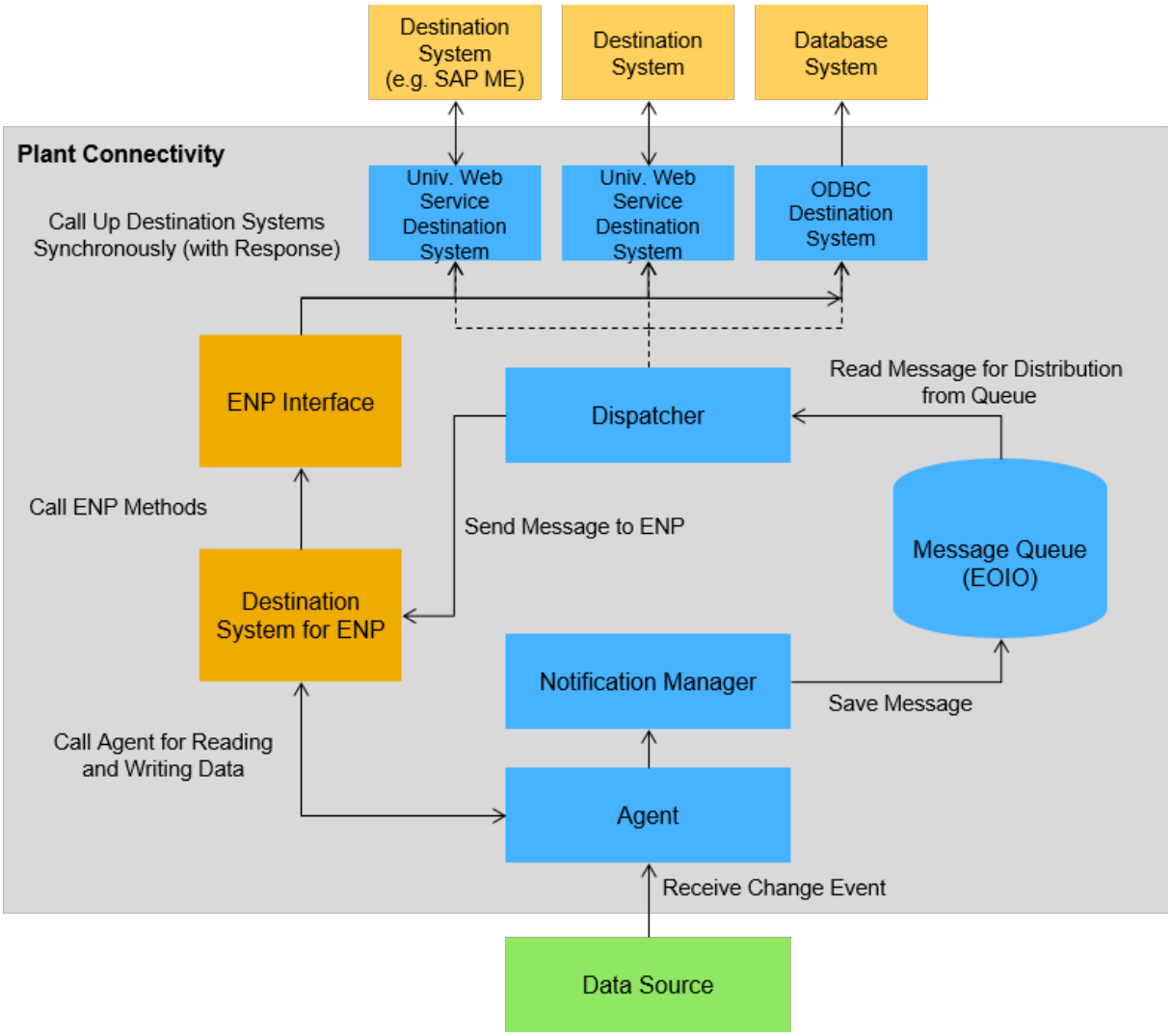
For more information about the integration with third-party systems in connection with enhanced notification processing (ENP), see the *Implementation Guide for Enhanced Notification Processing (ENP)* on SAP Help Portal under https://help.sap.com/pco. The Implementation Guide also contains example implementations.

3.6.1 Process with Enhanced Notification Processing (ENP)

Use

This document shows how enhanced notification processing (ENP) is integrated in the notification process.

Process



Notification Process with Enhanced Notification Processing (ENP)

1. PCo receives a change event from the connected source system.
2. The **notification manager** ensures that a notification message is built up.
3. The notification message is stored in the queue of notification messages (**message queue**). (See also: Notification Message Queue and Dispatch Settings [page 481].)
4. The **dispatcher** reads the notification message from the queue and forwards it to the destination system. Since enhanced notification processing (ENP) is attached in PCo in the form of a special destination system (**destination system for the notification enhancement**), the notification message is sent to this destination system.
5. The destination system for the notification enhancement forwards the notification message to the customer-owned enhancement or to the SAP standard enhancement *Destination System Calls with Response Processing*. This enhancement implements the enhanced notification processing (ENP) interface (`Interface ICustomLogic`).

6. Starting from this interface, the modules that you configured previously as destination systems are called synchronously. The ENP supports the following destination system types:
   - Universal Web service destination system (WSDL) for connecting an SAP ME system (see: Universal Web Service Destination System [page 273])
   - Universal Web service destination system (RESTful) for calling a RESTful Web service
   - Database system (see: ODBC Destination System [page 338])
7. Using the modules, the variable values taken from the data source, such as the SFC number, are transferred to the **third-party system** (for example, SAP ME).
8. The connected third-party system, such as SAP ME, can now send a response to the interface.
9. The interface can interpret the response and then trigger the following actions:
   - Call up an additional Web service or an additional supported destination system
   - Start an agent call to read or to write tag values in a source system
   - Send data to the ODBC interface

See also

Destination System Calls with Response Processing [page 81]

3.6.2 Implementing and Configuring Enhanced Notification Processing (ENP)

Use

You can either create your own implementation of the `ICustomLogic` interface or use an implementation from a third-party. The DLL of a third-party must be compiled for a 32-bit operating system.

Prerequisites

If you want to implement a customer-owned implementation yourself, you need a development environment for the `.NET`development. SAP recommends that you use *Microsoft Visual Studio 2008* or higher.

Process

1. Implementing your own enhancement in the *Microsoft Visual Studio*:
   - You create a class that implements the `ICustomLogic` interface.
   - You create an implementation for a constructor without parameters.

- You create implementations for the following interface methods:
  - `GetLogicModules`
  - `GetCustomVariables`
  - `Initialize`
  - `Send`
- You compile your implementation. A **DLL** (Dynamic Link Library) is thus generated from the class, which you then use for the configuration of enhanced notification processing (ENP) in the PCo Management Console.

> **i Note**
>
> For technical details about implementing enhanced notification processing (ENP), see the *Implementation Guide for Enhanced Notification Processing (ENP)* on *SAP Service Marketplace* under *service.sap.com* > *instguides* > *SAP Business Suite Applications* > *SAP Manufacturing* > *Plant Connectivity* > *Plant Connectivity 15.1*.

2. Making settings in the *PCo Management Console*:
   The concept of enhanced notification processing (ENP) envisages that the implementation of a customer-owned enhancement is created without specific reference to a particular *destination system* and a particular agent instance. The advantage of this concept is that you can use the same implementation for different processes.
   For that reason, you must make specific settings in the *PCo Management Console* to link your implementation with a specific destination system and a specific agent instance. You need to perform the following configuration activities:
   - Copy the DLL into the PCo system directory
     You copy the DLL that you generated for your implementation into the system directory of your PCo installation.

     > **i Note**
     >
     > In the case of a Windows 32-bit installation, you can find the system directory under the following path: `<Installation Drive>:\Program Files\SAP\Plant Connectivity\System`.
     >
     > In 64-bit systems, you can find the system directory under the following path: `<Installation Drive>:\Program Files (x86)\SAP\Plant Connectivity\System`.
   - Create a source system
   - Create a destination system of one of the following supported types and make the relevant settings:
     - Web Service Destination system (see Server Settings Tab (WS Destination System) [page 394])

       > **i Note**
       >
       > You must create a separate *Web Service Destination system* for each individual Web service operation that you want to call using your notification enhancement.
     - ODBC destination system (see ODBC Destination System: Configuration Tab [page 340].)
   - If you want to send data to a normal Web service (*Web Service Destination system*), you also need to create the configuration of the service operation.
     To do so, click on the *Operation Configuration* tab. Then choose the *service operation* you want. You then define the variables for the Web service fields that are to be filled dynamically by the notification enhancement. (See Operation Configuration Tab (WS Destination System) [page 396].)

- Create an agent instance for the source system
- Settings on the *Notification Processing* tab
  If you want the notification enhancement calls via the notification process to retain a specific sequence, you need to activate the option *Process Notification Messages Exactly Once in Order* on the *Notification Processing* tab in the agent instance settings.
- Assign a DLL to the agent instance
  On the *Notification Processing* tab of the agent instance, you assign the DLL and create the destination system for the notification enhancement. (See Enhanced Notification Processing [page 73].)
- Creating a notification
- Define enhanced notification processing as the destination of a notification
  On the *Destinations* tab, you need to define enhanced notification processing as a destination and assign the modules and variables. (See: Assignment of Modules and Variables (ENP) [page 75].)

3.6.2.1 Enhanced Notification Processing

Use

In the *Enhanced Notification Processing* screen area, you can specify whether you want to use an enhancement for your notification processes. You have the following options for the notification enhancement:

- SAP standard enhancement *Destination System Calls with Response Processing* and *Automatic Configuration Backup*
  These SAP standard enhancements are delivered with the standard installation of *SAP Plant Connectivity*. If you choose one of these options, the corresponding enhancement is included automatically in notification processing.
- Customer-Owned Enhancement
  With this option, you create your own enhancement implementation and assign it here. With the enhancement implementation, you can define your own functions that you want PCo to execute in the notification process. For example, you can interpret information from the data source and then call a specific sequence of Web services.

Moreover, you need to create the destination system for the notification enhancement. This destination system is a special destination system of the agent instance and is therefore not displayed in the destination systems area.

Prerequisites

If you want to use a customer-owned enhancement, you first need to create and compile your implementation. During the compilation, a *Dynamic Link Library (DLL)* is generated.

Procedure

1. Choose the *Enhanced Notification Processing* tab.
2. Select the notification enhancement you want:

Field	Description
No Enhanced Notification Processing	With this option, there is no enhanced notification processing.
Destination System Calls with Response Processing	With this option, the system automatically uses the SAP standard enhancement *Destination System Calls with Response Processing* (see also: Destination System Calls with Response Processing [page 81].)
Automatic Configuration Backup (New)	You can use this enhancement to create data backups of the configuration. For more information, see Setting Up an Automatic Backup [page 597].
Customer-Owned Enhancement	With this option, you create your own enhancement implementation and assign it in the *Details of the Enhancement Implementation* screen area. This implementation is an enhancement in the form of a Dynamic Link Library (DLL) with which you can implement your own functions. For example, you can interpret information from the data source and then call a specific sequence of destination systems (Web services).

3. If you have selected the *Customer-Owned Enhancement* option, you need to choose the *Browse* pushbutton to search for your implementation.
4. Then select the desired *Dynamic Link Library* (DLL) or *executable file* that you generated previously.
   If, in your implementation, you have only defined one class that implements the enhanced notification processing interface, this class appears automatically in the *Class* field. If you defined more than one class, you need to select the appropriate class.
5. If you want to delete the configuration for the DLL and the class, choose *Reset*.
6. Create a destination system for notification processing by choosing the *Create Destination System* function key.
   PCo displays the status of the destination system and its usage.

   > **i Note**
   >
   > You can then select this destination system on the *Destinations* tab of the notification by selecting the `/Enhanced Notification Processing` entry in the *Destination System Type* field. (See also: Assignment of Modules and Variables (ENP) [page 75].)

7. To delete the destination system, you can choose the *Delete Destination System* pushbutton. You can only delete the destination system if it is not being used in a notification.

More Information

For more information about using the customer-owned enhancement, see the **Implementation Guide for Enhanced Notification Processing** on the *PCo product page* under *Additional Information*.

3.6.2.2 Assignment of Modules and Variables (ENP)

Use

In the screen area *Module and Variable Assignment*, you define the modules and variables for enhanced notification processing (ENP).

Prerequisites

- On the *Notification Processing* tab, you have chosen either *Destination System Calls with Response Processing* or *Customer-Owned Enhancement* for your agent instance and you have created the destination system for the agent instance. (See: Enhanced Notification Processing [page 73].)
- You have created one or multiple destination systems. The following destination system types are supported:
  - Web service destination system [page 393]
  - Destination system type
  - ODBC destination system [page 338]

  For each Web service destination system, you have also configured the service operation and variable definition of the selected Web service. (See: Destination System: Operation Configuration Tab (WS Destination System) [page 396].)

Procedure

Add Destination System

1. In your notification, choose the *Add Destination System* icon on the *Destinations* tab.
   The *Add Destination System* dialog box appears.
2. In the *Destination System* field, select the entry */Enhanced Notification Processing*.
3. In the *Name* field, enter a name of your choice for the destination system and choose OK.
   You have assigned the destination system for enhanced notification processing. The system now displays the destination system in the *Destinations* screen area.

Agent and Destination System Calls

1. Expand the entry for the destination system and choose the *Module and Variable Assignment* row.

The following two assignment tables are now displayed:

- *Agent and Destination System Call Modules*
- *Assignment of Enhancement Variables*

2. In the *Agent and Destination System Call Modules* table, assign the agent instances and destination systems (*Web Service Destination systems*) that you want to call using enhanced notification processing. The table is structured as follows:

Column	Description
Module	All the modules that are provided by the implementation of notification processing are displayed in this column. The modules set up the connection to agent instances and destination systems that are to be called up from ENP.
Module Type	Specifies the type of module. The following module types are possible: ○ *Destination system call* You can use this module to call one of the supported destination systems. You must assign the destination system in the *Destination System* column. ○ *Agent call* From ENP, you can use this module type to specify that data coming from the connected destination system is transferred to or read by the current or other running agent instances.
Exception Handling	Select this checkbox if you want to deal with exceptions that occur in the destination system call in a controlled way. (See: Exception Handling in Enhanced Notification Processing [page 79].) If you do not select this checkbox, exception handling is not enabled. Without exception handling, every exception within the destination system leads to termination of notification message processing and to an error message in the agent log.
Edit	To configure exception handling, choose the *Edit* pushbutton.
Destination System	For a module of the type *destination system call*, you must select a destination system here. Only the permitted destination systems are offered in the dropdown box.
Agent Instance	In the case of a module of the type *Agent Call*, you must select an agent instance here. Only the permitted agent instances are offered in the dropdown box.

Assignment of Enhancement Variables

1. The *Destination System Variables* and *Source System Tags* tabs are displayed in the lower screen area *Assignment of Enhancement Variables*.
2. Choose the *Destination System Variables* tab. All the destination system variables that PCo determined for the destination system selected previously are displayed here. Now assign the appropriate notification enhancement variables to the destination system variables. The enhancement variables are offered in a dropdown box. The assignment table is as follows:

Column	Description
Destination System Call Module	All the modules for calling the destination system(s) are displayed here.
Destination System Variables	All destination system variables are displayed here that are determined from the configured destination system: ○ For an *ODBC destination system*, the system displays the table columns configured as variables. ○ In the case of a destination system of the type *Web service destination system*, you see the input and output parameters configured as variables as well as the calculated variables (only for Web service destination systems).
Category	Shows if the variable is a request variable, a response variable, or a calculated variable.
Data Type	Indicates the data type of the destination system variables.
Enhancement Variable	Here you assign the appropriate enhancement variable to each destination system variable. In the dropdown box, the system only offers the enhancement variables that you have defined in your implementation or that were provided by the SAP standard enhancement. You can only assign variables with matching data types.

> ⚠ Caution
>
> Note that you need to assign an enhancement variable to each request variable. However, you do not need to assign an enhancement variable to each response variable and to each calculated variable.

3. Choose the *Propose Assignment* pushbutton if you want the system to propose the appropriate enhancement variables. The prerequisite for this is that the enhancement variables have the same name or a similar name to the destination system variables, and have the appropriate data type.
   You can use this function most effectively when configuring the SAP standard enhancement *Destination System Calls with Response Processing* because same-name variables are used by preference there. (See also: Destination System Calls with Response Processing [page 81].)
4. Choose the *Source System Tags* tab. Here you can browse for tags for the selected agent instances and assign them to the enhancement variables. You only need to select these source system tags if you want to

read or write tag values by means of an agent instance in enhanced notification processing. (See: Assigning Source System Tags [page 78].)

5. If the names or data types of the modules and the variables are changed after you have configured them in the *PCo Management Console*, you receive a warning when you call the configuration dialog again. The system then removes invalid configurations automatically and you can then complete the configuration.

See Also

For more information about the assignment of modules and variables, see the *Implementation Guide for Enhanced Notification Processing (ENP)* on *SAP Service Marketplace* under *⁄instguides* > *SAP Business Suite Applications* > *SAP Manufacturing* > *Plant Connectivity* > *Plant Connectivity 15.1* .

3.6.2.2.1 Assigning Source System Tags

Context

On the *Source System Tags* tab, you can search for tags for the selected agent instances and assign them to the enhancement variables. You only need to select source system tags if you want to read or write tag values by means of an agent instance in response processing within enhanced notification processing.

Procedure

1. Click on the *Source System Tags* tab.

   The tab is displayed.
2. Choose the *Insert Row* or *Append Row* pushbutton.

   PCo displays a new row with the agent call module *agent*.
3. Choose the *Browse* pushbutton.

   PCo displays a dialog box.
4. If the namespace is empty, click *Browse* (next to the *Filter* field).

   The namespace of the source system connected via the agent appears.
5. To be able to display the entire hierarchy of the namespace, open the top node (*Root*) and then all lower-level nodes.
6. Click to select the tag or tags that you want and then choose *Add Selected Items*.

   The selected tags are displayed in the lower screen area (*Selected Objects*).

7. To close the dialog box, click *OK*.

   PCo now adds the selected tags to the *Source System Tags* tab. If you have selected more than one tag, the system creates a new row for each tag.
8. Now you assign the appropriate enhancement variable to each tag. The variables can be selected from the dropdown box.

   > **i** Note
   >
   > The correct data type can only be determined for an enhancement variable directly after the tags are browsed. If you leave the *Source System Tags* tab without first assigning the variables, the rows without variable assignment cannot be configured later. In this case, you need to delete these rows and select tags again.

3.6.2.2.2 Exception Handling

Prerequisites

- In enhanced notification processing:
  You have configured an agent instance that is to call up one or multiple destination systems using a customer-owned enhancement or the SAP standard enhancement *Destination System Calls with Response Processing*.
  You call the *Destinations* tab in the **notification**. You have assigned a *Web Service Destination system* to a module of the type *destination system call* here. See: Assignment of Modules and Variables (ENP) [page 75].
  You have activated exception handling for this module by selecting the *Exception Handling* checkbox.
- In the multiple call destination system:
  You have created a multiple call destination system with which one or multiple destination systems are to be called in a specific sequence. On the *Destination System Calls* tab, you have selected the *Exception Handling* checkbox in the *Sequence of Destination System Calls* table.

Context

You can react in a controlled way to exceptions, which can occur when destination systems are called, by configuring exception handling so you do not have to terminate the destination system call process but rather can continue working with defined values for the destination system output variables.

Exception handling can be called in the following applications:

- Enhanced notification processing
  You can activate exception handling for enhanced notification processing, for example, using the SAP standard enhancement *Destination System Calls with Response Processing* or a customer-owned enhancement.

- Multiple call destination system
  You can activate exception handling for individual steps in a multiple call destination system.

Errors can occur within the processing logic of a called destination system that go back to PCo as exceptions.

Some exceptions indicate actual errors, such as a destination system that is temporarily unavailable, incorrect input parameters, or errors in the processing of calculated variables in a Web Service Destination system.

However, some exceptions can be foreseen from the process flow or can be tolerated and do not need to lead to the termination of the processing of a notification message.

> **Example**
>
> A foreseeable exception might occur if in SAP ME an *SFC number* is to be set to *done* using PCo but this SFC was already set to *done* in a manual process. In this case, SAP ME sends an error message. This exception should no longer lead to an error message in the agent instance log but should rather be caught by exception handling.

Procedure

1. Choose  .

   The *Exception Handling for Module* dialog box appears.
2. Define a logical condition in the *Condition* field. Use the expression editor to formulate the condition. You can evaluate notification enhancement variables and the text for the exception message in the condition expression (variable `!EXCEPTION_MESSAGE!`). A return value of 0 means that the condition is false. A return value that does not equal 0 means that the condition is true. An empty condition expression has the same meaning as the return value 0, in other words, the condition is false.

   > **Example**
   >
   > You would like to define that the condition is true if the exception text contains the text "*Message 13979*". In this case, the condition expression is: `stringindexof('!EXCEPTION_MESSAGE!', "Message 13979")`
3. Specify what you want to happen in the case of a true condition and a false condition. In both cases, you can choose one of the following two options:
   - *Raise Exception*
     The exception is not caught. It is forwarded to the enhanced notification processing (ENP) that called the destination system. This corresponds to the behavior without exception handling. In this case, processing of the notification message is terminated.
   - *Catch Exception and Set Destination System Output Variables*
     The exception is caught. You can define the values for the destination system output variables in the *Set Value of Destination System Output Variables* table. You can use fixed values or expressions to determine the values.
     If an exception is caught, processing of the notification message continues. The return values then correspond to the values that you defined here in the table.

3.6.3 Destination System Calls with Response Processing

Definition

The SAP standard enhancement *Destination System Calls with Response Processing* is used within enhanced notification processing (ENP). You can use this standard enhancement to call up to three destination systems with the contents of a notification message, for example, from an OPC UA source system. Possible destination systems are:

- Multiple call destination system
- ODBC destination system
- OPC UA destination system
- Query destination system
- Universal Web service destination system
- Web service destination system

All these destination systems can return responses that can then be written back to the source system or to up to two additional source systems of selected agent instances. Alternatively, you can call up additional destination systems using the results of previous destination system calls.

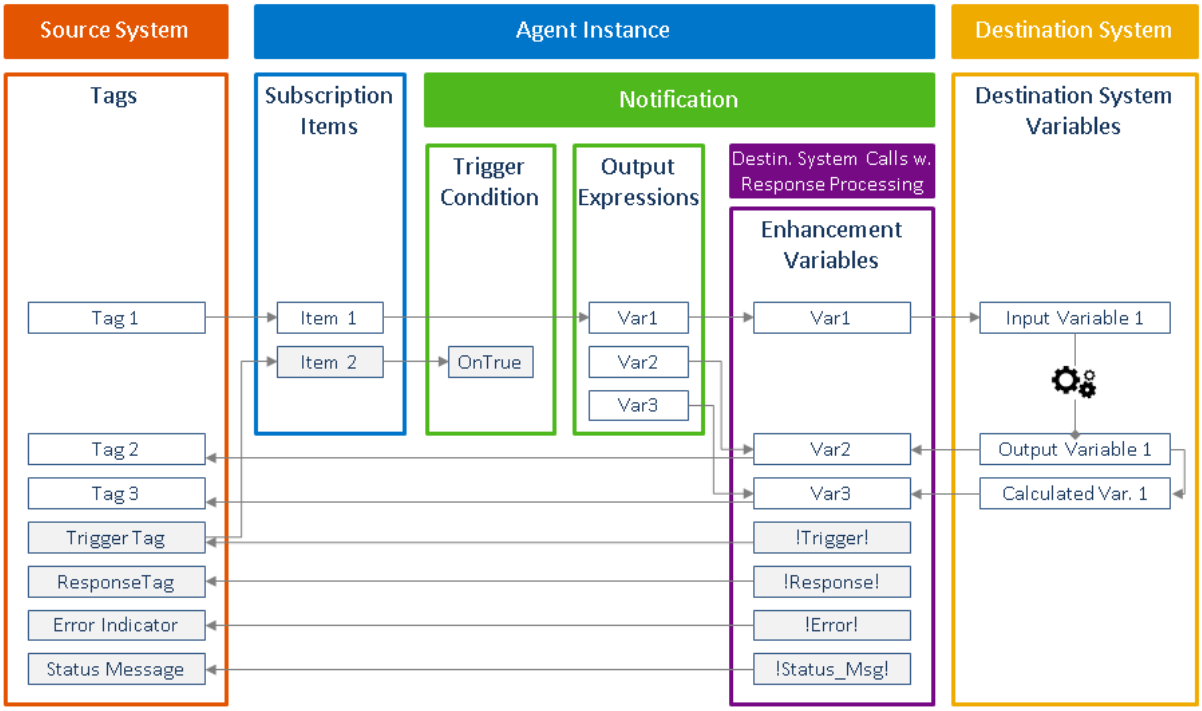
The special feature of the agent instance configuration is that when defining the output expressions for the notification, you define enhancement variables with the same name. In this way, you define both the variables for populating the input variables of the destination system as well as the variables for temporary storage of the output variables of a destination system.

Moreover, the SAP standard enhancement *Destination System Calls with Response Processing* offers four additional enhancement variables that are independent of the definition of the output expressions. The names of these variables start and end with an exclamation mark, for example, `!Response!`. With these special variables, the trigger tag can be reset, and information as to whether the destination system calls were successful can be written back to specific tags in the data source. Subsequent processes can react to the value change of these tags at the data source.

> **i Note**
>
> Unlike a customer-owned enhancement that you plan yourself, with *Destination System Calls with Response Processing* you do not have any implementation effort in an *integrated development environment (IDE)*, for example, *Microsoft Visual Studio*. You simply need to select the option *Destination System Calls with Response Processing* on the *Notification Processing* tab and then create a destination system for the notification enhancement. (See also: Enhanced Notification Processing [page 73].)

Use

The following graphic illustrates the functions of the SAP standard enhancement *Destination System Calls with Response Processing* with an example:



Destination System Calls with Response Processing

The graphic shows a destination system of the type *Web Service destination system* that is called using the tag values of a source system. The results of the Web service are then written back to the source system. Furthermore, the example contains a *calculated variable* in the Web service response message. (See also: Calculated Variables in a Web Service Response Message [page 402].)

The example shows which PCo configuration data is required and needs to be linked together:

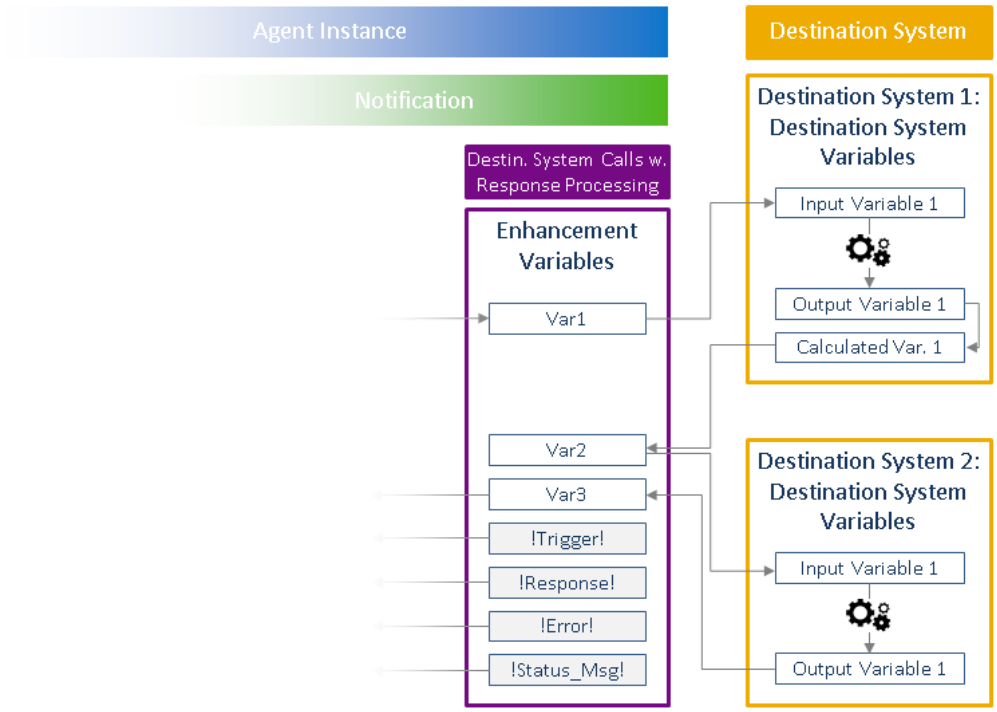
- You need to create subscription items and output expressions for the tags of the data source that serve to supply the Web service (subscription item `Elem. 1` and output expression `Var1` in the graphic shown above).
- For the *trigger tag* that is used in a trigger expression and that triggers the notification, you also create a subscription item (`Elem 2`).
- To be able to define the variables of the SAP standard enhancement, you define the output expressions (`Var2` and `Var3`). In the simplest case, you can generate them from the subscription items so that they have the same names as them.

At the runtime of the agent instance, the input variables of the Web service destination system are populated from the assigned variables of the standard enhancement (here: `Var1`). The output variables of the Web

service destination system are written back to the assigned variables of the standard enhancement. In the graphic above, these are the output expressions `Var2` and `Var3` that you have created without reference to a subscription item and have given an appropriate data type.

At the end of the destination system call, the standard enhancement writes the values of the enhancement variables back to the assigned tags of the data source. The trigger tag is also reset to then be able to trigger a new notification operation.

The standard enhancement *Destination System Calls with Response Processing* is able to write back the values of the enhancement variables to the source systems of up to three different agent instances. Moreover, the standard enhancement can call up to three destination systems one after the other for each notification message:



The output variables and the calculated variables of a destination system call, as in the example given above, can be stored temporarily in the enhancement variables (here: the output expressions `Var2` and `Var3`) and used for calling the next destination system. The SAP standard enhancement only writes the values of the enhancement variables back to the source system or source systems at the end of the call sequence.

> **i Note**
>
> The sequence of destination system calls and the times at which the values of the enhancement variables are written back to the source systems are, in the standard enhancement *destination system calls with response processing*, predefined and cannot be changed. Instead of the standard enhancement, you can also use a multiple call destination system [page 353] in conjunction with a query destination system [page 375] to be able to achieve greater flexibility in the process flow. Furthermore, the multiple call destination system provides conversion functions for output variables of any destination system type, whereas with the

standard enhancement *destination system calls with response processing* you can only convert output variables of Web service destination systems using calculated variables. The calculated variables only exist for the destination system type *Web service destination system*.

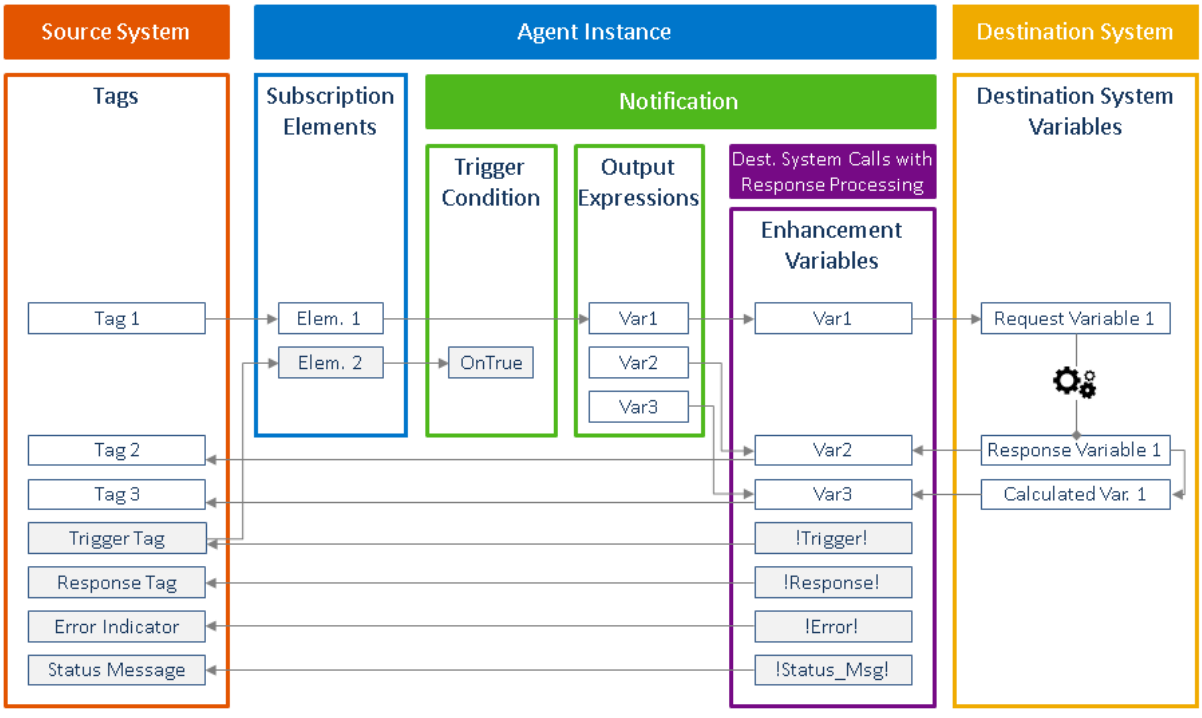
Activities

Configuring Destination System Calls with Response Processing [page 84]

3.6.3.1 Configuring Destination System Calls with Response Processing

Use

The following graphic gives an example of how the SAP standard enhancement *Destination System Calls with Response Processing* is included in enhanced notification processing and which objects are to be configured:



Procedure

To use the standard enhancement, do the following:

1. Open the *PCo Management Console* and create a **destination system**, for example, of the type *Web Service Destination system*. Configure the Web service as required. (See also: Server Settings Tab (WS Destination System) [page 394] and Operation Configuration Tab (WS Destination System) [page 396].)
2. Create an **agent instance** based on a source system that supports the notification process.
3. Choose the *Notification Processing* tab of your agent instance. Select the *Destination System Calls with Response Processing* option.
   Then create a destination system for the notification enhancement by choosing the *Create Destination System* function key. (See also: Enhanced Notification Processing [page 73].)
4. On the *Subscription Items* tab, create the subscription items for all source system tags that you need to populate the input parameters of the destination system call. (See also: Agent Instance: Subscription Items Tab [page 476].) Create a subscription item for the source system tag that is to be used later as a **trigger tag** in the trigger expression.
5. Create a **notification** and create output expressions on the *Output* tab. These output expressions define the enhancement variables simultaneously.
   First create **output expressions for the enhancement variables** that you need for populating the input parameters of the destination system call. As a rule, these output expressions use subscription items or constant values. Moreover, you need output expressions for the enhancement variables in which you want to temporarily store the results or interim results of destination system calls before writing the values back to the data sources. To do so, create empty output expressions that only consist of a name. For all output expressions, choose the appropriate data types that match the input and output parameters of the destination systems. For Web services, you can also use the calculated variables as output parameters.

   > **i Note**
   >
   > You do not need an output expression for the *trigger tag*.

6. On the *Notification* tab, define a *trigger expression* so that the destination system is only called if a specific source system tag (*trigger tag*) changes. The corresponding source system tag must be of the data type *Integer* and must be able to have the following values:
   - 0: The trigger is in the initial setting
   - 1: The notification process is to be triggered
   - 3: The parsing of the notification message has failed

   For that reason, define a trigger expression that triggers the notification as soon as the trigger tag has the value 1, for example, `'[subscription item of the trigger tag]' == 1` and also choose the trigger type *OnTrue*.
7. On the *Destinations* tab, choose the entry *Enhanced Notification Processing* as the destination and make the module and variable assignment.
8. Now make the following settings in the screen area *Agent and Destination System Call Modules*:
   1. Assign the agent instances - at whose data source you want to change the data tags after processing the notification message - to the modules *Agent*, *Agent_2*, and *Agent_3*. If you want to automatically reset the trigger tag after processing the notification message, you need to assign at least the agent instance of the trigger tag to a module here. If you do not assign any agent instance to a module, this module is not taken into account by the SAP standard enhancement.
   2. Assign your destination systems of the supported types to the modules *WS_Call*, *WS_Call_2*, and *WS_Call_3*. At the runtime of the agent instance, the destination systems are called up in exactly this order. You can also define exception handling for the destination system call modules. If you do not

assign any destination system to a module, this module is not taken into account by the SAP standard enhancement.

3. In the *Assignment of Enhancement Variables* screen area, on the *Destination System Variables* tab, you link the *request and response variables* of the destination system with the enhancement variables. In the case of a Web service, the calculated variables can also be linked here. Choose the *Propose Assignment* pushbutton if you want the system to automatically suggest the variable assignment. The prerequisite for this is that the enhancement variables have the same name as a destination system variable and the variable type is also compatible with the variable type of the Web service variables.

   > **i Note**
   >
   > You need to assign exactly one enhancement variable to each request variables of a destination system. On the other hand, you do not need to assign an enhancement variable to each response variable and to each calculated variable.

4. In the table on the *Source System Tags* tab, link the *source system tags* to the *enhancement variables*. In the selection list for the enhancement variables, you see the output expressions that match the data type of the source system tags and some special variables. These special variables are preceded and followed by exclamation marks:
   - *!Trigger!*
     Assign the trigger tag to these enhancement variables. The trigger tag is set to the initial value `0` before the first destination system is called. If there is an error, if parsing of the notification message has failed, it is set to the value `3`.
   - *!Response!*
     At the end of *Destination System Calls with Response Processing*, this enhancement variable is set to the value `1`. It shows the follow-on processes that the processing of a notification message is completed by the SAP standard enhancement.
   - *!ERROR!*
     This enhancement variable shows whether the destination system calls were successful (value is 0) or contained errors (value is 1) after the processing of a notification message by the SAP standard enhancement.
   - *!STATUS_MSG!*
     This enhancement variable contains a possible error message after the SAP standard enhancement has run.

   (See also: Assignment of Modules and Variables (ENP) [page 75].)

9. Start the **agent instance**.
   A value change to the trigger tag at the source system from *0* to *1* calls up the destination system. You can see the results of the destination system calls at the source system in the relevant tags that you assigned to the *response variables* and special *enhancement variables*. The trigger tag is set to the initial value *0* after successful execution or, if there are errors, to the value *3*.

3.7 Connection of Web Applications via WebSocket

Use

The WebSocket network protocol, in accordance with specification RFC6455, allows a high-performance, network-resource-efficient communication between a WebSocket server and a Web application configured as a WebSocket client.

Against this backdrop, PCo can be set up as a WebSocket server and, in particular, support **notification scenarios** for which the events provided by the source system are displayed directly on modern user interfaces.

> ✤ Example
>
> For example, you can use *SAP MII*'s *Self Service Composition Environment* as a user interface. The *Self Service Composition Environment* is a Web-based collaborative environment that you can use to configure and display content provided by PCo, for example. For more information, see the *SAP MII Documentation* under *SAP MII Self Service Composition Environment*.

The WebSocket has an exceptional position among the various PCo destination system types because the query and notification scenarios are technically combined. On the PCo side, a specific port is envisaged through which the bi-directional communication between PCo and the Web application takes place.

You make the connection settings for the WebSocket in the agent instance on the *Servers* tab. You also create the destination system here. On the *Destinations* tab for the notification, you need to select this destination system and make the variable assignment.

More Information

WebSocket Server [page 438]

Settings for the Notification Process with WebSocket [page 441]

3.8 Connection of External Data Sources to Bus. Suite Applications

Concept

SAP Plant Connectivity (PCo) easily facilitates the communication between *SAP Business Suite* applications and external data sources such as weighing systems, production machines, OPC servers, or process control systems. PCo agents and agent instances [page 412] are used to exchange data between the external data sources and PCo.

You can use PCo to transfer shop floor data directly to a Business Suite system such as SAP ERP, where you can process the data further in an application of your choice. PCo thus closes the gap between the control systems of the automation level and the software-supported production planning, for example, in SAP ERP.

PCo is integrated with Business Suite applications through an ABAP-based integration layer that is part of *SAP NetWeaver*.

The data transfer between the Business Suite systems and the PCo agents is based on RFC function calls of the type TCP/IP. Each PCo agent is assigned to a corresponding *RFC Destination* in the Business Suite system beforehand. Business Suite applications that want to exchange data with PCo agents run *SAP NetWeaver* methods for this purpose.

Prerequisites

The following technical prerequisites are necessary to be able to use Plant Connectivity (PCo) to include external data sources:

- *SAP Plant Connectivity* as of Version 2.2
  - Microsoft .Net framework as of Version 3.5, SP 1
- An SAP Business Suite system with SAP NetWeaver Application Server ABAP, as follows:
  - NW ABAP Release 7.31
  - NW ABAP Release 7.02, as of Support Package SAPKB70208
  - NW ABAP Release 7.01, as of Support Package SAPKB70109
  - NW ABAP Release 7.00, as of Support Package SAPKB70024
- TCP/IP communication options between PCo and SAP Business Suite system

Depending on the type of external data sources used, OPC data servers also need to be installed and configured.

Features

PCo provides two methods for Business Suite applications and external systems to exchange data:

- Using PCo queries
  PCo queries are queries that can access data from the external data source directly, starting from the Business Suite applications. You can use these queries to change the data in the data source directly. See also: Data Exchange Using Queries [page 89]
- By sending PCo notifications
  The notifications are triggered by the external data source. PCo sends the notification messages to the Business Suite applications for further processing. See also: Generation and Processing of Notifications [page 94]

More Information

For more information and sample implementations on integrating PCo with the SAP Business Suite, see SAP Note 1576651.

3.8.1 Data Exchange Using Queries

Use

Business Suite applications can use PCo to send queries to external data sources, for example, to read or change values of data source tags. The PCo agent instance, which is connected to the external data source, receives the query and interprets the statements it contains. PCo provides three types of query:

- Tag query
  These queries that refer to tags.
- Database query
  These queries are SQL queries that refer to database content. The SQL dialect used corresponds to the connected data source.
- Text query
  These queries are text-type queries with text-type content that you can define freely.

The results of the queries support certain data types, see: SQL Data Types [page 122]

More Information

Process Steps for the Data Exchange Using Queries [page 89]

3.8.1.1 Process Settings for the Data Exchange Using Queries

Use

The following steps are necessary for running and evaluating queries.

Process

Step	System	Activity
1	Business Suite system	**Defining an RFC destination (transaction SM59):** • You create an RFC connection of type TCP/IP (T) with the activation type *Registered Server Program*. See also: Defining an RFC Destination [page 91]. • You define the gateway options. • You can choose whether you define the data for secure network communication (SNC). See also: SNC Settings [page 92]
2	Business Suite system	**Assigning authorizations** You need to assign the necessary authorizations to the *User* that is used to exchange RFC data between the Business Suite system and PCo. The *Authorization Object* S_PCO_INT is available for this purpose. You can assign the authorization object to the user role. **i Note** You can display the authorization object and its documentation in the Business Suite system in transaction SE80 in the package S_PCO. Alternatively, the following roles for integration in transaction PFCG are available as templates that you can assign to the user: • SAP_BC_SRV_PCO_BS_INT_ADMIN • SAP_BC_SRV_PCO_BS_INT_USER
3	PCo	**Adding a source system** You select the *Source System Type* (for example, *OPC DA Agent*) and enter the server connection data.
4	PCo	**Adding a destination system** You select the destination system type *RFC Destination* and maintain the RFC client data: • Application server and additional connection data for associated SAP Business Suite system • *Type SAP NW* For more information, see RFC Destination System [page 347].

Step	System	Activity
5	PCo	**Adding an agent instance** You create the agent instance for the previously created source and destination system. • On the *Host* tab of the agent instance, enter the *Service User Name* and the *Service User Password* to run the PCo agent instance as an *MS Windows Service*. See also: Host Tab [page 413] • On the *Query Ports* tab, maintain the settings for the port type *SAP NW RFC Server*: You enter the RFC server settings. You enter the name of the registered server program that was defined in the RFC connection. You assign the destination system that provides the information for the repository structure of the *.Net Connector*. See also: Servers Tab (SAP NW RFC Server) [page 430]
6	Business Suite system	**Implementing method calls for the ABAP wrapper classes of package S_PCO** • You assign the name for the application handle that is transferred to PCo. • You create the method calls in suitable positions in the code of Business Suite applications (for example, user exit, BAdI) • For sample implementations, see 1576651.
7	PCo	**Starting an agent instance**
8	Business Suite system	**Running a Business Suite application**

3.8.1.1.1 Defining an RFC Destination

Context

In the ERP or SAP S/4HANA system, you need to use transaction SM59 to define the RFC destination for a PCo agent instance. The communication between a Business Suite system and PCo is established using *Remote Function Calls* (RFC modules).

You need to create exactly one RFC destination for each PCo agent instance.

Procedure

1. Make the following general settings in transaction SM59:
   - RFC Destination: This can be defined freely
   - Connection Type: **T** (TCP/IP connection)
2. Make the following entries on the *Technical Settings* tab:
   - Select *Registered Server Program*.
   - *Program ID*: Here you enter the value that you specified in the *Program ID* field in the PCo agent instance on the *Query Ports* tab.
   - *Gateway Host*: Here you enter the value that you specified in the *SAP Gateway Host* field in the PCo agent instance on the *Query Ports* tab.
   - *Gateway Service*: Here you adopt the value that you specified on the *Query Ports* tab.
   - You can use the *Test Connection* pushbutton to check the connection to PCo.
3. On the *Logon & Security* tab, you can enter the following if you want to implement the Secure Network Communication (SNC) component:
   - Select the *Active* radio button in the *Security Options* screen area.
   - Choose the *SNC* pushbutton. The *Change SNC Extension View: Details* dialog box appears.
   - In the *QoP* (Quality of Protection) field, enter the value that you set in PCo in this field on the *Query Ports* tab.
   - In the *Partner* field, enter the SNC name of the user, for example, `p:CN=D0XXXXX, O=SAP-AG, C=DE`. This name is in *User Maintenance* (transaction SU01) on the *SNC* tab.
     You also need to enter this name in the PCo Console on the *Query Ports* tab in the *User SNC Name* field.
   - For more information about SNC settings, see SNC Settings [page 92].

Results

You can now create the following data in the *PCo Management Console*:

- You create a *source system*.
- You create an *RFC destination system* of the type *SAP ODA*. See also: RFC Destination System [page 347]
- You create the *agent instance* and enter the RFC connection data on the *Servers* tab. See also: Servers Tab [page 419]

3.8.1.1.2 SNC Settings

This example describes the settings required if you want to use the **Secure Network Communications (SNC)** component for communication between the PCo Management Console and the connected Business Suite system.

This example describes only the **procedure in the internal SAP environment** when using *Single Sign-On (SSO)*.

> **i Note**
>
> You have installed SSO on the local physical machine. If SSO has not yet been installed, you can use *Run Advertised Programs* to find it.
>
> After SSO is installed, a certificate is automatically created for the current user.
>
> You can access the library for SSO under C:\Program\Files\SECUDE\Office\Security\secude\dll.

For information about the **general procedure** for SNC, see the **SNC User's Guide** under *http://service.sap.com/security* > *Security in Detail* > *Secure User Access* > *Authentication & Single Sign-On*.

SNC Settings for the RFC Server (SM59)

1. In transaction SM59 in the connected Business Suite system, select the *RFC Destination* (type TCP/IP connection) that you created previously. See also: Defining an RFC Destination [page 91]
2. Click on the *Logon & Security* tab.
3. Choose the *SNC* pushbutton.
   The *Change SNC Extension View: Details* dialog box appears.
4. In the *Partner* field, enter the SNC name of the user, for example, `p:CN=D0XXXXX, O=SAP-AG, C=DE`.

   > **i Note**
   >
   > This name is in *User Maintenance* (transaction SU01) on the *SNC* tab.
5. If required, enter a value in the *QoP* field and save your entries.
6. On the *Logon & Security* tab, select the *Active* option to activate SNC.

SNC Settings in the PCo Agent Instance (Query Ports Tab)

1. Click on the *Servers* tab of your agent instance.
2. Select the RFC server, for example, *NW RFC Server*.
3. In the *SNC* screen area, select the *Enabled* checkbox.
4. In the *User SNC Name* field, enter the SNC name for the user that you entered in the *Partner* field for your RFC destination in transaction SM59, for example, `p:CN=D0XXXXX, O=SAP-AG, C=DE`.
   . The entry in this field is mandatory.
5. In the *Server SNC Name* field, enter the SNC name of the Business Suite system, for example, `p:CN=UI3, O=SAP-AG, C=DE`. This entry is optional.
6. If the SNC library path differs from the path above, you can enter the path in the *Library* field (optional).

SNC Settings in the RFC Destination (Destination System)

1. In the *PCo Management Console*, click on the destination system (type *RFC Client Destination*) that you created for the Business Suite system.
2. Click the *RFC Client Settings* tab.
3. In the *SNC Settings* screen area, select the *Enabled* checkbox.
4. In the *User SNC Name* field, enter the SNC name that you entered in the *Partner* field for your RFC destination in transaction SM59, for example, `p:CN=D0XXXXX, O=SAP-AG, C=DE`. The entry in this field is mandatory.
5. In the *Server SNC Name* field, enter the SNC name of the Business Suite system, for example, `p:CN=UI3, O=SAP-AG, C=DE`. This entry is mandatory.
6. If the SNC library path differs from the path above, you can enter the path in the *Library* field (optional).
7. The user name and password are no longer required as logon information. The system uses the SSO user certificate.

3.8.2 Generation and Processing of Notifications

In some business processes it is useful to trigger SAP Business Suite activities when certain values or parameters of external data sources change. The following table provides some examples:

Business Process	Business Suite Activities
The state of a machine changes from "In Process" to "Malfunction"	• Automatic generation of a maintenance notification • Status change in the equipment master data record
The value of a temperature sensor exceeds the previously defined maximum temperature of 120°C (248°F)	Alarm message to the shift supervisor via e-mail or SMS
The weight of an intermediate product is determined by individual weighing	• Automatic generation of a time ticket confirmation • Order release of repeat order • Creation of a production inspection lot

In the examples mentioned above, specific Business Suite applications need to be addressed automatically. They receive the necessary data and information for the tags via notifications that PCo generates and sends as soon as the defined trigger conditions are met. There are two options for this:

- **Remote Subscription**
  With this technique you maintain the subscription to tags in the Business Suite system. The following applies:
  - The relevant tags are selected through subscriptions made by the Business Suite applications.

    > ⚠ Caution
    >
    > Expired subscriptions or changes to subscriptions in the Business Suite system are not displayed automatically in the *PCo Management Console*. To update the display in the *PCo Management Console*, you need to close and reopen the *PCo Management Console*.

- The processing of PCo notifications that are sent back to the Business Suite can be controlled using BAdI implementations on a case-by-case basis and in detail.

The advantage of remote subscription is that the business context can be assigned in detail (for example, up to the level of the individual production order) using the application handle [page 100].
For more information about the process steps in this technique, see Process Steps for Remote Subscription [page 95].

- **Subscription in the PCo Console**
  With this technique you can maintain the subscription of tags in the PCo Console directly. This technique is the same as the notification process with SAP MII.
  The processing of PCo notifications that are sent back to the Business Suite can be controlled using BAdI implementations on a case-by-case basis and in detail.
  For more information about the process steps in this technique, see Process Steps for the Subscription in the PCo Console [page 98].

> **i** Note
>
> The application handle is defined automatically by PCo. PCO_NOTIFY is always specified as the application (see Application Handle [page 100]).

3.8.2.1 Process Steps for Remote Subscription

Use

The following describes the steps for creating a remote subscription and for generating and processing notifications.

Process

Step	System	Activity
1	Business Suite system	**Defining an RFC destination (transaction SM59):** • You create an RFC connection of type *TCP/IP (T)* with the activation type *Registered Server Program*. See also: Defining an RFC Destination [page 91]. • You define the gateway options. • You can choose whether you define the data for secure network communication (SNC). See also: SNC Settings [page 92]

Step	System	Activity
2	Business Suite system	**Assigning authorizations** You need to assign the necessary authorizations to the *User* that is used to exchange RFC data between the Business Suite system and PCo. The *Authorization Object* S_PCO_INT is available for this purpose. You can assign the authorization object to the user role. **i Note** You can display the authorization object and its documentation in the Business Suite system in transaction SE80 in the package S_PCO. Alternatively, the following roles for integration in transaction PFCG are available as templates that you can assign to the user: • SAP_BC_SRV_PCO_BS_INT_ADMIN • SAP_BC_SRV_PCO_BS_INT_USER
3	PCo	**Adding a source system** You select the *Source System Type* (for example, *OPC DA Agent*) and enter the server connection data.
4	PCo	**Adding a destination system** You select the destination system type *RFC Destination* and define the RFC client data: • Application server and additional connection data for associated SAP Business Suite system • *Type SAP NW* For more information, see RFC Destination System [page 347].
5	PCo	**Adding an agent instance** You create the agent instance for the previously created source and destination system. • On the *Host* tab of the agent instance, enter the *Service User Name* and the *Service User Password* to run the PCo agent instance as an *MS Windows Service*. See also: Host Tab [page 413] • On the *Servers* tab, maintain the settings for the port type *SAP NW RFC Server*: You enter the RFC server settings. You enter the name of the registered server program that you have defined in the RFC connection data. You assign the destination system that provides the information for the repository structure of the *.Net Connector*. For more information, see SAP NW RFC Server [page 430].

Step	System	Activity

5	PCo	**Creating a notification template**
		
		You create a notification template [page 503] for the agent instance:
		
		• *Trigger* tab
		You select the trigger type *Always*.
		
		> **i Note**
		> Trigger type *Always* is the preferred trigger type for Business Suite integration scenarios. The other trigger types can only be used insufficiently for Business Suite integration scenarios.
		
		• On the *Remote Subscription* tab, you define the *Lifetime* for the associated tag subscription. The default value for the lifetime is an hour (3600 seconds). See also: Notification Template: Remote Subscription Tab [page 505]
		• On the *Reliable Connection* tab, you define the settings for reliable connections and reliable message delivery:
		  ○ Number of connection attempts if the PCo connection to the external data source is lost
		  ○ Lifetime of notification messages that could not be delivered. The default value for the lifetime is one day.
		• On the *Destinations* tab, you assign the PCo destination system for your Business Suite system, which is to receive and process the notification message. (See also: Destinations Tab [page 520].)

6	Business Suite system	**Creating classes** You create classes that are to process the notifications (transaction SE24). These classes inherit characteristics and functions from the superclass **CL_PCO_NOTIF_HANDLER** and overwrite the method **EXECUTE** with suitable implementations.
7	Business Suite system	**Implementing a BAdI** You implement the BAdI **BADI_S_PCO_HANDLE_NOTIF** (transaction SE19 or SE80 [package S_PCO, enhancement spot S_PCO_ES_BADI_HANDLE_NOTIF]): Depending on the BAdI filter value in the application of the *Application Handle*, the name of the previously created class that is to process the notification is returned.
8	Business Suite system	**Implementing method calls** While the subscription of tags is performed by the Business Suite application, the following steps also need to be performed to implement the method calls for the ABAP wrapper classes of package S_PCO: 1. Naming of the application handle that is transferred to PCo 2. Creating method calls in suitable sections of code (user exit, BAdI, or similar) of the Business Suite applications. For sample implementations, see the end of this guide in SAP Note 1576651.

Step	System	Activity
9	PCo	**Starting an agent instance**
10	Business Suite system	**Subscribing to tags** Subscribing to tags for which notifications are to be sent, provided that the trigger conditions have been met
11	Business Suite system	1. **Defining an application handle** You define an application handle to which the notification is assigned. When the notification is received, the Business Suite system can use the application handle to control how the notification is to be processed further by Business Suite applications. **i Note** The **test program RPCO_BS_INT_TEST** uses the **application SAPTESTING** and the **handle LOG** as default values An active BAdI implementation (BAdI_S_PCO_HANDLE_NOTIF) exists for the filter value of the **application SAPTESTING**. This BAdI implementation can be used to format and save the contents of the notification as application log messages. 2. **Assigning a notification template** In the code, you assign the notification templates that you created previously for the agent instance.
12	PCo	**PCo sends a notification** If the defined trigger conditions have been met, PCo uses an RFC call to send the notification as an XML document to the associated recipient in the Business Suite. • The function module S_PCO_RFC_EXECUTE_RECEIVE_EVNT is run in which the notification is processed: • It is first checked whether there is a class that processes the notification for the application that was defined in the application handle of the notification. • If this is the case, an instance of this class is generated and the EXECUTE instance method of the class is run.

3.8.2.2 Process Steps for the Subscription in the PCo Console

Use

The following section shows the steps that are necessary for the notification process with the subscription of tags directly in the PCo console.

Process

Step	System	Activity
1	Business Suite system	**Assigning authorizations** You need to assign the necessary authorizations to the *User* that is used to exchange RFC data between the Business Suite system and PCo. The *Authorization Object* S_PCO_INT is available for this purpose. You can assign the authorization object to the user role. **i Note** You can display the authorization object and its documentation in the Business Suite system in transaction SE80 in the package S_PCO. Alternatively, the following roles for integration in transaction PFCG are available as templates that you can assign to the user: • SAP_BC_SRV_PCO_BS_INT_ADMIN • SAP_BC_SRV_PCO_BS_INT_USER
2	PCo	**Adding a source system** You select the *Source System Type* (for example, *OPC DA Agent*) and enter the server connection data.
3	PCo	**Adding a destination system** You select the destination system type *RFC destination system* and maintain the RFC client data: • Application server and additional connection data for associated SAP Business Suite system • Type *SAP NW* For more information, see RFC Destination System [page 347].
4	PCo	**Adding an agent instance** You create the agent instance for the previously created source and destination system. • On the *Host* tab of the agent instance, enter the *service user name* and the *service user password* to run the PCo agent instance as a *Windows service*. See also: Host Tab [page 413] • On the *Subscription Items* tab, you select the tag to which you want to subscribe. See: Subscription Items Tab [page 476] • **No entries** are required on the *Servers* tab. Ensure that no server type is selected.

Step	System	Activity
5	PCo	**Creating a notification** • On the *Notification* tab the *Enabled* checkbox and the trigger type *Always* must be selected. • On the *Output* tab, you need to enter a subscription item, meaning the tag whose value is to be transferred to the notification. **Example** You have selected the tag **`TAG1`** on the Subscription Items tab. You now need to specify the following on the *Output* tab: *Name*: can be defined freely, for example, **`Tag1`** *Expression*: **`'TAG1'`** • On the *Destinations* tab, you need to specify the destination system created previously. See also: Destinations Tab [page 520]
6	PCo	**Starting an agent instance** You start the agent instance. PCo sends notification messages to the connected Business Suite system. PCo assigns an application handle [page 100] automatically.

3.8.2.3 Application Handle

Definition

A system-wide unique ID for an application in a Business Suite system.

Use

An *Application Handle* is used to assist communication between the Business Suite applications and PCo. When the queries are run, this application handle is transferred to PCo.

Integration

Remote Subscription

If you perform the subscription in the Business Suite system, each Business Suite application defines its own *Application Handle*. When the queries are run, this application handle is transferred to PCo.

The application handle is composed of two different parts:

- **APPLICATION**: Freely definable, 10-digit name for the application, which triggers the query or is to process the notification of the value change at a later point, for example:
  - PRODORDCON (production order confirmation)
  - MAINTMSGCR (creation of maintenance messages)
  - WEIGHINFO1 (Information about the weighing process)
- **HANDLE**: Freely definable content (a maximum of 80 characters) that can contain additional information about the application context, for example:
  - Order confirmation: Plant, order type, confirmation number
  - Maintenance message: Plant, equipment ID
  - Weighing process: Location, equipment ID, type of scales

When value changes of tags are subscribed to, the application handle is also transferred:

- PCo uses the application handle to group the subscribed tags.
- If notifications are created for this subscription, they contain the application handle of the related subscription and more.
- Application handle information can be evaluated specifically when the notification is processed.

> **Example**
>
> An application handle that contains the unique number of the order confirmation permits the related order data and operation data to be read.

Subscription in the PCo Console

For a subscription in the PCo Console directly, PCo automatically specifies the application handle:

APPLICATION: PCO_NOTIFY (fixed entry)

HANDLE: The **name of the notification** that you defined when you created the notification is stated here.

3.8.3 Test Program RPCO_BS_INT_TEST

Use

The test program RPCO_BS_INT_TEST facilitates a basic understanding of the communication between PCo and the SAP Business Suite applications. This program shows how easy it is to make the necessary implementations in the SAP Business Suite applications so that external data sources can be connected to the SAP Business Suite using PCo.

The test program focuses on the exchange of data between the SAP Business Suite and external data sources that have read and write access to tags. The following default values are predefined for the application handle for the test program:

- *Application*: SAPTESTING
- *Application Handle*: LOG

You can overwrite the default values of the test program with your own values as required.

If you want to use the test program to run the sample implementation for processing notifications, you must transfer the value **SAPTESTING** for the application (APPLICATION) of the application handle.

Features

The test program contains all functions that the standard SAP system provides for integrating external data sources using tag access:

- Displaying the supported characteristics of PCo agent instances (PCo agent features)
- Browsing namespaces of tag hierarchies
- Reading tags
- Writing value changes of tags
- Subscribing to tags that are to be used for notifications
- Displaying information about subscribed tags
- Extending subscriptions
- Deleting subscriptions
- Displaying notification templates
- Displaying application logs that were generated for the log object S_PCO

More Information

Initial Screen [page 102]

Results List [page 106]

3.8.3.1 Initial Screen

The following selection options are available on the initial screen of the test report:

- Parameters for Communication Between Business Suite and PCo Agents [page 103]
- Parameters for Processing Tags [page 104]
- Parameter for ALV Results Display
  User-specific ALV display variants can be maintained for displaying the results list. You can use the F4 help for the input field to select these variants.

The following function keys are available on the initial screen:

Function Keys on the Initial Screen [page 103]

3.8.3.1.1 Function Keys on the Initial Screen

The following function keys are displayed on the initial screen of the test program:

Function Key	Meaning
Execute	The test report is run according to the selection conditions maintained for the initial screen.
Get Variant	Previously created report variants can be loaded. The selection conditions for the initial screen were saved as a report variant previously (Save pushbutton).
Display Log	Jumps to the selection of application logs that were generated and saved as part of the Business Suite integration of Business Suite applications
PCo Agent Characteristics	Displaying the characteristics of the PCo agent instance that is connected to the Business Suite system via the chosen RFC connection, for example, read, write, subscribe, support native filters.
RFC Destination Characteristics	Enables the jump to the detailed view of the RFC connection maintenance for the selected RFC connection (transaction SM59)

3.8.3.1.2 Parameters for Communication Between Bus. Suite and PCo Agent

The following parameters are of significance for the communication between the Business Suite and the PCo agent:

Parameter	Meaning
RFC Connection of PCo Agent Instance	The F4 input help for the field lists all RFC connections of the *Type TCP/IP* that are available in the system.
RFC Connection Test	If you select the checkbox, when the object instances are generated the system performs an RFC connection test for the PCo agent that is to be called via the RFC connection. The result of the connection test is issued as a status message.
Application	The field content is interpreted as the name of the application for the application handle. It must have 10 characters.

Application Handle	The field content is interpreted as the handle content of the application handle. • The field content must have at least one character. • You can maintain a maximum of 80 characters as handle information.
Logging Active	If you select the checkbox, application logs are generated and saved automatically when test report functions are run.
Load Features Automatically	If you select this checkbox, the characteristics of the associated PCo agent instance are loaded and buffered when the object instances are generated.
Test Mode (Without PCo)	If you select this checkbox, the test report runs in test mode and uses hard-coded data to display results. • There is no communication with PCo. • Test mode is used mainly to simulate the interfaces.

3.8.3.1.3 Parameters for Processing Tags

The following parameters of the test program are relevant for processing tags:

- Checkbox **Namespace Browsing Mode**
  This checkbox has three modes:
  - *Preselection Using Tag Hierarchy*
    In this mode the tag hierarchy of the connected external data source is displayed as a tree structure. If you select individual hierarchy tags, the associated value of the tag, its fully qualified path details, and additional details are determined while the test report is running and are displayed in the results list. If you select a group folder, the system determines all tags (child nodes) that have the group folder as parent nodes. When the program is then run, the system determines additional detailed information on the tags and displays this in the results list.
  - *Preselection by Specifying Qualified Path ID (F4 Help)*
    If you select this mode, an additional input field appears on the initial screen for the fully qualified path. Proceed as follows:
    1. After you have chosen the F4 help for the input field, the tag hierarchy is displayed as a tree structure.
    2. You can select only one entry (tag or tag group).
    3. After you have made a selection, the input field receives the fully qualified path of the tag or tag group.
    4. Now run the program. The system reads additional data for the tag of the external data source and displays this in the results list.
       If you have selected a tag group, the system determines the tags that were assigned to the tag group directly as child nodes. If there are tags, the associated detailed information is displayed in the results list.
  - *Preselection Using Filter Value for Tags*

After you have selected this mode, you have two additional input fields available:

- Input field *Masking*:
  You can use the F4 help to select the *Filter Type* to be used to search for tags. The PCo settings for the associated agent instance determine whether the search is supported by the chosen filter type. The following filter types are available:
  - Filter type NATIVE:
    This filter uses the filter options of the connected data source. This filter access is usually of high performance. (See Filter Functions [page 266])
  - Filter type REGEX:
    This filter uses regular expressions (see Filter Functions [page 266]).
  - Filter type LEGACY:
    This filter uses the old filter logic of the PCo predecessor function UDS (Universal Data Sources) that SAP MII currently still uses to connect external data sources.
- Input field *Filter String for Tag Alias*:
  You can use placeholders (*) to search for the name or alias of tags. If the system finds tags, the detailed information is displayed in the results list when the test program is run.
  The following functional restrictions apply for using filters to determine tags:
  - No tags were found for which the search string is only part of the fully qualified path.

    > **Example**
    >
    > Searching for *Tag_A* provides the tag Root/Temperatur/TempTag_A, but not the tag Root/Tag_A/Temp_A

  - The fully qualified path information cannot be determined for these tags. However, the fully qualified path of tags is needed to subscribe to tags. Therefore, subscriptions are not possible. In the test program, the function keys for *Subscription*, *Extension of Subscriptions*, and *Deletion of Subscriptions* are therefore hidden in the ALV results display. In addition, the values in the column *Tag Alias (Abbreviated)* cannot be changed.

- Input field **Maximum Value (MaxRows)**
  This input parameter is used to restrict the number of the determined groups and tags. If a tag group was selected and the maximum number is restricted to `6`, the system reads the first six subgroups as well as the first six tags that are the child nodes for the selected tag group. If there are more tags or tag groups than were selected restrictedly, a relevant message is issued.

- Input field **Number of Decimals (Floating Point)**
  For floating point numbers the decimal display is chosen as the preference. A value of `0` or `.` means that the floating point numbers are displayed without restricting the number of decimal places. Other values restrict the number of decimals and lead to rounded floating point numbers. Through the internal display of decimal places of a floating point number of **ABAP data type f** through dual fractions, there is not an exact equivalent for each number that can be displayed in the decimal system, whereby there may be rounding errors for assignments and in intermediate results of calculations. These can only be avoided by using two-stage rounding.

3.8.3.2 Results List

Use

If tags were found for the selection conditions, the systems reads additional data for the tags and displays the results as a table overview (ALV grid).

Features

Displayed Information on Tags

The following data is displayed in the results list:

Field	Meaning
Tag Alias (Abbreviated)	States the name of the tag or the alias name of the tag (*Tag Alias*). The alias name can be changed when subscribing. While the associated subscription exists, the alias name is always displayed. Otherwise, the system issues the name of the tag.
Qualified Tag ID	States the entire path for the tag. The fully qualified path is displayed in the internal PCo format (using / as the separator): To be able to communicate with PCo, fully qualified paths of tags always need to be transferred to PCo in the internal PCo format.
Subscription (Icons)	Displays whether there are subscriptions for the tag.
Tag Value	The tag value is displayed as an abbreviation (maximum of 40 characters). • The cell is ready for input. • Tag values that are not valid are identified by the character string ---.
Value Change	Icon that states whether the value of the tag has been changed. To be able to display the values of the selected tags you then need to perform the function for writing the value change.
Short Text	Short text for the SQL data type of the tag (see SQL Data Types [page 122])

Timestamp	Timestamp in UTC format
Date and Time	Date and time of the tag

Function Keys

The following function keys are available:

Function Key	Meaning
Display Agent Properties	The system displays a dialog box that displays all functions available for the PCo agent.
Application Logs	The system displays the application logs that are used for the object S_PCO.
Display Subscriptions	Dialog box display of information about subscriptions that are available for the defined application handle: • Name of subscription • Name or alias of tag • Fully qualified path of tag in native display **Example** In contrast to the internal PCo display of path information, the native display of path information depends on the data source, for example, the OPC server or the assigned PCo agent instance. The OPC standard does not specify which separator indicator is to be used. The individual elements of the path information are often separated by a period. Example: Path details for tag "Temperatur_1" of tag group "TempSensor", which hangs directly on the root node 'Root' of the tag hierarchy: ○ **Internal PCo display:** Root/TempSensor/Temperatur_1 ○ **Native display:** Root.TempSensor.Temperatur_1
Notification Templates	The system shows a dialog box in which the name and description of the notification templates that are maintained for the selected PCo agent instance are displayed.

Subscribe	You can select the tags of the results list for which notifications are to be generated when the values of the tags change. • If you choose the *Subscribe* function key, the system displays the available notification templates that are maintained for the PCo agent. • After you have selected a notification query, the subscription is performed. • Tags for which subscriptions were performed successfully have a cube icon in the *Subscr.* column of the results list.
Delete Subscriptions	You can use this function key to remove subscriptions: • If you have not selected any entries in the results list, the system deletes all subscriptions that exist for the application handle specified. • If you have selected individual entries in the results list for which there are subscriptions, the system deletes these tags from the subscriptions.
Extend Subscriptions	You can use this function key to extend the validity period for all subscriptions of the application handle. The validity area is extended by the start of the validity period being set to the values for the current date and current time.
Write Values for Tag	You can use this function key to forward the changed values of the tags (icons in the column *Value Change*) to the external data source that is connected to the Business Suite via the PCo agent instance.
Update Display	You can use this function key to reload the detailed information for the selected tags.

3.8.3.3 F4 Input Help for Browsing for Namespaces

Use

The Business Suite applications require a simple option for the targeted processing of tags and tag groups of the external data source.

The elementary search help S_PCO_ELM_BROWSE_TAG in package S_PCO is a sample template for selecting a tag or a tag group. The namespace of tags is displayed in a dialog box. After the tag or tag group has been selected, the search help returns the fully qualified path of the selected object.

Features

The following search help parameters are available:

Search Help Parameter	Meaning
MAX_ENTRIES	Entry of the maximum value for selecting tags and tag groups. The entered number must have a value > 0.
RFC_DEST	Entry of the RFC connection for the selected PCo agent instance
SIMUL_MODE	Running the search help in simulation mode (X), whereby there is no communication with the PCo agent instance, hard-coded data is accessed instead.
TAG_ID	Output field for the first 255 characters of the fully qualified path for the selected object in the internal PCo format (with / as the separator)
TAG_DESCR	Output field for the description of the tag: • Is only filled with values in simulation mode • Currently, the description of the tag cannot be determined in productive mode: This PCo metadata object is not returned to the calling Business Suite application when the PCo query is run.
NODE_TEXT	Output field for the name of the tag or tag group
IS_GROUP	Output field for the indicator that states whether the selected object is a tag or tag group

Due to the width of 464 characters it is not possible to create a personal input help for the search help. The output of the fully qualified path must also be limited to a maximum of 255 characters due to the restrictions for the output of character strings in search helps/screens.

> **i** Note
>
> If you want to avoid the last-mentioned restriction, use the function module S_PCO_CALL_POPUP_NAMESP_BROWS (in package S_PCO, function group S_PCO) instead of the search help.

3.8.3.4 Error Handling

If errors occur in PCo during the processing of PCo queries, the PCo agent instance transfers relevant messages (PCo user messages) to the calling Business Suite application and triggers a class-based exception (class CX_PCO_BS_INT). The calling applications intercepts this exception.

PCo messages are stored in the table attribute PCO_MSG_OBJ of the exception object and can be evaluated by the Business Suite application.

The utility class CL_PCO_UTILITY provides additional methods for converting PCo messages into application log data records and saving them as an application log.

Note that while running F4 help, the exceptions that occurred may not be raised as error messages or termination messages (message type *Error* [E] or *Termination* [A]). This results in a short dump. Instead, use a more suitable message type (*Status* [S], *Information* [I], or *Warning* [W]) and use the ABAP command `DISPLAY LIKE` to set the display format *Error* or *Termination*:

```
MESSAGE lv_err_txt TYPE cl_pco_utility=>gc_msgty_stat DISPLAY LIKE
cl_pco_utility=>gc_msgty_error.
```

3.8.4 ABAP Sample Implementations

Use

The following sections contain ABAP sample implementations that demonstrate how easy it is to use ABAP to implement the exchange of information between external data sources and Business Suite applications. For communication with the external data source the relevant PCo agent instances must be created and started.

More Information

Example of Data Exchange Using Queries [page 110]

Examples of Generating and Processing Notifications [page 112]

3.8.4.1 Example of Data Exchange Using Queries

Transfer Default Values for Order Confirmation

The production progress of a discrete production is entered using confirmations. For this, the user creates a *Time Ticket Confirmation* in the ERP system in transaction CO11N. After the user has chosen the *Actual Data* pushbutton, the system suggests the input fields with the relevant data of the associated order operation. The machine that is used for production can provide the actual data for yield, scrap, and manual rework. Using PCo, this data is to be transferred to the relevant input fields automatically when the actual data is suggested.

Conversion

Read access for the data of the external data source is required. A suitable code position for implementing the PCo query is the include `ZXCOFU06` that belongs to the user exit `CONFPP01`. If the user chooses the *Suggest Actual Data* pushbutton, the user exit code is run.

Simplifications

For simplicity, the fully qualified path details for the data source tags are hard-coded. The fully qualified path must always be stated in the internal PCo format using / as the separator. Note the following:

- Use the test program `RPCO_BS_INT_TEST` or the search help `S_PCO_ELM_BROWSE_TAG` to determine the fully qualified path.
- If the fully qualified path of the tag is too long, use the function module `S_PCO_CALL_POPUP_NAMESP_BROWS` and evaluate the output parameter `ET_SEL_NODES`. You can take the fully qualified path as a string variable from the component `TAG_ID`.

Implementation of a User Exit

1. Call transaction CMOD and create any project.
2. Assign the enhancement `CONFPP01`.
3. Switch to the component view and place the mouse pointer on the `EXIT_SAPLCORF_101` entry of the function module exit.
4. Use the mouse to double-click.
   The system switches to the display of the function module `EXIT_SAPLCORF_101`.
5. Create the implementation for the user exit by placing the mouse pointer on `ZXCOFU11` in the change mode and then double-clicking.
6. Insert the sample code from the document (page 21 to 24) attached to SAP Note 1576651 as the source code.
7. Save and activate your changes.
8. Activate the user exit.

Sample Code Explanations

The sample code in SAP Note 1576651 consists of the following parts:

- Generating an application handle
- Instantiating the wrapper class for PCo integration (`CL_PCO_PAC`)
- Structuring the buffer table with information on tags whose values are to be read
- Calling the wrapper method to read tag information
- Converting the query results (data reference) to floating point values and assigning them to relevant fields of the confirmation structure

- Error handling or success message

Optimization Options

You have the following options for optimizing system performance:

- Dynamic determination of the path information using the work center of the operation that is confirmed:
  - You can use the feature classification for work centers or equipment to store path components as characteristic values.
- Evaluation of PCo user messages
  - **Table PCO_MSG_OBJ** of the exception object **lo_pco_exc**
  - Evaluation of export parameter **ET_PCO_MSG_OBJ** of the method **READ_TAG**

3.8.4.2 Examples of Generating and Processing Notifications

Use

Notifications are generated when the trigger conditions of a tag subscription have been met. PCo sends the notification to the destination systems that are assigned in the notification.

The following examples of generating and processing notifications are described:

- Conversion of Notifications to Time Ticket Logs [page 113]
- Automatic Running of Time Ticket Confirmations [page 115]

Prerequisites

For a subscription through Business Suite applications, PCo first creates a notifications for the transferred application handle. For this, the Business Suite application needs to select an existing notification template of the associated PCo agent instance when subscribing to tags (see Notification Template [page 503]). You need to create the notification template in the *PCo Management Console* for the PCo agent instance first. The notification template contains the following information:

Field/Setting in the Notification Template	Meaning
Trigger Type and *Trigger Expression*	*Trigger Type* `Always` is the preferred *Trigger Type* for Business Suite integration scenarios. The other trigger types can be used for Business Suite integration scenarios insufficiently only.

Subscription Lifetime (Seconds)	The default value for the lifetime is an hour (3600 seconds). See also: Notification Template: Remote Subscription Tab [page 505]
Settings for reliable connections and reliable message delivery	• You can set the number of attempts to restore the connection if the PCo connection to the external data source is lost. • You can state the lifetime of notification messages that could not be delivered. The default value for the lifetime is one day. See also: Notification: Reliable Connection Tab [page 518]
PCo destination systems to which the notification is transferred	On the *Destinations* tab of the notification template you assign the PCo destination system for your Business Suite system, which is to receive and process the notification message. You define the connection data for the Business Suite system when defining the relevant PCo destination system in PCo (see Destination System: RFC Client Settings Tab [page 347]).

3.8.4.2.1 Conversion of Notifications to Time Ticket Logs

Scenario

You use the test program **RPCO_BS_INT_TEST** to create subscriptions for the value change of tags. If the values of these tags change, PCo generates a notification and sends it to the Business Suite systems that were assigned as PCo destination systems in the notification template used. The system uses the notification content (header data, error messages, expressions) to generate application log entries. The test program can be used to display these application logs.

Conversion

You need a new class for processing notifications. This new class inherits from the class **CL_PCO_NOTIF_HANDLER** and redefines the instance method **EXECUTE**. Additionally, the BAdI **BADI_S_PCO_HANDLE_NOTIF** must be implemented for the filter value **APPLICATION = SAPTESTING**.

> **i** Note
>
> The ABAP objects mentioned are implemented in package S_PCO already.

Implementation of a Class for Processing Notifications (Class CL_PCO_SAPTESTING_NOTIF)

Program Blocks

- Setting the severity level to error when error messages are part of the notification
- Generating an application log entry that contains the name and description of the notification
- Evaluating the transferred expressions and generating the relevant application log entries
  Converting the expressions to instances of the exception class **CX_PCO_BS_INT** since these can be converted to application logs easily
- Adding error messages to the application log
- Saving the application log and updating the changes

ABAP Code for the Method EXECUTE (Class CL_PCO_SAPTESTING_NOTIF)

1. Define the method parameters:

Parameter	Type	Data Type	Optional	Description
IT_ERROR_MSG	Import	Type PCO_T_QUERY_MESSAGE_OBJ	x	Error messages for the notification (object instance)
IT_EXPR_DATA	Import	Type PCO_T_EXPR_DATA	x	PCo expression data
IS_APPL_HANDLE	Import	Type PCO_S_APPL_HANDLE	x	PCo: Data structure for application handle
IS_NOTIF_HEADER	Import	Type PCO_S_NOTIF_HEADER	x	PCo: Header data notification
IV_TEST_MODE	Import	Type BOOLE_D	x	'X': Test mode

2. Assign the exception class **CX_PCO_BS_INT** (PCo Suite Integration: Exception class).
3. Copy the ABAP code provided in the document (page 26 to 27) attached to SAP Note 1576651 to the method EXECUTE.

Implementation of Class CL_PCO_IM_SAPTESTING_NOTIF for the BAdI Implementation

Program Blocks

This class enables the return of the name of the class that processes the notifications (CL_PCO_SAPTESTING_NOTIF) if the application name of the application handle has the value *SAPTESTING*.

ABAP Code for Method IF_EX_S_PCO_HANDLE_NOTIF~GET_CLASS_NAME

1. Create the following parameters for the method:

Parameter	Type	Data Type	Optional	Description
IS_APPL_HANDLE	Import	Type PCO_S_APPL_HAN-DLE	No	PCo: Data structure for application handle
EV_CLASS_NAME	Export	Type SEOCLSNAME	No	Name of the class that processes the notification(s)

2. Assign the exception class **CX_PCO_BS_INT** (PCo Suite Integration: Exception class).
3. Copy the following ABAP code to the method:

```
METHOD if_ex_s_pco_handle_notif~get_class_name.
* Processing of notifications is executed with the help of classes that
* inherit super class CL_PCO_NOTIF_HANDLER.
* Based on entered application handle information this BAdI
* implementation returns the name of handler class

  CONSTANTS:
    lc_appl_name_saptesting  TYPE pco_s_appl_handle-appl
      VALUE 'SAPTESTING',
    lc_class_name_saptesting TYPE seoclsname
      VALUE 'CL_PCO_SAPTESTING_NOTIF'.

  CLEAR: ev_class_name.

  IF is_appl_handle-appl = lc_appl_name_saptesting.
    ev_class_name = lc_class_name_saptesting.
  ENDIF.

ENDMETHOD.
```

3.8.4.2.2 Automatic Running of Time Ticket Confirmations

Scenario

Time ticket confirmations are to be run automatically as soon as the values for the quantity produced, the scrap, and the rework change. These values are assigned to tags that have been subscribed to previously.

Conversion

A new class for processing notifications is required. This class inherits from the class **CL_PCO_NOTIF_HANDLER** and redefines the instance method **EXECUTE**. For a redefinition, an instance method can be implemented in a subclass without the interface being changed. Additionally, the BAdI **BADI_S_PCO_HANDLE_NOTIF** must be implemented for the filter value APPLICATION = PRODORDCON. The

application handle contains the unique number of the confirmation as a component of the handle. The confirmation number is used to supplement the remaining confirmation data such as order number and operation number when the confirmation is processed.

Simplifications

- No evaluation of error messages that could be part of the notification
- Use of hard-coded names for tags (evaluation of notification expressions)
- Reading the order number and operation data from the database (using SELECT)
- No reworking of confirmations for which the associated goods movement could not be performed

Implementations

Implementation of a Class for Processing Notifications (Class ZUD_RK_CONF) [page 116]

Implementation of Class ZUD_RK_BADI_KONF for BAdI Implementation [page 118]

3.8.4.2.2.1 Implementation of a Class for Processing Notifications (Class ZUD_RK_CONF)

Program Blocks

- Determining the confirmation number from the handle of the notification
- Reading the operation for the confirmation number
- Reading the order number of the associated production order from the database
- Determining the sequence number and operation number
- Generating the confirmation using the BAPI BAPI_PRODORDCONF_CREATE_TT
- Updating by calling the function module BAPI_TRANSACTION_COMMIT

ABAP Code of Method EXECUTE (Class ZUD_RK_CONF)

1. Create the following parameters for the method:

Parameter	Type	Data Type	Optional	Description

IT_ERROR_MSG	Import	Type PCO_T_QUERY_MESSAGE_OBJ	x	Error messages for notification (object instance)
IT_EXPR_DATA	Import	Type PCO_T_EXPR_DATA	x	PCo expression data
IS_APPL_HANDLE	Import	Type PCO_S_APPL_HANDLE	x	PCo: Data structure for application handle
IS_NOTIF_HEADER	Import	Type PCO_S_NOTIF_HEADER	x	PCo: Header data notification
IV_TEST_MODE	Import	Type BOOLE_D	x	'X': Test mode

2. Assign the exception class **CX_PCO_BS_INT** (PCo Suite Integration: Exception Class).
3. Copy the following ABAP code to the method:

```
METHOD execute.
  DATA:
    lt_conf        TYPE STANDARD TABLE OF bapi_pp_timeticket.
  DATA:
    ls_appl_handle TYPE pco_s_appl_handle,
    ls_cafko_ru    TYPE cafko_ru,
    ls_cafvc_ru    TYPE cafvc_ru,
    ls_afvc        TYPE afvc,
    ls_conf        TYPE bapi_pp_timeticket,
    ls_expr_data   TYPE pco_s_expr_data.

* Get confirmation number from handle information
  ls_conf-conf_no = is_appl_handle-handle.

  CALL FUNCTION 'CO_DB_CAFVC_RU_READ'
    EXPORTING
      rueck_imp    = ls_conf-conf_no
    IMPORTING
      cafvc_ru_exp = ls_cafvc_ru
    EXCEPTIONS
      not_found    = 1
      OTHERS       = 2.
  IF sy-subrc <> 0.
    RAISE EXCEPTION TYPE cx_pco_bs_int
      EXPORTING
        textid      = cx_pco_bs_int=>error_query_exec
        error_cause = 'Error reading order data'.
  ENDIF.
* Determine order number
  SELECT SINGLE aufnr FROM afko INTO ls_conf-orderid
     WHERE
       aufpl = ls_cafvc_ru-aufpl.

* Determine order operation data for which confirmation shall
* be executed
  SELECT SINGLE * FROM afvc INTO ls_afvc
    WHERE
      aplzl = ls_cafvc_ru-aplzl AND
      aufpl = ls_cafvc_ru-aufpl.

  ls_conf-sequence = ls_afvc-plnfl.
  ls_conf-operation = ls_afvc-vornr.
```

```
  LOOP AT it_expr_data INTO ls_expr_data.
    CASE ls_expr_data-name.
      WHEN 'UD_CONF_QUANT'.
        ls_conf-yield = ls_expr_data-value.
      WHEN 'UD_CONF_SCRAP'.
        ls_conf-scrap = ls_expr_data-value.
      WHEN 'UD_CONF_REWORK'.
        ls_conf-rework = ls_expr_data-value.
      WHEN 'UD_STATUS'.
        ls_conf-fin_conf = ls_expr_data-value.
    ENDCASE.
  ENDLOOP.

  INSERT ls_conf INTO TABLE lt_conf.

* Use BAPI to create confirmation
  CALL FUNCTION 'BAPI_PRODORDCONF_CREATE_TT'
    TABLES
      timetickets = lt_conf.

* Execute COMMIT WORK
  CALL FUNCTION 'BAPI_TRANSACTION_COMMIT'.
ENDMETHOD.
```

3.8.4.2.2.2 Implementation of Class ZUD_RK_BADI_KONF for BAdI Implementation

Program Blocks

Return of the name of the handler class (ZUD_RK_CONF) if the application name of the application handle has the value *SAPTESTING*.

ABAP Code for Method IF_EX_S_PCO_HANDLE_NOTIF~GET_CLASS_NAME

1. Create the following parameters for the method:

Parameter	Type	Data Type	Optional	Description
IS_APPL_HANDLE	Import	Type PCO_S_APPL_HAN-DLE	No	PCo: Data structure for application handle
EV_CLASS_NAME	Export	Type SEOCLSNAME	No	Name of the class that processes the notification

2. Assign the exception class **CX_PCO_BS_INT** (PCo Suite Integration: Exception Class).
3. Copy the following ABAP code to the method:

```
METHOD if_ex_s_pco_handle_notif~get_class_name.
```

```
CONSTANTS:
    lc_appl_name_prodordcon  TYPE pco_s_appl_handle-appl
      VALUE 'PRODORDCON',
    lc_class_name_zud_rk_conf TYPE seoclsname
      VALUE 'ZUD_RK_CONF'.

  CLEAR ev_class_name.

  IF is_appl_handle-appl = lc_appl_name_prodordcon.
    ev_class_name = lc_class_name_zud_rk_conf.
  ENDIF.
ENDMETHOD.
```

3.8.5 Appendix

Tips and Tricks [page 119]

Overview: Objects for Integration [page 121]

SQL Data Types [page 122]

3.8.5.1 Tips and Tricks

Converting Timestamp Formats

Timestamps that are generated and processed in ABAP do not have the timestamp format that is used in JAVA or other programming languages. The following ABAP expression can be used to convert the ABAP timestamp value to the ISO format *yyyy-mm-dd-hh:mm:ss* as of Basis Release 7.02:

```
ev_conv_tstmp = |{ iv_time_stamp TIMESTAMP = ISO TIMEZONE = 'UTC   ' }|
```

Linking Individual Quotation Marks Using Text String

Use Case: When linking string variables, the string variables are to be put in single quotation marks.

Example: Output of text string `'Template'Text' Module'`

```
CONCATENATE 'Vorlagen' 'text' '-Baustein' INTO lv_result.
```

Result: lv_result = `'Template Text Module'`

Solution: Use double quotation marks to enclose a text string with simple quotation marks. See **constant GC_TXT_QM** of the **class CL_PCO_QUERY:**

```
CONCATENATE 'Vorlagen' cl_pco_query=>gc_txt_qm 'text' cl_pco_query=>gc_txt_qm
'-Baustein' INTO lv_result.
```

Result: lv_result = `'Template'Text' Module'`

Linking Spaces Using Text String

Use Case: The linking of text string variables that contain a space (SPACE) results in a text string in which there is no longer a space.

Example: Output of text string `'ABCD EFGH'`

Result: lv_string_3 = 'ABCDEFGH'

Solution: Use simple quotation marks to disguise spaces.

See **constant GC_TXT_SP** of the **class CL_PCO_QUERY:**

```
CONCATENATE lv_string_1 cl_pco_query=>gc_txt_sp lv_string_2 INTO lv_string_3.
```

Result: lv_string_3 = `'ABCD EFGH'`

Converting and Formatting Floating Point Numbers

Floating point numbers can have either a scientific format (1.2345E01) or a decimal format (12.345). If floating point numbers are to be displayed in the ABAP system, one of three decimal point formats is used. The system uses the user master record (table USR01, field DCPFM) to determine the formatting settings of the user that is logged on. The following format templates are available in the system:

- Decimal format: 1,234,567.89 (USR01-DCPFM = 'X')
- Decimal format: 1 234 567,89 (USR01-DCPFM = 'Y')
- Decimal format: 1.234.567,89 (USR01-DCPFM = '')

Two cases need to be differentiated with regard to Business Suite integration:

- Floating point numbers are converted to string variables and added to the CDATA segment of the PCo query. The Business Suite application then uses an RFC call to transfer the content of the PCo query to PCo.
- The Business Suite receives floating point numbers as the result of PCo queries in the form of data references. The data references are converted to string variables and output in the results view of the test program.

When thousand separators are used, it is difficult to effect the correct formatting of the floating point numbers for the value display: There is no standard SAP solution for this problem. Approaches to this challenge, which are discussed on the Internet (for example, SDN thread 1449990 and SDN thread115017), do not provide a sufficient solution.

Transferring Floating Point Numbers from Business Suite to PCo:

To transfer floating point numbers from Business Suite applications to PCo it is necessary to convert the floating point number to a string value. The string value is then part of the query data string, which is then transferred to PCo using an RFC call. At the same time, the decimal separators are unified and the thousand separator is removed (see **class CL_PCO_UTILITY**, **method CONVERT_STRVAL_TO_SDT_DREF**). PCo can then process the string values formatted in this way directly without any further conversions.

Displaying Floating Point Numbers Transferred from PCo to the Business Suite:

In the **test program RPCO_BS_INT_TEST**, the floating point numbers are displayed in the decimal format. The system can use the **instance method FORMAT_FLOAT_CHAR** (**class CL_PCO_UTILITY**) to set the thousand and decimal separators according to the formatting settings in the user master record of the current user. In the method mentioned, a specific algorithm is defined that guarantees the correct placing of the thousand and decimal separators.

3.8.5.2 Overview: Objects for Integration

The Business Suite system provides the following objects for integration:

Package S_PCO

You can display all objects relevant for integration in transaction SE80 in package S_PCO.

Authorization Object S_PCO_INT

You need to assign the necessary authorizations to the user that is used to exchange RFC data between the Business Suite system and PCo. The authorization object S_PCO_INT is available for this purpose.

You can use the authorization object S_PCO_INT to control which activities are permitted for the user when integrating SAP Business Suite applications with Plant Connectivity (PCo). The actvities are defined as domain fixed values of the domain S_PCO_ACTIVITY. Currently, you can select the following activities:

- READ Read values
- WRITE Write values
- SUBSCR Subscribe to value changes
- UNSUBSCR Unsubscribe
- FEATURES Display agent features (agent functions)
- BROWSE Browse the tag hierarchy
- NOT_CBCK Perform follow-on activities for a PCo notification

In addition, you can use the SAP application (field S_PCO_APPL) and the active SAP user (field USER) to maintain authorizations.

Roles

The following roles of the transaction PFCG are available for the integration:

- SAP_BC_SRV_PCO_BS_INT_ADMIN

The authorization profile for the role *PCo Business Suite Integration: Administrator* supports the following activities:
    - You can create RFC connections to communicate between Business Suite applications and Plant Connectivity (PCo).
    - You can implement BAdI methods for the Business Suite integration.
    - Running RFC module calls to transfer data between PCo and Business Suite applications.
- SAP_BC_SRV_PCO_BS_INT_USER
  The authorization profile for the role *PCo Business Suite Integration: User* supports performing RFC module calls to transfer data between PCo and Business Suite applications

Business Add-Ins

- BADI_S_PCO_AUTH_CHECK
  You can use this BAdI to specify additional authorization checks. These authorization checks are performed by ABAP.
- BADI_S_PCO_CONV_DATA
  You can use this BAdI to convert values with an unknown SQL data type. (See SQL Data Types [page 122]).
- BADI_S_PCO_HANDLE_NOTIF
  You can use this BAdI to influence the processing of notification messages by the SAP application that created the notification.

3.8.5.3 SQL Data Types

To be able to transfer the query results, PCo uses *SQL Data Types*. These SQL data types need to be converted to *ABAP Data Types* for integration with the Business Suite.

The table provides an overview of the supported SQL data types:

PCo Code	SQL Data Type	Data Type Description	Size in Bytes	ABAP Data Type
-7	BIT	Single bit value; value range: 0,1	1 byte	INT1
-6	TINYINT	Very small number; value range: -128 to 127	1 byte	INT2
5	SMALLINT	Small number; value range: -32,768 to 32,767	2 bytes	INT2

4	INTEGER	Whole number; value range: -2,147,483,648 to 2,147,483,647	4 bytes	INT4
6	FLOAT	Floating point; value range: -1.7E308 to 1.7E308	8 bytes floating points	f or FTPT
7	REAL	Floating point; -3.4E38 to 3.4E38	4 bytes floating points	f or FTPT
8	DOUBLE	Large floating point value; value range: -1.7E308 to 1.7E308	8 bytes floating points	f or FTPT
1	CHAR	UTF-8 encoded String with a length up to 254 Structure: 1 byte length + n character	1 byte + n UTF-8 sequences	SSTRING
12	VARCHAR	UTF-8 encoded String with a length up to 65,536 Structure: 2 byte length + n character	2 byte + n UTF-8 sequences	STRING
-1	LONGVARCHAR	UTF-8 encoded String with a length up to 4,294,967,296 Structure: 4 byte length + n character	4 byte + n UTF-8 sequences	STRING
91	DATE	Date/time structure represented by the number of milliseconds since January 1, 1970 UTC.	8 bytes	Conversion to the format `d` required
92	TIME	Date/time structure represented by the number of milliseconds since January 1, 1970 UTC.	8 bytes	Conversion to the format `t` required

93	TIMESTAMP	Date/time structure represented by the number of milliseconds since January 1, 1970 UTC.	8 bytes	Conversion to the format `TIMESTAMPL` required
16	BOOLEAN	Boolean value; value range: false, true	1 byte	Boole_D

3.9 Integration with SAP ODA

Use

If you are using SAP ODA or intend to do so in the future, you must use the **SAP Plant Connectivity** component to establish the connection to an OPC server. For more information about SAP ODA, see the ERP or S/4HANA documentation under **SAP OPC Data Access (SAP ODA)** and under 1993447 .

Prerequisites

Settings for SAP ODA [page 125]

Features

SAP ODA Functions

In transaction `COOPC1`, in the *OPC DA Tags* view, you can use the following functions for the selected tag:

- **Reading and writing tags**
  If you choose the *Read Synchronously* pushbutton, the ERP system or the S/4HANA system can read the values of the selected tag from the OPC server directly and display them in the *Value* field.
  You can also use the *Write Synchronously* pushbutton to send values to the OPC server, which may be able to trigger certain functions there. PCo handles reading and writing as a query.

  > **i Note**
  >
  > If you write a FLOAT value, you must always use the English notation regardless of the language version you are using, meaning that you must always use `.` instead of `,` for decimal places.

  For more information, see the ERP or S/4HANA documentation under **Reading and Writing of Items**.
- **Subscribing to tags**
  If you choose the *Register* pushbutton, the *Test OPC Data Access* screen appears. If on this screen you choose the *Register* pushbutton, the ERP or S/4 system can subscribe to the tag of the OPC server. The

tag details are displayed in the overview. For more information, see the ERP or S/4 documentation under **Item Subscription**.

