Control Expert AssetLink

User Guide

EIO0000004195.07
03/2022

www.se.com

Legal Information

Table of Contents

Safety Information

Important Information

Read these instructions carefully, and look at the equipment to become familiar with the device before trying to install, operate, service, or maintain it. The following special messages may appear throughout this documentation or on the equipment to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

The addition of this symbol to a “Danger” or “Warning” safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.

This is the safety alert symbol. It is used to alert you to potential personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

⚠ DANGER

DANGER indicates a hazardous situation which, if not avoided, **will result in** death or serious injury.

⚠ WARNING

WARNING indicates a hazardous situation which, if not avoided, **could result in** death or serious injury.

⚠ CAUTION

CAUTION indicates a hazardous situation which, if not avoided, **could result** in minor or moderate injury.

NOTICE

NOTICE is used to address practices not related to physical injury.

Please Note

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this material.

A qualified person is one who has skills and knowledge related to the construction and operation of electrical equipment and its installation, and has received safety training to recognize and avoid the hazards involved.

Before You Begin

Do not use this product on machinery lacking effective point-of-operation guarding. Lack of effective point-of-operation guarding on a machine can result in serious injury to the operator of that machine.

⚠ WARNING

UNGUARDED EQUIPMENT

- Do not use this software and related automation equipment on equipment which does not have point-of-operation protection.
- Do not reach into machinery during operation.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

This automation equipment and related software is used to control a variety of industrial processes. The type or model of automation equipment suitable for each application will vary depending on factors such as the control function required, degree of protection required, production methods, unusual conditions, government regulations, etc. In some applications, more than one processor may be required, as when backup redundancy is needed.

Only you, the user, machine builder or system integrator can be aware of all the conditions and factors present during setup, operation, and maintenance of the machine and, therefore, can determine the automation equipment and the related safeties and interlocks which can be properly used. When selecting automation and control equipment and related software for a particular application, you should refer to the applicable local and national standards and regulations. The National Safety Council's Accident Prevention Manual (nationally recognized in the United States of America) also provides much useful information.

In some applications, such as packaging machinery, additional operator protection such as point-of-operation guarding must be provided. This is necessary if the operator's hands and other parts of the body are free to enter the pinch points or other hazardous areas and serious injury can occur. Software products alone cannot protect an operator from injury. For this reason the software cannot be substituted for or take the place of point-of-operation protection.

Ensure that appropriate safeties and mechanical/electrical interlocks related to point-of-operation protection have been installed and are operational before placing the equipment into service. All interlocks and safeties related to point-of-operation protection must be coordinated with the related automation equipment and software programming.

NOTE: Coordination of safeties and mechanical/electrical interlocks for point-of-operation protection is outside the scope of the Function Block Library, System User Guide, or other implementation referenced in this documentation.

Start-up and Test

Before using electrical control and automation equipment for regular operation after installation, the system should be given a start-up test by qualified personnel to verify correct operation of the equipment. It is important that arrangements for such a check are made and that enough time is allowed to perform complete and satisfactory testing.

⚠ WARNING

EQUIPMENT OPERATION HAZARD

- Verify that all installation and set up procedures have been completed.
- Before operational tests are performed, remove all blocks or other temporary holding means used for shipment from all component devices.
- Remove tools, meters, and debris from equipment.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Follow all start-up tests recommended in the equipment documentation. Store all equipment documentation for future references.

Software testing must be done in both simulated and real environments.

Verify that the completed system is free from all short circuits and temporary grounds that are not installed according to local regulations (according to the National Electrical Code in the U.S.A, for instance). If high-potential voltage testing is necessary, follow recommendations in equipment documentation to prevent accidental equipment damage.

Before energizing equipment:

- Remove tools, meters, and debris from equipment.
- Close the equipment enclosure door.
- Remove all temporary grounds from incoming power lines.
- Perform all start-up tests recommended by the manufacturer.

Operation and Adjustments

The following precautions are from the NEMA Standards Publication ICS 7.1-1995 (English version prevails):

- Regardless of the care exercised in the design and manufacture of equipment or in the selection and ratings of components, there are hazards that can be encountered if such equipment is improperly operated.
- It is sometimes possible to misadjust the equipment and thus produce unsatisfactory or unsafe operation. Always use the manufacturer's instructions as a guide for functional adjustments. Personnel who have access to these adjustments should be familiar with the equipment manufacturer's instructions and the machinery used with the electrical equipment.
- Only those operational adjustments actually required by the operator should be accessible to the operator. Access to other controls should be restricted to prevent unauthorized changes in operating characteristics.