Results in PCo

In the PCo Console, a subscription has the following results:

- In the agent instance, the entries for the individually logged activities are displayed on the *Log* tab. To be able to display all activities, you need to choose the *Refresh* pushbutton regularly.
- For example, the subscription is displayed on the *Subscription Items* tab as follows: *Name: Q630021000KEP_PCO_FLOAT*
  This name is structured as follows: system/client/plant/tag name
- When you open the agent instance, you see that PCo has created a further notification object in addition to the notification template. On the *Output* tab of the new notification you can see the tag that you have subscribed to in the ERP system.
  On the *Remote Subscription* tab of the notification template, the *Handle ID* (combination of system/client/plant/tag name) is now displayed.

Constraints

You can only use SAP ODA in ERP or S/4 systems if you use the **Process Management (PP-PI-PMA)** component.

> ⚠ Caution
>
> The following limitations apply to SAP ODA in conjunction with Plant Connectivity:
>
> - PCo does **not** currently support **Alarms & Events functions**. You can connect a source system of type OPC DA only. If you want to use the functions of the **OPC Alarms & Events** in the ERP or S/4HANA system, you must use a third-party middleware.
> - The settings *Access Type*, *Buffer Time*, and *Deadband* in transaction COOPC1 are no longer relevant. You need to make these settings in the *PCo Management Console*.
> - For the server settings in transaction COOPC1, you can select only the server name that is assigned to the PCo agent instance.

3.9.1 Settings for SAP ODA

If you are using SAP ODA for the first time, maintain the settings for the integration as follows:

1. You create an OPC DA source system in the *PCo Management Console*.
2. In the ERP system or in the S/4HANA system, you use transaction `SM59` to create an RFC destination for the agent instance. (See Defining an RFC Destination [page 91].)
3. In *the PCo Management Console*, you create an *RFC destination system* of type *SAP ODA* for your ERP or S/4HANA system, and make all the required settings on the *RFC Client Settings* tab. (See RFC Destination System [page 347].)
4. In the PCo Management Console, you create the *agent instance* and make the settings for the SAP ODA RFC server on the *Servers* tab. (See SAP ODA RFC Server [page 426].)

5. You create a notification template [page 503] with the following settings:
   - *Trigger* tab:
     - The *Enabled* checkbox is selected.
     - The *Always* trigger type is set.
     - All other fields remain empty.
   - *Remote Subscription* tab
     - Enter the *Lifetime* of the subscription in seconds (see Notification Template: Remote Subscription Tab [page 505]).
       When using the *PI Sheet* in ERP or S/4, you should set the lifetime to be longer than the time in which a PI sheet is open at once. For example, you can enter the shift duration or the duration until automatic logoff.
   - *Destinations* tab: Here, you specify the destination system that you have created for your ERP or S/4HANA system.
6. You start the agent.
7. In the ERP system or S/4HANA system, you use transaction `COOPC1` to make the following settings:
   - You define the basic settings.
   - You use the RFC connection described under point 3 to define the OPC server.
   - You make the settings for the OPC DA tag.

   For more information, see Settings in Transaction COOPC1 [page 127].

3.9.1.1 Defining an RFC Destination

Context

In the ERP or SAP S/4HANA system, you need to use transaction SM59 to define the RFC destination for a PCo agent instance. The communication between a Business Suite system and PCo is established using *Remote Function Calls* (RFC modules).

You need to create exactly one RFC destination for each PCo agent instance.

Procedure

1. Make the following general settings in transaction SM59:
   - RFC Destination: This can be defined freely
   - Connection Type: **T** (TCP/IP connection)
2. Make the following entries on the *Technical Settings* tab:
   - Select *Registered Server Program*.
   - *Program ID*: Here you enter the value that you specified in the *Program ID* field in the PCo agent instance on the *Query Ports* tab.

- *Gateway Host*: Here you enter the value that you specified in the *SAP Gateway Host* field in the PCo agent instance on the *Query Ports* tab.
- *Gateway Service*: Here you adopt the value that you specified on the *Query Ports* tab.
- You can use the *Test Connection* pushbutton to check the connection to PCo.

3. On the *Logon & Security* tab, you can enter the following if you want to implement the Secure Network Communication (SNC) component:
   - Select the *Active* radio button in the *Security Options* screen area.
   - Choose the *SNC* pushbutton. The *Change SNC Extension View: Details* dialog box appears.
   - In the *QoP* (Quality of Protection) field, enter the value that you set in PCo in this field on the *Query Ports* tab.
   - In the *Partner* field, enter the SNC name of the user, for example, `p:CN=D0XXXXX, O=SAP-AG, C=DE`. This name is in *User Maintenance* (transaction SU01) on the *SNC* tab.
     You also need to enter this name in the PCo Console on the *Query Ports* tab in the *User SNC Name* field.
   - For more information about SNC settings, see SNC Settings [page 92].

Results

You can now create the following data in the *PCo Management Console*:

- You create a *source system*.
- You create an *RFC destination system* of the type *SAP ODA*. See also: RFC Destination System [page 347]
- You create the *agent instance* and enter the RFC connection data on the *Servers* tab. See also: Servers Tab [page 419]

3.9.1.2 Settings in Transaction COOPC1

Use

You make the settings for SAP ODA in the ERP system or S/4HANA system in transaction COOPC1, or in Customizing under *Production Planning for the Process Industry* > *Process Management* > *SAP ODA (OPC Data Access)* > *Make Settings for SAP ODA*.

Procedure

Note the following for the settings in the ERP system or S/4HANA system for SAP ODA in conjunction with PCo:

- **Basic Settings**: Here you specify a *User Name* and the *Password*. This can also be the user name of the RFC service, for example. For SAP ODA, this user name must correspond to the user name that you specified in PCo for the destination system on the RFC Client Settings Tab [page 348].

- **OPC Server**: You need to make the following settings:
  1. Enter a name of your choice for the OPC server.
  2. Specify the RFC destination that you have defined for your PCo agent instance.
  3. Specify the OPC server ID. You can select only the server ID that is assigned to the PCo agent instance. Depending on which server ID you have selected, the checkboxes for the *Synchronous Data Access* and *Data Access Subscription* functions are selected automatically if supported by the selected OPC server. Both of these functions are ERP functions. Remove the selection only if you do not want to use or allow the function in the ERP system.
  4. It is not necessary to enter anything in the fields in the *Further Settings* area of the screen. You make these settings in the PCo Console directly.
- **OPC Data Access Items**
  1. Enter a name of your choice for the *OPC item*.
  2. Specify the *OPC server* that you previously specified in the step *Define OPC Server*.
  3. Select an *OPC Item ID*.

> **i** Note
>
> The following settings for the tag are no longer relevant in connection with PCo because they are maintained in the PCo Management Console:
>
> - Access type
> - Buffer time
> - Deadband

4 Configuration Elements in SAP PCo

You can create and configure the following elements in the Management Console:

- Source System [page 129]
- Destination System [page 270]
- Agent Instance [page 412]
- Notification [page 496]

4.1 Source System

A source system refers to a specific data source from which data is to be queried. You need to make an entry in the PCo console for the computer or control device from which you want to query data.

The connection between the data source and PCo is established using agents. Agents are .NET DLL assembly components. In the PCo Management Console, you can create one or multiple source systems for an agent. You can then define one or multiple agent instances for each of these source systems. The agent instance establishes the actual data flow between the data source and PCo. (See also: Agent Instance [page 412].)

The source system in PCo is an adapter that converts the protocol-specific representation of data into a general and unified view. The source system performs the following tasks:

- A source system translates PCo commands such as queries or configuration parameters, for example, for a reliable connection, into protocol-specific commands.
- A source system transforms the server data in the data source into a tag-based data structure that can be browsed. A tag-based data structure is already given for some data sources (for example, OPC servers). For other data sources, such as the file system source system, the content must be transformed, that is, the content of a directory must be displayed as a node structure so that it can be browsed.
- It converts the protocol-specific data types, such as the time stamp, into the .NET data types that are used in PCo.

The table below contains the source systems available as standard in PCo. In addition, you can see for which communication processes each source system can be used:

- Query process (see also: Query Process (with SAP MII) [page 43])
- Notification process (see also: Notification Process [page 35])
- Destination system calls with response processing (see also: Enhanced Method Processing (EMP) [page 57] and PCo as OPC UA Server and as Web Server [page 54])
- Retrieve and store queries with the query destination system (see also: Query Destination System [page 375])

Source System	Data Source	Query Process	Notification Process	Destination System Calls with Response Processing	Query Destination System
Asset Framework Source System [page 237]	OSIsoft PI Asset Framework (part of the OSIsoft PI system)	x	x	x	x
Citect Source System [page 240]	CitectSCADA system	x			x
File Monitor Source System [page 245]	Monitors a specific directory on a local computer or a remote computer	x	x		x
File System Source System [page 255]	Like file monitor source system but with enhanced functions	x	x	x	x
IP21 Source System [page 242]	Aspen InfoPlus 21 Database	x			x
Modbus Source System [page 188]	Sets up the connection to programmable logic controllers (PLC)	x	x	x	x
MQTT Source System [page 206]	Sets up the connection to an MQTT server	x	x		x
ODBC Source System [page 182]	Sets up the connection to any database management system using the *Open Database Connectivity Interface*	x			
OLE DB Source System [page 175]	OLE-DB-based data sources such as MS Excel, MS Access	x			
OPC A&E source system (OPC Source Systems [page 131])	OPC Alarms & Events Server (conforms to 1.10)	x			

Source System	Data Source	Query Process	Notification Process	Destination System Calls with Response Processing	Query Destination System
OPC DA source system (OPC Source Systems [page 131])	OPC Data Access Server (conforms to 2.05a or 3.0)	x	x	x	x
OPC HDA source system (OPC Source Systems [page 131])	OPC Historical Data Access server (conforms to 1.20)	x	x	x	x
OPC UA source system (OPC Source Systems [page 131])	OPC Unified Architecture (based on the specification 1.03)	x	x	x	x
OSISoft PI Source System [page 234]	OSIsoft PI system	x	x	x	x
Proficy Historian Source System [page 233]	GE FANUC Proficy Historian server	x	x	x	x
Socket Source System [page 262]	Sets up the connection to a TCP/IP-based socket server	x	x		
Timer Source System [page 229]	A timer is a source system that changes its value periodically at specified times. You can use timers to trigger actions using PCo at configurable, usually periodic, times.		x		

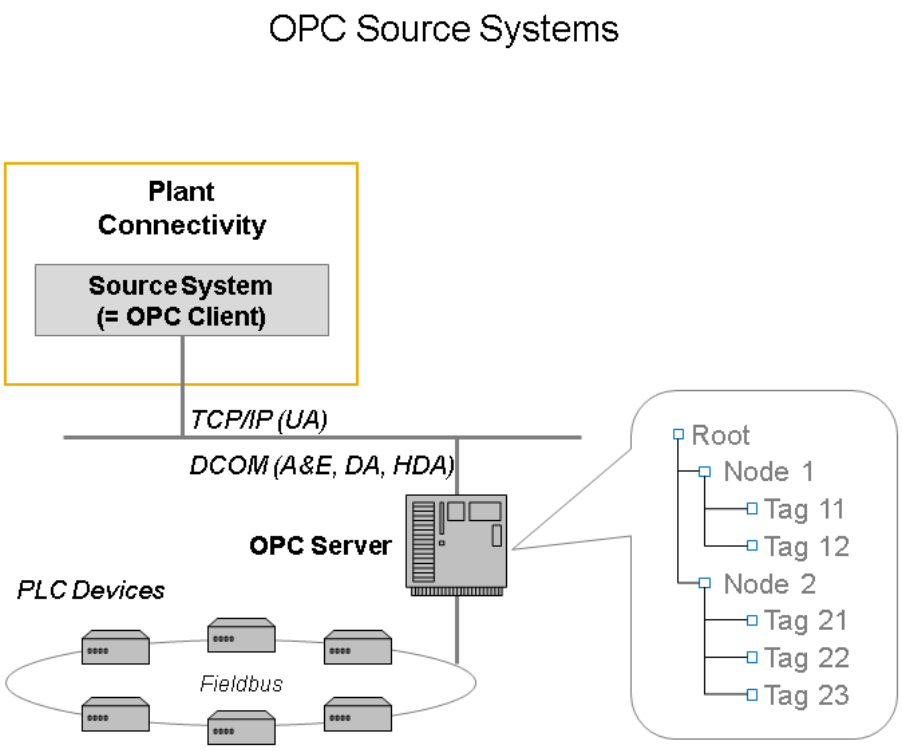
4.1.1 OPC Source Systems

Use

PCo provides the following agents to establish a connection to OPC servers:

- **OPC Data Access**
- **OPC Historical Data Access**
- **OPC Alarms & Events**

- OPC Unified Architecture



Overview of OPC Source Systems

Key Features

Supported Processes

All OPC agents supplied with PCo support the **notification and query process**.

In the notification process, if OPC agents are used, PCo continually transmits the tags and checks whether the stipulated conditions are met (exception: OPC DA source systems). If so, PCo sends a notification message to the destination system or systems.

If the source system is an **OPC DA system**, the source system checks the measurement values itself and sends only the deviating tag to PCo. PCo then sends the notification message to the destination system or systems.

Structuring of Tags

All data sources provided by an **OPC DA server** are organized in a namespace. This can be **flat** or **hierarchical**.

In a **flat namespace**, all tags lie on one level.

In a **hierarchical namespace**, the hierarchy nodes may represent the equipment and machines of a plant, for example. Leaves, which represent a temperature sensor or regulator, for example, are located on the nodes. The leaves are referred to as tags. Each tag has its own ID and can provide values.

Example

Line 2

- Machine 30
  - Temperature sensor 301
    - Status
    - Temperature

See Also

Basic Concepts of OPC [page 133]

4.1.1.1 Basic Concepts of OPC

OPC

OPC stands for *Open Platform Communications*. OPC is the name for standardized software interfaces that enable the exchange of data between the applications of a wide variety of manufacturers in automation technology.

OPC is a standard that defines manufacturer-independent interfaces for industrial applications using COM/DCOM technology. The OPC standard was conceived specially for the process control level. OPC servers enable access to different data sources (such as process control systems, programmable logic controllers, and temperature sensors) and thus provide process data that can be requested by OPC clients.

The *Plant Connectivity* component is an OPC client that can communicate with OPC servers. In this way, the exchange of data between applications is considerably simplified. OPC clients and OPC servers are currently PC-based systems on which a Microsoft operating system runs.

For more information about OPC settings, see the Web pages of the OPC Foundation at https://opcfoundation.org/ .

OPC Standards

The OPC Foundation provides a series of freely accessible OPC standards. These contain interface descriptions for different areas of automation technology. These interfaces enable the efficient exchange of data between the software components of different manufacturers.

Plant Connectivity supports the following OPC standards with its own agents:

- OPC Data Access 2.05a or 3.0

  Note

  1.x compliant servers are not supported.

- OPC Alarms and Events 1.10
- OPC Historical Data Access 1.20

- OPC Unified Architecture (1.03 Specification)

4.1.1.2 OPC DA Source System

With the OPC Data Access source system, you can set up a connection to an OPC DA server that is 2.05a or 3.0 compliant.

OPC DA source systems provide data from various data sources (such as from process control systems) that can be read by PCo. An OPC DA server makes it possible to access a large number of tags that can stem from many different data sources.

PCo provides the following tabs for configuring an OPC DA source system:

- Server Tab [page 134]
- Settings Tab [page 135]
- Aliases Tab [page 138]
- Reliable Connection Tab [page 142]

4.1.1.2.1 Server Tab

Prerequisites

For the OPC DA source system and the OPC DA agent instance to work correctly, both the server and the client must have mutual trust.

The mutual trust must be set up in the DCOM and, if possible, at the level of the domain services too.

Context

On this tab, you define the server settings of your OPC A&E, OPC DA, or OPC HDA source system.

Procedure

1. On the *Plant Connectivity Management Console* screen, select an OPC source system and click *Server*.
2. Select the OPC server in the *Server Name* field.
3. To display the available servers on the selected machine, choose the *Refresh* pushbutton.
4. To select a server, click the relevant row. The selected server is then highlighted in gray.

4.1.1.2.2 Settings Tab

OPC Data Access Source System

You have created a source system that is based on the OPC DA standard. See also: Basic Concepts of OPC [page 133] and OPC DA Source System [page 134].

1. To make the settings for the OPC DA source system, select the OPC DA source system in the *Plant Connectivity Management Console* and click on the *Settings* tab.
2. For the information you need to enter, use the following table:

OPC DA Settings

Field	Description
Acquisition Mode	Specifies whether, in the case of a *tag query*, PCo is to obtain the value of the queried tag from the OPC server **synchronously** or **asynchronously** in the query process.
Force Flat Namespace	If you choose the setting *On*, a flat namespace is created. In this case, all hierarchy information is removed from the namespace of the OPC DA server and stored at root level. **i Note** The setting *Force Flat Namespace* is only possible for `OPC DA 2.05a`.
Use Vendor Filter	If your source system supports the *OPC specification 3.0*, you can use this setting to specify that the **source-system-specific filter method is used during browsing and not the general filter method that applies to all source systems**. To do so, choose the setting *On*. When you enter the filter string, you need to adhere to the syntax specified by the software maker of your source system.
Synchronous Read Source	This field is only used in the query process. You can make the following settings: ○ From Device With this setting, the tags are read directly from a device. ○ From Cache With this setting, the tags are read from the cache of the OPC DA server. This setting results in better performance.

Field	Description
Acceptable Data Quality	With this setting, you define the OPC data quality of a tag that is to be allowed. When a tag is read out of the OPC source system, the data quality of the tag is also provided and can be taken into account accordingly. **Example** For example, you have set the data quality in PCo to *Good*. The OPC server checks the PCo settings and only sends tags with the appropriate data quality, meaning that a tag with poor data quality is not transferred to PCo in this case. The following data quality settings are possible: ○ **Any Quality** This is the **default setting**. This setting means that the data quality of a tag is not taken into account. ○ **Good** If you choose the *Good* setting, only tags with good data quality are used. If the data quality is poorer (*Uncertain* or *Bad*), PCo issues an error message. ○ **Uncertain** If you choose the *Uncertain* setting, only tags with the data quality *Good* or *Uncertain* are accepted. If the data quality is poor, PCo issues an error message. ○ **Bad** If you choose the *Bad* setting, PCo also accepts tags with poor data quality. ○ **Error** The OPC server sets the data quality of a tag to *Error* if the call has failed. The effect of the *Error* setting is that tags with incorrect data quality are also accepted by PCo.

Field	Description
Browsing Mode (DA 2.05a)	The browsing mode specifies the internal technique with which the hierarchical namespace of an OPC DA 2.05a server is to be browsed. The following settings are possible: ○ *Absolute Path* This is the default setting. With this setting, the direct path is given internally during browsing and the system goes directly to the desired hierarchy node. ○ *Relative Path* You must only use this setting if browsing is not successful using the default setting. This can occur on servers that do not quite conform to the OPC specification during browsing. With this setting, each level in the namespace is browsed. It is not possible to skip a level. **Example** If you are using an RSLinx server, you need to use the *Browsing Mode Relative Path* setting.

In the following area, you make the settings for a **group of tags**. These settings then apply to all tags of this group on the OPC server.

Group Settings

Field	Description
Name	Name of an OPC-server-internal group of tags. This group comprises all tags specified for a certain agent instance on the *Subscription Items* tab. You can choose any name for the group.
Update Rate	Indicates the rate at which the OPC server is to send values to PCo. If you specify 100 milliseconds, for example, the OPC server is to send values to PCo every 100 milliseconds.

Field	Description
Deadband	Limit value for filtering out measurement values. Here you can enter a percentage to define the bandwidth in which the value of a tag must at least change before the OPC DA source system reports the value to PCo as changed. **Example** You have defined a minimum and a maximum value for a specific tag on the OPC server. The percentage that you enter here in the *Deadband* field is applied to the difference between the minimum and maximum value. For example, on the OPC server, you have specified for the tag that a tag value is to be between 0 and 100. With a deadband of 10%, the tag value must now change down or up by 10 compared to the last reported value so that a notification is sent from the OPC server. If a tag value is 50 and changes to 52, no notification is sent. If the tag value changes later to 58, a notification is still not sent. Only when the tag value reaches 60 is a notification triggered from the OPC server.

4.1.1.2.3 Aliases Tab

Use

The *Aliases* tab is required in the query process [page 43] only.

An alias enables the selection of certain tags and groups them. These tags constitute a subset of the tags provided by the data source. This improves system performance since only the values of the selected tags are queried by the source system. See also: Filter Functions [page 266].

However, you can use the alias to rename the tags so that, for example, you can display shorter or more descriptive tag names in the destination system.

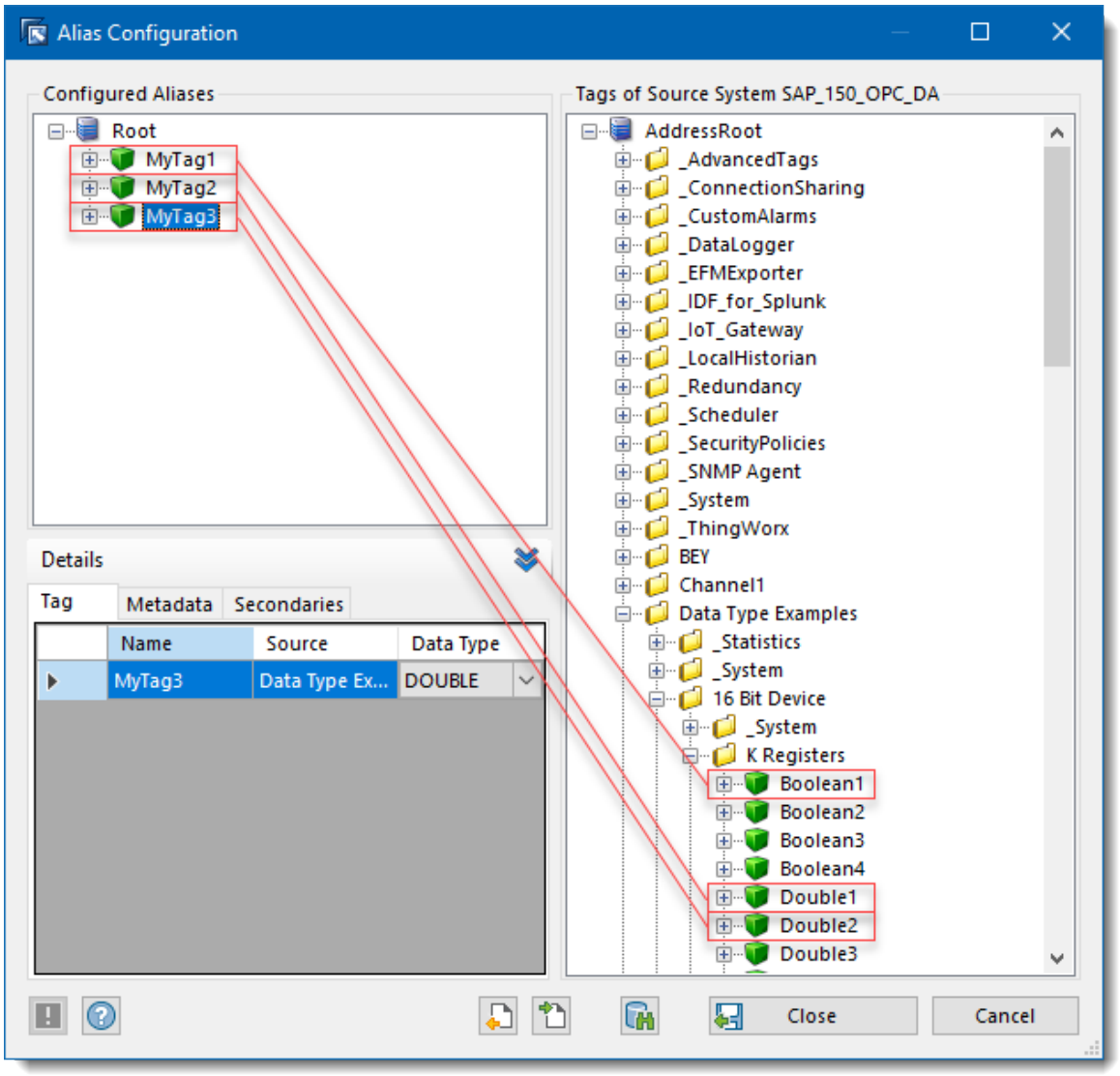
Integration

You can assign one alias only per agent instance. You assign the alias to the agent instance on the *Tag Query* tab. See also: Tag Query Tab [page 475]

Activities

1. To create an alias, select your source system and then click on the *Aliases* tab.
2. On the *Aliases* tab, choose the *Add Alias* pushbutton.
   The *Alias Configuration* dialog box appears.
3. In the *Alias Configuration* dialog box, choose *Browse for Tags* to search through all the data in the namespace of the source system, for example, the OPC server.
   PCo then displays the *Browse Information* dialog box.
4. In the *Browse Information* dialog box, you can enter the following:
   - In the *Mask* field, you can restrict the selection of tags by entering, for example, `Int*`. In this case, only the tags whose names begin with `Int` are selected from the source system.
   - You can have an alias generated automatically by the system if you select the *Generate Alias* checkbox. The system then displays the complete namespace. However, this approach is useful in exceptional cases only, for example, if you want to rename all tags.
   - Another option is to select the *Browse Secondary* checkbox.
     If you select this checkbox, PCo gets the available additional information (such as the quality of a measured value or the timestamp of the value generation) for each tag.
5. Choose *OK* in the *Browse Information* dialog box.
   The namespace of the source system is now searched.
6. PCo displays the namespace of the source system in the right-hand half of the screen.
7. To be able to display the entire hierarchy of the namespace, open the top node (*Address Root*) in the right-hand half of the screen and then all lower-level nodes.

8. Click to choose the tag or tags that you want. Use drag and drop to drag the tag or tags over to the *Root* node in the left screen area, as shown in the example:



9. If you want to display the **details for the selected tag**, click on the selected tag and choose the *Hide/Show* pushbutton.
   In the *Details* screen area, the system then displays all information for the tag:
   - The information about the selected tag is displayed on the **Tag tab**:
     - Name of the tag
     - Data type of the tag
   - **Metadata tab**
     Metadata is additional data for a tag that describes the properties of the tag. The metadata remains constant; for example, the description of the tag, the minimum or maximum value defined for the tag. The name, the value, and the data type are always displayed for each of the metadata.
   - **Secondaries tab**
     The secondaries for a tag are only displayed if you have previously selected the *Browse Secondary* checkbox.
10. Choose the *Close* button to save the selected tag(s).

Example

Using the query process, you want to check the temperature (tag 1) and pressure (tag 2) of a boiler on a regular basis. The source system supplies numerous different tags. On the *Aliases* tab, you can now group the two tags under an alias for the boiler.

Related Information

Import and Export [page 141]

4.1.1.2.3.1 Import and Export

You can choose the *Export Alias Configuration* button to save the configuration in an XML file with the extension `.alias`. You can then choose the *Import Alias Configuration* button to load the saved configuration into the alias dialog of another source system, provided that it has a comparable tag structure.

> **i Note**
>
> Note that the existing configuration is replaced during the import.

If you use a CSV file for the alias import, it must have the following column structure and column sequence:

- Tag Name
- Source
- Description
- Minimum
- Maximum
- Data Type
- Group

For the data type, you must use one of the types that is permitted for aliases. These differ from the .NET data types used otherwise in PCo. You can call the list of permitted types by creating an alias for any tag and showing the details. The list of permitted data types can be found in the dropdown list in the *Data Type* column.

4.1.1.2.4 Reliable Connection Tab

Context

You use this tab to maintain the connection to the source system. PCo checks the connection on a regular basis.

Procedure

1. On the *Plant Connectivity Management Console* screen, select a source system and click on the *Reliable Connection* tab.
2. For the information you need to enter, use the following table:

Field	Description
Max. Number of Retries	Maximum number of connection attempts before the connection counts as failed. The default value is `3`. If the attempt to establish a connection fails, the agent instance switches to the *Faulty* state. In this case, an icon in the form of a red square with a white cross is displayed for the agent instance. See also: Monitoring [page 602]. **i Note** There are no retries if this value is `0`.
Retry Interval (Seconds)	Indicates the number of seconds before the next send attempt. The default value is `30 seconds`, meaning that the PCo system checks the connection every 30 seconds. If the agent instance establishes that there is no connection, PCo makes a new connection attempt every 30 seconds, for example.

Next Steps

Unlike the other source systems, the OLE DB source system and the ODBC source system only have a passive connection check. For more information about the passive connection check, see Reliable Connection Tab (OLE DB / ODBC Source System) [page 181].

4.1.1.3 OPC UA Source System

With the OPC Unified Architecture source system, you can set up the connection to an OPC UA server that is based on the OPC UA specification with version 1.03.

You can use the OPC UA source system to read and write current tags. In addition, the reading of historical tags (tag values, metadata, and secondaries) is supported. However, historical tag data can only be read if the tags also support historical data.

Restrictions with historical tag data:

- The writing of historical tags is not supported.
- The reading of processed data, such as average, minimum, or maximum values is not supported for historical tags.

PCo provides the following tabs for configuring an OPC UA source system:

- Session Tab [page 143]
- Security Tab [page 154]
- Subscription Tab [page 164]
- OPC UA Source System: Structures Tab [page 167]
- Aliases Tab [page 138]
- Reliable Connection Tab [page 142]

4.1.1.3.1 OPC UA Source System: Session Tab

Use

On this tab, you define the connection settings to an OPC UA source system.

Prerequisites

- You have installed an OPC UA server (conforms to version 1.03 of the OPC UA specification).
- You have selected the OPC UA source system during the PCo installation.

Procedure

Making Discovery Settings

1. Click on the *Add Source System* icon and choose *OPC UA Source System* as the source system type. The *PCo Console* shows the *Session* tab.

2. With the help of the following table, enter the necessary information in the *Discovery* screen area:

Field	Description
OPC TCP	You select this checkbox to specify that the OPC TCP protocol is to be used for determining the server endpoint.
HTTP	You select this checkbox to specify that the HTTP transport is to be used for determining the server endpoint.
HTTP (Private)	You select this checkbox to specify that a secure HTTP transport is to be used for the discovery of the server endpoint.
Discovery Server	You enter the name of the server here from which you can call the endpoints that are supported by this server. **i Note** By choosing *Browse Domain*, you can also search for an available *discovery server*.
Server Endpoint	The URL address of the selected *server endpoint* is displayed here. This is a specific OPC UA server that is connected to the *discovery server*.
Update Server Configuration on Connection	If you have selected this checkbox, the configuration of the server endpoint that is stored locally temporarily is updated at the time of connection. You always need to select this checkbox if you want to use a server endpoint that you entered manually.

3. Choose **Start Discovery**.
   You first need to enter the server on which the local discovery server is installed. Using this function, you can then start to search for the endpoints of the discovery server. Existing entries are overwritten and the list is built up again. All available OPC UA server endpoints are then listed on the server.
4. Choose **Select Endpoint**.
   The function lists all server endpoints that were found the last time the discovery server was called. You can select an individual endpoint. You can also manually add or change endpoints here. (See: Select Endpoint [page 151].)

Session

You define the following here:

Field	Description
Name	You enter the name of the session here. This is a unique name to identify the OPC UA client session associated with the OPC UA source system.

Field	Description
Keep-Alive Interval (sec)	The interval is used to maintain the OPC UA server connection. **i Note** Decreasing the *Keep-Alive Interval* may reduce system performance.

Field	Description
Package Size for Read Service	You can use this setting to define that the read service calls of an OPC UA server are always packaged, thereby limiting the number of nodes that are read in an individual call. This value should not be smaller than **10**. The reason for this is that some OPC UA servers cannot process more than a specific number of nodes for a service call. If, nevertheless, more nodes are requested, the server terminates the service call with the service result `BadTooManyOperations`. An OPC UA server can publish such a restriction as a limitation. **i Note** You can find the relevant information in the server address space under *Server* > *ServerCapabilities* > *OperationLimits*. If the server publishes such a restriction, packaging is performed automatically in the OPC UA source system and in the derived agent instances. In other words, a call in which too many nodes are queried is divided up into multiple calls with the recommended maximum number of nodes or with fewer nodes. However, there are servers that have such a limitation but do not publish this in the address space. You can force packaging by entering a package size. If there is a restriction by the OPC UA server as well as a value in this field, the smaller of the two values is used. **Example** The OPC UA server publishes that the package size for reading cannot be larger than 100. In the OPC UA source system you have configured a maximum package size of **50**. All read queries are divided up into packages of 50 or fewer. **Example** The OPC UA server publishes that the package size for reading must not be larger than 100. In the OPC UA source system you have configured a maximum package size of **0**. Since the value 0 stands for `no limit`, all read queries are divided up into packages of 100 or fewer.

Field	Description
	Note This setting is also relevant for browsing. When the address space of OPC UA servers is browsed, read service calls are performed in PCo for reading attributes. The number of nodes that are read is a multiple of the number of nodes returned during browsing. If you get the result `BadTooManyOperations` during browsing, you can use this setting to try to help resolve the problem. You ought to be able to take an appropriate value from the server documentation; otherwise you need to find a value by trial and error.
Package Size for Write Service	You can use this setting to define that, for a write query, the query to the OPC UA server is divided up into multiple packages of the maximum size specified here. The value 0 also means here that the OPC UA client does not make any restrictions but rather tries to process each write query with only one individual query to the server. As with the read service, the server can publish a limit to the number of nodes that is valid when the write service is called. The PCo OPC UA client queries this information and uses it, if available, to restrict the write service.
Package Size for Browse Service	You can use this setting to define that the browse service calls of an OPC UA server are always packaged, thereby limiting the number of nodes that are read in an individual call. As in the other cases, a package size of 0 means that the OPC UA client does not restrict the number of tags during browsing. As with the read service, the server can publish a limit to the number of nodes that is valid when the browse service is called. The PCo OPC UA client queries this information and uses it, if available, to restrict the browse service.

Field	Description
Tolerant Type Checking	You use this indicator to specify that you want PCo to deactivate specific consistency checks. This is only required in rare cases if an OPC UA server does not provide specific attributes of individual nodes. The OPC UA agent instances in PCo try to read these attributes to perform consistency checks, for example, when an agent instance is started or when subscription items are registered. If determination of these attributes fails, the agent instance terminates the start. In such cases, setting this indicator might help to narrow down the cause of the problem. When you have set the indicator, check if the agent instance starts. To narrow down the cause of the problem even further, set the *log level* to at least *Warning* and search for `tolerant type check` in the log. **⚠ Caution** This indicator should not be set for productive operations. The consistency checks that are deactivated by the indicator are, from the viewpoint of SAP, an essential prerequisite for stable running of the software.
Support for Structured Data Types	If you select the checkbox, processing of structured data types is enabled and the *Structures* tab is ready for input. A structured data type describes a data type that consists of several components. The component of a structured data type has a name, a data type, and a namespace as attributes. The data type of the component might itself be a structured data type. In the simplest case, the data type of a component is a simple data type, for example, `string`, `double`, or `integer`. In the context of OPC UA servers, the term **component** corresponds to the term **field**. **→ Recommendation** Only select this checkbox if you really want to use structured data types, since this involves additional service calls. In this case, we recommend that you choose - on the *Structures* tab - all data types that you want to use. (See also: OPC UA Source System: Structures Tab [page 167].)

Field	Description
Support for Abstract Data Types	If you select the checkbox, abstract data types, such as `Number`, `Integer`, or `UInteger` , can be read. **i Note** The field is **only relevant for reading** tags. You **cannot** select tags with an abstract data type in the browser of the query destination system.
Acceptable Status Code	You can specify here that the tags that are to be read are to be filtered using the status code. The status code defines the minimum quality of the tag values. The following options are available: • Any Status Code In this case, all tag values are read. • At Least Uncertain With this setting, only the tag values whose status codes are good or uncertain are read. • At Least Good This is the default setting. With this setting, only tag values whose status codes are good are read. **i Note** The setting of the acceptable status code only affects the reading of tags in the query scenario.
Pending Requests	This field shows the maximum number of pending requests that are allowed before PCo tries to reconnect to the OPC UA server. Pending requests are requests for which the response of the OPC UA server is still pending. PCo regularly sends request messages to the OPC UA server and waits for response messages from the server. PCo only attempts to reconnect when the limit set here is exceeded. The default value is 10, which is usually sufficient. You should only increase this value if the computer is running on PCo, if it is not fast enough, and if the OPC UA server is overloaded.

Tag Persistence

By selecting the *Use URL for Storing the Namespace Information of Tag Node IDs* indicator, you specify that the namespace URL is always to be used for storing node IDs of tags. When a new OPC UA source system is created, the checkbox is selected as standard. The setting made here affects the source system and all the agent instances derived from it.

- If you select this checkbox, the namespace URL of the tag is adopted automatically when the tags are selected within a subscription, or the alias configuration, or in the variable assignment of a notification enhancement.

  > ### Example
  >
  > For a subscription, the URL is displayed in the namespace on the *Subscription Items* tab for the selected tag:
  >
  > Name: `Tag1`
  >
  > Source:`nsurl=<NamespaceURL>;s=Channel1.Device1.Tag1`.

- If you do **not** select the checkbox, the namespace index for storing the namespace information of the tag node ID is used. This information can become invalid for OPC UA servers that assign their namespace indexes dynamically.

  > ### Example
  >
  > In this case, the subscribed tag on the *Subscription Items* tab looks as follows:
  >
  > Name: `Tag1`
  >
  > Source: `ns=2;s=Channel1.Device1.Tag1`. The namespace is identified here using a namespace index and specified with ns=2.

> ### → Recommendation
>
> SAP recommends that you select this checkbox. The checkbox is **not** set for existing OPC UA source systems from release 2.3 or earlier. SAP recommends that you also select the checkbox for existing OPC UA source systems. The existing OPC UA source systems must then be migrated if these source systems or the agent instances generated with them have configuration data that is based upon the old notation (namespace index), that is, if there are alias configurations, subscription items, or callbacks in notification enhancements in the source system or its agent instances. If none of this applies, you can select the checkbox manually at any time and continue working directly with the source system. Otherwise, you need to delete and create again all alias configurations, subscription items, or callbacks for notification enhancements.

Test Connection

To test the session configuration, choose **Test Connection**.

> ### ℹ Note
>
> Before the test, you need to maintain the settings on the *Security* tab. A temporary session is established to validate the security and configuration settings.

More Information

Select Endpoint [page 151]

Specifications of the OPC Foundation: https://opcfoundation.org/developer-tools/specifications-unified-architecture

4.1.1.3.1.1 Select Endpoint

Use

When you choose the *Select Endpoint* pushbutton, you can choose (from a list of known server endpoints) the endpoint with which you want the OPC UA agent to connect.

Prerequisites

- You have specified a **discovery server** on the *Session* tab.
- You have created the **list of server endpoints** using the local discovery server. To create the list, choose *Start Discovery*. The system determines the possible endpoints and proposes an endpoint in the *Server Endpoint* field.

Procedure

Select Endpoint

1. Choose *Select Endpoint* on the *Session* tab.
   The *Configure Endpoints* dialog box appears, containing the list of possible endpoints.
2. Select the endpoint you want from the list and click *OK*.
   PCo adopts the selected endpoint server and displays it in the *Server Endpoint* field.

Add New Endpoint

1. If the endpoint you want is not in the list of endpoints or the list of endpoints has not been set up correctly, choose *Add New Endpoint* here in the *Configure Endpoints* dialog box to enter a new entry manually.

PCo displays the *Add Server Endpoint Description* dialog box. The dialog box contains the following fields with predefined values, which you can change:

Field	Description
Endpoint URL	When you create a new entry or change an entry, you need to enter a valid URL here. A simple validity check is performed. A valid URL starts with one of the following three character strings: ○ opc.tcp:// ○ http:// ○ https:// Then comes the **URL of the server**, using, ideally, a fully qualified domain name. This is followed – separated by colons – by a **port number** and, optionally, **path details**. **Example** `Opc.tcp://myserver.mydomain.corp:59000/OpcUA/Server42/` **Note** The selection options of the following three fields depend on the structure of this entry.
Security Mode	The security mode defines which steps are used for a secure connection setup (OpenSecureChannel request). You can choose between the following settings: ○ None For this setting, the request is neither signed nor encrypted. In this case, no certificates are used for a secure connection setup. ○ Sign For this setting, the request is not encrypted but is signed with the private key of the client application certificate so that the server (that has to trust the client certificate) can validate the request. ○ SignAndEncrypt For this setting, the client uses the public key of the server to sign the message and to encrypt it. **Note** These settings only have an effect on the connection setup or on later renewals of the connection.

Field	Description
Security Policy	The security policy allows the **cryptoalgorithm**, to a certain extent, to be chosen for setting up the secure connection. The following options are possible depending on the *security mode* defined previously: ○ Security mode *None*: `None` ○ Security modes *Sign* and *SignAndEncrypt*: The following security policies are available: ○ `Aes128_Sha256_RsaOaep` This security policy is intended for configurations with medium security requirements. ○ `Aes256_Sha256_RsaPss` This security policy is intended for configurations with high security requirements. ○ `Basic256Sha256` This security policy is intended for configurations with high security requirements. For more information about these security policies, see https://apps.opcfoundation.org/profilereporting/ in the section *Security Category* *SecurityPolicy*. There are also the deprecated security policies `Basic128Rsa15` or `Basic256`. They appear in the dropdown list with the addition of the word `(deprecated)`. **→ Recommendation** It is recommended that you select the security policy `Basic256Sha256` or `Aes256_Sha256_RsaPss` in all scenarios (except test scenarios) if the processing power of the client and the server permits this.
Encoding	The following options are available for encoding: ○ (OPC UA) Binary ○ (OPC UA) XML XML is only available if the endpoint URL starts with `http` or `https`.

2. Make your settings and confirm by choosing OK.
PCo generates a new entry in the list of endpoints. This entry consists of the URL of the endpoint and the settings for the *Security Mode*, *Security Policy*, and *Encoding* fields.

Example

```
opc.tcp://pwdf3200:51210/UA/SampleServer - [None:None:Binary]
```

3. To add the endpoint, confirm with *OK*.

> **i Note**
>
> If an endpoint is entered manually, no matching with a server takes place. However, certain aspects of the desired connection can only be determined reliably by such a matching. If you enter or manipulate the URL manually, therefore, you should always update the endpoint by setting up at least one test connection for which the corresponding setting has been selected.

Edit Endpoint

Choose *Edit Selected Endpoint* to adjust an existing entry in the list of endpoints.

PCo displays the *Add Server Endpoint Description* dialog box. The selected server endpoint is shown in the *Endpoint* field.

> **⚠ Caution**
>
> If you choose the *Edit Selected Endpoint* pushbutton, you overwrite the selected endpoint. If you apply the changes in this dialog, these changes are retained, even if you then choose *Cancel* in the higher-level dialog.

4.1.1.3.2 OPC UA Source System: Security Tab

Use

On this tab, you make the settings for secure connections and user authentication for OPC UA source systems.

To be able to identify the PCo system as a client to the OPC UA server and vice versa, *X.509 v3 certificates* are used, provided a secure connection is to be set up. In the context of OPC UA, the certificates used here are called *application certificates*.

Procedure

Certificates

1. You can generate and assign an application certificate for the Application Certificate [page 157] field.

   > **i Note**
   >
   > If you only need certificates for test purposes or work in an environment where no certificates, which have been issued by an official Certification Authority, are required, you can also generate the certificates here using the PCo certificate generator.

   PCo can identify itself to its OPC UA server using the self-signed certificate. To generate a certificate, choose the icon *Generate and Assign Application Certificate*.
   The *Generate Self-Signed OPC UA Client Certificate From Defaults* dialog box appears and you can make the settings for certificate generation here. See also: Generate and Assign a Self-Signed Certificate [page 158].

2. Choose *Change Application Certificate Assignment* to select and assign another certificate in the *certificate browser*.
3. Specify the *identification type* for the selected application certificate. If you select the *Identification by Subject* option, certificate rotation is supported. (See also: Identification Type of Certificates [page 593].)
4. If you choose *Remove Application Certificate Assignment*, the assignment of the generated certificate is removed. However, the certificate remains in the Microsoft certificate store and can be selected for other OPC UA source systems.
5. Choose the *Validation Options* pushbutton to define which checks are to be performed for the server certificates. For more information, see Validation Options for Server Certificates [page 161].
6. Select the *Send Certificate Chain* checkbox if the application certificate of the agent instance has been signed with a root certificate and is embedded in a certificate chain. This option allows you to control whether the agent instance should try to make this chain and send it to the server when a secure connection is being set up. In this case, the agent instance searches recursively in specific certificate stores for the certificate with which the application certificate or the CA certificate that was found last has been signed, and then sends the certificates that it has found to the server. The server needs to retain the missing certificates for a validation. Not every server supports the receipt of certificate chains. Deselect the checkbox if you want to connect the OPC UA agent instance with one of these servers. In this case, you need to make the certificate chain known to the server manually, if necessary.
   If you do not select the checkbox, the agent instance only sends its application certificate.

Session Authentication

In this screen area, you define the settings for authentication of the OPC UA session:

Field	Description
Authentication Mode	The authentication mode is used to authenticate the user session to the OPC UA server session. The following authentication modes are available: • *Anonymous* No additional entries are required for this setting. • *Certificate* If you choose this option, you can use the *Add* pushbutton to select a **session certificate**. • *User Name and Password* If you choose this option, you can enter a user name and a password.
Session Certificate	Here you can select an X.509 v3 certificate that is to be used by the OPC UA server to authenticate the user session. This field is only ready for input if you choose the *Certificate* option in the *Authentication Mode* field.
Identification Type	Select the identification type for the selected session certificate. (See also: Identification Type of Certificates [page 593].)

Field	Description
User Name	User name that is used by OPC UA to authenticate the user when establishing a secure session. This field is only ready for input if you choose the *User Name and Password* option in the *Authentication Mode* field.
Password	Password that is used by OPC UA to authenticate access to a user-specific session.

Certificate Storage Configuration for the Application Certificate of the UA Server

Store for Trusted Server Certificates

You enter the store type and the folder you want here. You can configure the store location for the certificates, which the OPC UA agent is to trust, to the granularity of the source system. You have the following options:

- *Store Type* **`Microsoft Certificate Store`**
  When a connection is being established, with this setting, an OPC UA client automatically searches in the Microsoft Certificate Store folder for a server certificate. You can select specific folders of the *Microsoft Certificate Store* here.
- *Store Type* **`File System`**
  With this option, you can specify the store location for the certificates, which the OPC UA agent is to trust, in the file system. You can specify specific directories in the directory tree. In this case, a subfolder is offered by default in the directory that is usually used under *MS Windows* for storing all-user configurations.

  > ⚠ Caution
  >
  > Note that this MS folder usually has the attribute *Hidden*.

  By choosing the *File System* setting, you define where the OPC UA agent is to search for the server certificate with a public key.

  > i Note
  >
  > This option does not copy any certificates into this store location. You need to copy the certificates into this store location yourself.

  If you choose a directory in the *Folder* field, the directory and the subfolder *certs* are created for this directory.

  > ⚠ Caution
  >
  > If PCo is changed or uninstalled, the directory and the subfolder *certs* are not deleted.

For a secure connection, the server uses either a self-signed X.509 v3 certificate or an X.509 v3 certificate that is embedded in a hierarchy of certificates. When the connection is being set up, the server sends the application certificate to the client in each case. The entire chain needs to be available to the client for the validation. At least one certificate from the chain needs to be in the store for trusted certificates and have been stored there accordingly by the system administrator in the subfolder *certs*. The remaining certificates can also be stored in the store for trusted issuers or be sent from the server.

> **i Note**
>
> Not all servers support the sending of a chain of certificates.

Store for Rejected Server Certificates

If an OPC UA client wants to set up a secure connection to an OPC UA server, he or she receives a certificate with a public key from the server. The client accepts this certificate if he or she regards it as trustworthy (see the previous point).

Otherwise, the certificate is stored in the *store for rejected server certificates* if it is valid. With this setting, you can define the store location for rejected server certificates. If the server is using a self-signed certificate, you can, after an unsuccessful connection attempt, copy the certificate from this store location to the *store for trusted certificates*. This establishes the trust relationship between the server and client on the PCo side. You need to make a root certificate known in another way, for example, manually.

Store for Trusted Issuer Certificates

If the application certificate of the UA server is embedded in a certificate hierarchy, the related root certificate needs to be available to establish a trust relationship to the server. You need to store this root certificate in the subfolder *certs* of the directory that can be configured with this option.

As in the case of trusted server certificates, this directory should only be writable for system administrators if it is created in the file system. After an unsuccessful connection attempt, however, you do not find the root certificate in the *store for rejected certificates*.

The root certificate can also be stored in the store for trusted certificates. In this case, the application certificate must not be stored there. This allows you to easily set up a trust relationship to multiple servers. In this case too, the complete certificate chain must be known to the OPC UA agent instance when a certificate chain is used.

If the server certificate is not valid, for example, because the validity interval is in the past and you want to take the opportunity to suppress a failed validation of the certificate, the invalid certificate needs to be stored in the *store for trusted certificates*, even if there is a valid root certificate for it. In this case too, however, the server certificate is not in the *store for rejected certificates*. Instead, you need to import it from the server.

4.1.1.3.2.1 Application Certificate for OPC UA Source Systems

Use

X.509 V3 Certificates for a Secure Connection

To be able to set up a secure connection, you need to have chosen the *security mode* **`Sign`** or **`SignAndEncrypt`** for the endpoint. In this case, in PCo, you need a valid X.509 v3 certificate. This certificate can either be self-signed, or the validity of the certificate needs to be guaranteed by a trusted certification authority (CA).

You can generate and automatically assign a self-signed certificate directly in the application. These self-signed certificates are stored as application certificates with a private key in the Microsoft certificate store. You can decide yourself whether you want to store the certificate under *current user* or under *local computer* in the

certificate store. In this area, you can still select the folder. For more information, see Generate and Assign a Self-Signed Certificate [page 158].

In both cases, the service user under whose user account the OPC UA agent instance is running, must be able to access the private key.

> **i** Note
>
> If you don't want to use the function for generating self-signed certificates, you need to import the application certificate for the PCo agent instance together with the private key into the Microsoft certificate store before usage in PCo. No other option is envisaged for providing the certificate for the OPC UA agent instance.

The certificate needs to enable the following usages:

- digitalSignature
- nonRepudiation
- keyEncipherment
- dataEncipherment

> **i** Note
>
> In the case of self-signed certificates, the usage **keyCertSign** must also be envisaged.
>
> These requirements are fulfilled automatically for the certificates generated in the application.

If you generate the certificates yourself, another URI, which identifies the application, should be given in the **Subject Alternative Name** field of the certificate.

Further Information

For more information, see https://opcfoundation.org/developer-tools/specifications-unified-architecture in *Part 6: Mapping, Version 1.03, Section 6.2.2. (Table 23)*.

4.1.1.3.2.2 Generate and Assign a Self-Signed Certificate

The *Generate and Assign Application Certificate* icon is available for the following objects:

- OPC UA source system
  You can choose this option in the *application certificate* screen area on the *Security* tab for the OPC UA source system. You can then set the parameters for certificate generation in the *Generate Self-Signed OPC UA Client Certificate From Defaults* dialog box that appears. The generated certificate is assigned automatically to the OPC UA source system as an application certificate. (See also: Application Certificate for OPC UA Source Systems [page 157].)
- OPC UA server
  You can choose this option in the *application certificate* screen area on the *Security Configuration* tab. You can set the parameters for certificate generation in the *Generate Self-Signed OPC UA Server Certificate*

From Defaults dialog box that appears. The generated certificate is assigned automatically to the OPC UA source system as an application certificate.

Mandatory Subject Components

This screen area contains the mandatory fields for generation of the certificate. Each certificate has an attribute **subject** of type `X.509DistinguishedName` in which the entity to which the certificate is assigned is described in more detail.

Define Mandatory Attributes

Field	Description
Common Name (CN=)	You enter the common name for the certificate here. In the case of **OPC UA source systems**, CN= is prefilled with the value **SAP PCo OPC UA client <name of the source system>** and in the case of OPC UA servers with the value **SAP PCo OPC UA client <name of the agent instance>**. **i Note** The abbreviation CN= is generated automatically. Do not enter this.
Organization (O=)	Enter a name here that describes the organization that operates the application, for example, the name of the company. You must make an entry here in order to comply with the OPC UA specification. **i Note** The abbreviation O= and all the other abbreviations below are generated automatically. Do not enter this.

Optional Subject Components

This screen area contains additional fields with attributes that you can fill or leave empty.

Define Optional Attributes

Field	Description
Locality (L=)	Freely definable
Country (C=)	Enter, for example, DE for Germany. A valid two-character country code is expected.
Organizational Unit (OU=)	Freely definable
Domain Component (DC=)	Enter the host name or the fully qualified host name of the computer here.
State or Province (S=)	Freely definable

Field	Description
Friendly Name	A name of your choice that you can assign to the certificate. The friendly name can, for example, make it easier to recognize the certificate (for example, in the certificate store). There is a separate column for this in the selection dialog for certificates.

Technical Settings

Define Technical Settings

Field	Description
Key Size (bit)	Specifies the size of the private key. 2048 bit is the default value.
Valid From Today Until	You define the validity end date of the certificate here. The default validity is one year.

Microsoft Certificate Store

Settings for the Microsoft Certificate Store

Field	Description
Certificate Store	The certificates are always stored as application certificates with a private key in the Microsoft certificate store. You can decide whether you want to store the certificate in the private user area or in the machine-specific storage area: • Current user • Local computer
Certificate Folder	You can select the folder here: • Personal • UA Applications • UA Machine Default
Private Key Is Exportable	If you select this checkbox, the certificate and the private key can be exported. If you do **not** select this checkbox, you **cannot** save the certificate with the private key outside the Microsoft certificate store. The advantage of this setting is greater security because the key cannot be stolen and therefore cannot be used on another device. The disadvantage is that you cannot make a backup of the private key. If the computer is defective, for example, you have to generate new certificates and also set up new trust relationships for the certificates.

Alternative Names

For OPC UA servers, the computer on which the server is running must be specified in the application certificate. The preferred method is to provide a DNS name or an IP address in the *Alternative Names* screen area. For certificate generation for OPC UA servers in PCo, you must therefore either enter the DNS or the IP address. Otherwise the certificate cannot be generated. You can enter one or more DNS names or IP addresses here that need to match the endpoints of the server. The fields can include lists that are separated by commas.

> **i** Note
>
> You cannot enter an application URI or other permitted entries for the alternative name. When certificate generation is used, an application URI is generated automatically.

Settings for Alternative Names

Field	Description
DNS	You enter the name or names for the domain name system here. For OPC UA servers, the default value for the domain name is the current computer name.
IP Address	Enter the IP address(es) of your computer here if you are using the IP address for the endpoint definition of an OPC UA server. It must match with the endpoint of the server. The IP address is the unique identification characteristic of a computer that defines its location in the Internet.

4.1.1.3.2.3 Validation Options for Server Certificates

Use

When you choose the *Validation Options* pushbutton, a dialog box appears in which you can define the behavior of the OPC UA source system in specific exception situations during certificate validation.

When a secure connection is being established, the OPC UA server must identify itself to the OPC UA client using a valid X5093 V3 certificate. In this case, the OPC UA client is the OPC UA agent instance of the PCo system.

Features

Empty Certificates

The OPC UA specification stipulates that the certificate must have a valid structure or must be empty. The following options are offered to define how to handle empty certificates:

- *Allow Empty Certificates*

This setting allows you to accept an empty certificate. With this exception, it is possible to do without certificate handling, and thus save the required storage space, for UA servers on devices with limited computing power.

> **i Note**
>
> It is **not** possible to set up a secure connection with an empty certificate, so in this case you need to select the entry `None` in the *Security Mode* field for the endpoint.

- *Allow Empty Certificates with Warning*
  This setting allows you to accept an empty certificate. In addition, a warning is issued that is written to the log when the agent instance is started.
- *Reject Empty Certificates*
  You use this setting to define that empty certificates are not accepted.

Suppress Validation Errors

In this screen area, you can define that errors that occur for specific certificate validations are to be suppressed:

- *Validity Period*
  With this setting, you define that the server certificate can also be accepted even if it has already expired or is not yet valid. If the validity of the server certificate has expired, the certificate is still accepted. However, a corresponding warning is written to the log.
- *Host Name*
  With this setting, you define that server certificates that cannot be identified correctly can also be accepted. In this case, too, a warning is written to the log if necessary.

  > **i Note**
  >
  > As an alternative, for the endpoint that you have configured for the server, you can try to specify the server in the same way as it appears in the attributes of the server certificate.

- *Trust Check for Server Certificate*
  If the certificate sent from the server when the connection is being set up is a valid certificate but you have not defined a trust relationship (in this case, the server certificate is not stored in the store for trusted certificates), you can nevertheless allow a connection to be set up by selecting the *Trust Check for Server Certificate* checkbox.

  > **⚠ Caution**
  >
  > With this setting, you can now also set up a connection with message encryption and signing enabled. However, since the server certificate is not trusted, this is still not considered to be secure communication.
  >
  > This option is not provided in the OPC UA specification and should only be used for tests. Under certain conditions, it might be the case that this option is not effective and that the connection setup is rejected. This is the case, for example, if the server certificate is invalid or if the server endpoint for the session authentication does not allow the *authentication mode* `Anonymous`.

> **i Note**
>
> For the options *Validity Period* and *Host Name*, a failure in the certificate validation is suppressed. This procedure corresponds to the specifications of the OPC UA specification. In the current implementation of PCo, the server certificate must be stored for this purpose in the store for trusted certificates. This also

applies if you are using a derived certificate in a hierarchy, and the root certificate is already stored. Otherwise, the connection setup fails.

Displaying Suppressed Validation Errors

You can display a summary of validation errors of certificates in a dialog box, even if you have set *Suppress Validation Errors*. PCo displays the error messages for the affected certificates when you choose the *Test Connection* pushbutton of an OPC UA source system. The message contains a list of the errors that were suppressed, followed by the certificate name.

4.1.1.3.2.4 Possible Causes of Connection Setup Failure

For PCo 15.1, a new version of the *OPC library*, which sets stricter standards when checking certificates than was the case until now, has been integrated into PCo. This means that even if you use an endpoint with the *security mode* `None`, a certificate check might fail, even though the configuration you are using worked in earlier releases. In this case, the connection to the OPC UA server is not set up. You get the following message:

```
Session Creation failed [BadCertificateHostNameInvalid].
```

This message usually appears if there is no match between the host name used in the server endpoint and the host name that the server uses in its application certificate. If the server simply uses only its host name `<serverhost>` in its certificate while the endpoint uses the fully qualified host name, the server certificate is not trusted and the connection cannot be set up.

Example

Example of a fully qualified server name:

```
<serverhost>.<mylocation>..<mycompany>.<topleveldomain>
```

To solve the problem, you can do the following:

- Modify the host name in the endpoint definition
- Modify the host name in the server certificate
- Store the application certificate of the server in the *Store for Trusted Server Certificates*, and on the *Security* tab for the OPC UA source system, under *Validation Options*, select the option *Host Name* in the *Suppress Validation Errors* screen area.

Note

SAP recommends that you use the fully qualified host name both in the definition of the server endpoint and in all application certificates.

4.1.1.3.3 OPC UA Source System: Subscription Tab

Procedure

1. On the *Plant Connectivity Management Console* screen, select an OPC UA source system and click on the *Subscription* tab.
2. For the information you need to enter, use the following table:

Field	Description
Area: Subscription Parameters	
Display Name	Name of subscription list. For OPC UA, the subscription list contains the tags to be monitored.
Event Interval (ms)	With this interval, you specify how often value changes are to be reported.
Keep-Alive Interval (sec)	If there is no value change, the UA server sends a message to PCo to this effect. Here you specify the interval at which a message is to be sent.
Lifetime Interval (sec)	Indicates the lifetime of a subscription when the session is interrupted so that the subscription can also be used by another client.
Maximum Notifications Per Event	Here you specify the maximum number of value changes per event that are to be passed on. In the case of OPC UA, an event can contain several value changes.

Field	Description
Acceptable Status Code	You can specify here that you want **notification messages** to be filtered using the status code. The status code defines the minimum quality of the notification messages sent from the OPC UA source system. **⚠ Caution** The setting of the acceptable status code only affects notification messages and **not** tag queries. The following options are available: ○ **Any Status Code** This is the default setting. In this case, all notification messages are processed. ○ **At Least Uncertain** With this setting, only notification messages whose status codes are good or uncertain are forwarded. ○ **At Least Good** With this setting, only notification messages whose status codes are good are forwarded. If you set the `log level` to *Information* or *Verbose* on the *Host* tab of the agent instance, an entry is written to the log when a notification message has been filtered out due to its status code. **⚠ Caution** If you use the setting *At Least Good* or *At Least Uncertain*, you are no longer informed about changes to tags if these have been filtered out due to their status code. Therefore, you cannot rely on the current value of the tag in question corresponding to the value from the last notification message.

Field	Description
Enable Edit Mode for Subscription Items	If you select this indicator, you can enable editing for subscription items for the agent instance of your OPC UA source system. It is not normally possible to manually enter or change subscription items in the case of OPC UA servers because the server is prone to error. Therefore, it is recommended that you use the *Browse* dialog to browse the address space of the server and select the tags that you want to subscribe. This is why the pushbuttons for manual editing of subscription items for OPC UA agent instances are not active as a rule. However, individual OPC UA servers do not allow you to browse the address space completely and so it is the case that specific IDs represent an entire group of tags and that individual elements of the group need to be accessed using a special syntax. This is described in the server documentation. In such a case, this indicator lets you enable manual editing of the subscription items for the agent instance.
Area: Monitored Item Parameters	
Queue Size	Here you specify how large the queue of value changes may be. The individual value changes are placed in the queue.
Discard Oldest	With this indicator, you specify that the oldest values in the queue are to be discarded. Since new value changes are continually placed in the queue, it makes sense to discard the oldest values.
Area: Retry Settings for Node Access After Server Restart	In this screen area, you define the settings for a new connection to the OPC UA server after it has been restarted. These settings allow a waiting period that enables the PCo system to go into a queue and to try to connect to the OPC UA server again for node access. This is a setting that is considered a workaround and should not normally be required.
Max. No. of Retries	Here you specify the maximum number of connection attempts to the OPC UA server. The default value is 0.

Field	Description
Retry Interval	Here you specify the number of milliseconds before the next connection attempt is made. The value *0* is set by default, which means that the agent switches to the *Faulty* status if it cannot read the node classes for its subscriptions.

4.1.1.3.4 OPC UA Source System: Structures Tab

Use

You use this tab to select structured data types. A structured data type describes a data type that consists of several components. The component in the OPC UA environment corresponds to a field.

You can use structured data types as follows:

- In conjunction with retrieve and store queries (see: Query Destination System [page 375].)
- As subscription items

For more information about functional limitations, see Functional Limitations [page 171].

In order that structured data types can be used, the OPC UA agent instance needs to request data type information from the server at runtime. This task is only performed semiautomatically in SAP PCo in the current release. It is technically possible to request this information if required, in other words, whenever the PCo OPC UA agent instance wants to store or retrieve at runtime. However, because the PCo UA agent instance communicates with an OPC UA server using mass queries, it doesn't make sense to check after such a query if all the data types have been recognized correctly, because this would lead to client-server roundtrips.

Instead, at design time you need to specify which structured data types you want to be used at runtime. You do this using the browse dialogs on the **Structures** tab. The type information is loaded for the data types that you select here after the agent instance has started and can be used at runtime.

Activating the Function

To activate the function, you select the checkbox *Support for Structured Data Types* on the *Session* tab (see also: OPC UA Source System: Session Tab [page 143]).

Procedure

1. Choose the *Add New Structured Data Type* pushbutton.
   The *Browse OPC UA Structures* dialog box appears in which you can browse the address space of the server. There are two tabs:
   - *Variables* tab
     You can navigate here to a node that you want to read at runtime or to which you want to write at runtime.
   - *Types* tab
2. If you select a node with the node class **Variable**, the following information is displayed for the selected node in the lower part of the dialog box:
   - Node ID
     The ID of the node is displayed. The ID consists of ID type, namespace URI, and identifier.
   - ID of the relevant data type
   - Data type
     The data type is displayed in the status bar. Here you can see which data type the node is:
     - Simple data type, for example, `string`, `double`, or `integer`
     - Object data type
     - Structured data type

   You can only select nodes here that correspond to an instance of a structured data type. From a technical viewpoint, this means that the data type **structure** is derived from the node with the ID ns=0; i=22. Data types from the namespace of the **OPC Foundation** (**http://opcfoundation.org/UA/**, this is the namespace with the index 0) should already be known and cannot therefore be selected.
3. Hierarchies of structured data types and abstract structured data types are supported. However, you must explicitly select all concrete data types that can occur at runtime. (See: Example: Configure Derived Data Types [page 170].)
4. Select a node and confirm with OK that you want the data type that belongs to this node to be used at runtime.
   The node is added to the list of data types that are to be used.
5. Instead of selecting the data type using an instance, you can also select the type directly on the **Types** tab. To do so, switch to the **Types** tab.
   This tab displays the type hierarchy on the OPC UA server below the structure data type. The details for the node type are displayed in the lower area of the dialog box.
6. Browse the tree on the *Types* tab until you find the data type you want. Select it and choose OK.
7. Once you have selected the types, you can display your selection in a hierarchical view on the *Structures* tab by choosing *Show Structured Data Types Hierarchy*.
   Your chosen data types are selected.
8. If you want to use structured OPC UA data types, which you have configured here, in a tag-based notification, you can import the OPC UA data types into the PCo data type repository and thereby generate the PCo data types that correspond to the OPC UA data types. (See also: Import OPC UA Data Types into the PCo Data Type Repository [page 169].)

Related Information

Behavior at Runtime [page 170]

Save the JSON Template of the Selected Structures [page 169]
Behavior at Runtime [page 170]
Functional Limitations [page 171]

4.1.1.3.4.1 Import OPC UA Data Types into the PCo Data Type Repository

Use

By importing structured OPC UA data types into the PCo data type repository, you generate corresponding structured data types and array PCo data types that you can use in subscription items, output expressions, input and output variables of specific destination system types, and in variables of multiple call destination systems. At runtime, the PCo agent instance is therefore able to convert OPC UA data objects into PCo data objects in a type-secure way. You can also perform a type conversion of data objects by defining data types in output expressions or variables in multiple call destination systems.

You can use the expression editor to access components of structured data objects. You can display the related functions in the expression editor in the *Functions* screen area using the function category *Structured Data Type Instances*.

Procedure

1. On the *Structures* tab, define the structured data types that you want to use. (See also: OPC UA Source System: Structures Tab [page 167].)
2. Choose the *Import Data Types into Data Type Repository* button.
   The system collects all OPC UA data types and uses them to generate proposals for corresponding structured and array PCo data types. These proposals are displayed in the *Import Data Types into Data Type Repository* dialog box. The dialog box also shows you whether or not OPC UA data types and corresponding PCo data types are already known in the PCo data type repository.
3. Continue processing as described under Functions in the Import Data Types Dialog Box [page 587].

4.1.1.3.4.2 Save the JSON Template of the Selected Structures

If no instance of the data type is available, you can generate an empty instance. To store a template in the file system, choose the *Save JSON Template of Selected Structures* pushbutton on the *Structures* tab.

The template that you generate is initialized with default values. To correctly initialize attributes that have structures as data type, you may also need to select the included data structures in order to include them in the generation of the template.

See the configuration of the server to find out if data types contain structured data types.

Related Information

Behavior at Runtime [page 170]

4.1.1.3.4.3 Example: Configure Derived Data Types

OPC UA supports the inheritance of data types. To be able to use derived structures, you may need to perform additional configuration steps:

If the server defines, for example, the data type **VehicleType**, it might also define the derived types **ElectricVehicleType** and **DieselVehicleType**. Using a type query, it could then ensure that, in the model on the server, instances of electric vehicles can only be refuelled at an electric socket and diesel vehicles only at diesel pumps.

On the other hand, one service parking space could be used by both types. For an object that represents a service parking space, it therefore makes sense to allow a reference **being serviced** to a **VehicleType**, without requiring a specialization to **ElectricVehicleType** or **DieselVehicleType**. At runtime, when a vehicle is being serviced in the model, the actual type of the current instance can be queried.

If, in such a scenario, you use **VehicleType** to model the data types for which you require support and then, when browsing the server via the instance search, choose the data types by selecting the variables, only **VehicleType** is stored as the data type to be supported. If an instance of **ElectricVehicleType** is then stored there at runtime, the actual type is not recognized automatically.

To make this possible, you need to explicitly select the derived types that you want to use after selecting the data type **VehicleType**. To do this, browse the type hierarchy on the *Types* tab and select the types you want. This means that all the types that you want to use need to be known at design time.

4.1.1.3.4.4 Behavior at Runtime

You can use structured OPC UA data types, which you have configured on the *Structures* tab, in store and retrieve queries and in subscription items for notifications. (See also: Query Destination System [page 375].)

If you want to use structured data types in queries, this data is formatted as a JSON string in PCo at runtime. This means that you receive the data in a retrieve query as a JSON string and, in return, you have to transfer the data for a store query as a JSON string.

The transformation of the instance of the structure into a JSON string occurs internally in PCo for a retrieve query. The JSON format is defined in PCo. To generate JSON strings from structured data objects, you can use the expression editor function `struct2json`.

You can generate a template for the write operation by reading an instance of the relevant structure and storing it at a location of your choice. (See also: Save the JSON Template of the Selected Structures [page 169].)

The number of characters that can be used for OPC UA names is greater than the number of characters permitted in JSON structure descriptions. Therefore, when the names of the attributes are transformed, there may be deviations. For example, the name in the PCo-internal JSON display might differ from the name on the OPC server. For that reason, you should always save an example of the instances in JSON format and use the naming convention in the example as a guide.

If you want to use structured OPC UA data types, which you have configured on the *Structures* tab, in a tag-based notification scenario, you can import the OPC UA data types into the PCo data type repository and thereby generate the PCo data types that correspond to the OPC UA data types. (See also: Import OPC UA Data Types into the PCo Data Type Repository [page 169].) You can then select the corresponding structured PCo data types or arrays of structured PCo data types in subscription items. The PCo agent instance converts the data objects into typed, structured PCo data types or array PCo data objects at runtime.

4.1.1.3.4.5 Functional Limitations

Namespace

Structured data types are defined in data libraries, called `Data Dictionaries` in the OPC UA environment. A data library is identified uniquely using its namespace URI. According to the OPC UA specification, a version number can also be issued, but this happens rarely and at a given time the namespace is unique in this sense.

At runtime, the namespaces are identified using a namespace index generated by the server. The namespace index can differ from session to session. You must therefore always use the namespace if you want to uniquely identify a data type. You can display the namespace (in which a data type is defined) in the dialog box with which you can browse the data types.

The namespace with the URI `http://opcfoundation.org/UA/` is always known and always has the namespace index 0. Data types from this namespace can be used in structured data types and be processed by PCo. If, on the other hand, you use data types from another namespace, for example, **http://someOpCUaVendor.com/UATypes**, these are only supported in SAP PCo if you are not using any attributes from other namespaces except from **http://someOpCUaVendor.com/UATypes** and **http://opcfoundation.org/UA/**.

Further Limitations

- If structured data types reference other structured data types, the referenced data types need to be defined either in the namespace of the OPC Foundation (**http://opcfoundation.org/ua/**) – this is the namespace with the index 0 – or in the same data library as the type being referenced.
- If you apply store or retrieve queries to structured data objects, PCo always stores or retrieves the entire structure as a JSON string. If you want to access individual components of the structure, you must do this

in query preparation or postprocessing using the functions of the expression editor. You can, for example, use output expressions or a multiple call destination system.

- Structured data types are not supported in the OPC UA destination system and for input and output parameters of methods of the PCo OPC UA server.
- Structured data types are not supported in historical retrieve query calls.
- If you use the cache mode *Access Using Aliases Only* for the agent instance, structured data types are not supported.
- With PCo, you can only access application-specific structured data types. The application-specific `Unions` provided in the OPC UA specification are not supported.
- Enumeration data types in structured data types are treated as integers at runtime. Invalid values must be caught by the server for store queries.

4.1.1.4 OPC HDA Source System

The OPC Historical Data Access source system establishes the connection to any OPC HDA server that is 1.20 compliant. PCo provides the following tabs for configuring an OPC HDA source system:

- Server Tab [page 134]
- Settings Tab [page 172]
- Aliases Tab [page 138]
- Reliable Connection Tab [page 142]

4.1.1.4.1 OPC HDA Source System: Settings Tab

Procedure

1. To configure settings for an OPC HDA source system, on the *Plant Connectivity Management Console* screen, select an OPC HDA source system and click *Settings*.
2. For information you need to enter, use the following table:

Field	Description
Force Flat Namespace	The following settings are possible: ○ *Off* This is the default setting. With this setting, the hierarchical namespace is displayed. ○ *On* If this setting has been made, all hierarchy information is removed from the namespace of the OPC DA server and stored at root level. This means that only a root folder is displayed, in which all tags are stored.
Acceptable Data Quality	With this setting, you define the OPC data quality of a tag that is to be allowed. When a tag is read out of the OPC source system, the data quality of the tag is also provided and can be taken into account accordingly. **Example** For example, you have set the data quality in PCo to *Good*. The OPC server checks the PCo settings and only sends tags with the appropriate data quality, meaning that a tag with poor data quality is not transferred to PCo in this case. The following settings are possible: ○ **Any Quality** This is the **default setting**. This setting means that the data quality of a tag is not taken into account. ○ **Good** If you choose the setting *Good*, only tags with good data quality are used. If the data quality is poorer (for example, *Uncertain* or *Bad*), PCo issues an error message. ○ **Uncertain** If you choose the *Uncertain* setting, only tags with the data quality *Good* or *Uncertain* are accepted. If the data quality is poor, PCo issues an error message. ○ **Bad** If you choose the *Bad* setting, PCo also accepts tags with poor data quality. ○ **Error** The OPC server sets the data quality of a tag to *Error* if the call has failed. The effect of the *Error* setting is that tags with this data quality are also accepted by PCo.

Field	Description
Sampling Interval (Sec)	In the notification process, this is the interval at which it is checked whether the value of a subscribed tag has changed. **Example** You have made the setting 5 seconds. In this case, the system checks every five seconds to see whether the value of a tag has changed.
Batch Processing	By choosing this indicator, you control whether you want the data source to return the selected tags all at once or individually in the query process. Batch processing is the default setting. Only deselect this indicator if your special source system does not support the batch selection of tags.

Next Steps

OPC Source Systems [page 131] and Basic Concepts of OPC [page 133]

4.1.1.5 OPC AE Source System

The OPC Alarms & Events source system establishes a connection to an OPC AE server that is 1.10 compliant. An OPC AE server can record and evaluate values from a variety of data sources and decide whether an event has occurred. PCo provides the following tabs for configuring an OPC AE source system:

- Server Tab [page 134]
- Settings Tab [page 174]
- Reliable Connection Tab [page 142]

4.1.1.5.1 Settings Tab

For OPC AE source systems, you can subscribe to alarms and events and their attributes. On this tab you specify whether you want the subscribed data to be transformed into XML or JSON format.

In the *serialization mode* field, you define how you want the attributes of a subscribed alarm to be transferred. You can choose one of the following options:

Serialization Mode

Value	Description
Standard XML	Only the values of the subscribed attributes are transferred in XML format here. This is the standard procedure that is also used in older PCo releases.
Extended XML	The attribute names and the values of the subscribed attributes are transferred in XML format here.
JSON	The attribute names and the values of the subscribed attributes are transferred in JSON format here.

4.1.2 OLE DB Source System

Definition

You can use the OLE DB source system to set up a connection to an OLE-DB-based data source and to execute database queries.

Structure

The following tabs are available for configuring an OLE DB source system:

- OLE DB Connection tab [page 176]
- Reliable Connection tab [page 181]

Integration

You can use the following OLE DB data sources:

- Microsoft Excel [page 178]
- Microsoft Access [page 179]
- OSIsoft PI
- Proficy Historian
- Aspen Tech InfoPlus.21

4.1.2.1 OLE DB Connection Tab

Prerequisites

You have created an OLE DB source system. You can use the OLE DB source system to set up a connection to an OLE-DB-based data source.

Context

OLE DB source systems can only be used in the query process [page 43].

Procedure

1. To configure the connection settings, on the *Plant Connectivity Management Console* screen, select an OLE DB source system and click the *OLE DB Connection* tab.
2. Enter the necessary data on the *OLE DB Connection* tab using the following table:

Field	Description
Variant Representation	OLE DB supports some data types from the connected data source. PCo supports the following data types: ○ Integer ○ Boolean ○ Date ○ Double ○ Float ○ String ○ Timestamp ○ Decimal If PCo does not find any of the mentioned data types in the database table to be read, PCo is to represent the data as follows: ○ *Convert Data According to Data in First Row* With this setting, PCo reads the first row of data and attempts to convert the data into one of the data types that are supported. ○ *Convert Data Into String* With this setting, PCo attempts to convert the column values into strings. The system displays a message if conversion is not possible. This is the standard setting. ○ *Flag Columns as "Unsupported"* With this setting, PCo sets the column values to `Unsupported`.
Convert Server Date and Time Values to UTC	This setting is only possible in conjunction with the setting *Convert Data According to Data in First Row*. This setting enables the conversion of date and time values to UTC so that they appear correctly in SAP MII. The SAP MII-PCo connection assumes any date or time value from the agent is in UTC; not all data sources return date and time values in UTC.
Server Time Zone	Specifies the time zone of the data server used to convert the local date and time values to UTC. This setting is only possible in conjunction with the setting *Convert Data According to Data in First Row*.

3. Choose the *Configure Connection String* pushbutton to determine and set the data provider. You can choose from the following data providers:
   - Microsoft Excel [page 178]
   - Microsoft Access [page 179]
   - OSIsoft PI

- Proficy Historian
- Aspen Tech InfoPlus.21

Next Steps

OLE DB Source System: Reliable Connection Tab [page 181]

4.1.2.1.1 Using Microsoft Excel

Prerequisites

You have created a source system of the type *OLE DB Agent*.

Context

The following procedure describes how you can use *Microsoft Excel 2003* or *Microsoft Excel 2007* as a data source.

Procedure

1. Select the OLE DB source system in the *PCo Console*.

   The system displays the *OLE DB Connection* tab.
2. Choose the *Configure Connection String* pushbutton.

   The *Data Link Properties* dialog box appears.
3. Click the *Provider* tab and choose the appropriate entry:

Data Source Used	Provider
Excel 2003 (*.xls)	*Microsoft Jet 4.0 OLE DB Provider*
Excel 2007 (*.xlsx)	*Microsoft Office 12.0 Access Database Engine OLE DB Provider*

4. Choose the *Next >>* function key or click the *Connection* tab.

   The *Connection* tab page appears.

5. On the *Connection* tab, enter the directory where the Excel file is located; for example, `C:\tmp\Test.xls`.

   If you use the browser key *(...)* , you must specify the file type **All Files (*.*)** so that the Excel file is displayed. You can ignore the user name and the password.
6. Choose the *Advanced* tab to specify the access authorizations:
   - Read
     In the *Access Permissions* area, select the *Read* entry if you only want to assign a read authorization.
   - ReadWrite
     Select *ReadWrite* if you want to grant a read and write authorization.
7. Choose the *All* tab. Double-click the *Extended Properties* entry.

   The *Edit Property Value* dialog box appears.
8. Enter the appropriate value in the *Property Value* field.

Data Source Used	Property Value
Excel 2003 (*.xls)	`Excel 8.0;HDR=Yes;IMEX=1`
Excel 2007 (*.xlsx)	`Excel 8.0;HDR=YES`

> **i** Note
>
> If your Excel sheet does not contain a header line, enter `HDR=NO`.

9. On the *All* tab, check that `False` has been set for the *Persist Security Info* entry.
10. On the *Connection* tab page, choose the *Test Connection* function key to test the connection to the Excel file.

    If the connection works, the system sends a success message.
11. Choose the *OK* function key to adopt the settings.

    The dialog box is closed.
12. Choose *Save*.

4.1.2.1.2 Using Microsoft Access

Prerequisites

You have created a source system of the type *OLE DB Agent*.

Context

The following procedure describes how you can establish the connection to a Microsoft Access database.

Procedure

1. In the PCo Console, select an OLE DB source system.

   The system displays the *OLE DB connection* tab.
2. Choose the *Configure Connection String* function key.

   The *Data Link Properties* dialog box appears.
3. Select the appropriate entry on the *Provider* tab:

Data Source Used	Provider
Access (*.mdb)	*Microsoft Jet 4.0 OLE DB Provider*
Access 2007 (*.accdb)	*Microsoft Office 12.0 Access Database Engine OLE DB Provider*

4. Choose the *Next >>* pushbutton or the *Connection* tab.

   The *Connection* tab appears.
5. Enter the directory where the database is located, such as `C:\Users\D022222\Desktop\Test.mdb`.
6. Enter the user name, such as `Admin`.
7. If you want to specify a password, you first need to select the *Allow saving password* checkbox.

   If you want to work without a password, you need to select the *Blank password* checkbox.
8. Choose the *Test Connection* pushbutton to test the connection to the database.
9. Choose the *Advanced* tab to specify the access authorizations:

   In the *Access permissions* area, select the *Read* entry so that users may read only.
10. On the *All* tab, check that `False` has been set for the *Persist Security Info* entry.
11. Choose the *OK* function key to adopt the settings.

    The dialog box is closed.
12. Choose *Save*.

4.1.2.2 Reliable Connection Tab (OLE DB / ODBC Source System)

Context

The OLE DB source system and the ODBC source system only have a passive connection check since you use the database connected as the source system for the query process [page 43] only.

PCo performs the connection check for these source systems only if a data query is received by PCo and no connection to the source system is open. In this case, PCo attempts to establish a connection. You can specify the number of connection attempts and the time between the individual attempts.

Procedure

1. On the *Plant Connectivity Management Console* screen, select a source system and click on the *Reliable Connection* tab.
2. For the information you need to enter, use the following table:

Field	Description
Maximum Number of Retries	Maximum number of connection attempts before the connection counts as failed. The default value is `3`. If the attempt to establish a connection fails after the specified number of attempts, the agent instance switches to the *Faulty* state. In this case, an icon in the form of a red square with a white cross is displayed for the agent instance. See also: Remote Client [page 602] **i Note** There are no retries if this value is `0`.
Retry Interval (Seconds)	Indicates the number of seconds before the next attempt. The default value is `30 seconds`, meaning that the PCo system checks the connection every 30 seconds. If the agent instance establishes that there is no connection, the agent makes a new connection attempt every 30 seconds.

4.1.3 ODBC Source System

Definition

You can use the ODBC source system to create a connection to any database management system that you want to use as a data source.

ODBC (Open Database Connectivity) is a standardized database interface that uses SQL as the database language. ODBC provides a programming interface that allows applications to be developed independently of the database management system being used if there is a corresponding ODBC driver for this.

Use

The ODBC source system can **only be used for the query process**. The query can be started by an SAP MII system or by an ERP system (SAP NW RFC server).

> **i** Note
>
> When using the ODBC source system, note the following:
>
> - The connection from PCo to the ODBC data source is only possible if you created and configured the desired database file in the ODBC Data Source Administrator as a Data Source Name (DSN).
>
> > **i** Note
> >
> > You can only use the *Data Source Name (DSN)*; *DSN less* is not supported.
>
> - You can only use Data Source Names of the type *User DSN* or *System DSN*. *File DSN* is not supported.

Structure

The following tabs are available for configuring an ODBC source system:

- ODBC Database Connection tab [page 183]
- Reliable Connection tab [page 181]

More Information

For more information about ODBC, see:

Wikipedia (engl.): http://en.wikipedia.org/wiki/ODBC

Microsoft ODBC Overview (MSDN): http://msdn.microsoft.com/en-us/library/windows/desktop/ms710252(v=VS.85).aspx

4.1.3.1 ODBC Source System: ODBC Database Connection Tab

Use

You create the connection to an ODBC source system [page 182] on this tab.

Prerequisites

You have created the database file that you want to use as the data source as a Data Source Name (DSN) in the ODBC Data Source Adminstrator.

See also: Creating a DSN in the ODBC Data Source Administrator [page 185]

Procedure

1. In the menu, choose *Plant Connectivity* > *New* > *Source System*. Choose the *ODBC source system* entry in the dialog box.
   PCo creates the ODBC source system.
2. On the *ODBC Database Connection* tab, enter the following data:

Field	Description
Data Source Name (DSN)	You can choose the database file that you created previously as a DSN as the data source here. Each entry is identified either as a *User DSN* or as a *System DSN*.
User Name and Password	If the data source requires authentication, you can enter the logon data here.

Field	Description
Variant Representation	The ODBC agent supports some data types from the connected data source. PCo supports the following data types: • Integer • Boolean • Date • Double • Float • String • Time stamp • Decimal If PCo does not find any of the mentioned data types in the database table to be read, PCo is to represent the data as follows: • *Convert Data According to Data in First Row* With this setting, PCo reads the first row of data and attempts to convert the data into one of the data types that are supported. • *Convert Data Into String* With this setting, PCo attempts to convert the column values into strings. The system displays a message if conversion is not possible. This is the default setting. • *Flag Columns as "Unsupported"* With this setting, PCo sets the column values to `Unsupported`.
Convert Server Date and Time Values to UTC	This setting is only possible in conjunction with the setting **First**. This enables the conversion of date and time values to UTC so that they are displayed correctly in SAP MII. The SAP MII-PCo connection assumes any date or time value from the agent is in UTC; not all data sources return date and time values in UTC.
Time Zone of Server	This setting is only possible in conjunction with the setting **Convert Data According to Data in First Row**. Here you enter the time zone of the data server used to convert the local date and time values to UTC.

Result

The connection to the ODBC source system has now been established. You can now create the agent instance for this.

> **i Note**
>
> If you want to execute the agent instance as a service, you must specify a user name and a password. The user name must correspond to the *User DSN*. Otherwise, the service cannot be started.
>
> In the agent instance, on the *Host* tab, you must therefore make the following settings:

Field	Setting
Run Host as an Executable	Do not set the indicator here. The agent instance is then executed as a Windows service.
User Name	You enter your Windows user here. You must enter the user name that was used to create the *User DSN*.
Password	You enter your Windows password here.

4.1.3.1.1 Creating a DSN in the ODBC Data Source Administrator

Context

If you are using *Microsoft Windows*, you must first create and configure the ODBC connection to your data source by means of the ODBC Data Source Administrator before you can create the *ODBC source system* in the *PCo Management Console*. To do so, you must create a Data Source Name (DSN) for the data source in the *ODBC Data Source Administrator*.

Procedure

1. You can start the *ODBC Data Source Administrator* from the start menu under *Run* *odbcad32.exe*.

   > **i Note**
   >
   > If you are using a 64-bit version of *Windows*, you must ensure that you are using the *ODBC Data Source Administrator* in the **32-bit version**. A 32-bit version and a 64-bit version of the *ODBC Data Source Administrator* are available in the 64-bit version of *Windows7*. In this Windows version, you start the 64-bit version of the *ODBC Data Source Administrator* from the start menu with Run. The *System Data Source Name* entries that are created using the 64-bit version are not taken into account by PCo, however.
   >
   > The valid 32-bit version is located in the subfolder SysWOW64 in the Windows directory; for example, under `C:\Windows\SysWOW64\odbcad32.exe`.

> If you create *User Data Source Name* entries instead of *System Data Source Name* entries, you can also use the 64-bit version of the *ODBC Data Source Administrator*.

After the `odbcad32.exe` is started, the *ODBC Data Source Administrator* dialog box is displayed.

2. You must add a Data Source Name for the database that you want to use as the data source. The following types are available:
   - **User DSN**
     On the *User DSN* tab, you can add a user-specific *Data Source Name (DSN)* for the data source of your choice. If you are using a *User DSN*, a connection is only created for the Windows user who is logged on and only that user can see and use this connection.
   - **System DSN**
     You can add an entry for the data source of your choice on the *System DSN* tab. All users who log on to the system can then see and use the *System DSN*.

> **i Note**
>
> You can create a *User DSN* and a *System DSN* with the same name. In this case, only the *User DSN* is displayed in the *PCo Management Console*; the *System DSN* with the same name is not displayed. For that reason, you must not use the same name for a *User DSN* and a *System DSN*. Duplicated names are not possible on the *User DSN* or *System DSN* tabs. The *ODBC Data Source Administrator* issues a message if an attempt is made to use duplicated names.

3. You have to select an ODBC driver for the data source. The ODBC drivers are provided by each database provider. For that reason, the dialog for configuring the ODBC connection can be built up differently.

Example

Example for creating a *System DSN* for an Excel file that you want PCo to use as the data source:

1. In the *ODBC Data Source Administrator*, click the *System DSN tab*.
2. Choose the *Add* pushbutton.
   The *Create New Data Source* dialog box appears.
3. Choose the driver; for example, `Microsoft Excel Driver (*.xls)`.
   The ODBC Microsoft Excel Set Up dialog box appears.
4. Enter the following data:
   - *Data Source Name*
     Here you can enter a name of your choice for the file that you want to use as a data source. This name is displayed later in PCo on the *ODBC Database Connection* tab with the addition `(System DSN)`.
   - *Description*
     You can enter a short description here.
   - *Version*
     Choose the Excel version of your file.
5. Choose the *Select Workbook* pushbutton to select the directory and the Excel file you want.
6. Confirm by choosing OK.
   The *System DSN* is now displayed in the *ODBC Data Source Administrator* and in the *PCo Management Console* on the *ODBC Database Connection* tab.

4.1.3.2 Reliable Connection Tab (OLE DB / ODBC Source System)

Context

The OLE DB source system and the ODBC source system only have a passive connection check since you use the database connected as the source system for the query process [page 43] only.

PCo performs the connection check for these source systems only if a data query is received by PCo and no connection to the source system is open. In this case, PCo attempts to establish a connection. You can specify the number of connection attempts and the time between the individual attempts.

Procedure

1. On the *Plant Connectivity Management Console* screen, select a source system and click on the *Reliable Connection* tab.
2. For the information you need to enter, use the following table:

Field	Description
Maximum Number of Retries	Maximum number of connection attempts before the connection counts as failed. The default value is `3`. If the attempt to establish a connection fails after the specified number of attempts, the agent instance switches to the *Faulty* state. In this case, an icon in the form of a red square with a white cross is displayed for the agent instance. See also: Remote Client [page 602] **i Note** There are no retries if this value is `0`.
Retry Interval (Seconds)	Indicates the number of seconds before the next attempt. The default value is `30 seconds`, meaning that the PCo system checks the connection every 30 seconds. If the agent instance establishes that there is no connection, the agent makes a new connection attempt every 30 seconds.

4.1.4 Modbus Source System

Definition

You can use the Modbus source system to set up a connection between PCo and programmable logic controllers (PLC) or other devices. This allows you to connect one or multiple devices to PCo as a data source to query or monitor specific tags. A Modbus source system can be used for query and notification processes.

The communication of the Modbus source system is based on the Modbus protocol that is used as standard for the communication with programmable logic controllers. Data communication with the source system is in the form of message frames.

Use

You can use the Modbus source system to forward data directly from the device to PCo. This data is sent to PCo in the form of data streams. The data stream is divided up into individual message frames. A message frame corresponds to one message. In the case of serial communication, the individual message frames are separated from each other with defined silent intervals. (See also: General Form of Message Frames [page 204].)

Data Transfer

The Modbus agent supports the following types of data transfer:

- TCP
- Serial communication

Functions of the Modbus Agent

The Modbus agent supports the following tag features in the PCo system:

- Retrieve
- Store
- Mass Store
- Subscribe
- Unsubscribe
- Native Mask
- Groups
- Secondaries
- Metadata
- Aliases

Tabs

The following tabs are available for configuring a Modbus source system:

- Modbus Agent tab [page 189]
- Modbus Units Tab [page 196]
- Tag Definition tab [page 198]
- Source System: Aliases tab [page 138]
- Source System: Reliable Connection tab [page 142]

See Also

http://www.modbus.org

4.1.4.1 Modbus Agent Tab

To create a Modbus source system, do the following:

1. Create a source system of the type *Modbus source system*.
2. Make the following settings on the *Modbus Agent* tab:

Field	Description
Modbus Client	The Modbus type *Modbus client* defines that PCo is the leading system. PCo can send queries to the connected devices. In this case, the devices act as servers.
Modbus Server	The Modbus type *Modbus server* defines that PCo is the subordinate system. PCo can receive queries from the connected devices. In this case, the devices act as clients.

Field	Description
Acceptable Quality	With this setting, you define the permitted data quality of the tags. When a tag is read out of the Modbus source system, the data quality of the tag is also provided and can be taken into account accordingly. The following data quality settings are possible: ○ **Any Quality** This setting means that the data quality of a tag is not taken into account and all tag values are accepted. ○ **Good** This is the default setting. If you choose the *Good* setting, only tags with good data quality are used. If the data quality is poorer (*Uncertain*, *Bad*, *Error*), PCo issues an error message. ○ **Uncertain** If you choose the *Uncertain* setting, only tags with the data quality *Good* or *Uncertain* are accepted. The status *Uncertain* is set if the Modbus server does not respond. ○ **Bad** If you choose the *Bad* setting, PCo also accepts tags with poor data quality. The Modbus server sets the data quality of a tag to *Bad* if a tag cannot be read. ○ **Error** The Modbus server sets the data quality of a tag to *Error* if the call has failed. The effect of the *Error* setting is that tags with incorrect data quality are also accepted by PCo.
Channel	You choose the communication channel here: ○ TCP ○ Serial communication
Response Timeout	Indicates the period in which the connected Modbus device must send a response. If the specified time is exceeded, an error message is issued. The default setting is 200 milliseconds.

Field	Description
Minimum Update Rate	Specifies the minimum time for the update rate in milliseconds (ms). ○ Specifies how often the data is to be read by the connected device ○ Specifies the minimum time that has to be passed before the Modbus source system sends the next value change to PCo. If you enter 100 ms, at least 100 ms need to have passed between the sending of two messages. **i Note** The value that you enter here is used together with the update rates of the individual tags (see: Tag Definition Tab [page 198]) to define the actual update rate for sending values from the Modbus source to PCo. (See: Determining the Update Rate [page 194].)

3. Make the settings for TCP or for serial communication:
   - Settings for TCP Communication [page 191]
   - Settings for Serial Communication [page 192]

4.1.4.1.1 Settings for TCP Communication

If you have chosen the setting `TCP` in the *Channel* field, the *TCP Communication* tab is ready for input:

Field	Description
Host	Specifies the host name of the Modbus server. This might be, for example, one SPS. **i Note** This entry is only required if you have set **`Modbus Client`** as the *Modbus type*.
Port	Specifies the TCP port. The standard port for the Modbus protocol is `502`.

Field	Description
Messages in Process	Specifies the maximum number of pending messages. These are messages that have not yet been processed by the connected server device. You can enter the maximum permitted number of messages here. This always makes sense if communication can also be asynchronous when TCP is used. In asynchronous communication, the Modbus agent does not wait for the response to the last message, but rather already sends the next message. If you enter a higher number of messages here, you can improve performance.
Modbus Client IPs	Specifies the list of permitted client systems. You can add as many IP addresses of devices here as you want. This is only possible if you have set PCo as the *Modbus server*. Since the Modbus protocol does not offer any direct security measures, the list of permitted clients is the only way of guaranteeing security.

4.1.4.1.2 Settings for Serial Communication

On the *Serial Communication* tab, you define the settings for serial communication:

Field	Description
Serial Port	Indicates the name of the serial port on the computer on which PCo is running.
Baud Rate	Indicates the amount of transferred data in bits per second. You can select the most common baud rates from the drop-down box. You can also enter a baud rate of your choice. **⚠ Caution** The baud rate depends on the device that you want to connect to PCo. Not all baud rates are compatible with all serial ports.

Field	Description
Parity Check	Here you enter the method that you want to use for detecting errors in data transfer. (See also: Parity Check.) If you use the parity check, an additional parity bit is added to the data bits to be transferred. The following options are available: • No Parity Check Do not perform a parity check. • Odd In this setting, the number of 1 bits in each character string that is transferred, including the parity bit, must always be **odd** so that the recipient accepts the transferred information as error-free. • Even In this setting, the number of 1 bits in each character string that is transferred, including the parity bit, must always be **even** so that the recipient accepts the transferred information as error-free. **Example** Transferred bits: 0110100**1** **1** is the check bit Bit sum = 4 x 1 = 4 The bit sum is even. Therefore, if you have set the checking method *Even*, there are no errors. **Caution** The type of parity check depends on the device that you want to connect to PCo.
Handshake	Here you specify the serial port signals that are used for the handshake. The options are as follows: • No Handshake • Request to Send **Caution** The setting depends on the device that you want to connect to PCo.

Field	Description
Read Timeout	The read timeout indicates the maximum interval permitted within a message frame in milliseconds. This maximum interval between two sent characters of a message frame must not last longer than is required for sending 1.5 characters. If you enter **0**, the value permitted by PCo is calculated based on the baud rate. If the permitted value is exceeded when a message frame is sent, the message frame is faulty. (See also: General Form of Message Frames [page 204].)
Silent Interval	Interval between the end of a message frame and the start of the next message frame. The message frames are separated in the data stream by the silent interval. Here you can enter the permitted silent interval in milliseconds. If you enter **0**, the value permitted by PCo is calculated based on the baud rate. (See also: General Form of Message Frames [page 204].)
Use ASCII Protocol	With this indicator, you specify that you want the data to be sent in the ASCII mode. You should only use this setting for test purposes. If you do not set the indicator, the RTU mode is used.

4.1.4.1.3 Determining the Update Rate

Use

Since the various tags, which together form the subscription items in a notification, can have different update rates, PCo determines a common update rate for this group of tags.

Prerequisites

- You have specified a *minimum update rate* on the *Modbus Agent* tab (see: Modbus Agent Tab [page 189]).
- You have specified an update rate for each tag on the *Tag Definition* tab (see: Tag Definition Tab [page 198]).

Features

The PCo system determines the update rate to be used using the *minimum update rate* and the *update rates* that you have defined for the individual tags on the *Tag Definition* tab. This is always the case if you have entered multiple tags on the *Tag Definition* tab. The smallest value that you have entered for the tags is always used. This value cannot be smaller than the minimum update rate.

Example

Example 1

Update Rate	Input Value
tag01	1000 ms
tag02	500 ms
Minimum Update Rate	20 ms

Result: In this case, the update rate of tag02 **500 ms** is used for all tags.

Example 2

Update Rate	Input Value
tag01	1000 ms
tag02	500 ms
Minimum Update Rate	2000 ms

Result: In this case, the minimum update rate of **2000 ms** is used for all tags because the update rate used must not be smaller than the minimum update rate.

Example 3

Update Rate	Input Value
tag01	1000 ms
tag02	500 ms
Minimum Update Rate	0 ms

Result: The update rate of tag02 **500 ms** is used for all tags.

4.1.4.2 Modbus Units Tab

Use

On this tab, you enter the devices that you want to connect to the PCo system as Modbus servers. In this case, PCo is set with the Modbus type *Modbus client* and therefore acts as leading system.

> **i** Note
>
> If you use PCo as the *Modbus server*, that is, as the subordinate system, there is only one unit on this tab and that is PCo itself.

Procedure

1. Make the settings for the first Modbus server unit:

Field	Description
Unit ID	ID of the programmable logic controller (PLC) or the device that is to be connected to PCo as a Modbus server.
Output Coils	Indicates the number of single bit outputs of the Modbus server that is to be connected. A maximum of 65536 coils can be defined. Each coil is assigned to a single bit. This data type can be changed by an application program.
Input Contacts	Indicates the number of single bit input contacts of the Modbus server that is to be connected. These input contacts can only be used in read-only mode. A maximum of 65536 contacts can be defined. Each contact is assigned to a single bit. This data type is usually provided by an input/output system.
Input Registers	Indicates the read-only area of 16-bit registers. A maximum of 65536 input registers can be defined. This data type can only be provided by an input/output system.
Holding Registers	The holding registers are the most frequently used 16-bit registers. They can be used in read and write mode. They enable, for example, the reading of measured values, counter readings, or the reading of device configuration. You can define a maximum of 65536 registers. This data type can be changed by an application program.

Field	Description
Files	Specifies the number of additional register areas. You can define up to 65535 files.
File Size	Specifies the number of 16-bit registers in each file. Each file can contain up to 10000 registers.

2. Enter the string format. You use the fields for the string format to define how the strings are displayed:

Field	Description
Little Endian Order	If you select this checkbox, the little endian order is used for the byte order. If you do not select the checkbox, the big endian order is used.
Encoding	You select the encoding format you want here: ○ ASCII ○ UTF-8 ○ UTF-16 (See also: Encoding (BC-I18).)

3. Define the *swapping mode* for the various *data types*.
   The Modbus protocol usually uses the big endian order for address and data values (if you have not selected the *Little Endian Order* checkbox). This means that whenever a numerical quantity that is larger than one byte is to be transferred, the byte with the largest value needs to be sent first. Therefore, the byte order always needs to be clearly defined in such a case.
   In the swapping mode table, each *Word* has two bytes and each *DWord* (Double Word) has 2 words. If you select a checkbox, for example, the Words checkbox, this means that the relevant data structure needs to be swapped.

Example

The following example shows how the swapping mode can influence the **32-bit values** if you have chosen the swapping modes *Bytes* or *Words* or both:

Example of Swapping Mode

	DWord			
	Word		Word	
	Byte	Byte	Byte	Byte
Input	0A	0B	0C	0D
Swap Bytes	0B	0A	0D	0C
Swap Words	0C	0D	0A	0B
Swap Bytes + Words	0D	0C	0B	0A

In the *Input* row, the bytes are displayed in the original order (without swapping mode). In the *Swap Bytes* row, the bytes have been swapped. In the *Swap Words* row, the words (byte pairs) have been swapped. In the *Swap Bytes and Words* option, both are swapped.

4. To enter an additional device, choose the *Create New Modbus Unit* pushbutton.

4.1.4.3 Tag Definition Tab

Use

On this tab, you enter the tags for the devices that you want to monitor. The tags are displayed in a structure. You can also create a namespace hierarchy for the tags.

The tab is divided into two areas:

The **structure or namespace hierarchy** of the selected tags is displayed in the left screen area. This area is empty at first. The structure is only displayed once tags have been entered.

The **input fields for entering tags** are offered in the right screen area.

Procedure

1. To enter the first tag, you enter details for the tag as follows:

Field	Description
Access Path	You enter the access path for the tag here. The symbol `.` is used as a separator within the path. The depth of the path can be freely defined.
Description	You can enter a description of your choice for the tag here.
Update Rate	You define the update rate for the selected tag in milliseconds here. The update rate specifies the minimum time that needs to have passed between two consecutive query operations (reading and sending the value). Since the tag is queried using the polling method (see: Polling), it is important to reduce the amount of communication between the Modbus devices. Therefore, it makes sense to define a larger update rate. **i Note** The value that you enter here is used together with the minimum update rate that you specify on the Modbus Agent [page 189] tab to define the actual update rate for sending values from the Modbus source to PCo. (See also: Determining the Update Rate [page 194].)
Deadband	Limit value for filtering out measurement values. Here you can enter a percentage to define the bandwidth in which the value of a tag must at least change before the Modbus device reports the value to PCo as changed. **i Note** The defined percentage can only be used for numerical tags. If you enter 0%, each value change of a tag triggers a notification message.

Field	Description
Type	The following types of tags are supported: ○ boolean ○ boolean [ ] ○ byte ○ byte [ ] ○ double ○ double [ ] ○ int16 ○ int16 [ ] ○ int32 ○ int32 [ ] ○ int64 ○ int64 [ ] ○ single ○ single [ ] ○ string **i Note** Tags of the type *boolean* cannot be defined in files. Tags of the type *byte* can only be defined in output coils or input contacts.
Unit ID	ID of the device or the PLC for which you want to define the tag. Here you enter the ID of the device that you defined previously on the *Modbus Units* tab.

Field	Description
Address	Here you enter the address of the selected tag. The following address overview indicates which addresses you need to specify for a specific tag (the *file number* is 0 in these cases): ○ Output coils 000000 ... 065535 (or H00000...H0FFFF) ○ Input contacts 100000 ... 165535 (or H10000 ... H1FFFF) ○ Input registers 300000 ... 365535 (or H30000 ... H3FFFF) ○ Holding registers 400000 ... 465535 (or H40000 ... H4FFFF) To specify the address in a file, use the interval 000000 ... 065535 (or H0000 ... H0FFFF). To define a tag of the type Boolean in an input address bit or holding address bit, you need to specify a bit offset. **Example** The following addresses are examples of Boolean tags: 400001.0, 401111.15, 300001.7, and so on. The bit offset must be between 0 and 15 (only in conjunction with input and holding register addresses).
File Number	Specifies the *file number* of the tag. You use 0 as the file number if the selected tag has been defined in output coils, input contacts, input registers, or holding registers.
Size	Specifies the size of the tag in the address space. If it is a bit address (output coil or input contact), you enter the size of the tag here in bits. If the address is an input register, holding register, or a file, the number of the 16-bit register is specified here. The system sets the size automatically if the selected tag is not an *array* or a *string*.

2. If you want to enter another tag, choose the *Add New Modbus Tag* pushbutton.
3. If you want to change an existing tag, select the tag in the structure and choose the *Edit Modbus Tag* pushbutton.
4. If you want to delete a tag, select the tag in the structure and choose the *Delete Modbus Tag* pushbutton.

5. You can use the *Export* pushbutton to export the configuration of the tags that you have entered here on the *Tag Definition* tab and store it as a CSV file, so that this tag configuration can be used for another Modbus source system. (See: Importing or Exporting Modbus Tags [page 202].)
6. You can use the *Import* pushbutton to import an existing tag configuration. (See: Importing or Exporting Modbus Tags [page 202].)

4.1.4.3.1 Importing or Exporting Modbus Tags

Use

You can use the *Import* and *Export* functions to import or export the configuration of selected tags using CSV files.

> **i** Note
>
> The CSV file (file with comma-separated values) contains the tags that you entered on the *Tag Definition* tab. The individual tags are separated from each other by a separator, for example, a comma.

Procedure

Exporting a CSV File

1. Choose the *Export* pushbutton.
   The *Import Modbus Tags* dialog box appears. PCo proposes a path for storing the file, for example, *C:\Program Files (x86)\SAP\Plant Connectivity\System*.
2. Enter the file name and choose *Save*.
   PCo exports the tags and stores them in a CSV file.

Importing a CSV File

1. Choose the *Browse* pushbutton to select the CSV file you want.
2. Select the CSV file you want.
   The file is analyzed and the relevant separator (for example, comma) is selected automatically. Upon import, PCo tries to select the relevant columns automatically. You can change the columns, however. The access path and the address need to be unique but the update rate, the deadband, type, unit ID, size, and file number can initially be filled with some default values if there is no relevant column (in this case, the entry in the *Column* field is empty).
3. Choose the import mode you want in the *Mode* field. There are two modes:
   - Overwrite
     The import mode *Overwrite* replaces all existing tags (those entered manually as well as imported tags) for which there are address conflicts or conflicts with the access path. Only a warning is issued.
   - Append
     In the import mode *Append*, the imported tags are added to the existing tags. Address or access path conflicts are interpreted as errors and the import of faulty tags is terminated.

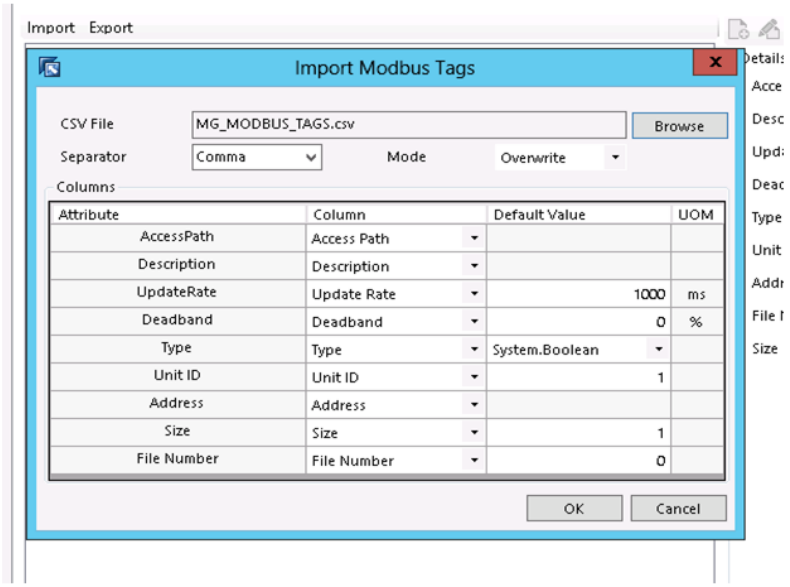
Check the attributes (access path, description, and so on) in the *Columns* screen area. You can select an entry, as required, for each attribute under *Column*. The access path and the address must be unique. The update rate, deadband, type, unit ID, size, and file number can be filled initially with some *default values*.

4. Confirm your entries by choosing the pushbutton *OK*.

Example

The screenshot shows what the *Import Modbus Tags* dialog box looks like after a CSV file has been selected:

Importing Modbus Tags



4.1.4.4 General Form of Message Frames

Message Frames in Serial Communication

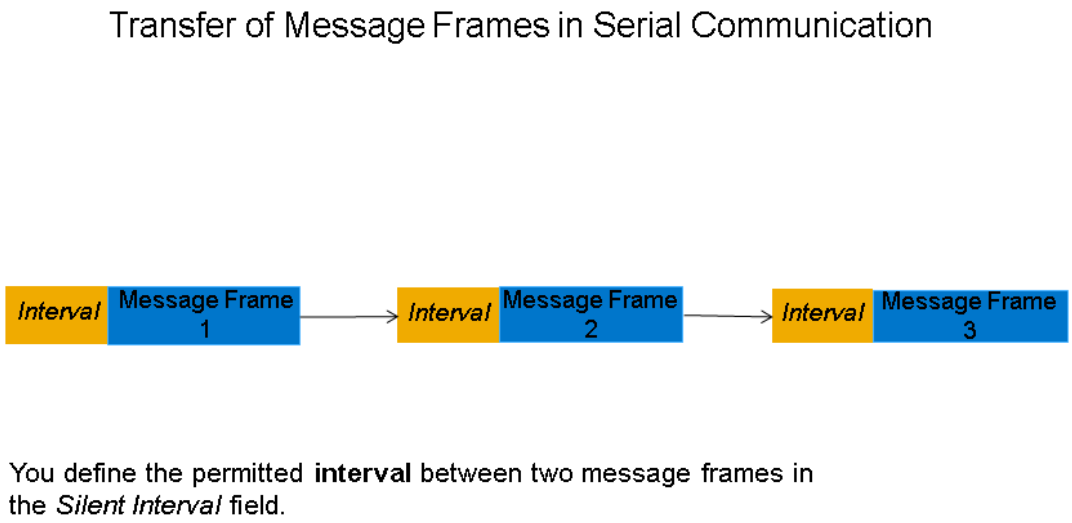
A message frame is built up as follows in serial communication:

Address	Function	Data	CRC Check
8 bits	8 bits	8 bits	16 bits

- **Address**
  Indicates the address of the receiver
- **Function**
  Indicates the purpose of the data transfer. The functions are standard functions in the Modbus protocol.

  > **Example**
  >
  > Examples of standard functions:
  > - READ INPUT STATUS: Reading of digital input statuses
  > - READ HOLDING REGISTERS: Reading of measured values, counter readings, or reading of device configuration

- **Data**
  Contains the actual information that is to be transferred. The data is always transferred as a multiple of 16-bit registers.
  The *Data* field is subdivided into the following:
  - Register
  - Number of registers to be transferred
  - Division of the information into read information or information to be stored
- **CRC Check**
  Specifies the check sum. The sender calculates the check sum from all bytes. The receiver calculates the value again and compares both results. The check is used to detect any transfer errors.
- **Sending Message Frames**
  There needs to be a silent interval of at least 3.5 characters between two message frames to separate the message frames from each other. In other words, the time between two sent message frames should be at least as long as is required for sending 3.5 characters. You define this interval in the *Silent Interval* field in the *Serial Communication* area. (See also: Settings for Serial Communication [page 192].)



The data stream should not be interrupted within a message frame. The individual characters must not be more than 1.5 characters apart, that is, the maximum time distance between two sent characters of a message frame must not last longer than is required for sending 1.5 characters. If there is a longer interval within a message frame, the message frame is regarded as incomplete and an error message is issued. You can define this maximum permitted interval in the *Read Timeout* field in the *Serial Communication* area.

Message Frames in TCP Data Transfer

A message frame is built up as follows when TCP is used:

MBAP Header	Function code	Data
7 bytes	1 byte	n bytes

- **MBAP Header**
  The MBAP Header (Modbus Application Protocol Header) contains the header data of the message frame, such as transaction ID, protocol ID, number of subsequent data bytes, unit ID.
- **Function code**
  Specifies the function code of the standard Modbus protocol, that is, the receiver is informed which action is to be performed.
- **Data**
  Contains the information that is to be transferred.

4.1.5 MQTT Source System

MQTT is a message protocol for machine-to-machine communication (M2M) and IoT. You can use the MQTT source system to set up a connection to an MQTT broker (MQTT server). The MQTT broker provides PCo with data in the form of MQTT messages. As a rule, in the context of MQTT, this is sensor data.

The MQTT source system takes on the role of **subscriber**. In order to be able to receive any type of message, the PCo client needs to connect with the MQTT broker and let it know for which topics it would like to receive a message. A topic is a string that has a URL-like structure and represents a type of subject of the message. This operation is called a subscription.

> ✤ Example
>
> The MQTT source system in PCo (subscriber) is to monitor the temperature data of a production plant, for example.

Communication between PCo and the MQTT server is possible using the following connections:

- TCP/IP
- TLS
- WebSocket
- Secure WebSocket connection

The following tabs are available for configuring the MQTT source system:

- Client Tab [page 206]
- Connection Tab [page 209]
- Security Settings Tab [page 215]
- Tag Definition Tab [page 219]
- Aliases Tab [page 138]
- Reliable Connection Tab [page 142]

4.1.5.1 Client Tab

This document describes the settings for the connection to the MQTT server.

To create an MQTT source system, do the following:

1. Create a source system of the type *MQTT source system*.

2. Make the following settings on the *Client* tab:

Settings for the MQTT Client

Field	Description
Client ID	ID that PCo uses to identify itself to the MQTT server. The client ID must be unique. Enter an ID with maximum 23 places. Numbers, lowercase letters, and uppercase letters are allowed. Alternatively, you can leave the *Client ID* field blank. When the connection to an MQTT server is established, a unique client ID is then generated automatically. This is important because there must always be a 1:1 relationship between the source system and the agent instance.

Server URI	You enter the URI of the MQTT server here. The URI schema can be built up as follows:
	
	○ TCP connection (mqtt)
	
	Example
	`mqtt://mo-92d012345.mo.sap.corp:1883`
	
	○ Secure Transport Layer Security (TLS) connection (mqtts)
	
	Example
	`mqtts://mo-92d012345.mo.sap.corp:8883`
	
	You can maintain security settings for the mqtts connection. (See also: Security Settings Tab [page 215].)
	
	○ WebSocket (ws)
	
	Example
	`ws://mo-92d012345.mo.sap.corp:8000/mqtt`
	
	○ Secure WebSocket connection (wss)
	
	Example
	`wss://mo-92d012345.mo.sap.corp:8001/mqtt`
	
	You can maintain security settings for the wss connection. (See also: Security Settings Tab [page 215].)

User Name	User name for the logon to the MQTT server

Field	Description
Password	Password for the logon to the MQTT server **i Note** The entry of user name and password is optional or dependent on the MQTT server.
Response Timeout	Indicates the period in which the MQTT server must send a response.
Keep-Alive Interval	Indicates the maximum time that is allowed to pass between two consecutive control packets that are sent by the PCo client. If there are no control packets that need to be sent, PCo sends a PINGREQ packet. If you set **0** here, no PINGREQ packets are sent.
Clean Session if Connection Was Interrupted	If you select the checkbox, any messages that have been missed during an interruption to the connection are not sent when the connection is restored. If you do not select the checkbox, and the connection is interrupted, the MQTT server sends all missed messages as soon as the connection has been restored.

3. Choose *Open the Client Configuration Wizard* if you want to make all client and connection settings in one step.
4. Choose *Test Connection* to test the connection to the MQTT server.
   PCo sends the CONNECT and DISCONNECT control packets to the MQTT server.

4.1.5.2 Connection Tab

Here you define the security-related connection data for the connection to the MQTT server.

Last Will Settings

Settings for the Last Message if Connection Interrupted

Field	Description
Topic Name	This is the name of the last message that is to be sent from the MQTT server to PCo if the connection between PCo and the MQTT server is interrupted. This message is called a last will message. If you enter something here, you can enter a message text in the *Message* field that you want to be sent in the following situations: • The MQTT server discovers an I/O error or a network outage. • The PCo client does not manage to communicate with the server within the defined keep--alive interval. • The PCo client closes the network connection without first sending a DISCONNECT packet. • The MQTT server closes the network connection due to a protocol error.
Message	Here you enter the message text that you want to be sent from the MQTT server to PCo if the connection is interrupted.

Field	Description
QoS	Quality of service: To make sure that a a sent message reaches the recipient, MQTT defines three different quality of service (QoS) levels with which a message can be sent: • *0 - At Most Once* When PCo receives the last message, no confirmation is to be sent to the MQTT server. • *1 - At Least Once* PCo sends the *Publish Acknowledge Packet* (PUBACK packet) to the MQTT server as soon as the last will message has arrived. • *2 - Exactly Once* PCo sends the *Publish Received Packet* (PUBREC) to the MQTT server as soon as the last will message has arrived. The server responds with the *Publish Release Packet* (PUBREL). PCo responds with the *Publish Complete Packet* (PUBCOMP).
Retain Message	By selecting this checkbox you define that the MQTT server has to store the last will message and send it to PCo, even if PCo is currently offline.

Client Certificate

Settings for the Client Certificate

Field	Description
Certificate	Here you can select a client certificate with a private key if you want to set up a secure connection (mqtts or wss). **i Note** The client certificate is always stored in the Windows certificate store.

Certificate Folders

Settings for the Certificate Folders

Field	Description
Store Type	Here you select the store for the server certificate of the MQTT server that you want to be validated. The following types are supported: • *Microsoft certificate store* When a connection is being established, with this setting, PCo automatically searches in the Microsoft certificate store folder for a server certificate. → Recommendation If you are using PCo based on Windows OS, you need to use the *Microsoft certificate store* option. • *File system certificate store* With this option, you can specify the store location for the certificates, which PCo is to trust, in the file system.
Trusted Certificates	Here you can specify the folder in which the trusted certificates are stored. If you have selected the *Microsoft certificate store*, this is the folder for the *trusted root certification authorities*. The system proposes this automatically. Example `Local Computer/Trusted Publishers` If you have chosen the *file system certificate store*, a directory is proposed in the file system with the following subfolders: • *certs*: Folder for trusted certificates If you store a certificate from the certificate chain in this folder, this certificate is trusted in the check. The certificate is regarded as not trusted if there is no certificate from the certificate chain stored here. • *crls*: Folder for certificate revocation lists • *private*: Folder for private certificates that are not used for Windows If you choose *Browse*, a dialog box appears where you can select another folder.

Field	Description
Issuer Certificates	Here you can specify the folder in which the certificates of a trusted issuer are stored. If you selected the *Microsoft certificate store*, this is the folder for the intermediate certificate authorities. This is proposed automatically. **Example** `Local Computer/Intermediate Certificate Authorities` If you have selected the *file system certificate store*, a directory is proposed in the file system with the subfolder *certs*. This folder is used to complete the certificate chain if the server does not send the complete certificate chain.
Rejected Certificates	Here you can specify the folder in which the rejected certificates are stored. If you are using the *Microsoft certificate store*, select *Untrusted Certificates* here. **Example** `Local Computer/Untrusted Certificates` If you have selected the *file system certificate store*, use a directory in the file system with the subfolder *certs* (folder for rejected certificates).

Certificate Validation Options

Validation Options

Field	Description
Revocation Check	In this field you define how the revocation check of the server certificate is to be performed. You have the following options: • *No Check on Revoked Certificates* No check is carried out. • *Check Online Revocation Lists* The online check is a secure procedure but it can have a negative impact on performance. → Recommendation This is the recommended setting in connection with the Microsoft certificate store. • *Check Offline Revocation Lists* ○ If you are using the **Microsoft certificate store**, you need to copy all the relevant certificate revocation lists into the **Trusted Root Certification Authorities** directory. ○ If you have selected the **file system certificate store**, you need to copy all related certificate revocation lists as `.crl` files into the revocation list folder.
Revocation Check Scope	Indicates the scope of the revocation check. You have the following options: • *Check End Certificate Only* Only the last certificate in a certificate chain is checked. • *Exclude Root Certificate from Check* • *Check Entire Chain* All certificates in a certificate chain are checked. → Recommendation This is the recommended setting.

Field	Description
Ignore Server Host Name	If you select this checkbox, the check results of the server host name are not taken into consideration. During the check, a comparison is made with the domain name system (DNS) name that is included in the certificate. The DNS name is the name of a server in a domain, for example: mo-90dxxxxxx.mo.sap.corp. → Recommendation SAP recommends that you do not set this indicator.
Ignore Validity Period	If you select this checkbox, you define that the validity period of the server certificate and the certificates in the certificate chain are not to be taken into consideration.

Proxy Settings

Field	Description
Proxy URI	You use this setting if you want the MQTT server to be connected using a proxy. PCo supports two types of proxy URIs: • http: • socks5: Example http://proxy:8083 socks5://proxy:8080
User Name	User name for the proxy (optional)
Password	Password for the proxy (optional)

Related Information

Recommended Security Settings [page 227]
Process for Validation of the Server Certificate Chain [page 228]
Asymmetric Hash Algorithms [page 228]

4.1.5.3 Security Settings Tab

On this tab, you make the settings for secure connections and user authentication for MQTT source systems.

To be able to identify the PCo system as a client to the MQTT server and vice versa, **X.509 v3 certificates** are used, provided a secure connection is to be set up. In the context of MQTT, the certificates used here are called application certificates.

You can maintain security settings, if you have defined a prefix for a secure connection (wss or mqtts) in the *Server URI* field on the *Client* tab.

- **mqtts://**
  For the secure Transport Layer Security (TLS) connection (mqtts), you can select an application certificate. PCo can use this certificate to identify itself to the MQTT server. Whether or not you have to specify a certificate depends on the MQTT server.
  You also define the settings for the certificate folders and the validation options. (See also: Certificate Folders [page 216] and Certificate Validation Options [page 218].)
- **wss ://**
  For the secure WebSocket connection, you can choose from three authentication methods. (See: Authentication Method (wss) [page 215].)
  If you have set authentication using a certificate, you must also make the settings for the certificate folders and the validation options. (See also: Certificate Folders [page 216] and Certificate Validation Options [page 218].)

Related Information

Recommended Security Settings [page 227]

4.1.5.3.1 Authentication Method (wss)

You can choose between three authentication methods for the WebSocket connection:

- No Authentication
  No logon data is required for this setting.
- Basic Authentication
  If you choose basic authentication, you need to specify, in the *Basic* screen area, a *user name* and *password* that are known to the MQTT server. They are used to build up the WebSocket connection.
  You can also select the *Send Credentials with Each Request* checkbox if you want the user name and password to be sent for each request.
- Client Certificate Authentication
  In this case, you have to select **a certificate with a private key**. The selected certificate is also used as an *application certificate*. In addition, the two screen areas *Certificate Folders* and *Certificate Validation Options* are ready for input. (See also: Certificate Folders [page 216] and Certificate Validation Options [page 218].)

Related Information

Recommended Security Settings [page 227]
Security Settings Tab [page 215]

4.1.5.3.2 Certificate Folders

This screen area is only ready for input if you have selected a certificate.

Settings for the Certificate Folder

Field	Description
Store Type	Here you select the store for the certificate that you want to be validated. The following types are supported: • *Microsoft certificate store* For this setting, when a connection is being established, PCo automatically searches in the Microsoft certificate store folder for a server certificate. → Recommendation Use the *Microsoft certificate store* . • *File system certificate store* With this option, you can specify the store location for the certificates, which PCo is to trust, in the file system.

Field	Description
Trusted Certificates	Here you can specify the folder in which the trusted certificates are stored. If you have selected the *Microsoft certificate store*, this is the folder for the *trusted root certification authorities*. The system proposes this automatically. **Example** `Local Computer/Trusted Publishers` If you have chosen the *file system certificate store*, a directory is proposed in the file system with the following subfolders: • *certs*: Folder for trusted certificates If you store a certificate from the certificate chain in this folder, this certificate is trusted in the check. The certificate is regarded as not trusted if there is no certificate from the certificate chain stored here. • *crls*: Folder for certificate revocation lists • *private*: Folder for private certificates If you choose *Browse*, a dialog box appears where you can select another folder.
Issuer Certificates	Here you can specify the folder in which the certificates of a trusted issuer are stored. If you selected the *Microsoft certificate store*, this is the folder for the intermediate certificate authorities. This is proposed automatically. **Example** `Local Computer/Intermediate Certificate Authorities` If you have selected the *file system certificate store*, a directory is proposed in the file system with the subfolder *certs*. This folder is used to complete the certificate chain if the server does not send the complete certificate chain.

Field	Description
Rejected Certificates	Here you can specify the folder in which the rejected certificates are stored. If you are using the *Microsoft certificate store*, select *Untrusted Certificates* here. **Example** `Local Computer/Untrusted Certificates` If you have selected the *file system certificate store*, use a directory in the file system with the subfolder *certs* (folder for rejected certificates).

4.1.5.3.3 Certificate Validation Options

This screen area is only ready for input if you have selected a certificate.

Validation Options

Field	Description

Revocation Check	In this field you define how the revocation check of the server certificate is to be performed. You have the following options:
	
	- *No Check on Revoked Certificates*
	No check is carried out.
	- *Check Online Revocation Lists*
	The online check is a secure procedure but it can have a negative impact on performance.
	
	→ Recommendation
	This is the recommended setting in connection with the Microsoft certificate store.
	
	- *Check Offline Revocation Lists*
	  - If you are using the **Microsoft certificate store**, you need to copy all the relevant certificate revocation lists into the **Trusted Root Certification Authorities** directory.
	  - If you have selected the **file system certificate store**, you need to copy all related certificate revocation lists as `.crl` files into the revocation list folder.

Field	Description
Revocation Check Scope	Indicates the scope of the revocation check. You have the following options: • *Check End Certificate Only* Only the last certificate in a certificate chain is checked. • *Exclude Root Certificate from Check* • *Check Entire Chain* All certificates in a certificate chain are checked. → Recommendation This is the recommended setting.
Ignore Server Host Name	If you select this checkbox, the check results of the server host name are not taken into consideration. During the check, a comparison is made with the domain name system (DNS) name that is included in the certificate. The DNS name is the name of a server in a domain, for example: `server.domain.com`. → Recommendation SAP recommends that you do not set this indicator.
Ignore Validity Period	If you select this checkbox, you define that the validity period of the client certificate and the certificates in the certificate chain are not to be taken into consideration.

4.1.5.4 Tag Definition Tab

You subscribe to the tags here and configure the MQTT messages that the MQTT broker (server) is to send to PCo.

Use

On this tab, you define the tags of the MQTT broker (server) that you want to monitor. Moreover, you configure the template for the MQTT messages that are to be sent from the MQTT broker (server) to the source system in PCo to transfer the tag values.

> ℹ Note
>
> The MQTT payload that is sent from an MQTT broker can have any structure, so you need to model the data structure here in PCo.

Configure the Subscription

Subscription Settings

Field	Description
Selected Subscription	The tags that you want to monitor are grouped together in one subscription. You can create a subscription here with default settings by choosing *Create Subscription*. You can delete the currently selected subscription by choosing *Delete Subscription*. You can select an existing subscription from the dropdown box.
Subscription Name	The name of the selected subscription is displayed here. You can change the name.

Field	Description
Topic Filter	In this field, you enter the name of the topic filter to which the selected subscription belongs. There is a 1:1 relationship between a topic filter and a subscription. Each time the MQTT server publishes a message for this topic filter, that is, sends a message to PCo, the system checks whether the topic matches the topic filter. If this is the case, the MQTT source system receives the message, parses it, and inserts the tag values in the assigned subscription. A topic filter is a non-empty UTF-8 string. The topic filter is used to filter topic names. The topic filter of a subscription can contain special wildcard characters that allow a client to subscribe simultaneously to several topics. **i Note** Wildcard characters can only be used here in the *Topic Filter* field. They are **not** allowed to be used in the *Topic Name* field of the MQTT destination system. You can use single-level (+) and multilevel wildcards (#): - The **#** character is a multilevel wildcard that corresponds to any number of levels within a topic. - The plus sign (**+**) is a single-level wildcard that can be used on every level in the topic filter. For more information, see Wildcards in the Topic Filter [page 222]. **i Note** If you have entered a wildcard character in the topic filter, an additional tag with the name `TopicName` appears when you are browsing the source system namespace. Like all other tags, this tag can be added to the subscription items.

Field	Description
QoS	Quality of service: To make sure that a sent message reaches the recipient, MQTT defines three different quality of service (QoS) levels with which a message can be sent: • *0 - At Most Once* This service quality level is suitable for transferring sensor data. Messages might be lost. The message reaches the recipient at most once. • *1 - At Least Once* This service quality level ensures that the message reaches the recipient. The recipient might receive the message more than once. • *2 - Exactly Once* This level makes sure that the message reaches the recipient exactly once.
Payload Type	Specifies the format of the MQTT messages. The following formats are supported: • *JSON* • *XML* • *URL-Encoded* • *Formatted String*

Message Configuration

The following functions are available for configuring a message:

- Configure MQTT Message from Template [page 224]
- Test Parsing of Payload [page 224]
- Message Configuration: Define Parameters Manually (for JSON, XML, and URL-Encoded) [page 225]
- Message Configuration: Context Menu [page 226]
- Import Data Types into the Data Type Repository [page 226]

4.1.5.4.1 Wildcards in the Topic Filter

You can use the following wildcards:

- Multilevel wildcards
  The **#** character is a multilevel wildcard that corresponds to any number of levels within a topic. # represents the parent element and any number of child levels. You need to enter it on its own or together with a separator for topic levels. In either case, it has to be the last character that you enter in the *Topic Filter* field.

> ### Example
>
> If you enter **sport/tennis/player1/#**, the client can receive messages that are published using the following topic names:
>
> - `sport/tennis/player1`
> - `sport/tennis/player1/ranking`
> - `sport/tennis/player1/score/wimbledon`
>
> If you enter **sport/#**, the client can receive messages whose topic name contains `sport`. This is possible because the character # includes the parent level.
>
> If you only enter the wildcard character **#**, every application message can be received.
>
> Here are some examples of **invalid** entries:
>
> - **sport/tennis#**
> - **sport/tennis/#/ranking**

- Single-level wildcards
  The plus sign **+** is a single-level wildcard. The plus sign can be used on every level in the topic filter, including the first and the last level. If it is used, it needs to occupy an entire filter level. You can use it on more than one level in the topic filter and it can also be used in conjunction with the multilevel wildcard.

  > ### Example
  >
  > If you enter **sport/tennis/+**, this matches to messages with the topic name `sport/tennis/player1` and `sport/tennis/player2`, but not to `sport/tennis/player1/ranking`.
  >
  > If you enter **sport/+**, this matches to `sport/`, but not to `sport`.
  >
  > **+** is a valid entry.
  >
  > **+/tennis/#** is a valid entry.
  >
  > **sport/+/player1** is a valid entry.
  >
  > If you enter **/finance**, this matches to `+/+` and to '/+', but not to '+'.
  >
  > **sport+** is **not a** valid entry.

The server must **not** correlate topic filters that start with a wildcard character (# or +) with topic names that start with a **$ character**.

`$SYS/` has been widely adopted as a prefix for topics that contain server-specific information or that control APIs. Therefore, applications cannot use a topic with a leading $ character for their own purposes. A subscription to # will **not**, therefore, lead to messages, which are published for a topic that starts with a $ character, being received.

A subscription to `+/monitor/Clients` will not receive any messages that have been published for `$SYS/monitor/Clients`.

A subscription to `$SYS/#` will receive messages that are published for topics that start with `$SYS/`.

A subscription to `$SYS/monitor/+` will receive messages that have been published for `$SYS/monitor/Clients`.

In order that a client can receive messages of topics that start with `$SYS/` as well as topics that do **not** start with a $ character, he or she needs to subscribe to # and `$SYS/#`.

4.1.5.4.2 Configure MQTT Message from Template

You can choose the button *Configure Message from Template* to insert a specified character string in the dialog box that appears. This always makes sense if you already have the payload, for example, from a description or documentation. When you choose *Apply*, this data is automatically transferred as a parameter to the *Message Configuration* table and to the *Tags* table. This automatically creates the message configuration that is to be sent to PCo from the MQTT server. This **function can be used for all available payload types**. This procedure is very efficient and avoids typing errors.

Example

The syntax for a parameter name is structured in the following way for the payload type **Formatted String**:

```
{$<variable name>[:<variable type> ]}
```

The parameter name (variable name) needs to start with a letter and can only contain letters and numbers. You can also specify the parameter type (variable type). The parameter type (variable type) might be, for example: `byte`, `short`, `int`, `long`, `double`, `string`. All parameter types that are offered in the combo box are possible. If you haven't defined a parameter type, the default type `string` is used.

Here is an example of a message template for the payload type **Formatted String**:

```
Length: {$length:double}m, Width: {$width:double}m, Height: {$height:double}m
```

The MQTT message that arrives later in PCo looks like this, for example:

Length: 2m, Width: 1m, Height: 3m.

The parameters have been filled with tag values from the MQTT broker (server) and are displayed in the *Tags* table in the lower screen area. In this case, the *Message Configuration* table in the lower screen area is not input-enabled and remains empty.

Example

Example of a **JSON** template:

```
{"Print": "{inPrint:string}","Temperature": "{inTemperature:string}"}
```

The print and temperature parameters are copied to the *message configuration*. The variables `inPrint` and `inTemperature` are displayed in the *Input Variables* table.

4.1.5.4.3 Test Parsing of Payload

You can use this function to test the parsing of the payload. You can use the function for the payload types **JSON** or **Formatted String**. The payload must be known in advance.

You choose the *Test Parsing of Payload* button. You can then insert the payload strings in the text field on the *Payload* tab and choose the *Test* pushbutton. If the configuration is OK, you can display the correct values of the configured tags on the Tags tab.

> **i Note**
>
> During this test, you are not subscribed to an MQTT server, but instead you use a known message format.

4.1.5.4.4 Message Configuration: Define Parameters Manually (for JSON, XML, and URL-Encoded)

If you have selected JSON, XML, or URL-encoded as the *payload type*, you can add the tag parameters manually. Using the parameters you have entered, PCo then generates a template for the MQTT messages.

To be able to add parameters (tags) of the type `object`, do the following:

1. Choose  below the message configuration screen area.
   The *Add Tag* dialog box appears.
2. Enter the following data:

   Adding a Tag

Field	Description
Parent Parameter	The root parameter is displayed here: `$out`
Parameter	Here you enter the name of the parameter that you want to be used in the MQTT message. This might be a multile-vel property name.
Data Type	You can enter the data type of the tag here.
Tag Name	You enter the tag name of your choice here.

   The tags that are entered are displayed in the *Message Configuration* table.

   > **i Note**
   >
   > If you define a parameter of the type `object`, the PCo data type of the tag must be set to `System.Object`. Using other structured or array data types from the PCo data type repository leads to runtime errors, even if they correspond exactly to the message structure. The data type of the subscription item must also be `System.Object`. If you want to convert a message into a structured or an array data type from the data type repository, you define an output expression with the desired data types within a notification.

3. To enter a parameter of type **array**, proceed as follows:
   To specify that a specific parameter is an array, you have to add [0] to the parameter name to define the first array element. Alternatively, you can use a dynamic index, for example, by entering [i].

If you try to extend the same parameter again, the *Array Configuration* dialog box appears. You enter, for example, `1` in the *Array Index* field.

4.1.5.4.5 Message Configuration: Context Menu

You can also use the context menu to quickly enter parameters (tags) of an MQTT message configuration.

The context menu contains the following functions:

- Extend [Selected Mapping]
- Delete [Selected Mapping]
- Propose Tag
- Remove Tag

4.1.5.4.6 Import Data Types into the Data Type Repository

You can use the *Import Data Types into Data Type Repository* pushbutton to import the entire message configuration as a single structured data type into the PCo data type repository, so that in future you only have to define a single root variable of the type structured data type. (See also: Functions in the Import Data Types Dialog Box [page 587].)

The *Select Data Type* pushbutton takes you to the PCo data type repository, where you can select the relevant previously imported structured data type. In this case, the input of the root parameter as the only input parameter is sufficient to convert JSON files and pass them as a data structure to an MQTT broker.

> **i Note**
>
> If you define a parameter of the type `object`, the PCo data type of the tag must be set to `System.Object`. Using other structured or array data types from the PCo data type repository leads to runtime errors, even if they correspond exactly to the message structure. The data type of the subscription item must also be `System.Object`. If you want to convert a message into a structured or an array data type from the data type repository, you define an output expression with the desired data types within a notification.

4.1.5.5 Recommended Security Settings

This document describes the recommended security settings that you make on the *Connection* tab.

Recommended Settings

If the PCo MQTT client is running on Windows OS, the following security settings are recommended:

Recommended Settings on the Connection Tab

Field	Recommended Setting
Store Type	`Microsoft Certificate Store`
Trusted Certificates/Issuer Certificates/Rejected Certificates	For the certificates, you choose a storage location on the local computer or the current user. **i Note** It might be the case that the certificates are not visible to the current user during runtime if you start the agent instance with different credentials. As a result, the agent instance cannot be started.
Revocation Check	`Check Online Revocation Lists`
Revocation Check Scope	`Check Entire Chain`
Ignore Server Host Name	Do **not** select the checkbox.
Ignore Validity Period	Do **not** select the checkbox.

Certificate Chains for Client Authentication

If you want to use certificate chains for client authentication, you need to make the following settings:

- You need to install all intermediate certificates in the store for intermediate certificate authorities.
- You need to install all trusted root certificates in the store for trusted root certification authorities.

i Note

The Microsoft certificate store is **always** used for all client certificates.

The configurable certificate store type is only used during validation of the server certificates. (See also: Process for Validation of the Server Certificate Chain [page 228].)

Related Information

Connection Tab [page 209]
Process for Validation of the Server Certificate Chain [page 228]
Asymmetric Hash Algorithms [page 228]

4.1.5.5.1 Process for Validation of the Server Certificate Chain

This document describes how the certificate chain of an MQTT server is validated.

Validation Process

1. PCo sets up the certificate chain using the certificates that the MQTT server has sent.
   The folder with the trusted certificates and the folder with the certificates of a trusted publisher are taken into account. The folder with the rejected certificates is only used for the temporary storage of rejected certificates.
2. If the certificate chain is incomplete, the server certificate being checked is regarded as untrusted.
3. If it was possible to set up the certificate chain and at least one certificate was taken for this from the trusted certificates folder, the certificate is regarded as trusted for the validation.
4. A revocation check is only performed if the certificate chain is complete and the certificate is therefore regarded as trusted.
5. During the online revocation check, the `crls` folder (under the trusted certificates folder) is updated in accordance with the last certificate revocation list.

Related Information

Recommended Security Settings [page 227]

4.1.5.5.2 Asymmetric Hash Algorithms

The following asymmetric hash algorithms are supported:

- SHA1 DSA (1.2.840.10040.4.3)
- SHA256 RSA (1.2.840.113549.1.1.11)
- SHA384 RSA (1.2.840.113549.1.1.12)
- SHA512 RSA (1.2.840.113549.1.1.13)

> **i Note**
>
> The number combination in brackets is the object identifier (OID) in each case.

The following asymmetric hash algorithms are **not** supported:

- MD2 RSA (1.2.840.113549.1.1.2)
- MD5 RSA (1.2.840.113549.1.1.4)

Related Information

Recommended Security Settings [page 227]
Process for Validation of the Server Certificate Chain [page 228]

4.1.6 Timer Source System

A timer is a configuration element that changes its value periodically at specified times.

You can use timers to trigger actions using PCo at configurable, usually periodic, times. You can use the timer source system, for example, for the following tasks:

- You want to record a reading from a machine at fixed, regular intervals in order to store a time series of measurement data for documentation purposes.
- You want to automatically trigger a cleaning operation in a production plant at a specific time in the week after the end of the shift.
- You can set up a periodic automatic data backup using the timer source system and an implementation of enhanced notification processing delivered by SAP. (See also: Setting Up an Automatic Backup [page 597].)

You can maintain one or multiple timers and their schedules in a timer source system.

If you create an agent instance for a timer source system, you can generate subscription items there for the configured timers. When the timer values change, notification messages are triggered and these can be used to call up any destination system, for example, a simulation destination system or a multiple call destination system. You need to configure any further actions in the multiple call destination system. (See also: Create Timers and Trigger Notification Messages [page 232].)

The following tabs are available for configuring the timer source system:

- Timer Configuration Tab [page 229]
- Reliable Connection Tab [page 142]

4.1.6.1 Timer Configuration Tab

On this tab, you create one or multiple timers and define the corresponding schedules.

1. Create a source system of the type *timer source system*.
   The *Timer Configuration* tab appears.
2. Choose *Add Timer*.
   The timer with the name *timer0* is added to the list of timers.
3. Change the name to `Timer1`, for example, if necessary and enter a description.
4. On the *Repetition* tab, enter the repetition parameters that are valid for the timer for one day; in other words, how often, in which time period, or at what time you want the timer to be repeated on a specific day. On the *Schedule* tab, you can define how you want this daily repetition pattern to be executed over days, weeks, and months. If you don't make any changes here, the repetition pattern you configured for one day is executed on a daily basis.
   Enter the following data for the timer on the *Repetition* tab:

Settings for the Repetition Frequency

Field	Description
Constantly at Fixed Time Intervals	With this (default) setting, the timer is executed at periodic time intervals after the agent instance has been started in the specified time interval. In addition, maintain the fields for the *time interval* in the *Parameters* screen area. You can maintain the time interval in hours, minutes, seconds, and milliseconds. **Example** Enter, for example, the value `30 seconds` in the *Time Interval* field. **Result:** The timer event **Send Notification Message** is executed every 30 seconds as soon as the corresponding agent instance has been started.
At Fixed Time Intervals Between Start and End Time	If you select this option, the timer events are only triggered within a specified time period between the *start and end time*. **Example** On December 5, you want to send a notification message every 60 seconds between 3:00 p.m. and 4:00 p.m. You have to configure the following: 1. Select this option button. 2. Enter `1 minute` in the field for the *time interval*. 3. Enter `03:00:00 PM` as the *start time* and `04:00:00 PM` as the *end time* . 4. Choose the *Scheduling* tab and select the *Monthly* option. 5. Select *December* and *day 5*.

Field	Description
Once a Day at Specified Start Time	If you select this option, the timer event is only executed once a day at a specified *start time*. **Example** You want a notification message to always be sent on Wednesdays at 3:30 p.m. 1. Select this option button. 2. Enter `03:30:00 PM` in the *Start Time* field. 3. Choose the *Scheduling* tab and select the *Weekly* option. Select *Wednesday*.

The times entered refer to the **local time zone of the user** who is configuring the timer source system. For clarity, the Management Console displays this time zone including the active summer time.

Note

Internally, the system uses **UTC time** for execution of the timer. During the shift from summer to winter time and vice versa, the timers therefore remain invariant with respect to the time shift. Therefore a timer that you configure during summer time, for example, is, after the shift to winter time, executed an hour earlier in relation to the local time that is then valid. A user of the same Management Console who has specified in his or her Windows settings a different time zone to the original time zone, sees the configured times converted to his or her time zone.

5. Define additional parameters:

Additional Parameters

Field	Description
First Execution	In this field, you specify when the timer should start: ○ Immediately after the agent instance start ○ At the start of the first minute after the agent instance start ○ At the start of the first hour after the agent instance start ○ At a specified start minute You can only select this option in conjunction with the setting *Constantly at Fixed Time Intervals*.

Field	Description
Start Minute	This field is only ready for input in conjunction with the execution option *At Specified Start Minute*. You specify here at exactly which minute after the next hour you want the timer to start for the first time. The timer runs from then on according to the time interval that you specified. **Example** You have entered `30 seconds` as the time interval, and entered **2 as the start minute**. If you now start the agent instance at 16:48, the timer is triggered for the first time at 17:02 and thereafter every 30 seconds.

6. Choose the *Scheduling* tab.
   On the *Scheduling* tab, you can choose between a daily, weekly, or monthly execution of the timer event.

   Settings for the Schedule

Field	Description
Daily	Select this option if you want the timer to be executed on a daily basis.
Recurrence [x] Day(s)	Specify whether you want the timer event to be triggered, for example, every day [1] or every 3 days [3].
Weekly	Select this option if you want the timer to be executed on a weekly basis.
Recurrence [x] Week(s) / Weekdays	Specify whether you want the timer event to be triggered, for example, every week [1] or every 2 weeks [2]. You can also select a specific day or days of the week on which you want the timer event to be executed, for example, Wednesday.
Monthly	Select this option if you want the timer to be executed on a monthly basis.
Months / Days	Select the month or months in which you want the timer event to be executed. Select one or more days too. This allows you to specify a fixed date, for example, September 5.

4.1.6.2 Create Timers and Trigger Notification Messages

This document uses an example to describe how you need to proceed so that the timer sends notification messages in the time intervals you have specified.

Procedure

1. Create a timer source system and define a timer, for example, `Timer1`.
2. Define for the timer **Timer1** that you want a notification message to be sent every 30 seconds. (See also: Timer Configuration Tab [page 229].)
3. Create an agent instance for the timer source system and on the *Subscription Items* tab, browse for and select the timer **Timer1**.
   Timer1 is added as a subscription item. It has the data type `System.DateTime` and cannot be changed.
4. Create a notification for the agent instance.
5. In the notification, click on the *Output* tab and choose *Generate Expressions*.
   Timer1 is added as an output expression.
6. Click on the *Destinations* tab and select a simulation destination system.
7. Start the agent instance.

Result:

The agent instance of the timer is started. A notification message is sent to the simulation destination system every 30 seconds. You can display the messages in the folder of the simulation destination system.

Every message contains the subscription item with the values [CDATA[event_time]] where [event_time] is the time of the notification message.

Notes:

- Even if you don't want to use the expression `Timer1` as the input parameter for the destination system call, you need to create it in any case, so that the notification is triggered for a timer event.
- You can also use the subscription item for the timer in the trigger condition of the notification if your requirements for execution of the timer are not covered by the configuration options of the timer source system.

4.1.7 Proficy Historian Source System

Use

The **GE Fanuc Proficy Historian** is an agent that establishes a connection to an existing GE FANUC Proficy Historian server.

You can use the agent in the notification and query processes.

The following tabs are available for configuring this source system:

- Server Settings tab (see below)
- Aliases tab [page 138]
- Reliable Connection tab [page 142]

Prerequisites

You have created a Proficy Historian source system.

Procedure

1. To configure settings, on the *Plant Connectivity Management Console*, select a Proficy Historian source system and click on the *Server Settings* tab.
2. For the information you need to enter, use the following table:

Field	Description
Server	Name of the machine on which Proficy Historian is installed
User Name	User-defined name for connecting to Proficy Historian. If a server is on a domain, enter the value as domain name/user name
Password	User's password for connecting to Proficy Historian
Minimum Elapsed Time	Minimum period of time that must have elapsed before a new event is sent. The time is specified in milliseconds. If more than one event occurs in the stipulated time, only one event is sent. **Example** You have specified 5 seconds as the minimum elapsed time between two events. Event 1 occurs and is sent. Event 2 occurs 4 seconds later and is therefore not sent. If the time between the two events is greater than the stipulated timespan, both events are sent. **Example** You have specified 5 seconds as the minimum elapsed time between two events. Event 1 occurs and is sent. Event 2 occurs 10 seconds later and is also sent.

To test the connection, choose the *Test Connection* pushbutton.

4.1.8 OSISoft PI Source System

Definition

Agent that provides the connectivity to an OSIsoft PI server. The agent supports the PI SDK of 1.3.6 or higher and establishes a connection with PI 3 version 3.3 or higher.

> **Note**
>
> If you want to use an *OSIsoft PI source system*, you need to install the *PI SDK version 1.3.6 or higher*.

Structure

You have the following tabs available for configuring a PI system:

- Server tab [page 235]
- Settings tab [page 236]
- Aliases tab [page 138]
- Reliable Connection tab [page 142]

4.1.8.1 PI Source System: Server Tab

Procedure

1. To configure the settings for a PI source system, on the *Plant Connectivity Management Console* screen, select a PI source system and click on the *Server* tab.
2. For the information you need to enter, use the following table:

Field	Description
PI server	Machine where the PI server is installed > **Note** > If a server is not available in the dropdown box, click the *Launch PI Connection Manager* pushbutton. The available PI servers are displayed in the *PI Connection Manager* dialog box and you can choose the server you want.
Use PI Trust	Determines that the PI server grants access based on server configuration
User Name	Specifies the user that enables the connection to the PI server

Password	Specifies the password that creates the connection to the PI server

Next Steps

OSISoft PI Source System [page 234]

4.1.8.2 PI Source System: Settings Tab

Procedure

1. To configure settings for a PI source system, on the *Plant Connectivity Management Console* screen, select a PI source system and click on the *Settings* tab.
2. For the information you need to enter, use the following table:

Field	Description
DigStr as String	If you select this checkbox, the DigStr values are interpreted as a string.
Call Type	○ If you select the call type *Asynchronous*, PCo submits a tag request to the PI server that can be terminated by the PI server. ○ If you select the call type *Synchronous*, PCo submits a tag request to the PI server that cannot be terminated by the PI server.
Use GetPoints2	○ If you are using a new version of the PI SDK, you must select the *Use GetPoints2* checkbox. In this case, all the searches use the *GetPoints2* method. This method results in better performance. ○ If you are using an older version of the PI SDK, you must deselect the checkbox. In this case, all the searches use the original *GetPoints* method.

Number of Threads	Here you can enter the number of parallel threads (processes) (see also: Thread). The default value is *6*. If you enter **1**, single threading is used. If you enter a value between **2** and **9**, multithreading is used. If there is a high data load, the performance improves if you use multiple threads.
Minimum Elapsed Time	Indicates the minimum time interval between two events. PCo creates a notification message for each event. The time specified here is the minimum time that needs to have passed between both notification messages. **Example** You have entered 5 milliseconds. This means that a notification message is only sent every 5 milliseconds, even if an event occurs every 4 milliseconds.
Policy for Storing Tag Values with Identical Time Stamps	You can specify here how you want PCo to deal with duplicate tag values. Sometimes there are two values with identical time stamps for one tag. You have the following options: ○ *Insert All Tag Values* ○ *Replace Existing Tag Values*

Next Steps

OSISoft PI Source System [page 234]

4.1.9 Asset Framework Source System

Use

PI Asset Framework (PI AF) is an integral part of the *OSiSoft PI system*. The PI system itself consists of software products that form an infrastructure for realtime data and events. The PI system is used to collect, analyze, and visualize as well as archive the data of, for example, production facilities. The *PI Asset Framework* enables the definition of a consistent hierarchical display of production facilities. A hierarchical breakdown into all parts of a production facility is possible down to the level of sensors and tags.

The Asset Framework Agent can establish the connection from PCo to an *OSISoft PI system*. In this way, you can use the AF system as a data source.

The AF source system can be used in PCo in the **query process and the notification process**. The AF source system is an historian system.

The AF agent supports the following tag features in the PCo system:

- Statistical Evaluations
- Retrieve
- Store
- History Store
- Mass Store
- Subscribe
- Unsubscribe
- Regex Mask
- Groups
- Secondaries
- Metadata
- Aliases

Structure

The following tabs are available for configuring an Asset Framework source system:

- Server tab [page 238]
- Settings tab [page 239]

4.1.9.1 Server Tab

Use

Here you make the settings for the connection to the server on which the *PI Asset Framework* is running:

Field	Description
Server	Machine on which the OSiSoft PI system is installed. **i Note** If a server is not available in the dropdown box, click the *Refresh Server List* pushbutton. You can then choose the server you want.
Use PI Trust	Determines that the server grants access based on the server configuration.

Field	Description
User Name	The user that enables the connection to the PI server
Password	The password that creates the connection to the PI server

More Information

Asset Framework Source System [page 237]

4.1.9.2 Settings Tab

Use

You make the settings for the connection to the Asset Framework source system here:

Field	Description
Monitor PI Points	If you choose the *Monitor PI Points* option, PCo can monitor the PI points on the PI server. In this mode, the Asset Framework agent works exactly like a legacy PI agent. See also: OSISoft PI Source System [page 234]
Monitor AF Assets	If you choose this option, PCo monitors the AF assets.
Max. Number of Events	Indicates the number of historical events that are to be called up during the data polling. (See also: Polling.)
Update Rate	Indicates the interval for polling the data in milliseconds. This is the time that needs to have passed between two consecutive query operations. The events must not be sent faster than specified here. **i Note** The parameters *Max. Number of Events* and *Update Rate* need to be selected in such a way that tag changes can be passed on by PCo in as timely a fashion as possible.
Present Digital State as String	With this setting, you define that the status of the PI points is to be represented as a string.

Field	Description
Policy for Storing Tag Values with Identical Time Stamps	You can specify here how you want PCo to deal with duplicate tag values. Sometimes there are two values with identical time stamps for one tag. You have the following options: • *Insert All Tag Values* • *Replace Existing Tag Values*

More Information

Asset Framework Source System [page 237]

4.1.10 Citect Source System

Definition

You can use a Citect source system to establish the connection to a CitectSCADA system.

CitectSCADA is a control system for the monitoring and control of production plants. The system records, historicizes, and supplies log data from the various systems in production. See also: https://www.citect.schneider-electric.com/scada/citectscada

Use

You can use the Citect agent in the query process [page 43] only.

Structure

You have the following tabs available for configuring the connection to a Citect source system:

- Settings tab [page 241]
- Aliases tab [page 138]
- Reliable Connection tab [page 142]

Integration

A CitectSCADA system does not structure its data as a hierarchy but in tables. However, the browser of the PCo agent displays the data in two levels.

You can specify the table from which the data is to be read out on the *Settings* tab.

The Citect agent can query the following metadata:

- Data type
- Description
- Minimum value
- Maximum value

4.1.10.1 Citect Source System: Settings Tab

Prerequisites

To be able to set up a Citect source system in the PCO Management Console, you need to copy the following files of the Citect SCADA installation to the PCo system directory, for example, `c:\Program Files\SAP \Plant Connectivity\System`:

- CTAPI.DLL
- CT_IPC.DLL
- CTENG32.DLL
- CTRES32.DLL
- CTUTIL32.DLL

It is necessary to copy the files since the Citect agent uses the CtApi interface of the CitectSCADA system.

If PCo is not installed on the same machine as the CitectSCADA system, the CitectSCADA configuration (citect.ini) must also contain the following line: `[CTAPI] Remote=1`

For more information, see the documentation for CitectSCADA.

Procedure

1. To configure the settings for a Citect source system [page 240], on the *Plant Connectivity Management Console* screen, select a Citect source system and click on the *Settings* tab.
2. For information you need to enter, use the following table:

Field	Description
Server	Here you enter the server on which the Citect SCADA is installed.
User Name	Here you enter the user name that you use to log on to the Citect SCADA system.
Password	Here you enter the user password that you use to log on to the Citect SCADA system.
Table	Here you specify the Citect table on which the Citect agent is to work: ○ TAG table ○ TREND table The TREND table provides historical data. ○ Blank If you do not enter anything, you can browse in the two tables as in a hierarchy.

Results

The connection to the source system has been created. You can now create the agent instance and, if necessary, the destination system.

4.1.11 IP21 Source System

Definition

You can use the IP21 source system to establish a connection to the *Aspen InfoPlus.21* database. Aspen InfoPlus.21 is a realtime database with configurable history repositories.

You can use the IP21 agent for the query process only. You can use the IP21 agent to read realtime and historical data.

Structure

You have the following tabs available for configuring an IP21 source system:

- Server Settings tab [page 243]
- Aliases tab [page 138]

- Reliable Connection tab [page 142]

Integration

The IP21 source system provides the following statistics functions:

- Minimum (`MIN`)
- Maximum (`MAX`)
- Arithmetic average (`AVG`)
- Time-weighted average (`TWA`)
- Integral function (time-weighted total; `TOT`)
- Standard deviation (`SDV`)
- Time-weighted standard deviation
- Variance
- Time-weighted variance
- Number of values with good data quality (`NUMBERGOOD`)
- Number of values with poor data quality (`NUMBERNOTGOOD`)

More Information

Statistics Functions for Agents [page 265]

4.1.11.1 IP21 Source System: Server Settings Tab

Prerequisites

- In the *Plant Connectivity Management Console* and the *IP21 database*, you need a Windows user with the appropriate authorizations.
- The IP21 agent uses the *IP21 DBApi interface of the IP21 database*. You therefore need to copy the following files of the IP21 installation to the PCo system directory, for example, `c:\Program Files\SAP\Plant Connectivity\System`:
  - `cimsrvapi.dll`
  - `CimWin32Util.dll`
  - `infoplus21_api.dll`
  - `ip21admin_client.dll`
  - `ip21ezrpcw32.dll`
  - `ip21winrpc32.dll`
  - `libc21.dll`

- The following fields must exist in the tag definition in the IP21 source system in order for the IP21 agent to function:
  - `IP_VALUE` (value of tag)
  - `IP_VALUE_TIME` (time of value change)
  - `IP_VALUE_QUALITY` (quality of tag)
  - `IP_#_OF_TREND_VALUES` (number of historic values)
  - `IP_TREND_VALUE` (historic value)
  - `IP_TREND_QSTATUS` (quality of historic value)
  - `IP_TREND_QLEVEL` (expected value of quality)
  - `IP_DESCRIPTION` (description)
  - `IP_GRAPH_MINIMUM` (minimum value)
  - `IP_GRAPH_MAXIMUM` (maximum value)

Procedure

You define the settings for the IP21 database here. (See also IP21 Source System [page 242].)

Field	Description
Server	Here you enter the server on which the IP21 database is installed.
Use Long Record Name, If Possible	By choosing this indicator, you define how you want the IP21 records to be read. If you set the indicator, the long record names are read. Until `IP21 Version 7.3` the record names of the tags were maximum 24 characters long. As of `IP21 Version 7.3` the record names can be up to 256 characters long. New methods have been introduced by IP21 for reading the new (long) record names. With the old methods, only records with 24 characters can be read. With the new methods, all records can now be read. To make sure that the new methods are used, you need to set the indicator.

Result

You can now define an alias. The following fields of the IP21 source system can be read as metadata by PCo when an alias is created:

- `IP_DESCRIPTION`
- `IP_GRAPH_MINIMUM`
- `IP_GRAPH_MAXIMUM`

All other fields of a selected tag are read as secondary elements.

More Information

Source System: Aliases Tab [page 138]

4.1.12 File Monitor Source System (Deprecated)

Definition

> ⚠ Caution
>
> This source system only still exists for reasons of compatibility. If you want to monitor a directory on a computer with a source system, use the file system source system [page 255].
>
> Should you still want to use the file monitor source system in exceptional cases, you have to select the *Allow Creation of Deprecated Configuration Elements* checkbox under *Tools* > *Options* > *Global Settings* > *Compatibility* (see: Compatibility Settings [page 17]).

You can use the file monitor source system to monitor a specific directory locally on your computer or remotely on another computer in the network.

In the *PCo Management Console*, you define which directory you want to be monitored using the file monitor source system. If PCo finds a file that meets the configuration criteria that you specified in the *PCo Management Console*, PCo generates, for example, a notification message and sends it to the connected destination system. In this way you can transfer file contents to the destination system. You can also define specific actions that can be performed for the files that were found, for example, renaming a file.

> ✤ Example
>
> Examples of usage of the file monitor source system are testing and measuring devices as well as special machines that are often equipped with PC applications for user guidance. The plant data and test results are stored in the log files. You use the file monitor source system to be able to read out this data. The file monitor source system checks the directories continuously for new files and transfers these data records.

Use

You can implement the file monitor source system in the notification process [page 35] and in the query process [page 43]. For more information about query functions for the file monitor source system, see Query Functions of the File Monitor Source System [page 253].

Structure

The following tabs are available for the file monitor source system:

Settings Tab [page 247]

Authentication Tab [page 253]

Reliable Connection Tab [page 142]

Integration

Access Authorizations

The file monitor source system allows access only to the folders that are not system folders.

> **i Note**
>
> The following folders are system folders:
>
> - Root of a disk drive
> - \Windows
> - \Program Files
> - \Program Files
> - \Users

The file monitor source system allows access only to the folders for which the user has one of the following access authorizations:

- All authorizations (*Full Control*)
- Individual authorizations *Read, Write, Delete*

The agent instance then runs using either a local system account or in a network in the user's domain that the file monitor source system uses.

To grant the user the authorizations for a particular folder, proceed as follows:

1. Call the Explorer.
2. Browse for the folder for which you want to grant access authorizations.
3. Click on the folder.
4. Choose the right-hand mouse button. Choose *Properties* from the context menu.
5. Click the *Security* tab.
6. Check the list *Group or User Names*.

   > **⚠ Caution**
   >
   > Ensure that the exact user that the file monitor source system uses is entered in the list. The user must be stated explicitly, it is not possible to use a group.

7. Ensure that either *Full Control* or *Read/Write/Delete* is set for the user.

The **authorization checks** are performed in the following places:

- While PCo performs the Search Folders function, the folder that you want to monitor or rename is checked.
- When the agent instance is started it is also checked whether there are appropriate authorizations for the folder. If the authorizations are missing, the agent instance cannot be started and the system issues a corresponding message.
- When a tag query is called it is also checked whether the authorizations exist.

4.1.12.1 File Monitor Source System: Settings Tab

Procedure

1. Create a source system of the type *file monitor source system* (see File Monitor Source System [page 245]).
2. Click the *Settings* tab.
3. Enter data with the help of the following table:

Field	Description
Folder to Monitor	Here you specify the folder that is to be checked for the existence of a specific file. You can use the *Browse* push-button to search for the folder in the directories. Example `C:\toscan\tester1` or `\\computer\share\folder`
Monitoring Frequency (ms)	Specifies how often the folder is to be checked. The default is 1000 milliseconds (1 second).

Field	Description
Processing Order	If you want to process the files, you can specify the sequence for processing here. The files are then sorted according to the time stamp that was saved when the file was created. PCo checks all files in the specified folder that fulfill the criteria you entered in the *File Mask* field. Then it sorts the list of these files according to the setting made here. The following options are available: ○ *Oldest Files First* Files with the oldest time stamp are processed first. This is the default setting. ○ *Newest Files First* Files with the newest time stamp are processed first. ○ *Random (Optimal Speed)* With this setting, the list of files is **not presorted** according to the time stamp. This leads to a faster transfer of files to PCo.
File Mask	In this field, you can specify the search criteria for the files that are relevant for the destination system. The agent instance then searches in the folder for the files that correspond to the specified file type. The default setting is `*.*` **Example** You can enter the following, for example: ○ `*.log` With this setting, PCo searches for files with the file type **.log**. PCo sends a notification message if a file of this type was found. ○ `abc*.*` ○ `file1.z??`
Pass File Contents to Destination	With this indicator, you specify that the file system source system is responsible for reading the file contents in the memory and then for transferring the data to the destination system. If you do not set the indicator, the file contents are not read; instead, only the file name will be sent to the destination system. Default: The indicator has been set.

Field	Description
Notify Only If File Is Unlocked	PCo triggers a notification message only upon removal of specific file attributes or of a lock. A file is read-only, or it is currently locked since it was opened by another application. In this case, PCo cannot send a notification message.
Ignore Unavailability of UNC Folders	You have configured a network folder specified with a UNC path as the folder to be monitored. If this folder is no longer accessible at the runtime of the agent instance, the agent instance usually gets the *faulty* status. If you set the *Ignore Unavailability of UNC Folders* indicator, PCo ignores this error and continues monitoring as soon as the folder is available again. The error is simply logged as an information message in the agent instance log. When the indicator is set, the system behavior corresponds to the behavior of PCo in *release 2.2*.

Field	Description
Action	Specifies how the file that was found is to be dealt with. The following actions can be selected: ○ *No Action* This is the default setting. With this setting, the file is not changed. The fields *Destination Folder* and *Rename Mask* are not ready for input. ○ *Rename* With this setting, you can rename the file and if necessary also move it to another folder. You need to specify the destination folder in the *Destination Folder* field. **i Note** If you have entered `*.*` in the *Rename Mask* field, the folder to be monitored and the destination folder can be the same. ○ *Move to* With this setting, you can move the file to another folder. You need to specify the destination folder in the *Destination Folder* field. You can use the *Browse* pushbutton to search for the folder in the directories. **⚠ Caution** The folder to be monitored and the destination folder cannot be the same. The *Rename Mask* field is not ready for input with this setting. ○ *Delete* With this setting, the found file is deleted. The fields *Destination Folder* and *Rename Mask* are not ready for input. ○ *Copy* You can use this setting to copy the file into another folder that you specify in the *Destination Folder* field.
Destination Folder	Specifies the destination folder to which the file is to be moved. (Only relevant for the actions *Move* and *Rename*).

Field	Description
Rename Mask	Here you can specify the new file name after renaming. You can rename the files according to the following options: ○ You can specify date and time elements that are then integrated into the changed file names. For example, you enter **`[LO]-[YYYY]-[MM]-[DD].[RO]`**, which means that the current date is inserted into the file names. See Rename File [page 251]. ○ You can extract the original file name and place it to the left or right side of the "." (period). ○ You can extend the file name using a prefix or a suffix. ○ You can keep the original file extension (the file type) or replace it with another one. ○ You can extract textual or numeric parts of the file name and use them for the new file name. ○ For more information, see Rename File [page 251].

4. Save your entries.

Results

You have created the source system. You can now create the agent instance and make settings for it. (See Example: Agent Instance Settings for a File Monitor Source System [page 254].)

4.1.12.1.1 Rename File

In the *Rename Mask* field, you can define the new file name. For example, you can integrate date and time elements into the new file names.

> ✤ Example
>
> **Example 1**
>
> The original file name is: `c:\temp\myfile.txt`
>
> In the *Rename Mask* field, you enter **`[LO].log`**.
>
> After being renamed, the file name is: `c:\temp\myfile.log`
>
> In this example, [LO] means that the original file name to the left of the . (period) is retained.
>
> **Example 2**
>
> The original file name is: `c:\temp\myfile.txt`

In the *Rename Mask* field, you enter **[LO]-[YYYY]-[MM]-[DD].[RO]**.

After being renamed, the file name is: `c:\temp\myfile-2011-02-04.txt`

In this example, **[LO]** means that the original file name is unchanged to the left of the inserted date. **[RO]** means that the original file name to the right of the . (period) is retained.

The table below states which values in the *Rename Mask* field can be used to rename the file name:

Input Value	Description
[LN]	To the left of the period, the original file name is retained. To the right of the period, you can insert numbers only.
[LT]	To the left of the period, the original file name is retained. To the right of the period, you can insert text only.
[LO]	To the left of the period, the original file name is retained. To the right of the period, you can insert numbers and text.
[RN]	To the right of the period, the original file name is retained. To the left of the period, you can insert numbers only.
[RT]	To the right of the period, the original file name is retained. To the left, you can insert text only.
[RO]	To the right of the period, the original file name is retained.
[YYYY]	The current year is inserted into the file name.
[MM]	The current month is inserted into the file name.
[DD]	The current day is inserted into the file name.
[HH]	The current hour (24 hour format) is inserted.
[NN]	The current minute is inserted.
[SS]	The current second is inserted.
[SEQ]	Insert an incremented number (adds +1 to each call).

4.1.12.2 Authentication Tab

Context

With the entries on the *Authentication* tab, you ensure that the source system has access to the directories to be checked that are stored on other PCs in the network.

Procedure

1. Click the *Authentication* tab.
2. Enter data with the help of the following table:

Field	Description
Authentication Required	You use this checkbox to define whether authentication is necessary. If you select the checkbox, the fields *Domain*, *User Name*, and *Password* are ready for input.
Domain	You enter your Windows domain here.
User Name	You enter the user name for the PC in the network on which the folder to be checked is stored.
Password	You enter the password.

3. You can use the *Test* pushbutton to check whether the user name and password are correct.
4. Save your entries.

4.1.12.3 Query Functions of the File Monitor Source System

Use

The file monitor source system supports *tag queries*. Starting from the connected destination system (for example, Business Suite system or SAP MII), you can start tag queries and thus display groups (directories), display tags (files), and read and write tags (files).

Features

The following tag queries are available:

- LIST GROUPS
  Listing of directories under a defined directory
- RETRIEVE
  Call content of a defined file
- STORE
  You can use this tag query to write a STRING to a file.

4.1.12.4 Example: Agent Instance Settings for the File Monitor Source System

Use

The example below describes how you can set up an agent instance for a file monitor source system to connect the agent instance of the source system with a simulated destination system for test purposes.

Prerequisites

You have created a source system. (See File Monitor Source System: Settings Tab [page 247].)

Procedure

1. Create an agent instance for the source system.
2. After the PCo system has created the agent instance, click on the *Host* tab. In the *Log Level* field, enter `Information` or `Verbose`.
   For more information about these settings, see Agent Instance: Host Tab [page 413].
3. On the *Subscription Items* tab, click on the *Browse* pushbutton at the lower right of the screen.
   The PCo system displays the *Address Root* node.
4. Expand the *Address Root* node and browse the list of available tags. Choose the *Add Selected Items* pushbutton for all subscription items to subscribe to the tags.
5. Click on the *Query Ports* tab and select the source system type **SAP MII** so that the SAP MII system can send tag query commands to the file monitor source system.
   For more information, see Agent Instance: Query Ports Tab [page 419].
6. Save the agent instance.
7. In the final step, you create the notification that sends the notification information to the destination system. To do so, click on the icon (+) to add a notification.
8. Enter a name and a description for the notification.

> ⚠ Caution
>
> You may not select the *Template* checkbox here since otherwise only a notification template is created.

9. Click *OK*.
10. Click on the *Output* tab in the screen area for the notification.
11. Choose the *Add Expression* pushbutton. You can now add the individual subscription items.
12. Now go to the *Destinations* tab and choose the *Add Destination System* icon.
13. Select a destination system for the simulation and enter a name and a description.
14. After you have chosen *OK*, save the notification.

Result

The connection for the simulation run is established for test purposes.

4.1.13 File System Source System

Use

With the file system source system, you can monitor a specific directory locally on your computer or in another network drive and send the contents of files to one or multiple destination systems. In addition, you can read and write the contents of text files within *enhanced notification processing*.

In the *PCo Management Console*, you define which directory is to be monitored using the file system source system. If PCo finds a file that meets the configuration criteria that you specified in the PCo Management Console, PCo generates, for example, a notification message and sends it to the connected destination system. In this way you can transfer file contents to the destination system. The files can be text files or binary files. You can also define specific actions that can be performed for the files that were found, for example, renaming a file.

The *file system source system* supports the following functions:

- Processing of binary files
  The content of binary files is sent to a destination system as a Base64-coded character string.
- Uploading of binary files (see: Upload Binary File Content [page 261])
- Selectable text encoding (“code pages”) for text files
- Reading and writing the contents of a text file within enhanced notification processing

Configuration

Settings Tab [page 256]

Authentication Tab [page 253]

4.1.13.1 Settings Tab

Context

On this tab, you make the settings for the folder that is to be monitored and for file processing.

Procedure

1. Create a source system of the type *file system source system* (see File System Source System [page 255]).
2. Click the *Settings* tab.
3. Enter data with the help of the following table:

Field	Description
Folder to Monitor	Here you specify the folder that is to be checked for the existence of a specific file. You can use the *Browse* push-button to search for the folder in the directories. When file contents are read and written within *enhanced notification processing (ENP)*, this folder also represents the root directory to which file names (which you define as tags on the *Source System Tags* tab of the ENP screen *Assignment of Enhancement Variables*) refer. **Example** `C:\toscan\tester1` or `\\computer\share\folder`
Monitoring Frequency (ms)	Specifies how often the folder is to be checked. The default is 1000 milliseconds (1 second).

Field	Description
Processing Order	If, within a notification operation, you want to react when files appear or files change, you can specify the processing sequence here. The files are then sorted according to the time stamp that was saved when the file was created. PCo checks all files in the specified folder that fulfill the criteria you entered in the *File Mask* field. Then it sorts the list of these files according to the setting made here. The following options are available: ○ *Oldest Files First* Files with the oldest time stamp are processed first by the file source system. This is the default setting. ○ *Newest Files First* Files with the newest time stamp are processed first by the file source system. ○ *Random (Optimal Speed)* With this setting, the list of files is **not presorted** according to the time stamp. This leads to the files being transferred more quickly to PCo.
File Mask	In this field, you can specify the search criteria for the files that are relevant for processing within a notification process. Only files that fulfill the predefined criteria are sent to a destination system. The default setting is *.*. **Example** You can enter the following, for example: ○ `*.log` With this setting, PCo searches for files with the file type **.log**. PCo sends a notification message if a file of this type is found. ○ `abc*.*` ○ `file1.z??`
Pass File Contents to Destination	If you set this indicator, the file contents are sent to a destination system. If you do not set the indicator, the file contents are not read; instead, only the file name is sent to the destination system. Default: The indicator is set.

Field	Description
Notify Only If File Is Unlocked	If this checkbox has been selected, PCo only triggers a notification message if the *read only* indicator for a file is not set or has been deselected. **⚠ Caution** If you do **not** select the checkbox and the *read only* indicator is set for a file, this can lead to problems when you try to delete, rename, or move the file.
Ignore Unavailability of UNC Folders	You have configured a network folder specified with a UNC path as the folder to be monitored. If this folder is no longer accessible at the runtime of the agent instance, the agent instance usually gets the *faulty* status. If you set the *Ignore Unavailability of UNC Folders* indicator, PCo ignores this error and continues monitoring as soon as the folder is available again. The error is simply logged as an information message in the agent instance log.
File Type	Here you enter the file type that is to be processed: ○ Text PCo assumes that the contents of a file consist of text. The text content is read using the encoding selected in the *Text Encoding* field and is sent to the destination systems as text. ○ Binary PCo assumes that the contents of a file consist of binary data. This might be pictures or executable files, for example. PCo converts the contents of a binary file into a Base64-coded character string and sends the character string to the destination systems.
Text Encoding	Using the chosen text encoding, the contents of the text files are read from the directory being monitored and converted by PCo into a Unicode character string before this is sent to the destination systems. This setting is only relevant if you have chosen the **file type text**.
Test File and Sample Text	You can use a text file to test the chosen text encoding. The file contents are displayed in the *Sample Text* field. This option is only available if you have chosen the **file type text**.

Field	Description
Action	Specifies how the file that was found is to be dealt with. The following actions can be selected: ○ *No Action* This is the default setting. With this setting, the file is not changed. The fields *Destination Folder* and *Rename Mask* are not ready for input. ○ *Rename* With this setting, you can rename the file and if necessary also move it to another folder. You need to specify the destination folder in the *Destination Folder* field. **i Note** If you have entered the character string `*.*` in the *Rename Mask* field, the folder to be monitored and the destination folder can be the same. ○ *Move* With this setting, you can move the file to another folder. You need to specify the destination folder in the *Destination Folder* field. You can use the *Browse* pushbutton to search for the folder in the directories. **⚠ Caution** The folder to be monitored and the destination folder cannot be the same. The *Rename Mask* field is not ready for input with this setting. ○ *Delete* With this setting, the file that is found is deleted. The fields *Destination Folder* and *Rename Mask* are not ready for input. ○ *Copy* You can use this setting to copy the file into another folder that you specify in the *Destination Folder* field.
Destination Folder	Specifies the destination folder to which the file is to be moved. (This is only relevant for the actions *Move* and *Rename*.)

Field	Description
Rename Mask	Here you can specify the new name of the file after it has been renamed. You can rename the files using the following options: ○ You can enter date and time elements that are then integrated into the changed file name. For example, you enter `[LO]-[YYYY]-[MM]-[DD].[RO]`, which means that the current date is inserted into the file name. See Rename File [page 251]. ○ You can extract the original file name and place it to the left or right side of the "." (period). ○ You can extend the file name using a prefix or a suffix. ○ You can keep the original file name extension (the file type) or replace it with another one. ○ You can extract textual or numeric parts of the file name and use them for the new file name. ○ For more information, see Rename File [page 251].

4. Save your entries.

4.1.13.2 Authentication Tab

Context

With the entries on the *Authentication* tab, you ensure that the source system has access to the directories to be checked that are stored on other PCs in the network.

Procedure

1. Click the *Authentication* tab.
2. Enter data with the help of the following table:

Field	Description
Authentication Required	You use this checkbox to define whether authentication is necessary. If you select the checkbox, the fields *Domain*, *User Name*, and *Password* are ready for input.
Domain	You enter your Windows domain here.
User Name	You enter the user name for the PC in the network on which the folder to be checked is stored.
Password	You enter the password.

3. You can use the *Test* pushbutton to check whether the user name and password are correct.
4. Save your entries.

4.1.13.3 Upload Binary File Content

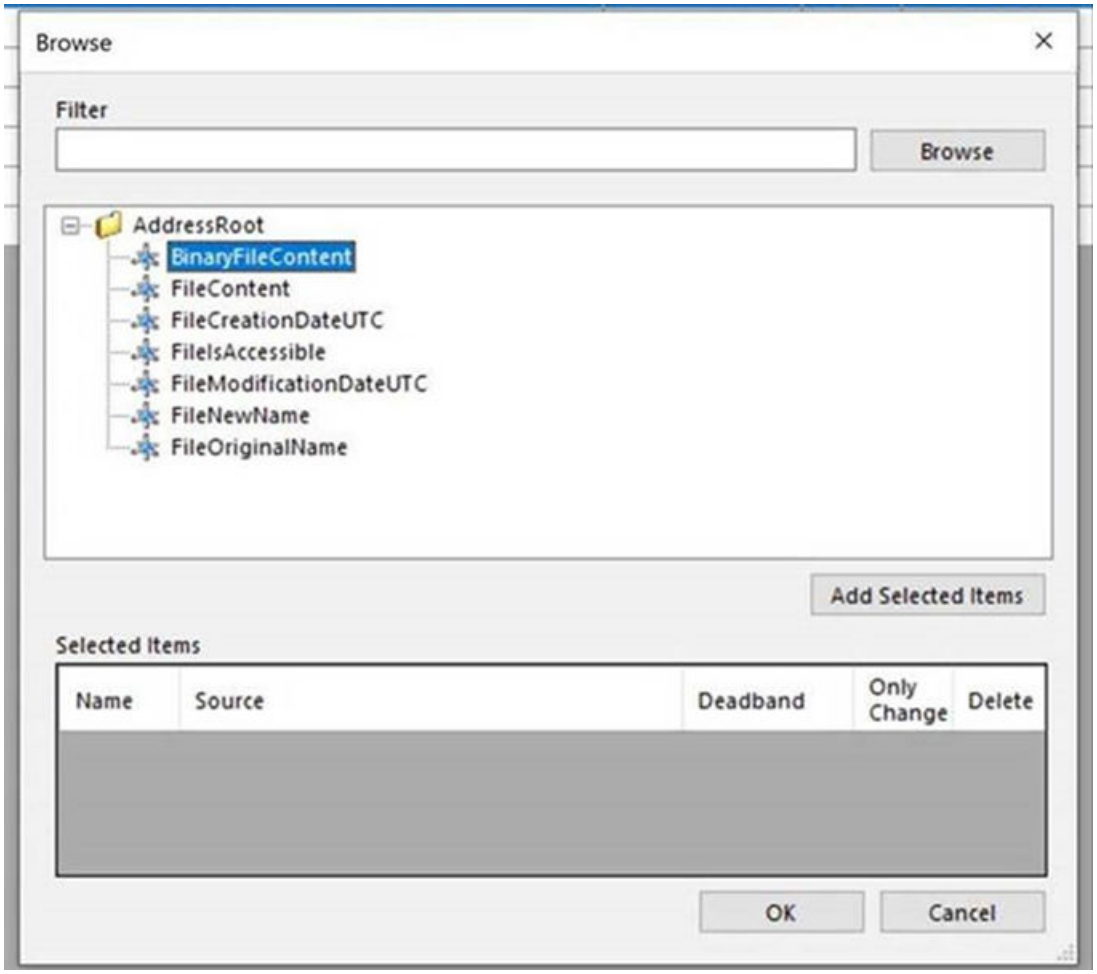
You use the file system source system and the corresponding agent instance to be able to send binary file content to a destination system in a notification process. You use, for example, the universal Web service destination system as the destination system. You can then use the universal Web service destination system to upload the file content (see also: Uploading a File (MIME Format Binary) [page 323]).

To do this, you must proceed as follows:

1. Create a file system source system and select the data type *Binary*. (See: Settings Tab [page 256].)
2. Create the corresponding agent instance.
3. Choose the browse function on the *Subscription Items* tab. Select the **BinaryFileContent** tag and, if necessary, other tags, and subscribe to them.
   The advantage of the **BinaryFileContent** tag is that the file contents are available as a byte array during processing and can be sent to the destination system without any further transformation.
4. Create a universal Web service destination system and maintain the settings for uploading files in binary format.

Example

The screenshot shows the browse dialog for subscribing to the tags.



4.1.14 Socket Source System

Context

With the socket source system you can provide a connection to a **TCP/IP-based socket server** that manages data, for example, in the warehouse or in production. This data is then transferred in the form of data streams first to PCo and then to a destination system, for example, to an SAP EWM system. The data streams are divided into individual telegrams using a user-defined delimiter or a message length you have defined. A telegram corresponds to a notification message.

You can connect the following types of destination systems:

- RFC destination (SAP EWM, SAP ERP)
- MII destination (SAP MII)
- Web Service Destination (SAP ME)

If you have connected your data source to the destination system using the socket source system, the destination system can receive notifications in the form of telegrams.

The following tabs are available for configuring a socket source system:

- **Socket** tab (see below)
- Source System: Aliases Tab [page 138]
  Source System: Reliable Connection Tab [page 142]

You can use the socket source system for the notification process and for the query process.

Procedure

1. To create a socket source system, in the menu choose *Plant Connectivity* > *New* > *Source System*. In the dialog window, choose the source system type *socket source system* and enter a name for the source system. Click OK.

   The *Socket* tab appears. On this tab, you can manage one or more TCP/IP connections and keep them open.
2. On the *Socket* tab, you can make the following settings:

Field	Description
Remove Terminator When Receiving Data	The socket source system receives an endless data stream from the connected data source (server). That is why you need a separator, such as #, to separate the individual messages in the data stream. The separator is also called a terminator. If you select the checkbox, the terminator that denotes the end of a message is removed from each message.
Socket Type	Specifies the version of the Internet protocol. You have the following options: ○ IPv4: Internet protocol version 4 ○ IPv6: Internet protocol version 6 Once you have selected one of these options, PCo automatically creates another row in the table for an additional socket connection.

Field	Description
Check Connection	If the checkbox is selected, the connection to the port is tested periodically in accordance with the settings on the *Reliable Connection* tab. See also: Source System: Reliable Connection Tab [page 142] **i Note** If you connect an EWM system to PCo, there is no connection check.
Name	Name of your choice
Address	IP address of the socket (server that is used as the data source) **⚠ Caution** In conjunction with an EWM system, you must not make any entries here. The EWM system generates a communication channel automatically between the socket server and PCo and stops it again.
Port	Port number of the socket **⚠ Caution** In conjunction with an EWM system, you must not make any entries here.
Terminator	Separator that is used to separate the individual messages within the data stream from each other. Enter **#**, for example.
Length	Length of the individual message within the data stream **i Note** You can specify either a terminator or the length of the message.

Next Steps

For more information about configuring PCo in conjunction with SAP EWM, see SAP Note 2120484.

4.1.15 Statistics Functions for Source Systems

Use

The following Historian data sources provide standard statistics functions for PCo source systems:

- PI source system
- Proficy source system
- IP21 source system
- Citect source system

Features

The following statistics functions are available:

- **Minimum**: Minimum value of sub-interval

  > ℹ Note
  >
  > The length of the sub-intervals is determined by the quotients of the selection interval and the interval count parameter or interval count parameters. The connected equidistant sub-intervals provide the selection interval.

- **Maximum**: Maximum value of sub-interval
- **Area**: Difference between the maximum and minimum value for the sub-interval
- **Total**: Total of the values in the sub-interval
- **Arithmetic average value**

$$Durchschnitt = \frac{\sum_{i=1}^{n} f(t_i)}{n}$$

- **Integral (time-weighted total)**

$$integral = \sum_{i=1}^{n} f(t_i) * \Delta t_i$$

- **Time-weighted (tw) average**

$$Durchschnitt_{tw} = \frac{integral}{\sum_{i=1}^{n} \Delta t_i}$$

- **Standard deviation**

$$Standardabweichung = \sqrt{\frac{1}{n} * \sum_{i=1}^{n} (f(t_i) - Durchschnitt)^2}$$

4.1.16 Filter Functions when Browsing for Tags

Use

You can use a filter to restrict the number of hits in searches for tags in the namespace of the data source. PCo provides two types of filters:

- *Native Filter* of the data source
- *Regex Filter* (regular expression)

Integration

- All agents except the OPC UA agent can use the *Native Filter*.
- All agents can use the *Regex Filter*.

Features

Native Filter

The *Native Filter* is a source-specific filter. It is passed on directly to the data source. The data source filters the data and returns only those tags that meet the filter criteria. For this reason, the *Native Filter* is very advantageous in terms of performance. However, there is no uniform implementation of the *Native Filter* by the

data sources. To understand the syntax of the filter, you need to read the documentation for the data server. Two *Native Filters* are described as examples:

- OPC DA server from KEPWare
  The filter enables you to search for a search term that occurs in the tag name.
  It supports wild card searches: * (any character string), ? (any single character), # (any single numerical character)
- PI server
  The PI server provides an SQL-type filter logic with the following options:
  - The filter of the PI server can search for metadata of tags, meaning that you can search for tags using the instance of attributes.

    > **Example**
    >
    > `tag = 'TagName'` or `description = 'DescriptionText'` or `pointtype = 12`

  - The filter allows arithmetical comparisons such as `pointtype > 12`.
  - The filter allows the logical chaining of expressions.

    > **Example**
    >
    > `tag = 'TagName' and pointtype = 12`

  - The filter supports wildcards for strings, such as `tag = 'sin*'`. The exact filter logic and its possibilities are described in the documentation for the PI server.

Regex Filter

As a rule, the Regex filter cannot be evaluated by the data source, rather only in PCo using the functions provided by `.NET` for the regular expressions. In order for this filter to be used, the tags must already be loaded in PCo.

The advantage of this filter is that it follows a uniform syntax, namely that of the regular expressions. The syntax of the regular expressions under .NET can be found in the Microsoft documentation.

The disadvantage of this filter is that all the data from the data source must be read into the PCo system before the filter can be used for the data.

Uses of the Filter

You can use the filter on the following tabs in the PCo system:

- *Subscription Items* tab
  The *Subscription Items* tab is at the agent instance. It is used only in the notification process [page 35] to select the tag or tags. When browsing to select the subscription items, the **native filter** is used (with the exception of OPC UA source systems). The PCo system displays the *Filter* field in the *Browse* dialog box. You can enter the filter criterion or criteria here.
  PCo then displays only the tags that meet the filter criteria. See also: Agent Instance: Subscription Items Tab [page 476].

  > **Note**
  >
  > The filter is not used on groups. This means that the groups appear even if they do not contain any tags that meet the filter criteria.

- *Aliases* tab

The *Aliases* tab is in the source systems. This tab is used only in the query process [page 43]. The **native filter** is used for browsing (with the exception of OPC UA source systems). You can restrict the selection of tags by entering a filter criterion in the *Mask* field.
PCo displays only the tags that satisfy the filter criteria. See also: Source System: Aliases Tab [page 138].

> **i** Note
>
> The filter is not used on groups. This means that the groups appear even if they do not contain any tags that meet the filter criteria.

- *Tag Query* tab
  The *Tag Query* tab is at the agent instances. This tab is used only in the query process [page 43]. See also: Agent Instance: Tag Query Tab [page 475].
  The native filter is used in connection with the *Cache Mode* field (with the exception of OPC UA source systems).
  If you enter **`Cache`** in the Cache Mode field and the value **`Tag00*`** as the filter criterion in the *Mask* field, for example, the **native filter** is used when the agent is started. All tags that meet the filter criterion are then loaded into the PCo cache.
  You can use the *Mask* screen for the *Cache Mode* **`Cache`** only. When the agent instance is started, all tags that meet the filter criteria you specified in the *Mask* field are loaded into the PCo cache.
  If you have set the *Cache Mode* to **`Alias`**, only the tags of the aliases selected in the *Alias* field are loaded into the PCo cache.
  In the case of the cache modes **`Cache`** and **`Alias`**, static queries to PCo, such as queries pertaining to existing tags of applications (such as SAP MII), are no longer forwarded to the data source, they are handled directly via the PCo cache instead. This concept enables the namespace to be restricted and the destination system (for example, SAP MII) to access the tag information more quickly.
  If you have set the *Cache Mode* to **`None`** or **`Demand`**, you cannot make any specifications regarding filter criteria since in these cases no tags are loaded into the PCo cache when the agent instance is started.

4.1.17 Functions for Source Systems

Add Source System

1. In the *Plant Connectivity Management Console*, choose *Plant Connectivity* > *New* > *Source System* in the menu or choose the *Add Source System* pushbutton.
2. Select a source system type, enter the required data, and choose *OK*.

Duplicate Source System

1. In the *Plant Connectivity Management Console*, select the source system you want and choose the *Duplicate Source System* pushbutton.
   The system creates a new source system with the same settings. The name of the original source system is copied and supplemented by *(1)*.
2. You can change the settings of the new source system.

Copying and Pasting a Source System

1. Select the desired source system and choose *Edit* > *Copy* from the menu.
2. Choose *Edit* > *Paste*.
   The system creates a new source system with the same settings.

Delete Source System

> **i** Note
>
> You cannot delete a source system if an agent instance is configured to use it.

1. On the *Plant Connectivity Management Console* screen, select a source system and choose *Edit* > *Delete*. Alternatively, you can choose the *Delete Source System* pushbutton.
2. Choose *Yes*.

Rename Source System

1. In the *Plant Connectivity Management Console*, select a source system and choose *Edit* > *Rename*.
2. Enter the required data and choose *OK*.

Changing Description of a Source System

After creating a source system, you can use this function to change the description of the object at any time by clicking on the source system and choosing *Edit* > *Change Description* in the menu. You can only display the description of a source system by placing the mouse pointer on this object. PCo then displays the description that you entered when you created the system.

Show Usage in Agent Instances

With this function, you can display, for a source system, the agent instance in which the source system is used. Click the source system you want and choose the *Show Usage in Agent Instances* pushbutton. PCo then displays a dialog box with the where-used list.

Context Menu

You can call a context menu for each source system by clicking on the source system with the right mouse button.

4.2 Destination System

Definition

A destination system is the system or the server to which notification messages are to be sent. You can use the maintenance tab of the destination system to configure the data flow from PCo to the connected destination system. You only need to create and configure destination systems if you want to use the notification process [page 35].

You can assign multiple destination systems to a single agent instance. This means that a single action that is triggered by a tag value change on a tag-based server, or a method call via PCo OPC UA and a Web server, can lead to more than one destination system being called.

The following SAP Business Suite systems can be connected to PCo as a destination system:

- SAP MII
- SAP ME
- SAP EWM
- Other SAP Business Suite systems with *SAP NetWeaver*, such as SAP ERP or S/4 HANA.
  The NetWeaver systems with the software component SAP_BASIS of the following releases are supported:
  - Release 7.00, as of support package SAPKB70208
  - Release 7.01, as of support package SAPKB70109
  - Release 7.02, as of support package SAPKB70024
  - Release 7.31
  - Release 7.40
  - Release 7.50
  - Release 7.5x
- Databases, for example, the SAP HANA database
- MQTT server
- Calling Web services

If the available destination system types are not sufficient for you, you can create additional destination system types as part of a customer-specific development. SAP Note 2470588 describes how you can develop a customer-specific destination system based on the template `CustomDestination` provided by SAP.