About the Book

Document Scope

This document describes the use and performance of EcoStruxure™ Control Expert AssetLink.

Validity Note

This document is valid for EcoStruxure™ Control Expert AssetLink V2.2 or later.

For product compliance and environmental information (RoHS, REACH, PEP, EOLI, etc.), go to www.se.com/ww/en/work/support/green-premium/.

The technical characteristics of the devices described in the present document also appear online. To access the information online, go to the Schneider Electric home page www.se.com/ww/en/download/.

The characteristics that are described in the present document should be the same as those characteristics that appear online. In line with our policy of constant improvement, we may revise content over time to improve clarity and accuracy. If you see a difference between the document and online information, use the online information as your reference.

Related Documents

Title of documentation	Reference number
PlantStruxure General Purpose Library for WSP	EIO0000002094

You can download these technical publications, the present document and other technical information from our website www.se.com/en/download/.

Introducing EcoStruxure™ Control Expert AssetLink

About AssetLink

Introduction

This overview introduces EcoStruxure™ Control Expert AssetLink. Use this tool to convert a control project from EcoStruxure™ Process Expert for AVEVA System Platform or EcoStruxure™ Control Expert or EcoStruxure™ Machine Expert to a AVEVA System Platform (ASP) project.

Current Customer Challenges

In the past, users have created narrowly focused and often painstakingly engineered custom tools to manage the integration of automated processes from PAC Modicon controllers with a ASP-based supervisory layer.

The implementation for such a tool requires the creation of logic and control sequences within Control Expert. Then, the users create the related communication objects and regular AppObjects that represent production assets in the ASP supervisory application.

The burdensome process of linking supervisory objects to the corresponding variables in the PAC controller can be prone to mistakes that are discovered only in the late stages of the product lifecycle.

This approach can obviously increase costs and risks.

Product Objectives

AssetLink is a build-time engineering tool that addresses the challenges above by assisting users in the engineering of the ASP supervisory application by automating the creation and updating of application based on data extracted from Control Expert or Machine Expert control projects.

These are the main design principles of AssetLink:

- ArchestrA IDE drives the process while AssetLink is delivered as aN ASP template.
- In this asset-oriented process, AssetLink can retrieve and combine multiple data from one or more variables in the control project and use them from AppObjects modeling Assets.
- AssetLink does not facilitate a one-time generation of temporary objects. Instead, it serves the entire lifecycle of an automation system.

  NOTE: This document refers to the *PlantStruxure General Purpose Library for ASP* (2018 R3 (GPL)), but you can use your own libraries.

By delivering mechanisms that facilitate the reuse of information already available from in a Control Expert project that controls both process machines and the process, the engineering of the supervisory application has these benefits:

- *quality:* The number of detected errors that are buried in the ASP supervisory layer can be greatly reduced.
- *cost:* The engineering effort and the risk of errors can be significantly reduced because process-control information is reused from the supervisory application.
- *delivery:* The reduced engineering effort accelerates the time of delivery for automation systems. The introduction of small changes during production is shorter and helps to limit risk.
- *innovation:* Patterns that are automatically recognized and that require a minimal effort to model increase flexibility in the implementation of different integration strategies (for example, naming conventions that are applied).

Product Licensing

AssetLink is a licensed product. Observe this license activation process:

Stage	Description
1	Install AssetLink.
2	Activate the AssetLink license.

NOTE:

- You can use the trial version for 30 days with full functionality.
- After trial period is expired, only 3 pattern files can be loaded.
- If the license is activated, restart the AssetLink tool to see the updated status of license.
- AssetLink 1.0 is node-locked license, so it uses the **License Manager** for the license activation. But AssetLink 2.1 has floating license, so you have to activate the license using the **Floating License Manager**.

Before You Begin

Introduction

Review this information before you use the EcoStruxure™ Control Expert AssetLink tool.