Configuration of Destination Systems

Destination Systems

Destination System	Description
Universal Web Service Destination System	The universal Web service destination system is a destination system type with which you can send notification messages to a Web service. The following types of Web services are possible: • Web service • RESTful Web service • OData service See also: Universal Web Service Destination System [page 273].
RFC Destination System	If you want to define a **Business Suite system** (for example, SAP ERP, SAP SCM, SAP EWM) as a destination system, you need to create a destination system of the type *RFC destination system*. See also: RFC Client Settings Tab [page 348].
OPC UA Destination System	You can use this destination system type to enable the call of a method of an OPC UA server. This OPC UA server may be configured on a remote system or on an agent instance of your own PCo installation. See also: OPC UA Destination System [page 333]
Multiple Call Destination System	The multiple call destination system is a destination system type that you can use to call other destination systems in a configurable sequence from a notification. This destination system type also provides functions for calculations and conversions of variables that are returned from a destination system or that are used to call a destination system. Multiple call destination systems also allow the modularization of functions, similar to functions or static methods in programming languages. You can bundle frequently used sequences of destination system calls in a multiple call destination system and reuse this multiple call destination system in various places, such as in other multiple call destination systems. See also: Multiple Call Destination System [page 353]

Destination System	Description
Query Destination System	The query destination system allows you to read tag values from the source system of an agent instance or to write tag values to the source system of an agent instance. The agent instance can be the same agent instance from which the query destination system is called or another agent instance that is running on the same PCo computer. See also: Query Destination System [page 375]
MQTT Destination System	You use the MQTT destination system to be able to send MQTT messages to an MQTT server. This allows you to forward data, which has been transferred to PCo, to an MQTT server. See also: MQTT Destination System [page 382].
ODBC Destination System	You use the *ODBC destination system* to be able to send notification messages to an ODBC database. With each notification message, the tag values and metadata of the selected tags are stored as table entries in a table (for example, a history table) in the connected database. You can use the values stored in the database table for evaluations later. See also: ODBC Destination System [page 338]
Simulation Destination System	The simulation destination system is a folder in a directory you have specified on your computer. You can use this destination system type to test the sending of notification messages very quickly. See also: Simulation Destination System: Configuration Tab [page 389].

Deprecated Destination Systems and Recommended Alternatives

The following destination systems are still part of the PCo solution for reasons of compatibility with earlier PCo releases. You can continue to use existing configurations. However, SAP does not recommend creating new instances of the following destination systems, since there are alternatives that continue to be developed by SAP from a functional point of view, but also with regard to security standards. If possible, use one of the available alternatives.

Deprecated Destination Systems and Their Alternatives

Destination System	Description	Alternatives
MII Destination System (Deprecated)	Until now, if you wanted to connect an **SAP MII system** to be able, for example, to call a transaction there, you could create a destination system of the type *MII destination system*. See also: MII Destination System [page 391]	SAP recommends that, for new configurations, you use a **universal Web service destination system** to call MII transactions. For more information about calling MII transactions using Web services, see Calling MII Transactions Using Web Services [page 38].
Web Service Destination System (Deprecated)	Until now, if you wanted to connect to a third-party system, for example, an **SAP ME system**, you could create a destination system of the type *Web service destination system*. See also: Web Service Destination System [page 393]	SAP recommends that you use a **universal Web service destination system** for new configurations.

4.2.1 Universal Web Service Destination System

Use

You use the universal Web service destination system to be able to send notification messages to the following types of Web services:

- Web services (WSDL)
- OData services
- RESTful Web services

The universal Web service destination system also supports arrays. For more information about maintaining arrays, see SAP Note 2601203.

You can also use the universal Web service destination system in enhanced notification processing if you want the destination system's response to be processed. (See also: Integration with Third-Party Systems Using Enhanced Notification Processing (ENP) [page 68].)

For more information, see SAP Note 2550579.

See Also

Web Service Settings Tab [page 274]

Operation Configuration Tab [page 286]

Security Settings Tab [page 279]

Proxy Settings Tab [page 327]

4.2.1.1 Web Service Settings Tab

Context

On this tab, you specify the connection settings for the service to which you want to send notification messages. (See also: Universal Web Service Destination System [page 273].)

Procedure

1. Choose the *Add Destination System* pushbutton. Then select the destination system type *Universal Web Service Destination System* and enter the name of the destination system.

   The *Web Service Settings* tab appears.
2. Enter the following data:

Field	Description
Description Type	You specify here which type of service you want to configure. You can choose one of the following options: ○ *No Service Description* Choose this option if you want to configure a RESTful Web service. ○ *WSDL* Choose this option if you want to configure a Web service. ○ *OData* Choose this option if you want to configure an OData service.

Field	Description
Description (URI)	Here you enter the URI of the service that you want. This is **only required for Web services and OData services**. You do not need to enter anything here for RESTful Web services. **Example** Enter the following example OData service, for example: `http://services.odata.org/V3/OData/OData.svc` You can use a file instead of the URI. If you choose *Load Service Description from File*, you can load the service locally from a file. In this case, the path for the selected file is displayed automatically.
Endpoint URI	Specifies the endpoint URI for a service operation. The URI is displayed automatically as soon as you have selected an operation of the WSDL service or OData service in the explorer (see step 4). This field can only be changed for RESTful services. For RESTful services, you can enter an endpoint URI as a constant value or as a template here. (See also: Configuring the Endpoint URI for RESTful Services [page 276].)
Timeout	Indicates the response time in milliseconds that must not be exceeded. If it is exceeded, the connection to the Web service is terminated.
Detailed HTTP Logging	Detailed HTTP logging is used for troubleshooting. If you select this checkbox, all HTTP requests and responses with the status `Raw` are logged in a file in the `Logs/UniWsDestination` directory. The log directory is on the same level as the PCo system directory. The log directories are stored with the name `<Agent Instance Name>.log`. `<Agent Instance Name>` is the name of the agent instance. The log contains up to five files for the same agent instance with a maximum file size of 4 MB. As soon as all files are filled up, the oldest file is deleted.

3. Choose *Open in Service Explorer* to display the service operations provided by the service.

Caution

If the explorer does not display any data but rather returns an error message, you still need to make the proxy setting. (See also: Proxy Settings Tab [page 327].)

The system displays the various service operations in the *Service Explorer* screen area.

The service explorer offers a tree view. The data model of the service is displayed in the form of a tree structure.

For a **WSDL Web service**, the root element is a specific service. The ports are one level below this. The individual service operations are at the lowest level.

In the case of the **OData service**, the following two types of tree view nodes are offered:

- Entity
- Operations of an entity

4. Click on the entity or operation that you want. If you click on the entity node, the *Read all* operation is selected automatically for an OData service.

   The selected operation is displayed at the bottom in the *Operation* field. At the same time, the URI of the selected endpoint is displayed in the *Endpoint URI* field.

5. If the connection to the server is temporarily unavailable, or you have persisted a file, you can also load the data model of the service from a file by choosing the *Load Service Description from File* pushbutton. You can search for the corresponding file. (See also: Load Service Description from File [page 278].)
6. By choosing *Save Service Description to File*, you can save the data model of the selected service locally so that you can also call up the service locally (as described under step 5).
7. Choose ENTER.

   The *Operation Configuration* tab appears.

Next Steps

Example: Create a Universal Web Service Destination System (OData) [page 329]

4.2.1.1.1 Configuring the Endpoint URI for RESTful Services

For a RESTful service, you can enter an endpoint URI here as a constant value or as a template.

The example below describes how you can change the URL dynamically for a universal Web service destination system. This is always necessary if multiple RESTful Web services have the same interface, but have different URLs. Normally all of these services run on a Web server and a part of the URL is the same for all. In this case, you only need to create one universal Web service destination system (RESTful).

Example

You want to determine the endpoint URI based on the value of the input variable `deviceId`. In this case, you need to enter the following:

```
https://myhost.com/myService/{deviceId}/dataAcquisition/temperature1
```

If the `deviceId` has the value MACHINE_01, the endpoint will be named later as follows:

```
https://myhost.com/myService/MACHINE_01/dataAcquisition/temperature1
```

You can specify as many input variables as templates as you want. You can even specify the entire endpoint URI as a variable, for example, `{myEndpointUri}`. In this way, the system can determine the endpoint URI of the Web service during runtime. For more information about the rules for defining templates, see Template Syntax [page 277].

4.2.1.1.1.1 Template Syntax

Use

Templates are used to determine the content of a character string at runtime, based on the values of the input variables used.

You can create templates for the following objects:

- MQTT source system
  You can create a template on *the Tag Definition* tab by choosing the button *Configure Message from Template* and entering a character string in the dialog box that then appears. This is only possible in conjunction with the **payload types** `JSON` and `Formatted String`.
  When you make your entries in the dialog box, the parameters in the *Message Configuration* table and the *Tag List* are updated automatically in the MQTT source system. The four buttons that are used for entering and removing tags (below the table) are inactive when you enter the tags in the dialog box.
- Universal Web service destination system
  You can create a template at the following places:
  - On the *Templates* tab
  - By choosing the button *Configure Message from Template* on the *Operation Configuration* tab. You use this button to automate the configuration of a request message for a RESTful service if you use the JSON payload type.
  - In the endpoint URI of a RESTful service
  - On the *Advanced Configuration* tab (on the Operation Configuration tab). You can enter header names and values in the *Headers* screen area. These header values are then adopted on the Request Message Configuration tab as input variables.

Structure of a Template

Each template contains a sequence of literal values and placeholders. The placeholders are displayed with curly brackets and have the following syntax:

```
{<variable name> [:<variable type>]}.
```

The variable name must be valid, that is, it must start with a letter and must only contain numbers and letters. The variable type is optional and can be omitted. In this case, the assumption is that the variable type is a `String`. You can specify each of the primitive data types allowed for Web services. You can specify each of the

primitive data types allowed for Web services. You can find a list of permitted data types under Add a Request or Response Mapping [page 289].

> **Example**
>
> We use the following template:
>
> `/weather/{state}/{city}?forecast={days:int}`
>
> If the corresponding variables for `state`, `city`, `days` have the values "`BW`", "`Mannheim`", and "`3`", the content of the character string is evaluated as follows:
>
> `/weather/BW/Mannheim?forecast=3`

If you want to add curly brackets in the payload, you need to use escape characters: `'{'` is an escape character for '{' and `'}'` is an escape character for '}'.

> **Example**
>
> The template `{{justText}}` corresponds to the payload `{justText}`. No input variables are added in this example.
>
> In contrast, the template `{{{myText}}}` determines the input variable `myText`, and `{text}` is specified as payload if the input variable has the value "`text`".

4.2.1.1.2 Load Service Description from File

You choose this pushbutton to load a service description from a file. This makes sense if you have already persisted the service description or if you want to import the configuration of a RESTful service from the Postman application.

When you choose this pushbutton, the *Load Service Description from File* dialog box appears. You can select the file type here:

- *WSDL Metadata Files (*.wsdl)*
- *ODATA Metadata Files (*.odata)*
- *Postman Collection Metadata Files (*.json)*
  Since the file format for the Postman Collection metadata allows more than one configuration to be persisted, you can select the configuration that you want to import in the dialog box that then appears.

4.2.1.2 Security Settings Tab

You define authentication, certificates, and validation options for certificates here.

Authentication

In this section, you define what kind of authentication you want, that is, how you want PCo to identify itself to its communication partner. You can choose one of the following options for authentication:

- *No Authentication*
  No logon data is required for this setting.
- *Basic Authentication*
  In this case, you have to enter a *user name and password* that PCo uses to log on to the destination system. You can also select the *Send Credentials with Each Request* checkbox if you want PCo to send the user name and password with each request.
- *Client Certificate Authentication*
  In this case, you need to select a client certificate with a private key on the *Certificate* tab. PCo uses this certificate to identify itself to its communication partner.
  This form of authentication is only possible in conjunction with a secure https connection.
- *OAuth Authentication*
  OAuth 2.0 is an open protocol for authentication. You can use the OAuth settings to set up secure authentication and authorization between PCo and the **SAP Digital Manufacturing Cloud**. Enter the URI of the OAuth service. To do so, choose *Edit OAuth Settings*. (See also: Configuration of OAuth Settings [page 283].)

If you have selected a **certificate** for authentication, you can determine how the certificate of the universal Web service destination system is to be identified during the runtime of the Plant Connectivity system by defining the *identification type* for the certificate. If you select the *Identification by Subject* option, certificate rotation is supported. (See also: Identification Type of Certificates [page 593].)

Certificate Folders

This screen area is only ready for input if you have set a secure **https connection** on the *Web Service Settings* tab. The settings under certificate folders and certificate validation options are used for the **server certificate**. The server certificate is the certificate with which the communication partner identifies itself to PCo.

Settings for the Certificate Folders

Field	Description
Store Type	Here you select the store for the certificate that you want to be validated. The following types are supported: • *Microsoft certificate store* When a connection is being established, with this setting, PCo automatically searches in the Microsoft certificate store folder for a server certificate. **→ Recommendation** If you are using PCo based on Windows OS, you need to use the *Microsoft certificate store* option. • *File system certificate store* With this option, you can specify the store location for the certificates, which PCo is to trust, in the file system.
Trusted Certificates	Here you can specify the folder in which the trusted certificates are stored. If you have selected the *Microsoft certificate store*, this is the folder for the *trusted root certification authorities*. The system proposes this automatically. **Example** `Local Computer/Trusted Publishers` If you have chosen the *file system certificate store*, a directory is proposed in the file system with the following subfolders: • *certs*: Folder for trusted certificates If you store a certificate from the certificate chain in this folder, this certificate is trusted in the check. The certificate is regarded as not trusted if there is no certificate from the certificate chain stored here. • *crls*: Folder for certificate revocation lists • *private*: Folder for private certificates If you choose *Browse*, a dialog box appears where you can select another folder.

Field	Description
Issuer Certificates	Here you can specify the folder in which the certificates of a trusted issuer are stored. If you selected the *Microsoft certificate store*, this is the folder for the intermediate certificate authorities. This is proposed automatically. **Example** `Local Computer/Intermediate Certificate Authorities` If you have selected the *file system certificate store*, a directory is proposed in the file system with the subfolder *certs*. This folder is used to complete the certificate chain if the server does not send the complete certificate chain.
Rejected Certificates	Here you can specify the folder in which the rejected certificates are stored. If you are using the *Microsoft certificate store*, select *Untrusted Certificates* here. **Example** `Local Computer/Untrusted Certificates` If you have selected the *file system certificate store*, use a directory in the file system with the subfolder *certs* (folder for rejected certificates).

Certificate Validation Options

This screen area is only ready for input if you have set a secure **https connection** on the *Web Service Settings* tab and have set authentication using client certificates.

Validation Options

Field	Description
Revocation Check	In this field you define how the revocation check of the server certificate is to be performed. You have the following options: • *No Check on Revoked Certificates* No check is carried out. • *Check Online Revocation Lists* The online check is a secure procedure but it can have a negative impact on performance. → Recommendation This is the recommended setting in connection with the Microsoft certificate store. • *Check Offline Revocation Lists* ○ If you are using the **Microsoft certificate store**, you need to copy all the relevant certificate revocation lists into the **Trusted Root Certification Authorities** directory. ○ If you have selected the **file system certificate store**, you need to copy all related certificate revocation lists as `.crl` files into the revocation list folder.
Revocation Check Scope	Indicates the scope of the revocation check. You have the following options: • *Check End Certificate Only* Only the last certificate in a certificate chain is checked. • *Exclude Root Certificate from Check* • *Check Entire Chain* All certificates in a certificate chain are checked. → Recommendation This is the recommended setting.

Field	Description
Ignore Server Host Name	If you select this checkbox, the check results of the server host name are not taken into consideration. During the check, a comparison is made with the domain name system (DNS) name that is included in the certificate. The DNS name is the name of a server in a domain, for example: `server.domain.com`. → Recommendation SAP recommends that you do not set this indicator.
Ignore Validity Period	If you select this checkbox, you define that the validity period of the client certificate and the certificates in the certificate chain are not to be taken into consideration.

4.2.1.2.1 Configuration of OAuth Settings

You can make the OAuth settings either directly here in PCo or in the Machine Model in the **SAP Digital Manufacturing Cloud**. The data is then transferred automatically from the cloud to the *PCo Management Console* and displayed here.

1. Check the settings that have been transferred from the **SAP Digital Manufacturing Cloud** or enter them here:

   OAuth Settings

Field	Description
Client ID	The client ID is a public identifier for the Web application, for example, a Web service, that is provided by the **SAP Digital Manufacturing Cloud** and can be executed by PCo. This ID must be unique for all clients that are managed by an authorization server.
Client Secret	The client secret is only known to the Web application and the authorization server.

Field	Description
URI	The endpoint of the **user authentication and authorization service** that is hosted by the authorization server (in the cloud). The service is responsible for checking the client ID presented by PCo, the client secret, the scope, and the SAML metadata sent by PCo. If the check is successful, the authorization server issues an access token. This access token (JavaScript Web Token) is added to the authorization header in each request that is sent to the Web application or a cloud service.
Issuer	Specifies the SAML issuer that issues the SAML assertions with which you can logon to a system that trusts the issuer. Enter, for example, **`PCoIDP`** here. The SAML issuer that provides the information about the identity provider is entered in this field. An identity provider is a trustworthy system that authenticates the user to Web applications (cloud) or digital resources.
Certificate	You select a certificate here that contains a private key. The certificate can also be self-signed. The self-signed certificate needs to be stored in the certificate store of the computer on which PCo is installed. Using encryption, the cloud can reliably ascertain the origin of PCo requests with the public key of this certificate. PCo uses this certificate in its role as identity provider. A SAML metadata descriptor can be generated based on the issuer and this certificate. You can generate the SAML metadata later by choosing the pushbutton *Generate SAML Metadata*.
Identification Type	You use the identification type to determine how the certificate is to be identified at runtime. If you select *Identification By Thumbprint*, the configured certificate is identified using the unique thumbprint. This is the default setting. If you select *Identification By Subject*, the subject of the certificate is used at runtime to select the appropriate certificate with the longest validity from the list of certificates in the selected storage location. This enables certificate rotation.

Field	Description
Name ID	You enter the user name or the e-mail address of the user here on whose behalf the **SAML assertions** are to be used. **i Note** A SAML assertion is an XML structure that contains properties of the user. SAML assertions are created by the identity provider (PCo) and consumed by the service provider (cloud). SAML assertions and the context information allow the service provider to make decisions about allowing or rejecting access requests of an authenticated user.
Scope	Contains the list of delimited scopes requested for the token. The scope provides a way to limit the number of access rights that are granted for an access token. **Example** An access token that has been issued to a client app may be granted READ and WRITE access to protected resources, or just READ access. You can implement your APIs so that each scope or each combination of scopes can be enforced. This entry is optional.
Audience	Identifies the recipient of the SAML assertion. This might be a Web application or a cloud service for which the SAML assertion token is intended.

2. Choose *Generate SAML Metadata*.

i Note

As described above, to make sure that the SAML metadata is generated correctly, you need to have entered the issuer, for example, **`PCoIDP`** and have selected a self-signed certificate with a private key.

The generated SAML metadata must then be made known to the authorization server in the cloud so that a trustworthy connection to the identity provider (PCo) can be established. You achieve this by uploading the XML file containing the SAML metadata to the authorization server.

4.2.1.3 Operation Configuration Tab

Procedure

1. You can display or change the following data in the upper screen area of the tab:

Field	Description
Endpoint URL	The URL of the service is displayed here. **i Note** For a RESTful Web service, you need to enter the URL of the RESTful Web service that you want here.
Operation	The operation selected previously is displayed here. **i Note** This field is empty for a RESTful Web service because the operation is already part of the URL.

2. For Web and OData services, the variables of the request are displayed in the lower screen area on the *Request Message Configuration* tab. The checkbox for the variables is already selected so no additional entries are required. (See also: Request Message Configuration Tab (RESTful) [page 287] and Request Message Configuration Tab (OData and Web Service) [page 307].)

   i Note

   If you have selected, for example, a `Read_all` operation of an OData service, **no** parameters are offered for the configuration on the *Request Message Configuration* tab; in other words, the table is empty. In this case, you only need to define the response parameters.

   In the case of a RESTful Web service, the request message parameters are not yet displayed. You need to enter these yourself by choosing *Add New Request Mapping*.

3. To configure the parameters for the response message, choose the *Response Message Configuration* tab.

   The response variables are displayed automatically. You can check them and supplement them. In the case of a RESTful Web service, the request message parameters are not yet displayed. You need to enter these yourself by choosing *Add New Response Mapping*.

   (See also: Response Message Configuration Tab (OData and WS) [page 313] and Response Message Configuration Tab (RESTful) [page 310].)

4. On the *Request Message Configuration* and *Response Message Configuration* tabs, you can import structured data types as well as array and enumeration data types from SOAP Web services and OData services into the PCo data type repository. Choose the *Import Data Types into Data Type Repository* button to start the import. (See also: Import Data Types into the Data Type Repository [page 309].)

As soon as the new data types of the Web service are created in the PCo data type repository, you can define input and output variables at root node level of the Web service data structure.

The advantage of this is that you only have to enter a single input variable (request configuration) or output variable (response configuration). In this case, you only have to enter a single input or output variable respectively in order to transfer complex and deeply nested data objects.

If you choose *Select Data Type*, you are taken from the request or response message configuration to the PCo data type repository, where you can select the relevant data type that you imported previously. (See also: Select Data Type (Selection Dialog) [page 588].)

5. All technical data for the selected operation is displayed on the *Advanced Configuration* tab. (See also: Advanced Configuration Tab [page 316].)
6. To be able to test the configuration of the service operation, choose  . (See also: Test Configuration [page 326].)

Next Steps

Example: Create a Universal Web Service Destination System (OData) [page 329]

4.2.1.3.1 Request Message Configuration Tab (RESTful)

Prerequisites

You have entered the URL for the RESTful Web service in the *Endpoint URL* field.

Context

This document describes the manual procedure for the **request message configuration for a RESTful Web service**. For the RESTful Web service, the table with the request parameters is still empty at first. To create the parameters, choose  . Alternatively, you can also call this function from the context menu of the table. (See also: Context Menu [page 303].)

Alternatively, you can enter the parameters using a template. (See also: Configure Message from Template [page 304].)

The parameters can be fixed values or variables. These are passed on to the request as a URL parameter or as JSON in the request body.

The variables that are defined here are mapped later on the *Destinations* tab in *output destination mapping* so that they contain the correct content of the inbound data.

Procedure

1. In the *Method* field, you select the method that you want to use for calling the RESTful Web service. The GET method is the default value.
2. Choose  to create a parameter.

   The *Add Request Mapping* dialog box appears.
3. In the dialog box, you enter the data for a parameter (see also: Add a Request or Response Mapping [page 289] and Examples of Request Message Parameters (RESTful) [page 305]).
4. Enter additional parameters if necessary.

   The *Required Parameter* checkbox indicates that the parameter is mandatory. The checkbox is selected automatically and cannot be changed. The fields *Parameter Name* and *Parameter Type* are grayed out in the request message configuration table because they cannot be changed. The  icon indicates that the field, for example, n1, is a higher-level field (type: **object**) that contains additional fields. The  icon indicates an input variable.
5. The *table of input variables* is displayed to the right of the configuration table. All parameters that you have flagged as input variables are included in the list.
6. If you choose *Select Data Type* for an input variable of the type structured data type, the PCo data type repository appears, where you can select the (previously imported) structured data type. (See also: Select Data Type (Selection Dialog) [page 588].)

   In this case, you only have to enter a single variable in order to transfer complex and deeply nested data objects. This reduces the configuration effort significantly.
7. If you want to change a value, click in the relevant cell and change the value. Choose `Enter` or click on another row to save.
8. If you want to remove a higher-level field (with all the variables) from the table completely, select the row for the field or the row of a variable and choose  .
9. If you select a variable in the table and choose  , the parameter becomes a fixed value again. The variable icon is then hidden.
10. If you choose  , the variable checkbox is selected and the variable name appears in the table.
11. You can use the *Import Data Types into Data Type Repository* pushbutton to import the entire message configuration **as a single structured data type** into the PCo data type repository, so that in future you only have to define a single root variable of the type structured data type. (See also: Import Data Types into the Data Type Repository [page 309].)

Example

Examples of Request Message Parameters (RESTful) [page 305]

4.2.1.3.1.1 Add a Request or Response Mapping

Context

This procedure describes how you can add parameters in the *Add Request Mapping* and *Add Response Mapping* dialog boxes. These might be variables or constants.

Procedure

1. Choose  to create a new parameter. Alternatively, you can also call the function from the context menu. (See also: Context Menu [page 303].)

   Either the *Add Request Mapping* or *Add Response Mapping* dialog box appears.
2. Enter the following in the dialog box:

Field	Description
Parent Parameter	For a **RESTful service**, the root element with the name **$in** is always displayed here for the request mapping or **$out** is displayed for the response mapping. This root element has the parameter type **object**. This name is reserved for bare body style mapping, which is used to serialize request or response parameters without any wrapper elements.

Field	Description
Parameter	Specify the name of the parameter here that you want to be mapped. The parameter you are editing must have a valid path sequence with reference to the parent parameter. **Example** If the parent parameter **$in.TagStoreValues** is used and you enter **`TimeStamp`** as the parameter, the full path is: `$in.TagStoreValues.TimeStamp` **Example** You define the following parameters (type *string*): **`Resource[0]`**, **`Resource[1]`**, **`Resource[2]`**. You then assign these to the constants “r0”, “r1”, and “r2”. The request body then looks like this: `{"Resource": ["r0","r1","r2"]}` If you create the fields (with type string) **`$in[0]`**, **`$in[1]`**, **`$in[2]`** instead and assign them to the same constants, the request body looks like this: `["r0","r1","r2"]` **Arrays** You can also specify an array as a parameter. (See also: Creating Arrays [page 302].) **Note** If you enter an invalid field, a small red symbol with an exclamation mark is displayed in front of the text box for the field.

Field	Description
Type	Select the data type of the parameter. The following types are supported: ○ binary [page 293] ○ base64Binary [page 293] ○ boolean [page 293] ○ byte [page 294] ○ date [page 294] ○ dateTime [page 294] ○ dateTimeOffset [page 295] ○ jsDate [page 296] ○ decimal [page 296] ○ double [page 297] ○ duration [page 298] ○ float [page 298] ○ GUID [page 298] ○ int [page 299] ○ long [page 299] ○ object [page 300] ○ sbyte [page 300] ○ short [page 300] ○ single [page 300] ○ string [page 301] ○ stream [page 301] ○ time [page 302] ○ timeOfDay [page 302] **i Note** The selected data type cannot be changed later.
Variable (for request mapping)	If you select this checkbox for the request mapping, the entry in the *Value* field is not interpreted as a fixed value but as an **input variable** that is filled later accordingly. The system automatically proposes the variable name in the *Value* field. The variable is then automatically included in the table of input variables. You can then map this variable to a subscription item in the notification object. You perform this mapping on the *Destinations* tab for the notification. (Path: *Agent Instance* > *Notification* > *Destinations* .)

Field	Description
Value (for request mapping)	If you have selected the *Variable* checkbox, the variable name is proposed here automatically. All the **request** variables have the prefix *in*. You can change the variable name here or change it later in the table. **i Note** The variable names must not begin with a digit, must not contain any spaces, and must not be C# keywords, for example, if, else, for, while. If it is a **fixed value**, you can enter the value here.
Variable (for response mapping)	The **response** variables are proposed with the prefix **out** and can be changed here and in the table. **i Note** The variable names of the response variables must not begin with a digit, must not contain any spaces, and must not be C# keywords, for example, if, else, for, while.

3. Choose the *Add* pushbutton.

   The parameter appears in the *Request or Response Message Configuration* table. In the table, only the values and the variable indicator can be changed.
4. Enter additional parameters if necessary.

   The fields *Parameter Name* and *Parameter Type* are grayed out in the table because they cannot be changed. The  icon indicates that the field, for example, n1, is a higher-level field that contains additional fields. The  icon indicates an input variable.

4.2.1.3.1.1.1 Allowed Data Types

When you create the parameters for a message request or for a message response, you must select the data type of the parameter. The following data types are supported:

- binary [page 293]
- base64Binary [page 293]
- boolean [page 293]
- byte [page 294]
- date [page 294]
- dateTime [page 294]
- dateTimeOffset [page 295]

- jsDate [page 296]
- decimal [page 296]
- double [page 297]
- duration [page 298]
- float [page 298]
- GUID [page 298]
- int [page 299]
- long [page 299]
- object [page 300]
- sbyte [page 300]
- short [page 300]
- single [page 300]
- string [page 301]
- stream [page 301]
- time [page 302]
- timeOfDay [page 302]

4.2.1.3.1.1.1.1 binary

This data type corresponds to a byte array (byte[]) that PCo puts into a serial sequence as a hexadecimal character string.

Example

CAFEBABE is a hexadecimal character string. This character string is converted by PCo into the following byte array:

[12*16+10, 15*16+14, 11*16+10, 11*16+14]

4.2.1.3.1.1.1.2 base64Binary

This data type corresponds to a byte array (byte[]) that PCo puts into a serial sequence as a base64 character string.

4.2.1.3.1.1.1.3 boolean

This data type specifies a boolean value.

The values are `True` or `False`.

4.2.1.3.1.1.1.4 byte

Represents an unsigned 8-bit integer with a value between 0 and 255.

Below you can see how the various formats and protocols influence the data output.

Settings

MIME Format and Protocol	Result
You have set **JSON** as MIME format and you are using any protocol. (See also: Request and Response Settings [page 318].)	A byte number is issued for the parameter for all protocols.
You have set **XML** and one of the following protocols: • ODATA V2 • ODATA V4 • SOAP	The parameter value is converted into a signed byte number.
You have set **XML** and the **ODATA V3** protocol.	PCo formats the parameter value as a hexadecimal character string.
You have set **UrlEncoded** and **any protocol**.	PCo converts the parameter value into a byte number.

4.2.1.3.1.1.1.5 date

Date

PCo adopts this data type in the format **yyyy-MM-dd**.

4.2.1.3.1.1.1.6 dateTime

This data type represents a point in time (date and time).

The display depends on the selected format and the protocol.

Settings

MIME Format and Protocol	Result
You have set **UrlEncoded** and **no protocol**.	The date and time are converted into the format `yyyy-MM-ddTHH:mm:ss.fff`.

MIME Format and Protocol	Result
UrlEncoded and one of the following protocols: • ODATA V2 • ODATA V3 • ODATA V4	The date and time are converted into the format `datetime'yyyy-MM-ddTHH:mm:ss.fff'`.
You have set **JSON** as MIME format and **no protocol**.	The date and time are converted into the format `yyyy-MM-ddTHH:mm:ss.fff`.
You have set **JSON** and the ODATA V2 protocol.	The date and time are converted into the format `/Date(ticks)/`. The ticks represent the milliseconds that have passed since January 1, 1070.
You have set **JSON** and the **ODATA V3** or **ODATA V4** protocol.	The date and time are converted into the format `datetime'yyyy-MM-ddTHH:mm:ss.fff'`.
You have set **XML** and **no protocol**.	The date and time are converted into the format `yyyy-MM-ddTHH:mm:ss.fff`.
You have set **XML** and one of the following protocols: • ODATA V2 • ODATA V3 • ODATA V4 • SOAP	The date and time are converted into the format `yyyy-MM-ddTHH:mm:ss.fffffff`.
You have set **XML** and the **ODATA V4** protocol.	The date is converted into the format `yyyy-MM-dd`.

4.2.1.3.1.1.1.7 dateTimeOffset

Represents a point in time relative to the Coordinated Universal Time (UTC) that is usually represented by a date and time.

The display of date and time is dependent on the selected format and the selected protocol.

Settings

MIME Format and Protocol	Result
You have set **UrlEncoded** and **no protocol**.	The date and time are converted into the format `yyyy-MM-ddTHH:mm:ss.fff`.
You have set **UrlEncoded** and one of the following protocols: • ODATA V2 • ODATA V3	The date and time are converted into the format `dateimeoffset'yyyy-MM-ddTHH:mm:ss.fffffffZ'`.

MIME Format and Protocol	Result
You have set **UrlEncoded** and **ODATA V4 protocol**.	The date and time are converted into the format `yyyy-MM-ddTHH:mm:ss.fffZ`.
You have set **JSON** and **no protocol**.	The date and time are converted into the format `yyyy-MM-ddTHH:mm:ss.fff`.
You have set **JSON** and the ODATA V2 or ODATA V3 protocol.	The date and time are converted into the format `dateimeoffset'yyyy-MM-ddTHH:mm:ss.fff'`.
You have set **JSON** and the **ODATA V4** protocol.	The date and time are converted into the format `yyyy-MM-ddTHH:mm:ss.fffffffZ`.
You have set **XML** and **no protocol**.	The date and time are converted into the format `yyyy-MM-ddTHH:mm:ss.fff`.
You have set **XML** and one of the following protocols: • ODATA V2 • ODATA V3	The date and time are converted into the format `yyyy-MM-ddTHH:mm:ss.fffffff`.
You have set **XML** and the **ODATA V4** protocol.	The date and time are converted into the format `MM-ddTHH:mm:ss.fffffffZ`.

4.2.1.3.1.1.1.8 jsDate

Converts date and time into ticks.

The date and time are converted here into the format `/Date(ticks)/`. The ticks represent the milliseconds that have passed since January 1, 1070.

4.2.1.3.1.1.1.9 decimal

This data type represents a decimal floating point number.

The display depends on the selected format and the protocol.

MIME Format and Protocol	Result
JSON, no protocol, or ODATA V4 protocol:	Conversion into a decimal number
JSON and ODATA protocols: • ODATA V2 • ODATA V3	Values are converted into a character string that represents a decimal number with the format `[0-9]+.[0-9]+M|m`.

MIME Format and Protocol	Result
UrlEncoded and ODATA protocols: • ODATA V2 • ODATA V3 • ODATA V4	Values are converted into a decimal number with the format `[0-9]+.[0-9]+M|m`.

4.2.1.3.1.1.1.10 double

Stores double-precision floating point numbers

This data type is used to store signed double-precision IEEE-64-bit (8 byte) floating point numbers. With the data type `double`, these numbers are twice as precise as with data type `float`. For that reason, the data type `double` is more suitable than the simple data type `float`.

Settings

MIME Format and Protocol	Result
JSON and no protocol or the ODATA V4 protocol	Representation as a double number with the format `[0-9]+.[0-9]+`
JSON and the following protocols: • ODATA V2 • ODATA V3	A character string that represents a double value with the format `{doubleValue}d .`
UrlEncoded, no protocol	Double number with the format `[0-9]+.[0-9]+`
UrlEncoded and the following ODATA protocols: • ODATA V2 • ODATA V3 • ODATA V4	A character string that represents a double value with the format `{doubleValue}d`.
XML and no protocol or SOAP protocol	Double number with the format `[0-9]+.[0-9]+`
XML and the following ODATA protocols: • ODATA V2 • ODATA V3 • ODATA V4	A character string that represents a double value with the format `{doubleValue}d`.

4.2.1.3.1.1.1.11 duration

Duration

This data type calculates the difference between two points in time of the data type `DateTime` and specifies the length of time.

Settings

MIME Format and Protocol	Result
JSON and all logs:	A character string that represents a duration in the format `P{days}DT{hours}H{minutes}M{seconds}.{milliseconds}S`
XML and all protocols:	A character string that represents a duration in the format `P{days}DT{hours}H{minutes}M{seconds}.{milliseconds}S`
UrlEncoded, all protocols:	Duration is displayed in the format `duration'P{days}DT{hours}H{minutes}M{seconds}.{milliseconds}S'`.

4.2.1.3.1.1.1.12 float

Single-precision floating point number

This data type represents a single-precision floating point number. The float data type is a real number and is displayed with 4 bytes. This is defined by the IEEE 754 standard.

Settings

MIME Format and Protocol	Result
JSON and no protocol or ODATA V4 protocol	Conversion into a real number with single precision
UrlEncoded, all protocols; XML, all protocols	Conversion into a real number with single precision
JSON and ODATA V2 protocol or ODATA V3 protocol	A character string that represents a single-precision real number in the format `{float}f`

4.2.1.3.1.1.1.13 GUID

Globally unique identifier

A globally unique identifier (GUID) is a character string that is used in distributed computer systems.

Settings

MIME format and Protocol	Result
JSON, no protocol, or ODATA V4 protocol	Character string that represents a GUID
JSON and ODATA V2 protocol, ODATA V3 protocol	Character string in the format `guid'{guid}'`
XML, no protocol, or SOAP protocol:	Character string that represents a GUID
UrlEncoded, no protocol	Character string that represents a GUID
UrlEncoded and ODATA V2, ODATA V3, ODATA V4:	Character string in the format `guid'{guid}'`

4.2.1.3.1.1.1.14 int

Integer

Integer is a data type that stores integer values.

4.2.1.3.1.1.1.15 long

Stores large integer values

You use the long data type to store integer values that are too large for the integer data type.

Settings

MIME Format and Protocol	Result
JSON and no protocol	Long integer value
JSON and ODATA V4 protocol	Long integer value
JSON and ODATA V2 protocol or V3 protocol	A character string that represents a long integer value in the format "`{longInteger}L`".
XML, no protocol	Long integer value
XML and ODATA V3 protocol or V4 protocol, and SOAP	Long integer value
XML, ODATA V2 protocol	A character string that represents a long integer value in the format `{longInteger}L`
UrlEncoded, no protocol	Long integer value
UrlEncoded, OData V2 protocol, ODATA V3 protocol, or ODATA V4 protocol	A character string that represents a long integer value in the format `{longInteger}L`

4.2.1.3.1.1.1.16 object

Freely defined structure

Object is a freely defined structure.

Example

A structure that describes a person:

{

Name: 'Martin',

Family: 'Smith',

Age: 50,

Department: 'PI'

}

4.2.1.3.1.1.1.17 sbyte

Signed byte

4.2.1.3.1.1.1.18 short

Short integer value

4.2.1.3.1.1.1.19 single

Stores single-precision floating point numbers

You can use this data type to store signed IEEE 32-bit (4 byte) single-precision floating point numbers.

Settings

MIME format and Protocol	Result
JSON, no protocol	Real number with single precision
JSON, ODATA V4 protocol	Real number with single precision

MIME format and Protocol	Result
JSON, ODATA V2 protocol, or OData V3 protocol	A character string that represents a single-precision real number in the format `{singleValue}f`
XML, all protocols	Real number with single precision
UrlEncoded, no protocol	Real number with single precision
UrlEncoded, OData V2 protocol, ODATA V3 protocol, or ODATA V4 protocol	A character string that represents a single-precision real number in the format `{singleValue}f`

4.2.1.3.1.1.1.20 string

Character string in the memory of the computer

A character string is a finite sequence of characters (such as letters, numbers, special characters, and control characters) from a defined character set. Characters in a character string can be repeated. The sequence of characters is defined. A character string can also be empty, that is, it can contain no characters and have the length 0.

Settings

MIME Format and Protocol	Result
JSON, all protocols	A character string is returned.
UrlEncoded, all protocols	A URL-escaped character string, enclosed in single quotation marks, is returned.

4.2.1.3.1.1.1.21 stream

Character string

Data type for a character string that is sent from the network.

Settings

MIME Format and Protocol	Result
JSON, all protocols	Character string (string)
XML, all protocols	Character string (string)
UrlEncoded, all protocols	A URL-escaped character string (without quotation marks)

4.2.1.3.1.1.1.22 time

Data type for the time

This data type enables you to store the time.

Settings

MIME Format and Protocol	Result
JSON, no protocol	A character string that represents the time in the format `hh:mm:ss.fff`.
JSON, ODATA V2 protocol	A character string that represents the time in the format `time'{hh:mm:ss.fff }'`.
JSON, ODATA V3 or ODATA V4 protocol	A character string that represents a duration in the format `time'{P{days}DT{hours}H{minutes}M{seconds}.{milliseconds}S'`
XML, all protocols	A character string that represents the time in the format `hh:mm:ss.fff`
UrlEncoded, no protocol	A character string that represents the time in the format `hh:mm:ss.fff`
UrlEncoded, ODATA V2 protocol	A character string that represents the time in the format `time'{hh:mm:ss.fff }'`
UrlEncoded, OData V3 protocol or ODATA V4 protocol	A character string that represents a duration in the format `time'{P{days}DT{hours}H{minutes}M{seconds}.{milliseconds}S'`

4.2.1.3.1.1.1.23 timeOfDay

Character string for time

This data type represents a character string that displays the time of day in the format `hh:mm:ss.fff`.

4.2.1.3.1.1.2 Creating Arrays

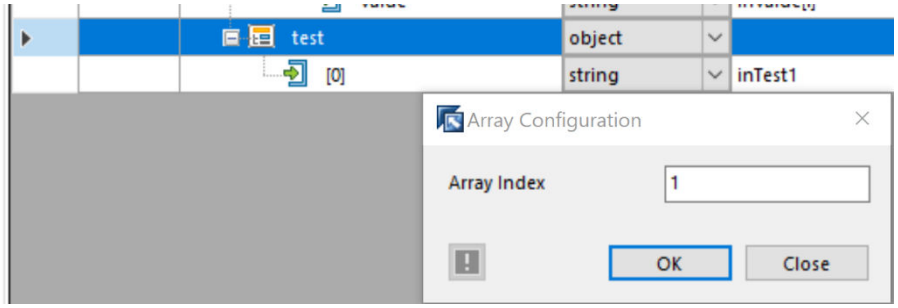
Use

You can specify an **array** as a parameter in the request message configuration and in the response message configuration.

Procedure

To create an array directly in the message configuration table, right click to call the context menu. (See also: Context Menu [page 303].)

If the selected object is an **array**, and you choose *Extend* from the context menu, the *Array Configuration* dialog box appears, where you enter the *array index*:



In this dialog box, you can configure the array mapping by entering the array index. The array index can be a number or an index variable.

- To include an array with a **dynamic size** in the path, you must enter the index variable `i` or any other variable name.
- To include an array with a **fixed size**, you must enter a number.

For more information with examples about creating arrays and for array mapping, see SAP Note 2601203.

4.2.1.3.1.1.3 Context Menu

A context menu is now available in the request message configuration table and the response message configuration table to allow you to quickly enter variables there.

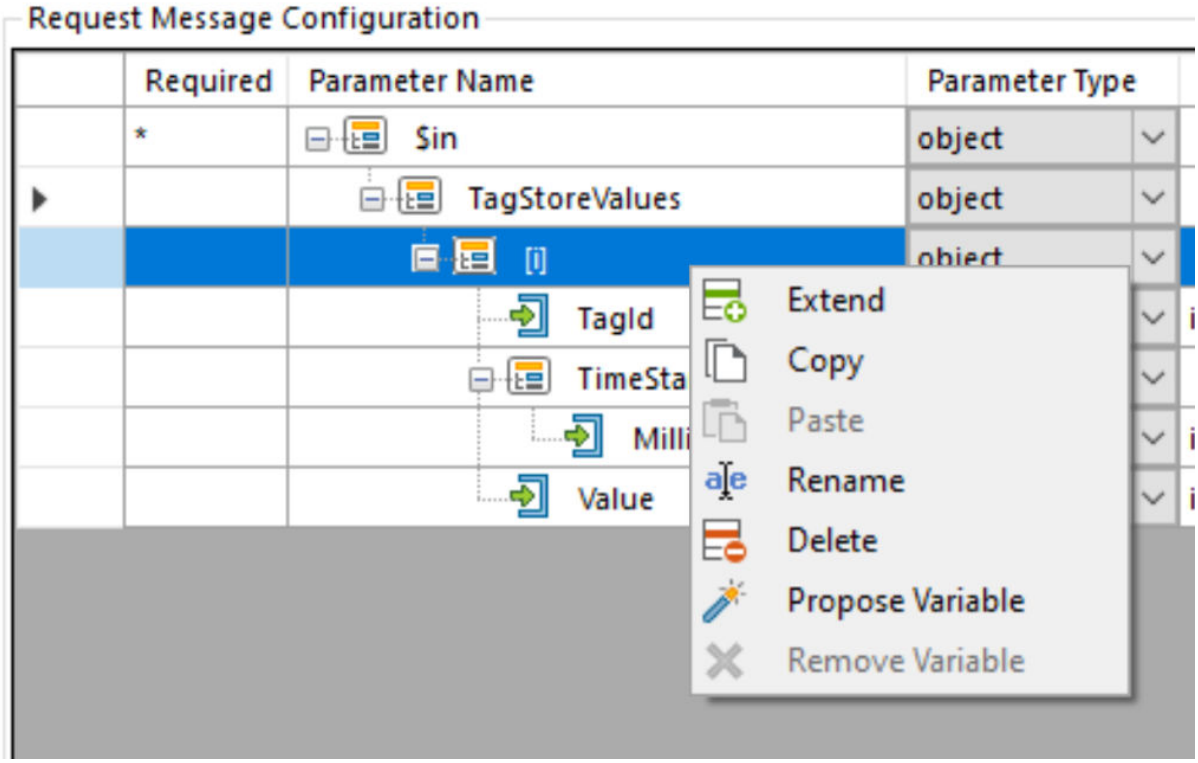
You can choose from the following functions in the context menu:

- Extend
  Depending on the selected service type, the context menu allows you to extend the object structure, the array, or the predefined type that originates from the WSDL or ODATA description.
  In the case of a RESTful Web service, the *Add Request Mapping* dialog box appears if you want to add new properties to an object. If the object under the cursor was an array, the array configuration dialog box appears. (See also: Creating Arrays [page 302].)
- Copy/Paste
  You can copy the structure of the selected parameter and later overwrite an existing parameter with the structure of the copied parameter. This functionality is useful if you want to quickly configure similar structures.
- Rename
  You can rename a parameter.
- Delete
  You can delete the selected parameter.
- Propose Variable
  If you choose this function, PCo proposes a variable name for the selected parameter.

- Remove Variable
  You can use this function to remove the variable for the selected parameter.

Example

The following figure shows how you can use the context menu to set up an object structure:



4.2.1.3.1.2 Configure Message from Template

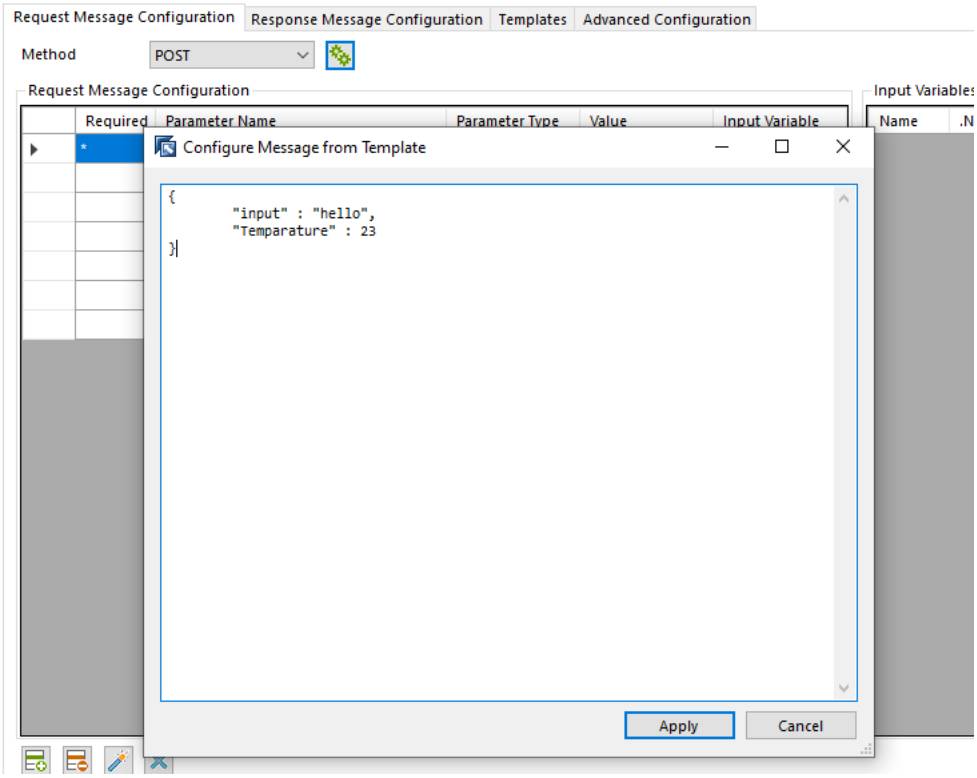
If you choose *Configure Message from Template*, you can automate the configuration of a request message for a RESTful service by entering a predefined JSON character string in the dialog box that appears. When you choose *Apply*, this data is automatically transferred to the *Request Message Configuration* table as a parameter.

> **i Note**
>
> You can also use the context menu to edit the request message configuration, and then choose *Configure Message from Template* so that you can view the syntax of the generated JSON string.

For more information about entering a template and its syntax, see Template Syntax [page 277].

Example

The following figure shows the dialog box in which you can enter a JSON string:



4.2.1.3.1.3 Examples of Request Message Parameters (RESTful)

Example 1

Field	Input
Parameter Name	`device`
Parameter Type	`string`
Value	`ABF3209`
Input Variable	Checkbox **not** selected

Result: With each request, PCo sends a parameter (fixed value) with the name *device* and the value *ABF3209*.

Example 2

Field	Input
Parameter Name	`sensor`
Parameter Type	`string`
Value	`sensorID`
Input Variable	Checkbox selected

Result: With each request, PCo sends a parameter (variable) with the name *sensor*. The value of this variable is filled dynamically using the mapping in the *notification*. In the *notification*, the field has the name *sensorID*.

Example 3

Field	Input
Parameter Name	`message[0].measurementvalue`
Parameter Type	`string`
Value	`measurementvalue`
Input Variable	Checkbox selected

Field	Input
Parameter Name	`message[0].unit`
Parameter Type	`string`
Value	`unit`
Input Variable	Checkbox selected

Result: We have now created a static array with only one entry. This entry consists of two lower-level entries:

- Measured value
- Unit

In the case of JSON, the array would look like this:

"message":[{"measurementvalue":"<value>", "unit":"<unit>"}]

4.2.1.3.2 Request Message Configuration Tab (OData and Web Service)

Prerequisites

On the *Web Service Settings* tab, you have selected the service and the operation.

Context

This document describes the procedure for creating the request message configuration for the operation of an OData service or Web service.

The *Request Message Configuration* tab contains the table with the parameters for the *request message* and the *list of input variables*.

> **i** Note
>
> If, in the case of OData, you have selected the `Read_all` operation, the table remains empty because the `Read_all` operation does not require any request parameters.
>
> The parameters of the selected operation are displayed automatically for all other operations.

The parameters of the request message can be fixed values or variables. Each parameter has a name, a type, a value, and the indicator showing whether it is a variable.

Procedure

1. PCo has determined the available request parameters for the selected operation and displays them in the *Request Message Configuration* table.

Field	Description
Required Parameter	If the checkbox is selected, this is a required parameter. An entry is required in the *Value* field in this row. The checkbox for required parameters is selected automatically and cannot be changed. If the checkbox is not selected, the entry is optional.

Field	Description
Parameter Name	Specifies the name of the parameter that is provided by the service (cannot be changed)
Parameter Type	Specifies the type of the parameter that is provided by the service (cannot be changed)
Value	This field contains the value 0 at first. You can either enter a fixed value or specify a variable for the operation field here. **i Note** Ideally you choose  and then the system proposes a variable name. **i Note** The variable names must not begin with a digit, must not contain any spaces, and must not be any C# key words. If the value entered is not a correct variable or is not the correct literal for the given type, a small red icon appears at the beginning of the row. If you move the mouse over it, PCo displays a tooltip with the error description.
Input Variable	Checkbox with which you specify that the selected parameter is a variable. The checkbox is not selected. If you do not select the checkbox, the value entered is interpreted as a fixed value.

2. Select a row and choose  .

   The system checks if the parameter is a variable. If it is, the system proposes a variable name (with the prefix *in*) and selects the *Input Variable* checkbox. In addition, the  icon is displayed in front of the parameter.
3. All parameters that you have marked as variables are displayed in the *list of input variables*. The entry in the *Value* field is used as the variable name.

> **Note**
>
> Later you can map this variable to a subscription item in the notification object. You perform this mapping on the *Destinations* tab for the notification and thereby fill the service with the values from the data source.

4. You can use the *Import Data Types into Data Type Repository* button to import structured data types, array data types and enumeration data types from SOAP Web services and OData services. This simplifies data entry considerably, since only one variable then needs to be selected from the data type repository. (See also: Import Data Types into the Data Type Repository [page 309].)
5. By choosing  , you can convert a variable back into a fixed value.

Example

Example: Create a Universal Web Service Destination System (OData) [page 329]

4.2.1.3.2.1 Import Data Types into the Data Type Repository

Use

You can import structured, array, and enumeration data types from SOAP Web services and OData Web services as customer-specific data types **in the universal Web service destination system** into the PCo data type repository. For RESTful Web services, you can import the defined message structure as a structured PCo data type into the PCo data type repository.

During the import, PCo data types are created that correspond to the data types of the Web service. You can then assign these data types to the variables of the Web service using the data type selection dialog. In this way, it is possible, for example, to transfer a variable with a deeply nested structured data type to the Web service call.

When importing the data types, you must specify one or more namespaces for the new PCo data types. The namespaces must follow the namespace rules. (See also: Namespaces [page 586].)

Procedure

To import the data types of a Web service, proceed as follows:

1. Create a universal Web service destination system and select **`WSDL`** (for a SOAP Web service) or **`OData`** as the *description type*.

Configure the destination system so that the Web service operations offered by the Web server appear in the service explorer.

2. Navigate then either to the *Operation Configuration* > *Request Message Configuration* or *Operation Configuration* > *Response Message Configuration* tab.
3. Choose the *Import Data Types into Data Type Repository* button.
   The system collects all Web service data types from the Web service descriptions and uses them to generate proposals for corresponding PCo data types. These proposals are displayed in the *Import Data Types into Data Type Repository* dialog box. The dialog box also shows you whether or not Web service data types and corresponding PCo data types are already known in the PCo data type repository.
   If you have configured a RESTful Web service, the defined message structure is imported into the data type repository as a structured PCo data type in this step.
4. Perform the import. (See also: Functions in the Import Data Types Dialog Box [page 587].)

Related Information

PCo Data Types [page 582]

4.2.1.3.3 Response Message Configuration Tab (RESTful)

Context

This document describes the procedure for **response message configuration for a RESTful Web service**. For the RESTful Web service, the table with the response variables is empty at first. You have to choose  to create the parameters.

Procedure

1. Choose the *Response Message Configuration* tab.
2. The following checkboxes are displayed on the tab:

Checkbox	Description
Map HTTP Status	With this checkbox, you can specify that the `HTTP_STATUS` variable is included in the list of variables. **i Note** You cannot rename this variable. The variable always has the type *int*. **i Note** If you have selected the *Map HTTP Status* checkbox, any exceptions that occur when a Web service is called are ignored. Therefore, you first need to check if such exceptions could occur.
Return Raw Response	With this checkbox, you can define that the response of the service that is called is sent completely and without parsing. If you select the checkbox, all mapped variables, with the exception of HTTP_STATUS, are removed from the list of variables. The RAW_RESPONSE variable is included in the list of variables. **i Note** You cannot rename this variable. The variable always has the type *string*.

3. Choose  to create a parameter.

   The *Add Response Mapping* dialog box appears.
4. Enter the following in the dialog box:

Field	Description
Parameter	Specify the path for the parameter that is to be mapped. For a RESTful service, you can also specify the object **$out** as a root element here. This name is reserved for body style Web requests that are used for serializing the response parameters (without wrapper elements). **i Note** If you enter an invalid parameter, a small red symbol with an exclamation mark is displayed in front of the text box for the parameter.
Type	Select the parameter type. The following types are supported: ○ binary [page 293] ○ base64Binary [page 293] ○ boolean [page 293] ○ byte [page 294] ○ date [page 294] ○ dateTime [page 294] ○ dateTimeOffset [page 295] ○ jsDate [page 296] ○ double [page 297] ○ duration [page 298] ○ float [page 298] ○ GUID [page 298] ○ int32 ○ single [page 300] ○ string [page 301] ○ stream [page 301] **i Note** The parameter type cannot be changed later.
Variable	The variable name is proposed here automatically. All the response variables have the prefix *out*.

5. Choose the *Add* pushbutton.

   The variable is copied into the *Response Message Configuration* table and into the table of *output variables*. In the table, only the variable name (in the *Output Variable* column) can be changed.

6. If you choose *Select Data Type* for an output variable of the type structured data type, the PCo data type repository appears, where you can select the (previously imported) structured data type. (See also: Select Data Type (Selection Dialog) [page 588].)

In this case, you only have to enter a single output variable in order to transfer complex and deeply nested data objects. This reduces the configuration effort significantly.

7. If you want to remove a parameter from the table completely, select the row and choose  .

Related Information

Add a Request or Response Mapping [page 289]
Test Configuration [page 326]

4.2.1.3.4 Response Message Configuration Tab (OData and WS)

Context

The document describes how you need to proceed when configuring the response parameters for an OData or a Web service.

Procedure

1. Choose the *Response Message Configuration* tab.

All the parameters of the selected operation are already on this tab. The following data is displayed on the tab:

Checkbox	Description
Map HTTP Status	With this checkbox, you can specify that the `HTTP_STATUS` variable is included in the *list of variables*. **i Note** You cannot rename this variable. The variable always has the type *int*. **i Note** If you have selected the *Map HTTP Status* checkbox, any exceptions that occur when a Web service is called are ignored. Therefore, you first need to check if such exceptions could occur.
Return Raw Response	With this checkbox, you can define that the response of the service that is called is sent completely and without parsing. If you select the checkbox, all mapped variables, with the exception of `HTTP_STATUS`, are removed from the list of variables. The RAW_RESPONSE variable is included in the *list of variables*. **i Note** You cannot rename this variable. The variable always has the type *string*.

Screen Area	Description
List of Output Variables	All selected output variables of the response are displayed later in this area.
Response Message Configuration	This table contains the output parameters that you can configure.

2. You can configure the following for a parameter in the *Response Message Configuration* area:

Field	Description
Required Parameter	The checkbox indicates that an entry is required in the *Value* field in this row. The checkbox for required parameters is selected automatically and cannot be changed. If the checkbox is not selected, the entry is optional.
Parameter Name	The name of the parameter **cannot be changed**. It is based on the metadata description of the selected operation.
Parameter Type	The type of the parameter **cannot be changed**. It is based on the metadata description of the selected operation.
Output Variable	In the *Output Variable* column, you can specify either the variable name of the assigned variable for a parameter, or the *Create* pushbutton is displayed. The *Create* pushbutton is always displayed if the parameter is an array or a complex object with lower-level parameters.

3. Click on the *Create* pushbutton for the parameter you want.

   The data structure for the parameter is opened and additional fields are offered in the structure.

4. If you select a row and choose  , the system checks if it is a variable and proposes a name for the output variable, for example, *outID*.

   The variable is flagged with the icon  and added to the list of output variables.

5. You can use the *Import Data Types into Data Type Repository* button to import structured data types, array data types and enumeration data types from SOAP Web services and OData services. This simplifies data entry considerably, since only one variable then needs to be selected from the data type repository. (See also: Import Data Types into the Data Type Repository [page 309].)

6. To delete the variable name, choose  .

Next Steps

Example: Create a Universal Web Service Destination System (OData) [page 329]

4.2.1.3.5 Templates Tab

On this tab, you can configure the templates for the query part of the URI and for the request body. The two templates are independent of each other.

This tab is subdivided into two further tabs:

- Query
  You can specify a JSON string here that must start with a `?`. The JSON character string you enter here is appended later to the endpoint URI.
  The query template is used together with the endpoint URI to form the complete URL when a particular Web service is called.

  > **Example**
  >
  > The following query template is valid:
  >
  > ```
  > /weather/{state}/{city}?forecast={days:int}
  > ```

  Leading and trailing slashes are optional in the query template. The default data type of the input variables is `string`. However, you can specify one of the permitted primitive Web service data types.
  For more information, see Templates Tab [page 315].
- Request
  You can enter data for the message body of the request here. This request template uses the same syntax but is sent as part of the request body. To be able to configure it, you must select a method that is allowed for the request body:
  - PUT
  - MERGE
  - PATCH
  - DELETE
  - POST

  The *Message Configuration* table remains empty and is disabled when you make an entry in the request template.

Every change in the templates updates the list of input variables.

4.2.1.3.6 Advanced Configuration Tab

Context

This tab is used for additional settings. All fields contain default values. However, you can change these settings.

Procedure

Enter the following data on the *Advanced Configuration* tab:

Screen Area	Description
Headers	In this table, you can enter the headers that are to be sent with the Web service call (request). Each row in this table represents a single header and its value. The header values can be constant or variable (a template). (See: Template Syntax [page 277].) If you select a row in the header row table, you can delete the row by selecting it and pressing the delete key. **Example** You enter the following: Headers **Field** / **Input** *Header Name* / `headerOne` *Value* / `header{Example1}` **Result**: The value `Example1` is added to the table of input variables on the *Request Message Configuration* tab.
Message Bundling Settings	In the *Bundling Index* field, you can enter the parameter `i` to activate the **message bundling** function. Message bundling is only suitable for Web service destination systems for which there is at least one input parameter of type **array**. For these parameters, use input variables with the notation `<variablename>[i]`. In order that the messages are bundled, you also need to make an entry in the *Fixed Number of Messages* field or in the *Maximum Accumulation Time* field on the *Message Delivery* tab for the notification. For more information, see Message Bundling [page 579].

Screen Area	Description
Limit	In this field, you can specify the maximum number of simultaneous HTTP calls. This parameter specifies the maximum number of connections to the same server. The largest number you can enter is **100**. **i Note** This parameter not only affects the current destination system, but also **all universal Web service destination systems for the same server**. Therefore, you need to make sure that you have maintained the parameters in all Universal Web service destination systems for the same server consistently. Otherwise, the setting of the destination system that was called first (randomly) takes effect.
Request Settings	The settings for the data transfer for the request are displayed in this screen area. See: Request and Response Settings [page 318]
Response Settings	The settings for the data transfer for the response are displayed in this screen area. See: Request and Response Settings [page 318]
Session Configuration	In this screen area, you can make **security settings for OData and RESTful Web services** to prevent cross-site request forgery attacks. See: Session Configuration [page 325]

Next Steps

The OData protocol has a special convention for URL encoding.

For more information about OData URL conventions, see http://www.odata.org/documentation/odata-version-2-0/uri-conventions/ .

4.2.1.3.6.1 Request and Response Settings

The technical settings for data transfer of the request and response messages are displayed in the lower screen area of the *Advanced Configuration* tab. This data is determined automatically using the selected operation.

i Note

The settings can **only be changed for RESTful Web services**.

You can only change the settings for OData and Web services if you choose  .

Field	Description
MIME Format	You can define the MIME format here. (MIME = Multipurpose Internet Mail Extensions) The MIME format defines the format that is to be used for the request and response messages. The following formats are supported: • *UrlEncoded* You use this format for UrlEncoded messages if the message is sent as part of the URL. In this case, an HTTP request is used without a body and the *content type* is not relevant. • *FormUrlEncoded* You use this format for Form-UrlEncoded messages in conjunction with the *content type* `application/x-www-form-urlencoded`. • *JSON* You use this format for OData and RESTful services in conjunction with the *content type* `application/json`. • *Text* You use this format to be able to send a text as part of the Web request body. If you select this format, you must set the option `text/plain` in the *Content Type* field. • *XML* You use this format for XML or SOAP messages in conjunction with the *content type* `text/xml` or `application/soap+xml`. • *Multipart* You use this format if you want to upload or download multiple files, such as texts or images. The prerequisite is that you are using a RESTful Web service. PCo then adds the content type `multipart/form-data` automatically. As a result, the system automatically displays the parameters required for uploading or downloading on the *Request Message Configuration* tab or on the *Response Message Configuration* tab. (See also: Uploading and Downloading Files [page 320].) • *Binary* You use this format if you want to upload or download a file. The prerequisite is that you use a RESTful service. The content type `application/octet-stream` is added automatically. (See also: Uploading and Downloading Files [page 320].)

Field	Description
Content Type	The content type defines the content of the request and response bodies. The following content types are supported: • `application/json` • `application/soap+xml` • `application/x-www-form-urlencoded` • `text/plain` • `text/xml` • `multipart/form-data` • `application/octet-stream`
Protocol	The following protocols are supported: • *None* • *OData (version 2, 3, and 4)* • *Soap* • *Soap12*
Encoding	You can select the encoding format you want here: • *ascii* • *utf-8* • *utf-16* (See also: Encoding (BC-I18).)

4.2.1.3.6.1.1 Uploading and Downloading Files

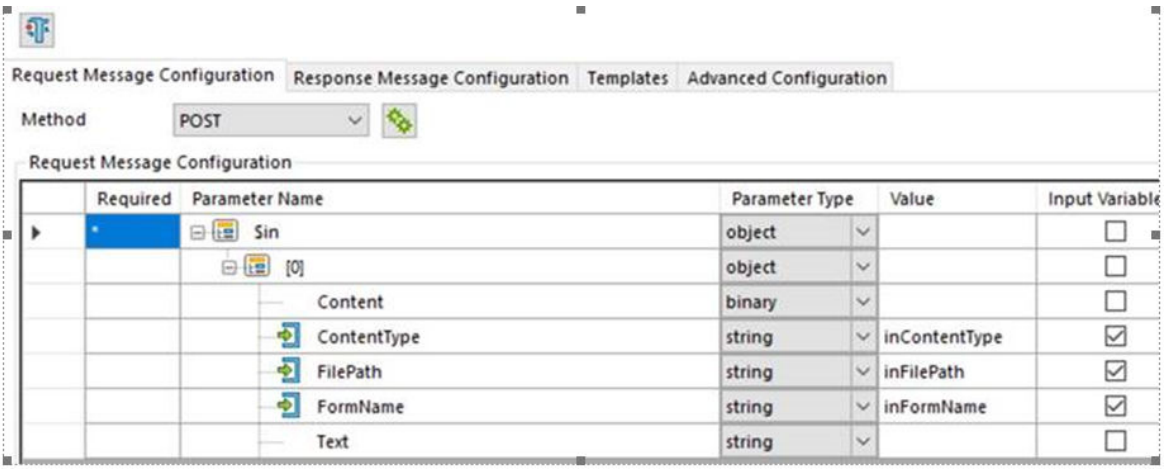
Use

The universal Web service destination system can be used to upload and download files (for example, texts, images, or byte arrays) to a remote server. The prerequisite is that you use a RESTful service. You can use this feature by selecting the **Multipart** option in the *MIME Format* field. The new MIME format is offered in the request settings and in the response settings. The **request setting** of the MIME format is used for **uploading** files, while the **response setting** of the MIME format is used for **downloading** files. The parameters for uploading or downloading files are then displayed on the Request Message Configuration or Response Message Configuration tab. (See also: Uploading Multiple Files (MIME Format Multipart) [page 321] and Downloading Multiple Files (MIME Format Multipart) [page 322].)

If you only want to upload or download **one file**, you have to select the new MIME format **Binary**. (See also: Uploading a File (MIME Format Binary) [page 323] and Downloading a File (MIME Format Binary) [page 324].)

Example

1. Choose the *Operation Configuration* tab and enter the *endpoint URI*. The endpoint URI has to be the URI of the server to which the file is to be sent.
2. Choose the *Advanced Configuration* tab.
3. In the *Request Settings* screen area, in the *MIME Format* field, choose the *Multipart* option.
   The entry `multipart/form-data` then appears automatically in the *Content Type* field.
4. Choose the *Request Message Configuration* tab.
   The available parameters are displayed here.
5. Select the parameters you want to use and, if necessary, change the values proposed for the input parameters. (See also: Uploading Multiple Files (MIME Format Multipart) [page 321].)
   Result:



6. If you want to upload multiple files, you have to select the root parameter $in and select the *Extend* entry in the context menu.
   The *Array Configuration* dialog box appears.
7. Choose the *OK* button to add another array (that is, another form) for a file.

4.2.1.3.6.1.1.1 Uploading Multiple Files (MIME Format Multipart)

If you select **Multipart** in the *MIME Format* field in the *Request Settings* area, PCo sends a message to the remote server that contains multiple forms. Each form is represented as an array element. Once the **Multipart** option has been selected, a fixed set of parameters is displayed on the *Request Message Configuration* tab.

The structure of the forms is reflected in the parameters that are offered. You can configure these parameters for the request message:

Parameters for Uploading Files

Parameter Name	Description
$in[0].FormName	Mandatory string parameter for defining the form name.

Parameter Name	Description
$in[0].FilePath	String parameter for defining the absolute path to a file that is to be used as the form content. This parameter cannot be used together with **$in[0].Text** or **$in[0].Content**.
$in[0].Text	String parameter for defining the form content. This parameter cannot be used together with **$in[0].Content** or **$in[0].FilePath**.
$in[0].Content	Binary parameter for defining the form content if it cannot be represented as a character string. This parameter cannot be used together with **$in[0].Text** or **$in[0].FilePath**.
$in[0].ContentType	Optional parameter for specifying the content type for the form. This can be any valid content type, for example, `text/plain`.

On the *Request Message Configuration* tab, you can only select one of the alternative **parameters $in[0].Text, $in[0].Content, and $in[0].FilePath as the input variable**. You can change the names of the proposed input variables.

If you want to upload more than one form, you can extend the root parameter **$in** and thus add more forms to the array.

> **i** Note
>
> Parameters that you have not flagged as input variables are no longer displayed when you leave the destination system and reopen it.

4.2.1.3.6.1.1.2 Downloading Multiple Files (MIME Format Multipart)

You can use the universal Web service destination system to download files from a remote server. The content of the downloaded files is retained in the memory and can be further processed by PCo.

If you select **Multipart** in the *MIME Format* field in the **response settings**, PCo can receive a message from the remote server containing multiple forms. Each form is represented as an array element. The structure of the forms is reflected in the following set of parameters that can be used to configure the response message:

Parameters for Downloading Files

Parameter Name	Description
$out[0].FormName	String parameter for the name of the form to be received.
$out[0].FileName	String parameter for the file name of the form to be received.

Parameter Name	Description
$out[0].Text	String parameter for the form content if the content type of the received form is recognized as a text. The following content types are supported: • text/plain • text/xml • application/x-www-form-urlencoded • application/json • application/soap+xml
$out[0].Content	Binary parameter for the content of the form received if it is not recognized as a text.
$out[0].ContentType	Content type of the form received.

If you expect further forms, you can expand the root parameter **$out** to add more forms to the array.

> **i** Note
>
> Parameters that you have not flagged as input variables are no longer displayed when you leave the destination system and reopen it.

4.2.1.3.6.1.1.3 Uploading a File (MIME Format Binary)

You can use the universal Web service destination system to upload a file in the MIME format binary to a remote server.

If you select **Binary** in the *MIME Format* field in the *Request Settings* area, PCo is configured so that it sends a binary message to the remote server. The content of the message is determined by the following fixed set of parameters that can be used for configuring the request message:

Parameter

Parameter Name	Description
$in.FilePath	String parameter for defining the absolute path to a file that is used for the content of the HTTP request. This parameter must **not** be used together with **$in.Text** or with **$in.Content**.
$in.Text	String parameter for defining the content of the HTTP request. This parameter must **not** be used together with **$in.Content** or with **$in.FilePath**.

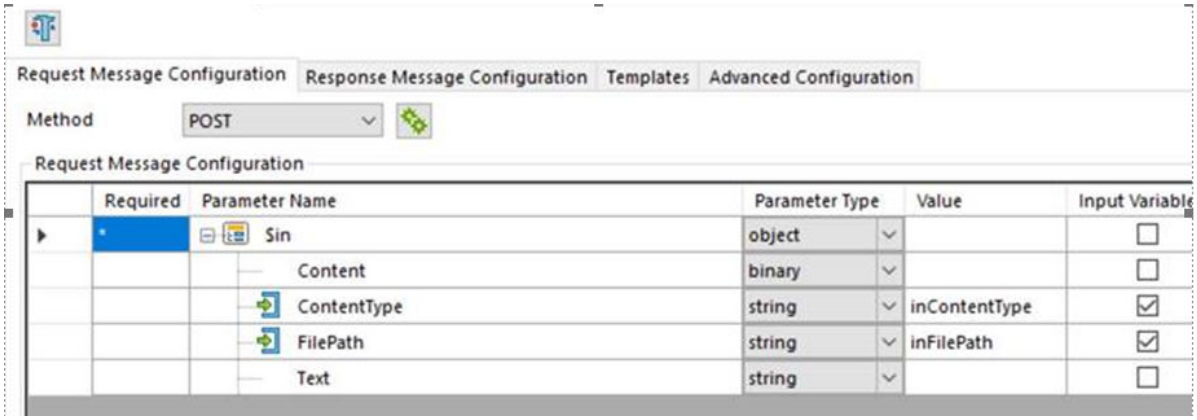
Parameter Name	Description
$in.Content	Binary parameter for defining the content of the HTTP request if it cannot be represented as a character string. This parameter must not be used together with **$in.Text** or with **$in.FilePath**.
$in.ContentType	Optional parameter for specifying the content type for the payload. This can be any valid content type, for example, `text/plain`.

Depending on how the HTTP request is created, only one of the alternative parameters $in.Text, $in.Content, and $in.FilePath can be flagged as an input variable. Parameters that are not required are deleted.

You can change the names of the proposed input variables.

Example

The following example shows the input parameters for uploading a file. A path to the file (FilePath) and the content type (ContentType) are used as the input variable here.



4.2.1.3.6.1.1.4 Downloading a File (MIME Format Binary)

You can use the universal Web service destination system to download a file in the MIME format binary from a remote server.

If you select **Binary** in the *MIME Format* field in the *Response Settings* area, PCo is configured so that it receives a binary message from the remote server. The content of the message is determined by the following fixed set of parameters that can be used for configuring the request message:

Parameters for Downloading Files

Parameter Name	Description
$out.Text	String parameter for the content of the HTTP response when its content type is recognized as a text. The following content types are supported: • text/plain • text/xml • application/x-www-form-urlencoded • application/soap+xml
$out.Content	Binary parameter for the content of the HTTP response if it is not recognized as a text.
$out.ContentType	Content type of the HTTP response

> **i** Note
>
> Parameters that you have not flagged as input variables are no longer displayed when you leave the destination system and reopen it.

4.2.1.3.6.2 Session Configuration

Use

In this screen area you can make security settings for ODATA and RESTful Web services to prevent cross-site request forgery attacks.

Prerequisites

You can only change the settings in this screen area if you first choose  .

Features

Field	Description
Use Sessions	Select this indicator so that you can use session tokens. The tokens are unique and become invalid after a given time.
Session URL	Specifies the URL under which the tokens and session cookies can be called up.
Tokens	Displays the token. You can also enter a list of token names here, separating the individual values by a comma.
Keep Session Open	By selecting this checkbox, you specify that an existing session token is to be used. If you do not set the indicator, session tokens and cookies are called up before each request.

4.2.1.3.7 Test Configuration

Prerequisites

You have configured the request message and the response message.

Procedure

1. Choose the *Test Configuration* button.

   The *Test Request* dialog box appears. The dialog box consists of four tabs:

Tab	Description
Request Variables	This tab contains a table with the request variables that you defined previously. Enter a meaningful value for each variable.
Response Variables	This tab contains a table with the response variables. Once a test call of the Web service has been executed successfully, the variables from the response are entered here automatically.

Tab	Description
Raw Request	The data for the request is displayed here.
Raw Response	Here you can display the raw data for the response that the server has sent. The data is displayed in raw form, in other words, before it is converted to response variables in the unmarshalling operation. The response contains the following data: ○ HTTP status ○ Header ○ Cookies ○ Response text

2. Choose the *Test* button.

   If the configuration has not yet been tested, the message `Web service not called` is displayed in the dialog status bar.

   When the test has been executed successfully, you get the message `Test OK`.
3. When a test is completed successfully, choose the *Map Response* button.

   By choosing this button, you can automatically create the *response message configuration* based on the response received after a successful test. You can only use the function for a RESTful Web service in conjunction with a JSON response.

4.2.1.4 Proxy Settings Tab

Use

To enable PCo to read the data of the service entered, you first need to make the proxy settings.

Procedure

Choose the *Proxy Settings* tab and enter the following:

Field Name	Description
Use Default Proxy	If you select this option, the default proxy server is used. **i Note** To specify the proxy server that you want to be used by default, do the following: 1. In the Internet Explorer menu, choose *Tools* > *Internet Options*. 2. Choose the *Connections* tab. 3. Choose the *LAN Settings* pushbutton. In the dialog box that appears, select the *Use Automatic Configuration Script* checkbox and enter the proxy server in the *Address* field. 4. Confirm your entries by choosing the pushbutton *OK*.
Bypass Proxy	You can use this option if the host of the Web service is in the local network. If this is the case, this is the recommended setting as it improves performance.
Use Specified Proxy	If you select this option, the three fields below become ready for input and you can enter a specific proxy server.
Proxy URI	You enter the Uniform Resource Identifier of the proxy server here. (See also: Uniform Resource Identifier.) You need to use the HTTP protocol schema, such as `http://proxy:8080`. **i Note** The use of the HTTPS protocol schema is not required for the proxy because communication takes place using the proxy itself. The communication direction is as follows: *PCo* > *Proxy* > *Cloud*. Communication between the proxy and the cloud server is carried out automatically using HTTPS, if you previously set the endpoint URI of the cloud server in HTTPS format on the *Web Service Settings* tab.
User Name	You only need to enter the user name here if authentication is required for the proxy server.

Field Name	Description
Password	You enter the password for the proxy server here.

4.2.1.5 Example: Create a Universal Web Service Destination System (OData)

Use

The following example describes the process for creating a universal Web service destination system. The OData service provided by Microsoft, `http://services.odata.org/V3/OData/OData.svc`, is used as the example service.

Procedure

Defining Web Service Settings

1. Choose *Add Destination System*. Then choose the destination system type *Universal Web Service Destination System*.
2. Enter the name of the destination system and choose OK.
   The *Web Service Settings* tab appears.
3. In the *Description Type* field, select the option *OData*.
4. Enter the example service in the *Description URI* field:
   `http://services.odata.org/V3/OData/OData.svc`.
5. Choose the *Explore* pushbutton.
   All available data and service operations for the selected service are displayed in the *Service Explorer* screen area.
6. Choose the first node *ODataDemo.Advertisement* by clicking on this node.
   You have now selected the operation *Read:ODataDemo.Advertisement*. This operation is displayed at the bottom in the *Operation* field.
7. Press ENTER.
   The *Operation Configuration* tab appears.

Creating the Operation Configuration

1. The following data is displayed for the selected service operation on the *Operation Configuration* tab:

Field	Data Displayed
Endpoint URL	`http://services.odata.org/V3/OData/OData.svc`

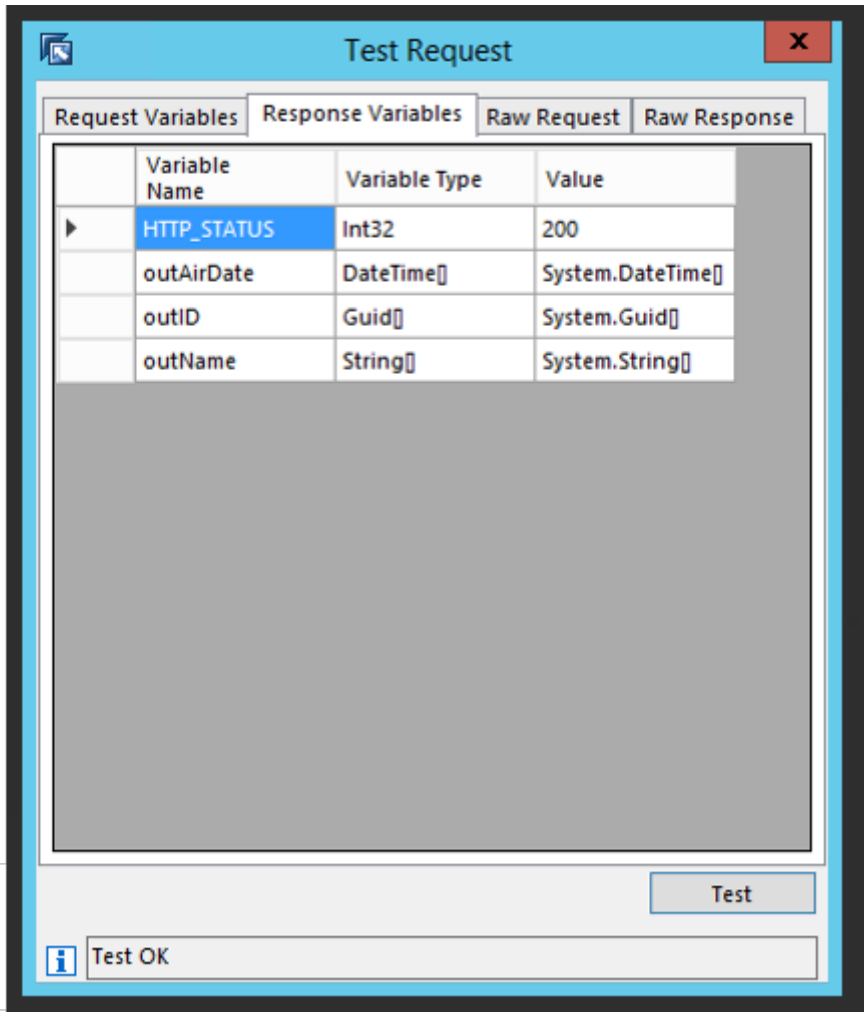
Field	Data Displayed
Operation	`Read:ODataDemo.Advertisement`

2. Since you have selected the `Read:ODataDemo.Advertisement` operation of an OData service, the *Request Message Configuration* tab remains empty because the operation does not need any *request parameters*. The parameters of the selected operation are displayed automatically for all other operations.
3. Choose the *Response Message Configuration* tab in the lower screen area. Select the *Map HTTP Status* checkbox.
   The `HTTP_STATUS` variable is displayed in the *list of output variables*.
4. A row with the *parameter* `value`, the *parameter type* `ODataDemo.Advertisement[]`, and the *Create* pushbutton is displayed in the *Response Configuration* table. Choose the *Create* pushbutton.
   The *Array Mapping* dialog box appears.
5. Enter the letter **`i`** (index variable) in the *Array Index* field to be able to create an array with a dynamic size. Click *OK*.
   The system displays a new node with the name *[i]* in the *Response Message Configuration* table. In addition, the *Create* pushbutton appears at the end of the row.
6. Choose the *Create* pushbutton.
   The node [i] is expanded and displays the parameters *ID*, *AirDate*, and *Name*. A complex parameter with the name *FfeaturedProduct* is also displayed.
7. Select the row of the field *ID* and choose  at the bottom of the screen.
   PCo displays `outID[i]` in the *Variable* field. The variable is displayed with an icon  .
8. Repeat this for *AirDate* and *Name*.
   The variables are displayed in the list of output variables.

Test Configuration

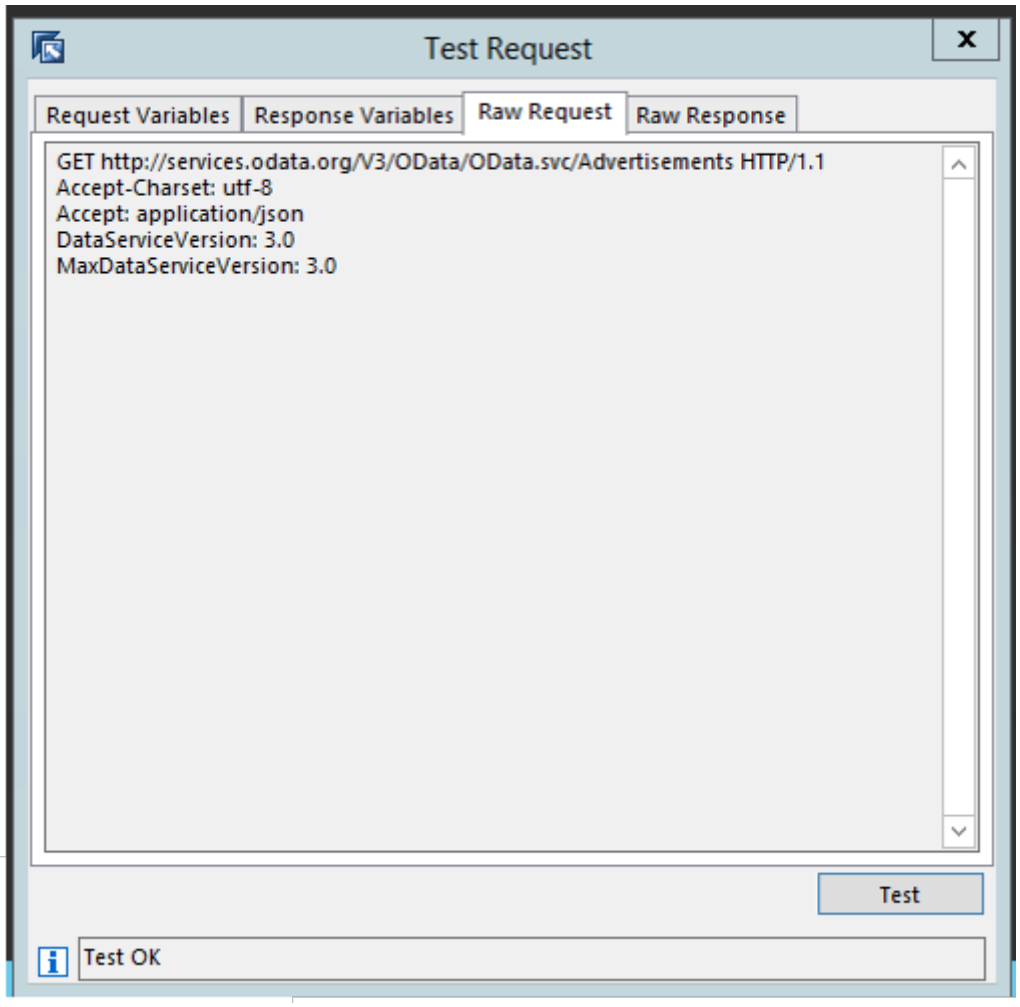
1. Choose the *Test Configuration* pushbutton.
   The *Test Request* dialog box appears. The following tabs are displayed:
   - The *Request Variables* tab is empty at this point because this `Read_all-Operation` does not require any *request variables*.
   - The *Response Variables* tab contains all variables from the *list of variables*.
   - The *Raw Request* and *Raw Response* tabs already contain values.
   - The message *Web service not called* is displayed in the status bar of the dialog window.
2. Choose the *Test* pushbutton.
   If the test was successful, the message *Test OK* appears in the status bar.
   - The *Value* column on the *Response Variables* tab is filled with the symbol *[ ]*. This means that more than one value has been read.

If you had selected `HTTP_STATUS`, the value *200* (*HTTP Status OK*) is now displayed for this variable:



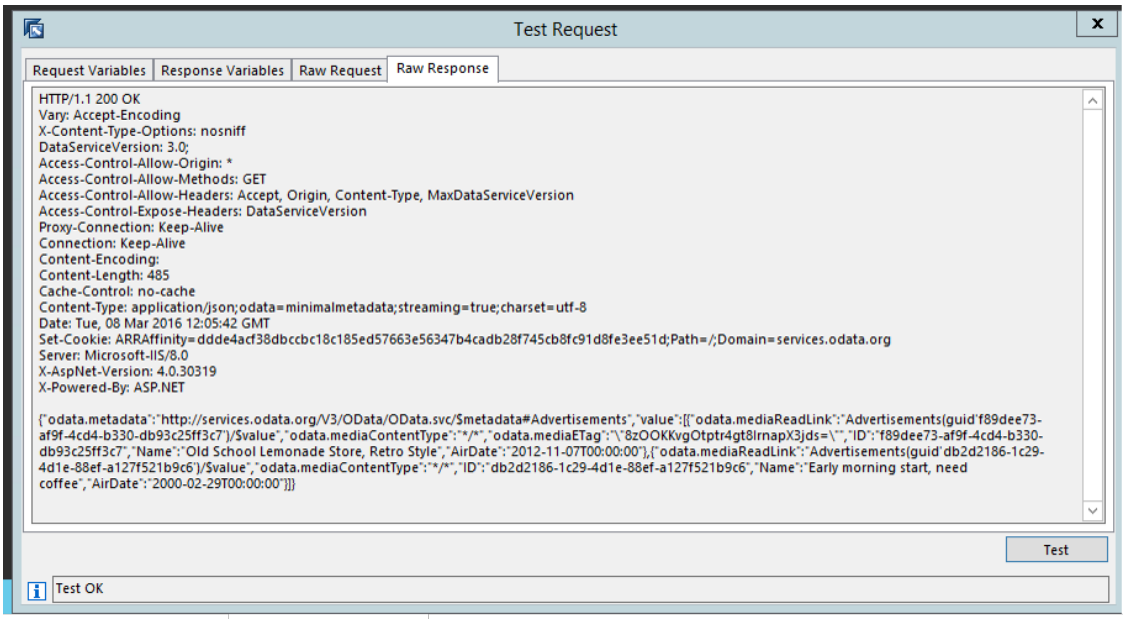
Example of Response Variables

- The *Raw Request* tab now contains the URI and header data that is sent during the request as plaintext:



Example of Raw Request

- The *Raw Response* tab contains the response from the server in the form of plaintext. The response text must be a valid JSON object. The response looks like this:



Example of Raw Response

3. Close the *Test Request* dialog box.

4.2.2 OPC UA Destination System

Definition

Destination system type that allows a method to be called on an OPC UA server. This server may be configured on a remote system or on an agent instance of your own PCo installation.

Use

You use an OPC UA destination system to be able to connect an OPC UA server.

You can use OPC UA destination systems in the following notification scenarios:

- Tag-based notification
  The OPC UA destination system is used here as a destination of a static or a versioned notification. PCo sends the changed tag values of the source system that is to be monitored using the notification messages to the OPC UA destination system. The OPC UA destination system calls the method defined previously in the OPC UA destination system to the remote server, and transfers the values. PCo can predefine fixed values for specific parameters of the method.
- Method notification
  The OPC UA destination system is used here as the destination for method notifications. The parameter values are forwarded from the method of the OPC UA server, which has been configured on an agent

instance of a PCo system, via the method notification and the OPC UA destination system to the remote server. On the remote server, the method defined in the destination system is called and executed. In this scenario, the remote server can also send a response to PCo. For more information, see Method Notification [page 55].

The following tabs are available for configuring an OPC UA destination system:

- *Session* tab
  You specify here the OPC UA server that you want to connect by entering the name of a valid client configuration. You define a client configuration beforehand using the OPC UA source system. (See: Session Tab [page 334].)
- *Security* tab
  You cannot make any changes here. All the relevant settings are copied over automatically from the assigned client configuration, that is, from the assigned OPC UA source system. (See:OPC UA Source System: Security Tab [page 154].)
- *Method* tab
  The *Method* tab allows a selected OPC UA method, which is provided by the OPC UA server, to be configured. As soon as you have selected the method, the input and output parameters provided by the method are determined and displayed. (See: Method Tab [page 335].)
- *Reliable Connection* tab
  You use this tab to maintain the connection to the OPC UA server. PCo checks the connection regularly if a problem with the connection has been detected. (See: Reliable Connection Tab [page 337].)

4.2.2.1 Session Tab

Use

If you want to connect a server that is based on OPC UA to PCo, you need to create an OPC UA destination system [page 333].

Prerequisites

You have created an OPC UA source system that you want to assign as an *OPC client*.

Procedure

1. Choose the *Add Destination System* function key. Then choose the destination system type *OPC UA destination system*.
   PCo displays the tabs for configuring the OPC UA destination system.
2. On the *Session* tab, in the *Name* field, select a valid *client configuration*.
   You created the client configuration previously using an OPC UA source system. From the viewpoint of the destination system, the configuration data is only readable here.

3. Choose *Test Configuration* to check if a connection to the OPC UA server can be established.
4. Save your entries.

More Information

For more information about creating a client configuration, see OPC UA Source System: Session Tab [page 143] and OPC UA Source System: Security Tab [page 154].

4.2.2.2 Method Tab

Prerequisites

You have created an OPC UA destination system and selected a *client configuration* on the *Session* tab.

Context

On the *Method* tab you can select and configure an OPC UA method that is provided by the connected OPC UA server (remote server).

If you are using tag-based notifications, you can send the values of the subscribed tags to the selected remote server method later using notification messages.

If you are using method notifications, the parameter values of the server method are forwarded to the remote method on the remote server by means of the method notification.

The remote method is called from the OPC UA destination system.

Procedure

1. To select a method, choose *Browse* in the *Method Selection* screen area.

   The *Browse Methods* dialog box is displayed. You can use this to select a method from the connected OPC UA server. For selecting the method, starting from this dialog box, a separate connection is set up to the OPC UA server. When you close the window, this is disconnected.
2. Choose the *Browse* pushbutton in the dialog box.

   A view of the address space is displayed that enables methods to be selected.
3. To display the hierarchy of the address space, open the top node (*Address Root*) and then all lower-level nodes. Select the method you want.

4. Confirm your selection by choosing OK.

   The input parameters and output parameters of the selected method are displayed in the corresponding screen areas.

   > **i** Note
   >
   > You can change the data type for output parameters, if required. For OPC UA, it is possible to define methods with output parameters that have an abstract data type. If you choose such a method, you can only use this once you have assigned a concrete data type to the method. You can assign a concrete data type here.

5. Save your entries.
6. You can call and test the selected method by choosing *Test Method*.

   The *Test Call of OPC UA Method* dialog box appears. The dialog box sets up its own connection to the remote server. The connection is set up when you leave the dialog box. (See also: Test Method [page 336].)

4.2.2.2.1 Test Method

Use

The *Test Method* function can be used to call the selected OPC UA method on the remote server for test purposes.

Procedure

Calling and Testing the Method

1. Choose the *Test Method* pushbutton.
   The *Test Call of OPC UA Method* dialog box appears. The input and output parameters of the method are displayed.
2. Enter the required values for the input parameters in the *Input Parameters* screen area.
3. Choose *Call Method* to execute the method.
   After the method has been executed, a status message appears at the lower edge of the test dialog box, informing the caller if execution was successful or not. If the execution is not successful, further messages are issued in a dialog box. The status message disappears after three seconds. If the method has output parameters, the result values of the output parameters are displayed in the *Output Parameters* screen area.
4. Click on the *Glasses* icon in the *Details* column for a selected row to display the output result of the *Value* column in a separate window. The name of the parameter whose result is displayed is shown in the title row of the results dialog box.

   > **i** Note
   >
   > Use this function in particular if the returned content is very large and cannot be readily viewed in the test dialog.

The relevant value is displayed in the *Result Value for Parameter* dialog box.

Search

You have the option to browse the displayed result in the dialog box.

1. Enter a search text in the *Search* field.
2. Confirm your entry by choosing `Enter`.
   The search result is highlighted in orange. A search with * (asterisk) as a wildcard is supported.

Example

You want to browse the following text:

```
sfc: HMI_CHIP_REDS_BLL-10000001, HMI_XTSM, complete=false
```

Input in Search Field	Result
sfc	**sfc**: HMI_CHIP_REDS_BLL-10000001, HMI_XTSM, complete=false
sfc*	**sfc: HMI_CHIP_REDS_BLL-10000001, HMI_XTSM, complete=false**
***REDS**	**sfc: HMI_CHIP_REDS**_BLL-10000001, HMI_XTSM, complete=false
REDS	**sfc: HMI_CHIP_REDS_BLL-10000001, HMI_XTSM, complete=false**

> **i** Note
>
> Other combinations are possible. The overview is just provided to aid understanding of the example.

4.2.2.3 Reliable Connection Tab

Context

You use this tab to maintain the connection to the remote server. PCo checks the connection regularly if a problem with the connection has been detected.

Procedure

1. Select an OPC UA destination system and choose the *Reliable Connection* tab.
2. Use the following table to help you enter the required data:

Field	Description
Maximum Number of Retries	Maximum number of connection attempts before the connection counts as failed. The default value is 0. **i Note** There are no retries if this value is 0.
Retry Interval (Seconds)	Indicates the number of seconds before the next send attempt. The default value is 0 seconds.

4.2.3 ODBC Destination System

Use

The *ODBC destination system* is a destination system type that you use to be able to connect an ODBC database to PCo and to send notification messages to this database. Using the ODBC destination system, in a selected table in the connected database, you can create data records for specific tags to generate, for example, simple time series' for one or more tags in a history table. With each notification message that is sent to the destination system, one or more new entries are created in the table. You can use the tag values stored in the table for evaluations later.