Software Requirements

This software is required for AssetLink V2.2 operations:

- Wonderware System Platform 2014 R2 SP1 or Wonderware System Platform 2017 or AVEVA System Platform 2020 or AVEVA System Platform 2020 R2
- EcoStruxure™ Process Expert for AVEVA System Platform
- Control Expert (formerly Unity Pro)
- Machine Expert (formerly SoMachine)
- OPC DA server v3.61 or later or OPC UA Server v1.0 or later
- Any XML editor (Recommended - Altova XMLSpy for Schemas and XML Documents)
- Schneider Electric License Manager 2.5 or later
- .Net Framework 4.7.2 or later
- Telemetry Server Communication Drivers 2020 (Build 82.7416)
- Remote Connect R2.5.1
- Microsoft Office Excel (Required to support SCADAPack)

Display Settings

In your computer's **Display** settings, select these **Scale and layout** settings to implement the best resolution of the AssetLink tool on your monitor:

- *size (text, apps, etc.):* 100%
- *resolution:* 1920 x 1080
- *orientation:* landscape

Prerequisites

Readers of this document should have a working familiarity with these software programs:

- **EcoStruxure™ Control Expert** (formerly Unity Pro)**:** Control Expert is the configuration tool for PAC Modicon projects. Your choice of libraries is not limited to the PAC Modicon General Purpose Library (GPL), but AssetLink applies systematic rules that AssetLink applies to other libraries to automatically determine which ASP application objects are created and retrieved from the Control Expert project.
- **EcoStruxure™ Machine Expert** (formerly SoMachine)**:** Machine Expert saves engineering time through intuitive machine programming with one of the most modern and powerful tool-based software concepts on the market.
- **OPC Factory Server** (OFS)**:** AssetLink manages application object I/O references that indicate OPC DA Items through the standard Wonderware DIO OP client when it is connected to the OFS.
- **AVEVA System Platform** (ASP)**:** This industrial software platform uses ArchestrA technology for HMI operations management, SCADA supervision, and production and performance management. ASP contains an integrated set of services and an extensible data model to manage plant control and information management systems. It supports both the supervisory control layer and the manufacturing execution system layer, presenting them as a single information source. Modular applications sit on top of the ASP Platform.
- **General Purpose Library** - **Wonderware System Platform 2018 R4**: Ensure that Wonderware System Platform galaxy is created with GPL templates in ready to use state.

Before you begin the usage of Ecostruxure™ AssetLink tool, export the appropriate source file, page 11.

Exporting Control Projects

Introduction

The EcoStruxure™ Control Expert AssetLink tool requires a source control project that adheres to these file formats:

- *.xsy:* Export an .xsy file from Control Expert (formerly Unity Pro).
- *.xml:* Export an .xml file from Machine Expert (formerly SoMachine).
- *.xls:* Export an .xls file from Remote Connect.

Export a Source File from Control Expert

Create an .xsy source file for the conversion:

Step	Action
1	Open the source control project in Control Expert.
2	Access the **Export** dialog box (**File** > **Export Project**).
3	Enter a project name in the **File name** field.
4	Scroll to **Data (*.XSY)** in the **Save as type** field.
5	Click the **Export** button.

Export a Source File from Machine Expert

Create an .xml source file for the conversion:

Step	Action
1	Open the source control project in Machine Expert.
2	In the **Project Explorer**, right/click **Project Name > Add Object > Add Symbol configuration**.
3	Build the project.
4	Select (check) the sections you want to include in the variable (*.xml*) file.
5	Rebuild the project.
6	Save the project to generate a source project in a directory with this extension: `*_Application.MyController.XML extension`

Export a Supporting File from Remote Connect

Create an .xls source file for the conversion:

Step	Action
1	Open **SCADAPack x70 Logic EditorRemote Connect**..
2	Click on **SCADAPack x70 Controller Settings - DeviceDTM** .
3	From the **Context Menu** select **Additional Functions** and click on **Export to Excel File**.
4	A pop-up window will appear. Click **Browse** and select the folder in which the file has to be saved and provide the name for the *.xls* file. **NOTE:** Ensure that the .xsy and .xls have the same names and are saved in the same location.

Installation

AssetLink Installation

Instructions

Install the EcoStruxure™ Control Expert AssetLink program:

Step	Action
1	Double-click the AssetLink installation file (`Ecostruxure Control Expert - Asset Link.msi`). **NOTE:** The installation of .Net Framework 4.7.2 or later is a prerequisite for the installation of AssetLink. If .Net Framework is not installed, the system installs it automatically. For systems that run Windows 8.1 and Windows Server 2012, update the window before you install .Net Framework. If the installation fails, install any necessary upgrades. After a successful installation, restart your system and continue with the installation procedure.
2	Wait a few moments for the installation wizard to open and click the **Next** button.
3	Accept the licensing agreement and click the **Next** button.
4	Enter the appropriate information on the **Customer Information** page and click the **Next** button. **NOTE: License Manager** 2.5.0 and **Floating License Manager** 2.5.0 will be installed automatically.
5	In the **Destination Folder** dialog box, click the **Change** button to navigate to a storage location for the installation files. Alternately, you can accept the default destination folder: `C:\ProgramData\Schneider Electric\Ecostruxure Control Expert - Asset Link` **NOTE:** Windows hides the `ProgramData` folder by default. Perform the steps mentioned in the table below to view the folder.
6	Click the **Next** button to see the **Ready to Install the Program** dialog box.
7	Click the **Install** button and wait for the installation to finish.
8	Click the **Finish** button to complete the installation.