The following ODBC databases are supported:

- MS SQL Server
- SAP HANA database

With an *ODBC destination system*, you can configure the following:

- For the configuration, you can choose any table that you have defined already in the connected database, for example, in the SAP HANA database.
- From the given table structure, you can choose the fields that you want to be filled with tag values or fixed values based on the notification messages.
- If required, you can create a table from PCo. This table has a predefined structure based on the idea of generating simple time series (tag ID, time stamp, tag value) using PCo. This table is created automatically in the connected database after saving.

As soon as the agent instance is running, the tag values are stored as table entries according to the field assignment made, with each notification message in the selected table. The history tables generated in this way are then available for all types of evaluations.

> **Note**
>
> You can also use an ODBC destination system in enhanced notification processing if you want the destination system's response to be processed. (See Integration with Third-Party Systems Using Enhanced Notification Processing (ENP) [page 68].)

Prerequisites

You can only create an ODBC destination system if you have previously configured the connection to the desired database in the ODBC Data Source Administrator:

- Set Up the Connection to the SQL Server [page 343]
- Set Up the Connection to the SAP HANA Database [page 344]

Integration

To be able to send tag values and metadata with notification messages to the database and save them persistently in a table, the following configuration steps are necessary:

1. Create a source system.
2. Create an ODBC destination system and configure data records for the desired table. (See ODBC Destination System: Configuration Tab [page 340].)
3. Create an agent instance and subscribe to tags.
4. Create a notification and generate the output values automatically on the *Output* tab by choosing, for example, the *Generate Expressions* pushbutton.
5. Select a destination system, and assign destination variables. (See Assignment of Destination Variables to Output Values (ODBC Destination) [page 345].)

More Information

For more information about ODBC, see:

Wikipedia: http://en.wikipedia.org/wiki/ODBC

Microsoft ODBC Overview (MSDN): http://msdn.microsoft.com/en-us/library/windows/desktop/ms710252(v=VS.85).aspx

4.2.3.1 ODBC Destination System: Configuration Tab

Use

On this tab, you set up the connection to the database and create the table in which you want the values of the subscribed tags to be stored later. (See also: ODBC Destination System [page 338].) The following database types can be used:

- MS SQL server
- SAP HANA database

Prerequisites

You have created and configured the *data source* that you want to use for your database in the *ODBC Data Source Administrator*. (See Setting Up the Connection to the SQL Server [page 343] and Setting Up the Connection to the SAP HANA Database [page 344].)

Procedure

Create ODBC Destination System

1. Choose the *Add Destination System* pushbutton and then choose the destination system type *ODBC destination system*.
   The *Configuration* tab appears.
2. You establish the connection to the desired database in the *ODBC data source* screen area.

Field	Description
Data Source	Choose the ODBC data source that you created previously in the *ODBC Data Source Administrator* for your database.
User Name and *Password*	Enter the logon data for the ODBC data source that you created previously in the *ODBC Data Source Administrator* for your database.

3. Choose the *Test Connection* pushbutton to check if a connection has been established.
4. If you select the *Raise Exception on Write or Connection Errors* checkbox, the ODBC destination system generates an exception if an error occurs when writing to the database.

> **Example**
>
> Here are some examples of errors that can occur when writing to the database:
>
> - The connection to the database cannot be established in the meantime.
> - The data record to be written violates a constraint defined in the database table, for example, the prohibition of duplicate values for primary keys.

The generated exception can be used in the following two ways:

- If you have assigned the ODBC destination system directly to a notification, an exception triggers the redelivery of the notification message. This means that the rewriting of the data record in a defined retry interval and with a defined number of attempts is triggered again. You make these settings on the *Message Delivery* tab in the notification.
- If you call the ODBC destination system from a multiple call destination system, you can select the *Exception Handling* checkbox in the corresponding step of the multiple call destination system. This enables you to react in a controlled manner to the occurrence of a database write error in your multiple call destination system.

Create Table

You can create a table in PCo with a predefined structure. This table is created automatically in the connected database after saving. On creation, this table already contains a data record that you need to configure. The data record consists of the following four table fields. In the database, these table fields form the columns of the history table that are then filled with values when the agent instance is running.

- *IID*: Tag ID
- *IQUALITY*: Quality of the tag value
- *ITIMESTAMP*: Time stamp
- *IVALUE*: Tag value

1. To create a new table, choose the *Create Table* pushbutton in the *Definition of Table Records* screen area. The *Create Table* dialog box appears.
2. Enter the name of the table.
   The table name is structured as follows:
   <schema>.]<history_table>]

   > **Example**
   >
   > **PCO.**HISTORY

   Entering the database schema is optional. If you do not enter anything, PCo automatically uses the user name that you use to log on to the database as the schema.

   > **Example**
   >
   > User name **`Mustermann`** and table name **`TAG_History`** lead to `MUSTERMANN.TAG_HISTORY`.

   The system then displays a table with a data record. The names of the four table fields are hard-coded and cannot be changed.
3. Define the settings for the data record (see *Add Record*).

Add Record

If you have already created a table, you can choose this table and add as many records to it as you want.

1. Choose the *Add Record* pushbutton to select the table you want and to add a new data record.
   The *Choose History Table* dialog box then appears.
2. If necessary, enter **`PCO`** in the *Filter* field, if a great many tables are displayed for selection. The system then only displays the PCo-relevant tables.
3. Select the table you want.

The system now displays a new data record in the form of a predefined assignment table. The following example shows the structure of a data record:

Column	Description
Required	Indicates that an entry is necessary for this table field. In this case, an asterisk (*) is displayed. **i Note** An asterisk (*) is always displayed in the following cases: ○ SQL type of the table field is INTEGER ○ SQL type of the table field is TINYINT ○ You have selected the *Not Null* column for a table field for a user-defined table in the HANA DB.
Name	In the first row of a data record, the name of the history table for which you want to create a data record is specified. In the following rows, the table fields of the table are displayed, for example, ITIMESTAMP.
Type	Specifies the SQL type of the table field. The ID, for example, always has the type INTEGER.
Value	Here you specify a value for the table field. If it is a variable, the following applies: The variable must start with a letter. The variable must only consist of letters, digits, and underscores, for example, `VAL1`.
Is Variable	By selecting this checkbox, you specify that the entry in this row is to be a variable. If you do not select the checkbox, the table field is treated as a constant.

4. Enter a value for each table field. Specify for each table field if it is a variable or a constant by selecting the *Is Variable* checkbox for a variable.

Example

Required	Name	Type	Value	Is Variable
	PCO.HISTORY	*Record*	Pushbutton for deleting the entire data record is displayed.	

Required	Name	Type	Value	Is Variable
*	*IID*	*INTEGER*	Enter **1**, for example.	No selection required because it is a constant.
*	*IQUALITY*	*TINYINT*	Enter **1**, for example.	No selection required because it is a constant.
	ITIMESTAMP	*TIMESTAMP*	Enter **TS1**, for example.	Set selection for variable.
	IVALUE	*VARCHAR*	Enter **VAL1**, for example.	Set selection for variable.

The entries are a prerequisite for PCo being able to assign the subscribed tags to the variables, and for the tag values, which are sent using the notification messages, being written later to the history table on the database.

4.2.3.1.1 Setting Up the Connection to the SQL Server

Use

If you want to use an *MS SQL Server* as an ODBC destination system, you first need to create and configure the ODBC connection to this server using the *ODBC Data Source Administrator* before you can create the ODBC destination system in the *PCo Management Console*. To do so, you need to create a Data Source Name (DSN) for the SQL server in the *ODBC Data Source Administrator*.

Procedure

1. You can start the *ODBC Data Source Administrator* from the start menu under *Run* > *odbcad32.exe*.

> **i** Note
>
> If you are using a 64-bit version of *Windows*, you must ensure that you are using the *ODBC Data Source Administrator* in the **32-bit version**. A 32-bit version and a 64-bit version of the *ODBC Data Source Administrator* are available in the 64-bit version of *Windows7*. In this Windows version, you start the 64-bit version of the *ODBC Data Source Administrator* from the start menu by choosing Run. The *System Data Source Name* entries that are created using the 64-bit version are not taken into account by PCo, however.
>
> The valid 32-bit version is located in the subfolder SysWOW64 in the Windows directory; for example, under **C:\Windows\SysWOW64\odbcad32.exe**.

> If you create *User Data Source Name* entries instead of *System Data Source Name* entries, you can also use the 64-bit version of the *ODBC Data Source Administrator*.

After the `odbcad32.exe` is started, the *ODBC Data Source Administrator* dialog box is displayed.

2. In the *ODBC Data Source Administrator*, click the *System DSN tab*.
3. Choose the *Add* pushbutton.
   The *Create New Data Source* dialog box appears.
4. Choose the entry `SQL Server` from the list of drivers. Then choose *Finish*.
   The *Create a New Data Source to SQL Server* dialog box is displayed.
5. Enter the following data:
   - *Name*
     Here you can enter a name of your choice for the SQL server that is then used as the ODBC data source. This name is then displayed later in PCo on the *Configuration* tab of the ODBC destination system.
   - *Description*
     You can enter a short description here.
   - *Server*
     You enter servers in the *Server* field, for example, **`vmwxxxx`**.
6. Choose *Next*.
   A dialog box appears in which you enter the authentication data.
7. Enter the logon data, if necessary. Then choose the *Client Configuration* pushbutton and enter the number of the port in the next dialog box and choose *Next*.
   The dialog box appears in which you can specify the default database.
8. Select the option *Change Default Database to:* and choose the **`PCODB`** entry. Choose *Next*.
9. In the following dialog box, you can apply the settings. Then choose *Finish*.
   The *ODBC Microsoft Server Set Up* dialog box appears.
10. Choose the *Test Connection* pushbutton.
    If the connection can be established, a success message is issued.
11. Confirm by choosing *OK*.
    The SQL server is then displayed on the *System DSN* tab and can be selected on the *Configuration* tab in the *PCo Management Console*.

4.2.3.1.2 Setting Up the Connection to the SAP HANA Database

Install SAP HANA Studio

1. Install the SAP HANA Studio.
   For more information, see http://help.sap.com/saphelp_hana/sap_hana_studio_installation_update_guide_en.pdf.
2. Install the *SAP HANA Client* for *Microsoft Excel*.
   For more information, see http://help.sap.com/hana/SAP_HANA_Client_Installation_Update_Guide_en.pdf.

ODBC Data Source Administrator

To configure the 32-bit ODBC data source in the *ODBC Data Source Administrator*, proceed as follows:

1. Navigate to the **`Windows/SysWOW64`** directory and start the tool `odbcad32.exe`.
   The *ODBC Data Source Administrator* appears.
2. In the *ODBC Data Source Administrator*, choose the *System DSN* tab and click the *Add* pushbutton.
   The dialog box for selecting the driver appears.
3. Choose the entry **SAP HANA for MS Excel** as the driver and configure the data source to be able to connect to the HANA server.
4. Confirm by choosing *OK*.
   The SQL server is then displayed on the *System DSN* tab and can be selected on the *Configuration* tab in the *PCo Management Console*.

4.2.3.2 Assignment of Destination System Variables (ODBC Destination System)

Prerequisites

You have created a notification and automatically generated the output values on the *Output* tab by choosing the *Generate Expressions* pushbutton.

Context

In this processing step, you need to assign the output values of the notification to the variables of the ODBC destination system that you created on the *Configuration* tab. (See: ODBC Destination System: Configuration Tab [page 340]. This is the prerequisite for the connected database table being filled with tag values as soon as the agent instance is running.

Procedure

1. Click the *Add Destination System* pushbutton on the *Destinations* tab.

   The *Add Destination System* dialog box appears.
2. In the *Destination System* field, select the ODBC destination system that you created previously.
3. In the *Name* field, enter a name of your choice for the destination system and choose *OK*.

   The system now displays the destination system in the *Destinations* screen area.

4. Expand the entry for the destination system and click on *Output Destination Mapping*.

   The assignment table containing the destination system variables of the ODBC destination system opens on the right. The table consists of four columns with the following meaning:

Column	Meaning
Destination System Variable	The name of the variable that you created previously on the *Configuration* tab of the ODBC destination system is displayed here. (See: ODBC Destination System: Configuration Tab [page 340].)
SQL Type	The SQL type of the destination system variable is displayed here, for example, NVARCHAR.
Notification Output	For each variable, choose an appropriate output value from the dropdown box.
Attribute	For each variable, choose an appropriate attribute from the dropdown box. You can choose from the following four attributes: ○ *Time Stamp* The time stamp of the last change to the output value ○ *Quality* The quality of the output value, for example, *Good* or *Bad* ○ *Value* The actual value of the output value ○ *Output Name* The name of the output value ○ *Source* The source of the tag on which the output value is based. This attribute is taken from the subscription item, provided the expression for the output value contains exactly one subscription item. PCo checks if the assignment is correct and issues a corresponding error message if necessary.

5. Assign each destination system variable a value in the *Notification Output* column and an *Attribute*.

   After you have made the assignment, the table looks like this, for example:

Destination System Variable	SQL Type	Notification Output	Attribute
TS1	TIMESTAMP	Int1	Timestamp
VAL1	NVARCHAR	Int1	Value

Destination System Variable	SQL Type	Notification Output	Attribute
TS2	TIMESTAMP	Int2	Timestamp
VAL2	NVARCHAR	Int2	Value

6. Save the assignments.

Results

You can now start your agent instance in the *PCo Management Console*. PCo transfers the current tag values to the database table. You can now check, for example, in the *SAP HANA Studio*, if the new data records have arrived in your history table.

4.2.4 RFC Destination System

Use

If you want to connect a Business Suite system, you need to create an *RFC destination system*. The following types of Business Suite systems are supported:

- SAP NetWeaver
- SAP EWM
- SAP ODA (component SAP ODA in ERP)

Procedure

You make the settings for the RFC destination on the following tabs:

- RFC Client Settings Tab [page 348]
- Security Settings Tab [page 351]

More Information

Defining an RFC Destination [page 91]

4.2.4.1 RFC Client Settings Tab

Procedure

1. To configure RFC client settings, on the *Plant Connectivity Management Console* screen, select the RFC destination system and click *RFC Client Settings*.
2. For the information you need to enter for the **connection settings**, use the following table:

Field	Description
Type	Here you enter the Business Suite system with which you want to use PCo: ○ *SAP NW* Specify *SAP NW* if you want to connect to any Business Suite system. ○ *SAP ODA* Specify *SAP ODA* if you want to use the component *SAP ODA* in the ERP system and want to replace the *ODA Connector* with PCo. ○ *SAP EWM* Specify *SAP EWM* if you want to connect to an EWM system.
System ID	ID that identifies the connected system **i Note** You can find the system ID in *SAP Logon*. In *SAP Logon*, in the context menu for the system entry, choose *Properties* > *Connection tab* and copy the value from the *System ID* field.
Client	Client of the Business Suite system to be connected.
Language	The language with which the user is to log on to the Business Suite system. This entry is only required if there is no default logon language stored in the Business Suite system. Enter the ISO code for the language to be used, for example, **EN** for English or **DE** for German.

Field	Description
SAP Router	The SAP Router makes it possible to connect through a firewall. Enter the SAP router parameters in the following format: `/H/hostname//S/portnumber/H/`. The entry here is optional.
Max. Number of Connections	In the *Max. Number of Connections* field, you can specify the maximum possible number of RFC connections that can be open simultaneously at any given time. If the maximum number of parallel RFC connections has been reached, every further client activity that makes an additional connection necessary triggers an *RfcResourceException*.

3. In the **Load Balancing** screen area, choose *Yes* to activate load balancing. With load balancing, you can have the SAP Message Server assign an application server. In this case, the application server is determined according to a load balancing procedure. The following fields are ready for input in this case:

Field	Description
Message Server	Name of the message server of the Business Suite system. (See also: Message Server.) **i Note** You can find the name of the message server in *SAP Logon*, in the context menu of the system entry, under *Properties* > *Connection tab*. You can copy the name from the *Message Server* field, for example, **`q63main.xxx.yyy.zzz`**.
Logon Group	Enter the logon group here. **i Note** You can find the logon group in *SAP Logon*, in the context menu of the system entry, under *Properties* > *Connection tab*. You can copy the name of the logon group from the *Group/Server* field, for example, **`PUBLIC`**.

4. If you **do not want to work with load balancing**, choose *No*. The following fields are then ready for input:

Field	Description
Application Server	Enter the name of the server of the Business Suite system to which you want to connect, for example, the server on which the ERP system runs. **i Note** You can find the server name in *SAP Logon* by calling the properties of the relevant system entry in the context menu and choosing the *Connection* tab. You can copy the server name from the *Message Server* field, for example, `q63main.xxx.yyy.zzz`.
System Number	Business Suite system ID used for direct connection. **i Note** The system number corresponds to the instance number in *SAP Logon*. In *SAP Logon*, in the context menu for the system entry, choose *Properties* > *Connection tab* and copy the value from the *Instance Number* field.

5. You can use the **tracing settings** to trace the RFC communication with this agent instance in a file. The tracing file is stored in the current working directory of the PCo agent instance.

Field	Description
Enabled	If you select this checkbox, the *Trace* function is enabled.
Level	States the trace level of the *.Net Connector (NCo)*. The trace level is the degree of detail in which steps performed during message processing are traced by the system. The following settings are possible: ○ Level 1 With this setting, only the *Remote Function Calls* are traced. ○ Level 2 With this setting, the calls from public *API Methods* are also traced. ○ Level 3 With this setting, the calls from internal *API Methods* are traced in addition to the calls mentioned above. ○ Level 4 With this setting, the *HEX Dumps* are traced for the RFC log in addition to all calls previously mentioned.

6. Choose the *Check Connection* pushbutton to create a temporary connection to the RFC server.
7. Choose the *Simulate System Call* pushbutton to check whether the selected type (for example, SAP EWM) matches the system you have set.

> **i Note**
>
> If necessary, you need to make additional settings on the *Security Settings* tab before testing the connection.

4.2.4.2 Security Settings Tab

Procedure

1. Specify which *logon procedure* you want to use:
   - *User Name and Password*
   - *Secure Network Communication (SNC)*
2. If you have selected the *user name and password logon procedure*, enter a *user name* and *password*.
3. If you have selected the *Secure Network Communication (SNC) logon procedure*, make the **SNC settings**.

   With this procedure, you use the **Secure Network Communications (SNC)** component for communication between the *PCo Management Console* and the connected Business Suite system that you want to connect to PCo. In this case, you can use **Single Sign-On**.

Field	Description
User SNC Name	Specifies the user name that is used for SNC. You also need to enter this name in the Business Suite system in transaction `SM59` in the *Partner* field, for example, **`p:CN=D0XXXXX, O=SAP-AG, C=DE`**. **i Note** You can find this user name in *User Maintenance* (transaction SU01) of the SAP system on the *SNC* tab, or alternatively in the SAP system menu under *System* > *Status*.

Field	Description
Server SNC Name	You enter the server name of the connected SAP system here, for example, `p:CN=UI3, O=SAP-AG, C=DE`. **i Note** You can find the SNC server name in *SAP Logon* in the context menu of the relevant system entry under *Properties* > *Network tab* > in the *SNC Name* field. **⚠ Caution** You do not have to enter the *SNC server name* if you are using **load balancing**.
Quality of Protection	Specifies the quality of protection (QoP) of the data transfer: ○ **1** *Only Secure Authentication (Digital Signature)* ○ **2** *Protection of the Integrity of the Data (Digital Signature and Encryption)* ○ **3** *Confidentiality of the Data (Digital Signature, Encryption, and User Authentication)* ○ **8** *Default (Default Value Defined by the ERP System)*, profile parameter **snc/data_protection/use**. ○ **9** *Maximum Available (Maximum Value Supported by the Security Product)*, profile parameter **snc/data_protection/max**. If the value is below the minimum required value (profile parameter **snc/data_protection/min**), it is increased automatically. If the value is above the maximum available value (profile parameter **snc/data_protection/max** or maximum value of the external security product), the communication is terminated.
Library	Here you enter the full path and the name of the security library of a non-SAP product. This library is then used for the SNC communication. If you use **Single Sign-On (SSO)**, you do not need to enter anything here. You can use the path C:\Program\Files\SECUDE\Office\Security\secude\dll to access the SSO library. If, in conjunction with SSO, the path differs from the standard path, enter this here.

For more information, see SNC Settings [page 92] and the *SAP NetWeaver* documentation under *Secure Network Communications (SNC)*.

4. Choose the *Check Connection* pushbutton to create a temporary connection to the RFC server.
5. Choose the *Simulate System Call* pushbutton to check whether the selected type (for example, *SAP EWM*) matches the system you have set.

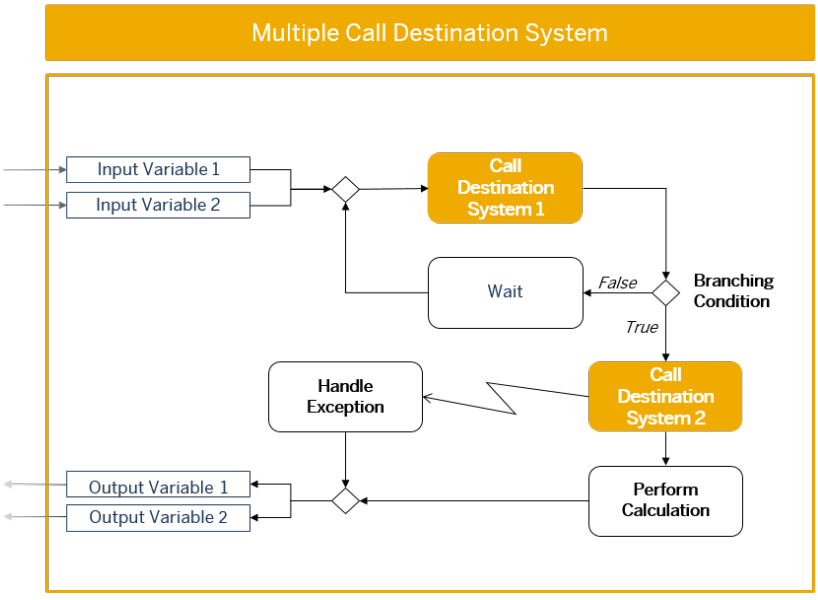
4.2.5 Multiple Call Destination System

Use

A multiple call destination system (MCD) is a type of destination system that you can use to call other destination systems in a configurable sequence from a notification.

In addition to this, an MCD offers other functions that can be combined with each other. The following graphic gives an overview of the main functions:

Main Functions of a Multiple Call Destination System



- Calling other destination systems in a configurable sequence
  You can define the sequence in which you want individual destination systems to be called. You can configure linear call sequences as well as sequences with branching conditions or loops. You can also configure an MCD so that processing is suspended in a specific step and resumed at a later point in time, for example, when a production machine reports that it has processed an order.
  You can call the following types of destination systems from a multiple call destination system:
  - Multiple call destination system
  - ODBC destination system

- OPC UA destination system
- Query destination system
- Web service destination system
- Universal Web service destination system
- Simulation destination system
- MQTT destination system

- Usage of built-in functions
  The multiple call destination system offers a series of built-in functions that you can use like destination system calls in your process logic. You can choose from the following options:
  - Change variable values
  - Read and write array element values
  - Trigger an exception
  - Wait (for a specified duration)
  - Perform calculations in a specific step

  For more information, see Predefined Functions in the Multiple Call Destination System [page 368].
- Calculations or conversions of variable values
  The destination system can calculate or convert variable values that are returned from a destination system or that are used to call from another destination system. A multiple call destination system thereby provides options for converting output variables of a destination system that otherwise only exist as calculated variables in the destination system type Web service destination system.
  However, you can also use a multiple call destination system on its own for variable calculation or conversion, similar to a function or static method in a programming language. In this case, you do not need to configure any destination system calls in a multiple call destination system.

  > **Example**
  >
  > You define a multiple call destination system to generate a character string from an SFC and a plant. You can reuse this multiple call destination system as a module in other multiple call destination systems.

- Concatenation of several multiple call destination systems
  You can concatenate several multiple call destination systems for the modularization of functions.

  > **Example**
  >
  > MCD1 and MCD2 both call MCD3 that provides commonly used functions, such as a frequently-used sequence of Web service calls or a character string operation.

- Controlled and case-by-case handling of exceptions that can occur when destination systems are called
- Buffering of values
  If you flag local variables of a multiple call destination system as permanent, these variables retain their value for consecutive destination system calls that take place using the same agent instance. In this way, you can collect data in a multiple call destination system, or compare values from previous calls with the values of the current call.

Integration

Like every other PCo destination system type, a multiple call destination system is called from a **notification scenario**. This could be a **tag-based notification** or a **method notification**.

A multiple call destination system can have input or output variables as an interface to the notification or to other multiple call destination systems. In a tag-based notification, the input variables of the multiple call destination system are filled by the output expressions of the notification.

In a method notification, these are the input parameters of the notification. In a method notification, you can assign the output variables of the multiple call destination system to the output parameters of the OPC UA method that is connected to the notification. (See also: Method Notification [page 55].)

In combination with the query destination system [page 375], the multiple call destination system offers a more flexible alternative to destination system calls with response processing (see: Destination System Calls with Response Processing [page 81]). The benefits are as follows:

- More than three destination system calls are possible
- Loops and branching conditions
- More flexible handling of exceptions that can occur for destination system calls

See also

Example of Using an MCD [page 356]

Variables of a Multiple Call Destination System [page 357]

Configuring a Multiple Call Destination System [page 360]

Branching Conditions [page 363]

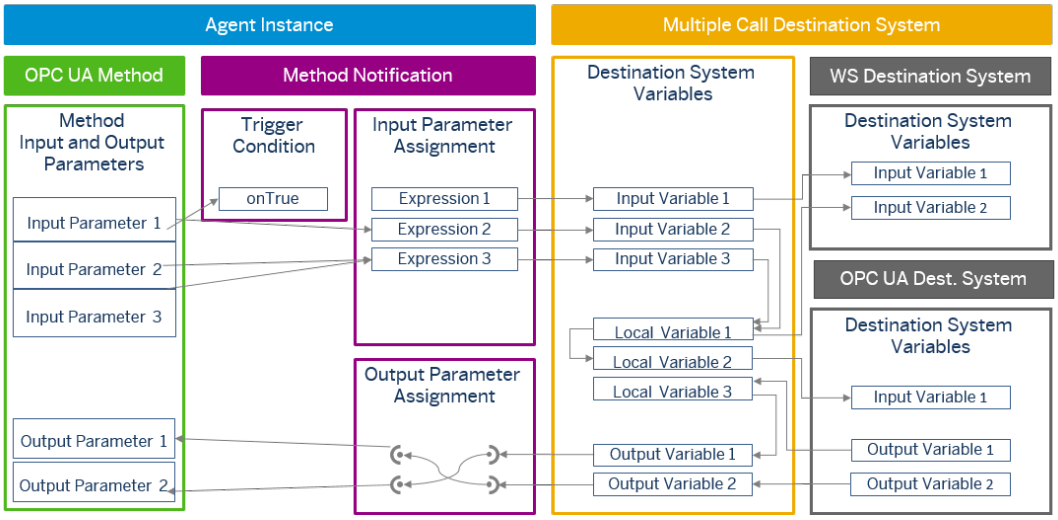
Activating Exception Handling [page 367]

Testing a Multiple Call Destination System [page 374]

4.2.5.1 Example of Using an MCD

The following graphic shows an example of using a multiple call destination system in a method notification:

Multiple Call Destination System (Example)



The multiple call destination system shown offers three input variables and two output variables. In the multiple call destination system, a sequence of destination system calls is configured that consists of a Web service destination system and an OPC UA destination system.

Furthermore, the graphic shows that the multiple call destination system has local variables that are used during notification processing to convert or temporarily store values.

> **i Note**
>
> In earlier PCo versions, local variables were called temporary variables.

The Web service and OPC UA destination systems shown in the example can be called with current values from the input variables, output variables, or local variables of the multiple call destination system. After a destination system (for example, an OPC UA destination system) has been called successfully, the output variables of the called destination system can be assigned to the variables of the multiple call destination system.

For more information about the significance and calculation of variables, see Variables of a Multiple Call Destination System [page 357].

4.2.5.2 Variables of a Multiple Call Destination System

Categories of Variables

You can define the variables to be used for the multiple call destination system on the *Variables* tab. A multiple call destination system offers three categories of variables:

- Input variables
- Output variables
- Local Variables

Input variables and output variables define the interface of the multiple call destination system to a notification or to another multiple call destination system that calls the multiple call destination system.

In a tag-based notification, the **input variables** are filled by the output expressions of the notification.

In a method notification, the **input variables** are filled by the input parameters of the notification. In the method notification, you can assign the **output variables** of the multiple call destination system to the output parameters of the method that is connected to the notification.

Local variables can only be seen and used within a multiple call destination system. They can be used for converting variable values for destination system calls, for storing interim results, or for defining fixed values within a multiple call destination system.

> **i** Note
>
> What all variable categories have in common is that the variable values and the update count are initialized and recalculated here for each call of the multiple call destination system. It is not possible to define variables that keep their value during several calls of a multiple call destination system.
>
> You can also flag local variables as **permanent**. This means that these variables retain their value for consecutive destination system calls that take place using the same agent instance.

Value Assignments

The type of assignment of variable values depends on the variable category:

- Input variables
  Input variables get an initial value at the start of the call of a multiple call destination system. This value results from the assignment of output expressions or input parameters of the notification to the input variables. Moreover, input variables can be updated by the output variables of destination systems that are called by the multiple call destination system.
- Local and Output Variables
  In contrast to input variables, values of output variables and local variables are always calculated or are set by the output of destination system calls. Calculations are formula expressions in which you can use fixed values and other variables of the multiple call destination system. You define the calculation expressions on the **Variables** tab.
  If you select the *Permanent* checkbox, you can specify that a local variable becomes a **permanent local variable**. The prerequisite for this is that you have selected the *Exclusive Lock* checkbox on the *General Settings* tab. For the **permanent local variables**, you define an **initial value** that is assigned to and retained

by these variables when the agent instance is started. Other variables of the multiple call destination system cannot be used to calculate permanent local variables.
Initial values can be constant values, such as `0`, `1`,`""`, `true`, or `false` , but also calculations without using variables, such as `datenow`, `emptymap`, `arrayCreate("System.String", 3)` , or `structNew()` .
Permanent local variables can only change their value through the output of destination system calls, for example, through the return values of a universal Web service destination system. This includes the call of the built-in functions, such as `@Calculation` or `@Increment`.

> **i Note**
>
> Using relevant error messages, the system indicates if variable values would remain undefined due to missing calculations or value assignments.

If the *Make Permanent Local Variables Persistent* checkbox on the *General Settings* tab of the multiple call destination system is selected, the variable values are also retained after an agent instance is restarted. If necessary, they can be reset to their initial values for each agent instance.

> **i Note**
>
> Persistent storage of permanent variables is not possible if you call the multiple call destination system from the destination system of enhanced notification processing (ENP). In this case, selecting the *Make Permanent Local Variables Persistent* doesn't have any function.

Calculation Expressions and Calculations of Values

You define the calculations that are to be made using the expression editor (see Expression Editor [page 545]). In the calculation expressions, you can use multiple call destination system variables, the update count, the functions of the expression editor, and fixed values.

If a calculation expression only consists of fixed values or functions that are not dependent on other variables, the calculation is made at the start of the call of a multiple call destination system.

> **Example**
>
> Examples of calculation expressions that are executed at the start of the call:
>
> - `"1000"`
> - `datenow`

If the calculation of a variable depends on other variables, a calculation is always made as soon as the value of a used variable changes and when all variables used in an expression already have a value. The result of using variables in calculation expressions is a specific calculation sequence. (See: Example of Calculating Variable Values [page 359].)

> **i Note**
>
> Alternatively or in addition to the implicit recalculation of variables mentioned above, you can explicitly trigger the recalculation of a variable value by calling the predefined function `@Calculation` in one step of the multiple call destination system. (See also: Predefined Functions in the Multiple Call Destination System [page 368].)

Update Count

Each variable of a multiple call destination system has, in addition to its actual value, an update count. The update count increases with each calculation or value assignment of a variable. The update count of a variable can be accessed in calculation expressions with the suffix `.updateCount`.

> **Example**
>
> The update count of a variable `inVar1` is addressed in a calculation expression as `'inVar1.updateCount'`.

You can use update counts, for example, in branching conditions [page 363] to define a termination condition for loops.

Data Types of Variables

For each variable of a multiple call destination system you can define a data type that you select corresponding to the variables of the called destination systems or to the output parameters of the method from which the multiple call destination system is called. Using local variables allows you to convert data types.

4.2.5.2.1 Example of Calculating Variable Values

A multiple call destination system has the input variable `inSfc`, the output variables `outNextStep` and `outNextStepRevision` as well as the local variables `locSite` , `locSfcRef` , and `locNextStep`.

Enter the following on the *Variables* tab page:

Input Variables Table

Variable Name	Data Type
`inSfc`	System.String

Output Variables Table

Variable Name	Data Type	Calculation or Initial Value	Comment
`outNextStep`	System.String	`stringsplit('locNextStep', ",", 1)`	Input of calculation expression
`outNextStepRevision`	System.String	`stringsplit('locNextStep', ",", 2)`	Input of calculation expression

Local Variables Table

Variable Name	Data Type	Calculation or Initial Value	Comment
`locSite`	System.String	`"1000"`	Input of fixed value `"1000"`
`locSfcRef`	System.String	`"SFCBO:" & 'locSite' & "," & 'inSfc'`	Input of calculation expression The variable is needed to call a Web service.
`locNextStep`	System.String	No input required	This variable receives the result of the Web service call.

In this example, the system calculates at the start of the call of the multiple call destination system the variable `locSite` (because it is a fixed value expression), as well as `locSfcRef` from the variable `inSfc`. The variable `locNextStep` gets its value from the output of the called Web service. The variables `outNextStep` and `outNextStepRevision` get their values as soon as the variable `locNextStep` gets its value from the Web service result.

4.2.5.3 Configuring a Multiple Call Destination System

Procedure

The configuration procedure varies depending on the usage of the multiple call destination system. Two typical procedures are described below:

- Configuration of a **linear call sequence** of destination systems
  With this configuration, the aim is to call other destination systems in a linear call sequence from a notification. (See: Configuring a Linear Call Sequence of Destination Systems [page 361].)
- Configuration of a multiple call destination system using **suspend and resume**
  In automation tasks, it is often the case that a production machine receives a task from PCo, for example, when an OPC UA method is called. However, processing this task might take a long (for example, more than five seconds) and unforeseeable amount of time. The production machine later confirms processing by calling an OPC UA method or a Web service operation of the PCo server. With the **suspend and resume** configuration option, you can configure the call for starting the task and configure further processing after confirmation by the production machine in just one multiple call destination system. This improves the readability of the configuration compared to modeling the same process in several multiple call destination systems, and enables continued usage of variables that were available before the order started. See also: Configuring Suspend and Resume [page 362]

For more information about advanced procedures, see Branching Conditions [page 363] and Activating Exception Handling [page 367].

More Information

Multiple Call Destination System [page 353]

Variables of a Multiple Call Destination System [page 357]

Branching Conditions [page 363]

Activating Exception Handling [page 367]

4.2.5.3.1 Configuring a Linear Call Sequence of Destination Systems

Procedure

1. Create a new destination system of the type *multiple call destination system*.

   The *Destination System Calls* tab appears.
2. To add a new **step** in the *sequence of destination system calls* table on the *Destination System Calls* tab, choose  . For more information, see Steps [page 363].
3. In the newly created step, in the *Destination System* field, choose the **destination system** that you want to be called later.

   PCo displays the input and output variables of the selected destination system in the *Assignment of Variables* table.
4. In the *Description* field, you can add a **description** for each individual step.
5. Assign the variables of the multiple call destination system to the input and output variables of the destination system. You can do this in two ways:

   - In the lower screen area, in the **Assignment of Variables** table, choose  to create a new multiple call destination system variable.
     The new multiple call destination system variable then receives the appropriate data type for the variable of the destination system. The system proposes the name of the destination system variable as the name of the multiple call destination system variable. You can subsequently change the name and data type of the multiple call destination system variable on the *Variables* tab.
   - Choose the *Variables* tab.
     You create an input, output, or local multiple call destination system variable here.
     Go back to the *Destination System Calls* tab and, in the *Variable Name* column, select, from the dropdown list, the multiple call destination system variable that you created earlier. Make sure that the data types of destination system and multiple call destination system variables are the same or can at least be converted into each other. If the data types are different, a data type conversion takes place at the runtime of the multiple call destination system, which can fail, depending on the current value of a variable.

> **i Note**
>
> You need to supply all input variables of a destination system with variables of the multiple call destination system. On the other hand, assigning destination system output variables to multiple call destination system variables is optional.

Next Steps

Activating Exception Handling [page 367]

Branching Conditions [page 363]

4.2.5.3.2 Configuring Suspend and Resume

Context

To resume a process after it has been suspended, make the following settings for a multiple call destination system:

Procedure

1. On the *Destination System Calls* tab, select the *Suspend and Resume* checkbox in the step you want.
2. If, after resuming the process, you want to evaluate information, for example, that was given by the caller of a method that triggers the resume, you define resume variables. To do so, select the step you want and choose *Edit Resume Variables*.
    a. You can add a new resume variable manually by choosing *Add New Variable* in the *Edit Resume Variables* dialog box. Enter a name and a data type for the variable.
    b. If you have already defined a notification that is called when the process is resumed, choose the agent instance and the relevant notification in the *Edit Resume Variables* dialog box. Then choose *Create Variables from Notification Output Expressions*.
    c. The system then proposes resume variables that correspond to the output expressions of the notification and the appropriate data type. Only transfer the proposed resume variables that are required for further processing.
3. Choose OK to close the *Edit Resume Variables* dialog box.
4. Switch to the notification from which you want the multiple call destination system to be called when execution of the process is resumed.
5. Map the multiple call destination system to the notification on the *Destination* tab.
6. Choose *Output Destination Mapping* in the notification and choose the call mode *Resume*.
7. In the *Output Expression Containing the Resume Handle* field, choose the output expression that contains the value of the resume handle at runtime.
8. In the *Destination System Call Step to Resume* field, select the step of the multiple call destination system where execution of the process was suspended and for which it should be resumed.
9. If the selected step has resume variables, these variables appear in the *Assignment of Destination System Input Variables to Output Values* table. Map the resume variables to the output variables of the notification here.

4.2.5.3.3 Steps

A step of a multiple call destination system represents a processing unit. This unit can contain optional actions that are processed in the following sequence:

1. Call a destination system
2. Handle an exception that occurred when the destination system was called (see also: Activating Exception Handling [page 367]).
3. Evaluate a branching condition after destination system call (see also: Branching Conditions [page 363]).
4. Suspend and resume (see also: Configuring Suspend and Resume [page 362]).
5. Evaluate a branching condition after processing is resumed (see also: Branching Conditions [page 363]).

Steps are numbered in ascending order. The steps are executed according to their step number, starting with the lowest step number. The system proposes step numbers in increments of ten. You can change the step number manually to change the sequence of destination system calls.

In this case, it is the step number that is crucial for execution and not the sequence in which the steps are displayed in the table.

> **i Note**
>
> If you choose  , the table is sorted by step number and therefore by the call sequence.

In addition to the step numbers you can mark each **step** as a **jump target**. A **jump target** is a user-defined label that can be used in branching conditions. It helps simplify configuration of a multiple call destination system and improves readability of the configuration. If you enter jump targets and don't use them in branching conditions, they are used only for visual structuring purposes and don't have any function.

To assign a jump target, enter a character string in the *Jump Target* column that is unique in the multiple call destination system, for example, `StartPrint` or `ErrorHandling`.

> **i Note**
>
> The jump target `Exit` is predefined and cannot be configured.

4.2.5.3.4 Branching Conditions

Use

You can use branching conditions to change the otherwise linear processing of steps in a multiple call destination system. A branching condition is a calculated expression whose result refers to the step at which processing is to be continued.

The result of the branching condition might be a step number or the jump target of the next step. It is preferable to use jump targets in branching conditions rather than step numbers for the following reasons:

- Jump targets remain stable when steps are renumbered. If you use step numbers in branching conditions, you need to manually adjust the branching conditions when steps are renumbered.

- The system can identify non-existent jump targets in branching condition expressions and make you aware of configuration errors. It is not possible to check for step numbers that don't exist.
- Jump targets and their use in branching conditions can improve the readability of the configuration.

You can define two branching conditions in one step:

- *Branching Condition (After Destination System Call)*
  This branching condition is checked directly after the destination system is called but before processing is suspended, if the *Suspend and Resume* checkbox is selected. In a branching condition (after destination system call) you usually check the direct response of a destination system call.
- *Branching Condition (After Resume)*
  This branching condition can only be used in a step for which the *Suspend and Resume* checkbox is selected. This branching condition is checked after processing of the multiple call destination system has resumed. Usually, in a branching condition (after resume), you check the content of resume variables to control the further processing flow. See also: Configuring Suspend and Resume [page 362].

If a destination system call and a branching condition are defined in one step, the branching condition is checked after the destination system call.

> **i Note**
>
> A branching condition works in a similar way to the command "GOTO n" in the programming language BASIC, where n is the calculation result of the branching condition. After the calculation of the branching condition, the system goes to the step number that is greater than or equal to the result n in the branching condition. If n is greater than the highest step number, the call chain ends at this point. The result of a branching condition can also be a jump target. The jump target **Exit** is predefined and ends the call sequence.

You define a branching condition by selecting a row in the *Sequence of Destination System Calls* table and choosing  or *Edit Branching Condition (After Resume)*. You can then define your branching condition in the expression editor. (See: Expression Editor [page 545].)

In the branching conditions, you can use multiple call destination system variables and their update count, as well as the functions of the expression editor and fixed values.

Alternatively, you can edit the branching condition directly in the table.

Related Information

Examples of Branching Conditions [page 365]

4.2.5.3.4.1 Examples of Branching Conditions

Example 1

You define the following sequence of destination system calls:

Step Number	Jump Target	Destination System	Branching Condition (After Destination System Call)
10		`WebService_1`	`if ('locOutWebService_1' >= 0, 20, 30)`
20		`WebService_2`	`99`
30		`WebService_3`	

The local variable `locOutWebService_1` is filled by a return value of the destination system `WebService_1`.

In this example, the system first calls the destination system *WebService_1* in step 10. After the destination system is called, the branching condition of step 10 is checked. If the value of the variable *locOutWebService_1* is greater than or equal to 0, the system jumps to step 20. The destination system *WebService_2* is called there. After the destination system `WebService_2` is called, the branching condition of step 20 is checked. A `fixed value 99` is there instead of a condition. However, since 99 is larger than the highest step number in the sequence (that is, larger than 30), the call sequence ends at this point. Therefore, the destination system `WebService_3` is not called if the value of the variable `locOutWeb-Service_1` is larger than or equal to 0.

If the value of the variable `locOutWebService_1` were smaller than 0, the system would go from step 10 to step 30 and execute the destination system `WebService_3`.

Alternatively, you can configure the above example with jump targets. Jump targets are used in a similar way to variables in the expression of a branching condition, meaning that they are enclosed by single quotation marks. However, the name of the jump target must be prefixed by an exclamation mark, for example:

- '!WS2'
- '!WS3'
- '!Exit'

Step Number	Jump Target	Destination System	Branching Condition (After Destination System Call)
10		`WebService_1`	`if ('locOutWebService_1' >= 0, '!WS2', '!WS3')`

Step Number	Jump Target	Destination System	Branching Condition (After Destination System Call)
20	WS2	`WebService_2`	`'!Exit'`
30	WS3	`WebService_3`	

Example 2

If you want to evaluate a branching condition without calling a destination system first, leave the destination system field blank in the corresponding step:

Step Number	Jump Target	Destination System	Branching Condition (After Destination System Call)
10			`if ('inValue' >= 0, '!A', '!B')`
20	A	`WebService_1`	
30	B	`WebService_2`	

In this example, no destination system is called in step 10; only a branching condition is checked. Depending on the current value of the variable inValue, the system branches to step 20 or 30. Note in this example that there is no further branching condition in step 20. If inValue is greater than or equal to 0, the system also executes step 30 after step 20.

Example 3

You can use branching conditions to realize loops:

Step Number	Jump Target	Destination System	Branching Condition (After Destination System Call)
10	LoopStart	`WebService_1`	`if ('locOutWebService_1.updateCount' < 10, '!LoopStart', '!WS2')`
20	WS2	`WebService_2`	

In this example, we assume again that the variable `locOutWebService_1` is filled by a return value of the destination system `WebService_1`. Each multiple call destination system variable has an update count,

the `.updateCount`. In this example, the destination system `WebService_1` is called at the start. This call fills the variable `locOutWebService_1` with a value. Every time the destination system *WebService_1* is called, the update count `locOutWebService_1.updateCount` increases by 1. Therefore, the branching condition in step 10 means that the destination system `WebService_1` **is to be called ten times** before the system continues with step 20.

Example 4

If you have selected the *Suspend and Resume* checkbox in a specific step, you can prevent execution of the process being suspended by specifying a different step as the jump target in the *Branching Condition (After Destination System Call)*. If you want to check in the *Branching Condition (After Destination System Call)* whether or not the system is to suspend processing, you can use the `suspend` function as a jump target for suspending processing. (See also: Functions in the Expression Editor [page 555].)

Example 5

Depending on the success of a destination system call, you want to suspend or terminate processing of the multiple call destination system. Let's assume that the success of a destination system call lies in the value of the boolean variable `outCallSuccess`. You can define the following *branching condition (after destination system call)*:

```
if ('outCallSuccess' == true, suspend, '!Exit')
```

In this example, processing of the multiple call destination system is suspended if the value of the variable `outCallSuccess` is true. Otherwise processing goes to the predefined jump target **Exit** and leaves the multiple call destination system.

> **i** Note
>
> Note that the system does not provide any protection against endless loops that have been configured unintentionally.

4.2.5.3.5 Activating Exception Handling

Context

You can react in a controlled way to exceptions that can occur when destination systems are called by configuring exception handling, to avoid terminating the process of the multiple call destination system and to continue working with defined values for the destination system output variables.

> **Example**
>
> You call an SAP ME Web service from a multiple call destination system to complete an SFC number. If you call the Web service with an SFC number that has already been completed, the Web service call responds with an exception. However, you want to deal with this situation in a controlled way so that the PCo agent instance does not stop in this case.

Procedure

1. In the *Sequence of Destination System Calls* table, you select the *Exception Handling* indicator in one or multiple steps.

   PCo displays the *Exception Handling for Module* dialog box.
2. Configure the exception conditions in the same way as for exception handling for enhanced notification processing. You can assign values to the destination system output variables that the variables are to take if an exception occurs. (See: Exception Handling [page 79].)
3. If you have activated exception handling in one step, two additional variables of the category exception handling variable appear in the variable assignments list for this step:
   - After the destination system call, the variable `!Exception!` contains the information as to whether an exception has occurred.
   - The variable `!ExceptionMessage!` contains the exception text.

   You can assign multiple call destination system variables to both these variables and thereby respond to the exceptions that have occurred in a branching condition, for example.

4.2.5.3.6 Predefined Functions in the Multiple Call Destination System

Definition

The multiple call destination system offers several predefined functions that you can use like a destination system call. In contrast to a destination system call, you do not need to configure a destination system for a function call.

Procedure

1. Navigate to the *Destination System Calls* tab in a multiple call destination system.
2. Create a step in the *Sequence of Destination System Calls* area.

3. In the *Destination System* field, select a function from the list.

> **i Note**
>
> Functions are at the end of the destination system list and begin with the at sign (@).

4. Then assign variables of the multiple call destination system to the input and output variables of the selected function.

List of Predefined Functions

@ArrayGetValue

This function reads the value of an array element.

Input Variables

Input Variable	Description
inArray (System.String)	Array
inIndex (System.Int32)	Index of the element to be read

Output Variable

Output Variable	Description
outValue (System.String)	Value of the array element

@ArraySetValue

This function sets the value of an array element.

Input Variables

Input Variable	Description
inArray (System.String)	Array that is to be changed
inIndex (System.Int32)	Index of the element to be written
inValue (System.String)	Value of the array element

Output Variable

Output Variable	Description
outArray (System.String)	The changed array

@Calculation

You can use this function to perform a calculation explicitly in one step and assign the result to a variable of the multiple call destination system.

Input Variable

Input Variable	Description
inExpression	A calculation expression that you formulate using the expression editor. In the calculation expression, you can use variables of the multiple call destination system and functions of the expression editor.

Output Variable

Output Variable	Description
outResult	Result of the calculation Assign the result to a variable of the multiple call destination system.

@Increment

This function increases the value of a numerical multiple call destination system variable by one.

Input Variable

Input Variable	Description
inVar (System.Int64)	Variable that contains the value before the increase

Output Variable

Output Variable	Description
outVar (System.Int64)	Variable that contains the value after the increase

@RaiseException

This function raises an exception. An exception can be used in another multiple call destination system to trigger exception handling. Or it can be used in a tag-based notification to cancel delivery of a notification message and to initiate a repeat delivery for the notification message.

Input Variable

Input Variable	Description
inExceptionMessage (System.String)	Text of the exception Note that the exception text always has the prefix `Configured exception raised:`.

@SetValue

This function sets the value of a multiple call destination system variable.

Input Variable

Input Variable	Description
inValue (System.String)	Variable that contains the value that is assigned to the variable **outAssignedVariable**.

Output Variable

Output Variable	Description
outAssignedVariable (System.String)	Variable whose value is to be changed

@Wait

Waits for a specific amount of time.

Note: The wait function blocks an execution thread for the specified time so it is not possible to stop an agent instance during this time. You can recognize that this is the case because it might take longer than usual to shut down an agent instance.

Input Variable

Input Variable	Description
inMilliseconds (System.Int32)	Wait time in milliseconds

4.2.5.4 General Settings Tab

You can make the following settings here:

General Settings

Field	Description
Log Level	You can use the log level to define with what level of detail you want the multiple call destination system to write messages to the agent instance log at the runtime of the agent instance. (See: Log Level [page 372].)
Exclusive Lock	If you select this checkbox, you define that only one thread can run through the process logic of a multiple call destination system at a specific time. The multiple call destination system is locked for other threads at this time. If you do not select the checkbox, the MCD can be used simultaneously by multiple threads.

Field	Description
Make Resume Handles Persistent	If you select this checkbox, you define that you want the resume handles to be stored and thus still be available after the agent instance is restarted. You can display the used resume handles on the *Resume Handles* tab. A resume handle is a unique ID with which a resume point can be identified within a multiple call destination system.
Make Permanent Local Variables Persistent	If you select this checkbox, you define that you want the permanent local variables to be stored permanently and thus still be available after the agent instance is restarted. If you do not select the checkbox, the values are not stored permanently. After the agent instance is restarted, the variables are reset to their initial value.
Reset pushbutton	You can use the *Reset* pushbutton to reset the values and the update counter of persisted permanent local variables that are associated with the selected agent instance. To do this, you need to stop the selected agent instance and choose the *Reset* pushbutton. After the agent instance is restarted, the permanent local variables receive their initial values defined in the configuration.

4.2.5.4.1 Log Level

Use

You can define a *log level* on the *General Settings* tab. This log level determines the level of detail in which the multiple call destination system writes messages to the agent instance log at the agent instance runtime.

Only one level is taken into consideration at the multiple call destination system and it is less detailed than the log level defined at the agent instance. This makes it possible to only write to the log the most important messages of specific, non-critical multiple call destination systems, while more critical multiple call destination systems can produce more detailed messages for error analysis and are easier to find in the log.

> **i** Note
>
> Using the log level setting at the multiple call destination system, it is not possible to write more detailed messages to the log than is set at the agent instance. The log level set here does not affect the log level that you have set for testing the configuration. (See: Testing a Multiple Call Destination System [page 374].)

Example

At the multiple call destination system, you set **`Warning`** in the *Log Level* field and use this multiple call destination system in an agent instance with the log level **`Information`**. Since the log level of the multiple call destination system is less detailed than that of the agent instance, the multiple call destination system only writes messages with event types **`Warning`**, **`Error`**, and **`Critical`** to the log during the runtime of the agent instance, while the agent instance and other processes also generate messages with the event type **`Information`**.

4.2.5.5 Resume Handles Tab

On this tab, you can display resume handles.

Definition

A resume handle is a unique ID with which a resume point can be identified within a multiple call destination system. PCo generates the resume handle according to your configuration, in order to save the current status when processing of the multiple call destination is suspended. This status is called again when processing is resumed.

Display Resume Handles

1. Choose the *Resume Handles* tab for your multiple call destination system in which you have configured suspend and resume.
2. Select the relevant agent instance from the *Agent Instance* field (at the bottom of the tab page) and then choose the *Refresh* pushbutton.
   The resume handle is displayed in the overview. The step in which the process was suspended and will be resumed later is also displayed.

Related Information

General Settings Tab [page 371]

4.2.5.6 Testing a Multiple Call Destination System

Use

The PCo Management Console offers a test function for the multiple call destination system. You can use this function to test a multiple call destination system without having to include this destination system in an agent instance with a notification.

Prerequisites

- You have configured a multiple call destination system and resolved all the configuration errors of the multiple call destination system displayed by the PCo Management Console.
- You have saved all changes that you made within the PCo Management Console.
- If you call other destination systems from your multiple call destination system that require a running PCo agent instance, for example, a query destination system or an OPC UA destination system, you start these agent instances before the test.
- If you call other destination systems from your multiple call destination system that require a running server, for example, a Web server, you make sure that these servers are available.

Procedure

Test Configuration

1. Choose  on the *Destination System Calls* tab.
   The *Test Configuration* dialog box appears.
2. Enter values for the input variables of your multiple call destination system in the *Variables* table.
3. Choose the *log level*.

   > **i Note**
   >
   > The number of log messages depends on the selected log level. The test function only displays messages whose source is the multiple call destination system that is to be tested. If you use the same multiple call destination system in a running agent instance, you also see, in the log of the agent instance, the messages of the destination systems called from the multiple call destination system.

4. Choose *Run / Continue to End* to run the multiple call destination system without interruption.
5. Choose *Run in Debug Mode / Continue to Next Message* to execute the multiple call destination system step by step from log message to log message.

   > **i Note**
   >
   > Since the log messages are technical messages, they are only issued in English (as is the case with the agent instance log).

The system executes the next steps without interruption to the end.

6. To switch off the debug mode, choose *Run / Continue to End*.
   The next steps are then executed without interruption to the end.
7. If the multiple call destination system no longer responds during execution, or if you want to end execution in debug mode, choose *Stop Debugging*.
8. If you choose *Reset Variable Values* after the test, the values of the output variables and local variables as well as the update count of all variables are reset first.

   > **i** Note
   >
   > The values of the input variables are retained at first and can be used for a new test.

9. If you choose *Reset Variable Values* again, the values of the input variables are deleted.

Screen Area: Variables

During and after execution of the test, you see the values and the update count for the input variables, output variables, and local variables of the multiple call destination system in the *Variables* screen area.

Screen Area: Log Messages

You can see the following information in the *Log Messages* screen area:

- Execution of the multiple call destination system
- Total duration of the test run (after successful execution)

> **i** Note
>
> The displayed duration of the test run can only give a rough indication of the execution time in the productive system because the test run also contains and measures further factors such as starting and pausing the destination systems as well as the output of log messages in the configuration test dialog.

If you choose *Copy Log Messages to Clipboard*, all displayed log messages are copied to the Windows clipboard. From there, you can insert the messages in an external text editor, for example.

> **i** Note
>
> Local variables that you have flagged as **permanent** are set to their initial value during each test run and cannot be carried over to the next test run.

4.2.6 Query Destination System

Use

The query destination system allows you to read current tag values from the source system of an agent instance or to write tag values to the source system of an agent instance. The agent instance can be the same agent instance from which the query destination system is called or another agent instance that is running on the same PCo computer.

> **i Note**
>
> Queries of historical or aggregated data as well as feature queries are not supported by the query destination system.

You can define three types of queries:

- *Retrieve query*
  A retrieve query reads values of tags from a source system and returns them to the caller of the query destination system.
- *Store query*
  A store query writes values of tags to a source system and waits until the write operation is finished.
- *Asynchronous store query*
  An asynchronous store query writes values of tags to a source system and does not wait until the write operation is finished.

> **i Note**
>
> Only certain source system types can be used in query destination systems. For a list of the source system types provided by SAP that can be used, see Source System [page 129].
>
> If you use a customer-specific agent, the functions for queries of current tag values must be implemented there.

Examples

Example 1: Your PCo agent instance provides an OPC UA method that reads a specific text file from a file system and returns the content of the file to the method caller.

Example 2: Your PCo agent instance provides an OPC UA method that sets the value of a tag in a Modbus source system.

Both examples show how PCo can be used for wrapping a variety of functions using OPC UA methods.

More Information

Creating a Query Destination System with Defined Tag Names [page 377]

Creating a Query Destination System with Parameterizable Tag Names [page 378]

Predefined Output Variables in a Query Destination System [page 380]

Recommendations for Using the Query Destination System [page 380]

Working with Arrays in the Query Destination System [page 380]

4.2.6.1 Creating a Query Destination System with Defined Tag Names

Prerequisites

You have created a source system and created an agent instance for this source system. You need to assign this agent instance to the query destination system. The source system type needs to support tag queries and the query type of the query destination system that you want.

Context

The document describes how you can create a query destination system [page 375] and define the names of the tags to be read or written in the query destination system.

Procedure

1. Create a new destination system of the type *query destination system*.
2. Select the query type in the *query definition* area:

Query Type	Description
Retrieve Query	Reads values of tags from a source system and returns them to the caller of the query destination system.
Store Query	Writes values of tags to a source system and waits until the write operation is finished.
Asynchronous Store Query	Writes values of tags to a source system and does not wait until the write operation is finished.

3. In the *Agent Instance* field, choose the agent instance that you created previously.
4. In the *Variables* screen area, choose *Browse Source System and Add Tag as New Variable*.

   The browse window appears.
5. Choose the *Browse* pushbutton.

   The address node of the agent instance appears.
6. Select one or more tags. Choose *Add Selected Items* and close the browse window by choosing *OK*.

> ⚠ Caution
>
> The query destination system **only supports primitive data types**. If you select tags with data types that the query destination system does not support, such as arrays, or structured data types, the

system issues a warning message. You can ignore the warning and add the tags with the data type `System.String`. However, it depends on the data source whether it is possible to convert the unsupported data type into `System.String` during the runtime of the agent instance. Check the log of the agent instance, which is connected with the data source, for corresponding error messages.

The system now generates a variable for each selected tag and displays them in the *Variables* screen area. You can change the variable name.

Depending on the query type, these variables represent the input variables (for store queries) or the output variables (for retrieve queries) of the query destination system. They appear, for example, when used in a multiple call destination system under their variable name and with the suffix `_Value`.

> **Example**
>
> You have defined the variable name **`Double`** in the query destination system. If you use the query destination system in a multiple call destination system, the variable is displayed there as the destination system variable `Double_Value`.

Next Steps

Creating a Query Destination System with Parameterizable Tag Names [page 378]

Recommendations for Using the Query Destination System [page 380]

4.2.6.2 Creating a Query Destination System with Parameterizable Tag Names

Prerequisites

You have created a source system, for example, an OPC UA source system, and an agent instance. The source system needs to support tag queries and the query type of the destination system.

Context

This procedure describes what you need to configure so that the names of the tags that are to be read or written can be determined dynamically at the runtime of the query destination system.

> ### Example
>
> Your source system has the four OPC UA tags **Tag1** through **Tag4** that can all be found under the same node `Machine1`:
>
> - `nsurl=PCo;s=Machine1.Tag1`
> - `nsurl=PCo;s=Machine1.Tag2`
> - `nsurl=PCo;s=Machine1.Tag3`
> - `nsurl=PCo;s=Machine1.Tag4`
>
> On a case-by-case basis you want to write a value to one of the four tags, but you only want to configure one query destination system.

Procedure

1. Create a new destination system of the type *query destination system*.
2. Select the *query type* in the query definition area, for example, *store query*.
3. Select the agent instance in the *query definition* area.
4. In the *Variables* screen area, choose *Browse Source System and Add Tag as New Variable*.

   The browse window appears.
5. In the browse window, choose only one of the four tags, for example, `Tag1` (see example above). Choose *Add Selected Items* and close the browse window by choosing *OK*.

   The system now generates a variable for the tag. You can change the variable name to **`Tag`**, for example.
6. In the *Source System Tag* column of the *Variables* table, replace the part of the tag path that is determined dynamically at the runtime of the agent instance with the character string **`{0}`**.

   In this example, therefore, you change `nsurl=PCo;s=Machine1.Tag1` to **`nsurl=PCo;s=Machine1.Tag{0}`**.

   > ### ⚠ Caution
   >
   > As soon as you change a tag path manually, the system can no longer ensure that the source system tag you entered is compatible with the data type displayed. You may need to adjust the data type.
7. Depending on the query type, the query destination system now has an input variable (for store queries) or an output variable (for retrieve queries) with the name `Tag_Value` for the tag value. With the addition of the parameter **`{0}`**, the query destination system also gets an input variable with the name `Tag_Parameter_0` of data type `System.String`. You transfer the character string, which is to be implemented at the runtime of the agent Instance instead of the **parameter {0}** to this input variable.
8. You can also make additional parts of the tag name parameterizable. To do so, replace further parts of the original tag name with the character strings **`{1}`**, **`{2}`**, **`{3}`**, and so on.

   Each of these new parameters generates a new input variable with the suffixes `_Parameter_1`, `_Parameter_2`, `_Parameter_3`, and so on. In the example given above, you can also make the node name of the tag parameterizable by using the following character string as a tag name:

   `nsurl=PCo;s={1}.Tag{0}`

4.2.6.3 Predefined Output Variables in a Query Destination System

Query destination systems of query type **Retrieve Query** or **Store Query** provide two predefined output variables that you can use, for example, in a multiple call destination system, to react to errors in the query.

Output Variables

Output Variable	Description
`!outEventType!`	Events that occur within a query destination system can have the severity levels **Critical**, **Error**, **Warning**, **Information**, or **Verbose**. The output variable contains the highest severity of all events that occurred during a query. The implementation of the source system involved, which executes the query at agent runtime, determines which messages are generated and with which severity.
`!outMessage!`	The output variable contains all messages that occurred during a query in a single language-independent character string.

4.2.6.4 Recommendations for Using the Query Destination System

SAP recommends the following for using a query destination system:

- Integrate the query destination system in a notification or in a multiple call destination system. (See also: Multiple Call Destination System [page 353].)
- Even if you do not need a multiple call destination system for your actual business process, it is recommended that you integrate the query destination system in a multiple destination system for test purposes and test it from there. (See also: Testing a Multiple Call Destination System [page 374].)
- At the runtime of the query destination system, you need to make sure that the PCo agent instance to which the query destination system refers is running.

4.2.6.5 Working with Arrays in the Query Destination System

Using Array-Typed Tags

If the tag from which you want to read or into which you want to write has the type of an array, you cannot directly assign a PCo variable with an array data type to the tag. Instead, the system automatically assigns the

type `System.String` to such a variable. However, you can still work with array-typed tags in PCo. To do this, you have to **integrate the call of the query destination system** with which you read or write the tag **into a multiple call destination system**. You can then make the necessary conversions between string and array, manipulate the elements of the array, or use the array for further destination system calls. In the multiple call destination system, define the required local variables and use the functions of the expression editor to calculate the relevant variable.

Reading an Array-Typed Tag

If you read an array-typed tag using the query destination system in *Retrieve Query* mode, the assigned variable contains a string in which the elements of the array are separated by commas.

> ✤ Example
>
> An array that consists of three Int32 numbers generates a string of the form `1, 2, 3` when being read via the query destination system.

In the expression editor, you have to use the **function** `stringSplitToArray` to convert the string that has been read into a variable of a suitable array data type. You can use the array variable for further functions and destination system calls.

> ✤ Example
>
> The function `stringsplittoarray('StringValue', ",")` generates an array consisting of three integer numbers 1, 2, and 3 from the variable **StringValue** with the content **1, 2, 3**, and can eventually be assigned to a variable of the type `System.Int32[]`.

Changing Values of an Array

After you have converted the value read from the tag to a variable of an array data type as described above, you can use the **function** `arrayGetValue` of the expression editor to gain read access to individual elements of the array. Call the **function** `arraySetValue` if you want to change the value of an array element.

> ✤ Example
>
> Calling the function `arrayGetValue('IntegerArray', 1)` returns the value of the second element (in this case, it is 2) for the array generated above consisting of integer numbers, and can be assigned to a variable of the type `System.Int32`.

> ✤ Example
>
> Calling the function `arraySetValue('IntegerArray', 7, 1)` returns a copy of the array variable `IntegerArray`, where the second array element was overwritten with the number 7. The variable `IntegerArray` itself is not changed by the call of the function `arraySetValue`, unless you explicitly assign the result of the calculation `arraySetValue('IntegerArray', 7, 1)` to the variable `IntegerArray` again.

Writing to an Array-Typed Tag

To write to an array-typed tag, you have to call the query destination system in *Store Query* mode. The string variable, whose content you want to write to the tag, must contain the array in JSON format. In the expression editor, use the **function** `toJson` to convert the array into the string with the correct format.

> **Example**
>
> The integer array with the elements 1, 2, and 3 is converted to the `JSON string [1,2,3]` when the function `toJson('IntegerArray', false)` is called. This string can eventually be written back to the array-typed tag of the type `System.Int32[]` via the query destination system.

Depending on the source system, the array may be completely replaced by the new value. It is not possible to access individual elements or subareas using PCo. Therefore, make sure that the length of the array corresponds to the array length defined in the source system.

PCo does not support multidimensional arrays.

If you want to write an array in an OPC UA server as described, the server must have declared the `ValueRank` of the tag with the **value 1**. The **ValueRank 1** denotes a **one-dimensional** array. Writing in tags with a **ValueRank 0** (**one or more dimensions**) or **ValueRank -2** (**any dimension**) is not possible in this way.

Handing Arrays of the Type System.Byte[]

There is a different system behavior for tags with the data type `System.Byte[]`.

- During the **write** process via the query destination system, the system expects the string that is to be written to be Base64-encoded. You can use the functions `base64encode` and `base64decode` to convert strings to the Base64 format and back in the expression editor. You can use the functions `getByteArrayFromStringUtf8` and `getStringFromByteArrayUtf8` for a conversion between strings and byte arrays.
- During the **read** process, a byte array is returned as a string in which integers are separated by commas. As described above, you can then use the function `stringSplitToArray` to convert to an array that can be assigned to a variable of the type `System.Byte[]`.

4.2.7 MQTT Destination System

You can use the MQTT destination system to send data to an MQTT broker.

You use the MQTT destination system to be able to send MQTT messages to an MQTT broker (MQTT server) that is in the cloud, for example. This allows you to publish data, which has been transferred to PCo, to an MQTT broker. The **MQTT destination system** takes on the role of **publisher**.

> **Note**
>
> The central aspect of MQTT is an event-driven publish/subscribe architecture. For MQTT, there is a central server (broker) to which the data sender and data recipient are connected equally. Messages are sent

(published) or received (subscribed) using topics. A topic is a string that has a URL-like structure and represents a type of subject of the message.

The following tabs are available:

- *Client* tab
- *Security Settings* tab
- *Message Settings* tab

4.2.7.1 Client Tab

Prerequisites

You have created and configured an MQTT source system.

Procedure

1. Select an MQTT source system in the *Client* field.
   All the settings are adopted from the selected MQTT source system. (See also: MQTT Source System: Client Tab [page 206].)
2. If you choose *Navigate to the Client*, you can navigate to the MQTT source system and change the settings there.
3. Change the ID in the *Client ID* field, if necessary.
   The *Client ID* field can be edited; that is, you can overwrite the client ID that was copied over from the source system. If you change the ID, the connection to the MQTT server cannot be shared. The decision about whether or not you should change the client ID depends on the specific scenario. A connection to the MQTT server can only be shared by MQTT destination systems that are used by the same agent instance and have the same client ID.
   Alternatively, you can leave the field blank. In this case, PCo creates a new client ID when the connection is being established. This makes sense if multiple connections are to be made to MQTT servers, and ensures that the client ID is unique in each case. If you enter a client ID manually, you must ensure that the ID is unique for each broker instance.

4.2.7.2 Security Settings Tab

Use

On this tab, you can enter an **application certificate that differs from the** MQTT client, if required, for the MQTT destination system. This might be useful if you want to connect to the same MQTT server across several MQTT destination systems that each have a different client ID. In this case, you can create **a common MQTT source system** for the server connection, but maintain a standalone application certificate for each of the MQTT destination systems, that is, for each of the client IDs.

Prerequisites

The security settings are only ready for input if you have defined a prefix for a secure connection (**wss** or **mqtts**) in the *Server URI* field on the *Client* tab.

Procedure

In the *Origin of Certificate* screen area, select one of the two options:

Origin of Certificate

Radio Button	Description
Inherited from Client	With this option, the certificate of the corresponding MQTT source system (client) is valid for the MQTT destination system. This is the default setting.
Defined in Current Destination System	Select this option if you want to define your own application certificate for the destination system. You can then choose the *Select Certificate* pushbutton to select an **application certificate**.

4.2.7.3 Message Settings Tab

You use the *Message Settings* tab to configure the template for the MQTT messages that are to be sent later to the MQTT server.

Payload Settings

In the upper part of the screen, you can define the following data for the MQTT messages:

Payload Settings

Field	Description
Topic Name	You enter the topic name for the MQTT message here. The following restrictions apply when you enter the topic name: • It must not start with a `$` character because this character is reserved for the purposes of the MQTT server. • It must not contain any wildcard characters, such as `#` or `+`. (See also: Wildcards in the Topic Filter [page 222].) If you **do not enter anything** for the topic name here, the MQTT destination system automatically provides an input variable of the type `String` with the name **inTopicName**. In this way, the topic can be determined for the message during runtime. You can enter a **constant value** or a **template** here. For example, if you want to determine the topic name based on the value of the input variable **MachineID**, you can enter the following character string, for example: **`Measurement recording/{MachineID}/Temperature`**. The string {MachineID} is then displayed as an input variable `MachineID` in the table of input variables. At runtime, the machine ID is read and replaced dynamically.

Field	Description
QoS	Quality of service: To make sure that a sent message reaches the recipient, MQTT defines three different quality of service (QoS) levels with which a message can be sent: • *0 - At Most Once* This service quality level is suitable for transferring sensor data. Messages might be lost. The message reaches the recipient at most once. • *1 - At Least Once* This service quality level ensures that the message reaches the recipient. The recipient might receive the message more than once. • *2 - Exactly Once* This level makes sure that the message reaches the recipient exactly once.
Payload Type	Specifies the format of the MQTT messages. The following formats are supported: • *JSON* • *XML* • *URL-Encoded* • *Formatted String*

Generate Mapping from Template

You can choose the button *Generate Mapping from Template* to insert a specified character string in the dialog box that appears. The template generates an MQTT message that is to be sent from PCo to the MQTT server. (See: Generate Mapping from Template [page 386].)

Message Configuration Screen Area

In the lower part of the screen, you can add payload parameters if you have chosen JSON, XML, or URL-formatted as the *payload type*. Using the parameters you have entered, PCo generates a template for the MQTT messages. (See: Message Configuration Screen Area [page 387].)

4.2.7.3.1 Generate Mapping from Template

You can choose the button *Generate Mapping from Template* to insert a specified character string in the dialog box that appears. The template generates an MQTT message that is to be sent from PCo to the MQTT server.

This is possible for all payload types that are offered. When you choose *Apply*, this data is automatically transferred to the *Message Configuration* table as a parameter, unless you have selected the payload type **Formatted String**.

If you enter the parameters manually, you can display the template string for the MQTT message here.

> ### Example
>
> The syntax for entering a parameter name is structured in the following way for the payload type **Formatted String**:
>
> ```
> {$<variable name>}
> ```
>
> The parameter name (variable name) needs to start with a letter and can only contain letters and numbers.
>
> Here is an example of a message template for the payload type **Formatted String**:
>
> ```
> Length: {$length}m, Width: {$width}m, Height: {$height}m
> ```
>
> The MQTT message that is sent later from PCo looks like this, for example:
>
> *Length: 2m, Width: 1m, Height: 3m*.
>
> The parameters have been filled with tag values from the MQTT server and are displayed in the *Tags* table in the lower screen area. In this case, the *Message Configuration* table in the lower screen area is not input-enabled and remains empty.

> ### Example
>
> Example of a **JSON** template:
>
> ```
> {"Print": "{inPrint:string}","Temperature": "{inTemperature:string}"}
> ```
>
> The print and temperature parameters are copied to the *message configuration*. The variables `inPrint` and `inTemperature` are displayed in the *Input Variables* table.

4.2.7.3.2 Message Configuration Screen Area

In the lower part of the screen, you can add payload parameters if you have chosen JSON, XML, or URL-formatted as the *payload type*. Using the parameters you have entered, PCo generates a template for the MQTT messages.

1. Select the root parameter **$in** and then choose  to add a new parameter in the *Message Configuration* table.
   The *Add Parameter* dialog box appears.

2. Enter the following data in the dialog box:

   Add Parameter

Field	Description
Parameter	Here you enter the name of the parameter that you want to be used in the MQTT message.
Data Type	You enter the data type of the parameter here.
Variable	If you select this checkbox, the value entered in the *Value* field is not interpreted as a fixed value but as an input variable that is to be filled later accordingly. The parameter is then automatically included in the list of input variables. You can then map this variable to a subscription item in the notification object. You perform this mapping on the *Destinations* tab for the notification.
Value	If you have selected the *Variable* checkbox, the variable name is proposed here automatically. All the input variables have the prefix *in*. You can change the variable name here or later in the table. If it is a fixed value, you can enter the value here.

   The parameter is displayed in the *Message Configuration* table and included in the *input variables* table.

3. To delete a parameter, select the row in the table and choose *Remove Variable*.
4. You can use the *Import Data Types into Data Type Repository* pushbutton to import the entire message configuration into the PCo data type repository as a single structured data type. The *Select Data Type* pushbutton takes you to the PCo data type repository, where you can select the relevant previously imported structured data type. (See also: Datentypen in das Datentypverzeichnis importieren [page 388].)

4.2.7.3.2.1 Datentypen in das Datentypverzeichnis importieren

Mit der Drucktaste *Datentypen in das Datentypverzeichnis importieren* können Sie die gesamte Nachrichtenkonfiguration als einen einzigen strukturierten Datentyp in das PCo-Datentypverzeichnis importieren, so dass Sie zukünftig nur noch eine einzige Root-Variable vom Typ eines strukturierten Datentyps definieren müssen.

Die Datenstruktur kann dann beim Anlegen eines MQTT-Zielsystems oder bei der Verwendung eines MQTT-Zielsystems in einem Mehrfachaufruf-Zielsystem wieder ausgewählt werden. Bei der Verwendung des MQTT-Zielsystems, z.B. in einem Mehrfachaufruf-Zielsystem, ist dann nur noch die Eingabe einer einzigen Eingabevariablen erforderlich.

Mit der Drucktaste *Datentyp auswählen* gelangen Sie in das PCo-Datentypverzeichnis, wo Sie den zugehörigen, zuvor importierten, strukturierten Datentyp auswählen können. Die Eingabe des Root-Parameters als einziger

Eingabeparameter ist in diesem Fall ausreichend, um JSON-Dateien umzuwandeln und als Datenstruktur an einen MQTT-Broker zu übergeben. (Siehe auch: Functions in the Import Data Types Dialog Box [page 587])

4.2.8 Simulation Destination System: Configuration Tab

Context

The *simulation destination system* is a type of destination system that you can use for testing by sending notification messages to a folder in a specific directory on your computer. This allows you to test the notification process very quickly.

Procedure

1. Create a destination system of the type *simulation destination system*.
2. Click on the *Configuration* tab.
3. Enter the following data:

Field	Description
Destination	Directory in which the notification messages are to be saved. PCo proposes a directory, for example, `C:\Users\[user name]\Documents`. Choose the *Browse* pushbutton to search the directory and create a new folder. If you enter **`NUL`** as the directory, the notification messages are sent but they are not saved.
Format	Selection option for the format in which the notification message file is stored. You can choose between output into an XML file or a binary file. XML file is the default setting as only the XML format is human-readable.

File Name Composed of Time Stamp and Counter	If you select this checkbox, the file name of a notification message contains a 24-digit number that consists of a time stamp in milliseconds and a six-digit counter from 000000 to 999999. If the counter exceeds 999999, it starts afresh at 000000. The counter starts again at 000000 when you restart the agent instance.
	
	Example
	`NotificationMessage_6367613827905991060000000.xml`
	`NotificationMessage_6367613828003915800000001.xml`
	`NotificationMessage_6367613828103929900000002.xml`
	
	The generated files can be sorted in a Windows Explorer window by name to clearly determine the time sequence in which the files have been generated.
	
	Each notification message gets a unique 24-digit number at the runtime of an agent instance. If you have two or more agent instances running in parallel, you should avoid that simulation destination systems of different agent instances write to the same directory, because the uniqueness of file names between the agent instances is not guaranteed.
	
	If you do not select this checkbox, the file name of a notification message contains its technical ID.
	
	Example
	NotificationMessage_3ee57e7f-311e-4305-a4eb-bf5f451febd0.xml

Sleep	Queue time (in milliseconds) of the destination system until the next notification message is accepted. You can use this to simulate a slow network connection. The notification messages that are not accepted are placed in a queue and processed from this queue.
Failures Percentage	Percentage of notification messages that should fail. This percentage is intended for simulation purposes to test the saving and processing of faulty notification messages. If, for example, you enter 50%, half of the messages are treated as faulty messages and are displayed on the *Message Retries* tab. (See also: Agent Instance: Displaying Messages [page 490].)

4.2.9 MII Destination System (Deprecated)

> **i Note**
>
> This destination system type is deprecated. If you want to call an MII transaction using Plant Connectivity, use the universal Web service destination system in future.
>
> For more information about calling MII transactions using Web services, see Calling MII Transactions Using Web Services [page 38].
>
> For compatibility reasons, the MII destination system remains part of the PCo delivery.
>
> Should you still want to use the MII destination system in exceptional cases, you have to select the *Allow Creation of Deprecated Configuration Elements* checkbox under *Tools* > *Options* > *Global Settings* > *Compatibility* (see: Compatibility Settings [page 17].)

You use an MII destination system if you are using the notification process and, for compatibility reasons, want to continue to connect an SAP MII system by calling transactions directly.

> **i Note**
>
> Depending on the NetWeaver version of your MII destination system, logon problems may occur via the MII destination system (see also SAP Note: 2436742).
>
> If the problem cannot be solved by changing the server configuration, you have to call the MII transaction from the universal Web service destination system. To do this, follow the description given above.

The following tabs are available for setting up the connection to an SAP MII system:

- Server Tab [page 391]
- Advanced Tab [page 392]

4.2.9.1 Server Tab

Use

You must create and configure an SAP MII destination system in the notification process only.

Procedure

You can create SAP MII as the destination system as follows:

1. In the *Plant Connectivity Management Console*, select an SAP MII destination system and click on the *Server* tab.

2. Enter the following data:

Field	Description
Server Name	Enter the name of a valid SAP MII server, such as **`pwdf6351.wdf.sap.corp`**. You can also choose the *Browse* pushbutton to call up a list with all servers in the local network. You can then select a server from the list.
Port	Enter a port number to use for communication with SAP MII, such as **`50000`**. **i Note** If you enter **`0`**, no port is used.
Use Secure Sockets	If secure communication is required, select this checkbox. The HTTPS protocol is then used.
Allow Untrusted Server Certificate	If you do not want to allow an untrusted server certificate, select this checkbox.
Version	Here you select the version of the SAP MII system that you want to connect to as a destination system.
User Name	User name with which you log on to the MII system
Password	Password with which you log on to the MII system

3. To check the configuration, click the *Test Connection* function key.
   A message dialog displays the connection status.
4. Click the *Save Destination System* icon to save your entries.

4.2.9.2 Advanced Tab

Prerequisites

The entries made on the *Server* tab must have led to a connection being made.

For more information, see Server Tab [page 391].

Procedure

1. Select an SAP MII destination system and click on the *Advanced* tab.
2. Enter the following data:

Field	Description
Keep Alive	This indicates how long the connection to SAP MII is kept alive so that queries do not have to log on each time. Enter the *Keep Alive* interval in milliseconds here.
Request Timeout	Indicates how long PCo waits for a response from the SAP MII server before a timeout occurs. Enter a value in milliseconds.

4.2.10 Web Service Destination System (Deprecated)

> **i Note**
>
> This destination system type is deprecated. Use the destination system type *Universal Web Service Destination System* instead (see: Universal Web Service Destination System [page 273].)
>
> Should you still want to use the Web service destination system in exceptional cases, you have to select the *Allow Creation of Deprecated Configuration Elements* checkbox under *Tools* > *Options* > *Global Settings* > *Compatibility* (see: Compatibility Settings [page 17].)

If you want to connect to a third-party system, such as an SAP ME system, you need to create a Web Service Destination system. (See: Integration with Third-Party Systems Using Web Services [page 66].)

This destination system type makes it possible for the PCo agent instances to send notification messages to a selected Web service. SAP ME provides its own Web services. PCo retrieves variable values, such as the SFC, from the data source and transfers them to the SAP ME system, for example, to a particular ME activity. PCo can predefine fixed values for specific parameters of the Web service.

You can also use this destination system in enhanced notification processing if you want the destination system's response to be processed. (See also: Integration with Third-Party Systems Using ENP [page 68].)

The following tabs are available for configuring a Web Service Destination:

- Server Settings Tab (WS Destination System) [page 394]
- Operation Configuration Tab (WS Destination System) [page 396]
- Advanced Settings (WS Destination System) [page 408]

4.2.10.1 Server Settings Tab (WS Destination System)

Context

If you want to connect a third-party system, such as SAP ME, to PCo, you need to create a Web Service Destination system.

Procedure

1. Choose *Add Destination System*. Then choose the destination system type *Web Service Destination System*.
2. Enter the following data on the *Server Settings* tab:

Field	Description
WSDL URL	Enter here the URL for the WSDL with which the third-party system (for example, SAP ME) makes its Web services available.
Timeout	Length of time in milliseconds in which the client application waits for a response. If there is no response during this time period, processing is interrupted.

Field	Description
Detailed Logging	You can use this setting to generate a log file in the `<installation directory>\WSLogs` folder that contains detailed information about the message sent and the response received. The following file name is created for the log file: `WS_[notification id].log` The following options are available: ○ *No Logging* With this setting, the PCo system does not perform detailed logging. ○ *Only for Failed Web Service Calls* With this setting, the PCo system logs errors or exceptions. ○ *Only for Successful Web Service Calls* With this setting, the PCo system generates a log only if a request was successful, meaning that there were no errors and no exceptions. ○ *For All Web Service Calls* With this setting, a log is always generated.
Authentication	Specifies how the destination system verifies the digital identity of a sender of communication data. You have the following options: ○ *HTTP Basic Authentication* This is the default authentication. In this case, you need to specify a user name and a password for the destination system (for example, for SAP ME). **i Note** If you do not select an *https Web service*, the logon data is not transmitted securely. ○ *No Authentication* No logon data is required for this setting.
User Name	Valid user name for the third-party system The user name is only required if the *HTTP Basic Authentication* setting was selected previously.
Password	Valid password for the third-party system

3. Choose *Call Services*.

   All available services are displayed in the *Service Bindings* screen area.

In the background, PCo establishes a connection between the destination system and the server on which the Web service is stored. A proxy class is generated for the Web service.

4. If several services or several endpoint URLs are displayed for a service, you click the service you want to select.
5. Save your entries.

Next Steps

For more information about the configuration of the Web Service Destination in connection with enhanced notification processing, see the *Implementation Guide for Enhanced Notification Processing (ENP)* which can be found in the *Additional Information* section in the SAP Help Portal under https://help.sap.com/viewer/p/SAP_PLANT_CONNECTIVITY. See also SAP Note 1741665.

4.2.10.2 Operation Configuration Tab (WS Destination System)

Prerequisites

You have created a Web service destination system, made all necessary settings on the *Server Settings* tab, and have selected a Web service.

Context

On the *Operation Configuration* tab, you can **configure a selected service operation** that is provided by the Web service. This makes it possible to send notification messages with value changes in the subscribed tags to the third-party system. PCo sends the values of the subscribed tags to the selected Web service.

On this tab, you can select a service operation and create a **request message** for this service operation. If you want to use a notification enhancement, you can also configure a **response message**.

First, in the request message, you assign variables or fixed values to the fields that are provided by the service operation. You can assign fixed values directly to the fields. The current values of the variables are assigned to the relevant fields before the Web service is called. In the next step, you can assign variables to the fields of a response message that you then continue processing in a notification enhancement, for example, in the SAP standard enhancement *Destination System Calls with Response Processing* (see also: Destination System Calls with Response Processing [page 81]).

> **Note**
>
> For more information about the Web services provided by SAP ME, see the product page for SAP ME under https://help.sap.com/viewer/product/SAP_MANUFACTURING_EXECUTION/15.4/en-US?task=discover_task.

Procedure

1. Click on the *Operation Configuration* tab.

   The *Service Operations* field displays all operations that are provided by the selected Web service.
2. Click on an operation, for example, *Start (StartRequestMessage_sync StartRequest_syn)*.

   The tabs for configuring the request message and response message are displayed in the lower screen area:

   Request Message Configuration Tab [page 397]

   Response Message Configuration Tab [page 401]

Example

Example of Configuring a Service Operation [page 406]

Next Steps

For more information about configuring a service operation in conjunction with a notification enhancement, see the *Implementation Guide for Enhanced Notification Processing (ENP)* on the PCo product page, and SAP Note 1741665.

4.2.10.2.1 Request Message Configuration Tab

Prerequisites

You have selected a service operation.

Context

You create the request message on this tab. Then, in the request message, you assign variables or fixed values to the fields that are provided by the service operation. You can assign fixed values directly to the fields. The current values of the variables are assigned to the relevant fields before the Web service is called.

The *Request Message Configuration* table is displayed in the right screen area. It contains the fields of the selected operation. The *list of variables* is displayed in the right screen area. The variables are displayed here as soon as you have marked a field as a variable.

Procedure

1. Choose the *Create Request Message* pushbutton.

   The system displays the *Request Message Configuration* table for the selected service operation.

Column	Description
Required Parameter	The asterisk * indicates that an entry is required in this row in the *Value* field. If there is no asterisk, the entry is optional.
Parameter Name	Specifies the name of the service operation parameter, such as: ○ Message Header ○ SiteRef ○ Site
Parameter Type	Specifies the type provided by the Web Service, for example: ○ StartRequestMessage_sync ○ String ○ SiteRef

Column	Description
Value	In this column, you can either enter a fixed value or specify a variable for the service operation field. You can assign these variables later to the subscription items of the agent instance and fill the Web service with the values of the data source. The following functions are provided as pushbuttons: ○ *Create Object* If you choose this pushbutton, the selected object is created in the request message. In the table, PCo shows further objects for this higher-level object. ○ *Delete Object* If you choose this pushbutton, you can delete an object in the request message. The object is then removed from the table. ○ *Create Array* If you choose this pushbutton, you can create an array for an object. (See: Creating Arrays [page 400].)
Input Variable	If you have selected the *Input Variable* checkbox for a row, the contents of your entry in the *Value* field are interpreted as a variable. The variable is then shown in the list of input variables. You can assign this variable to a subscription item in the notification object that you define for integration with a third-party system. You perform this mapping on the *Destinations* tab for the notification. (See also: Integration with Third-Party Systems Using Web Services (WSD) [page 66].) **⚠ Caution** If you are using the Web Service Destination together with a notification enhancement, you need to make sure you use unique variable names within the Web service configuration. A variable name that is used in the request message configuration must not be used again in the configuration of the response message. If you do not select the checkbox, the value entered is interpreted as a fixed value.

2. Enter the required data as described in the example [page 406].
3. Now you can test the Web service call by choosing the *Test Request Message* pushbutton. Even if you want to define a *response message*, you need to choose the *Test Request Message* pushbutton.

   The *Set Variables* dialog box appears.

4. Enter the values for the variables and click OK.

   If the Web service call was successful, a response is generated. PCo now displays the *Response Message Configuration* tab with the response message fields. All the response message fields are displayed with current values.

   You can now assign variables to the response message fields that you can use in a notification enhancement. For more information, see Response Message Configuration Tab [page 401].

4.2.10.2.1.1 Creating Arrays

Procedure

You can create an array for an object by choosing the *Create Array* pushbutton. If you choose the pushbutton, PCo displays the relevant dialog box.

The following types of arrays are possible:

- Array with predefined size
  You can give the *Array Size* field a fixed number. PCo then creates a corresponding number of nodes for the selected field.
- Array with dynamic size
  If you are using enhanced notification processing, you can also define arrays with a dynamic size. The size of the array is then determined by the program logic. In this case, you can specify a letter in the *Array Size* field, for example, `k`. PCo then creates a subnode for this indexed variable with the description k. The lower-level variables of the subnode then acquire the name *<Variable Name>[<Index Variable>]*, for example, `SFC[k]`.
- Two-dimensional arrays
  When you use enhanced notification processing, you can also define two-dimensional arrays.
  Analog to one-dimensional arrays where you define an individual control variable, you define two control variables for two-dimensional arrays. You then call the lower-level variables *<Variable Name>[Index Variable 1][Index Variable 2]*, for example, `DESCRIPTION[m][n]`.

More Information

For more information about arrays, see the *Implementation Guide for Enhanced Notification Processing (ENP)* (paragraph 5.3.2.) on *SAP Service Marketplace* under *service.sap.com* > */instguides* > *SAP Business Suite Applications* > *SAP Manufacturing* > *Plant Connectivity* > *Plant Connectivity 15.1* >.

4.2.10.2.2 Response Message Configuration Tab

Prerequisites

You have tested the Web service call using the *Test Request Message* pushbutton and entered values for the request variables in the dialog box. If the call was successful, a response is generated.

Context

On this tab, you can assign variables to the fields of a response message that you then continue processing in a notification enhancement, for example, in the SAP standard enhancement *Destination System Calls with Response Processing* (see also: Destination System Calls with Response Processing [page 81]).

> **i Note**
>
> Defining a response message configuration only makes sense if you want to use the Web Service Destination together with a notification enhancement, for example, with the SAP standard enhancement *Destination System Calls with Response Processing*.

The *Response Message Configuration* tab is divided up into three screen areas:

- Response Message Configuration Table
- List of Selected Output Variables
- Calculated Variables

Procedure

1. If the call of the Web service was successful, PCo displays the fields for the response message on the *Response Message Configuration* tab. All the response message fields are displayed with the current values of the response.

   You can now assign variables to the response message fields that you can use in a notification enhancement.

   The following data is displayed in the *Response Message Configuration* table:

Column	Description
Parameter Name	The parameters that have been generated for the response message are displayed here.

Column	Description
Parameter Type	Specifies for each parameter the object type provided by the Web Service, for example: ○ StartRequestMessage_sync ○ String ○ SiteRef
Value	The values determined by the Web service are displayed here.
Output Variable	Here you can assign a variable name to each field that shows a variable. These names are then displayed in the list of output variables. **i Note** Make sure that variable names are unique within the request and response message configurations. A variable name that is already being used in the request message configuration must not be used again in the configuration of the response message.

2. In addition to the variables that you assign to the fields of a response message, you can define calculated variables in the lower screen area that you can also use in a notification enhancement. For more information about calculated variables, see Calculated Variables in a Web Service Response Message [page 402].
3. Save your entries.

4.2.10.2.2.1 Calculated Variables in a Web Service Response Message

Use

In addition to the variables that you assign to the fields of a response message, you can define calculated variables in the lower screen area of the *Response Message Configuration* tab. With calculated variables, you can convert response variables using expressions, add additional information to them, or convert them into another data type.

Calculated variables are available as additional response variables in variable assignment of enhanced notification processing. (See: Assignment of Modules and Variables (ENP) [page 75].)

Example

Examples of use cases of calculated variables are:

- Conversion of the value of a response variable into another unit
- Character string operations to continue processing a substring of a response variable or to append additional information to a character string.
- Conversion of the data type of a response variable to be able to assign the variable value to a source system tag with a predefined data type.
- Case-dependent values: If, for example, the value of a response variable is smaller than 0, 0 needs to be written to the tag of the data source; otherwise, the unchanged value.

More Information

Use Cases for Calculated Variables [page 404]

Defining and Testing Calculated Variables [page 403]

4.2.10.2.2.1.1 Defining and Testing Calculated Variables

Use

In the *Calculated Variables* screen area, you can specify calculated variables by defining variable names and expressions that you create using the expression editor (see: Expression Editor [page 545].) You can use the variables of the response message configuration in the expressions.

> ⚠ Caution
>
> Make sure that the variable names are unique within the configuration of a Web service. This also applies to the names of the calculated variables that are dealt with exactly like the response message variables.

Prerequisites

- You have defined a request message for your Web service destination system on the *Request Message Configuration* tab.
- You have tested the request message by choosing the *Test Request Message* pushbutton.
- In the *Set Variables* dialog box, you have entered values for the request message variables you defined to be able to test the Web service.
- On the *Response Message Configuration* tab, you first defined the response variables by assigning the variables to individual fields of the response message.

Procedure

To generate calculated variables, do the following:

1. Generate a new calculated variable by choosing the *Add Calculated Variable* pushbutton.
2. Define a unique name for the calculated variable. Note that the name of the calculated variable must not correspond to an existing request variable or response variable or to another calculated variable.
3. Choose a *data type* for the calculated variable. The result of the expression is converted at the runtime of the agent into the selected data type, if this is possible.
4. Define an expression by selecting a row and choosing the pushbutton *Edit Expression for Selected Calculated Variable*.
5. In the *Expression Editor* dialog box, define an expression using the available functions and operators. In your expressions, you can use the response variables that are available in the list of variables.
6. Close the expression editor.
7. Save your changes.
8. Check the function of your calculated variables by choosing the *Test Calculation* pushbutton.
   The system calculates, for example, the results of the calculated variables using the values for the response variables that are displayed in the *Response Message Configuration* table. The results are displayed in the *Calculation Result* column.

Example

Use Cases for Calculated Variables [page 404]

4.2.10.2.2.1.2 Use Cases for Calculated Variables

Example 1: Extracting a Character String from a Response Variable

The service operation `startSfc` of the SAP ME Web service *SfcStartServiceWSClient* returns the material reference in the *itemRef* field as the following character string:

```
ItemBO:1000,MATERIAL_0001,REV1
```

You have assigned the variable *RespItemRef* to the relevant field *itemRef* of the response message.

You now want to extract the material MATERIAL_0001 from the material reference. Since the individual components of the material reference are separated by commas in the character string, you can define the following variables in the *Calculated Variables* screen area on the *Response Message Configuration* tab to fulfill this task:

Field	Input
Calculated Variable	**`CalcMaterial`**

Field	Input
Data Type	**`System.String`**
Expression	**`stringsplit('RespItemRef',",",2)`** **Note** The **`stringsplit`** function splits the character string *ItemBO:1000,MATERIAL_0001,REV1* into the substrings separated by a comma (",") as separator and returns the second substring, in this case the material number `MATERIAL_0001`.

Example 2: Conversion Into Another Unit

A Web service returns the current weather data of a town in the USA. The temperature is given in degrees Fahrenheit as a whole number. However, you need the temperature in degrees Celsius (as a whole number). You have assigned the variable *RespTemperatureFahrenheit* to the relevant field with the temperature value in Fahrenheit.

In the *Calculated Variables* screen area, you can make the following entries for the conversion:

Field	Input
Calculated Variable:	**`CalcTemperatureCelsius`**
Data Type:	**`System.Int32`**
Expression:	**`('RespTemperatureFahrenheit'-32)*5/9`**

Example 3: Conversion Into Another Data Type

A Web service returns a numeric value with the data type `Integer`. However, you need the value as a character string with the data type `String` to write it to a tag of the type `String` at the data source. You have assigned the variable **`RespValueInteger`** to the relevant field in the Web service response.

The entries for the conversion might look as follows:

Field	Input
Calculated Variable	**`CalcValueString`**

Field	Input
Data Type	`System.String`
Expression	`'RespValueInteger'`

Example 4: Case-Dependent Values

A Web service returns a numeric value in the range -100 to +100. However, you want to filter values smaller than 0 and write them as 0 to the source system. You have assigned the variable *RespValue* to the relevant field in the Web service response.