Follow these steps to see the `ProgramData` folder:

Step	Action
a.	Open any Windows **File Explorer** folder.
b.	Click the **View** tab.
c.	Expand the **Options** menu and scroll to **Change folder and search options** to open the **Folder Options** dialog box.
d.	Click the **View** tab.
e.	In the **Advanced settings** pane, double-click **Files and Folders** and **Hidden files and folders** to expand them.
f.	Select **Show hidden files, folders, and drives**.
g.	Press the **Apply** or **OK** button to implement the changes.

Installation Contents

The installation folder contains these sub-folders:

Folder	Description
AssetLink Template and Pattern Schema	This folder contains `$EsxCEAssetLink.aaPKG` AssetLink template and `PACConnectorSchema.xsd` file. The .xml schema in the `PACConnectorSchema.xsd` file can be used by standard .xml editors to manually create patterns that conform to the appropriate syntax and structure, page 47. When you copy this file to every folder that contains patterns, AssetLink checks their validity too.
GPL Patterns	The General Purpose Library includes a set of patterns that are copied to folders that can be accessed by AssetLink or used as examples for the creation of new patterns. You can dedicate any folder to this purpose, but the use of a network folder allows AssetLink to access the patterns from workstations that run ArchestrA IDE.
GPL ASP Templates	The ASP base template `$EsxCEAssetLink` and its derived template `$aESxPACConnector` are delivered in a ASP object export file (. aaPDF), which is imported to Galaxy where AssetLink is used from, via the standard services for such purpose in the ArchestrA IDE from the action `Galaxy.Import.Object(s)` action. Once imported, ASP templates appear under the Template Toolbox 'EcoStruxure Plant.'
User guide	This folder contains a .pdf version of the AssetLink user guide (EIO0000004195).
Control Expert Variable File	This is a variable file for the control project that is used to get asset details.
Demo Templates and Patterns	These default patterns are used for demonstration purposes.
Machine Expert Template and Patterns	These patterns include templates that are used in Machine Expert and the patterns that are created for those templates.
SCADAPack Demo Templates and Patterns	These patterns include templates and patterns that are used in SCADAPack for demonstration purposes.
Release Notes	These release notes accompany the AssetLink delivery.

NOTE: For the installation of Modicon Libraries - General Purpose Library refer Appendix A, page 90

Using Demo Templates

Overview

This section describes how to use demo templates in the AssetLink.

Follow these steps to use demo templates in the AssetLink:

Step	Action
1	Open **Demo Templates and Patterns** folder from the installed location of the AssetLink.
2	Create new **Galaxy**.
3	Import **Galaxy Styles**.
4	Import **Script Function Libraries**.
5	Import demo templates (`$aPSxAnalogInput.aaPKG`, `$aPSxMotor.aaPKG`).
6	Import AssetLink template from the installed location.
7	Configure **Demo Patterns** in the pattern's path and then **Browse Control Project** in the **Generation** tab.

Work Flow

Process Overview

Getting Started

Follow these steps in ArchestrA IDE:

Step	Action
1	Import the ASP templates you want to use: • AppObjects and Infrastructure Templates as needed • AssetLink templates Import `$EsxCEAssetLink.aaPKG` template from installed path or from default path *C:\ProgramData\Schneider Electric\Ecostruxure Control Expert - Asset Link\Asset Link Template and Pattern Schema* to use the AssetLink tool.
2	Create the Galaxy communications infrastructure: • Create the required number of instances for WSP OPCClient DIOs (device integration objects) to communicate with PAC Modicon controllers through OFS. **NOTE:** Refer to the *PlantStruxure General Purpose Library for WSP Supervision Services User Guide* for more information. • In high-availability systems, create the required number of instances of ASP Redundant DIOs.
3	Create AssetLink objects in Galaxy: • We recommend that you use AssetLink from derived templates from $EsxCEAssetLink. (You do not have to create AppObject instances because they are used in build time; there is no requirement to deploy and execute such objects in run time.) • In most of cases, it is sufficient to create a single connector for each PAC Modicon controller. • If you required data at different scan rates from the same controller, we recommend that you create a single connector per controller and scan rate and segregate the patterns that are used from different connectors that point to the same controller.
4	Open the AssetLink object from the ArchestrA IDE: • AssetLink automatically connects to the Galaxy from which it was instantiated. • The login credentials are requested if Galaxy is secured.