The entries for the conversion of the data might look as follows:

Field	Input
Calculated Variable	`CalcValueFiltered`
Data Type	`System.Double`
Expression	`if('RespValue'<0,0,'RespValue')`

4.2.10.2.3 Example of Configuring a Service Operation

Concept

The following example shows how you create an object for the service operation *Start (StartRequestMessage_sync StartRequest_syn)*.

Activities

1. Click on the *Operation Configuration* tab.
2. Click the service operation *Start (StartRequestMessage_sync StartRequest_syn)* to select it.
3. Choose the *Create Request Message* pushbutton.
   In the screen section *Request Message Configuration*, the system displays the table with the following data:

Field Name	Field Type	Value
`StartRequest_Sync`	`StartRequestMessage_sync`	-

`MessageHeader`	`Basic BusinessDocumentMessageHeader`	*Create Object* pushbutton
`StartRequest`	`StartRequest`	*Create Object* pushbutton

4. Click the `StartRequest` field name and in the *Value* column, choose the *Create Object* pushbutton.
   The system displays two new rows in the table:
   `SiteRef`
   `SfcRequest`
5. Select the `SiteRef` object row by clicking on the name of the `SiteRef` field. Choose the *Create Object* pushbutton.
   The PCo system now shows a new row with the object `Site` (field name `Site` and field type `String`).
6. In the row of the `Site` object in the *Value* column, enter the value **`SITE`** and mark the checkbox in the *Variable* column.
7. Select the row of the object `SfcRequest` and choose the *Create Array* pushbutton in the same row.
   The system shows the *Input Box* dialog box.
8. Enter **`1`** as the array size and choose *OK*.
   The PCo system shows a new row with the field name 0 and the field type `StartRequestSFC`.
9. Select the new row and choose the *Create Object* pushbutton.
   The PCo system shows several new rows with new objects.
10. Keep navigating in this structure until you have found the field names specified in the following table; you do this by marking a row and then choosing the *Create Object* pushbutton. For the string variables, enter the values from the following table:

Field Name	Value	Select Variable Checkbox
SiteRef > *Site*	**`SITE`**	Yes
SfcRef > *Sfc*	**`SFC`**	Yes
ResourceRef > *Resource* > *Value*	**`RESOURCE`**	Yes
OperationRef > *Operation*	**`OPERATION`**	Yes
OperationRef > *Operation*	**`#`**	No
UserRef > *UserID*	**`USER`**	Yes

11. Save your entries.
    The configuration of your *Web Service Destination* for SAP ME is thus completed.
    You can now create an agent instance and a notification. The values you have entered here can then be used in the notification in the *Output* tab. See also: Integration with Third-Party Systems Using Web Services (WSD) [page 66].

4.2.10.3 Advanced Settings (WS Destination System)

Context

You can use the settings on this tab to improve performance during the runtime of PCo if you are using a Web Service Destination. The values in the fields are predefined. You rarely need to change these values.

Procedure

1. You can change the following connection settings:

Field	Description
Use Default Proxy	This checkbox specifies that the default proxy that you defined in the connected system is to be used. The checkbox is selected by default. It is usually not necessary to deselect the checkbox. If you deselect the checkbox, you can speed up the Web service calls, because in this case, for a Web service call, the proxy server and the related rule set do **not** need to be determined.
Bypass Proxy on Local Network	If you select the checkbox, the proxy on the local network is bypassed. The checkbox is not selected by default. **i Note** Since the networks are usually very complex, you should only use this checkbox with the involvement of your local network administrator.
Maximum String Size	This parameter specifies the maximum length of string literals in response messages. This parameter is determined by the *.NET Framework* and is used to prevent denial-of-service attacks (DoS attacks). The default value is 65536 characters. This default value is usually sufficient. If this value needs to be changed, the system issues a corresponding error message.

Field	Description
Maximum Depth	This parameter specifies the maximum number of nesting levels of SOAP messages. This parameter is determined by the *.NET Framework* and is used to prevent DoS attacks. The default value is 32 levels. This default value is usually sufficient. If this value needs to be changed, the system issues a corresponding error message.
Maximum Array Size	This parameter specifies the maximum number of entries that are permitted in an array for response messages. This parameter is determined by the *.NET Framework* and is used to prevent DoS attacks. The default value is 16384 entries. This default value is usually sufficient. If this value needs to be changed, the system issues a corresponding error message.
Maximum Received Message Size	This parameter specifies the maximum size of response messages permitted in bytes. The maximum value that you define here also influences the total memory usage of PCo (during runtime) because the buffers are assigned according to the size of the response messages. The default value is 512 KB . Depending on the scenario used, you can increase or decrease the value. If you decrease the value, the memory usage gets lower. If you increase the value, it is also possible to receive larger response messages. **i Note** In connection with *SAP ME 6.0* in particular, SOAP error messages can exceed 2 MB. Therefore, SAP recommends that you increase the value if there are error messages.
Keep Temporary Files	This checkbox is only used for error analysis. Coding is generated every time a notification message is sent to the *Web Service Destination system*. This generated code is written to a file and can be kept for analysis purposes. If you select the checkbox, the file is stored in a temporary directory, for example, under *C:\Users\<user_name>\AppData\Local\Temp\PCo_Generated_Files*.

2. If you have changed the values in the fields and want to restore the default settings, choose the *Reset to Default* pushbutton.

4.2.11 Functions for Destination Systems

> **i Note**
>
> You cannot change a destination system if its associated agent instance is running.

Add Destination System

1. On the *Plant Connectivity Management Console* screen, choose *Plant Connectivity* > *New* > *Destination System*.
2. Enter the type, name, and a description for the destination system and choose *OK*.

Duplicate Destination System

1. On the *Plant Connectivity Management Console* screen, select the destination system you want and choose the *Duplicate Destination System* pushbutton.
   The system creates a new destination system with the same settings. The name of the original destination system is copied and supplemented by *(1)*.
2. You can change the settings of the new destination system.

Copying and Pasting a Destination System

1. Select the destination system of your choice and choose *Edit* > *Copy* from the menu.
2. Choose *Edit* > *Paste*.
   The system creates a new destination system with the same settings.

Delete Destination System

1. On the *Plant Connectivity Management Console* screen, select a destination system and choose *Edit* > *Delete*.
2. Choose *Yes*.

Rename Destination System

1. On the *Plant Connectivity Management Console* screen, select a destination system and choose *Edit* > *Rename*.
2. Enter the required data and choose *OK*.

Changing the Description of a Destination System

After creating a destination system, you can use this function to change the description of the object at any time by clicking the destination system and choosing *Edit* > *Change Description* in the menu. You can only display the description of a destination system by placing the mouse pointer on this object. PCo then displays the description that you entered when you created the system.

Save Destination System

1. If you have made a change to a destination system, on the *Plant Connectivity Management Console* screen, select a destination system and choose *Plant Connectivity* > *Save*.
2. Choose *Yes*.

Show Usage in Agent Instances

With this function, you can display, for a destination system, the agent instances and notifications in which the destination system is used. Click the destination system you want and choose the *Show Usage in Agent Instances* pushbutton. PCo then displays a dialog box with the where-used list. The function is also available in the context menu.

Context Menu

You can call a context menu for each destination system by clicking on the destination system with the right mouse button.

4.3 Agent Instance

Definition

Agent instances are the central components of PCo. They establish the data flow between the data source and PCo and enable the processing and forwarding of notification messages. Moreover they process queries and method calls.

Agent instances are based on the agent that implements the special source system type and are **each connected with exactly one source system**.

However, you can also **create agent instances without a source system**. Then you can run the agent instance as an OPC UA server or as a Web server.

When you start agent instances, each instance is executed as a standalone Windows service or standalone program in the background. Therefore, you can run multiple agent instances at the same time and independently of each other. The number of agent instances that can be executed at the same time is only restricted by the capacity of the computer on which PCo is installed. You can monitor the status of the individual agent instances at runtime using the Management Console or the remote client. If there are fatal errors, the agent instance stops and is displayed as faulty.

Before you can start an agent instance, you need to make your settings. Configuration of an agent instance depends on its intended purpose and on the agent used. Depending on the functionality of the agent used, specific configuration options for the agent instance are hidden. For example, if you create an agent instance for an agent that does not support notifications, the tab pages for notification processing and for notification message queues are not available.

Structure

The following tabs are available for configuring an agent instance:

- Host Tab [page 413]
- Log Tab [page 415]
- Servers Tab [page 419]
- Tag Query Tab [page 475]
- Subscription Items Tab [page 476]
- Notification Processing Tab [page 481]

The following tabs are also available in the notification process for checking the notification messages:

- *Queued Messages* tab
- *Message Failures* tab
- *Expired Messages* tab
- *Messages for Bundling* tab

For more information about agent instances, see Starting and Stopping an Agent Instance [page 493] and Agent Instance: Displaying Messages [page 490].

4.3.1 Host Tab

Context

You define the settings for the host on this tab.

> **i** Note
>
> If the agent instance is running, the *Host* tab is largely disabled. You can only change the *Log Level* of the running agent instance, which has an immediate effect.

Procedure

1. On the *Plant Connectivity Management Console* screen, select an agent instance and choose the *Host* tab.
2. Enter the necessary data in the **Log Level** field using the following table:

Log Level	Description
Off	No log information is output. This setting is not recommended.
Critical	Only critical messages are logged. This setting is rarely used.
Error	Critical and error messages are logged. This is the default setting.
Warning	Critical, error, and warning messages are logged.
Information	Critical, error, warning, and information messages are logged. Information messages are used for important agent instance events such as starting and stopping the system.
Verbose	Messages of all log levels are logged as well as messages with the status `Verbose`. You should use this setting only if you want to diagnose a vulnerability. **⚠ Caution** Do not use the *Verbose* setting for a longer period. This has a negative impact on system performance and consumes a lot of memory.

> **i Note**
>
> You can also change the log level for a running agent instance if you start the *Management Console* as an *administrator*. With this function, you can, for example, create a more detailed log temporarily when problems occur in productive operation, without first having to stop the agent instance.

> **i Note**
>
> *Warning*, *Error*, and *Critical* messages are also sent to the Windows Event Log.

3. If you select the **Run Agent Instance as an Executable** checkbox, the agent instance runs as a "normal" Windows process instead of as a Windows service. The agent instance runs with the rights of your user account while you are logged on. When you log off, the agent instance is stopped. Please bear this in mind for productive use.

   > **i Note**
   >
   > In this case, steps 4, 5, and 6 are not necessary.

   If you do **not select** this checkbox, the agent instance is run as a **Windows service**.

   > **i Note**
   >
   > A Windows service is created automatically for each agent instance. You can display the list of all services by right-clicking on the computer name in the Windows Explorer, choosing the *Manage* option, and then selecting *Services and Applications*.

4. If the agent instance is to run as a Windows service, you can specify the **service user name** and the **service user password**:
   - The default service user password is set as `\localsystem`.
   - However, you can choose to enter the domain user name instead: `domain\username`.
5. With the help of the following table, enter the necessary information in the **Service Start Mode** field:

Service Start Mode	Description
Automatic	The agent instance is started automatically when the operating system starts.
Automatic (Delayed Start)	The agent instance is started automatically when the operating system starts. However, the start is delayed (default in the Windows system: two minutes). You can use this start mode if there are connection problems between the agent instance and communication partners that also do not start until the operating system is restarted. **i Note** If necessary, check using the *Maintain Dependent Services* pushbutton whether you can model the dependency of the agent instance on other services, in order to improve the reliability of starting the agent instance after the operating system is restarted.

Service Start Mode	Description
Manual	The agent instance is started manually via the *Plant Connectivity Management Console* or the Windows service. This is the default setting.
Disabled	The agent instance is never started.

6. If you want to display or configure dependent services of the agent instance, choose the *Maintain Dependent Services* pushbutton.

   If you add any services, they are started before the agent instance service.

   > **i** Note
   >
   > The following dependent service is set as standard:
   >
   > - *Windows Event Log*

7. Change the value in the **Startup Timeout** field as required. The default setting is 5 minutes. This is the maximum amount of time that can be taken to start the agent instance. After this, the start process is terminated.
8. If necessary, choose a starting group for the agent instance in the **Starting Group** field or create a new one. By assigning agent instances that belong together from a business point of view to one starting group, you can start and stop agent instances together. (See Using Starting Groups for Agent Instances [page 494].)

   > **i** Note
   >
   > You should maintain the relevant attributes of the services of the agent instances entirely in the PCo Management Console. Changes that you make to the Windows services of the agent instances in the **Microsoft Management Console (MMC)** are not included in the PCo configuration. The changes made in the **MMC** can therefore be overwritten by PCo.

4.3.2 Log Tab

Use

PCo writes a separate log for each agent instance. The raw data from the log is saved in a Windows Event Log. The Windows Event Log is a Microsoft component with which you can display and manage event logs of the individual agent instances.

The name of the Windows Event Log that is assigned to the agent instance is displayed in the status bar. You can evaluate the relevant log for each agent instance via the *Management Console*.

> **i** Note
>
> Log data is only written by running agent instances. However, you can access the logs regardless of the agent instance's operation status.

Prerequisites

You have defined which columns you want to display in the log. (See also: Configuring the Log Display for the Agent Instance [page 20].)

Procedure

Selection Criteria

1. On the *Plant Connectivity Management Console* screen, select an agent instance and click *Log*.
2. Use the following table to enter the required data to restrict the selection of required log data:

Field	Description
Start Date	First date and time from which to display log entries. Activate the selection field by selecting the checkbox or pressing the spacebar if the focus is on this field. If you deselect the checkbox, all entries in the log since the start of record keeping are selected.
End Date	Date and time until which log entries are to be displayed. Activate the selection field by selecting the checkbox or pressing the spacebar if the focus is on this field. If you deselect the checkbox, all logs available since the start date are displayed.
Filter Level	Defines the entry types to display. For example, if you only want to display the logged error messages for the event type *Error* and *Critical*, choose the *Error* entry. If you do not specify anything, all message types are selected from the log.
Number of Entries	Indicates the number of entries that are selected by default. This allows you to shorten the runtime when selecting the log. In the status bar, you see whether there are further entries, in addition to the displayed entries, that satisfy the selection conditions.

3. Click *Refresh*.
   The log entries are now selected according to the above dates and filter conditions. The log is always sorted in descending order according to the time of the logged events; in other words, the most recent events are displayed first.

Log Functions (Pushbuttons)

The following functions are provided for the log by means of pushbuttons that are located beside the selection criteria:

Function	Meaning
Delete All Log Entries	If you choose this pushbutton, you can delete all log data.
Export to CSV File	With this pushbutton, you can export log data using **<agent name>_yyyy.mm.dd_hh.mm.ss.csv** to a CSV log file into a folder of the user's choice. You can choose the time period and the filter level of the log data that is to be exported. In the user-specific settings, if required, you can change the separator used to separate the columns of the CSV file. The semicolon is the default separator. (See also: Configuring the Log Display for the Agent Instance [page 20].)
Manage Windows Event Log	By choosing this pushbutton, you start the *Windows Event Viewer* for the Windows log that is assigned to the agent instance. In the *Properties* of this log, you can specify, for example, the maximum size of the log and what should happen when this maximum size is reached.

Log Columns

The following information is provided in the log for the agent instance:

Column	Meaning
Date/Time	Time at which the event was registered by the agent instance. The last three digits represent the milliseconds. The time displayed refers to the time zone of the user that is logged on.
Event Type	Severity of the registered event. A distinction is made between the following event types in PCo: • *Critical* Serious errors that mostly lead to a running agent being stopped • *Error* Common errors • *Warning* Warnings that indicate an unusual agent status • *Information* • *Verbose* Diagnostic messages that can be used in error analysis
Server Name	Name of the server on which the messages were registered. This is usually the computer name on which PCo is installed.

Column	Meaning
Thread ID	ID of the thread in which the event was registered. As a rule, notifications from the data source and queries by the agent are processed in parallel in different threads to increase the data throughput. Using the thread ID, you can group together and assign to each other the events that originate from a specific notification or a specific query.
Process ID	ID of the Windows process that has been assigned to the running agent. The process ID does not change during the runtime of the agent. You can recognize in changes to the process ID if the agent instance has been stopped in the meantime.
Process Name	Name of the Windows process that has been assigned to the running agent instance. The name corresponds as a rule to the name of the agent instance.
Source	Module of the PCo program in which the event has been triggered. This information is mainly of importance for SAP development.
Event ID	Indicates the point in the program at which the event was logged. This information is mainly of importance for SAP development.
Message	Message text The message text is issued in English in most cases.
Stack Trace	Call hierarchy up until the error occurred. The stack trace is only written to the log for exceptions.

Search Capabilities

You can use the search functions to select displayed log entries that fulfill specific search criteria. The following search options are available in the *Search For* field:

- *Messages*
  Log entries whose message text contains a specific character string are selected. No distinction is made here between uppercase and lowercase.
- *Event ID*
  Log entries of a specific event ID are selected.
- *Event Type*
  Log entries of a selected event type are selected.
- *Source*
  Log entries that originate from a selected PCo program are selected.

Select one of the search options and then enter the search criterion. Choose *Execute Search* to execute the search and to automatically select appropriate log entries according to the search criteria.

Additional functions in conjunction with the search functions are provided by means of the following pushbuttons. You can also use these functions if you have selected individual or multiple log entries manually.

Function	Description
First / Previous / Next / Last Selected Log Entry	You can use these arrow keys to go to the first, previous, next, or last selected entry.
Select Log Entries with Same Thread IDs	First select one or multiple log entries using the search function or manually. Then choose this pushbutton to select additional log entries that have the same thread IDs as the previously selected entries. This function makes it easier to identify sequences of related log entries. ⚠ Caution The system can use a thread ID multiple times for different, independent operations. Therefore it is not always certain that two neighboring log entries originate from the same operation.
Copy Selected Log Entries to Clipboard	The system copies the selected log entries to the clipboard. The individual columns are separated by the tab character.

More Information

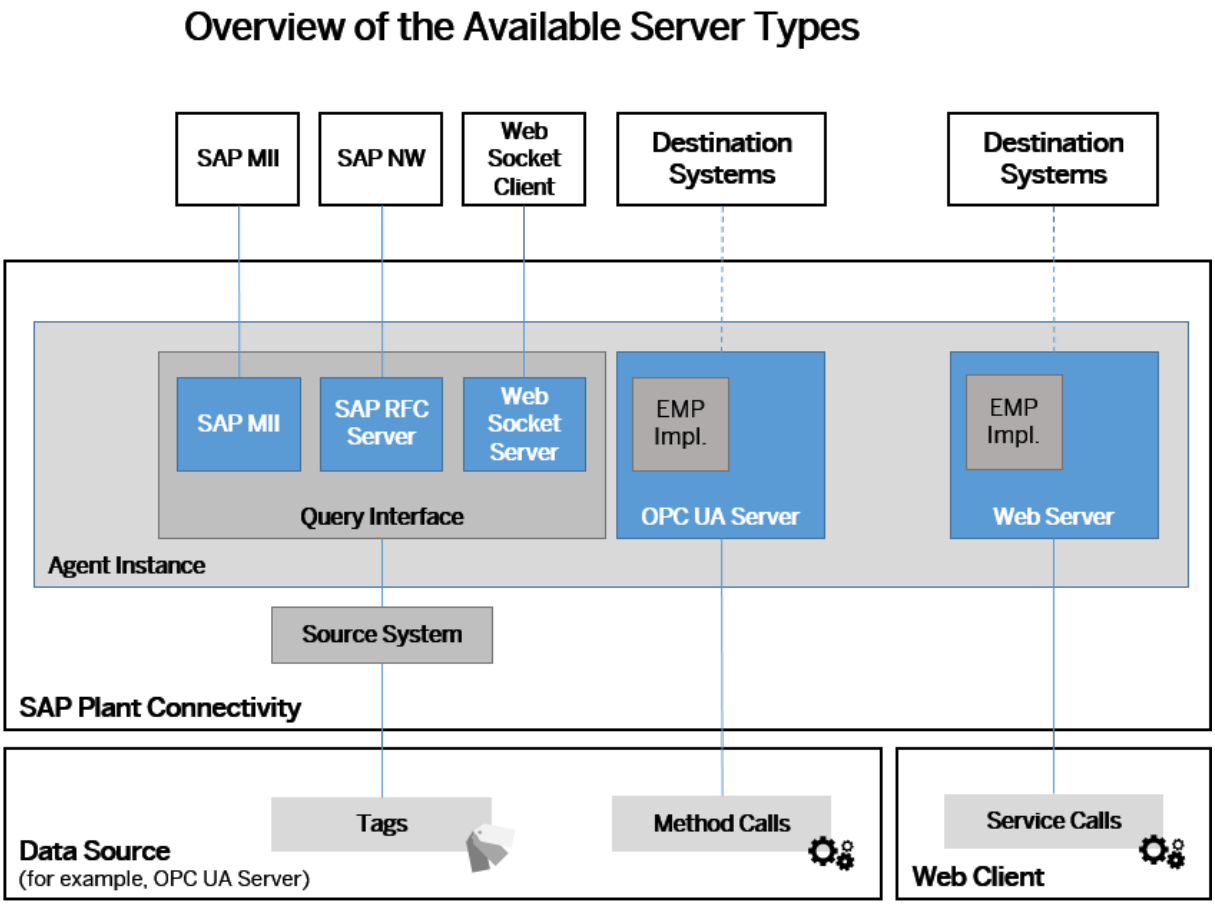
Logging [page 591]

4.3.3 Servers Tab

Use

You make the settings for the various server types supported by PCo on the *Servers* tab of the selected agent instance.

PCo provides various query interfaces for this. The following graphic gives an overview of the various server types.



Key Features

The following *server types* are available:

- SAP MII Query Server
  If you are using an SAP MII system of release 12.2 or higher for the query process in connection with PCo, you need to select the *SAP MII Query Server* checkbox.
  Queries are initiated by an external system; by SAP MII in this case. The data flow is from an external system in the direction of PCo. In this case, PCo is the server. The communication uses ports. TCP/IP ports are used for PCo. See also: SAP MII Query Server [page 421]
- SAP MII query server (before 12.2)
  If you are using an SAP MII system below 12.2 in the query process, you need to select the *SAP MII Query Server (Before 12.2)* checkbox. See also: SAP MII Query Server (Before 12.2) [page 424]
  SAP NW RFC server
  If you select the *SAP NW RFC Server* checkbox, you can use PCo in connection with any NW-based application and all PCo agents. See also: SAP NW RFC Server [page 430]
- SAP ODA RFC server
  If you want to use the SAP ODA functions in the ERP system, you need to select the *SAP ODA RFC Server* checkbox. This setting is only possible in connection with an OPC DA agent. See also: SAP ODA RFC Server [page 426]

- SAP EWM RFC server
  If you select the *SAP EWM RFC Server* checkbox, you can connect SAP EWM systems to PCo and send telegrams using PCo. The settings for the SAP EWM RFC server can only be used in connection with the socket agent. See also: SAP EWM RFC Server [page 434]
- WebSocket server
  If you select the *WebSocket Server* checkbox, you can connect a Web-based client to PCo. See also: WebSocket Server [page 438]
- OPC UA server
  By selecting the *OPC UA Server* checkbox, you can run the agent instance as an OPC UA server that provides methods for execution. See also: OPC UA Server [page 455] and PCo as OPC UA Server and as Web Server [page 54]
- Web server
  By selecting the *PCo Web Server* checkbox, you can run the agent instance as a Web server that provides service operations in the form of methods. (See also: PCo Web Server [page 464] and PCo as OPC UA Server and as Web Server [page 54].)

> **i** Note
>
> You can activate multiple servers of various types at the same time on one agent instance. However, make sure that port numbers are used uniquely. The Management Console issues a warning if port numbers are used multiple times and allows you to check the port numbers are unique when you configure the individual server types.

4.3.3.1 SAP MII Query Server

Context

If you are using an **SAP MII system of release 12.2 or higher** in connection with PCo and you want to use the query process [page 43], you have to select the *SAP MII Query Server* checkbox and make the relevant settings.

Procedure

1. On the *Plant Connectivity Management Console* screen, select an agent instance and click on the *Servers* tab.
2. Select the *SAP MII Query Server* checkbox.

   The PCo system displays the fields where you specify the relevant properties.
3. Make the following settings:

Field	Description
Port	Port for the connection to SAP MII
Security	Here you can specify whether PCo is to communicate with SAP MII using a connection secured with TLS. If you want to use this type of connection, select the *Connection with TLS* option.
Server Certificate	You can enter a server certificate here. PCo can use this certificate to set up a secure connection to MII. For more information about setting up a secure query connection with MII, see Setting Up a Secure Connection for MII Queries [page 422].

4.3.3.1.1 Setting Up a Secure Connection for MII Queries

TLS Connection

If you want to set up the connection between SAP MII and PCo using TLS, you have to make the corresponding settings in both systems. In the security settings of the Management Console, you define a server certificate for the MII query server that the SAP MII system must trust as a client. Conversely, you have to store a client certificate in MII so that MII can authenticate itself to the PCo system as a server. PCo must also trust this certificate.

Generation of Certificates for PCo and MII

You require certificates with a private key for both the MII system and the PCo computer. You can generate the certificates yourself or procure them from your IT organization. It is important that the attribute **CN** (Common Name) contains the name of the respective server or client and that the certificate is suitable for server and client authentication. The *Use of Certificates* section in the *PCo Security Guide* contains instructions for generating a self-signed certificate using the Windows PowerShell.

Import the PCo certificate to the certificate folder *Personal* of the Windows Certificate Store *Local Computer*.

For the MII certificate, create a keystore view of the SAP NetWeaver instance on which SAP MII is installed. Then import the certificate into this keystore view. You can make these settings in NetWeaver in the menu under *Configuration* > *Certificates and Keys*.

Establishing Trust Relationship Between PCo and MII

Export the public key of the PCo server certificate, and import this key into the keystore view of the MII system mentioned above.

You must also export the public key of the MII client certificate and import it into the *Trusted Publishers* certificate folder of the Windows Certificate Store *Local Computer*.

If your IT organization has generated the certificates, you must ensure that the certificates of your intermediate and root certification authorities are also stored in the relevant certificate stores of the Windows system and in the MII keystore view.

Settings in the PCo Management Console

1. In the *SAP MII Settings* screen area, select for the connection a port that is not being used.
2. In the *Security* field, select the *Connection with TLS* setting for the MII server.
3. In the *Server Certificate* field, assign the server certificate with private key generated in the first step to the MII server.
   This certificate should be located in the *Personal* certificate folder of the certificate store *Local Computer*.

Settings in SAP MII

1. In SAP MII, create a data server of type *PCoConnector* in the *Data Services* > *Data Servers* area.
   To set it up, you need the **URL** of the *management service* in PCo. You can find this information in the PCo Management Console when you call the menu under *Tools* > *Options* > *Global Settings* > *Main Service* and choose the pushbutton *Copy WSDL URL to Clipboard*.
2. On the detail screen of the **PCoConnector**, enter the following information:

Detail Screen PCoConnector

Field	Description
Agent Name	The name of the agent instance in PCo.
Agent Port	The port number that you defined in the MII query server settings of the agent instance.
Agent IP	Either the name of the PCo computer or its IP address.
Certificate Key Store	The name of the keystore view in which you have stored the MII client certificate and the public key of the PCo server certificate.
Certificate Name	Can be left empty if the MII certificate contains the server name in the attribute CN.
Use SSL	The checkbox must be selected.

> **i Note**
>
> You can also have the system transfer the public key of the MII certificate to the PCo system when the connection is first established between MII and PCo. When the connection is established, it is stored in the directory `C:\ProgramData\SAP\PCo\CertificateStores\RejectedCertificates\certs`, and can be imported from there to the Windows certificate folder for trusted certificates.

4.3.3.2 SAP MII Query Server (Before 12.2)

Context

If you are using an **SAP MII system lower than version 12.2** and you want to use the query process [page 43], you need to select the *SAP MII Query Server (before 12.2)* server type on the Servers tab [page 419] and make the relevant settings.

Procedure

1. On the *Plant Connectivity Management Console* screen, select an agent instance and click on the *Servers* tab.
2. Select the server type *SAP MII Query Server (before 12.2)*.

   The fields needed for specifying the attributes for this server type are then displayed.
3. For the information you need to enter, use the following table:

Field	Description
Port	The port for the connection to be used is proposed. **i Note** Ports must not be used multiple times. A port is to be used for one agent instance only.

Field	Description
Include Quality	In this field, you can specify whether and how quality information is also to be provided for the measurement values that are to be transmitted. You have the following options: ○ *No Quality Information* ○ *Quality Information as String* ○ *Numeric Quality Information* ○ *Quality Information as String and Numeric*
Calculate Minimum and Maximum for Intervals Without Value Changes	Specifies that the value from one of the neighboring intervals is to be used for an evaluation interval in which there are no value changes. **i Note** The logic for calculating the minimum and maximum is determined by the data source because the statistical evalulations are performed by the data source. The indicator is not set as standard. The indicator only takes effect if you have set the UDC connector in MII.
Assign Statistical Values to End of Interval	Specifies that the calculated statistical values are always assigned to the end of the evaluation interval. The indicator is not set as standard. The indicator only takes effect if you have set the UDC connector in MII. **i Note** IP21 and Proficy systems always return the results of the statistical calculations at the end of an evaluation interval. In these cases, the indicator has no effect.

i Note

Note that you need to define unique tag names in the namespace of the data source when using this server type together with a UDC data server in the SAP MII system. For that reason, set up the connection between MII and PCo in the query process using a *PCo Connector* if your SAP MII system supports this type of data server.

4.3.3.3 SAP ODA RFC Server

Prerequisites

You have created an *RFC destination system* for the ERP or S/4HANA system (see RFC Destination System [page 347]).

Context

If you want to use the SAP ODA functions in the ERP system, you need to select the *SAP ODA RFC Server* checkbox on the Servers tab [page 419]. This setting is only possible in connection with an OPC DA agent. For more information, see the ERP or S/4HANA documentation under **SAP OPC Data Access (SAP ODA)**.

> **i** Note
>
> You always need to make the settings on the *Servers* tab in connection with *SAP ODA*, irrespective of whether you are using queries or the remote subscription function.

Procedure

1. On the *Servers* tab, select the *SAP ODA RFC Server* checkbox.
2. For SAP ODA, you make the following settings:

Field	Description
Registration Using Message Server	Select this checkbox if you want to set up the connection using a message server with load balancing instead of using an RFC gateway. If you want to set up the connection using a message server, you need to specify the host name of the message server, the system ID, and the logon group. For a connection using an RFC gateway, you need to specify the gateway host name and the gateway service. The Management Console hides the entries that are not required, in accordance with your selection.

Field	Description
Program ID	The *program ID* is used to identify the RFC destination in transaction `SM59`, from which PCo is to receive the requirements. The *program ID* must be unique and must be the same as that specified in the ERP system in transaction `SM59` on the *Technical Settings* tab.
SAP Gateway Host	Specifies the host name of the RFC gateway. The connection is set up via the *Gateway Host*. This is the machine on which the gateway process runs. You also need to specify the gateway host in the ERP system in transaction `SM59` on the *Technical Settings* tab. You need to specify the gateway host if you do not want to set up the connection using a message server. **i Note** For more information, see the SAP NetWeaver documentation under **Security Settings in the SAP Gateway** and **Checking Security Configuration**.
SAP Gateway Service	Specify either **`sapgw<SysNr>`** or the port number directly if the service is not defined in the service file.
SAP Message Server Host	Specifies the name of the message server if the connection is to be set up using load balancing.
SAP Logon Group	Here you specify the name of the logon group that is to be used for load balancing.
SAP System ID	The system ID of the SAP system to which you want to set up a connection.
SAP Router	The SAP Router makes it possible to connect through a firewall. Enter the SAP router parameters in the following format: **`/H/hostname//S/portnumber/H/`**
Number of Threads	Specifies the maximum number of parallel incoming RFC requests that the RFC server can process. If you enter **5** for example, five users can send a request to the ERP system in parallel.
Compression	You use this checkbox to specify that table values larger than 8 KB are to be compressed in the RFC interface.
SNC	

Field	Description
Enabled	If you select this checkbox, the **Secure Network Communication** component is enabled. For more information, see SNC Settings [page 92] and, in the SAP NetWeaver documentation, see *Secure Network Communications (SNC)*.
User SNC Name	SNC name of the user. You also need to enter this name in the ERP system in transaction `SM59` in the *Partner* field, for example, **`p:CN=D0XXXXX, O=SAP-AG, C=DE`**. **i Note** The user name for SNC is in *User Maintenance* (transaction SU01) on the *SNC* tab.
Server SNC Name	You enter the name of the connected ERP system, for example, **`p:CN=UI3, O=SAP-AG, C=DE`**. **i Note** You can find the SNC server name in *SAP Logon* in the context menu of the relevant system entry under *Properties* > *Network tab* in the *SNC Name* field.
QoP (Quality of Protection)	Specifies the quality of protection (QoP) of the data transfer: ○ **1** *Only Secure Authentication (Digital Signature)* ○ **2** *Protection of the Integrity of the Data (Digital Signature and Encryption)* ○ **3** *Confidentiality of the Data (Digital Signature, Encryption, and User Authentication)* ○ **8** *Default (Default Value Defined by the ERP System)*, profile parameter **snc/data_protection/use** ○ **9** *Maximum Available (Maximum Value Supported by the Security Product)*, profile parameter **snc/data_protection/max** If the value is below the minimum required value (profile parameter **snc/data_protection/min**), it is increased automatically. If the value is above the maximum available value (profile parameter **snc/data_protection/max** or maximum value of the external security product), the communication is terminated.

Field	Description
Library	Here you enter the full path and the name of the security library of a non-SAP product. This library is then used for the SNC communication. If you use **Single Sign-On (SSO)**, you do not need to enter anything here. You can use the path C:\Program\Files\SECUDE\Office\Security\secude\dll to access the SSO library. If in conjunction with SSO, the path differs from the standard path, enter this here.
Tracing	
Enabled	If you select this checkbox, the *Trace* function is enabled. In this case, the RFC communication is traced in a file with this agent instance. The tracing file is stored in the current working directory of the PCo agent instance.
Level	States the trace level of the *.Net Connector (NCo)*. The trace level is the degree of detail in which steps performed during message processing are traced by the system. The following settings are possible: ○ Level 1 With this setting, only the *Remote Function Calls* are traced. ○ Level 2 With this setting, the calls from public *API Methods* are also traced. ○ Level 3 With this setting, the calls from internal *API Methods* are traced in addition to the calls mentioned above. ○ Level 4 With this setting, the *HEX Dumps* are traced for the RFC log in addition to all calls previously mentioned.
Repository System	Here you enter the destination system that you have created for the connected ERP system.

3. Choose *Test Registration* to test the registration at the SAP Gateway. For a successful test, you only need to make entries in the following fields:
   - *Program ID*
   - *SAP Gateway Host*
   - *SAP Gateway Service*
   - Or, when using **load balancing**:
     - *SAP Message Server Host*
     - *SAP System ID*
     - *SAP Logon Group*

Results

You have specified the connection data for SAP ODA. You can now enter the required connection data in the ERP system in transaction `SM59`.

4.3.3.4 SAP NW RFC Server

Prerequisites

You have created a destination system of the type *RFC destination system* for the Business Suite system (see RFC Destination System [page 347]).

Context

If you want to connect a Business Suite system such as ERP to PCo and use the query process, you need to select the *SAP NW RFC Server* checkbox on the Servers tab [page 419].

Procedure

1. On the *Servers* tab, select the *SAP NW RFC Server* checkbox.
2. You make the following settings:

Field	Description
Registration Using Message Server	Select this checkbox if you want to set up the connection using a message server with *load balancing* instead of using an RFC gateway. If you want to set up the connection using a message server, you need to specify the host name of the message server, the *system ID*, and the *logon group*. For a connection using an RFC gateway, you need to specify the gateway host name and the gateway service. The Management Console hides the entries that are not required, in accordance with your selection.

Field	Description
Program ID	The *Program ID* is used to identify the RFC destination in transaction `SM59`, from which PCo is to receive the requirements. The *Program ID* must be unique and must be the same as that specified in the SAP system in transaction `SM59` on the *Technical Settings* tab.
SAP Gateway Host	Specifies the host name of the RFC gateway. The connection is set up via the *Gateway Host*. This is the machine on which the gateway process runs. You also need to specify the *gateway host* in the ERP system in transaction `SM59` on the *Technical Settings* tab. You need to specify the *gateway host* if you do not want to set up the connection using a message server. **i Note** For more information about security settings for the SAP Gateway, see the *SAP NetWeaver* documentation under **Security Settings in the SAP Gateway** and **Checking Security Configuration**.
SAP Gateway Service	Specify either **`sapgw<SysNr>`** or the port number directly if the service is not defined in the service file.
SAP Message Server Host	Specifies the name of the message server if the connection is to be set up using **load balancing**.
SAP Logon Group	Here you specify the name of the logon group that is to be used for load balancing.
SAP System ID	The system ID of the SAP system to which you want to set up a connection.
SAP Router	The SAP Router makes it possible to connect through a firewall. Enter the SAP router parameters in the following format: **`/H/hostname//S/portnumber/H/`**
Number of Threads	Specifies the maximum number of parallel incoming RFC requests that the RFC server can process. If you enter **5** for example, five users can send a request to the SAP system in parallel.
Compression	You use this checkbox to specify that table values larger than 8 KB are to be compressed in the RFC interface.

Field	Description
SNC	
Enabled	If you select this checkbox, the **Secure Network Communication** component is enabled. For more information, see SNC Settings [page 92] and the *SAP NetWeaver* documentation under *Secure Network Communications (SNC)*.
User SNC Name	SNC name of the user. You also need to enter this name in the Business Suite system in transaction `SM59` in the *Partner* field, for example, **`p:CN=D0XXXXX, O=SAP-AG, C=DE`**. **i Note** This name is in *User Maintenance* (transaction SU01) on the *SNC* tab.
Server SNC Name	You enter the name of the connected SAP system, for example, **`p:CN=UI3, O=SAP-AG, C=DE`**. **i Note** You can find the SNC server name in *SAP Logon* in the context menu of the relevant system entry under *Properties* > *Network tab* in the *SNC Name* field.
QoP (Quality of Protection)	Specifies the quality of protection (QoP) of the data transfer: ○ **1** *Only Secure Authentication (Digital Signature)* ○ **2** *Protection of the Integrity of the Data (Digital Signature and Encryption)* ○ **3** *Confidentiality of the Data (Digital Signature, Encryption, and User Authentication)* ○ **8** *Default (Default Value Defined by the ERP System)*, profile parameter **snc/data_protection/use** ○ **9** *Maximum Available (Maximum Value Supported by the Security Product)*, profile parameter **snc/data_protection/max** If the value is below the minimum required value (profile parameter **snc/data_protection/min**), it is increased automatically. If the value is above the maximum available value (profile parameter **snc/data_protection/max** or maximum value of the external security product), the communication is terminated.

Field	Description
Library	Here you enter the full path and the name of the security library of a non-SAP product. This library is then used for the SNC communication. If you use **Single Sign-On (SSO)**, you do not need to enter anything here. You can use the path C:\Program\Files\SECUDE\Office\Security\secude\dll to access the SSO library. If in conjunction with SSO, the path differs from the standard path, enter this here.
Trace Settings	
Enabled	If you select this checkbox, the *Trace* function is enabled. In this case, the RFC communication is traced in a file with this agent instance. The tracing file is stored in the current working directory of the PCo agent instance.
Level	States the trace level of the *.NetConnector*. The trace level is the degree of detail in which steps performed during message processing are traced by the system. States the trace level of the *.Net Connector (NCo)*. The trace level is the degree of detail in which steps performed during message processing are traced by the system. The following settings are possible: ○ Level 1 With this setting, only the *Remote Function Calls* are traced. ○ Level 2 With this setting, the calls from public *API Methods* are also traced. ○ Level 3 With this setting, the calls from internal *API Methods* are traced in addition to the calls mentioned above. ○ Level 4 With this setting, the *HEX Dumps* are traced for the RFC log in addition to all calls previously mentioned.
Repository System	Here you enter the destination system that you have created for the connected SAP system.

3. Choose *Test Registration* to test the registration at the SAP Gateway. For a successful test, you only need to make entries in the following fields:
   - *Program ID*
   - *SAP Gateway Host*
   - *SAP Gateway Service*
   - Or, when using **load balancing**:
     - *SAP Message Server Host*

- *SAP System ID*
- *SAP Logon Group*

Next Steps

For more information about the integration of PCo with the SAP Business Suite, see Connection of External Data Sources to Business Suite Applications [page 87] and the note 1576651.

4.3.3.5 SAP EWM RFC Server

Prerequisites

You have created a destination system of the type *RFC Destination* for the EWM system (see RFC Destination System [page 347]).

Context

If you select the *SAP EWM RFC Server* checkbox on the Servers tab [page 419], you can connect SAP EWM systems to PCo and send telegrams using PCo.

> **i Note**
>
> The settings for the SAP EWM RFC server can only be used in connection with the socket agents.

Procedure

1. Select this destination system on the *Servers* tab.
2. For SAP EWM, you make the following settings:

Field	Description
Registration Using Message Server	Select this checkbox if you want to set up the connection using a message server with load balancing instead of using an RFC gateway. If you want to set up the connection using a message server, you need to specify the *host name* of the message server, the *system ID*, and the *logon group*. For a connection using an RFC gateway, you need to specify the *gateway host name* and the *gateway service*. The Management Console hides the entries that are not required, in accordance with your selection.
Program ID	The *Program ID* is used to identify the RFC destination in transaction `SM59`, from which PCo is to receive the requirements. The *Program ID* must be unique and must be the same as that specified in the EWM system in transaction `SM59` on the *Technical Settings* tab.
SAP Gateway Host	Specifies the host name of the RFC gateway. The connection is set up via the *Gateway Host*. This is the machine on which the gateway process runs. You also need to specify the gateway host in the EWM system in transaction `SM59` on the *Technical Settings* tab. You need to specify the gateway host if you do not want to set up the connection using a message server. **i Note** For more information, see the SAP NetWeaver documentation under **Security Settings in the SAP Gateway** and **Checking Security Configuration**.
SAP Gateway Service	Specify either **`sapgw<SysNr>`** or the port number directly if the service is not defined in the service file.
SAP Message Server Host	Specifies the name of the message server if the connection is to be set up using load balancing.
SAP Logon Group	Here you specify the name of the logon group that is to be used for load balancing.
SAP System ID	The system ID of the SAP system to which you want to set up a connection.
SAP Router	The SAP Router makes it possible to connect through a firewall. Enter the SAP router parameters in the following format: **`/H/hostname//S/portnumber/H/`**

Field	Description
Number of Threads	Specifies the maximum number of parallel incoming RFC requests that the RFC server can process. If you enter **5** for example, five users can send a request to the EWM system in parallel.
Compression	You use this checkbox to specify that table values larger than 8 KB are to be compressed in the RFC interface.
SNC	
Enabled	If you select this checkbox, the **Secure Network Communication** component is enabled. For more information, see SNC Settings [page 92] and, in the *SAP NetWeaver* documentation, see *Secure Network Communications (SNC)*.
User SNC Name	SNC name of the user You also need to enter this name in the Business Suite system in transaction `SM59` in the *Partner* field, for example, **`p:CN=D0XXXXX, O=SAP-AG, C=DE`**. **i Note** You can find this name in *User Maintenance* (transaction SU01) on the *SNC* tab, or alternatively in the EWM system menu under *System* > *Status*.
Server SNC Name	You enter the server name of the connected EWM system, for example, **`p:CN=UI3, O=SAP-AG, C=DE`**. **i Note** You can find the SNC server name in *SAP Logon* in the context menu of the relevant system entry under *Properties* > *Network tab* in the *SNC Name* field.

Field	Description
QoP (Quality of Protection)	Specifies the quality of protection (QoP) of the data transfer: ○ **1** *Only Secure Authentication (Digital Signature)* ○ **2** *Protection of the Integrity of the Data (Digital Signature and Encryption)* ○ **3** *Confidentiality of the Data (Digital Signature, Encryption, and User Authentication)* ○ **8** *Default (Default Value Defined by the ERP System)*, profile parameter **snc/data_protection/use**. ○ **9** *Maximum Available (Maximum Value Supported by the Security Product)*, profile parameter **snc/data_protection/max**. If the value is below the minimum required value (profile parameter **snc/data_protection/min**), it is increased automatically. If the value is above the maximum available value (profile parameter **snc/data_protection/max** or maximum value of the external security product), the communication is terminated.
Library	Here you enter the full path and the name of the security library of a non-SAP product. This library is then used for the SNC communication. If you use **Single Sign-On (SSO)**, you do not need to enter anything here. You can use the path C:\Program\Files\SECUDE\Office\Security\secude\dll to access the SSO library. If in conjunction with SSO, the path differs from the standard path, enter this here.
Trace Settings	
Enabled	If you select this checkbox, the *Trace* function is enabled. In this case, the RFC communication is traced in a file with this agent instance. The tracing file is stored in the current working directory of the PCo agent instance.

Field	Description
Level	States the trace level of the *.Net Connector (NCo)*. The trace level is the degree of detail in which steps performed during message processing are traced by the system. The following settings are possible: ○ Level 1 With this setting, only the *Remote Function Calls* are traced. ○ Level 2 With this setting, the calls from public *API Methods* are also traced. ○ Level 3 With this setting, the calls from internal *API Methods* are traced in addition to the calls mentioned above. ○ Level 4 With this setting, the *HEX Dumps* are traced for the RFC log in addition to all calls previously mentioned.

3. Choose *Test Registration* to test the registration at the SAP Gateway. For a successful test, you need to make entries in the following fields:
   - *Program ID*
   - *SAP Gateway Host*
   - *SAP Gateway Service*
   - Or, when using **load balancing**:
     - *SAP Message Server Host*
     - *SAP System ID*
     - *SAP Logon Group*

4.3.3.6 WebSocket Server

Context

You can connect PCo as a WebSocket server to Web applications. The following processes are supported:

- Notification Process
  The events provided by the source system can be displayed directly on modern user interfaces (for example, HTML5 interfaces) using the notification process. (See: Settings for the Notification Process with WebSocket [page 441].)

> **Example**
>
> You can use SAP MII's *Self Service Composition Environment (SSCE)* as a user interface. MII can send specific JSON messages to PCo to subscribe to tags. PCo then sends the changes to the user interface continuously.

- Query Process
  The connected client can send messages directly to the PCo WebSocket server. For example, you can query a specific tag in an OPC source system using a JSON retrieve message. PCo processes the query and returns a corresponding result.

  > **Note**
  >
  > A WebSocket destination system is not needed in the query process.

Procedure

1. Select the *WebSocket Server* checkbox and make the following connection settings:

Field	Description
Allow Changes to Tag Values	Indicates that the tag values from the source system can be changed. If you select the checkbox, the tag values of the data source can be changed using store queries if the tag can be changed in the source system.
Unsecured Protocol	Select this connection setting if you want to use an unsecured protocol.
Secured Protocol (WSS)	Select this connection setting if you want the communication to run with a secured protocol. In this case, you need to assign a certificate in the *Certificate* field.
Port	Here you need to enter the PCo port through which the WebSocket connection is to be established and kept alive. **Example** SAP MII can be connected to PCo via a WebSocket. The MII system needs to be registered at the PCo port. To do this, in SAP MII, go to *Data Service* > *Data Server* > *Connection* and specify the PCo port in the *WebSocket Port* field.

Field	Description
Server Certificate	When using a secured protocol, you must assign a certificate here from the certificate store. You can use this certificate to identify PCo to the WebSocket-based client.
Request Client Certificate	Specifies that the client has to identify itself to PCo with a certificate. In the case of a WebSocket connection, the client is the Internet browser. If the indicator is set, the PCo server can ask that the client identifies itself with a certificate. This allows you to increase the security of the connection. However, for some browsers, it is not easily possible to request client certificates. In this case, deactivate the requesting of client certificates by deselecting the indicator. **i Note** For a secure connection between the server (PCo) and a client, the server identifies itself with a certificate that you can assign here on the tab. The client can now check if the certificate is trustworthy. A certificate is trustworthy under the following circumstances: ○ The certificate is in the certificate store of the PCo server ○ Or the root certificate from which the server certificate has been issued is in the client's certificate store (in this case, this is the certificate store of the Internet browser)
Client Root Certificate	In this field, you can specify for the client the root certificate that is to be used to authenticate the client. The **client certificate** inherits the trustworthiness from the trusted root certificate. PCo only allows client certificates for the connection to the PCo WebSocket server that inherit from the specified root certificate. If you do **not** specify a root certificate, PCo allows all client certificates that inherit from a root certificate from Microsoft's trusted root certificate store (for example, Verisign) or from another trusted certification authority.

Field	Description
Message Format	Here you enter the data format for transferring the notification messages. The following data formats are available: ○ JSON ○ XML **i Note** In the case of queries, PCo responds automatically in the format in which the query was made by the client.

2. To create a destination system for the WebSocket, choose the *Create Destination System* function key.

   PCo creates the destination system automatically and displays the information *Destination system used in configuration* in the *Status* field. If no destination system exists, the information *Destination system not created* is displayed in the *Status* field.

   i Note

   You can then select this destination system on the *Destinations* tab of the notification. The destination system is not displayed in the *Destination Systems* area, however.

   The destination system enables PCo to send notifications later to all Web applications that have registered via the port you specified.

3. To delete the destination system, you can choose the *Delete Destination System* pushbutton. You can only delete the destination system if it is not being used in a notification.

Related Information

Connection of Web Applications via WebSocket [page 87]
Settings for the Notification Process with WebSocket [page 441]
JSON Messages for WebSocket Connections [page 442]

4.3.3.6.1 Settings for the Notification Process with WebSocket

This document describes the settings for the notification process.

Settings

1. On the *Servers* tab for the agent instance, select the *WebSocket Server* checkbox and make the settings. You specify the WebSocket destination system here too. (See also: WebSocket Server [page 438]

2. Create a **notification template** by selecting the agent instance and choosing *Add Notification* in the context menu.
3. In the *Add Notification* dialog box that appears, select the option *Notification Template for Remote Subscriptions*. Choose the *Next Step* pushbutton.
4. Enter the notification name in the dialog box that appears. This name must **not** be `Template`. You can enter a description if necessary. Then choose OK.
   The notification tabs are displayed. You do not have to enter anything on the *Notification*, *Remote Subscription*, or *Message Delivery* tabs.
5. Choose the *Destinations* tab.
6. In the *Destination System Type* field select the entry */Destination System for WebSocket* and enter a name. Choose OK.
7. Save your entries.
   PCo assigns the WebSocket destination system and creates the notification template.

Result

You can now start the agent instance. PCo then connects with the WebSocket client.

In the WebSocket client you can now send JSON messages, for example, to search for data or subscribe to tags. PCo sends the data to the connected user interface.

4.3.3.6.2 JSON Messages for WebSocket Connections

PCo supports the following six JSON messages:

Message	Description
`GetTemplates`	This message lists all notification templates for the running agent instances for which you have created a WebSocket destination system. **Example** `{ Action: "GetTemplates" }`
`GetFeatures`	This message lists all functions that are supported by the running agent instance. **Example** `{ Action: "GetFeatures" }`

Message	Description

`Subscribe`	With this message, you can subscribe to the current WebSocket to receive notifications by using the given template.
	
	Sample Code
	
	`SUBSCRIBE command in JSON Format:`
	`{`
	`  Action: "Subscribe",`
	`  Params: {`
	`  TagNames: [`
	`  { Name: "Data Type Examples.16 Bit Device.Registers.Double1", IsNative:true, "Alias":"Double1"} ,`
	`  { Name: "Data Type Examples.16 Bit Device.Registers.Short1", IsNative:true, "Alias":"Short1"},`
	`  ],`
	`  "Template":"myTemplate"`
	`  }`
	`}`

`Unsubscribe`	With the Unsubscribe message you can remove subscriptions. You can specify that the WebSocket is no longer subscribed to specific tags or entire notifications.
	
	If you want the WebSocket client to remove the subscription for the entire notification, the parameters must not be set.
	
	To reduce the load on the source system, the Web clients need to send this command before the WebSocket closes (when you leave the website). Otherwise, the subscription for the notification is maintained.
	
	Sample Code
	
	`UNSUBSCRIBE command in JSON Format:`
	`{`
	`  Action: "Unsubscribe",`
	`  Params: {`
	`  TagNames: [`
	`  { Name: "Data Type Examples.16 Bit Device.Registers.Double1", IsNative:true, "Alias":"Double1"} ,`
	`  { Name: "Data Type Examples.16 Bit Device.Registers.Short1", IsNative:true, "Alias":"Short1"},`
	`  ],`
	`  }`
	`}`

Message	Description

`Reconnect`	With the Reconnect message, you can reconnect the Web-Socket with an existing notification using the given handle.
	Sample Code
	`Reconnect command in JSON Format:`
	`{`
	`Action: "Reconnect",`
	`Params: { Handle : "87726337-1d44-47b3-8f65-91508b6c085e" }`
	`}`

`Retrieve`	With this message you can read tag values for the query process.
	Sample Code
	`RETRIEVE command in JSON Format:`
	`{`
	`Action: "Retrieve",`
	`Params: {`
	`TagNames: [`
	`{ Name: "Data Type Examples.16 Bit Device.K Registers.Short1", IsNative:true, Alias:"Short1"},`
	`{ Name: "Data Type Examples.16 Bit Device.K Registers.Short2", IsNative:true, Alias:"Short2"}`
	`],`
	`}`
	`}`

Message	Description

`Retrieve` (with Metadata and Secondaries)	With this message you can read tag values (with metadata and secondaries) for the query process.
	Sample Code
	`RETRIEVE command with Metadata and Secondaries in JSON Format:`
	`{`
	`Action: "Retrieve",`
	`Params: {`
	`TagNames: [`
	`{ Name: "Data Type Examples.16 Bit Device.K Registers.Short1", IsNative:true, Alias:"Short1"},`
	`{ Name: "Data Type Examples.16 Bit Device.K Registers.Short2", IsNative:true, Alias:"Short2"}`
	`],`
	`MetaDataList: [`
	`"Item Description",`
	`"DDE Access Name"`
	`],`
	`SecondaryList: [`
	`"Quality",`
	`"Item Timestamp"`
	`]`
	`}`
	`}`

Message	Description

`Response`	With this message you can respond to retrieve commands. The response contains metadata and secondaries.
	Sample Code
	`Response for RETRIEVE command with Metadata and Secondaries in JSON Format`
	`{ "TagValues":[{ "Alias":"Short1", "DateTime":"\/ Date(1435063691578)\/", "Value":"0", "MetaData":[{ "Name":"Item Description", "Value":"16-Bit signed integer" }, { "Name":"DDE Access Name", "Value":"Data Type Examples_16 Bit Device" }], "Secondary":[{ "Name":"Quality", "Value":"Good, Non-specific, Not limited" }, { "Name":"Item Timestamp", "Value":"23.06.2015 12:48:11" }] }, { "Alias":"Short2", "DateTime":"\/ Date(1435063691578)\/", "Value":"0", "MetaData":[{ "Name":"Item Description", "Value":"16-Bit signed integer" }, { "Name":"DDE Access Name", "Value":"Data Type Examples_16 Bit Device"`

Message	Description
	`}, ], "Secondary":[ { "Name":"Quality", "Value":"Good, Non-specific, Not limited" }, { "Name":"Item Timestamp", "Value":"23.06.2015 12:48:11" } ] } ], "Success":true, "Error": { "Code":null, "Message":{"Value":null} } }`
`Store`	With this message, you can write tag values using the given time stamp. **Sample Code** `STORE command in JSON Format: { Action: "Store", Params: { TagValues: [{ TagName: { Name: 'Data Type Examples.16 Bit Device.K Registers.Short1', IsNative: true, Alias: 'Test' }, Value: '111', DateTime: '2015-03-10T12:00:00' }, { TagName: {Name: 'Data Type Examples.16 Bit Device.K Registers.Short2', IsNative: true, Alias: 'Test2' }, Value: '222', DateTime: '2015-03-10T12:00:00' }] } }`

4.3.3.6.3 XML Messages for WebSocket Connections

PCo supports the following XML messages:

XML Messages

Message	Description

`GetTemplates`	This message lists all notification templates for the running agent instances for which you have created a WebSocket destination system.
	Sample Code
	`GetTemplates command in XML Format:`
	`<?xml version="1.0" encoding="utf-8"?>`
	`<GetTemplatesMessage xmlns="urn:sap.com:pco.contracts" />`

`GetFeatures`	This message lists all functions that are supported by the running agent instance.
	Sample Code
	`GetFeatures command in XML Format:`
	`<?xml version="1.0" encoding="utf-8"?>`
	`<GetFeaturesMessage xmlns="urn:sap.com:pco.contracts" />`

Message	Description

`Subscribe`	With this message, you can subscribe to the current WebSocket to receive notifications by using the given template.
	Sample Code
	`SUBSCRIBE command in XML Format:`
	`<?xml version="1.0" encoding="utf-8"?>`
	`<SubscribeMessage xmlns="urn:sap.com:pco.contracts">`
	`<TagNames>`
	`<Tag xmlns="urn:sap.com:pco.contracts.items">`
	`<Name>Data Type Examples.16 Bit Device.K Registers.Double1</Name>`
	`<IsNative>true</IsNative>`
	`<Alias>Double1</Alias>`
	`</Tag>`
	`<Tag xmlns="urn:sap.com:pco.contracts.items">`
	`<Name>Data Type Examples.16 Bit Device.K Registers.Short1</Name>`
	`<IsNative>true</IsNative>`
	`<Alias>Short1</Alias>`
	`</Tag>`
	`</TagNames>`
	`<Template>myTemplate</Template>`
	`</SubscribeMessage>`

Message	Description

`Unsubscribe`	With the Unsubscribe message you can remove subscriptions. You can specify that the WebSocket is no longer subscribed to specific tags or entire notifications.
	
	If you want the WebSocket client to remove the subscription for the entire notification, the parameters must not be set.
	
	To reduce the load on the source system, the Web clients need to send this command before the WebSocket closes (when you leave the website). Otherwise, the subscription for the notification is maintained.
	
	Sample Code
	
	`UNSUBSCRIBE command in XML Format:`
	`<?xml version="1.0" encoding="utf-8"?>`
	`<UnsubscribeMessage xmlns="urn:sap.com:pco.contracts">`
	`<TagNames>`
	`<Tag xmlns="urn:sap.com:pco.contracts.items">`
	`<Name>Data Type Examples.16 Bit Device.K Registers.Double1</Name>`
	`<IsNative>true</IsNative>`
	`<Alias>Double1</Alias>`
	`</Tag>`
	`<Tag xmlns="urn:sap.com:pco.contracts.items">`
	`<Name>Data Type Examples.16 Bit Device.K Registers.Short1</Name>`
	`<IsNative>true</IsNative>`
	`<Alias>Short1</Alias>`
	`</Tag>`
	`</TagNames>`
	`</UnsubscribeMessage>`

`Reconnect`	With the Reconnect message, you can reconnect the WebSocket with an existing notification using the given handle.
	
	Sample Code
	
	`Reconnect command in XML Format:`
	`<?xml version="1.0" encoding="utf-8"?>`
	`<ReconnectMessage xmlns="urn:sap.com:pco.contracts">`
	
	`<Handle>87726337-1d44-47b3-8f65-91508b6c085e</Handle>`
	`</ReconnectMessage>`

Message	Description

`Retrieve`	With this message you can read tag values for the query process.
	Sample Code
	`RETRIEVE command in XML Format:`
	`<?xml version="1.0" encoding="utf-8"?>`
	`<RetrieveMessage xmlns="urn:sap.com:pco.contracts">`
	`<TagNames>`
	`<Tag xmlns="urn:sap.com:pco.contracts.items">`
	`<Name>Data Type Examples.16 Bit Device.K Registers.Short1</Name>`
	`<IsNative>true</IsNative>`
	`<Alias>Short1</Alias>`
	`</Tag>`
	`<Tag xmlns="urn:sap.com:pco.contracts.items">`
	`<Name>Data Type Examples.16 Bit Device.K Registers.Short2</Name>`
	`<IsNative>true</IsNative>`
	`<Alias>Short2</Alias>`
	`</Tag>`
	`</TagNames>`
	`</RetrieveMessage>`

Message	Description
`Retrieve` (with Metadata and Secondaries)	With this message you can read tag values (with metadata and secondaries) for the query process. **Sample Code** RETRIEVE command with Metadata and Secondaries in XML Format: <?xml version="1.0" encoding="utf-8"?> <RetrieveMessage xmlns="urn:sap.com:pco.contracts"> <TagNames> <Tag xmlns="urn:sap.com:pco.contracts.items"> <Name>Data Type Examples.16 Bit Device.K Registers.Short1</Name> <IsNative>true</IsNative> <Alias>Short1</Alias> </Tag> <Tag xmlns="urn:sap.com:pco.contracts.items"> <Name>Data Type Examples.16 Bit Device.K Registers.Short2</Name> <IsNative>true</IsNative> <Alias>Short2</Alias> </Tag> </TagNames> <MetaDataList> <Item xmlns="urn:sap.com:pco.contracts.items"> <Name>Item Description</Name> </Item> <Item xmlns="urn:sap.com:pco.contracts.items"> <Name>DDE Access Name</Name> </Item> </MetaDataList> <SecondaryList> <Item xmlns="urn:sap.com:pco.contracts.items"> <Name>Quality</Name> </Item> <Item xmlns="urn:sap.com:pco.contracts.items"> <Name>Item Timestamp</Name> </Item> </SecondaryList> </RetrieveMessage>

Message	Description
Response	With this message you can respond to retrieve commands. The response contains metadata and secondaries.

Sample Code

```
Response for RETRIEVE command with
Metadata and Secondaries in XML
Format:
<?xml version="1.0"
encoding="utf-8"?>
<RetrieveMessageResponse
xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/
XMLSchema"
xmlns="urn:sap.com:pco.contracts">
  <TagValues>
    <TagValue><Alias>Short1</Alias>

<DateTime>2015-05-21T07:06:03.20434
93Z</DateTime>
    <Value>0</Value>
    <MetaData>
      <Name>Item Description</Name>
      <Value>16-Bit signed
integer</Value>
    </MetaData>
    <MetaData>
      <Name>DDE Access Name</Name>
      <Value>Data Type Examples_16
Bit Device</Value>
    </MetaData>
    <Secondary>
      <Name>Short1_Quality</Name>
      <Value>Good, Non-specific,
Not limited</Value>
    </Secondary>
    <Secondary>
      <Name>Short1_Item Timestamp</
Name>
      <Value>21.05.2015 07:06:03</
Value>
    </Secondary>
   </TagValue>
   <TagValue>
     <Alias>Short2</Alias>

<DateTime>2015-05-21T07:06:03.20434
93Z</DateTime>
     <Value>0</Value>
     <MetaData>
       <Name>Item Description</
Name>
       <Value>16-Bit signed
integer</Value>
     </MetaData>
     <MetaData>
       <Name>DDE Access Name</Name>
       <Value>Data Type
Examples_16 Bit Device</Value>
```

Message	Description

	`</MetaData>`
	`<Secondary>`
	`<Name>Short2_Quality</Name>`
	`<Value>Good, Non-specific, Not limited</Value>`
	`</Secondary><Secondary>`
	`<Name>Short2_Item Timestamp</Name>`
	`<Value>21.05.2015 07:06:03</Value>`
	`</Secondary>`
	`</TagValue>`
	`</TagValues>`
	`<Success>true</Success>`
	`<Error><Message xmlns="urn:sap.com:pco.contracts.error" /></Error>`
	`</RetrieveMessageResponse>`

Message	Description

`Store`	With this message, you can write tag values using the given time stamp.
	Sample Code
	`STORE command in XML Format:`
	`<?xml version="1.0" encoding="utf-8"?>`
	`<StoreMessage xmlns="urn:sap.com:pco.contracts">`
	`<StoreTagValues>`
	`<StoreTagValue xmlns="urn:sap.com:pco.contracts.items">`
	`<TagName>`
	`<Name>Data Type Examples.16 Bit Device.K Registers.Short1</Name>`
	`<IsNative>true</IsNative>`
	`<Alias>Test</Alias>`
	`</TagName>`
	`<Value>11111</Value>`
	`<DateTime>2015-03-10T12:00:00</DateTime>`
	`</StoreTagValue>`
	`<StoreTagValue xmlns="urn:sap.com:pco.contracts.items">`
	`<TagName>`
	`<Name>Data Type Examples.16 Bit Device.K Registers.Short2</Name>`
	`<IsNative>true</IsNative>`
	`<Alias>Test2</Alias>`
	`</TagName>`
	`<Value>22222</Value>`
	`<DateTime>2015-03-10T12:00:00</DateTime>`
	`</StoreTagValue>`
	`</StoreTagValues>`
	`</StoreMessage>`

4.3.3.7 OPC UA Server

Use

If you select the option *OPC UA Server* on the *Servers* tab for the agent instance, you can run the agent instance as an OPC UA server. This provides methods for execution using the OPC UA protocol. (See also: PCo as OPC UA Server [page 54].)

Procedure

1. You define the server endpoint under which the OPC UA server can be addressed. OPC UA clients can connect with the server via this endpoint. (See: OPC UA Server Settings Tab [page 456].)
2. You define the security configuration. (See: Security Configuration Tab [page 459].)
3. You define the server methods. (See: Server Method Definitions Tab [page 461].)

> **i Note**
>
> In addition to the settings available in dialog, you can make special settings in an XML configuration file `PCoUaServerTemplate.Config.xml` that is located in the system directory of the PCo installation. The configuration file is based on the standards of the OPC foundation. The parameters stored there are mixed with the settings in the Management Console, with the latter settings taking precedence. However, as a rule, you do not need to change the XML configuration file.

> **⚠ Caution**
>
> The configuration file `PCoUaServerTemplate.Config.xml` may be overwritten when the PCo installation is upgraded.

4.3.3.7.1 OPC UA Server Settings Tab

Context

You define server endpoints on this tab.

Procedure

1. Choose *Add Endpoint*.

   PCo displays the *Add Server Endpoint Description* dialog box.
2. In the *Endpoint URL* field, define the URL at which the OPC UA server can be addressed for the clients.

   PCo proposes an example URL. When you create a new entry or change an entry, you need to enter a valid URL here. A simple validity check is performed. A valid URL starts with one of the following three character strings:

   - opc.tcp://
   - http://
   - https://

   Then comes the URL of the server, where, ideally, you use a fully qualified domain name. This is followed – separated by colons – by a port number and, optionally, path details.

> ### Example
>
> Example of a valid URL:
>
> `opc.tcp://myserver.domain.com:56711/MyApplication`

PCo checks if the port number is still being used on other active servers of the PCo instance and allows you, if necessary, to choose a port number that has not yet been used.

3. The selection options for the following three fields depend on the structure of the URL entry:

Field	Description
Security Mode	The security mode defines which steps are used for a secure connection setup (OpenSecureChannel request). You can choose between the following settings: ○ *None* For this setting, the request is neither signed nor encrypted. In this case, no certificates are used for a secure connection setup. ○ *Sign* For this setting, the request is signed with the private key of the server application certificate so that the client (that has to trust the server certificate) can validate the request. ○ *SignAndEncrypt* For this setting, the server uses the public key of the client to sign the message and to encrypt it. **i Note** These settings only have an effect on the connection setup or on the later renewal of the connection.

Field	Description
Security Policy	The security policy allows the **cryptoalgorithm**, to a certain extent, to be chosen for setting up the secure connection. The following options are possible depending on the *security mode* defined previously: ○ Security mode *None*: `None` ○ Security modes *Sign* and *SignAndEncrypt*: The following security policies are available: ○ `Aes128_Sha256_RsaOaep` This security policy is intended for configurations with medium security requirements. ○ `Aes256_Sha256_RsaPss` This security policy is intended for configurations with high security requirements. ○ `Basic256Sha256` This security policy is intended for configurations with high security requirements. For more information about these security policies, see https://apps.opcfoundation.org/profilereporting/ in the section *Security Category* > *SecurityPolicy*. There are also the deprecated security policies `Basic128Rsa15` or `Basic256`. These can still be used if you have explicitly allowed their use. See: Compatibility Settings [page 17]. They appear then in the dropdown list with the addition of the word `(deprecated)`. **→ Recommendation** It is recommended that you select the security policy `Basic256Sha256` or `Aes256_Sha256_RsaPss` in all scenarios (except test scenarios) if the processing power of the client and the server permits this.
Encoding	The following options are available for encoding: ○ (OPC UA) Binary ○ (OPC UA) XML XML is only available if the endpoint URL starts with http or https.

4.3.3.7.2 Security Configuration Tab

Use

On this tab, you make the settings for secure connections and user authentication for OPC UA servers.

Procedure

Application Certificate

To be able to identify the PCo system as a server to the OPC UA client and vice versa, **X.509 v3 certificates** are used, provided a secure connection is to be set up. In the context of OPC UA, the certificates used here are called **application certificates**.

> **i** Note
>
> If you only need certificates for test purposes or work in an environment where no certificates, which have been issued by an official Certification Authority, are required, you can create the application certificate here directly.

1. You can generate and assign an application certificate for the *Application Certificate* field by choosing the icon *Generate and Assign Application Certificate*.
   The *Generate Self-Signed OPC UA Server Certificate From Defaults* dialog box appears where you can make the settings for certificate generation. See also: Generate and Assign a Self-Signed Certificate [page 158].
2. Choose *Change Application Certificate Assignment* to select and assign another certificate in the *certificate browser*.
3. If you choose *Remove Application Certificate Assignment*, the assignment of the generated certificate is removed. However, the certificate remains in the Microsoft certificate store and can be selected for other OPC UA servers.
4. Specify the *identification type* for the selected application certificate. If you select the *Identification by Subject* option, certificate rotation is supported. (See also: Identification Type of Certificates [page 593].)
5. Select the *Send Certificate Chain* checkbox if the application certificate of the server has been signed with a root certificate and is embedded in a certificate chain. This option allows you to control whether the server should try to make this chain and send it to the client when a secure connection is being set up. In this case, the server searches recursively in specific certificate stores for the certificate with which the application certificate or the CA certificate that was found last has been signed, and then sends the certificates that it has found to the client. The client needs to retain the missing certificates for a validation. Not every OPC UA application supports the receipt of certificate chains.
   Deselect the checkbox if you want to connect the OPC UA server with one of these servers. In this case, you need to make the certificate chain known to the server manually, if necessary. If you do **not** select the checkbox, the server only sends its application certificate.

Certificate Storage Configuration for the Application Certificate of the UA Client

The configuration for the certificate store of the OPC UA server is symmetrical to the corresponding settings of the OPC UA source system; only the roles of client and server are swapped. When the connection is set up between the OPC UA server and OPC UA client, the OPC UA server and the client exchange certificates with public keys to set up the connection.

Store for Trusted Client Certificates

Here you enter the *store type* and the *folder* you want. You can configure the store location for the certificates, which the OPC UA server is to trust, to the granularity of the server. You have the following options:

- *Store Type Microsoft Certificate Store*
  When a connection is being established, with this setting, an OPC UA server automatically searches in the Microsoft Certificate Store folder for a client certificate. You can select specific folders of the Microsoft Certificate Store here.
- *Store Type File System*
  With this option, you can specify the store location for the certificates, which the OPC UA server is to trust, in the file system. You can specify specific directories in the directory tree. In this case, a subfolder is offered by default in the directory that is usually used under MS Windows for storing all-user configurations.

  > ⚠ Caution
  >
  > This MS folder usually has the attribute *Hidden*. By choosing the *File System* setting, you define where you want the OPC UA server to search for the client certificate with a public key.

  > i Note
  >
  > This option does not copy any certificates into this store location.
  >
  > You need to copy the certificates into this store location yourself. If you choose a directory in the *Folder* field, the directory and the subfolder *certs* are created for this directory. These directories are neither deleted when there is a change nor when PCo is uninstalled.

Store for Rejected Client Certificates

If an OPC UA server wants to set up a secure connection to an OPC UA client, he or she receives a certificate with a public key from the client. The server accepts this certificate if he or she regards it as trustworthy (see the previous point).

Otherwise, the certificate is stored in the store for rejected client certificates. With this setting, you can define the store location for these certificates. If the client is using a self-signed certificate, you can, after an unsuccessful connection attempt, copy the certificate from this store location to the *store for trusted certificates*. This establishes the trust relationship between the server and client on the PCo side. You need to make a root certificate known in another way, for example, manually.

Store for Trusted Issuer Certificates

If the application certificate of the UA client is embedded in a certificate hierarchy, the related root certificate needs to be available to be able to validate the client certificate. You need to store this root certificate in the subfolder *certs* of the directory that can be configured with this option.

As in the case of trusted client certificates, this directory should only be writable for system administrators if it is created in the file system. If a connection attempt is unsuccessful, however, you do not then find the root certificate in the store for rejected certificates.

Alternatively, you can store the root certificate in the store for trusted client certificates. If the client then sends a valid certificate, that is, if it has all the attributes that are envisaged by the OPC UA specification for application certificates, and all attributes have valid values, you do not need to store the client certificate in the store for trusted certificates if a valid root certificate is available.

> **i Note**
>
> For the OPC UA server, you do not have the option to suppress specific validations of the client certificate and thereby to accept, in spite of these errors, expired and untrustworthy certificates as well as certificates with a faulty host name.

4.3.3.7.3 Server Method Definitions Tab

Use

On the *Server Method Definitions* tab you can add new method definitions to the server and adjust or delete existing definitions.

Procedures

Creating Method Definitions

1. To create new methods, choose *Add New Method*.
   The *Add Methods to Server* dialog box appears.
2. You have the following options:
   - *Create Method Definition Manually*
     In this case, you can manually create a method for the direct destination system call. Enter a unique method name in the dialog box that then appears.
   - *Load Method Definitions from Assembly*
     You can load predefined methods from an assembly. To do so, select an appropriate assembly from the dropdown list. When you create methods from an assembly, in the next step you can choose for which methods you want a method notification to be created. Only methods with an assigned method notification are active and are visible and executable for connected OPC UA clients. (See also: Enhanced Method Processing (EMP) [page 57].)

Maintaining Method Definitions

On the *Server Method Definitions* tab you can edit methods for direct destination system calls after creation. To do so, click on the method you want and in the detail screen on the right, maintain the input parameters, output parameters, and descriptions.

If the data type of the input parameter or output parameter is an **array**, you can also specify the intended length of the array in the *Array* field. The OPC UA clients that connect with the PCo OPC UA server are informed of the given array length of the parameter. The default value for the array length, 0, means that the array can have any length at runtime. For array lengths other than 0, the OPC UA server checks at runtime whether the actual array length of the input or output parameter corresponds to the specified length.

By selecting the *Asynchronous* checkbox, you can define whether you want the method to be run asynchronously at runtime:

- Checkbox is not selected.

In this case, the method is executed **synchronously**. PCo directly triggers the actions linked with the method notification such as execution of the EMP implementation and the destination system call configured there and then waits until these actions are completed. Only then is control returned to the caller of the method. The output parameters of the method are filled with the assigned output variables of the destination system call or of the EMP implementation.

- Checkbox is selected.
  In this case, PCo also triggers execution of the method notification by calling the method, but returns process control directly to the caller of the method. The caller does not then have to wait for the end of method processing. You can also define output parameters for asynchronous methods, but these cannot be supplied from the output variables of the destination system call or the EMP implementation. Instead, PCo returns the initial value of the output parameter data type in each case to the caller.
  As soon as you have selected the *Asynchronous* checkbox for a method, the *Message Delivery* tab appears for the relevant method notification. You can then make the settings specifically for this method notification. (See also: Message Delivery Tab [page 518].)
  For an asynchronous method call, PCo generates a notification message that is placed in the queue for notification messages and is delivered in a background process.
  The settings described for the *Notification Processing* tab also apply to placing the method notification in the queue for message notifications. (See: Notification Processing Tab [page 481].)

You can use asynchronous methods, for example, to inform the SAP ME system about the completion of an operation without receiving the information about the next operation that is to be executed, because this is already determined in the process flow.

Icons

Below is an overview of the icons that are provided for maintaining the server methods:

Icon	Description
Add New Method	You choose this icon to add a new method.
Delete Selected Method or Node	Choose this icon if you want to delete a selected method. After a confirmation prompt, generated method notifications are deleted. If you want to remove an assembly from the configuration, select the node for the assembly and choose this icon. The assembly and all assigned method notifications are also deleted.
Duplicate Selected Method	The selected method is copied. It gets a new name and internally a new ID so that it can be identified as a standalone method during browsing via an OPC UA client. The parameters of the original method are adopted and any existing method notifications are also copied.
Change Method Name	Choose this icon if you want to change the method name. This function has no effect on method notifications that have already been created; in other words, they retain their original name. However, you can change the name of the method notification via the Management Console.

Icon	Description
Create Notification for Selected Method	Choose this icon to create a method notification for the method. The system proposes a name derived from the method name that you can change later. Output expressions are generated automatically for the notification from the input parameters of the method. **i Note** Therefore, we recommend that you do not create the method notification until the interface for the method has been defined completely. Method definitions that are from an assembly of enhanced method processing (EMP) are only displayed on this tab and cannot be changed. However, you can choose the icon *Create Notification for Selected Method* to create method notifications for selected methods and activate the method this way.
Navigate to First Assigned Method Notification	Choose this icon to navigate to the first method notification that is assigned to the method.
Display Related OPC UA Destination Systems	Choose this icon to call a where-used list for the selected method. The where-used list finds destination systems within your own PCo installation that can call this method via the OPC UA server of the agent instance. Moreover, there might be callers of the method from external clients; these are not included in the where-used list, however.
Reload Method Definitions from Assembly	Choose this icon to compare the OPC UA server configuration with a new version of the EMP assembly. New methods can be loaded and the description of existing methods can be adjusted. **i Note** However, changing the interfaces of existing methods, deleting methods, or changing their name using a new version of the EMP assembly is not envisaged. The Management Console rejects the reloading of such a modified assembly with an error message.

4.3.3.8 PCo Web Server

You can also run a PCo agent instance as a Web server.

To do so, select the *PCo Web Server* option on the *Servers* tab for the agent instance. Using various Web service protocols, this provides service operations in the form of methods for execution.

> **i** Note
>
> The PCo Web server supports the SOAP, REST, and ODATA protocols.

The following settings are required:

1. You define the **Web service endpoint** at which the Web server can be called. Web clients can connect with the server via this endpoint. (See: PCo Web Server Settings Tab [page 464].)
2. You define the server methods. (See: Web Server Method Definitions Tab (Web Server) [page 469].)

Related Information

PCo as OPC UA Server and as Web Server [page 54]

4.3.3.8.1 PCo Web Server Settings Tab

Context

On this tab, you define the server endpoint under which the Web server can be called.

Procedure

1. Choose the *Add Endpoint* pushbutton.

   PCo displays the *Add Server Endpoint Description* dialog box.
2. In the *Endpoint URL* field, define the URL under which the Web server can be called by the clients.

   PCo proposes an example URL. When you create a new entry or change an entry, you need to enter a valid URL here. A simple validity check is performed.

   > **i** Note
   >
   > A valid URL starts with one of the following character strings:
   > - http://

With this URI scheme, you can only choose the options **`None`** or **`Basic`** in the *Authentication* field. Authentication using a **`certificate`** is **not possible**.

- https://
  The advantage of this URI scheme is that communication can be secured using a TLS certificate. With this URI scheme, you can choose between all three options in the *Authentication* field.

Then comes the URL of the server, where, ideally, you use a fully qualified domain name. This is followed – separated by colons – by a port number and, optionally, path details.

Example: **https://localhost:<Port>/PCoWebServer**

PCo checks if the port number is still being used on other active servers of the PCo instance and allows you, if necessary, to choose a port number that has not yet been used.

3. Make the following settings:

Settings for the Endpoint

Field	Description
Service Mode	The service mode allows the user to select a protocol or an architecture style for the Web service. You can choose between the following settings: ○ SOAP-based Web service This Web service is closely linked to the server; in other words there is a fixed contract between the client and the server. The client updates the service definition as soon as there are changes on the server. ○ REST-based Web service The architecture of this Web service is designed in such a way that a client knows the URL of the entry point and processes the stateless resources using a standard interface. This processing is performed by the use of http methods such as GET, POST, UPDATE, and DELETE operations.
OData	If you set the indicator, the OData protocol is activated.
Target Namespace	You need to create a target namespace if you are using parameters with the same name. You can only distinguish between parameters with the same name if you use namespaces. The target namespace is only input enabled in conjunction with the **SOAP-based** setting. The namespace `http://tempuri.org` is proposed by default. You can overwrite this proposal. Once you have saved, the namespace that you entered is created.

Field	Description
Content Type	The content type defines how the resources are displayed. You can choose one of the following options, depending on the service mode you defined previously: ○ XML ○ JSON
Authentication	The Web client can perform the calls of the Web service operations with or without access control. ○ Certificate If you choose the *Certificate* option, you need to select a server certificate. In addition, you need to make settings for the certificate folder. (See also: Certificate Folders [page 467].) **i Note** The *Certificate* option is only possible in conjunction with the https URI scheme. No authorization check takes place during runtime. Only the certificate is checked. ○ If you choose the *Basic* option, the Windows user needs to provide valid logon data. This option is possible for the `http` and for the `https` URI scheme. After authentication, PCo performs a role-based authorization check. **i Note** The Windows user who authenticates herself or himself to the Web server needs to belong to the Windows user group `PCoWebServer`. The Web server cannot be called for Windows users who, even though they can authenticate themselves, do not belong to the user group. ○ If you choose the option *None*, the client can consume the Web service without logon data needing to be provided. No authorization check takes place either.

Field	Description
TLS / Server Certificate	You always need to choose a TLS certificate here if you are using `https` as the URI scheme for the endpoint. TLS is a technology concept that enables secure communication between client and server using the https protocol. The TLS server certificate is also required in conjunction with the *Certificate* option in the *Authentication* field. **i Note** The certificate must match the domain server name (host name) that is used in the endpoint. This means that, in the settings for the certificate, the `Subject Name` needs to match with the host name of the server endpoint, otherwise no connection can be established between PCo and the client.
Certificate Folders	You make the settings for the certificate folders here. These settings are used during runtime for validation of the certificate sent by the client. (See also: Certificate Folders [page 467].) **i Note** The fields are only ready for input if you have chosen the *certificate* option in the *authentication* field.

Results

The endpoint for the Web server has been created. In the next step, you need to create the method definition. (See: Web Server Method Definitions Tab (Web Server) [page 469].)

4.3.3.8.1.1 Certificate Folders

This screen area is only ready for input if you have chosen client certificate authentication.

You make the settings here for how the client certificate is to be validated. During runtime, the client sends the certificate to the PCo server. PCo checks the submitted certificate against the settings that you have made here.

Settings for Certificate Folders

Field	Description
Store Type	Here you select the store for the certificate that you want to be validated. The following types are supported: • Microsoft certificate store When a connection is being established, with this setting, PCo automatically searches in the Microsoft certificate store folder for a server certificate. → Recommendation If you are using PCo based on **Windows OS**, you need to use the Microsoft certificate store option. • File system certificate store With this option, you can specify the store location for the certificates, which PCo is to trust, in the file system.
Trusted Certificates	Here you can specify the folder in which the trusted certificates are stored. If you have selected the Microsoft certificate store, this is the folder for the trusted root certification authorities. The system proposes this automatically. Example Local Computer/Trusted Publishers
Issuer Certificates	Here you can specify the folder in which the certificates of a trusted issuer are stored. If you selected the **Microsoft certificate store**, this is the folder for the intermediate certificate authorities. This is proposed automatically. Example Local Computer/Intermediate Certificate Authorities If you have selected the **file system certificate store**, a directory is proposed in the file system with the subfolder `certs`. This folder is used to complete the certificate chain if the server does not send the complete certificate chain.

Field	Description
Rejected Certificates	Here you can specify the folder in which the rejected certificates are stored. If you are using the **Microsoft certificate store**, select **Untrusted Certificates** here. **Example** Local Computer/Untrusted Certificates If you have selected the **file system certificate store**, use a directory in the file system with the subfolder `certs` (folder for rejected certificates).
Revocation Check	In this field you define how the revocation check of the server certificate is to be performed. You have the following options: • **No Check on Revoked Certificates** No check is carried out. • **Check Online Revocation Lists** The online check is a secure procedure but it can have a negative impact on performance. **→ Recommendation** This is the recommended setting in connection with the Microsoft certificate store. • **Check Offline Revocation Lists** ○ If you are using the **Microsoft certificate store**, you need to copy all the relevant certificate revocation lists into the `Trusted Root Certification Authorities` directory. ○ If you have selected the **file system certificate store**, you need to copy all related certificate revocation lists as .crl files into the revocation list folder.

4.3.3.8.2 Web Server Method Definitions Tab (Web Server)

On the *Web Server Method Definitions* tab, you can add new method definitions to the server and adjust or delete existing definitions.

Related Information

Creating Method Definitions [page 470]
Maintaining Method Definitions [page 470]
Result of Method Maintenance [page 473]
Icons [page 474]

4.3.3.8.2.1 Creating Method Definitions

1. To create new methods, choose *Add New Method*.
   The *Add Methods to Server* dialog box appears.
2. You have the following options:
   - *Create Method Definition Manually*
     In this case, you can manually create a method for the direct destination system call. Enter a unique method name in the dialog box that then appears.
   - *Load Method Definitions from Assembly*
     You can load predefined methods from an assembly. To do so, select an appropriate assembly from the dropdown list. When you create methods from an assembly, you can choose, in the next step, for which methods you want a method notification to be created.
     Only methods with an assigned method notification are active and are visible and executable for connected clients. (See also: Enhanced Method Processing (EMP) [page 57].)

4.3.3.8.2.2 Maintaining Method Definitions

On the *Web Server Method Definitions* tab, you can edit methods for direct destination system calls after creation.

To do so, click on the method you want and in the detail screen on the right, maintain the *description*, *input parameters*, and *output parameters*.

Fields

Field	Description
HTTP Method	The following methods are available for a RESTful or OData service and can be selected: • GET • POST • PUT • DELETE **i Note** If you do not select anything, the GET method is used automatically. For SOAP, these methods are only displayed. They are not selectable.
Asynchronous	You can use this checkbox to specify whether you want the method to be executed asynchronously or synchronously at runtime. (See also: Asynchronous [page 471].)
Include Parameters in Message Body	You can use this checkbox to specify whether or not you want the input parameters to be included in the message body. (See also: Include Parameters in Message Body [page 472].)

4.3.3.8.2.2.1 Asynchronous

You can use the *Asynchronous* checkbox to specify whether you want to execute the method asynchronously or synchronously at runtime.

Setting	Result
Checkbox is **not** selected	In this case, the method is executed **synchronously**. PCo directly triggers the actions linked with the method notification such as execution of the EMP implementation and the destination system call configured there and then waits until these actions are completed. Only then is control returned to the caller of the method. The output parameters of the method are filled with the assigned output variables of the destination system call or of the EMP implementation.

Setting	Result
Checkbox is selected	In this case, the method is executed **synchronously**. In this case, PCo also triggers execution of the method notification by calling the method, but returns process control directly to the caller of the method. The caller does not then have to wait for the end of method processing. You can also define output parameters for asynchronous methods, but these cannot be supplied from the output variables of the destination system call or the EMP implementation. Instead, PCo returns the initial value of the parameter in each case to the caller. **Example** You can use asynchronous methods to inform the SAP ME system about the completion of an operation without receiving the information about the next operation that is to be executed, because this is already determined in the process flow.

4.3.3.8.2.2.2 Include Parameters in Message Body

The *Include Parameters in Message Body* checkbox allows you to specify whether or not you want the input parameters to be included in the message body.

You use this checkbox in conjunction with the **`REST_based`** *service mode* and you use it if you want to use arrays.

Example

You have created the **`test`** method, and you want to use arrays. The endpoint then looks like this: `https:// localhost:8081/ PCoWebServer/test`.

You enter the following input parameters:

- inParameter1
- inParameter2

You have selected the data type *system.String[]*.

Possible Settings

Setting	Result
Checkbox is **not** selected	In this case, the PCo server expects the client to send a bare request. The request then looks like this: **Example** `https:// …../ test ? inParameter1 = a,b,c,d&inParameter2 = e,f,g` a b c are example values. The individual values must be separated by a comma.

Checkbox is selected	In this case, the server expects the client to send a wrapped request. This means that the input parameters are contained in the HTTP message body.
	If you are working with JSON, the request looks like this:
	`{"inParameter1": ["a","b"],`
	`"inParameter2": ["c","d"]}`

4.3.3.8.2.3 Result of Method Maintenance

After you have selected a method and defined the appropriate input and output parameters, PCo generates a WSDL (for SOAP methods) or a URL (for REST or OData methods), with which the method can be called. **This WSDL or URL is not displayed in the PCo Management Console.**

Example

As method, the service operation `wsCdyne` and the input parameter `inZipCode` were used for the following examples. The WSDL or the URL is structured as follows:

- SOAP:
  `http//:localhost:<port>/PCoWebserver?wsdl`
- REST:
  `http//:localhost:<port>/PCoWebserver/wsCdyne?inZipCode=1,2,3,4,5`
- OData:
  `http//:localhost:<port>/PCoWebserver/wsCdyne?inZipCode='12345'`

If you are using enhanced method processing (EMP), the URL is structured in accordance with the following examples (assuming that parameters of type **string** are being used):

Example

REST:

`http(s)://<machine_name>:<port>/<Application_Name>/<EMP_Name>/<operation_name>?<param1=1,2,3,4,5>&<param2=1,2,3,4,5>……`

4.3.3.8.2.4 Icons

Below is an overview of the icons that are provided for maintaining the server methods:

Icon	Description
Add New Method	You choose this icon to add a new method.
Delete Selected Method or Node	Choose this icon if you want to delete a selected method. After a confirmation prompt, generated method notifications are deleted. If you want to remove an assembly from the configuration, select the node for the assembly and choose this icon. The assembly and all assigned method notifications are also deleted.
Duplicate Selected Method	The selected method is copied. It gets a new name and internally a new ID. The parameters of the original method are adopted and any existing method notifications are also copied.
Change Method Name	Choose this icon if you want to change the method name. This function has no effect on method notifications that have already been created; in other words, they retain their original name. However, you can change the name of the method notification via the Management Console.
Create Notification for Selected Method	Choose this icon to create a method notification for the method. The system proposes a name derived from the method name that you can change later. Output expressions are generated automatically for the notification from the input parameters of the method. **i Note** Therefore, we recommend that you do not create the method notification until the interface for the method has been defined completely. Method definitions that are from an assembly of enhanced method processing (EMP) are only displayed on this tab and cannot be changed. However, you can choose the icon *Create Notification for Selected Method* to create method notifications for selected methods and activate the method this way.
Navigate to First Assigned Method Notification	Choose this icon to navigate to the first method notification that is assigned to the method.

Icon	Description
Reload Method Definitions from Assembly	Choose this icon to compare the Web server configuration with a new version of the EMP assembly. New methods can be loaded and the description of existing methods can be adjusted. **i Note** However, changing the interfaces of existing methods, deleting methods, or changing their name using a new version of the EMP assembly is not envisaged. The Management Console rejects the reloading of such a modified assembly with an error message.

4.3.4 Tag Query Tab

Context

You use the *Tag Query* tab to specify the tags required for the query in the query process.

Procedure

1. On the *Plant Connectivity Management Console* screen, select an agent instance and click on the *Tag Query* tab.
2. For information you need to enter, use the following table:

Field	Description
Cache Mode	○ *Access to Data Source Only* With this setting, all data is retrieved directly from the data source for queries. With this setting, performance is slowest. ○ *Access to Cache, to Data Source as Required* With this setting, for a query, the cache is first checked to see whether there are tags and metadata. If not, the tags are retrieved from the data source and added to the cache. ○ *Access Using Aliases Only* You use this setting if you have previously created **an alias for the source system**. An alias is a user-defined structure for tags, metadata, and secondary elements. The alias is loaded when the agent starts. See also: Source System: Aliases Tab [page 138]. **i Note** Using an alias improves performance and is therefore the recommended approach. ○ *Access to Cache Only* With this setting, all tags are queried from the data source and stored in the cache. Metadata is retrieved when a tag is requested.
Mask	Here you can restrict selection of the tags. For example, if you specify INT*, only those tags whose name begins with *Int* are chosen from the source system. Note: This field is only valid in combination with the cache mode *Cache*. See: Filter Functions [page 266].
Alias	Name of the alias file to be used. **i Note** You can use this field only if you have chosen the *Alias* setting in the *Cache Mode* field.

4.3.5 Subscription Items Tab

Use

On the *Subscription Items* tab, you can subscribe to tags for an agent instance, as well as change and delete the subscriptions. You thereby specify the tags that you want the source system to transfer in the notification

process. In the case of static notifications, the subscription items are valid and active for the entire agent instance as soon as the agent instance has been started.

> **i** Note
>
> If you are using versioned notifications, you maintain the subscription items for each notification version. In the case of a running agent, only the subscription items that are assigned to the current notification version are active.

Procedure

1. If the agent instance contains versioned notifications, you can select via a **tree** on the left-hand side whether you want to maintain subscription items for **static notifications** or for an individual notification version of a **versioned notification**. Select the relevant node. If your agent instance only contains static notifications, the tree is hidden.
2. You can choose *Browse for Tags* to browse the namespace of the source system and select one or multiple tags. Alternatively, you can choose *Add New Subscription Item Manually* to add an individual tag specifically. (See also: Adding Tags [page 480].) The selected tags are then displayed in the table of subscription items with their attributes:

Field	Description
Name	Specifies the tag name.
Source	Specifies the name of the tag in the namespace of the source system, for example, *Channel1.DataTypes.Boolean*.

Field	Description
Deadband	Here you can specify an absolute value for the deadband [page 580]. The deadband respresents a limit value for filtering out tag values. The deadband specifies the value by which the tag value must change at least so that PCo sends a notification message to the destination system. **Example** **Example 1** You have specified the value **1** as the deadband here. The last measured tag value was 33. The subsequent tag value is 34.5. The difference is 1.5. Result: 1.5 > 1 In this case, PCo sends a notification message to the destination system. **Example 2** The last measured tag value was 33. The subsequent tag value is 33.5. The difference is 0.5. Result: 0.5 < 1 In this case, PCo does **not** send a notification message to the destination system.
Only Changes	You use this indicator to specify that you only want PCo to send a notification message to the destination system if the tag value has changed in comparison to its predecessor. If the value remains the same, no notification message is sent. This allows you to avoid the sending of superfluous notification messages. **Note** You can use this indicator in particular for **nonnumerical tags**, for example, tags with the data type *String*. It is recommended that you use the deadband for numerical tags since, due to measurement fluctuations, it is very unlikely that the value of a tag is the same in two consecutive notifications.

Field	Description
Data Type	Specifies the data type of the selected tag. The data type is determined automatically, for example, during browsing. The available data types might be restricted for technical reasons and in addition be limited by the data types that are available in the .Net Framework. If a data type is not supported, the tags are not offered for selection during browsing. They are only displayed but cannot be adopted into the list and need to be added manually. (See also: Unsupported Data Types for OPC UA Applications [page 479].)

3. To change the attributes of a subscribed tag, select the row of that tag and choose *Edit Subscription Item*. The system displays the *Edit Subscription Item* dialog box with all the attributes of the selected tag. All the attributes can be changed.
   If you are using **static notifications**, the name and data source need to be unique within all static notifications of an agent instance. Therefore, it is not possible to use a tag name or a source twice within the same static notification or in different static notifications of the same agent instance.
   If you are using **versioned notifications**, the name and data source only need to be unique within a notification version. Therefore, within an agent instance in different versioned notifications and even in different versions of the same notification, you can use tag names and data sources multiple times.

   > ⚠ Caution
   >
   > The expression for the data source cannot be checked by the source system when the subscription items are maintained manually. Any incorrect spellings are registered when the agent instance starts and are traced in the log for the agent instance.

4. To delete subscription items, select one or more rows and choose *Delete Selected Subscription Items*.
5. Click on the title row of the table to sort the list of subscription items by the content of the selected column.

> ℹ Note
>
> If you create subscription items for a draft notification, the system, for technical reasons, first generates a name that is preceded by the name of the draft notification. When the draft is activated, the names of the subscription items are adjusted automatically and the prefix is omitted. This automatic naming also considers the expressions as well as the trigger condition of the notification.

4.3.5.1 Unsupported Data Types for OPC UA Applications

Specific data types are not supported for the subscription of OPC UA tags.