Configuration Process

Select the configuration options:

Step	Action
1	Open the AssetLink **Configuration** tab.
2	In the **Source** field, select the type of source project in the pull-down menu: • **Control Expert/UnityPro** (*.xsy* files) • **SoMachine/Machine Expert** (*.xml* files) • **Process Expert** (*.xsy* files) • **SCADAPack** (*.xsy* files)
3	Make a selection in the **Protocol** pull-down menu: • **OPC UA** • **OPC DA** • **OPC UA EMBEDDED** • **Dynamic DNP3** **NOTE:** • The protocol **OPC UA EMBEDDED** is not applicable for **SoMachine/ Machine Expert**. • For the communication, **OPC DA** is not case sensitive. Whereas, **OPC UA** and **OPC UA Embedded** is case sensitive. Hence, we need to ensure control logic has deployed objects with right set of naming conventions followed for the entire logic.

Step	Action
	• The protocol **Dynamic DNP3** is applicable only for **SCADAPack**.
4	Make a selection in the **Device** field. **NOTE:** A Machine Expert control project does not require the selection of an OPC UA device.
5	In the **DIO Name** field, select an OPCClient template instance name.
6	Click the **Refresh** button to fetch the updated DIO name in the pull-down menu.
7	Enter an address in the **OI Address Reference** field. **NOTE:** Refer to the parameter descriptions for the **Configuration** tab, page 19.
8	In the **Optional Prefix of the AppObject Tagname** field, enter a prefix. **NOTE:** Refer to the parameter descriptions for the **Configuration** tab, page 19.
9	Click the browse button (**...**) associated with the **Patterns Path** field and drive to the folder that contains the respective templates.
10	Click the browse button (**...**) associated with the **Pattern Project** field to view control projects that were exported with the latest variables. **NOTE:** The selection of a pattern project is required only when you need new patterns for generation or old patterns for updating.
11	Click the browse button (**...**) associated with the **Control Project** field to view control projects in the .xsy and .xml formats that (by default) exported variables for instance generation. **NOTE:** • These variables are not necessarily the latest. • If **Process Expert** is selected as a source, it is suggested to have one System in **EPE** which is equivalent to one galaxy in System Platform. In case, if multiple systems from **EPE** occurs in the same galaxy, the respective identical objects coming from two different systems from **EPE** will have an impact in System Platform.

DNP3 Configuration

The **DNP3 Protocol** has nine parameters out of which three parameters **Telemetry Server Name, Outstation** (refer Configuration Tab, page 19), **Hierarchy** refer Configure Hierarchy, page 38) have to be configured by the user and the remaining are configured internally by **AssetLink**.

Follow the steps mentioned below to verify the addressing format for the WSP Objects.

- Open the Object Editor by double clicking on the generated object in the Model tab of **WSP**.
- Open the **Attributes Tab** in the Object Editor.
- Click  and then click  .
- Enable the **I/O Read** and **I/O Read/Write** under the **Enabled Features** list.
- Select an **Attribute** and view its details. This **Attribute** will contain the addressing format of the **Dynamic DNP3** protocol.

Create Patterns

Use these steps to create a new pattern after you have browsed the pattern projects for the latest or new variable tags:

Step	Action
1	In the AVEVA System Platform object browser, create an instance of a pattern tag name (with or without a prefix name) for the pattern to be generated
2	On the AssetLink **Patterns** tab, click the **Create Pattern** button to open the **Create Pattern** dialog box.

Step	Action
3	Populate these fields: • **TagName:** Assign a variable tagname to the new pattern. **NOTE:** Without a prefix, the **TagName** matches the one in the control project. • **Prefix in the Tagname:** Enter a prefix to be added to the tagname during the creation of an instance.
4	Click the **Generate Pattern** button in the **Create Pattern** dialog box to generate the new pattern.

Update Patterns

Update an existing pattern:

Step	Action
1	Open the AssetLink **Patterns** tab.
2	Click the **Refresh Pattern** button to load the latest .xml pattern files in the **Patterns** grid.
3	Click the **Update Pattern** button to open the **Update Pattern** dialog box.
4	Change the rule for a new tagname and enter the new value for **Tagname** to update the pattern rules. **NOTE:** This step is optional because AssetLink retrieves a tagname from the existing pattern by default.
5	Enter a new value in the **Prefix in the Tagname** field to update the pattern rule.
6	Click the **Update Pattern** button in the **Update Pattern** dialog box to update the existing pattern.