The data type is determined automatically during browsing for tags on the *Subscription Items* tab. However, the available data types might be restricted for technical reasons and in addition be limited by the types available in the .Net Framework. OPC UA applications in particular work partly with data types that are not supported.

- Data type `Integer` that comprises the types `SByte`, `Int16`, `Int32`, and `Int64`
- Data type `UInteger` that comprises the data types `Byte` , `UInt16`, `UInt32`, and `UInt64`
- Data type `Number` that comprises the types `Integer`, `Double`, `Float`, `UInteger`

In this case the tags are not offered for selection when you are browsing. They are displayed, but cannot be adopted into the list. If possible, you should configure your OPC UA server for interaction with SAP Plant Connectivity so that it only uses supported data types.

You can add unsupported data types manually and then select the data type manually. However, there is a risk of data loss if the value of such a variable at runtime does not match the type. (See also: Adding Tags [page 480].)

4.3.5.2 Adding Tags

Procedure

Adding a Tag

To add a tag that is to be monitored by the system, proceed as follows:

1. In the *Plant Connectivity Management Console*, click on the *Subscription Items* tab.
2. Choose the *Browse* pushbutton.
   The *Browse* dialog box appears.
3. In this dialog box you can choose the *Browse* pushbutton to display and select the tags of the source.
   The system shows the *Address Root*. You can open the tree and display the individual tags.
4. Select a tag and choose the *Add Selected Items* pushbutton. Repeat this step to select all of the tags that you want.
5. Choose the *OK* pushbutton to adopt the selected tags.
6. Instead of selecting the tags individually, you can specify a filter criterion in the *Filter* field. PCo then displays only the tags that meet the filter criterion. See Filter Functions [page 266].

Adding a Tag Manually

Some source systems support adding subscription items manually. Adding tags manually allows you to enter the tag name directly without browsing. Adding tags manually is necessary if a data type is not supported, for example, for OPC UA applications. In this case the tags are not offered for selection when you are browsing. They are only displayed but cannot be adopted into the list and need to be added manually. (See also: Unsupported Data Types for OPC UA Applications [page 479].)

1. In the *Plant Connectivity Management Console*, click on the *Subscription Items* tab.
2. Click on the *Add New Subscription Item Manually* icon.
   The *Add Subscription Item* dialog box appears.
3. Enter the necessary data such as the name of the tag and a value for the deadband, if required.

See also

Tags [page 10]

4.3.6 Notification Processing Tab

You can make the following settings on this tab:

- Notification Message Queue and Dispatch Settings [page 481]
- Enhanced Notification Processing [page 73]

4.3.6.1 Notification Message Queue and Dispatch Settings

Definition

Before notification messages are delivered within a tag-based notification process or an asynchronous method call, they are placed in a queue. On the one hand, the queue decouples source system events from delivery of notification messages to the destination systems. So if there is a delay in delivery to a destination system, this doesn't block processing of further notification events at the source system. On the other hand, the queue makes sure that delivery retries can be started even if delivery attempts have failed.

For each agent instance, individual queues are used for sending notification messages. The queues are set up automatically as soon as you start an agent instance. The following queues are created:

- Main queue
- Queue for failed messages
- Queue for expired messages
- Queue for messages that are waiting to be bundled

The queues for an agent instance are removed when you delete the agent instance.

Procedure

In the *Notification Message Queue and Dispatch Settings* screen area on the *Notification Processing* tab, you can make the following settings:

Field	Description
Storage Method	Here you define how notification messages are to be stored in queues. You have the following options: • *In Memory Only* • *Microsoft Message Queuing (MSMQ)* • *File System* For more information, see Storage Method [page 486].

Field	Description
Keep Messages in Memory	You can only select this option in conjunction with the *storage method* **`file system`**. If you choose this option, all notification messages from the main queue, the queue for failed messages, and the queue for messages for bundling are kept in memory by PCo. **i Note** If there is an average volume of more than two notification messages per second, these options can increase the throughput of the queues. However, we recommend that you do not use this option for large notification messages of 100 kB and more. For the storage methods *In Memory Only* and *Microsoft Message Queuing (MSMQ)*, all notification messages are kept exclusively or additionally in the memory.
Resend Failed Messages Automatically	If you choose this option, the system tries automatically after five to six minutes to resend failed notification messages that have been moved to the corresponding queue. The system determines the number of dispatch threads used for the resending of failed messages dynamically so that timely processing of messages from the main queue is guaranteed. If you **do not select** this option, but you still want to resend failed notification messages, you have to **manually** select the *Resend All* function on the *Message Failures* tab for the agent instance.
Keep Expired Messages	If you select the checkbox, notification messages that have expired (see Notification: Message Delivery Tab [page 518]) are moved into the *Expired Messages* queue. They can be displayed there for evaluation purposes. The *Expired Messages* queue grows continuously if you do not delete notification messages from time to time. If you do not select the checkbox, the notification messages are deleted automatically and immediately from all queues when they expire.

Field	Description
Process Notification Messages Exactly Once in Order	By selecting this checkbox, you can define at agent instance level that all notification messages are delivered individually and exactly in the order in which they were placed in the queue in accordance with the delivery type Exactly Once in Order. **i Note** In this case, the delivery of the messages is sequential. The **Max. Dispatch Threads** field is not relevant to the delivery type *Exactly Once in Order* and so you cannot change it. This setting overwrites the *Process Notification Messages Exactly Once in Order* setting that you have set at notification level. For more information, see Processing Notification Messages Exactly Once in Order [page 487].
Keep Copies of Queued Notification Messages in Journal Queue	If you select this checkbox, a copy of each notification message is stored in the *Journal Messages* folder of the *Microsoft Message Queuing* component. This option is only available in the **storage method Microsoft Message Queuing (MSMQ)**. **i Note** The *Journal Message Queue* folder is located, for example, in *Windows 7* under *Control Panel* > *Administrative Tools* > *Computer Management* > *Services and Applications* > *Message Queuing* > *System Queues* > *Journal Messages*. You should only use this setting in the test phase for testing and for tracking errors, otherwise the contents of the directory can become too large and can have an adverse effect on performance.

Field	Description
Make Queued Notification Messages Recoverable	If you select the checkbox, MSMQ stores the notification messages on the hard disk. The notification messages then remain in the queue even if *Windows* has to be restarted. If you do not select the checkbox, the notification messages are only saved in the working memory. This leads to a better performance. Irrespective of this checkbox, notification messages are kept in the queue if the agent instance has to be restarted. This option is only available in the **storage method Microsoft Message Queuing (MSMQ)**.
Max. Queued Messages	This field specifies the maximum number of notification messages that are allowed to be in the message queue. This includes messages on the *Queued Messages* and *Message Failures* tabs. When the specified number of messages has accumulated, the connection to the source system is terminated and the agent instance gets the status *Error*. You can enter `values up to 1,750,000,000`. The default value is *1000*. At 80% of the value, the system issues a warning of the event type *Critical* in the message log. If you start the agent instance again when the maximum number of messages has already been exceeded, the upper limit is increased temporarily to 120% of the value so that you can resolve the situation when the agent instance is running. If you keep notification messages in the memory, choose a value here that is large enough so that no memory problems can occur during the runtime of the agent instance. For more information about estimating the memory requirements of message queues, see the *Performance and Sizing Guide* on the PCo product page.

Field	Description
Max. Dispatch Threads	This field specifies the maximum number of threads per processor core that are to be used for the asynchronous, parallel sending of notifications. The limits of this field are a minimum of 1 thread and a maximum of 250 threads per processor core. Therefore, you are effectively configuring a maximum number of parallel threads that results from the number of processor cores multiplied by the value specified in this field. The default value is 5. **Example** You enter a field value of **5**. Your PCo computer has eight processor cores. You thereby configure a maximum number of parallel threads of 8 x 5 threads = 40 threads. **Caution** At the runtime of the agent instance, the system only provides a limited number of possible parallel threads. The parallel threads are also used by other PCo processes, such as session handling for OPC UA functions. Therefore, you should not use the maximum value of 250 for the dispatch threads. Moreover, the possible message throughput of the destination systems used determines the number of dispatch threads that can be used with meaningful effect. For example, a connection to a Web server using a universal Web service destination system only provides a limited number of open HTTP connections. A large number of dispatch threads used that have a relatively small number of usable HTTP connections, can lead to a system load that is too high and can lead to long waiting times when agent instances are stopped. When the agent instance is started, a warning message appears in the *agent log* if the actual number of the maximum number of parallel threads you configured exceeds 60% of the threads provided by the system. The *Max. Dispatch Threads* field is not relevant to the delivery type *Exactly Once in Order* and so **you cannot change it**. In the case of the delivery type *Exactly Once in Order*, delivery is always synchronous and sequential.

4.3.6.1.1 Storage Method

Use

Before notification messages are delivered to a destination system, they are stored temporarily in a queue. Furthermore, notification messages that are to be bundled, that have failed to be delivered, or that have expired, are stored in queues.

Storage Methods

Method	Description
In Memory Only	With this option, the notification messages are kept in the memory of the PCo computer at the runtime of the agent instance and are not saved in addition. When an agent instance is stopped or the PCo computer is shut down, all notification messages in the queues are lost. Of all the storage methods, this option requires the fewest system resources.
Microsoft Message Queuing (MSMQ)	With this option, all the notification messages are kept in the memory of the PCo computer and saved using the Windows service **Microsoft Message Queuing**. Microsoft Message Queuing (MSMQ) is a Windows component developed by Microsoft. For more information about MSMQ, see the Microsoft documentation under http://msdn.microsoft.com/en-us/library/ms711472(v=VS.85).aspx . The notification messages are not lost when an agent instance is restarted. If you also use the option *Make Queued Notification Messages Recoverable*, the messages are not lost when the Windows operating system is restarted. **i Note** If you use this option, notification messages must not be bigger than 4 MB. Note that the MSMQ framework also temporarily stores the notification messages in the computer memory. To allow quick access to the notification messages, PCo also keeps the messages in the memory.

Method	Description
File System	With this option, the notification messages are stored in the local file system of the PCo computer. The notification messages are not lost when an agent instance or the Windows operating system is restarted. **i** Note SAP recommends this method for large notification messages (larger than 100 kB) that occur at a relatively low frequency (maximum one message every two seconds). If you **do not select** the *Keep Messages in Memory* option, this storage method places the least load on the main memory.

4.3.6.1.2 Process Notification Messages Exactly Once in Order

Use

By selecting this checkbox, you can specify that the notification messages are delivered in accordance with the delivery type **Exactly Once in Order (EOIO)**.

If you use the setting at agent instance level, each notification message that is generated is an EOIO notification message. However, you can also make the EOIO setting on the *Message Delivery* tab of a notification so that the setting only applies to specific notifications. As a result, there might be EOIO notification messages alongside non-EOIO notification messages in the agent instance queue that are delivered in different ways:

EOIO notification messages are delivered individually and exactly in the order in which they were placed in the queue.

The names of the notifications and the names of the destination system assignments to the notifications determine the order in which notification messages are placed in the queue. If an event occurs at a source system that meets the trigger conditions of multiple notifications, the notifications are processed in alphabetical order. If multiple destination systems are assigned to a notification, the name of the destination system assignment determines the order in which the notification messages are placed in the queue (see also: Destinations Tab [page 520].) You can change the sequence by renaming the notifications or destination system assignments to a notification.

Example

In your agent instance, you have defined the notifications **Notif1**, **Notif2**, **Notif3**. Each of these notifications has two destination system assignments with the names a and b. All notifications have the *trigger type* `Always` so that each source system event generates six notification messages in total. In accordance with

the names of the notifications and the destination system assignments, notification messages are placed in the queue in the following sequence:

1. Notif1 > a
2. Notif1 > b
3. Notif2 > a
4. Notif2 > b
5. Notif3 > a
6. Notif3 > b

If an EOIO notification message cannot be delivered, this notification message blocks the sending of any further EOIO notification messages to prevent one of these other messages being delivered first.

An EOIO notification message that cannot be delivered remains in the main queue until it can be delivered successfully. If the maximum number of retries is reached and the EOIO notification message could still not be delivered, PCo stops the agent instance. The failed EOIO notification message remains in the main queue so that it can be sent again as soon as you restart the agent. (See also: Notification Message Queue and Dispatch Settings [page 481].)

Non-EOIO notification messages are sent in parallel processes. For this, you can specify the maximum number of dispatch threads. PCo tries to automatically resend non-EOIO notification messages that could not be delivered successfully. (See: Notification: Reliable Connection Tab [page 518] and Agent Instance: Displaying Messages [page 490].)

4.3.6.2 Enhanced Notification Processing

Use

In the *Enhanced Notification Processing* screen area, you can specify whether you want to use an enhancement for your notification processes. You have the following options for the notification enhancement:

- SAP standard enhancement *Destination System Calls with Response Processing* and *Automatic Configuration Backup*
  These SAP standard enhancements are delivered with the standard installation of *SAP Plant Connectivity*. If you choose one of these options, the corresponding enhancement is included automatically in notification processing.
- Customer-Owned Enhancement
  With this option, you create your own enhancement implementation and assign it here. With the enhancement implementation, you can define your own functions that you want PCo to execute in the notification process. For example, you can interpret information from the data source and then call a specific sequence of Web services.

Moreover, you need to create the destination system for the notification enhancement. This destination system is a special destination system of the agent instance and is therefore not displayed in the destination systems area.

Prerequisites

If you want to use a customer-owned enhancement, you first need to create and compile your implementation. During the compilation, a *Dynamic Link Library (DLL)* is generated.

Procedure

1. Choose the *Enhanced Notification Processing* tab.
2. Select the notification enhancement you want:

Field	Description
No Enhanced Notification Processing	With this option, there is no enhanced notification processing.
Destination System Calls with Response Processing	With this option, the system automatically uses the SAP standard enhancement *Destination System Calls with Response Processing* (see also: Destination System Calls with Response Processing [page 81].)
Automatic Configuration Backup (New)	You can use this enhancement to create data backups of the configuration. For more information, see Setting Up an Automatic Backup [page 597].
Customer-Owned Enhancement	With this option, you create your own enhancement implementation and assign it in the *Details of the Enhancement Implementation* screen area. This implementation is an enhancement in the form of a Dynamic Link Library (DLL) with which you can implement your own functions. For example, you can interpret information from the data source and then call a specific sequence of destination systems (Web services).

3. If you have selected the *Customer-Owned Enhancement* option, you need to choose the *Browse* pushbutton to search for your implementation.
4. Then select the desired *Dynamic Link Library* (DLL) or *executable file* that you generated previously.
   If, in your implementation, you have only defined one class that implements the enhanced notification processing interface, this class appears automatically in the *Class* field. If you defined more than one class, you need to select the appropriate class.
5. If you want to delete the configuration for the DLL and the class, choose *Reset*.
6. Create a destination system for notification processing by choosing the *Create Destination System* function key.
   PCo displays the status of the destination system and its usage.

> **i Note**
>
> You can then select this destination system on the *Destinations* tab of the notification by selecting the `/Enhanced Notification Processing` entry in the *Destination System Type* field. (See also: Assignment of Modules and Variables (ENP) [page 75].)

7. To delete the destination system, you can choose the *Delete Destination System* pushbutton. You can only delete the destination system if it is not being used in a notification.

More Information

For more information about using the customer-owned enhancement, see the **Implementation Guide for Enhanced Notification Processing** on the *PCo product page* under *Additional Information*.

4.3.7 Displaying Messages

Use

In the PCo Management Console, you can display the following notification messages:

- Queued messages
- Bundled notification messages
- Messages that cannot be delivered
- Expired messages

Procedure

Displaying Queued Messages

1. On the *Plant Connectivity Management Console* screen, select an agent instance, click on the *Queued Messages* tab, and choose the *Refresh* pushbutton.
   All the notification messages that are currently queued are displayed on this tab. Messages that are currently being sent are not displayed. If PCo does not succeed in sending a notification message, the message is displayed on this tab with entries in the *Last Attempt* and *Current Attempt* columns. Since this list is not updated automatically, you need to choose the *Refresh* pushbutton.

   > **⚠ Caution**
   >
   > The entries in the *Last Attempt* and *Current Attempt* columns are only up-to-date if the agent is running or has paused. If the agent has been stopped, the entries are not up-to-date.

2. Click on *Display* to display the details for a notification message.
3. If you choose the *Delete All* pushbutton, you can remove a notification message from the queue.

Display Messages for Bundling

If you have activated message bundling on the *Message Delivery* tab, PCo bundles multiple individual messages into one bundled notification message. (See: Message Bundling [page 579] and Notification: Message Delivery Tab [page 518].) To be able to display the messages that are to be bundled, do the following:

1. On the *Plant Connectivity Management Console* screen, select an agent instance, click on the *Messages for Bundling* tab, and choose the *Refresh* pushbutton.
   PCo displays all the notification messages that are to be bundled into one larger notification message.
2. To send one or more notification messages manually, select the message or messages and choose the *Send All* pushbutton.
3. To delete the notification messages, choose the *Delete All* pushbutton.

Displaying Message Failures

1. On the *Plant Connectivity Management Console* screen, select an agent instance, click on the *Message Failures* tab, and choose the *Refresh* pushbutton.
   PCo displays all notification messages that could not be sent, even after multiple attempts.
2. To send one or more notification messages manually, select the message or messages and choose the *Resend All* pushbutton. To delete the notification messages, choose the *Delete All* pushbutton.

Displaying Expired Messages

1. On the *Plant Connectivity Management Console* screen, select an agent instance, click on the *Expired Messages* tab, and choose the *Refresh* pushbutton.
   PCo displays all notification messages that could not be sent and that have exceeded the predefined *Lifetime*.
2. Select a message and choose the *Delete All* pushbutton.

4.3.8 Functions for the Agent Instance

Prerequisites

You have created a source system.

Procedure

Add Agent Instance

1. In the *Plant Connectivity Management Console*, choose ▶ *Plant Connectivity* > *New* > *Agent Instance* ▶ in the menu.
2. Select a source system.
3. Enter the required data and choose *OK*.

Duplicate Agent Instance

1. On the *Plant Connectivity Management Console* screen, select the agent instance you want and choose the *Duplicate* pushbutton in the Agent Instances area.
   The system creates a new agent instance with the same settings. The name of the original agent instance is copied and supplemented by *(1)*.

2. You can change the settings of the new agent instance.

Add an Agent Instance Using Drag and Drop

You can use *Drag and Drop* to automatically create an agent instance for each source system.

1. Click on the source system for which you want to create an agent instance and drag it to the *Agent Instances* screen area.
   PCo then automatically generates the agent instance with all default settings.
2. Check and supplement the settings for the agent instance.

Delete Agent Instance

> **i Note**
>
> You cannot delete the agent instance when it is being executed.

1. On the *Plant Connectivity Management Console* screen, select an agent instance and choose *Edit* > *Delete*.
2. Choose *Yes*.

Rename Agent Instance

1. On the *Plant Connectivity Management Console* screen, select an agent instance and choose *Edit* > *Rename*.
2. Enter the required data and choose *OK*.

Changing the Description of an Agent Instance

After creating an agent instance, you can use this function to change its description at any time by clicking on the agent instance and choosing *Edit* > *Change Description* in the menu. You can only display the description of an agent instance by placing the mouse pointer on this object. PCo then displays the description that you entered when you created the agent instance.

Show Usage in Agent Instances

With this function, you can display the notifications in which the agent instance is used. Click the agent instance you want and choose the *Show Usage in Agent Instances* pushbutton. PCo then displays a dialog box with the where-used list. The same function is available for source systems and destination systems.

Additional Functions for the Agent Instance

Starting and Stopping an Agent Instance [page 493]

Using Starting Groups for Agent Instances [page 494]

Exporting and Importing an Agent Instance [page 598]

Context Menu

You can call a context menu for each agent instance by clicking on the agent instance with the right mouse button. The following functions are available in the context menu for an agent instance:

- Start Agent Instance
- Stop Agent Instance
- Start Agent Instance in Starting Group
- Stop Agent Instance in Starting Group

- Export Agent Instance
- Add Notification
- Duplicate Agent Instance
- Delete Agent Instance
- Rename Agent Instance
- Change Agent Instance Description
- Add Notification
- Copy and Paste
- Show Usage in Agent Instances

4.3.9 Starting and Stopping an Agent Instance

Starting an Agent Instance

To set up the data flow between the data source, PCo, and the destination system, or in order that you can use the agent instance as a server, you need to start the agent instance.

> **i Note**
>
> In the notification process with SAP MII, PCo sends a notification message as soon as the agent instance has been started.

On the *Plant Connectivity Management Console* screen, select an agent instance and choose *Plant Connectivity* > *Start Agent Instance*.

Stopping an Agent Instance

If you want to stop the data flow or stop the server, you need to stop the agent instance.

> **i Note**
>
> While the agent instance is running, the change options for the agent instance and the source and destination systems involved are restricted. However, you can change the log level of the agent instance at runtime and add and activate versioned notifications (see also: Versioned Notifications [page 498].)

On the *Plant Connectivity Management Console* screen, select an agent instance and choose *Plant Connectivity* > *Stop Agent Instance*.

Stopping All Agent Instances

In the menu under *Plant Connectivity* > *Stop All Agent Instances* , you can use the *Stop All Agent Instances* function to stop all active agent instances in one step, after you have clicked *Yes* when the confirmation prompt appears. A pushbutton is provided for this function in the *agent instance* area in the toolbar.

Starting Groups

You can assign agent instances to one starting group and start and stop the agent instances in that group together. For more information, see Using Starting Groups for Agent Instances [page 494].

4.3.10 Using Starting Groups for Agent Instances

Use

By assigning agent instances that belong together from a business point of view to one starting group, you can start and stop agent instances together. For example, if you have agent instances that are required for running a specific production line, you can start them all at the same time and, if required, stop them all again too.

Procedure

Create New Starting Group

1. Choose the *Host* tab of the agent instance.
2. Create a new starting group by choosing *Create New Starting Group*. In the dialog box that appears, enter a name for the starting group and save your entry.
   The agent instance is now assigned to a starting group. The new starting group is displayed in the dropdown list and is available for assignment to other agent instances.

Assigning an Agent Instance to an Existing Starting Group

1. To assign an agent instance to an existing starting group, go to the *Host* tab of the agent instance.
2. From the dropdown list for the starting group, choose an existing entry in order to assign the agent instance to this starting group and then save.
   The agent instance is now assigned to a starting group. In the agent instance tree, the name of the agent instance is preceded by the name of the starting group in square brackets.
3. To sort the agent instance tree, choose *Sort Agent Instance Tree*.
   The agent instances are now sorted in ascending, alphabetical order. The agent instances with assignment to a starting group are always displayed at the start of the list.

Removing Assignment of the Agent Instance to the Starting Group

If you want to remove the assignment of the agent instance to the starting group, choose the blank entry in the *Starting Group* field on the *Host* tab of the agent instance.

You can delete a starting group in its entirety by choosing the corresponding icon. The starting group is then removed and deleted entirely from all agent instances that were assigned to this starting group.

> **i** Note
>
> If a starting group no longer has any agent instance assigned to it, it is deleted.

Renaming the Starting Group

If you want to rename a starting group, choose *Change Starting Group Name in All Agent Instances* on the *Host* tab of one of the assigned agent instances. You can then enter a new name. The new starting group name is adopted in all agent instances that were assigned to the starting group in question.

Starting Agent Instances of a Starting Group

To start all agent instances of a starting group together, choose *Start Agent Instances in Starting Group* on the *Host* tab of one of the assigned agent instances.

Alternatively, you can execute this function from the context menu of the agent instance or by choosing the relevant icon above the agent instance tree.

Stopping Agent Instances of a Starting Group

To stop all agent instances of a starting group at the same time, choose *Stop Agent Instances in Starting Group* on the *Host* tab of one of the assigned agent instances. This function is also available in the context menu of the agent instance or as an icon above the agent instance tree.

Starting All Starting Groups

You can start all agents that are assigned to any starting group in a defined sequence. To do so, choose *Plant Connectivity* > *Start All Starting Groups* from the *PCo Management Console* menu.

Plant Connectivity then starts the agent instances in the respective starting groups in the ascending, alphabetical order of the starting group names. The agent instances of the next starting group are only started when all agent instances of the previous starting group were able to be started successfully. The automatic starting operation is terminated if one of the agent instances could not be started.

> **Example**
>
> You have assigned agent instances to the starting groups A, B, and C. If you now execute the function *Start All Starting Groups*, the agent instances in starting group A are started first. Once all the agent instances in starting group A are running, the agent instances in starting group B are started, followed by those in starting group C as soon as the instances in starting group B are running. If, for example, the start of an agent instance in starting group B terminates, the agent instances in starting group C are not then started.

Stopping All Starting Groups

To stop all agent instances (that are assigned to any starting group) in ascending, alphabetical order of the starting group names, choose the function *Plant Connectivity* > *Stop All Starting Groups* in the *PCo Management Console* menu.

The agent instances are stopped asynchronously. The system does not wait for the agent instances of one starting group to stop before the instances of the next starting group are stopped.

4.4 Notification

Definition

You use notifications to define under which circumstances, with which data, and to which destination you want PCo to send information when a specific event occurs. A distinction is made between two classes of notifications:

- Tag-based notification [page 497]
  The static and the versioned notifications form the basis for the notification messages that are to be generated and sent later. The notification messages are always generated when events occur that fulfill the defined conditions. A notification message informs the destination system about the event that has occurred. See also: Static Notification [page 498] and Versioned Notification [page 498].
- Method Notification [page 55]
  You can use a method notification to activate a method of an OPC UA server or a Web server and define which action you want to be executed when the method is called up. Here you can also define a destination system call, if necessary, that is executed when the method is called.

You can add multiple notifications to an agent instance. This is important if, for example, you want to send a notification message to different destination systems.

Use

You can use notifications in the notification process [page 35] to send messages to destination systems via PCo when specific events occur. A notification always contains the following information:

- Conditions
  When the user-defined event occurs, PCo checks the conditions and decides whether a *notification message* needs to be created and sent to the destination system.
  These conditions are also evaluated for method notifications, however these conditions need to be formulated so that only exactly one method notification is executed when the method is called.
- Output
  You define the output expressions and output values that you want to be output in the message. In the notification process, the output expressions are based on the subscription items of the agent instance.
  For method notifications for an implementation of enhanced method notification (EMP), the output expressions are determined from the input variables that are provided by the EMP implementation.
  In each case, the output expressions form the input parameters of the subsequent destination system call.
- Message delivery
  In the notification process you define how the messages are to be delivered. For method notifications, the corresponding tab is hidden because the destination system call is made directly and not using the message queue.
- Destinations
  You can specify one or more destination systems for a notification message.
  For method notifications you can only define one destination system.

> **i Note**
>
> For EMP methods that do not provide for a destination system call, you can only maintain the conditions for the method notification. None of the other notification attributes are relevant and so they are hidden.

When you add a notification, you can specify, by selecting the relevant option in the dialog box, which type of notification you want the system to create. See also: Functions for Notifications [page 543].

4.4.1 Tag-Based Notifications

Definition

You use tag-based notifications in the classic notification process. Tag-based notifications form the basis for the notification messages that are to be generated and sent later. The notification messages are always generated when events occur that fulfill the defined conditions. A notification message informs the destination system about the event that has occurred. In the classic notification process, the conditions and output expressions are based on the subscription items of the agent instance. The subscription items are the tags that you want to be monitored.

The following types of tag-based notifications are available:

- Static notifications [page 498]
- Versioned notifications [page 498]
- Notification templates (remote subscription) [page 503]

> **i Note**
>
> For PCo to create a notification object, you must **not** select the *Template* checkbox in the dialog box when adding the notification. If you select the *Template* checkbox, PCo creates the notification template [page 503] object. This object is required for integration with the Business Suite only.

Tab Pages

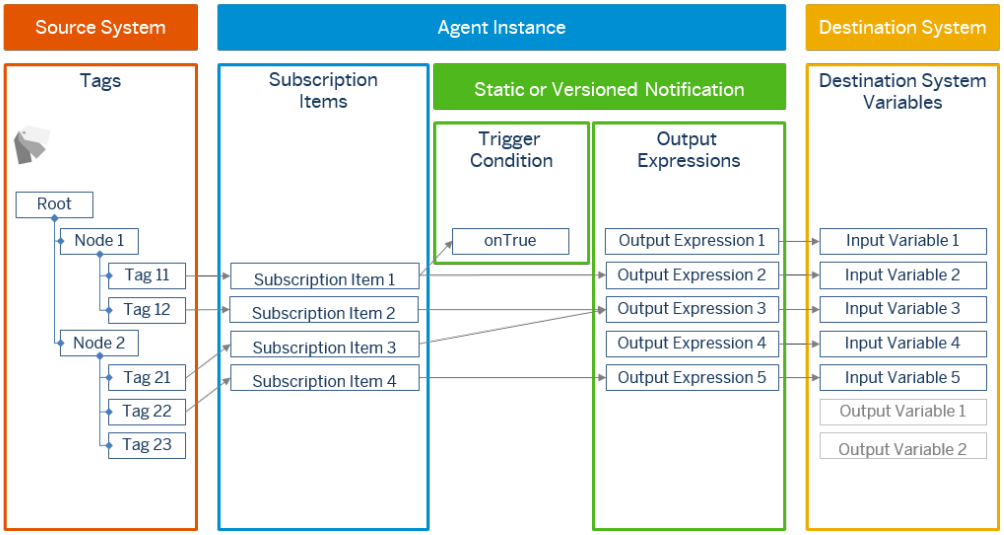
For more information about the tab settings for tag-based notifications, see the following:

- Notification Tab [page 506]
- Output Tab [page 512]
- Message Delivery Tab [page 518]
- Destinations Tab [page 520]

Example

The following example shows the connection between the different configuration elements for tag-based notifications. The example is valid for static and versioned notifications.

Tag-Based Notification (Example)



4.4.1.1 Static Notifications

A static notification corresponds to a normal notification. Normal notifications will be described in future as static notifications if you are using versioned notifications and want to make a distinction between the two types of notifications.

> **i** Note
>
> Existing notifications are automatically converted into static notifications when agent instance configurations are adopted from earlier PCo versions (lower than PCo 15.0 SP03).

The following tabs are available for configuring a normal notification:

- Notification Tab [page 506]
- Output Tab [page 512]
- Message Delivery Tab [page 518]
- Destinations Tab [page 520]

4.4.1.2 Versioned Notifications

Advantages of Versioned Notifications

In PCo, a **versioned notification** consists of one or multiple individual notifications with different versions.

The benefits of using versioned notifications instead of normal (static) notifications are as follows:

- Versioned notifications can be managed and maintained centrally for multiple PCo installations via SAP MII. For more information, see the *SAP Manufacturing Integration and Intelligence* documentation under **Notification Management**. In this case, the versioned notifications can only be displayed in PCo. Changes need to be made directly in SAP MII.
- However, you can create versioned notifications locally for an individual PCo installation via the *PCo Management Console*. The versioned notifications created in the *PCo Management Console* can only be changed in the *Management Console*.
  The benefits of creating versioned notifications locally in the *PCo Management Console* are as follows:
  - In contrast to normal notifications, versioned notifications can be added to a running agent instance and be activated in due course without the agent instance having to be paused. This ensures that notification messages, which have not yet been sent, of the notification that was active until now can continue to be processed without any loss of data.
  - You can pause the output of an individual versioned notification for a running agent and resume it later, without having to stop the agent instance for this.

Versions of a Notification

The following notification versions can occur:

- Draft version
  Each newly created versioned notification first generates a notification with the status *Draft*. These notification versions can also be added when an agent instance is running and can be changed as required. You maintain the subscription items of a draft notification specifically for this notification so that there are no reciprocities or naming conflicts with the subscription items of other notifications. A draft notification does not send any notification messages when an agent instance is running. There can only be one draft version for one versioned notification.
- Current version
  You generate the current version of a notification by activating the draft version. A current version behaves like a normal, static notification and sends notification messages based on the subscription items assigned to it when an agent instance is running. A versioned notification can have maximum one current version.
- Expiring version
  The expiring version of a notification results when a draft version is activated from the previous current notification version. It makes sure that any notification messages in the message queue that are not yet sent can still be delivered. However, the expiring version of a notification no longer receives any data from the source system. Therefore, the assigned subscription items are removed from it. If the agent instance is running, the expiring version is deleted automatically after a specific time, provided that all messages that are still in the queue are processed. If the agent is stopped, you can delete the expiring notification versions manually.

Features

You can interrupt the output of a versioned notification both when an agent is running and when it is stopped. The running and the expiring notification version no longer send any notification messages until you resume

the output again. You can also use versioned notifications within the same agent instance together with normal static notifications.

See Also

Creating and Activating a Draft Version [page 500]

4.4.1.2.1 Creating and Activating a Draft Version

Use

This document describes what you need to do if you want to create versioned notifications locally in PCo.

Procedure

1. Create a draft version
   You can add the draft version of a versioned notification to a stopped agent instance or a running agent instance. (See: Adding a Draft Version [page 501].)
2. Select subscription items
   For versioned notifications, the subscription items are assigned to the individual notification version and not to the entire agent instance. Only the subscription items that are assigned to the current notification version are active. Therefore, you maintain the subscription items only after you have created the draft version of the versioned notification. Using a pushbutton, you can switch from the *Notification* tab of a notification version directly to the assigned subscription items. (See: Subscription Items Tab [page 476].)
3. Define triggers
   You define the trigger conditions on the *Notification* tab. (See: Notification Tab [page 506].)
4. Make settings for message delivery
   You specify how you want PCo to deliver notification messages. (See: Message Delivery Tab [page 518].)
5. Define output expressions
   You define the tags and the context elements that you want to be output in the notification messages. Context elements are pieces of additional business information that you can add to the individual output expressions. (See: Output Tab [page 512].)
6. Define destination systems
   You add one or multiple destination systems to which you want notification messages to be sent. (See Destinations Tab [page 520].)
7. Activate a draft notification
   You activate the draft notification by choosing the *Activate Draft Versioned Notification* pushbutton on the *Notification* tab. (See: Functions for Versioned Notifications [page 502].)

Result

The versioned notification can now be used for sending notification messages. The status *Current* is displayed in the *Version* field.

4.4.1.2.1.1 Adding a Draft Version

Use

You can add the draft version of a versioned notification to a stopped agent instance or a running agent instance. You have several options for this.

Procedure

Create New Without Template

1. Choose the function *Add Notification* either from the context menu of an agent instance, by choosing the pushbutton, or in the *PCo Management Console* menu.
   PCo displays the *Add Notification* dialog box.
2. You define the following in the dialog box:
   - Choose *Versioned Notification* as the notification type.
   - In the *Agent Instance Name* field, specify to which agent instance you want the versioned notification to be added.
   - Choose the name of the versioned notification from the list of existing versioned notifications or enter a new name. The name of the related draft notification is proposed automatically by the system in the *Name* field and cannot be changed.
   - Optionally, enter a description of the versioned notification and confirm your entries.
     The *Notification* tab appears.

Duplicate an Existing Version of a Versioned Notification

1. Choose an existing notification version.
2. Call up the *Duplicate* function from the context menu or from the taskbar of the agent instance tree.
   PCo displays the *Copy Versioned Notification* dialog box.
3. Choose whether you want to create a new versioned notification or want to add a new draft version to the selected versioned notification.
   If you have chosen to create a new versioned notification, the *PCo Management Console* generates this new notification with an automatically generated name that you can change later.

   > ⚠ Caution
   >
   > You can only have **one** draft version for each versioned notification.

4. Save your entries.
   When you duplicate a notification version, the related subscription items are copied in addition to the attributes of the notification.

Copy an Existing Version of a Versioned Notification

1. Choose an existing notification version.
2. Call up the *Copy* function from the context menu or from the taskbar of the Management Console.
3. Select the agent instance to which you want to add the notification version as a copy and choose the *Add* function.
4. If you want to add the notification version to the same agent instance, you can choose, as with duplicating, whether you want to create a new draft notification or a new versioned notification.
   If you want to add the notification to another agent instance, a new versioned notification is always created with an automatically generated name.
5. Save your entries.
   When you copy a notification version, the related subscription items are copied in addition to the attributes of the notification.

> **i** Note
>
> You can use the copy function of a versioned notification to quickly configure agent instances for several source systems of the same type by assigning the configured notification version of the source agent instance to other agent instances using the clipboard. Since the subscription items are also copied, the notifications to the destination agent instances are immediately ready for use after activation. The prerequisite for this is that the source systems have the same type and the same tag structure.

4.4.1.2.1.2 Functions for Versioned Notifications

Use

In the *Versioned Notification* screen area on the *Notification* tab, central functions that affect the entire versioned notification with the related notification versions are offered. These functions lead directly to changes in the configuration database; the display is then refreshed. Therefore, the *PCo Management Console* prompts you to save any unsaved changes to the agent instance before the function is triggered.

Features

Activating the Draft Version of a Versioned Notification

If you choose the *Activate Draft Versioned Notification* pushbutton, the draft version of a notification is converted into a current version. The subscription items assigned to that draft version are activated and the version of the notification that is now current starts sending notification messages.

When a draft version is activated, the version that was current until now (if there was one) becomes an expiring version. An expiring version of a versioned notification does not receive any more changes from the source system, but still processes the notification messages that remain in the message queue.

Pausing the Versioned Notification

You can pause the current notification version by choosing the *Pause Current Versioned Notification* pushbutton. The current version and any expiring notification versions no longer send any notification

messages. Moreover, the current notification version no longer processes any change messages from the source system.

Resuming the Versioned Notification

If you choose the *Resume Paused Versioned Notification* pushbutton, the current version and any expiring notification versions revert to their normal status. The current version processes change messages from the source system again; both versions send notification messages again.

Deleting the Versioned Notification

When you call this function, you can choose between the following options:

- Delete all notification versions
- Only delete current notification versions
  If you choose this option, you can also decide if you want the current version to become an expiring version so that any notification messages still in the queue can be sent.
- Delete draft version only

Unlocking a Versioned Notification

Versioned notifications that were created via SAP MII cannot normally be changed in the Management Console. In exceptional cases, you can choose the *Unlock* pushbutton to make it possible to change the versioned notification in the *Management Console*.

> ⚠ Caution
>
> After unlocking, you can no longer maintain the versioned notification via SAP MII. It is not possible to reverse the unlocking. Therefore, only use this function to clean up configuration inconsistencies and for test purposes.

4.4.1.3 Notification Template

Definition

The notification template is a copy template for notification messages. You need to create the notification template in PCo. For PCo to create a notification template, you need to select the *Template* checkbox in the dialog box when you are adding the notification.

Use

You only need to create the notification template in the case of a **remote subscription**, for example, if you want to connect a Business Suite system or a WebSocket client to PCo as a destination system and want to use the **notification process**. See also: Connection of External Data Sources to Business Suite Applications [page 87] and Process Steps for Remote Subscription [page 95].

> ⚠ Caution
>
> The PCo agent that you want to use for a remote subscription must allow notifications and be query-enabled.

The notification template contains all parameters that are required for a subscription and for the later notification messages, such as:

- Trigger conditions
- Lifetime of subscription
- Settings for reliable message delivery
- Destination systems

For a remote subscription, you do not subscribe a tag in the *PCo Management Console* but in the runtime of the agent in the connected destination system. The subscribed tags are displayed on the *Subscription* tab of the agent instance. The destination system must allow query commands for creating, changing, resubscribing to, and deleting a remote subscription.

If you create a notification by remote subscription from a notification template in the runtime of the agent, the notification created in this way uses the settings from the notification template. The notification is then ready for use and if there are value changes, it creates notification messages that correspond to the defined parameters.

Structure

The following tabs are available:

- *Notification* tab
  Here you can set the trigger conditions of a notification template analogous to those of a regular notification. (See: Notification Tab [page 506].)
- *Remote subscription* tab
  You can define the lifetime of the remote subscription here. You create the remote subscription itself in the connected destination system. (See: Remote Subscription Tab [page 505]
- *Message delivery* tab
  You can specify here how you want PCo to deliver notification messages. (See: Message Delivery Tab [page 518].)
- *Destinations* tab
  On the *Destinations* tab, you enter the destination system for the notifications. (See: Destinations Tab [page 520].)

4.4.1.3.1 Notification Template: Remote Subscription Tab

Use

On this tab, you can specify the lifetime of the remote subscription. You create the remote subscription itself in the connected destination system if it supports the relevant query command for a remote subscription. This is the case, for example, for a WebSocket client.

> ⚠ Caution
>
> The following source systems support the remote subscription:
>
> - **OPC DA**
> - **OPC UA**
> - **OSIsoft PI**
> - **Proficy**
> - **Modbus**

Prerequisites

You have created a notification template [page 503].

Procedure

Click on the *Remote Subscription* tab. The following fields are displayed:

Field	Description
Handle ID	Identifier of the remote subscription. With this identifier, the remote subscription can be rerun, changed, or deleted by a destination system that supports the query commands for the remote subscription. The handle ID is displayed in the notification that was created from a notification template using the remote subscription. The handle ID is empty in a notification template.
Template	States that the notification is a notification template.

Field	Description
Subscription Lifetime	Here you enter the duration in seconds for how long you want the remote subscription that is created from the notification template to be retained in the PCo system. The default value is set as 3600 seconds. This means that the PCo system deletes the remote subscription after an hour has passed if there has been no new remote subscription in that time. **⚠ Caution** If you enter **0**, the lifetime is infinite.
Update Rate	Here you enter the update rate for subscriptions in milliseconds. The subscription update rate indicates the maximum speed at which data changes may be sent using a remote notification for the tags of this subscription. If you enter 0, this means that you want PCo to work with the highest speed that is practically possible. **i Note** The value entered here is only effective in conjunction with OPC DA and Modbus source systems.

4.4.1.4 Configuration of Tag-Based Notifications

The following section provides an overview of how to configure tag-based notifications.

4.4.1.4.1 Notification Tab

Use

On the *Notification* tab, you define the settings that are to be used to trigger a notification message. PCo always sends a notification message to the destination system when the trigger condition, which you use the expression editor to define here, has been met. See also: Expression Editor [page 545] and Formulating and Calculating Expressions [page 547]

For versioned notifications, you can call central functions from this tab, provided the notification was created locally in PCo.

> **i Note**
>
> Versioned notifications that were created in SAP MII cannot be changed in the *PCo Management Console*.

Screen Area: Trigger and Trigger Expression

This screen area is displayed for all tag-based notifications (static and versioned notifications, as well as notification templates).

1. For **static notifications**, check that the *Enabled* indicator has been set if you want the notification to be executed. This is the default setting. For **versioned notifications**, this indicator cannot be changed directly but is set automatically via the status change upon activation and upon pausing and resuming the output.
2. Define the trigger conditions that you want to be checked when an event occurs.
   In the notification process the value change of a subscription item is the event that triggers the check of the trigger conditions. You can formulate the condition expression using the **subscription items**. Depending on the trigger type, the trigger condition must have the value *True* or *False* to trigger the notification. For more information, see Trigger Types [page 508].
3. For **static** and **versioned** notifications, check the setting in the *Triggering Subscription Items* screen area. You can use these settings to specify which subscription items are to trigger a notification. For more information, see Triggering Subscription Items [page 511].

Screen Area: Versioned Notification

The *Versioned Notification* screen area is **only shown if the selected notification is a versioned notification**. The following information is displayed:

Field	Description
Name	Specifies the name of the versioned notification
Description	Specifies the description of the versioned notification
Version	The status of a notification version is displayed in the *Version* field: • Draft • Current • Current - Paused • Expiring • Expiring - Paused For more information, see Versioned Notifications [page 498].

Field	Description
Created by SAP MII	The indicator is set automatically if the notification version has been created by SAP MII. In this case, the notification version can only be changed in SAP MII.

The following functions are provided via pushbuttons for a versioned notification:

- Activate draft versioned notification
- Pause current versioned notification
- Resume paused versioned notification
- Delete versioned notification
- Unlock notification version created by SAP MII

For more information, see Functions for Versioned Notifications [page 502].

4.4.1.4.1.1 Trigger Types

Procedure

1. Set the *trigger type* in the *Trigger* screen area. You use the trigger type to define the trigger conditions that are checked when an event occurs.

   > **i Note**
   >
   > In addition to the rules listed in the following table, the setting in the *Triggering Subscription Items* screen area influences whether a static or a versioned notification is triggered.

Trigger Type	Description
Always	The event always triggers the defined follow-up action. **Example** In the notification process, a notification message is delivered whenever the value of a subscription item changes. *Always* is the default setting. **i Note** For the setting *Always* you do not need to define a condition using the expression editor.

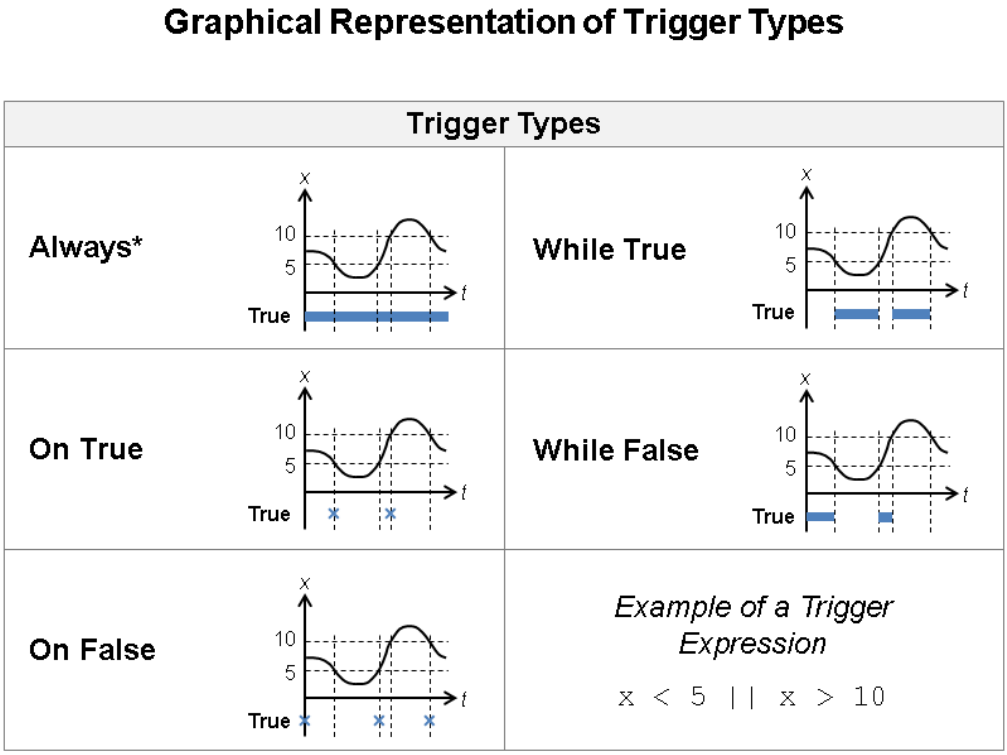
Trigger Type	Description
OnTrue	If you set *OnTrue* and PCo evaluates the trigger expression that you specified as `TRUE`, PCo triggers the follow-up action. The trigger expression must then be evaluated as `FALSE` before it can be evaluated again as `TRUE` for triggering a further follow-up action. You use the setting *OnTrue* if you want to receive a notification message if a rare event occurs. **Example** You are monitoring a boiler with PCo. You specify that the `boiler temperature tag` is to be sent to PCo if the measured value falls short of or exceeds the permissible temperature bandwidth. For this you select the *trigger type* `OnTrue`. You use the **expression editor** to enter the formula `TAG_TEMP<57 || TAG_TEMP>63` as the trigger condition. If the measured value lies below 57°C (134.6°F) or above 63°C (145.4°F), PCo creates a notification message and sends it to the destination system.
OnFalse	With this setting PCo always triggers a follow-up action if the trigger expression that you defined is evaluated as `FALSE`. The trigger expression must then be evaluated as `TRUE` before it can be evaluated again as `FALSE` for triggering a further follow-up action.
WhileTrue	If PCo evaluates your defined trigger expression as `TRUE`, PCo generates follow-up actions. This happens until PCo evaluates the expression as `FALSE`. You choose the setting *WhileTrue* only for events that occur frequently. For **method notifications**, *WhileTrue* has the same effect as *OnTrue*. **Example** You want to monitor a belt that has frequent interruptions. Therefore, you want to receive notification messages from PCo constantly while the belt is running.

Trigger Type	Description
WhileFalse	If your defined trigger expression is evaluated as `FALSE`, PCo generates follow-up actions. This happens until the expression is evaluated as `TRUE`. For **method notifications**, the trigger types *WhileFalse* and *OnFalse* have the same effect. **Example** You want to monitor a belt that has frequent interruptions. Therefore, you want to receive notification messages from PCo constantly while the belt is running.

2. If you select the *trigger type* OnTrue, OnFalse, WhileTrue, or WhileFalse, you need to use the expression editor to create a **Boolean expression** to define the trigger condition. The expression must be able to be evaluated as true or false.
3. Save your entries.

Example

See the graphic for examples of trigger types in the **notification process**:



Trigger Types

4.4.1.4.1.2 Triggering Subscription Items

For **static and versioned notifications**, you can use the settings in the *Triggering Subscription Items* screen area to specify which subscription items are to trigger a notification message.

Settings for Triggering Subscriptions

Field	Description
Only Subscription Items Used in Trigger Condition	The subscription item that is to trigger a notification message due to a value change must be used in the trigger condition of the notification. **i Note** This option **cannot be selected** for the *trigger type* `Always`. In addition, the system checks the trigger condition that you define using the expression editor.
Subscription Items Used in Trigger Condition or Output Expressions	The subscription item that is to trigger a notification message due to a value change **must** be used in the **trigger condition or in the output expressions** of the notification. In addition, the system checks the trigger condition for the *trigger types OnTrue, OnFalse, WhileTrue, and WhileFalse*. For trigger type *Always*, the trigger condition is **not** checked, even if it is displayed in the *Trigger Expression* screen area. The subscription items referenced there are not taken into account either when the notification is triggered.
All Subscription Items	For the trigger type *Always*, **each subscription item** triggers this notification due to a value change. For the trigger types *OnTrue, OnFalse, WhileTrue, and WhileFalse*, the system also checks the trigger condition.

4.4.1.4.2 Output Tab

Prerequisites

On the *Subscription Items* tab you have already added the tags, which you want to monitor, as subscription items. See also: Subscription Items Tab [page 476]

Context

You use the *Output* tab in the notification process [page 35] to define the information that is to be passed on to a destination system using the notification. In the case of the notification process, the output values are formed

from subscription items and, if necessary, a message text. In the simplest scenario, the output expressions have a 1:1 relationship to the subscription items.

However, you can also define complex expressions that combine multiple elements of the data source with one another. You can change the values of the expressions using the expression editor functions, if necessary. The expression editor provides arithmetic functions and string operators for this purpose. The following table contains examples of output expressions based on subscription items.

Relationship of Output Expression to the Subscription Item	Examples of Output Expressions
1:1 relationship between subscription item and output expression: Subscription item **`Item 1`** is assigned to the output expression 1.	`'Item1'`
1:1 relationship between subscription item and output expression: Subscription item **`Item 1`** is assigned to the output expression 1 and, in addition, 10 is to be added.	`'Item1'+ 10`
Several subscription items are combined in one expression.	`'Item2' & "," & 'Item3'`
The output expression has no relationship to a subscription item, for example, if the plant 1000 or the time stamp are to be output.	`"1000"` `datenow`

Procedure

1. Select the notification and click on the *Output* tab.

   The *Output Expressions* table is still empty.
2. In the simplest scenario, where there is a 1:1 relationship between a subscription item and the expression being output, choose the *Generate Expressions* icon.

   With this function, you can generate expressions for the subscription items automatically. These expressions are then displayed in the *Output Expressions* table.

   > **i Note**
   >
   > By clicking on the header row of the table of output expressions, you can sort the table by the content of the selected column.
3. The following additional functions are offered through icons:

Icon	Function Description
Add Expression Using Expression Editor	You can use this function to add an expression using the expression editor. See: Adding an Expression [page 515]
Edit Expression	You can use this function to change an expression selected in the table in the expression editor. See: Formulating and Calculating Expressions [page 547].
Delete Expression	You can use this function to delete a selected expression.
Generate Expressions	You can use this function to automatically generate expressions for subscription items. See: Generating Expressions [page 516]
Determine Name of Selected Expressions	If you have created or renamed output expressions manually, you can use this function to adapt the name to the standard settings. The new name is derived from the subscription item in the expression, as long as this produces a unique name that has not yet been used.
Copy Selected Output Expressions to Clipboard	You can use the clipboard to transfer selected output expressions from one notification to another notification of the same agent instance or another agent instance. This function copies the selected output expressions and the corresponding context items to the clipboard.
Paste Output Expressions from Clipboard	This function inserts the output expressions and their dependent context items from the clipboard.
Remove Context Items from Selected Output Expressions	This function deletes the context items of the selected output expressions. (See also: Maintaining Context Items [page 517].)
Add Context Items from Clipboard to the Selected Output Expressions	You can use the clipboard to copy context items from an output expression and add them to other output expressions. You can use this pushbutton to add the context items that you previously copied to the clipboard to the selected output expressions (see also: Maintaining Context Items [page 517].)
Show Context Items	To create the business context of an expression, you need to choose *Show Context Items*. The *Context Items* screen area then appears. See: Maintaining Context Items [page 517]

Icon	Function Description
Testing a Notification Delivery	You can use this function to generate a notification message in XML format and to send it to the destination system. See: Testing a Notification Delivery [page 517] **i Note** This function is not available for method notifications.

Next Steps

Expression Editor [page 545]

Examples of Message Texts [page 578]

4.4.1.4.2.1 Adding an Expression

Procedure

1. Click the *Add Expression* pushbutton on the *Output* tab.

   The *Expression Editor* dialog box appears.
2. In the lower screen area of the dialog box, select a tag in the *Subscription Item* field.

   The tag is then shown in the *Expression* field.
3. Click *OK*.

   The expression editor is closed. The expression is displayed on the *Output* tab.
4. If necessary, choose another *data type* for the expression. The data type is proposed from the subscription items that are used in the expression.
5. If you have subscribed to more than one tag, you need to add a separate expression for each tag.
6. Select the *Name* field and change the name of the expression as required.
7. To change the expression, click the *Expression* field and then choose the *Edit* pushbutton.

   This opens the expression editor again.

4.4.1.4.2.2 Generating Expressions

Use

In the simplest scenario (a 1:1 relationship between a subscription item and the expression being output), you can choose the *Generate Expressions* pushbutton to generate expressions from subscription items automatically. You can then edit or delete the expressions or add additional expressions. When generating the expressions, the PCo system copies the name of the subscription item, for example, `Double1`, into the *Name* field, and the newly generated expression into the *Expression* field, for example, `'Double1'`. This ensures that the value of the tag is output in the notification message when a value is changed.

Procedure

1. In the *PCo Management Console*, select the notification and choose the *Output* tab.
2. Choose the *Generate Expressions* pushbutton.
   The *Generate Output Expressions* dialog box appears.
3. In the *Generate Output Expressions* dialog box, select one of the three options for the automatic generation of output expressions:
   - *Generate Only Missing Output Expressions*
     The system checks which subscription items already have an output expression with the same name and generates new output expressions only for the missing subscription items. This is the default setting.
   - *Regenerate All Output Expressions*
     The system generates output expressions for all subscription items. If an output expression with the same name already exists, it is replaced by the standard expression. Output expressions added manually previously remain unchanged. If data types of subscription items have changed in the meantime, the system automatically adjusts the data types of the output expressions.
   - *Generate Output Expressions for Subscription Items with Matching Source Names*
     With this option, you can control the copying of subscription items to output expressions by entering filter criteria that consider the source name of the subscription items.
     - If the *Regular Expression* checkbox is **not** selected, all subscription items, whose source name contains the filter text entered, are selected. This check is not case-sensitive.
     - If you select the *Regular Expression* checkbox, the system selects the subscription items whose source name matches the regular expression entered. This evaluation is case-sensitive.

> **Example**
>
> A subscription item has the source name `Data Type Examples.16 Bit Device.K Registers.Double4`. If you enter the filter text **`k registers`**, this subscription item would be selected, as well as all other subscription items with the text *K Registers* in the source name.

> **Example**
>
> If you specify the regular expression **`^(.*)16 Bit Device(.*)$`**, all subscription items that have the string *16 Bit Device* in the source name would be selected.

4. To change a generated expression, select it and choose the *Edit Expression* pushbutton. This opens the expression editor. The expression editor can also be started by double-clicking the expression.

4.4.1.4.2.3 Maintaining Context Items

1. To maintain context items, you need to display the *Context for Selected Output Expression* screen area. To do so, choose the *Show Context Items* pushbutton.
   The *Management Console* then shows the screen area for context items. It remembers the display status of this screen area for all additional calls.
2. To **create a context item**, you select the expression to which you want to add a notification context and choose the *Add Context Item* pushbutton in the screen area for context items.
   The Management Console shows a new row in the lower screen area.
3. Enter a *name* for the context item, choose a *data type*, and assign a *value* to the context item.
4. Confirm your entries by pressing the Tab key or by clicking with the mouse on another screen area.
5. To be able to **assign context items that are already created to one or multiple other expressions via the clipboard**, you do the following:
   - Select one or more context items and choose the *Copy Context Items to Clipboard* pushbutton.
   - Select the expression to which you want to add the context items in the clipboard and choose the *Add Context Items from Clipboard* pushbutton.
   - Alternatively, you can select several rows in the table of expressions and choose the *Add Context Items from Clipboard to Selected Output Expressions* pushbutton.
6. To **delete context items**, select the context items that you want to to delete and choose the *Delete Context Item* pushbutton. Alternatively, you can select one or multiple expressions and, in the screen area for expressions, choose the *Remove Context Items from Selected Output Expressions* pushbutton to remove all context items from the selected expressions.

> **i** Note
>
> The *Management Console* indicates an error if you assign multiple context items with the same name to one expression. If the systems that are receiving and processing the notification messages do not support this case, you can correct your entries accordingly. However, the *Management Console* allows you to save context items with the same name for one expression and allows the sending of notification messages generated from this.

4.4.1.4.2.4 Testing a Notification Delivery

Procedure

1. To test the notification, choose the *Test Notification Delivery* pushbutton on the *Output* tab.

   The *Notification Test Dialog* dialog box appears.

2. Choose the proposed *Destination*.

> **i Note**
>
> To be able to select a destination, you need to have entered a destination system for the notification on the *Destinations* tab (for example, SAP MII).

Enter a value for the expression, choose an appropriate data type, and click on *Deliver* to test the notification delivery.

The PCo system then generates a sample message in XML format and sends this notification message to the destination system. If, for example, you have set SAP MII as the destination system and have defined an e-mail as the transaction, SAP MII sends a notification message by e-mail.

3. Choose the *View Sample* pushbutton to display the notification message in XML format.
4. Choose the *Save Sample* pushbutton to save the notification message.

4.4.1.4.3 Message Delivery Tab

Context

On this tab, you define for a notification how you want PCo to deliver notification messages.

For tag-based notifications, this tab is displayed as standard. For method notifications, the *Message Delivery* tab is only displayed if you have selected the *Asynchronous* checkbox on the *Server Method Definitions* tab (OPC UA server or PCo Web server) for the relevant method.

Procedure

1. On the *Plant Connectivity Management Console* screen, select a notification and click on the *Message Delivery* tab.
2. In the Reliability screen area, enter the required data for **reliable message delivery** using the following table:

Field	Description
Max. Number of Retries	You can enter the maximum number of retries here. The default value is **0**. This means that the retry process is switched off for newly created notifications. In this case, the retry interval is not relevant and is therefore set to 0.

Field	Description
Retry Interval (Seconds)	Here you can specify the number of seconds before the next attempt to send the message. The default value is **0 seconds**.

3. In the *Failed Message Persistence* screen area, specify how you want failed notification messages to be stored. The failed messages are displayed in the agent instance on the *Message Failures* tab. You have the following options:
   - *Keep All Messages*
     Saves all failed messages.
   - *Keep First Message*
     Saves the first failed message and ignores subsequent messages.
   - *Keep Last Message*
     Saves the most recent failed message and removes previous messages.
   - *Keep Messages Until Expiry*
     You use this setting to specify that all failed messages that are older than the time you entered in the *Delete Messages After* field are to be deleted. For example, if you entered `60 min` and a new failed message comes along, all stored failed messages that are older than 60 minutes are deleted.
   - *Keep No Messages*
     Does not save any failed messages.
4. In the *Lifetime* screen area, specify how long you want messages to be kept in the queue.

   If the notification message cannot be delivered after the specified duration has elapsed, it is displayed in the agent instance on the *Expired Messages* tab.
5. In the *Message Bundling* screen area, you can activate the **message bundling** function by selecting at least one of the checkboxes:
   - *Fixed Number of Messages*
   - *Maximum Accumulation Time*

   Message bundling is a procedure in which notification messages can no longer be sent individually; instead, they are bundled in one large notification message and sent together to the connected destination system.

   For more information, see Message Bundling [page 579].
6. Select the *Process Notification Messages Exactly Once in Order* checkbox if you want to use this function for the notification. By selecting the checkbox, you can define that the notification messages of this notification are sent individually and exactly in the order in which you put them into the queue. For more information, see Process Notification Messages Exactly Once in Order [page 487].

Next Steps

Agent Instance: Displaying Messages [page 490].

4.4.1.4.4 Destinations Tab

Prerequisites

- You have created a destination system and defined the connection data.
- If you are using enhanced notification processing, you first need to have created a destination system for the notification enhancement on the *Notification Processing* tab of the agent instance. (See Enhanced Notification Processing [page 73].)

Context

On this tab, you enter the destination systems that you want the notification to use. In the **notification process**, you configure one or multiple destination systems to which you want the notification messages to be sent.

The *Destinations* tab is used to configure a notification as well as to configure a **notification template**.

> **i** Note
>
> In the case of a method notification, you configure exactly one destination system that you want to be called when the method call is being processed. The *Destination* tab is shown for this purpose. (See: Destination Tab [page 542].)

Procedure

1. On the *Plant Connectivity Management Console* screen, select a notification and click on the *Destinations* tab.
2. Click the *Add Destination System* icon.

   The *Add Destination* dialog box appears.
3. In the dropdown box, select the destination system that you want to use for your notification.

There is a detailed procedure for the following destination systems:

Destination Systems

Destination System	Description
SAP MII Destination System (Deprecated)	Choose the MII destination system and enter the MII transaction. See: Define Data for SAP MII System [page 522]. **i Note** This option is only still available for compatibility reasons. Instead, use the **call of MII transactions using Web services**.
Calling MII Transactions Using Web Services	To be able to call an MII transaction in an MII system as part of a notification process, SAP recommends that you create one of the following destination systems and select it here on the *Destinations* tab: ○ Universal Web service destination system ○ Multiple call destination system ○ OPC UA destination system See also: Assignment of Notification Output (Call of MII Transactions) [page 523].
Universal Web Service Destination System	Select the universal WS destination system. See: Assignment of Notification Output (Universal WS Destination System) [page 524].
Multiple Call Destination System	Select the multiple call destination system. See: Assignment of Notification Output (Multiple Call Destination System) [page 525].
OPC UA Destination System	Select the OPC UA destination system. See: Assignment of Notification Output (OPC UA Destination System) [page 526].
Enhanced Notification Processing (ENP)	Select the */Enhanced Notification Processing* entry (see: Assignment of Modules and Variables (ENP) [page 75].)
ODBC Destination System	See: Assignment of Destination System Variables (ODBC Destination System) [page 345].
Web Service Destination System (Deprecated)	See: Assignment of Notification Output (WS Destination System) [page 536].

4. Choose a name and a description for the destination system assignment.

   The name of the destination system assignment is relevant if you have assigned multiple destination systems in a tag-based notification and have selected the *Process Notification Messages Exactly Once in Order* setting. In this case, the name of the destination system assignment determines the sequence in which notification messages are placed in the queue.
5. Assign output expressions to the destination system variables in the *Output Destination Mapping* area and, if necessary, make further settings for the destination system.

6. Click *OK*.

4.4.1.4.4.1 Define Data for SAP MII System

Prerequisites

- You have created a destination system for SAP MII. (See MII Destination System [page 391].)
- In the SAP MII system, you have defined a transaction for the notification, for example, `Send Mail`. (See Notification Process (with SAP MII) [page 35].)

Procedure

1. In the *Add Destination System* dialog box, choose the MII system to which you want to send the notification messages and specify a name. Click OK.
2. Expand the entry and click on *Destination*.

   PCo displays the directory structure of the *SAP MII Server* and displays additional fields in the lower screen area.
3. Click on the directory in which you have stored the MII transaction. Then enter the following data:

Field	Description
Transaction Name	If you click on your MII directory, the previously defined transactions are displayed. Click on the transaction you want. The transaction name then appears in the *Transaction Name* field.
Input Parameter Name	When the MII transaction is selected, this field is supplied with the parameters of the selected transaction. The transaction parameters are shown in this field. Select the input parameters you want.
Use Asynchronous Mode	The notification is sent to SAP MII asynchronously (if selected).

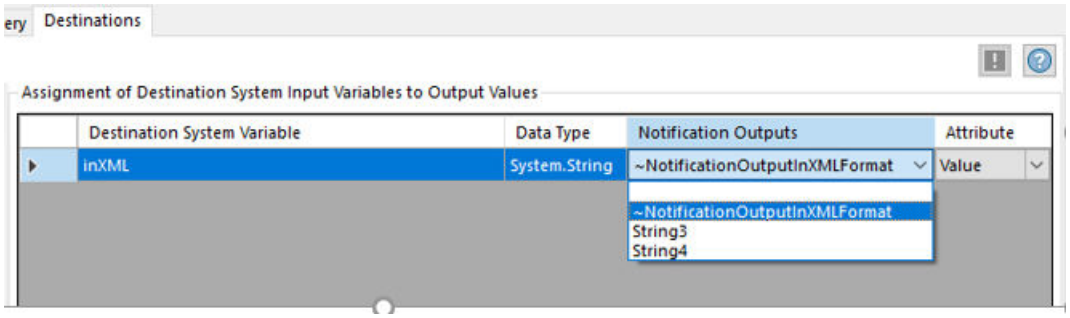
Field	Description
Persistence	Determines whether the transaction is to be saved in SAP MII: ○ *Always Save Transaction in MII* With this option, the transaction is always saved in SAP MII. ○ *Do Not Save Transaction in MII* With this option, the transaction is never saved in SAP MII. ○ *Only Save Transaction in MII If Errors Occur* With this option, the transaction is only saved in SAP MII if an error occurs.

4.4.1.4.4.2 Assignment of Notification Output (Call of MII Transactions)

In addition to the output expressions you have configured for some destination system types, the dropdown list in *the Assignment of Notification Output* area offers the special expression `~NotificationOutputInXMLFormat` that contains the content of the notification in XML format at runtime. You can use this output value to call an existing MII transaction, which was configured originally as the destination of an MII destination system, using a Web service.

Example

The example shows the assignment (mapping) of the notification output `~NotificationOutputInXMLFormat` to the destination system variable `inXML`:



Related Information

Settings for the Notification Process with SAP MII [page 37]
Assignment of Web Service Call as Destination of a Notification [page 42]

4.4.1.4.4.3 Assignment of Notification Output (Universal WS Destination System)

Prerequisites

- You have created a universal Web service destination system and configured the service operation and variable definition of the selected Web service for the destination system.
- On the *Output* tab for the notification, you have defined expressions and output values.

Procedure

1. Click the *Add Destination System* pushbutton on the *Destinations* tab.
   The *Add Destination System* dialog box appears.
2. In the *Destination System* field, select the universal Web Service destination system that you created previously. In the *Name* field, enter a name of your choice for the destination system and choose *OK*.
3. Expand the entry for the destination system and click on *Output Destination Mapping*.
   The assignment table containing the available *destination system variables* of the Web service opens on the right.
4. Assign a value from the *Notification Output Values* column to each destination system variable.

   > **i Note**
   >
   > Choose the *Propose Assignment* pushbutton if you want the system to propose the appropriate output values. The prerequisite for automatic assignment is that the Web service variable and the output expression have the same name.

5. Choose the *attribute* to be output for each output value. You can choose the following attributes:

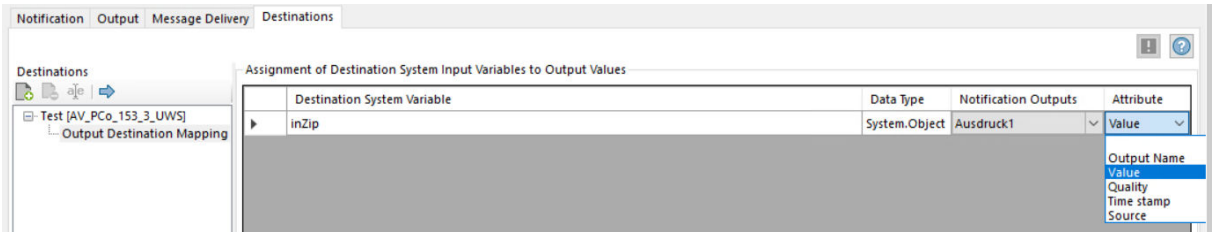
   Attributes

Attribute	Description
Output Name	Name of output value
Value	The actual value of the output value; this attribute is mapped by default.

Attribute	Description
Quality	The data quality of the tag on which the output value is based
Time Stamp	The time stamp of the last change to the output value
Source	The source of the tag on which the output value is based; this attribute is taken from the subscription item, provided the expression for the output value contains exactly one subscription item.

Example

The example shows the assignment (mapping) of the notification output to a variable:



4.4.1.4.4.4 Assignment of Notification Output (Multiple Call Destination System)

Procedure

1. Click the *Add Destination System* pushbutton on the *Destinations* tab.
   The *Add Destination System* dialog box appears.
2. In the *Destination System* field, select the multiple call destination system that you created previously. In the *Name* field, enter a name of your choice for the destination system and choose *OK*.
   The system now displays the destination system in the *Destinations* screen area.
3. Expand the entry for the destination system and click on *Output Destination Mapping*.

The following fields are displayed in the *Call Mode* screen area:

Call Mode

Field	Description
Enter at First Step	Select this option if you want processing of the multiple call destination system to start at the first step. This is the default option.
Resume	You can only select this option if you are using the **Suspend and Resume** function and you want processing of the multiple call destination system to be resumed at a step other than the first step. The following two fields then become ready for input.
Output Expression Containing the Resume Handle	Choose the output expression that contains the value of the resume handle at runtime. The resume handle is used to restore the status in which processing was suspended.
Destination Call Step to Resume	Choose the step of the multiple call destination system where execution of the process was suspended and for which it should be resumed.

In the lower screen area, the system displays the *Assignment of Destination System Variables to Output Expressions* table. The destination system variables are already shown.

4. Assign an output expression and an attribute to each variable:
   - If you have selected the *Enter at First Step* option, assign the output expressions to the **input variables** of the multiple call destination system.
   - If you have chosen the *Resume* option, assign the output expressions to the **resume variables** of the multiple call destination system.

4.4.1.4.4.5 Assignment of Notification Output (OPC UA Destination System)

Prerequisites

- You have created an OPC UA destination system and configured the method to be called for the destination system by browsing in the OPC UA server. This also defines the input and output parameters of the method and you can call the method as the destination system of a tag-based notification.

  > **i** Note
  >
  > If you call the **method as the destination** of a tag-based notification, you can **only supply the input parameters of the method with values**, but you cannot receive or process any output parameters of the method. Any return values are ignored. Exceptions during the method call cause the notification message to fail.

If your method has output parameters and you want to process the output parameters and handle the exceptions of the method, you should call the OPC UA method indirectly using a multiple call destination system, and configure this multiple call destination system as the destination of the notification.

- On the *Output* tab for the notification, you have defined expressions and output values.

Procedure

1. Click the *Add Destination System* pushbutton on the *Destinations* tab.
   The *Add Destination System* dialog box appears.
2. In the *Destination System* field, select the OPC UA destination system that you created previously. In the *Name* field, enter a name of your choice for the destination system and choose *OK*.
3. Expand the entry for the destination system and click on *Output Destination Mapping*.
   On the right, the assignment table opens with the existing input parameters of the OPC UA method.
4. Assign a value from the *Notification Output Values* column to each input parameter.

   > **i Note**
   >
   > Choose the *Propose Assignment* pushbutton if you want the system to propose the appropriate output values. The prerequisite for the automatic assignment is that the method parameter and the output expression have names that are at least in part similar.

5. Choose the attribute to be output for each output value. You can choose the following attributes:

   Attributes

Attribute	Description
Output Name	Name of output value
Value	The actual value of the output value; this attribute is mapped by default.
Quality	The data quality of the tag on which the output value is based
Time Stamp	The time stamp of the last change to the output value
Source	The source of the tag on which the output value is based; this attribute is taken from the subscription item, provided the expression for the output value contains exactly one subscription item.

4.4.1.4.4.6 Assignment of Modules and Variables (ENP)

Use

In the screen area *Module and Variable Assignment*, you define the modules and variables for enhanced notification processing (ENP).

Prerequisites

- On the *Notification Processing* tab, you have chosen either *Destination System Calls with Response Processing* or *Customer-Owned Enhancement* for your agent instance and you have created the destination system for the agent instance. (See: Enhanced Notification Processing [page 73].)
- You have created one or multiple destination systems. The following destination system types are supported:
  - Web service destination system [page 393]
  - Destination system type
  - ODBC destination system [page 338]

  For each Web service destination system, you have also configured the service operation and variable definition of the selected Web service. (See: Destination System: Operation Configuration Tab (WS Destination System) [page 396].)

Procedure

Add Destination System

1. In your notification, choose the *Add Destination System* icon on the *Destinations* tab.
   The *Add Destination System* dialog box appears.
2. In the *Destination System* field, select the entry */Enhanced Notification Processing*.
3. In the *Name* field, enter a name of your choice for the destination system and choose OK.
   You have assigned the destination system for enhanced notification processing. The system now displays the destination system in the *Destinations* screen area.

Agent and Destination System Calls

1. Expand the entry for the destination system and choose the *Module and Variable Assignment* row.
   The following two assignment tables are now displayed:
   - *Agent and Destination System Call Modules*
   - *Assignment of Enhancement Variables*

2. In the *Agent and Destination System Call Modules* table, assign the agent instances and destination systems (*Web Service Destination systems*) that you want to call using enhanced notification processing. The table is structured as follows:

Column	Description
Module	All the modules that are provided by the implementation of notification processing are displayed in this column. The modules set up the connection to agent instances and destination systems that are to be called up from ENP.
Module Type	Specifies the type of module. The following module types are possible: ○ *Destination system call* You can use this module to call one of the supported destination systems. You must assign the destination system in the *Destination System* column. ○ *Agent call* From ENP, you can use this module type to specify that data coming from the connected destination system is transferred to or read by the current or other running agent instances.
Exception Handling	Select this checkbox if you want to deal with exceptions that occur in the destination system call in a controlled way. (See: Exception Handling in Enhanced Notification Processing [page 79].) If you do not select this checkbox, exception handling is not enabled. Without exception handling, every exception within the destination system leads to termination of notification message processing and to an error message in the agent log.
Edit	To configure exception handling, choose the *Edit* pushbutton.
Destination System	For a module of the type *destination system call*, you must select a destination system here. Only the permitted destination systems are offered in the dropdown box.
Agent Instance	In the case of a module of the type *Agent Call*, you must select an agent instance here. Only the permitted agent instances are offered in the dropdown box.

Assignment of Enhancement Variables

1. The *Destination System Variables* and *Source System Tags* tabs are displayed in the lower screen area *Assignment of Enhancement Variables*.
2. Choose the *Destination System Variables* tab. All the destination system variables that PCo determined for the destination system selected previously are displayed here. Now assign the appropriate notification

enhancement variables to the destination system variables. The enhancement variables are offered in a dropdown box. The assignment table is as follows:

Column	Description
Destination System Call Module	All the modules for calling the destination system(s) are displayed here.
Destination System Variables	All destination system variables are displayed here that are determined from the configured destination system: ○ For an *ODBC destination system*, the system displays the table columns configured as variables. ○ In the case of a destination system of the type *Web service destination system*, you see the input and output parameters configured as variables as well as the calculated variables (only for Web service destination systems).
Category	Shows if the variable is a request variable, a response variable, or a calculated variable.
Data Type	Indicates the data type of the destination system variables.
Enhancement Variable	Here you assign the appropriate enhancement variable to each destination system variable. In the dropdown box, the system only offers the enhancement variables that you have defined in your implementation or that were provided by the SAP standard enhancement. You can only assign variables with matching data types.

> ⚠ Caution
>
> Note that you need to assign an enhancement variable to each request variable. However, you do not need to assign an enhancement variable to each response variable and to each calculated variable.

3. Choose the *Propose Assignment* pushbutton if you want the system to propose the appropriate enhancement variables. The prerequisite for this is that the enhancement variables have the same name or a similar name to the destination system variables, and have the appropriate data type.
   You can use this function most effectively when configuring the SAP standard enhancement *Destination System Calls with Response Processing* because same-name variables are used by preference there. (See also: Destination System Calls with Response Processing [page 81].)
4. Choose the *Source System Tags* tab. Here you can browse for tags for the selected agent instances and assign them to the enhancement variables. You only need to select these source system tags if you want to read or write tag values by means of an agent instance in enhanced notification processing. (See: Assigning Source System Tags [page 78].)

5. If the names or data types of the modules and the variables are changed after you have configured them in the *PCo Management Console*, you receive a warning when you call the configuration dialog again. The system then removes invalid configurations automatically and you can then complete the configuration.

See Also

For more information about the assignment of modules and variables, see the *Implementation Guide for Enhanced Notification Processing (ENP)* on *SAP Service Marketplace* under *⁄instguides* > *SAP Business Suite Applications* > *SAP Manufacturing* > *Plant Connectivity* > *Plant Connectivity 15.1* .

4.4.1.4.4.6.1 Exception Handling

Prerequisites

- In enhanced notification processing:
  You have configured an agent instance that is to call up one or multiple destination systems using a customer-owned enhancement or the SAP standard enhancement *Destination System Calls with Response Processing*.
  You call the *Destinations* tab in the **notification**. You have assigned a *Web Service Destination system* to a module of the type *destination system call* here. See: Assignment of Modules and Variables (ENP) [page 75].
  You have activated exception handling for this module by selecting the *Exception Handling* checkbox.
- In the multiple call destination system:
  You have created a multiple call destination system with which one or multiple destination systems are to be called in a specific sequence. On the *Destination System Calls* tab, you have selected the *Exception Handling* checkbox in the *Sequence of Destination System Calls* table.

Context

You can react in a controlled way to exceptions, which can occur when destination systems are called, by configuring exception handling so you do not have to terminate the destination system call process but rather can continue working with defined values for the destination system output variables.

Exception handling can be called in the following applications:

- Enhanced notification processing
  You can activate exception handling for enhanced notification processing, for example, using the SAP standard enhancement *Destination System Calls with Response Processing* or a customer-owned enhancement.

- Multiple call destination system
  You can activate exception handling for individual steps in a multiple call destination system.

Errors can occur within the processing logic of a called destination system that go back to PCo as exceptions.

Some exceptions indicate actual errors, such as a destination system that is temporarily unavailable, incorrect input parameters, or errors in the processing of calculated variables in a Web Service Destination system.

However, some exceptions can be foreseen from the process flow or can be tolerated and do not need to lead to the termination of the processing of a notification message.

> **Example**
>
> A foreseeable exception might occur if in SAP ME an *SFC number* is to be set to *done* using PCo but this SFC was already set to *done* in a manual process. In this case, SAP ME sends an error message. This exception should no longer lead to an error message in the agent instance log but should rather be caught by exception handling.

Procedure

1. Choose  .

   The *Exception Handling for Module* dialog box appears.
2. Define a logical condition in the *Condition* field. Use the expression editor to formulate the condition. You can evaluate notification enhancement variables and the text for the exception message in the condition expression (variable `!EXCEPTION_MESSAGE!`). A return value of 0 means that the condition is false. A return value that does not equal 0 means that the condition is true. An empty condition expression has the same meaning as the return value 0, in other words, the condition is false.

   > **Example**
   >
   > You would like to define that the condition is true if the exception text contains the text "*Message 13979*". In this case, the condition expression is: `stringindexof('!EXCEPTION_MESSAGE!', "Message 13979")`
3. Specify what you want to happen in the case of a true condition and a false condition. In both cases, you can choose one of the following two options:
   - *Raise Exception*
     The exception is not caught. It is forwarded to the enhanced notification processing (ENP) that called the destination system. This corresponds to the behavior without exception handling. In this case, processing of the notification message is terminated.
   - *Catch Exception and Set Destination System Output Variables*
     The exception is caught. You can define the values for the destination system output variables in the *Set Value of Destination System Output Variables* table. You can use fixed values or expressions to determine the values.
     If an exception is caught, processing of the notification message continues. The return values then correspond to the values that you defined here in the table.

4.4.1.4.4.6.2 Assigning Source System Tags

Context

On the *Source System Tags* tab, you can search for tags for the selected agent instances and assign them to the enhancement variables. You only need to select source system tags if you want to read or write tag values by means of an agent instance in response processing within enhanced notification processing.

Procedure

1. Click on the *Source System Tags* tab.

   The tab is displayed.
2. Choose the *Insert Row* or *Append Row* pushbutton.

   PCo displays a new row with the agent call module *agent*.
3. Choose the *Browse* pushbutton.

   PCo displays a dialog box.
4. If the namespace is empty, click *Browse* (next to the *Filter* field).

   The namespace of the source system connected via the agent appears.
5. To be able to display the entire hierarchy of the namespace, open the top node (*Root*) and then all lower-level nodes.
6. Click to select the tag or tags that you want and then choose *Add Selected Items*.

   The selected tags are displayed in the lower screen area (*Selected Objects*).
7. To close the dialog box, click *OK*.

   PCo now adds the selected tags to the *Source System Tags* tab. If you have selected more than one tag, the system creates a new row for each tag.
8. Now you assign the appropriate enhancement variable to each tag. The variables can be selected from the dropdown box.

   > **i Note**
   >
   > The correct data type can only be determined for an enhancement variable directly after the tags are browsed. If you leave the *Source System Tags* tab without first assigning the variables, the rows without variable assignment cannot be configured later. In this case, you need to delete these rows and select tags again.

4.4.1.4.4.7 Assignment of Destination System Variables (ODBC Destination System)

Prerequisites

You have created a notification and automatically generated the output values on the *Output* tab by choosing the *Generate Expressions* pushbutton.

Context

In this processing step, you need to assign the output values of the notification to the variables of the ODBC destination system that you created on the *Configuration* tab. (See: ODBC Destination System: Configuration Tab [page 340]. This is the prerequisite for the connected database table being filled with tag values as soon as the agent instance is running.

Procedure

1. Click the *Add Destination System* pushbutton on the *Destinations* tab.

   The *Add Destination System* dialog box appears.
2. In the *Destination System* field, select the ODBC destination system that you created previously.
3. In the *Name* field, enter a name of your choice for the destination system and choose *OK*.

   The system now displays the destination system in the *Destinations* screen area.
4. Expand the entry for the destination system and click on *Output Destination Mapping*.

   The assignment table containing the destination system variables of the ODBC destination system opens on the right. The table consists of four columns with the following meaning:

Column	Meaning
Destination System Variable	The name of the variable that you created previously on the *Configuration* tab of the ODBC destination system is displayed here. (See: ODBC Destination System: Configuration Tab [page 340].)
SQL Type	The SQL type of the destination system variable is displayed here, for example, NVARCHAR.

Column	Meaning
Notification Output	For each variable, choose an appropriate output value from the dropdown box.
Attribute	For each variable, choose an appropriate attribute from the dropdown box. You can choose from the following four attributes: ○ *Time Stamp* The time stamp of the last change to the output value ○ *Quality* The quality of the output value, for example, *Good* or *Bad* ○ *Value* The actual value of the output value ○ *Output Name* The name of the output value ○ *Source* The source of the tag on which the output value is based. This attribute is taken from the subscription item, provided the expression for the output value contains exactly one subscription item. PCo checks if the assignment is correct and issues a corresponding error message if necessary.

5. Assign each destination system variable a value in the *Notification Output* column and an *Attribute*.

   After you have made the assignment, the table looks like this, for example:

Destination System Variable	SQL Type	Notification Output	Attribute
TS1	TIMESTAMP	Int1	Timestamp
VAL1	NVARCHAR	Int1	Value
TS2	TIMESTAMP	Int2	Timestamp
VAL2	NVARCHAR	Int2	Value

6. Save the assignments.

Results

You can now start your agent instance in the *PCo Management Console*. PCo transfers the current tag values to the database table. You can now check, for example, in the *SAP HANA Studio*, if the new data records have arrived in your history table.

4.4.1.4.4.8 Assignment of Notification Output (WS Destination System)

Prerequisites

- For the third-party system that is to be connected, for example, SAP ME, you have created a Web Service destination system, and for the destination system you have configured the service operation and variable definition of the selected Web service. (See also: Operation Configuration Tab (WS Destination System) [page 396].)
- On the *Output* tab for the notification, you have defined expressions and output values. (See also: Integration with Third-Party Systems Using Web Services (WS Destination System) [page 66]

Procedure

1. Click the *Add Destination System* pushbutton on the *Destinations* tab.

   The *Add Destination System* dialog box appears.
2. In the *Destination System* field, select the Web Service destination system that you created previously. In the *Name* field, enter a name of your choice for the destination system and choose OK.

   The system now displays the destination system in the *Destinations* screen area.
3. Expand the entry for the destination system and click on *Output Destination Mapping*.

   The **assignment table containing the destination system variables of the Web service** opens on the right.
4. Assign a value from the *Notification Outputs* column to each variable.

   > **i Note**
   >
   > Choose the *Propose Assignment* pushbutton if you want the system to propose the appropriate output values. The prerequisite for automatic assignment is that the Web service variable and the output expression have the same name.
5. Choose the attribute to be output for each output value. You can choose the following attributes:
   - Value
     The actual value of the output value; this attribute has been assigned as standard up to now
   - Time stamp
     The time stamp of the last change to the output value
   - Quality
     The data quality of the tag on which the output value is based
   - Name
     The name of the output value
   - Tag source

The source of the tag on which the output value is based; this attribute is taken from the subscription item, provided the expression for the output value contains exactly one subscription item.

6. After you have made the assignment, the table looks like this, for example:

Destination System Variables (Web Service Variables)	Notification Outputs	Attributes
SFC	**SFC**	**Value**
SITE	**SITE**	**Value**
OPERATION	**OPERATION**	**Value**
DATA_SOURCE	**DATA_TAG**	**Source**
DATA_TIMESTAMP	**DATA_TAG**	**Source**

4.4.2 Method Notifications

Use

You can use a method notification to activate a method of a PCo OPC UA server or a PCo Web server and define which action you want to run if the method is called up by a client. See also: Methods [page 11].

The functions offered in the method notification are related to the notification in the notification process but differ in detail, depending on the type of method.

- For methods that you have configured manually (methods of the type *direct destination system call*), you define the trigger condition based on the method input parameters. You then configure the output expressions that are forwarded to the input variables of the destination system call. The output expressions are also based on the method input parameters. You can assign exactly one destination system to the notification. If the destination system delivers return values, such as the OPC UA destination system, Web service destination system, or the multiple call destination system, you can assign the output variables of the destination system call to the output parameters of the method.
- For methods that you have loaded into the PCo configuration using an EMP implementation, a distinction is made between methods with and methods without a destination system call.
  - For **EMP methods without a destination system call**, you can only configure the trigger condition based on the method input variables. Processing of the method call is adopted directly from the implementation. The input and output parameters of the method are received directly from the implementation or returned to PCo.
  - For **EMP methods with a destination system call**, you can also define a trigger condition. The implementation of method processing provides for a destination system call at the appropriate place in the logic. The EMP implementation provides special input variables for the destination system call that you need to assign to output expressions in the method notification.
    You can **configure exactly one destination system in the notification**. The output variables of the destination system call are assigned to the output variables provided by the EMP implementation so that the implementation can continue to work with the result of the destination system call. The input

and output parameters of the method are received directly from the EMP implementation or returned to PCo.

Runtime Behavior of Method Calls

You can create multiple method notifications for a method. However, you need to maintain the trigger conditions so that no more than one of the notifications is executed at runtime. Depending on the input parameters of the method, you can trigger various actions, for example, various destination system calls.

> **i** Note
>
> If no method notification is executed at runtime due to the trigger conditions, process control goes back directly to the caller. Any output parameters are supplied with their initial value.

The method notification is processed directly when the method is called; in other words, in technical terms, in the same thread of the Windows process through which the agent instance is running. If the method is configured as a **synchronous method**, PCo waits until the end of processing of the destination system call or of processing in the EMP implementation. Only then does process control go back to the caller, with the output parameters being supplied according to the configuration. For **asynchronous methods**, process control returns to the caller immediately after processing is triggered while the destination system call runs in an asynchronous thread in the background. Any output parameters of the method are supplied with the initial value.

Configuration Options

The method notification reveals its potential in particular in conjunction with the multiple call destination system and the definition of suitable trigger conditions. The configuration involves the following steps:

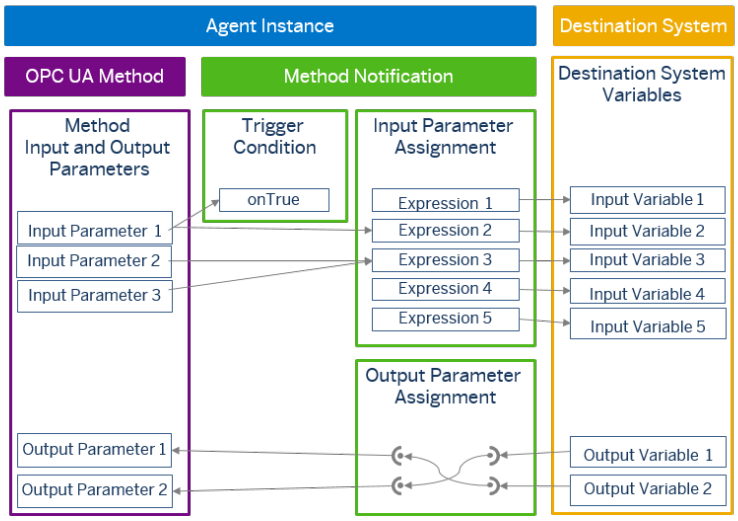
- You define a method for the direct destination system call. See also: Server Method Definitions Tab (OPC UA Server) [page 461] or Web Server Method Definitions Tab (Web Server) [page 469].
- You create a method notification for the method.
- You configure a multiple call destination system as a destination system, which you supply with the input parameters of the method.
  In the multiple call destination system, you can program complex processes, such as the multiple call of various destination systems, processing of interim results, branching conditions, and loops using configuration. See also: Multiple Call Destination System [page 353].
- By creating multiple method notifications for the same method that have different trigger conditions, you can, depending on the input parameters of the method, call different multiple call destination systems that model different processes.

> **i** Note
>
> If the configuration options described are not flexible enough, you can implement an enhanced method processing (EMP) in which you can program the actions when calling a method.

Example

The example shows variable assignment in method processing with a direct destination system call. The example also applies to Web service methods.

Method Notification (Example)



4.4.2.1 Notification Tab (Method Notification)

Use

On the *Notification* tab, you define the settings that are to be used to trigger notification processing. PCo executes the action linked to the method notification if the trigger condition, which you define here using the expression editor, is fulfilled.

Screen Areas

Screen Area: Assigned Method

The following information is displayed:

Field	Description
Method Name	Indicates the name of the assigned method and the assembly from which the method has been loaded. If the method has been configured manually, the information *Direct Destination System Calls* is displayed.
Processing Type	Indicates how the method notification is processed: • **Direct destination system call** The destination system linked with the method notification is called. • **Enhanced method processing without destination system call** Processing of the method call takes place entirely in the implementation of the enhanced method processing (EMP). • **Enhanced method processing with destination system call** The method call is processed by the EMP implementation that calls the configured destination system at the appropriate place.

By clicking on the arrow after the method name, you can navigate directly to the definition of the method in the OPC UA server.

Screen Area: Trigger

1. For method notifications, check that the *Enabled* indicator has been set if you want the notification to be executed. This is the default setting.
2. Define the trigger conditions that you want to be checked when an event occurs.
   For method notifications, the call of the assigned method is the event that triggers the check of the trigger conditions. Here you can use the input parameters of the method to formulate the condition expression. If the check of the trigger condition results in the value *true*, processing of the EMP implementation or the destination system call is triggered as a follow-up action, depending on the method type.

   > **i Note**
   >
   > Value changes in the input parameters for consecutive method calls are not relevant in the evaluation of the trigger conditions. Each method call is evaluated irrespective of earlier calls.

   For more information, see Trigger Types [page 508].

More Information

PCo as OPC UA Server and as Web Server [page 54]

Methods [page 11]

Expression Editor [page 545] and Formulating and Calculating Expressions [page 547]

4.4.2.2 Assignment of Input Parameters Tab

Use

You use the *Assignment of Input Parameters* tab to define the information that is to be passed on to a destination system using the notification.

> **i** Note
>
> The *Assignment of Input Parameters* tab is only offered for methods with direct system call and for EMP methods that provide for a destination system call. For EMP without a destination system call, only the *Notification* tab is offered.

The output values of the notification are based on different data sources, depending on the notification type:

- For method notifications for **methods with direct destination system call**, the input parameters of the method are used for generating the output expressions.
- For method notifications for an **EMP method with destination system call**, the output expressions are derived from the input variables that the EMP implementation provides for the destination system call.

  > **i** Note
  >
  > For EMP methods, the input parameters of the method are adopted directly from the EMP implementation; PCo does not provide for any configuration option here.

Prerequisites

- You have configured an OPC UA server or a Web server on the agent instance and loaded a method definition from an EMP assembly into the PCo configuration.
- You have created a method notification.

Procedure

1. Select the method notification and click on the *Assignment of Input Parameters* tab.
   The table with the output expressions is still empty.
2. In the simplest scenario, where there is a 1:1 relationship between an EMP input variable and the expression being output, choose the *Generate Expressions* icon. This function generates all expressions for the EMP input variables automatically. These expressions are then displayed in the *Output Expressions* table.

> **i Note**
>
> By clicking on the header row of the table of output expressions, you can sort the table by the content of the selected column.

For more information about the icons, see Output Tab [page 512].

More Information

PCo as OPC UA Server and as Web Server [page 54]

4.4.2.3 Destination Tab

Use

This tab is only displayed for method notifications that can execute a **destination system call**. In this case, you configure **exactly one destination system** that you want to be called when the method call is being processed. In contrast to the notification process, the return values of destination systems are supported here provided these systems deliver return values. You can write back the output variables of the destination system call to the EMP implementation or directly to the method output parameters. (See also: Assignment of Output Parameters/Variables Tab [page 542].)

> **i Note**
>
> If the method belongs to an EMP implementation that does not provide for a destination system call, the *Destination* tab is hidden.

More Information

Destinations Tab [page 520]

4.4.2.4 Assignment of Output Parameters/Variables Tab

Use

This tab is only displayed for method notifications that can execute a **destination system call**.

> **i Note**
>
> The tab displays a different title, depending on the type of method notification:
>
> - If it is a method notification for a **method with direct destination system call**, the term **parameter** is used because the parameters of the method flow into the expressions.
> - For **methods for an EMP implementation**, the term **variable** is used. It is the input and output variables of the EMP implementation that flow into the expressions here or to which the return values of the destination system call need to be assigned.

Features

Depending on the type of method execution, you can assign the output variables of the destination system call for further processing here:

- In the case of a method of type **direct destination system call**, you can assign the output variables of the destination system call directly to the output parameters of the method here. If the method does not provide for any output parameters, the table with the assignment options remains empty.
- If the method originates from an **EMP implementation**, here you can assign the output variables of the destination system call to the output variables that the EMP implementation provides. The result of the destination system call can then be processed further within the EMP implementation.

  > **i Note**
  >
  > The output parameters of the EMP method itself are supplied directly from the implementation. A configuration option is not envisaged for this.

4.4.3 Functions for Notifications

Prerequisites

You can only add, delete, rename, and copy or duplicate a notification for an agent instance if the corresponding agent instance is not running.

Procedure

Add Notification

1. On the *Plant Connectivity Management Console* screen, select an agent instance and click on the *Add Notification* icon.
   The *Add Notification* dialog box appears.

2. Select one of the following options:
   - Static notification
   - Notification template for remote subscriptions

     > **i Note**
     >
     > If you select this option, the system does not create a real notification. Instead the system creates a notification template that is used later to create a real notification in the Business Suite system, for example.

   - Versioned notification
   - Method notification
3. Choose the *Next Step* pushbutton.
   The dialog box for notification maintenance appears.
4. Enter the name and the description for the notification.
5. Click *OK*.
   The new notification is displayed below the corresponding agent instance.
6. If you want to create a **versioned notification**, you need to select the *Versioned Notification* option. For more information about how to create versioned notifications, see Adding a Draft Version [page 501].

Duplicate Notification

1. On the *Plant Connectivity Management Console* screen, select a notification and click on the *Duplicate* icon.
2. The duplicated notification is displayed below the original notification.

Add a Notification Using Drag&Drop

You can create a notification for any destination system by clicking on the *destination system* and dragging it to the *agent instance* of your choice. PCo then automatically creates a notification with the standard settings. The selected destination system is already entered on the *Destinations* tab. The prerequisite for this is that the agent used supports the notification process.

Copy and Paste a Notification

To copy notifications, you can use the Windows functions *Ctrl+C* and *Ctrl+V*, or proceed as follows:

1. On the *Plant Connectivity Management Console* screen, select a notification and choose *Edit* > *Copy*.
2. Then choose the *Paste* function.
3. The copied notification is displayed below the original notification.

> **i Note**
>
> You can also insert a copied notification below another agent instance, either using the *Edit* menu or *Ctrl+C*.

Delete Notification

1. On the *Plant Connectivity Management Console* screen, select a notification and choose the *Delete Notification* icon.
2. Click *Yes* to confirm the deletion.

Rename Notification

1. On the *Plant Connectivity Management Console* screen, select a notification and choose *Edit* > *Rename*.

2. Enter the new name and click *OK*.
3. The renamed notification is displayed in the list.

Change the Description of a Notification

After creating a notification, you can use this function to change its description at any time by clicking on the notification and choosing *Edit* > *Change Description* in the menu. You can only display the description of a notification by placing the mouse pointer on this object. PCo then displays the description that you entered when you created the agent instance.

Context Menu

You can call a context menu for each notification by clicking on the notification with the right mouse button. The following functions are available in the context menu:

- Add agent instance
- Import / export agent instance
- Duplicate notification
- Delete notification
- Rename notification
- Change notification description

4.4.4 Expression Editor

Definition

The expression editor is used in multiple places in the Management Console. You can use it to formulate expressions and conditions that are based on the variables or subscription items that are available in the respective context. The expression editor supports you with the correct syntactical formulation of the expression or the condition and provides a large number of functions with which you can perform mathematical and logical calculations, date calculations, and string operations. The expression editor is used for the following purposes:

- Defining conditions
  With the help of the expression editor you formulate a condition that is either false or true at runtime. This use case occurs, for example, when you define the trigger condition of a notification or for exception handling.
- Branching conditions
  You can use branching conditions to change the otherwise linear sequence of destination system calls when using the multiple call destination system. In this case, the result of a calculation is a step number or a jump target in a multiple call destination system.
- Defining output expressions and calculating variables
  For this use case, you use the available variables or subscription items to perform a calculation. The result of the calculation is then used, for example, as an output expression of a notification or is assigned to another variable in the multiple call destination system.

> **i Note**
>
> The expression editor uses American numerical formatting and is not localized.
>
> The functions in the expression editor are only provided with English names.

For internal calculations, the expression editor converts all numerical values into floating point numbers of the .NET data type `System.Double`. This applies to integer values of the data types `System.Byte`, `System.SByte`, `System.Int16`, `System.Int32`, `System.Int64`, `System.UInt16`, and `System.UInt32`, as well as to the floating point data type `System.Single`. This might lead internally to loss of accuracy for calculations and conversions between data types. Since the data type `System.Double` is used internally for all numerical values, the PCo expression editor cannot completely cover the value range of the data type `System.Int64`.

Use

You can call the expression editor from the following objects:

- **Notification Tab**
  On the *Notification* tab, you can use the expression editor to define the trigger conditions for triggering a notification message. You need to create a Boolean expression that can be evaluated as true or false. If you use the default setting *Trigger Type Always*, a conditional expression is not necessary. See also: Notification Tab [page 506]
- **Output Tab**
  On the *Output* tab, you can use the expression editor to specify which subscribed tag is to be output and, if required, calculate the values that are to be output using the expression editor functions, for example, from subscribed tags or constants. See also: Output Tab [page 512] and Examples of Message Texts [page 578]
- **Destination System Calls Tab for the Multiple Call Destination System**
  Using branching conditions, you can change the otherwise linear sequence of destination system calls that are configured in the form of steps by calculating the next step number to be executed or the next jump target using calculation expressions (for example, conditions or fixed values) so that another sequence is used. (See also: Multiple Call Destination System [page 353] and Branching Conditions [page 363]
  On this tab, you can also define calculations using the predefined function `@Calculate`.
- **Variables and Calculations Tab for the Multiple Call Destination System**
  The existing variables of the multiple call destination system or functions of the expression editor can be linked for calculating additional variable values.
- **Exception Handling for the Multiple Call Destination System and for Enhanced Notification Processing**
  You can use the expression editor to define the condition for handling an exception. The variables of the multiple call destination system or of enhanced notification processing are available for the formulation. In addition, you can define within exception handling which values are to be assigned to the variables depending on the exception condition. (See also: Exception Handling [page 79].)
- **Definition of Calculated Variables of a Web Service Destination System**
  You can use the output variables of a Web service destination system call to formulate expressions that you assign to new calculated variables. (See also: Calculated Variables in a Web Service Response Message [page 402].)

Structure

Depending on the use case, the expression editor is called with a list of variables or a list of subscription items that you can use to formulate the expression. For reasons of simplicity, we talk about variables below.

The expression editor consists of the following parts:

- *Expression* input field
  You enter the expression in this field. In the simplest scenario, there is a 1:1 relationship between the variable and the expression.
  When you enter an expression, the syntax of the expression is checked. An error icon is displayed if there are syntax errors in the expression. To view the error messages, either display the quick info for the error icon or click the exclamation mark pushbutton to display the error list.
  The elements of an expression that you have entered completely and correctly are identified by syntax highlighting in different colors. (See also: Syntax Highlighting in the Expression Editor [page 554].)
- List of *subscription items* or *variables*
  In this list, depending on the context, you find all subscription items or variables that you can use in an expression.
  You can insert an element at the current cursor position in the *Expression* input field either by double-clicking the element in the list or selecting the element and choosing *Insert*.
- List of jump targets
  If you are using the expression editor in the context of a multiple call destination system, you will find the *jump targets* that you defined in the multiple call destination system in this list. You can, for example, define a calculation expression that results in a jump target.
- *Functions* list
  This list contains all the functions that you can use in the expression editor. You add a function at the current cursor position in the *Expression* input field either by double-clicking the function or selecting the function and choosing *Insert*. (See also: Functions in the Expression Editor [page 555].)

More Information

Formulating and Calculating Expressions [page 547]

4.4.4.1 Formulating and Calculating Expressions

The document describes how you can use the expression editor to formulate and calculate expressions.

Using the Expression Editor

When the expression editor is started, the system automatically fills the list of subscription items or the variables (depending on the context). In general, you can form calculation expressions using the following elements:

- Subscription items or variables

  > Example
  >
  > `'double1'`, `'inVar1'`, or `'inVar1.updateCount'`

- Constants

  > **Example**
  >
  > `"SfcBO: ","City: "` or `3.14`

- Functions

  > **Example**
  >
  > `sin('double1'),"SfcBO: " & 'inVar1'`, or `'double1' > 0`

You add **variables or subscription items** at the current cursor position in the expression, either by double-clicking on the relevant element in the list or selecting the element and then choosing *Insert*. Alternatively, you can type in the name of the variable or subscription item in the *Expression* field. Note that the names need to be **enclosed with single quotation marks** and are case-sensitive.

From the list of functions, you add **functions** at the current cursor position in the expression in the same way as you add variables or subscription items. Then you replace the placeholders inserted by the system for function parameters with real variable names or subscription item names, constants, or functions.

Moreover, you can **search for functions** by choosing *Search* in the expression editor and entering a search term in the *Function Search* dialog box. The system searches for your search term in the function names and function descriptions. This search is not case-sensitive. When you have found the function you want, click on it and choose *Select*. The chosen function is selected in the list of functions of the expression editor and you can copy it from there into the expression.

Since the expression editor is used in various contexts, three typical use cases are described below.

Calculating an Output Expression from Subscription Items

You can use the expression editor to formulate output expressions in notifications. Depending on the use case, you can use subscription items or input parameters of methods, constants, and functions in the output expressions.

> **Example**
>
> You want to copy the subscription item `'temperatureBoiler'` unchanged into an output expression. The expression is as follows:
>
> `'temperatureBoiler'`

> **Example**
>
> You want to create an expression in which the subscription item `'temperatureBoiler'`, which renders a temperature value in degrees Fahrenheit, is converted into degrees Celsius. The expression is as follows:
>
> ('temperatureBoiler'-32)*5/9

Example

You want to generate an output expression in which an SFC from the subscription item `'sfc'` and the plant number from the subscription item `'plant'` are concatenated. The expression is as follows:

```
"SFCBO:" & 'sfc' & "," & 'plant' & ",#"
```

Example

You want to generate an output expression that always contains a constant value, for example, the plant number 1000:

```
1000
```

Formulating a Boolean Expression

Boolean expressions are used, for example, in trigger conditions of notifications or in exception handling in multiple call destination systems and enhanced notification processing (ENP). The result always has the data type Boolean with the possible values `true` or `false`.

Example

You want to formulate a trigger condition that is true if the value of the subscription item `'temperatureBoiler'` is greater than 100:

```
'temperatureBoiler' > 100
```

Example

You want to formulate a condition for exception handling that is true if the variable `!EXCEPTION_MESSAGE!` contains the character string Error 123:

```
stringindexof('!EXCEPTION_MESSAGE!', "Error 123") > 0
```

Formulating a Branching Condition in a Multiple Call Destination System

Branching conditions in a multiple call destination system are calculation expressions whose result is to be a numeric integer value that is greater than 0. The result is the number of the next step that is to be executed.

Validating an Expression

You can choose the *Validate* pushbutton to check the expression that has been entered. PCo issues an error message if you have not entered the expression correctly.

Debugging an Expression

You can choose the *Debug* pushbutton to check which values or texts PCo issues after a tag value is entered:

1. In the *Expression Editor* dialog box, choose the *Debug* pushbutton.
   The *Expression Debug* dialog box appears.
2. Enter values for the variables and choose the *Evaluate Expression* pushbutton to evaluate the expression and populate the *Result* screen area.

   > **i** Note
   >
   > To enter values for variables of array data types, use the JSON notation for arrays.
   >
   > **Example**: To enter an array of data type `System.Double`, write `[42,17.5,-3.14]`.
   >
   > JSON notation is also used for the input and output of variable values for structured data types. You also need to name the data types of the structures in the JSON character string. To do this, define a JSON type property character string. The expression debug dialog box proposes `__type` as a character string.
   >
   > **Example**: You want to enter a value for a variable of the structured data type `PqmMeasuresStruct` . In the JSON character string, enter the fully qualified names of all used structured data types using the property `__type`: `{"__type": {"namespace":"","name":"PqmMeasuresStruct"},"Measures":[{"__type": {"namespace":"","name":"PqmMeasureStruct"},"tenantId":"myTenant","equipmentId":"00b26c44-542a-45bf-9145-46321e1e6d37"}]}`
3. The calculated value is displayed in the *Result* screen area. Values from array data types or structured data types are displayed in JSON notation.
4. Choose *Close*.

Data Types

Choose the *Data Types* pushbutton to open the data type selection dialog box. There you can select a data type from the PCo data type repository and insert it in the expression.

More Information

Functions in the Expression Editor [page 555]

Operators [page 572]

Examples of Message Texts [page 578]

4.4.4.2 String Literals in Expressions

Use of String Literals

You can use string literals to define **fixed** strings in expressions. String literals contain double quotation marks at the start and end of the string.

> ✤ Example
>
> `"This is a string"`

String literals are created within the **expression editor** with the data type `System.String`.

The use of special characters in string literals requires special attention in the expression editor. Here are the characters:

Special Characters

Character	Description
"	Double quotation marks
\	Backslash
cr	Carriage return
lf	Line feed
crlf	Carriage return and line feed
tab	Horizontal tab (tab key)

The system setting *Allow Verbatim String and Escape Sequences* determines how these characters in string literals are interpreted by the expression editor and how the user needs to enter them. You can find this setting in the menu under *Tools* > *Options* > *Global Settings* > *Compatibility*. (See also: Compatibility Settings [page 17].)

> ℹ Note
>
> The *Allow Verbatim String and Escape Sequences* checkbox has been introduced with PCo 15.5 (SP03). Expressions that you created in earlier PCo versions are interpreted as if the checkbox has **not** been selected.

Allow Verbatim String and Escape Sequences

If you have selected the checkbox, the following rules apply in the string literals:

- Escape sequence
  A combination of a **backslash ** and a letter or a number is interpreted as an escape sequence. You can use an escape sequence to define characters that cannot be printed, for example: Carriage return (cr), tab,

Unicode characters, or characters reserved in string literals, such as the double quotation mark or the backslash. The expression editor allows the following escape sequences:

Escape Sequences

Escape Sequence	Resulting Character
`\'`	Single quotation mark '
`\"`	Double quotation mark "
`\\`	Backslash \
`\a`	Acoustic signal
`\b`	Backspace
`\f`	Form feed
`\n`	Line feed `lf`
`\r`	Carriage return `cr`
`\t`	Horizontal tab `tab`
`\v`	Vertical tab
`\uxxxx`	Unicode character with hex value xxxx, for example, `\u00AD`
`\0 - \9`	Unicode characters 0 to 9

- Verbatim string @
  If you prefix a string literal with the **verbatim string @**, all characters within the double quotation marks are interpreted literally. This means that backslashes \ are **not interpreted as escape sequences** either. Double quotation marks are an exception: If you want your string to contain double quotation marks, the existing double quotation marks have to be enclosed in additional double quotation marks.

Allow Verbatim String and Escape Sequences Checkbox Not Selected

If you have **not selected the checkbox**, as in the previous default setting, the following rules apply in string literals:

- Backslashes \ are not interpreted as escape sequences, but are interpreted literally.
- To be able to insert the following characters in character strings, you must use special functions of the expression editor and link them to the other parts of the string using the operator **&**:

  Special Functions in the Expression Editor

Character	PCo Function
"	`doublequote`
`cr`	`cr`
`lf`	`lf`
`crlf`	`crlf`

Character	PCo Function
`tab`	`tab`

> **i Note**
>
> Note that you can continue to use the PCo functions mentioned in the table if you have selected the *Allow Verbatim String and Escape Sequences* checkbox. However, in this case, these PCo functions make an expression unnecessarily complicated because there are simpler alternatives in the form of escape sequences and the verbatim string.

Related Information

Examples of Strings [page 553]
Functions in the Expression Editor [page 555]

4.4.4.2.1 Examples of Strings

The following table uses examples to show how you can implement strings in the expression editor depending on whether or not the *Allow Verbatim String and Escape Sequences* checkbox is selected:

Examples: Checkbox Selected and Not Selected

Desired String	Checkbox Selected: Use of Escape Sequences \	Checkbox Selected: Use of the Verbatim String @	Checkbox Not Selected
Hello	`"Hello"`	`@"Hello"`	`"Hello"`
C:\Temp\File.txt	`"C:\\Temp\ \File.txt"`	`@"C:\Temp\File.txt"`	`"C:\Temp\File.txt"`
\\Server\File.txt	`"\\\\Server\ \File.txt"`	`@"\\Server \File.txt"`	`"\\Server\File.txt"`

Desired String	Checkbox Selected: Use of Escape Sequences \	Checkbox Selected: Use of the Verbatim String @	Checkbox Not Selected
`{"a":"1","b":"2"}`	`"{\"a\":\"1\",\"b\":\"2\"}"`	`@"{""a"":""1"",""b"":""2""}"`	`"{" & doublequote & "a" & doublequote & ":" & doublequote & "1" & doublequote & "," & doublequote & "b" & doublequote & ":" & doublequote & "2" & doublequote & "}"`
Line1 `lf` Line2	`"Line1\nLine2"`	@"Line1 Line2"	`"Line1" & lf & "Line2"`
abc `tab` def	`"abc\tdef"`	`"abctabdef"`	`"abc" & tab & "def"`
`abc\tdef`	`"abc\\tdef"`	`@"abc\tdef"`	`"abc\tdef"`

> **i Note**
>
> As you can see from the examples, if the *Allow Verbatim String and Escape Sequences* checkbox is selected, backslashes in string literals are interpreted differently.
>
> If you migrate agent instance and destination system configurations from PCo version 15.5.2 or older and want to select the checkbox, you may have to manually adjust your expressions to the notation with escape sequences or the verbatim string.

4.4.4.3 Syntax Highlighting in the Expression Editor

In the *Expression* field, the elements of an expression are highlighted in color. Different colors are used to distinguish between the different elements in the expression as follows:

- Blue: Variables or subscription items
- Dark green: Functions
- Dark red: String literals and number literals
- Orange: Parentheses
- Purple: Variables of a multiple call destination system

Syntax highlighting only works correctly if the expression does not contain any syntax errors. This is why no colors (or the wrong colors) are displayed until you have finished entering an expression. Syntax highlighting cannot highlight errors in an expression.

> **i Note**
>
> Syntax highlighting is set by default, but you can deactivate it in the menu under *Tools* > *Options* > *User Settings* > *Expression Editor* by deselecting the *Highlight Syntax of Expressions* checkbox. (See also: Settings for the Expression Editor [page 22].)

4.4.4.4 Functions in the Expression Editor

You can choose the following functions in the expression editor:

Function	Description
`abs(value)`	Returns the absolute value of the input. This function ensures that only the absolute value of the input is returned in the output (for example, in an e-mail). In other words, the absolute value of the subscribed tag or the variable is output if you have not entered any additional calculation in the expression. This is the default setting.
`acos(value)`	Returns the inverse cosine of the input; result is an angle between 0 and pi.
`append(list, value)`	Adds a value `value` to a list `list` and returns the new list. The value `value` must have an elementary data type.
`arrayAppendValue(array, value)`	Adds a value to an array.

Function	Description
`arrayCreate(elementDataTypeName, length)`	Creates an array of type elementary data type with an initial length. `elementDataTypeName` denotes the element data type. Only the following element data types are allowed: • System.Boolean • System.Byte • System.Char • System.DateTime • System.Double • System.Guid • System.Int16 • System.Int32 • System.Int64 • System.SByte • System.Single • System.String • System.UInt16 • System.UInt32 If the length is greater than 0, the array is prefilled with elements that have the initial values of the relevant element data type.
`arrayCreateFromValues(arrayDataType, [value1,value2,..])`	Creates an array from a list of values. `arrayDataType` is the data type of the returned array. The length of the array is determined by the number of values. The values are copied to the array. The values must be convertible to the element data type of the array data type. The following array data types are allowed: • System.Boolean[] • System.Byte[] • System.Char[] • System.DateTime[] • System.Double[] • • System.Int16[]System.Guid[] • System.Int32[] • System.Guid[System.Int64[] • System.SByte[] • System.Single[] • System.String[] • System.UInt16[] • System.UInt32[]

Function	Description
`arrayGetValue(array, index)`	Returns the value of an array element. `index` is a numeric value between 0 and the array length - 1.
`arrayLength(array)`	Returns the length of an array.
`arraySetValue(array, value, index)`	Sets the value of an array element. `index` is a numeric value between 0 and the array length - 1.
`asciitohex`	Converts an ASCII character string into a hexadecimal character string. **Example** If you enter `asciitohex("PCo is great")` in the expression editor, the result `50436f206973206772656174` is returned.
`asciitostring(asciiCode)`	Returns the character (Unicode UTF-16) that is represented by a numeric ASCII code. The result is a string that contains this character.
`asin(value)`	Returns the inverse sine of the input; result is an angle between 0 and pi.
`atan(value)`	Returns the inverse tangent of the input; result is an angle between 0 and pi.
`average(x1, ...)`	Returns the average of multiple values
`base64decode(encodeStringValue)`	Decodes the input string using the decoding algorithm `Base64` based on the UTF-8 format.
`base64encode(stringValue)`	Encodes the input string using the encoding algorithm `Base64` based on the UTF-8 format.
`booleantostring(booleanValue)`	Converts the Boolean string into a value based on the UTF-8 format.
`ceiling(value)`	Rounds the input value up to the next largest integer
`containskey(map, key)`	Checks if a key is in the map; if the key exists in the map, it returns `true`.
`containsvalue(map, value)`	Checks if a value is in the map; if the value exists in the map, it returns `true`.

Function	Description
`ConvertTimeZone`	Converts a UTC date into the specified time zone. The time zone is determined using the time zone ID, for example, Central Pacific Standard Time. The values for the time zone IDs are stored in the registry under `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Time zone`.
`convertunits(value, from, to)`	Converts the input value from the `From` unit of measure to the `To` unit of measure. Note that the values of `From` and `To` are case-sensitive. The values must correspond to the values in the conversion table. This function makes evaluations at runtime only.
`cos(value)`	Returns the cosine of the input value
`cosh(value)`	Returns the hyperbolic cosine of the input value
`cr`	Returns the carriage return character. You can insert a carriage return in a string literal without this function if you have selected the *Allow Verbatim String and Escape Sequences* checkbox in the PCo system settings. **Example** `"a\rb"` See also: String Literals in Expressions [page 551].
`crlf`	Returns the character combination carriage return and line feed. You can insert the carriage return and line feed in a string literal without using this function if you have selected the *Allow Verbatim String and Escape Sequences* checkbox in the PCo system settings. **Example** `"a\r\nb"` See also: String Literals in Expressions [page 551].

Function	Description
`csvParse(csvString, recordSeparator, fieldSeparator, quoteCharacter, containsHeader)`	Parses the comma-separated CSV character string `csvString` into an array of structured objects. The result is an untyped array of untyped structured objects, where each structured object represents a record, and each component of a structured object represents a field. "Untyped" means that the resulting array is of data type "`System.Object`", has the element data type "`System.Object`", and the structured objects are of data type "`System.Object`". `recordSeparator` denotes the character that delimits the records. `fieldSeparator` denotes the character that delimits the fields of a record. Each record must have the same number of fields. Field values are always character strings, even if they represent numbers, dates, times, or Boolean values. `quoteCharacter` denotes a character string that is used to enclose fields. Typically, these are double quotation marks. If no character string is to be used to enclose fields, `quoteCharacter` must be set to an empty string "". Escape characters in fields are not supported. The character for `recordSeparator` must not occur in field values that are enclosed by `quoteCharacter`. Likewise, `quoteCharacter` inside a field enclosed by the same `quoteCharacter` is not allowed. If `containsHeader` is true, the components of the structured objects are named after the fields of the first record. If `containsHeader` is false, the components are named sequentially as "0", "1", "2", and so on. See also: Application Examples for csvParse(csvString, recordSeparator, fieldSeparator, quoteCharacter, containsHeader) [page 574].
`dateadddays(startDate, addDays)`	Adds days to the date value
`dateaddhours(startDate, addHours)`	Adds hours to the date value
`dateaddminutes(startDate, addMinutes)`	Adds minutes to the date value
`dateaddmonths(startDate, addMonths)`	Adds months to the date value
`dateaddseconds(startDate, addSeconds)`	Adds seconds to the date value
`dateaddyears(startDate, addYears)`	Adds years to the date value

Function	Description
`datebetween(date, startBoundaryDate, endBoundaryDate)`	Determines whether the value is between the `startBoundarydate` and `endBoundarydate` values; if the value is on or after the `startBoundarydate` value and prior to or on the `endBoundarydate` value, it returns the value `True`.
`datecompare(date1, date2)`	Returns an integer that identifies which date is later. If `date1` occurs after `date2`, the function returns `1`; if `date1` and `date2` are identical, it returns `0`; if `date1` occurs before `date2`, it returns `-1`.
`datediff(date1, date2)`	Returns the number of milliseconds between two date values
`datediffdays(date1, date2)`	Returns the number of days between two date values
`datediffhours(date1, date2)`	Returns the number of hours between two date values
`datediffminutes(date1, date2)`	Returns the number of minutes between two date values
`dateformat(dateValue, fromFormat, toFormat)`	Converts the value to a different format
`datefrommilliseconds(milliseconds)`	Converts the milliseconds that have elapsed since January 1, 0001 00:00:00 UTC into a UTC date value.
`datefromseconds(date)`	Gets the date from the value that is the number of seconds since January 1 0001 00:00:00 UTC
`datefromsecondsformatted(seconds, format)`	Gets the date from the value that is the number of seconds since January 1 0001 00:00:00 UTC, and converts it to a string of `format`.
`datefromxmlformat(date, toFormat)`	Converts the XML value to the specified format
`datenow`	Returns the current system date and time. The precision of the date and time values depends on the system clock and the Windows operating system.
`dateParse(dateString, fromFormat)`	Parses the string `dateString` and converts it into a UTC date object. `fromFormat` is a Microsoft .NET date and time format string, for example, "MMMM dd, yyyy" or "o". For more information, see the documentation for the related C# method DateTime.TryParseExact() under https://docs.microsoft.com/en-us/dotnet/api/system.datetime.tryparseexact?view=netframework-4.8 .

Function	Description
`datetolongstring(date)`	Converts the specified date value into a `character` string that contains the date and the time with a precision of up to a ten millionth of a second.
`datetomilliseconds(date)`	Converts the specified `date` value into milliseconds that have elapsed since January 1 0001 00:00:00 UTC. The result is expressed in whole and fractional milliseconds. The precision depends on the resolution of the system clock and the Windows operating system.
`datetoseconds(date)`	Converts the specified `date` value into seconds
`datetoticks (date)`	Converts the specified `date` value into ticks. Ticks represent the number of 100-nanosecond intervals that have elapsed since January 1 0001 00:00:00 UTC.
`datetoxmlformat(dateString, fromFormat)`	Converts the XML value to the specified format
`day(date)`	Returns the day of the month from the `date` value
`dayofweek(date)`	Returns the day of the week from the `date` value; if the function returns `1`, the day is Sunday; `7` is Saturday.
`dayofyear(date)`	Returns the day of the year (1-365) from the `date` value
`daysinmonth(date)`	Returns the number of days in the month from the `date` value
`decode(encodedStringValue)`	Decodes the input string using the decoding algorithm Base64 based on the UTF-32 format.
`doublequote`	Returns the character " (double quotation marks). You can insert double quotation marks in a string literal without using this function if you have selected the *Allow Verbatim String and Escape Sequences* checkbox. **Example** `@"a""b"""` or `"a\"b\""` See also: String Literals in Expressions [page 551].
`e`	Mathematical constant e = 2.7182818284590451
`emptylist`	Returns an empty list
`emptymap`	Returns an empty map

Function	Description
`encode(stringValue)`	Encodes the input string using the Base64 encoding algorithm based on the UTF-32 format
`exp(value)`	Returns a string specifying `e` (the base of natural logarithms) raised to a power This function complements the action of the log function and is referred to as the antilogarithm.
`false`	Boolean value for 'false'
`find(list, value)`	Returns the index of the first occurrence of the value, or if the value is not found, it returns `-1`
`first(list)`	Returns the first element in the list
`floor(value)`	Rounds the value up to the next integer
`format(value, format)`	Returns the value in the specified format
`fromJson(jsonString, [targetDataTypeNamespace,] targetDataTypeName, typeInfoPropertyName)`	Creates an instance of a structured, elementary, or array data type from the JSON-formatted character string `jsonString`. The target data type must exist in the PCo data type repository and is specified exactly by its namespace `targetDataTypeNamespace` and its name `targetDataTypeName`. If the PCo data type is a built-in data type, `targetDataTypeNamespace` is an empty string or can be omitted. If the PCo data type is a structured data type, or contains components of structured data types, the argument `typeInfoPropertyName` must specify the property name that defines the type information for structured data types, for example, `"__type"` or `"$type"`. If no structured data types are involved, `typeInfoPropertyName` can be an empty character string. (See also: Processing Character Strings in JSON Format [page 588].)
`fromUnixMilliseconds(milliseconds)`	Converts the milliseconds that have elapsed since January 1, 1970 00:00:00 UTC into a UTC date value.
`getByteArrayFromStringUtf8(string)`	Converts a UTF-8 encoded string to a byte array.

Function	Description
`getChangeDateTime (value)`	Returns the change date and the change time of the subscription item `value`. Note that the function only returns meaningful data in the context of notification expressions. Only the following source system types return the information about the change date and the change time of the subscription item: • Modbus source system • OPC DA source system • OPC HDA source system • OPC UA source system • Asset Framework source system • OSISoft PI source system • Proficy Historian source system
`getconversionfactor(fromUnit, toUnit)`	Returns the value that is used to convert a unit of measure into another unit of measure. Note that the values of `fromUnit` and `toUnit` are case-sensitive. The values must correspond to the values in the conversion table. This function makes evaluations at runtime only.
`getOperationFromOperationRef(operationRef)`	Returns the operation from an operation reference of SAP ME.
`getOperationRef(site, operation, revision)`	Returns a string that represents an operation reference in SAP ME.
`getQuality(value)`	Returns the quality of the subscription item `value`. Note that the function only returns meaningful data in the context of notification expressions. Only the following source system types return information about the quality of the subscription item: • Modbus source system • OPC DA source system • OPC HDA source system • OPC UA source system • Asset Framework source system • OSISoft PI source system • Proficy Historian source system
`getResourceFromResourceRef(resourceRef)`	Returns the resource from a resource reference of SAP ME.

Function	Description
`getResourceRef(site, reosurce)`	Returns a string that represents a resource reference in SAP ME.
`getRevisionFromOperationRef(operationRef)`	Returns the version from an operation reference of SAP ME.
`getSfcFromSfcRef(sfcRef)`	Returns the SFC from the SFC reference of SAP ME.
`getSfcRef(site, sfc)`	Returns a string that represents an SFC reference in SAP ME.
`getSiteFromOperationRef(operationRef)`	Returns the production site from an operation reference of SAP ME.
`getSiteFromResourceRef(resourceRef)`	Returns the production site from a resource reference of SAP ME.
`getSiteFromSfcRef(sfcRef)`	Returns the production site from an SFC reference of SAP ME.
`getStringFromByteArrayUtf8(byteArray)`	Converts a byte array into a UTF-8 encoded string.
`guid`	Globally unique identifier
`hextoascii`	This is the inverse of `asciitohex`.
`hour(date)`	Returns the hour of the specified `date` value
`insert(list, location, value)`	Returns a new list with the input list parameter updated so that the value can be inserted in the specified location
`isNull(value)`	Returns the value `true` if `value` is null.
`json2struct(jsonString)`	Creates structured, elementary, or array data objects from the JSON-formatted character string `jsonString`. The data type of the data object is System.Object. Use function `fromJson` if you want the data objects to have defined data types.
`keys(map)`	Returns a list of all key values in the map
`last(list)`	Returns the last element in the list

Function	Description
`lf`	Returns the line feed character. You can insert line feeds in a string literal without using this function if you have selected the *Allow Verbatim String and Escape Sequences* checkbox in the PCo system settings. **Example** `"a\nb"` See also: String Literals in Expressions [page 551].
`ln(value)`	Returns the natural logarithm of the value
`localformat(value, format)`	Converts the value to a string using the specified format
`localnumber(value)`	Converts the specified value to a numerical value using the local server's regional settings
`log(value)`	Returns the Base10 logarithm of the specified value
`logn(value, base)`	Returns the logarithm base of the value
`match(stringValue, matchString)`	Indicates whether a match was found in the specified string This function uses the regular expression specified in the pattern and is not case-sensitive.
`match2(stringValue, matchString, isCaseSensitive)`	Indicates whether a match was found in the specified string The function uses the regular expression specified in the pattern and is case-sensitive.
`max(x1, ...)`	Returns the largest `x1` value

Function	Description
`maxValue(value1, [value2,value3,..])`	Compares values and returns the maximum value. The values can have the following data types: • System.Boolean • System.Byte • System.Char • System.DateTime • System.Double • System.Guid • System.Int16 • System.Int32 • System.Int64 • System.SByte • System.Single • System.String • System.UInt16 • System.UInt32 Array data types of these data types are also allowed.
`min(x1, ...)`	Returns the smallest `x1` value
`minValue(value1, [value2,value3,..])`	Compares values and returns the minimum value. The values can have the following data types: • System.Boolean • System.Byte • System.Char • System.DateTime • System.Double • System.Guid • System.Int16 • System.Int32 • System.Int64 • System.SByte • System.Single • System.String • System.UInt16 • System.UInt32 Array data types of these data types are also allowed.
`minute(date)`	Returns the minute value from the specified `date` value
`month(date)`	Returns the month from the specified `date` value

Function	Description
`monthend(date)`	Returns the date for the first day of the next month
`monthstart(date)`	Returns the date for the first day of the month
`not(value)`	Returns the logical negation of the value; if the value is `1`, the function returns `0`; if the value is `0`, it returns `1`.
`nulldate`	Date value used to represent the absence of a value
`nullnumber`	Numeric value used to represent the absence of a value
`nullstring`	String value used to represent the absence of a value
`number(stringValue)`	Converts the specified string to a numerical value
`pi`	Mathematical constant π = 3.1415926535897931
`put(map, name, value)`	Returns a new map with the input map parameter updated with the value set to the specific key
`random(lowValue, highValue)`	Returns a random number in the specified range; you must enter positive values
`randomflag(percentProbability)`	Returns a random value based on the specified percentage; values less than or equal to 0 return `False`, values greater than 100 return `True`
`remove(list, value)`	Returns a new list with the value parameter removed
`round(value)`	Rounds the input value to the nearest integer
`scaledmax(dataSetMin, dataSetMax)`	Returns a rounded value for the maximum value; `dataSetMin` is the minimum value and `dataSetMax` is the maximum value
`scaledmin(dataSetMin, dataSetMax)`	Returns a rounded value for the minimum value; `dataSetMin` is the minimum value and `dataSetMax` is the maximum value
`scaledvalue(value, currentMinimum, currentMaximum, newMinimum, newMaximum)`	Returns the value, which was originally scaled based on the `currentMinimum` and `currentMaximum` values, which are scaled based on the range from `newMinimum` to `newMaximum` You can use this function for animation.
`second(date)`	Returns the seconds from the specified `date` value

Function	Description
`set(list, index, value)`	Returns a new list with the input list parameter updated with the value set in the specified location
`sin(value)`	Returns the sine of the specified value
`singlequote`	Returns the character ' (apostrophe).
`sinh(value)`	Returns the hyperbolic sine of the value
`size(collection)`	Returns the size of the collection
`sort(list)`	Returns the list sorted
`sqrt(value)`	Returns the square root of the specified value; you must enter a value greater than or equal to **0**.
`stringindexof(stringValue, searchString)`	Finds the pattern in the string and returns its starting index
`stringleft(stringValue, newStringLength)`	Returns a substring of the specified string, which represents the number of characters from the left of the specified string
`stringlength(stringValue)`	Returns the length of the string
`stringlower(stringValue)`	Returns a lowercase version of the string
`stringpart(stringValue, startIndex, length)`	Returns a substring of specified length (`length`) starting with the value specified for `startIndex`. The `startIndex` starts with 1. **Example** `stringpart("123test-abc", 1, 3)` returns the string “123”.
`stringreplace(stringValue, searchString, replacement)`	Tests the string against the `searchString`; if the string matches, the function replaces the pattern with the `replacement` string.
`stringright(stringValue, length)`	Returns a substring of the `stringValue`, which represents the `length` from the right of the string

Function	Description
`stringsplit(stringValue, separator, substringToReturnIndex)`	Splits a string `stringValue` at each occurrence of the separation string `separator` into substrings and returns the substring with the number `substringToReturnIndex`. **Example** **Example 1** `stringsplit("ItemBO: 1000;YE_SMARTMETER;A", ";", 1)` separates the character string `"ItemBO: 1000;YE_SMARTMETER;A"` at each semicolon into the three substrings `"ItemBO:1000"`, `"YE_SMARTMETER"`, and `"A"` and returns the first substring. The result is `"ItemBO:1000"`. **Example 2** `stringsplit("AA;;BB;", ";", 4)` separates the character string `"AA;;BB;"` at each semicolon into the four substrings `"AA"`, `""`, `"BB"` , and `""`, and returns the fourth substring. The result is `""`. **Example 3** `stringsplit("ABCDEF", "CD", 2)` separates the character string `"ABCDEF"` at the point `"CD"` into the two substrings `AB` and `EF` and returns the second substring. The result is `"EF"`.
`stringsplittoarray(stringValue, separator)`	Splits a string 'stringValue' into substrings for each occurrence of the separation string 'separator' and returns a string array that contains these substrings.
`stringtoboolean(stringValue)`	Converts a string value to a Boolean value
`stringtrim(stringValue)`	Removes gaps from both sides of the string
`stringupper(stringValue)`	Returns the string in capital letters
`struct2json(object)`	Creates a JSON-formatted character string with indentation from an `object` instance of a structured, elementary, or array data type. The JSON-formatted character string doesn't contain any data type information. If you want to create a JSON character string that contains information about the data types of structured objects or if you want to control the indentation, you can use the function `toJson`.

Function	Description
`structArrayAppend(value,array)`	Adds a value to an array.
`structArrayGetValue( array, index)`	Returns the value of an array element. `index` is a numeric value between 0 and the array length -1.
`structArrayLength(array)`	Returns the length of an array.
`structArrayNew([[arrayDataTypeNamespace], arrayDataTypeName])`	Creates an array. `arrayDataTypeNamespace` and `arrayDataTypeName` specify the array data type. If no argument is used, the function creates an array object of element data type `System.Object`.
`structArraySetValue(array, value, index)`	Sets the value of an array element. `index` is a numeric value between 0 and the array length -1.
`structCompDelete(obj, [component 1, ..., component n-1], component)`	Deletes the component `component` under the component path `component 1` to `component n-1` of a structured data type instance `obj`. The components `component` and `component1` to `component n-1` can be single strings, string arrays, or string lists. If you use arrays and lists, the elements of the arrays and lists are added to the component path. The function returns a copy of `obj` without the deleted component. If the component path does not exist, a copy of `obj` is returned. All data type information is removed from the resulting object. You can use this function to forward only relevant components to a destination system. See also: Application Example for structCompDelete(obj, [component 1, ..., component n-1], component) [page 577].
`structCompExists(obj, [component 1, ..., component n-1], component)`	Checks if the component `component` of a structured data type instance `obj` is available under the component path `component 1` to `component n-1`. The result is a Boolean value `true` or `false`. `component` and `component1` to `component n-1` can be simple strings, string arrays, or string lists. If you use arrays and lists, the elements of the arrays and lists are added to the component path.
`structCompGet(obj, [component 1, ..., component n-1], component)`	Gets the component `component` of a structured data type instance `obj` under components `component1` to `component n-1`. `component` and `component1` to `component n-1` can be simple strings, string arrays, or string lists. If you use arrays and lists, the elements of the arrays and lists are added to the component path.

Function	Description
`structCompSet(value, obj, [component 1, ..., component n-1], component)`	Sets or updates the value of the component `component` under the component path `component 1` to `component n-1` of the structured data type instance `obj` to `value`. If the data type of `obj` is not specified and the component or component path do not exist, the component path and the component are created. `component` and `component1` to `component n-1` can be simple strings, string arrays, or string lists. If you use arrays and lists, the elements of the arrays and lists are added to the component path. The function returns a copy of `obj`.
`structNew([[structuredDataTypeNamespace], structuredDataTypeName])`	Creates an instance of a structured data type. If this is used with arguments, the function creates a structured object of data type `System.Object`.
`suspend`	Is only used within a multiple call destination system to check if a branching condition returns the result "suspend".
`tab`	Returns the horizontal tab character. You can insert horizontal tabs in a string literal without using this function if you have selected the *Allow Verbatim String and Escape Sequences* checkbox in the PCo system settings. **Example** `"a\tb"` See also: String Literals in Expressions [page 551].
`tan(value)`	Returns the tangent of the input value
`tanh(value)`	Returns the hyperbolic tangent of the input value
`tempfile(extension)`	Returns a unique file name with an extension
`today`	Current system date; time field is set to midnight
`toJson(object, indented, [typeInfoPropertyName])`	Creates a JSON-formatted character string from the `object` instance of a structured, elementary, or array data type. The Boolean value `indented` specifies whether the JSON character string is to be indented or not. The optional argument `typeInfoPropertyName` specifies the property name that defines the type information for the structured data types, such as `"__type"` or `"$type"`. If the argument `typeInfoPropertyName` is not used, the character string doesn't contain any type information. (See also: Processing Character Strings in JSON Format [page 588].)

Function	Description
`true`	Boolean value for the adjective is true
`toUnixMilliseconds(dateTime)`	Converts the date value `dateTime` into the number of milliseconds that have elapsed since January 1, 1970 00:00:00 UTC.
`values(map)`	Returns a list of all values in the map
`xmldecode(encodeStringValue)`	Decodes the input string as if it were an XML-coded string
`xmlencode(stringValue)`	Encodes the input string to comply with XML standards The function converts the following characters to their XML equivalents: • < to `<` • > to `>` • & to `&` • ' to `&apos;` • " to `&quot;`
`xmlencodename(value)`	Encodes the input string as XML code name
`year(date)`	Returns the year from the `date` value
`yearend(date)`	Returns the date of the first day of the first month of the year after the year specified in the `date` value
`yearstart(date)`	Returns the date of the first day of the first month of the year specified in the `date` value

4.4.4.5 Operators

You can use the following logical and mathematical operators in expressions:

Operator	Description
(	Left parenthesis
)	Right parenthesis
>	Greater than
<	Less than

>=	Greater than or equal to
<=	Less than or equal to
==	Equal to
!=	Not equal to
&&	Logical conjunction (And)
||	Logical disjunction (Or) This operator is used to enter an 'or' condition. **Example** You use the expression editor to enter the formula `'TAG_TEMP'<57 ||'TAG_TEMP'>63` as the trigger condition. If the value of the selected tag TAG_TEMP lies below 57°C (134.6°F) or above 63°C (145.4°F), PCo creates a notification message and sends it to the destination system.
+	Plus
–	Minus
*	Multiply
/	Divide
%	Modulo operation
^	Power
&	Concatenation You can use this operator to add a text to the expression. This text is then attached to the value of the tag in the notification message. **Example** You have specified the following expression in the editor: `'Double1' & " degrees Celsius"` Double1 is the name of the tag. The text that is to be output must be put in quotation marks. `44 degrees Celsius`, for example, is then output in the notification message.

4.4.4.6 Examples for Using Functions in the Expression Editor

This section contains examples of the functions that you can select in the expression editor:

Application Example for structCompDelete(obj, [component 1, ..., component n-1], component) [page 577]

Application Examples for csvParse(csvString, recordSeparator, fieldSeparator, quoteCharacter, containsHeader) [page 574]

Related Information

Functions in the Expression Editor [page 555]

4.4.4.6.1 Application Examples for csvParse(csvString, recordSeparator, fieldSeparator, quoteCharacter, containsHeader)

Example 1

You can use the function `csvParse(csvString, recordSeparator, fieldSeparator, quoteCharacter, containsHeader)` to parse the following CSV character string into an array of structured objects:

```
Time,Temperature,PressureCRLF
09/14/2021 07:55:56,76.5,1023.34CRLF
09/14/2021 08:55:42,73.0,1024.07CRLF
09/14/2021 09:55:58,71.0,1017.1CRLF
```

The following conditions should apply:

Function Parameters	Description	Character Used
`recordSeparator`	Record separator: Carriage return (CR) + line feed (LF)	CRLF

Function Parameters	Description	Character Used
`fieldSeparator`	Field separator	Comma
`quoteCharacter`	Quotation marks	No quotation marks
`containsHeader`	Contains header	Yes

Expression used:`csvParse('csvString',crlf,",","",true)`

Result:

```
[
    {
        "Time": "09/14/2021 07:55:56",
        "Temperature": "76.5",
        "Pressure": "1023.34"
    },
    {
        "Time": "09/14/2021 08:55:42",
        "Temperature": "73.0",
        "Pressure": "1024.07"
    },
{
        "Time": "09/14/2021 09:55:58",
        "Temperature": "71.0",
        "Pressure": "1017.1"
    }
]
```

Example 2

In example 2, you want to parse the following **CSV string**:

```
"09/14/2021 07:55:56"->"true"->"-12"->"Good"CR
"09/14/2021 07:56:50"->"false"->"0"->"Bad"CR
"09/14/2021 07:57:52"->"true"->"42.07"->"Unknown"
```

The following conditions should apply:

Function Parameters	Description	Character Used
`recordSeparator`	Record separator: Carriage return (CR)	CR
`fieldSeparator`	Field separator	Tab (→)
`quoteCharacter`	Quotation marks	Double quotation marks
`containsHeader`	Contains header	No

Expression:`csvParse('csvString',cr,tab,doublequote,false)`

Result:

```
[
    {
        "0": "09/14/2021 07:55:56",
        "1": "true",
```

```
        "2": "-12",
        "3": "Good"
    },
    {
        "0": "09/14/2021 07:56:50",
        "1": "false",
        "2": "0",
        "3": "Bad"
    },
{
        "0": "09/14/2021 07:57:52",
        "1": "true",
        "2": "42.07",
        "3": "Unknown"
    }
]
```

> **i** Note
>
> The record separator in the last record is optional.

Example 3

In example 3, you want to parse the following **CSV string**:

```
'13, Main Street' '' 'Springfield'CR
'Dietmar-Hopp-Allee 11CRBuilding 1' ''EVZ'' 'Walldorf'CR
```

The following conditions should apply:

Function Parameters	Description	Character Used
`recordSeparator`	Record separator: Carriage return (CR)	CR
`fieldSeparator`	Field separator	Space
`quoteCharacter`	Quotation marks	Single quotation marks
`containsHeader`	Contains header	No

Expression:`csvParse('csvString',cr," ",singlequote,false)`

Result:

This leads to an error because the first and second fields of the second record contain the record separator CR and the quotation mark '.

4.4.4.6.2 Application Example for structCompDelete(obj, [component 1, ..., component n-1], component)

You can use this function to delete the component `component` under the component path `component 1` to `component n-1` of a structured data type instance `obj`. (See also: Functions in the Expression Editor [page 555].)

The following section explains how the example structures are used:

`obj`:

```
{
    "quality": "good",
    "values":
    {
        "temperature": 78,
        "pressure": 42
    }
}
```

`componentArray`:

```
[
    "values",
    "pressure"
]
```

Output Structures

Expression Example	Result
`structCompDelete('obj',"quality")`	`{ "values": { "temperature": 78, "pressure": 42 } }`
`structCompDelete('obj',"values")`	`{ "quality": "good" }`
`structCompDelete('obj',"values","temperature")`	`{ "quality": "good", "values": { "pressure": 42 } }`

Expression Example	Result
`structCompDelete('obj','componentArray')`	`{ "quality": "good", "values": { "temperature": 78 } }`
`structCompDelete('obj',"AnInvalidComponent")`	`{ "quality": "good", "values": { "temperature": 78, "pressure": 42 } }`

Related Information

Structured Data Types [page 584]

4.4.4.7 Examples of Message Texts

Concept

The following examples explain how you need to formulate an expression to achieve a specific output in a notification message.

Features

- **Output an example of a value and add text:**
  In this case, the value of a tag is output (such as the temperature measured) and the term "degrees" is added. You enter the following expression in the *Expression* field:
  `'Double1' & " degrees"`
  Result:
  PCo sends the following expression in the notification message to the destination system:
  55 degrees

- **Example of a condition:**
  If the value of a tag is greater than 44, the value and degrees should be output in the notification message; for example, `45 degrees`. If the value is less than 44 degrees, the text `Inform Administrator` should be output. Enter the following expression in the *Expression* field:
  `if('Double1'>44, 'Double1' & " degrees", "Inform Administrator")`
  Result:
  If the value of a tag is less than or equal to 44 degrees, PCo sends the following expression in the notification message:
  Inform Administrator
- **Output an example of a value and then convert it:**
  The value of a tag specifies the temperature in degrees Celsius. The temperature should also be converted into degrees Fahrenheit. To do this, you enter the expression **`'Double1' & " degrees Celsius, " & ('Double1'*1.8+32) & " Fahrenheit"`**.
  Result:
  PCo sends the following expression in the notification message:
  32 degrees Celsius, 89.6 degrees Fahrenheit

4.4.5 Message Bundling

Use

Message bundling is a procedure in which notification messages can no longer be sent individually; instead, they are bundled in one large notification message and sent together to the connected destination system.

You can activate message bundling on the *Message Delivery* tab for the notification.

Message bundling is only supported by the following types of destination systems:

- *Simulation destination system*
- *MII destination system*
- *RFC destination system* (only RFC destination systems of the type **SAP NW**)
- *ODBC destination system*
- *Universal Web service destination system*.

> **i** Note
>
> - Message bundling is used for all destination systems that you assign to the notification.
> - Message bundling cannot be activated if you have set *Process Notification Messages Exactly Once in Order* on the *Host* tab.

Features

In the *Message Bundling* screen area, you can activate the function by selecting at least one of the checkboxes that are described below. At the same time, you define the criteria here according to which you want messages to be bundled.

Field	Description
Fixed Number of Messages	If you select this checkbox, a bundled notification message is created as soon as the number of individual messages you have specified has been reached. If you have only selected this criterion for message bundling, a bundled notification message contains exactly the number of individual messages you have specified here. The default setting is 10 messages.
Maximum Accumulation Time	Select the checkbox and, if required, enter a time period in seconds (the default setting is 60 seconds). In periodic intervals, after the agent has started, PCo bundles all individual messages that have accumulated within a notification for a specific destination system into one notification message. If you have selected both checkboxes, a bundled message is generated and sent as soon as one of the criteria has been fulfilled. In this case, a bundled notification message might contain fewer individual messages, or the bundled notification message is created in a shorter time period, as soon as the specified number of messages has been reached.

Example

You have entered **2** in the *Fixed Number of Messages* field. This leads to the following:

A message is generated due to a tag value change. It is put in the queue. A second message is generated after a further tag value change. It is also put in the queue. Now the messages are bundled; in other words, both messages are bundled into one notification message. This notification message is sent to the destination system.

You can display the bundled notification messages on the *Messages for Bundling* (agent instance) tab. (See also: Agent Instance: Displaying Messages [page 490].)

4.4.6 Deadband

Use

In the notification process, you can specify a value for the deadband (see also: Deadband). This serves as a limit value to filter out measurement values.

Integration

- You can specify the *deadband* for **OPC DA source systems** on the *Settings* tab (in the Group Settings area) of the source system. For OPC DA source systems, you can only specify the deadband as a percentage. The

deadband percentage indicates the bandwidth in which the value of a tag must at least change for the source system to transfer the tag to PCo. See also: OPC DA Source System: Settings Tab [page 135].

- For all other agents, you specify the deadband for the agent instance on the *Subscription Items* tab. On this tab, you select the tag or tags. You can also specify a *deadband* here. The deadband can only be specified as an absolute value here. See also: Agent Instance: Subscription Items Tab [page 476].)

Example

You are monitoring the tags of an OPC DA source system. On the OPC server, you have defined a minimum and a maximum value for the tag that is to be monitored.

In PCo, on the *Settings* tab of the OPC DA source system, you have defined a **deadband of 10%**.

The percentage that you enter in the *Deadband* field is applied to the difference between the minimum and maximum value. You have specified for the tag that is to be monitored that you want the tag value to always lie between 0 and 100. With a deadband of 10%, a tag value must change down or up by 10 compared to the last reported value so that a notification is sent from the OPC server. If a tag value is 50 and changes to 52, no information is sent to PCo. If the tag value changes later to 58, still no information is sent. Only when the tag value reaches 60 is information issued to PCo from the OPC server. The OPC source system then sends the tag value to PCo. In this case, PCo creates a notification message and sends it to the destination system, for example, to an SAP MII system. The SAP MII system can then send a mail to a previously defined recipient or create a maintenance notification in the ERP system for this.

4.5 Handling of Data Types

This section describes the PCo data types and their usage.

Related Information

PCo Data Types [page 582]
Functions in the Import Data Types Dialog Box [page 587]
Select Data Type (Selection Dialog) [page 588]
Processing Character Strings in JSON Format [page 588]

4.5.1 PCo Data Types

PCo Data Type Repository

The data types for a specific PCo installation are stored in the PCo data type repository. The data types are defined here for the following configuration elements:

- Subscription items
- Output expressions and their calculations and context items
- Input and output variables of destination systems
- Input and output parameters of methods
- Variables and calculations in multiple call destination systems

Data Type Categories

The PCo data types are divided into the following categories:

- Elementary Data Types [page 583]
- Structured Data Types [page 584]
- Array Data Types [page 584]
- Enumeration Data Types [page 584]

There are restrictions with regard to the support of the various PCo data type categories. (See: Support of Data Type Categories [page 585].)

Namespaces

PCo data types are organized in namespaces so that a data type can be assigned to a particular business context, if necessary. A PCo data type is identified by the combination of namespace and data type name. (See also: Namespaces [page 586].)

In the *Select Data Type* dialog box, the data types provided by SAP are displayed with the namespace description *(SAP data types)*.

You can change the namespace and name when importing data types from any proposed PCo data type.

Customer-Specific Data Types

You can create customer-specific data types in the PCo data type repository by importing data types from the following configuration elements:

- OPC UA source system (see: Import OPC UA Data Types into the PCo Data Type Repository [page 169])
- MQTT source system (see: Import Data Types into the Data Type Repository [page 226])
- MQTT destination system (see: Datentypen in das Datentypverzeichnis importieren [page 388])
- Universal Web service destination system (see: Import Data Types into the Data Type Repository [page 309])

Subsequent changing or deletion of customer-specific data types from the PCo data type repository is not supported.

4.5.1.1 Elementary Data Types

These data types are based on the elementary .NET data types. PCo supports the following elementary .NET data types:

Elementary Data Types

Data Type	Description
`System.Boolean`	Boolean value: `true` or `false`
`System.Byte`	8-bit unsigned integer
`System.Char`	Character as a UTF-16 code unit
`System.DateTime`	Time, typically expressed as a date and time of day
`System.Double`	Double-precision floating point number
`System.Guid`	Globally unique identifier (GUID)
`System.Int16`	16-bit signed integer
`System.Int32`	32-bit signed integer
`System.Int64`	64-bit signed integer
`System.Object`	Base data type of all PCo data types; it is the root of the data type hierarchy.
`System.SByte`	8-bit signed integer
`System.Single`	Single-precision floating point number
`System.String`	Text as a sequence of UTF-16 code units
`System.UInt16`	16-bit unsigned integer
`System.UInt32`	32-bit unsigned integer

4.5.1.2 Structured Data Types

A structured data type describes a data type that consists of several components. The component of a structured data type has a name, a data type, and a namespace as attributes.

> **i** Note
>
> The name of the namespace can contain any characters so a period in the name has no significance.

The structured data types are used to generate structured data objects that represent a list of uniquely named components and their values. The components of a data structure themselves have data types that are elementary, structured, array, or enumeration data types. In this way it is possible to create deeply nested data structures. Structured data types can be derived from one another to map a type hierarchy.

The PCo data type repository also allows abstract structured data types. Abstract data types define basic components of derived data types, but data objects cannot be created from abstract data types. A structured data object of type `System.Object` can contain any number of components with any data types.

4.5.1.3 Array Data Types

These data types are the basis for the generation of one-dimensional array objects. An array is a data structure that consists of a sequence of similarly structured data. Individual elements of an array can be referenced by their index that indicates the position of the element in the sequence.

Array data types have an element data type that determines which data type the array elements must have. Element data types can be elementary, structured, enumeration, and even array data types. Multidimensional arrays can be realized by using array data types as element data types of array data types.

4.5.1.4 Enumeration Data Types

These data types are used to define fixed values that have a unique textual value and a unique integer value.

> **✤** Example
>
> The enumeration data type `StoreErrorType` has the following textual fixed values and related numeric fixed values (that are given in brackets): `TagNotFound (0)`, `DataTypeConversionError (1)`, `UnknownError (2)`.

4.5.1.5 Support of Data Type Categories

The various PCo data type categories are supported as follows:

	Elementary Data Types Based on .NET Data Types*	Array Data Types Based on Elementary .NET Data Types**	Other Array Data Types	Structured Data Types	Enumeration Data Types
OPC UA Source Systems	X	X	X	X	(X)***
Other Source System Types	X	X	-	-	-
Output Expressions	X	X	X	X	X
Context Items of Output Expressions	X	X	-	-	-
Multiple Call Destination System	X	X	X	X	X
Universal Web Service Destination System	X	X	X	X	X
MQTT destination system	X	X	X	X	X
Simulation Destination System	X	X	X	X	X
Other Destination Systems	X	X	-	-	-
OPC UA Server and Web Server Methods	X	X	-	-	-

* `System.Boolean`, `System.Byte`, `System.Char`, `System.DateTime`, `System.Double`, `System.Guid`, `System.Int16`, `System.Int32`, `System.Int64`, `System.SByte`, `System.Single`, `System.String`, `System.UInt16`, `System.UInt32`, `System.Object`. The source system type and destination system type determine which of these data types is supported.

** `System.Boolean[]`, `System.Byte[]`, `System.Char[]`, `System.DateTime[]`, `System.Double[]`, `System.Guid[]`, `System.Int16[]`, `System.Int32[]`, `System.Int64[]`, `System.SByte[]`, `System.Single[]`, `System.String[]`, `System.UInt16[]`, `System.UInt32[]`. The source system type and destination system type determine which of these data types is supported.

*** OPC UA enumeration data types can only be used as component data types of structured data types. Therefore, you cannot import OPC UA enumeration data types as enumeration data types into the data type repository. Furthermore, you cannot assign an enumeration data type to a subscription item that has an OPC UA data type.

4.5.1.6 Namespaces

A namespace organizes data types in the PCo installation environment, for example, from a business perspective. The combination of namespace and data type represents a unique data type definition.

> **Example**
>
> The namespaces `ns1` and `ns2` exist. Within both namespaces, there is a data type called `MyDataType`. If you wanted to generate structure objects of both data types using the function `structNew()`, you would use the following notation:
>
> ```
> structNew("ns1", "MyDataType")
> ```
>
> ```
> structNew("ns2", "MyDataType")
> ```

Data types delivered by SAP and integrated firmly in PCo are located in a namespace identified by an **empty string ""**. Moreover, namespaces that start with `"sap."` – this is not case sensitive – are reserved for SAP. However, customer-specific data types must be organized in customer-specific namespaces, for example, `ns1` or `ns2`.

> **Note**
>
> In some functions of the expression editor, specifying the SAP namespace is optional so that, for example, the following two expressions are equivalent:
>
> - `structNew("", "TimeStampUnixStruct")`
> - `structNew("TimeStampUnixStruct")`

4.5.2 Functions in the Import Data Types Dialog Box

In the *Import Data Types into Data Type Repository* dialog box, you can use the following functions to complete the import of your data types:

Functions

Function	Description
Change PCo Data Type	You can change the namespace and name of any proposed PCo data type. To do this, choose the *Change PCo Data Type* button in the row of the relevant PCo data type. In the next dialog box, you can select a different namespace or create a new one by overwriting the given namespace. (See also: Namespaces [page 586].)
Create Array Data Type	You can also create a corresponding array data type of each proposed PCo data type by selecting the *Create Array Data Type* checkbox.
Restore the Original State	You can use the *Restore the Original State* button to undo changes that you made to namespaces, names, and array data types in the dialog box.
Cancel Data Type Import	You can use the *Cancel* button to cancel the data type import.
Import	You can use the *Import* pushbutton to apply your changes and trigger the import of the data types into the PCo data type repository.
Save All Changes	The *Save All Changes* pushbutton updates the PCo configuration database. Until then, you can undo the data type import by choosing *Reload Configuration*.

> **i Note**
>
> PCo does not support the creation of customer-specific **elementary** PCo data types.

Related Information

Import Data Types into the Data Type Repository [page 309]
Import OPC UA Data Types into the PCo Data Type Repository [page 169]

4.5.3 Select Data Type (Selection Dialog)

For configuration elements that support data types from the PCo data type repository, you can select the data type in a dialog box instead of from a dropdown box. As a rule, the selection dialog usually has four tab pages, corresponding to the four data type categories:

- Elementary data types
- Array data types
- Structured data types
- Enumeration data types

Details for the data types are displayed on the relevant tab pages. You can click on the links to navigate to each element or component data type and use the arrow keys to navigate back to the starting point.

Choose a data type or a component of a structured data type by selecting the row and then leave the dialog box by choosing *Select*. Alternatively, you can double-click on a data type or a component to select it.

> **i** Note
>
> If only elementary data types are permitted for a particular use case, for example, in the query destination system, the remaining tab pages are hidden.

Related Information

PCo Data Types [page 582]

4.5.4 Processing Character Strings in JSON Format

SAP Plant Connectivity can serialize data objects into a character string with JSON-compliant notation. In other words, it can convert a structured data object into a sequential display format in JSON format. The conversion is also supported in the opposite direction: A data object can be deserialized from a character string in JSON notation

JSON stands for **JavaScript Object Notation**. The data objects can have elementary, structured, array, or enumeration data types and be nested to any depth. Circular references of objects and data types are not supported.

Serialized objects can be used, for example, to process text files using the file system source system in which business objects, such as machine or order data, is available in a structured text form. An additional example of this application is the transfer of structured data to SAP MII using JSON-formatted character strings.

To generate a JSON character string from a data object, use the function `toJson()`. The counterpart is the function `fromJson()` for generating a data object with a specific target data type from a JSON character string.

The various PCo data type categories are serialized and deserialized as follows:

- Data objects of elementary data types are displayed as a character string in a culture-independent format.

  Example

  - Boolean values: `true, false`
  - Integer values: `42, -1898847`
  - Decimal values: `29.45, -123.45`
  - Strings: **This is a string**
  - Date values: `2019-08-20T12:14:57.9013335Z`

- A structured object is displayed as an unordered list of properties and values, for example:

  Sample Code

  ```
  {
          "category": "reference",
          "author": "Max Mustermann",
          "title": "C# Programming Guide",
          "isbn": "9999-13345-65665-0897",
          "price": 55.0,
          "publishingDate": "2019-08-20T12:28:58.5936598Z"
  }
  ```

  In addition, the JSON character string can contain information about the PCo data types of the structured objects. In the function `toJson()`, you define the name of a property that denotes the type information, for example, **"__type"**.

  Sample Code

  ```
  {
          "__type": {
            "namespace": "ns1",
            "name": "Book"
          },
          "category": "reference",
          "author": "Max Mustermann",
          "title": "C# Programming Guide",
          "isbn": "9999-13345-65665-0897",
          "price": 55.0,
          "publishingDate": "2019-08-20T12:28:58.5936598Z"
  }
  ```

  To be able to deserialize a structured data object with a specific PCo data type from a JSON character string, the function `fromJson()` also needs the type information property to be specified.

- Arrays are displayed as an ordered list of elements separated by commas. The following example shows an array consisting of two elements that represent structured data objects with type information:

  Sample Code

  ```
  [
        {
          "__type": {
            "namespace": "ns1",
            "name": "FreightBicycle"
          },
          "maxCapacity": 13,
  ```

```
        "hasTrailer": true,
        "color": "Black",
        "price": 1399.99
      },
      {
        "__type": {
          "namespace": "ns1",
          "name": "SportBicycle"
        },
        "admissionForTourDeFrance": false,
        "color": "Yellow",
        "price": 888.88
      }
]
```

- Data objects that are represented within PCo as “list”, for example, an object that was generated by the function `emptylist`, are converted like arrays into a JSON character string.

  > **i Note**
  >
  > Note that the reverse scenario, namely generating a “list” data object from a JSON character string, is not possible.

- Data objects that are represented within PCo as “map”, for example, an object that was generated using the function `emptymap`, are converted like arrays from key-value pairs into a JSON character string. The key of a map entry is denoted with the property `key`, the value with the property `value`.
  Example:

  > **Sample Code**
  >
  > ```
  > [{
  >         "key": "key1",
  >         "value": "2019-08-20T12:45:27.8129832Z"
  >     }, {
  >         "key": "key2",
  >         "value": 12.34
  >     }, {
  >         "key": "key3",
  >         "value": true
  >     }
  > ]
  > ```

  > **i Note**
  >
  > Note that the reverse scenario, namely generating a “map” data object from a JSON character string, is not possible.

5 Special Functions of the Management Console

5.1 Logging

The following types of logging are provided:

- **Audit log**
  The audit log documents all changes that a user has made to the PCo configuration, for example:
  - Date/time of change
  - User that made the change
  - Changed object
  - The operation that was carried out, such as *Add* or *Update*
  - Previous value and new value

  To be able to display the audit log, choose *View* > *Audit Log* from the PCo Console menu.

  To define the displayed columns for the audit log, choose *Tools* > *Options* from the PCo Console menu.
- **Agent instance log**
  This log logs the activities or errors of the agent instance. This depends on the setting in the *Log Level* field on the *Host* tab for the agent instance (see Agent Instance: Host Tab [page 413]).
  Logging is performed using Windows Event Logs where the log entries are optimized for display in the Management Console. The log is updated in **the English language**, irrespective of the language settings in the Management Console.
  A separate event log, whose name is generated automatically when the agent is created (*SAP_<sequence number>*) is created for each agent instance.
  You can display the logs on the *Log* tab for the agent instance. (See Agent Instance: Log Tab [page 415].)
  You can choose the *Manage Log* function key to call the Windows Event Viewer of the event log that is assigned to the agent. You can specify the maximum size of the log there and control whether, when this maximum size is reached, old log entries are overwritten or if you want an archive to be created automatically. If required, you can also delete old log entries here.

  > ⚠ Caution
  >
  > If you have set the *log level* to **Verbose** or **Information**, a very large number of messages are written to the log with the result that older messages might be pushed out of the log. Therefore, only set this *log level* for diagnostic purposes and make sure that the log is large enough. By default, event logs are created with a size of 4 MB for new PCo agent instances and old entries are overwritten when this size is reached. Approximately 2000 entries can be written to the log for each megabyte of log size.

- **Windows Event Log Usage**
  The Windows Event Log *Usage* is also used for logging agent instances. All warning and error messages for an agent instance are also sent to this Windows log. Moreover, messages created by the operating system also appear there in relation to agent instances, such as messages for starting and stopping the agent service or serious exceptions. You can use the Windows Event Viewer to display these log entries.

5.2 Certificate Overview

Use

With the certificate overview, you can gain an overview of all certificates that are stored in the PCo configuration. The certificate overview evaluates the security configuration of all relevant configuration elements and checks the certificates that are found.

To call the certificate overview, choose *View* > *Certificate Overview* from the menu.

> ### Example
>
> You can use the certificate overview to monitor when certificates expire or when you need to identify the configuration elements for which appropriate certificates need to be maintained after a configuration is imported from another computer.

Prerequisites

In the Management Console, you have used at least one configuration element for which certificates are required. The certificate overview supports the following certificate usages:

- **Host for the cloud services**: Server certificate and certificate for internal Web socket communication
- **MII query server**: Server certificate
- **MQTT client:** Application certificate
- **OPC UA client:** Application certificate
- **Web server:** Server certificate for each endpoint
- **Universal Web service destination system:** Certificate for authentication
- **WebSocket:** Server certificate and the root certificate for the WebSocket as client
- **Management services**: Certificate for transport security (Transport Layer Security TLS)
- **Main service**: Certificate for transport security of configuration services (Transport Layer Security TLS)

Functions

As of release 15.4, the certificate overview informs you **when a certificate is going to expire**. An error symbol is displayed beside the certificates whose validity is due to expire within a specified time interval.

You can define this *time interval* below the certificate overview. The default time interval is 30 days. This allows you to identify at an early stage which certificates need to be renewed.

On the *Configured Certificates* tab, you can see the application certificates that you have defined yourself for specific configuration elements, for example, the application certificates for the OPC UA source systems or the

server certificates of the OPC UA server. The main details are displayed for each certificate. The error icon at the beginning of each table row informs you of any issues with the respective certificate.

> **i Note**
>
> If you click on the respective configuration element, you can navigate to the screen on which the certificate can be configured. When you return to the certificate overview, the last row you processed is highlighted.

On the *Trusted Certificates* tab, you see, for each store location, the certificates that you trusted explicitly when configuring the connections by moving them from the store location for rejected certificates to the store location for trusted certificates. For each store location, you see the configuration elements that refer to this store location. You navigate to the store locations and to the configuration elements by clicking on a link or using the arrow button.

On the *Trusted Issuer Certificates* tab, you see, for each store location, the issuer of the certificates that you have trusted explicitly. The functions on this tab are analogous to those on the *Trusted Certificates* tab.

> **i Note**
>
> On the tabs for trusted certificates and trusted issuer certificates, a direct assignment to the configuration elements is not possible because this would necessitate a temporary test connection. For that reason, it is possible that multiple configuration elements that come into consideration for usage of the certificate are displayed for each store location and certificate. To gain a better overview of the certificates received, it might therefore be useful, during configuration, to define a separate store location for these certificates for each configuration element.

If no certificates are stored in the PCo configuration, a corresponding message is displayed instead of the certificate overview.

5.3 Identification Type of Certificates

When you configure server or client certificates or certificates for authentication in Plant Connectivity, you can define the identification type of the certificate for some applications. This enables you to determine how you want the selected certificate to be identified by the Plant Connectivity system at runtime. The identification type is supported for the following configuration elements:

- OPC UA source system
- OPC UA server
- Universal Web service destination system

The following options are available in the *Identification Type* field:

- If you select the *Identification By Thumbprint* option, the unique thumbprint identifies the configured certificate. This is the default setting.
- If you select the *Identification By Subject (Allows Certificate Rotation)* option, the subject of the certificate is used at runtime to select the appropriate certificate from the list of certificates in the selected storage location. If there are multiple certificates with this subject, the certificate with the latest valid-to date is used.

> **i Note**
>
> - The system does not select a certificate that is not yet valid.
> - The certificate with the latest valid-to date is selected from all certificates that are currently valid or that have possibly expired.

With this setting, the system supports **certificate rotation**. The certificate can be replaced at runtime without the configuration needing to be changed.

> **i Note**
>
> Certificate rotation is not a function of Plant Connectivity. You must provide a new certificate yourself, for example, using the corresponding functions of the operating system.

During productive operation, the certificate is automatically renewed once it has expired, provided it is assigned to the same subject as the previously configured certificate. Therefore, the new certificate retains the original subject of the certificate but has a new thumbprint and typically a longer validity, that is, a later valid-to date.

Further Notes

Make sure that the trust relationship between the server and client is retained after a certificate exchange. When doing so, consider the following aspects:

- If you have established the trust relationship using the certificate of a certification authority (CA) and then renew the client certificate or the server certificate that was signed by that CA, the trust relationship is preserved and you do not need to do anything.
- However, if you have stored the client certificate in the trust store of the server to establish the trust relationship, you probably need to update the trust store of the server with the renewed application certificate. Conversely, this applies to a server certificate that the client must trust. The renewed server certificate may have to be stored in the client's trust store in this case too.

5.4 Backing Up and Restoring PCo Configuration Data

Use

By backing up the PCo configuration data, you can make sure that you can restore your PCo configuration at any time if necessary. However, you can also use the backup function to distribute stable configurations as a whole to other computers in your company, for example, from a test computer to a productive computer. The function still allows you to switch back and forth between various configuration options on one computer.

SAP recommends that you perform a backup after important configuration changes. The following configuration data can be backed up and restored:

- Source systems

- Destination systems
- Agent instances
- System settings

> **Note**
>
> The logs of the agent instances and the notification messages that are still in the queue are not backed up.

Prerequisites

To restore the PCo data after a backup, you need to stop all agent instances. Moreover, make sure that no application, such as a text editor, a Windows Explorer, or a command prompt, is accessing the configuration folders of Plant Connectivity.

Features

Backup

Two types of data backup are supported:

- Password-protected backup files
  The password-protected files are generated by the PCo Management Console when the *Create Backup* function is executed. You define the password when you save. The password-protected files can then be shifted freely from one computer to another.
  When you save, PCo proposes a name for the backup file. The file name that is proposed has the following format: **computer name_timestamp.pbf**

  > **Example**
  >
  > `WDFN33345212A_2015.06.17_15.46.36.pbf`

  The folder `%ProgramData%\SAP\PCo\Backup` is provided as the default folder for data backups. However, you can choose a different file name and a different folder for storing the backup file.
- Local backup files
  During a new installation or upgrade and before restoring a backup, PCo automatically generates local backup files. These are not password-protected and can only be used for restoring data on the local computer. The local backup files are stored in the `%ProgramData%\SAP\PCo\Backup` folder under the name **computer name(local)_time stamp.pbf**.

  > **Note**
  >
  > You cannot generate local backup files manually.

You can use the *Create Backup* function to save the configuration of the PCo system. To do so, proceed as follows:

1. In the menu, choose *Plant Connectivity* > *Create Backup* .

The *PCo Configuration Backup* dialog box appears. PCo has already proposed a file name and the default storage folder for backup files.

2. Choose *Save*.
   The *Enter Password* dialog box appears.
3. Enter the password twice and confirm your entries.
   You receive a success message containing the path where the backup file is stored.

Restoring Data

You can use the *Restore Backup* function to restore the configuration of a previously-saved state. To do so, proceed as follows:

1. Stop the running agent instances, and choose *Plant Connectivity* > *Restore Backup* in the menu.
   The *Restore Backup* dialog box appears.
2. Select the backup file that you want to restore.
3. If you want to restore a password-protected backup file, you need to enter the password that you defined when you created the backup.
4. Specify whether you also want to restore system settings and settings for cloud integration from the backup file.
   However, this only makes sense in exceptional cases, for example, if a PCo installation is to be completely rebuilt after a new installation. When system files are adopted from a backup from another computer, problems may occur with the configured ports for the main service.

When you execute the restore function, the entire configuration, including the source systems, destination systems, agent instances, and, if applicable, the system settings, are restored to the previously-saved state. A local backup is performed automatically before recovery.

If parameters of the main service have changed as a result of restoring the backup, this is restarted automatically. The Management Console may also be restarted to enable access to the changed configuration services. If necessary, correct the settings for the main service by choosing *Tools* > *Options* > *Global Settings* > *Main Service*. (See also: Configuring the Main Service [page 14].)

> ⚠ Caution
>
> The restore function **replaces** the entire configuration. Any source systems, destination systems, and agent instances that you generated after creating the backup are deleted when the configuration is restored. Alternatively, use the function for exporting and importing the configuration if you only want to transfer individual agent instances and their dependent source and destination systems between computers.
>
> If the backup contains source systems that use libraries from other software vendors, such as **Aspentech IP21** or **GE Proficy Historian**, the recovery may fail if these libraries are not yet set up for the current PCo installation. Follow the instructions in the *Post-Installation* section of the *Installation Guide* to prepare PCo for using these source system types.

5.4.1 Setting Up an Automatic Backup

Use

You can use the timer source system and an implementation of enhanced notification processing delivered by SAP to set up a periodic automatic data backup.

Set Up Automatic Data Backup

In the menu of the Management Console, choose *Plant Connectivity* > *Set Up Automatic Backup*, to generate an example configuration **automatically** for creating automatic data backups. The system creates a source system and an associated agent instance with the name `_AutoBackupConfig`. Adjust the execution plan of the timer in the source system `_AutoBackupConfig` to suit your needs and start the agent instance.

Set Up Automatic Data Backup Manually

If you want to set up the automatic data backup manually, proceed as follows:

1. Create a timer source system Q and define a timer T in it with the appropriate execution plan. (See also: Timer Source System [page 229].)
2. Create an agent instance A for source system Q and browse to generate a subscription item S for timer T.
3. On the *Notification Processing* tab, in the *Enhanced Notification Processing* screen section, select the option *Automatic Configuration Backup*.
   This selects the default implementation for automatic backup of the configuration. (See also: Enhanced Notification Processing [page 73].)
4. Create the destination system for this notification processing by choosing the *Create Destination System* pushbutton on this tab.
5. Create a static notification for agent instance A.
6. On the *Output* tab of the notification, generate the output expression for subscription item S by choosing the *Generate Expressions* pushbutton.
7. On the Destinations tab, assign the destination system */Enhanced Notification Processing* to the notification. No other settings are required.

Local Backup

If you choose **Start Agent Instance** after setting up the automatic backup, the Management Console stores a local backup of the configuration at each time specified by the timer of the timer source system in the directory

`%programdata%\SAP\Backup`. If necessary, you can import this local backup without entering a password, but it is bound to the computer on which it was created.

5.5 Exporting and Importing Configuration Elements

Use

You can use the **Export/Import** function to export the configuration of one or more agent instances with all dependent settings (such as the configuration of the source system) to an XML file and then import this XML file into another PCo system. You can also export the configuration of individual source and destination systems. This reduces the configuration effort required when setting up further PCo instances.

> **i** Note
>
> Note that PCo does not provide a function for exporting or importing the PCo data type repository. If your configuration elements use customer-specific data types and you want to use export/import to transfer these configuration elements to other PCo instances, you must ensure that the PCo data type repository are identical on all PCo instances. You can find the PCo data type repository as a file `DataTypeRepository.json` in the PCo configuration directory.

Procedure

Exporting Configuration Elements

1. Starting from the *Plant Connectivity Management Console* screen, choose the menu path ▶ *Plant Connectivity* ▶ *Export/Import Configuration* ▶ or choose the relevant icon from the taskbar.
   The *Import and Export Tool* dialog box appears.
2. Choose the option *Export Configuration Elements*.
   The *Agent Instances*, *Source Systems*, and *Destination Systems* tabs are displayed.
3. Choose the configuration elements on the tabs that you want to export.
   The system automatically determines any dependencies to other configuration elements and selects them. It makes sure that the configuration is always exported consistently. The system then creates an XML file.
4. Save the XML file and enter a password.
   The PCo system issues a message saying that the export was successful.
5. If you only want to export a specific agent instance, you can select it and then call the context menu using the right mouse button. In the context menu, choose the *Export Agent Instance* function. The agent instance you want is already selected in the *Import and Export Tool* dialog box.

Importing Configuration Elements

1. Starting from the *Plant Connectivity Management Console* screen, choose the menu path ▶ *Plant Connectivity* ▶ *Export/Import Configuration* ▶ or choose the relevant icon from the taskbar.

The *Import and Export Tool* dialog box appears.

2. In the dialog box, select the *Import Configuration* option and choose *Browse*.
   The *Open* dialog box appears.
3. Select the XML file you want and choose *Open*.
   The PCo system adopts the path of the XML file.
4. Choose *OK* and enter the password.
   The system imports the settings from the XML file.
   If conflicts arise, the *Import Conflicts To Be Resolved* dialog box appears. Here you can overwrite the existing version of a configuration element or import the element as a copy.

5.6 Starting the PCo Management Console in Display Mode

Use

You can start the *PCo Management Console* in display mode, if required. For example, you can use this function if you want to give a support colleague (with a limited user account) access to displaying Plant Connectivity logs or configuration data.

Prerequisites

The *PCo Management Console* always starts automatically in display mode if it is started by a Windows user who does not have a user account with administrator rights.

The *PCo Management Console* can be started in display mode, if required, by means of a command line parameter but also by an administrator user. (See: Setting the Command Line Parameters [page 600])

Features

If the *PCo Management Console* is running in display mode, you can display the current configuration of all source and destination systems as well as all agent instances. The corresponding symbols in front of the agent instances provide information about the status of each agent instance. You can choose the logs for the agent instances and export them; you can also choose where the log is stored. You can display notifications that could not yet be sent by an agent instance or which are faulty. You can adjust the user-specific options of the *PCo Management Console*, such as the language settings or the date format being used.

> ⚠ Caution
>
> The following restrictions apply in display mode:
>
> - You cannot start or stop any agent instances
> - You cannot change the configuration of source and destination systems or agent instances
> - You cannot change any settings that affect Plant Connectivity as a whole, such as the settings for the *Management Host* or for *SLD registration*

- You cannot delete any logs or notifications
- You cannot import any configurations of agent instances

> **i Note**
>
> To enable the display mode, central and agent-specific files are copied from the *PCo Management Console* into a temporary directory in the user profile. For that reason, additional short delays are possible especially when you are selecting logs. The temporary files are deleted when the *PCo Management Console* is closed.

If required, when the *PCo Management Console* is running in display mode, you can switch to the administrator mode by choosing the *Run as Administrator* pushbutton. If the *PCo Management Console* has been started by a user who does not have administrator rights, you have to enter the user name and password of a user with administrator rights. If you can switch modes, the *PCo Management Console* is shut down and restarted in administrator mode.

> **i Note**
>
> The starting of the *PCo Management Console* in administrator mode can be blocked by global system settings if you only have a restricted user account. If so, you receive an error message from the operating system when you start other programs from the context menu entry *Run as Administrator*.

5.6.1 Setting the Command Line Parameters

Context

You can start the *PCo Management Console* in display mode from an administrator user, if required, using a command line parameter.

Procedure

1. Find the entry for the *PCo Management Console* in the start menu. If you have a standard installation, you can find the PCo Management Console under *All Programs* > *SAP* > *Plant Connectivity* > *Management Console*.
2. Create a link on your desktop by dragging the start menu entry for the *PCo Management Console* to the desktop while holding down the CTRL (Control) key.
3. To call the context menu, right-click the link. Choose *Properties*.
4. In the *Properties* dialog box, in the *Destination* field, add the switch addition **`/DisplayMode`** after the quotation marks to the path for the `ManagementConsole.exe` file that is to be run so that the entry looks like the following example: *"C:\Program Files (x86)\SAP\Plant Connectivity\System \ManagementConsole.exe" /DisplayMode*

Results

If you start the *PCo Management Console* using the link you have created on the desktop, it is opened in display mode.

5.7 Controlling the Cursor Using the TAB Key

Use

In the *PCo Management Console*, you can move the cursor forward without the mouse by using the `TAB` key to reach specific fields or pushbuttons. You can speed up navigation between the left and right sides of the user interface by using the key combination `ALT-ENTER`.

If you have selected an entry in the list of the source systems, destination systems, or agent instances, you can use the key combination `ALT-ENTER` to move the cursor directly to the detailed display for the relevant system or agent instance, instead of using the `TAB` key to move slowly over the individual fields to get there.

> **i** Note
>
> The key combination is displayed in the PCo menu under *Display* > *Switch Details/Items*.

Procedure

1. Choose the object you want, such as the source system, on the left-hand side using the `TAB` key and the arrow keys.
2. Choose the `ALT-ENTER` key combination.
   The cursor then moves to the right-hand side to the first tab of the selected object.
3. You can now move the cursor forward within the tab using the `TAB` key or switch tabs using the arrow keys.
   The cursor then moves from field to field and also to the pushbuttons. To call the function of a pushbutton, choose the `ENTER` key.

6 Remote Client

Use

The *Remote Client* is available for monitoring and administration of your PCo systems in production. The *Remote Client* is a snap-in of the *Microsoft Management Console (MMC 3.0)*, which is provided by SAP together with the *SAP Plant Connectivity* component. With the *Remote Client*, you can monitor all PCo systems with their individual agent instances at a glance as well as start and stop the agent instances.

Monitoring the PCo systems makes sense in particular if you are using the **notification process**.

In addition to monitoring using the *Remote Client*, you can use active monitoring in connection with SAP MII.

> ℹ Note
>
> The language in which the *Remote Client* is displayed depends on the language of the Windows operating system. If a display language is set for Windows that is not supported by PCo, the *Remote Client* is displayed in English.

Prerequisites

- You have installed the Remote Client together with PCo.

  > ⚠ Caution
  >
  > Note that the Remote Client cannot be installed as a standalone component. You need to install at least the *PCo Core Components* together with the Remote Client because only then can the necessary PCo core files be installed. This is a prerequisite for the Remote Client then working.

- You ensure that the main service is started and that the management services are activated. To make settings for the main service, choose *Tools* > *Options* > *Global Settings* > *Main Service* in the menu of the *PCo Management Console*.
- If necessary, you perform an SLD registration in the *PCo Management Console*. This is only necessary, however, if you want to adopt the PCo systems in the *Remote Client* using the *Use System Landscape Directory (SLD)* function. (See also: Registering PCo in the SLD [page 22].)
- In the authorization management of the PCo system on which the Remote Client is executed, you have enabled the appropriate access for the Remote Client. You can find the relevant settings in the menu under *Tools* > *Authorization Management*. (See also: Authorization Management [page 24].)
- On the PCo system that is to be accessed using the Remote Client, appropriate access to the management services must be allowed in authorization management. (See also: Authorization Management [page 24].)

Features

Call the Remote Client

The *Remote Client* is an external PCo component. Therefore, you call the *Remote Client* by choosing *Start* > *Programs* > *SAP* > *Plant Connectivity* > *Remote Client*.

Set the Properties of the Remote Client

In the *Remote Client*, click on *SAP Plant Connectivity Systems* and call up the context menu. Choose the *Properties* entry. The *Plant Connectivity Remote Client Properties* dialog box appears.

Field	Description
Connection Refresh Period (Sec)	Specifies how often you want the connection to the monitored PCo system to be checked. If 60 seconds has been specified, a check is made every 60 seconds to see if the PCo system is running.
Connection Check Timeout	If 60 seconds has been specified, and there has been no response from the monitored PCo system for 60 seconds, the connection to this PCo system is terminated.
Log Level	Here you can set the level of detail for the log, for example, verbose, error.
Selection Period (Days)	Specifies in which past period the logs of the agent instances are to be exported.
Allow Import of Manually Changed Configurations	You set this indicator if you want to send manually changed configuration files to a PCo system by means of the *Remote Client*. The destination PCo system must have at least the release status *15.0* or *2.304* (PCo 2.3 with Support Package 4). **⚠ Caution** SAP does not assume any liability for the effects of the import of a modified configuration file.

Add PCo System

In the *Remote Client*, using the context menu, you can add the PCo systems that you want to monitor. You can choose from the following approaches:

- Adding a PCo System Using Host Name [page 605]
- Adding a PCo System via SLD [page 606]

After you have added the PCo systems to be monitored in the Remote Client, you can check the state of these PCo systems with all agent instances.

Monitoring PCo Systems

The *Remote Client* displays a list of all installed PCo machines. For each PCo system, you can display the agent instances that you have created. The system status of the PCo systems is displayed in different colors:

- Gray: Not accessible
- Green: Accessible

The system status of the agent instances is also displayed:

- Green: Agent instance running
- Red: Agent instance faulty
- Gray: Agent instance stopped

In the Remote Client you can start, stop, or restart an agent instance without being logged on to the affected PCo system, by selecting the agent instance in the Remote Client and choosing the right-hand mouse button.

Active Monitoring (with SAP MII)

The **Active Monitor** component, which you can select during the PCo installation, supports you in monitoring agent instances during productive operation. The agent instances are monitored automatically, so that outages of communication connections are reported immediately and actively. In active monitoring, you receive an e-mail from SAP MII if a PCo instance cannot be reached or an agent instance changes its status to *Error*. Such a change in status always occurs if the agent loses its connection to the data source, for example.

You also need a configured SAP MII transaction for active monitoring. An administrator can configure a warning that is sent to the SAP MII transaction if the PCo system cannot be reached from the *Remote Client* or if the agent instance runs with errors. In this case, the corresponding MII transaction can be configured to send an e-mail.

Active monitoring checks every 60 seconds whether the agent instances are running or whether they are faulty or cannot be reached. If an agent instance cannot be reached, this is an emergency situation and SAP MII sends an appropriate e-mail.

To activate active monitoring in connection with SAP MII, proceed as follows:

- In **SAP MII**, you create a transaction and define an e-mail that is to be sent if the PCo system to be monitored can no longer be reached.
- In the *Remote Client*, you select the PCo system for which you want to enable active monitoring and choose *Enable Active Monitoring* from the context menu (right mouse button).
  The *SAP MII Alert Configuration* dialog box appears. You enter the following data:

Field	Description
Server Name	Name of the MII server
Port	Port of the MII server
Use Secure Sockets	You can select this checkbox to set up a secure connection to MII.
Version	Choose the version of the SAP MII system with which you want to connect.
User Name and Password	User name and password with which you log on to the MII system

Field	Description
Transaction Name	Name of the MII transaction that you created previously in SAP MII. If you have set up a connection to SAP MII, the transactions defined previously are displayed in the *Transaction Details* area. Click on the transaction you want. The transaction name then appears in the *Transaction Name* field.
Name of the Input Parameter	Name of the input parameter of the selected transaction When the MII transaction is selected, this field is supplied with the parameters of the selected transaction. The transaction parameters are shown in this field. Select the input parameter.

Active Monitoring (with Standard Windows Operating System Functions)

In this form of active monitoring, you are actively informed when certain events occur when agent instances are executed. The notification that you want can be configured using the standard Windows operating system functions. For more information, see the SAP Note: 2869474.

More Information

Additional Monitoring Tools [page 607]

6.1 Adding a PCo System Using Host Name

Procedure

1. In the *Remote Client*, click on *SAP Plant Connectivity Systems*. From the menu, choose *Action* > *Add System to Monitoring*.

   The *Plant Connectivity Remote Client Settings* dialog box appears.
2. Choose the *Use Host Name* function key.

   A new dialog box appears.
3. Enter the host name (for example, `vmw3346`).
4. Check the *Security Settings*. *Windows* and *Use SSL* must be selected.
5. Choose the *Get System Information* function key.

The system displays the system name and other system information in the lower part of the dialog box. If required, you can change the description of the system that is to be added.

6. Choose the *Add System* function key.

   The PCo system is displayed in the list of monitored PCo systems in the *Remote Client Settings for Plant Connectivity* dialog box.

7. Choose the function key *OK* to adopt the selected PCo system in the *Remote Client*.

6.2 Adding a PCo System via SLD

Prerequisites

You have carried out an SLD registration. See also: Registering PCo in the SLD [page 22]

Context

This function provides you with an overview of all existing PCo systems in the system landscape.

Procedure

1. In the *Remote Client*, click on *SAP Plant Connectivity Systems*. From the menu, choose *Action* > *Add System to Monitoring*.

   The *Plant Connectivity Remote Client Settings* dialog box appears.

2. Choose the *Use SLD* function key.

   A new dialog box appears.

3. In the *SLD URL* field, enter the URL and the port for the System Landscape Directory (for example, `http://pwdf6412.wdf.sap.corp. 50000`), and enter the *user name* and *password*.

4. Choose the *Read Registered Systems* function key.

   The system displays a list of PCo systems.

5. Select a PCo system and choose the *Select System* function key.

   The *Plant Connectivity Remote Client Settings* dialog box appears.

6. Here, check that *Windows* and *Use SSL* are selected.

7. Choose the *Call System Information* function key. If required, change the description of the system to be added.

8. If the selected PCo system is connected, the *Add System* function key is active and you can add the system.
9. Repeat the steps to add further PCo systems to the remote client.

Results

The selected PCo systems are displayed in the *Remote Client* with their agent instances.

6.3 Additional Monitoring Tools

Use

Performance Monitor

In addition to the Remote Client, PCo uses the *Performance Monitor* provided by Windows. If you install PCo, *Performance Counters* are created automatically. These performance counters, in conjunction with the counters provided by Windows as standard, facilitate the monitoring of PCo systems.

> **i** Note
>
> To call the Performance Monitor, proceed as follows:
>
> 1. Click *Start*.
> 2. Click the *Start Search Box*.
> 3. Enter **`perfmon`** and choose `Enter`.

The following table provides an overview of performance counters that can be used to monitor running agent instances:

Performance Counter	Description
Notification Deliveries/Second	This performance counter states the number of delivered notification messages per second. You can use this performance counter to monitor a running agent instance; this can be combined with the CPU utilization to be able to recognize the effect of the delivery of notification messages on the processor.
Notification Failures	This performance counter monitors the failed notification messages of a running agent instance. A notification message is always regarded as failed if it could not be sent within the set number of retries. Failed messages are displayed on the *Message Failures* tab.

Notification Retries	This performance counter monitors the number of retries for sending a notification message. If PCo fails when sending a notification message, the notification message is displayed on the *Message Retries* tab.
DispatchThread Count and Dispatch Retry Thread Count	This performance counter monitors the total load for the agent instance with regard to sending notification messages.
Working Set	This performance counter measures the memory consumption caused by the agent instance.

Interface Monitor

Separate monitoring tools are available for the interfaces RFC and HTTP/HTTPS:

Interface	Monitor	Description
HTTP/HTTPS	.NET diagnostics	The configuration file `ManagementConsole.EXE.CONFIG` can be implemented for network tracing for the PCo console or the agent instance. For this, the configuration file needs to be supplemented with the code below.
RFC	Monitoring for RFC interface with transaction `STAD`	You can use the transaction `STAD` to call various statistics data for RFC that inform you of the progress of the actions you have performed. You can filter this data according to various criteria (for example, by user, time, server name, and so on) on the selection screen of the transaction. For more information, see the *SAP NetWeaver documentation* under **Analyzing RFC Statistical Data** and **Displaying a Business Transaction Analysis**.

.NET Network Monitoring

In the configuration file `ManagementConsole.EXE.CONFIG` you need to add the following section of code manually to be able to implement .NET Network Monitoring:

```
<system.diagnostics>

<trace autoflush="true"/>

<sources>
```

```
<source name="System.Net" maxdatasize="1024">
<listeners>
<add name="NetworkMonitor"/>
</listeners>
</source>
<source name="System.Net.Sockets" maxdatasize="1024">
<listeners>
<add name="NetworkMonitor"/>
</listeners>
</source>
</sources>
<sharedListeners>
<add name="NetworkMonitor" type="System.Diagnostics.TextWriterTraceListener"
initializeData="NetworkMonitor_trace.log"/>
</sharedListeners>
<switches>
<add name="System.Net" value="Verbose"/>
<add name="System.Net.Sockets" value="Verbose"/>
</switches>
</system.diagnostics>
```

After you have inserted the code, you can display the file `NetworkMonitor_trace.log` in the PCo system directory. This is a text file in which it is logged what happens in the network when destination systems and agent instances are configured.

> ⚠ Caution
>
> You need to delete this section of code when you have finished monitoring. If you do not delete this section, system performance deteriorates and detailed log files are created.

More Information

For more information about monitoring, see the *Application Operations Guide* in the SAP Help Portal under https://help.sap.com/viewer/p/SAP_PLANT_CONNECTIVITY.

7 Notes for Installing SAP PCo

You can find the installation guide for PCo on SAP Help Portal under https://help.sap.com/pco in the section *Installation and Upgrade*.

Important Disclaimers and Legal Information

Hyperlinks

Some links are classified by an icon and/or a mouseover text. These links provide additional information.
About the icons:

- Links with the icon : You are entering a Web site that is not hosted by SAP. By using such links, you agree (unless expressly stated otherwise in your agreements with SAP) to this:
  - The content of the linked-to site is not SAP documentation. You may not infer any product claims against SAP based on this information.
  - SAP does not agree or disagree with the content on the linked-to site, nor does SAP warrant the availability and correctness. SAP shall not be liable for any damages caused by the use of such content unless damages have been caused by SAP's gross negligence or willful misconduct.
- Links with the icon : You are leaving the documentation for that particular SAP product or service and are entering a SAP-hosted Web site. By using such links, you agree that (unless expressly stated otherwise in your agreements with SAP) you may not infer any product claims against SAP based on this information.

Videos Hosted on External Platforms

Some videos may point to third-party video hosting platforms. SAP cannot guarantee the future availability of videos stored on these platforms. Furthermore, any advertisements or other content hosted on these platforms (for example, suggested videos or by navigating to other videos hosted on the same site), are not within the control or responsibility of SAP.

Beta and Other Experimental Features

Experimental features are not part of the officially delivered scope that SAP guarantees for future releases. This means that experimental features may be changed by SAP at any time for any reason without notice. Experimental features are not for productive use. You may not demonstrate, test, examine, evaluate or otherwise use the experimental features in a live operating environment or with data that has not been sufficiently backed up.
The purpose of experimental features is to get feedback early on, allowing customers and partners to influence the future product accordingly. By providing your feedback (e.g. in the SAP Community), you accept that intellectual property rights of the contributions or derivative works shall remain the exclusive property of SAP.

Example Code

Any software coding and/or code snippets are examples. They are not for productive use. The example code is only intended to better explain and visualize the syntax and phrasing rules. SAP does not warrant the correctness and completeness of the example code. SAP shall not be liable for errors or damages caused by the use of example code unless damages have been caused by SAP's gross negligence or willful misconduct.

Bias-Free Language

SAP supports a culture of diversity and inclusion. Whenever possible, we use unbiased language in our documentation to refer to people of all cultures, ethnicities, genders, and abilities.

www.sap.com/contactsap

THE BEST RUN SAP