Update Galaxy

After browsing the control project pattern in the configuration procedure (above), follow these steps to generate an AppObject:

Step	Action
1	Open the AssetLink **Generation** tab.
2	Click the **Browse Control Project** button populate the **Generation** grid with a list of possible instances.
3	Click the check boxes that correspond to the instances you want to generate.
4	Click the **Generate Object** button and wait for the generation process to finish.
5	Confirm the completed generation by viewing the generated objects in the AVEVA System Platform's object browser.

Migration

Upgrade to AssetLink 2.2	**AssetLink 1.0.0/ 1.0.1 user**	**AssetLink 1.0.2 SP2 user (Imported $EsxCEAssetLink without deleting existing 1.0.1 template)**	**AssetLink 1.0.2 SP2 user (Imported $EsxCEAsset-Link after deleting existing 1.0.1 template)**	**AssetLink 1.0.2 SP2 user (Imported $EsxCEAsset-Link without deleting existing 1.0.2 SP2 template)**	**AssetLink 2.0.0/ 2.0.1/ 2.0.2 user**	**AssetLink 2.1 user**
Installation Procedure	User has to uninstall AssetLink 1.0.0/ 1.0.1 and install AssetLink 2.2.	User has to uninstall AssetLink 1.0.2 SP2 and install AssetLink 2.2.	User has to uninstall AssetLink 1.0.2 SP2 and install AssetLink 2.2.	User has to uninstall AssetLink 1.0.2 SP2 and install AssetLink 2.2.	User has to uninstall AssetLink 2.0.0/ 2.0.1/ 2.0.2 and install AssetLink 2.2.	User has to install AssetLink 2.2 for upgrading.
Using AssetLink Template	Re-create AssetLink object in ArchestrA IDE. Delete existing AssetLink 1.0.0/ 1.0.1 template and Import AssetLink 2.2 template - `$EsxCEAsset-Link.aaPKG`.	Re-create AssetLink object in ArchestrA IDE. Delete existing AssetLink 1.0.1 template and Import AssetLink 2.2 template - `$EsxCEAsset-Link.aaPKG`.	Import AssetLink 2.2 template - `$EsxCEAsset-Link.aaPKG`.	Import AssetLink 2.2 template - `$EsxCEAsset-Link.aaPKG`. (above existing 1.0.2 SP2 template)	Import AssetLink 2.2 template - `$EsxCEAsset-Link.aaPKG`. (Above the existing 2.0.0/ 2.0.1/ 2.0.2 template)	Import AssetLink 2.2 template - `$EsxCEAsset-Link.aaPKG`. (Above the existing 2.0.0/ 2.0.1/ 2.0.2/ 2.1.0 template)
Impact	Since previous version of AssetLink template is deleted, user has to re-enter settings of AssetLink. No impact in already generated ASP AppObjects.	Since previous version of AssetLink template is deleted, user has to re-enter settings of AssetLink. No impact in already generated ASP AppObjects.	Since it is new import of AssetLink, there is no impact. No impact in already generated ASP AppObjects.	Since it is import on previous version of AssetLink, there is no impact. No impact in already generated ASP AppObjects.	Since it is import on previous version of AssetLink, there is no impact. No impact in already generated ASP AppObjects.	Since it is import on previous version of AssetLink, there is no impact. No impact in already generated ASP AppObjects.

AssetLink Operations

Introduction

Use the instructions in this chapter to operate EcoStruxure™ Control Expert AssetLink.

Tabs

Configuration Tab

Galaxy Settings

Configure the **Galaxy Settings** on the **Configuration** tab:

Parameter	Description
Source	Select the type of source project in the pull-down menu: • **Control Expert/UnityPro** (*.xsy* files) • **SoMachine/Machine Expert** (*.xml* files) • **Process Expert** (*.xsy and .xml* files) • **SCADAPack** (*.xsy* files)
Protocol	Select a protocol in the pull-down menu: • **OPC UA** • **OPC DA** • **OPC UA EMBEDDED** • **Dynamic DNP3** **NOTE:** • The protocol **OPC UA EMBEDDED** is not applicable for **SoMachine/Machine Expert** • The protocol **Dynamic DNP3** is applicable only for **SCADAPack**
Root Area*	Select the type of **Root Area** in the pull-down menu. For more details refer to the section Create/ Recreate, page 21. **NOTE:** • This option is available only for **Source** of type **Process Expert**. • Click to fetch the updated **Root Area** in the pull-down menu. • If there are any refinement changes done for the area generated with previous type, those changes will be lost as the entire area will be recreated with the new type.
Area*	Select the type of **Area** in the pull-down menu. For more details refer to the section Create/ Recreate, page 21. **NOTE:** • This option is available only for **Source** of type **Process Expert**. • Click to fetch the updated **Area** in the pull-down menu.
Device Name**	This device name is an alias for the PAC Modicon controller in the OFS configuration. **NOTE:** A Machine Expert control project does not require the selection of an OPC UA device.
DIO Name**	This is the name of the OPCClient template instance.

Parameter		Description
Scan Group**		Select the scan group that is used from the previously selected DIO or Redundant DIO Instance.
OI Address Reference**		The referenced address combines these components: • *OI gateway:* This is the OI (operation integration) gateway that the user configures for communications. • *tag address:* The tag address is generated by the **Protocol** and **Device** parameters. Verify this reference in the OI gateway with the browsing tag and enter the appropriate name in this field. **NOTE:** • This field is enabled when: ◦ **OPC UA** protocol is implemented or ◦ **OPC UA EMBEDDED** is enabled • For an example of OI Address Reference see, page 22.
DNP3 Configuration***	**Telemetry Server Name**	It is an editable field in which the user has to enter the name of the Telemetry server.
	Outstation Full Name	It is an editable field in which the user has to enter the name of the Outstation. **NOTE:** Example: **TEST.A1.A2** where A2 is the outstation name and TEST.A1 is the location.
Optional Prefix of the AppObject Tagname		You can download the same control project to multiple PAC Modicon controllers (for example, Process OEM). In such cases, the project variables have the same name in each controller, but because each instance represents a different asset from the perspective of the supervisory entity, it has a different ASP AppObject Tagname. The **Optional Prefix** that you configure is added to each AppObject Tagname that AssetLink generates to create a uniquely named AppObject for each control project despite the identical variable names. **NOTE:** • The value of this prefix is added to the application object during generation. • Do not use an optional prefix if the variable names in the control project are those that you want to use for AppObjects.

NOTE:

* This option is disabled for the source **SCADAPack**

** This option does not appear when the selected source is **SCADAPack**

*** This option appears only when the **SCADAPack** is selected as the source.

Create/ Recreate

Root Area:

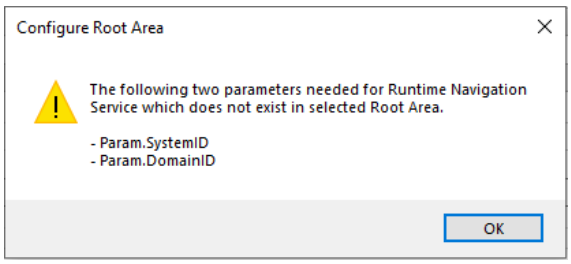
Step	Action
1	Configure the **Galaxy Settings** in the **Configuration** tab, by providing the pattern folder location, variable file.
2	Select **Process Expert** in the **Source** and select the type of **Root Area**.
3	Open **Generation** tab and Click **Browse Control Project** to update the objects list. Select the objects and click **Generate Objects**. **Result**: • If the galaxy does not have the **Root Area** then depending on the selected **Root Area** type the plant hierarchy is created. • If already there exists a **Root Area** in the galaxy, but the selected **Root Area** is different from the existing one then the current hierarchy is deleted and a new one is created depending on the plant hierarchy and a pop-up message is displays the time taken for recreating the hierarchy

Area:

Step	Action
1	Configure the **Galaxy Settings** in the **Configuration** tab, by providing the pattern folder location, variable file.
2	Select **Process Expert** in the **Source** and select the type of **Area**.
3	Open **Generation** tab and click **Browse Control Project** to update the objects list. Select the objects and click **Generate Objects**. **Result**: • If the galaxy does not have an **Area** then depending on the selected **Area** type the plant hierarchy and the **Area** is created. • If already there exists an **Area** in the galaxy, but the selected **Area** is different from the existing **Area** then the current hierarchy is deleted and a new one is created depending on the plant hierarchy.

NOTE:

- You can see the created objects in the **Model** tab of the ASP
- If the parameters required for Runtime Navigation Service do not exist for the selected **Root Area** then an alert message (shown below) will appear.



- If renamed object occurs for recreate, then creation of new object is the action that takes place.

Update/ Move

Root Area:

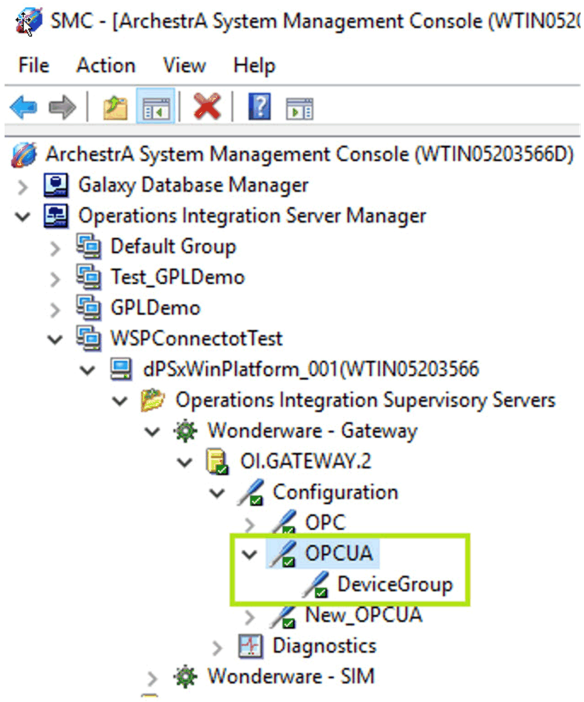
Step	Action
1	Configure the **Galaxy Settings** in the **Configuration** tab, by providing the pattern folder location, variable file.
2	Select **Process Expert** in the **Source** and select the type of **Area**.
3	Open **Generation** and Click **Browse Control Project** to update the objects list. Select the objects and click **Generate Objects**. **Result**: In the existing system change in plant hierarchy may change the internal ASP attributes of the Root Area, then the corresponding attributes of that system is updated and displayed in the Attributes tab of that system.

Area:

Step	Action
1	In the ASP **Model** tab drag and drop the objects to the required folder.
2	In **Generation** tab, click **Browse Control Project** to update the objects list. **Result**: Depending on the change in the plant hierarchy the corresponding **Areas** will be moved to the respective folders.

OI Address Reference Example

Follow these steps to see the components of the OI address reference:

Step	Action
1	Enter `run` in your computer's **Start** menu.
2	In the **Run** window, type the command `smc.msc` to open the **SMC** management console.
3	Expand the navigation tree in the **SMC** console to find the first part of the OI address in the **OI.GATEWAY** configuration. This information corresponds to user-defined names. In this case, `OPCUA.DeviceGroup` is the first part of the OI address: 

Step	Action
4	In the navigation tree, click **DeviceGroup** to open the **Node Type** dialog box.
5	On the **DeviceGroup Parameters** tab, click the **Browse OPCUA Server** button to view the variable tags.

This is a sample variable tag:

`/DA/0 : PLCSim!AnalogInput1_1_AINPUT1_ST.STW`

These are the components of the sample variable tag:

- `/DA/0`: This is the second part of the OI address.
- `PLCSim`: This is the Device name. (See the note below.)
- `AnalogInput1_1_AINPUT1_ST.STW`: This is the variable tag. (See the note below.)

  NOTE: You defined these particular values when you created an instance and added values to the **Configuration** tab.

Using the first part of the OI address from the table above and the second part from the list above, we see that the completed OI address is `OPCUA.DeviceGroup./DA/0`.

Patterns Settings

Configure the **Patterns Settings** on the **Configuration** tab:

Parameter	Description
Patterns Path	Enter the path to the .xml pattern files that AssetLink applies (The common pattern schema .xsd files has to be in this same folder). These patterns are scanned each time this path changes after you reopen AssetLink or press the **Refresh Patterns** button in the **Patterns** tab.
Pattern Project	This is the .xsy file that is used to create and update the pattern.

If trail period has expired, it will display a message "**AssetLink trail period expired. Activate a license and restart Assetlink Tool**". For more details, refer to product licensing, page 10 section.

Control Project Settings

Configure the **Control Project Settings** on the **Configuration** tab:

Parameter	Description
Control Project	This field contains the full name of the file that AssetLink scans for variables: • *.xsy* : Files with this extension correspond to the selection of a Control Expert project in the **Source** field. **NOTE:** *.xml* file of same name as *.xsy* file has to be available in the same path for Plant Model creation, page 42. • *.xml* : Files with this extension correspond to the selection of a Machine Expert project in the **Source** field .

Object Wizard

The attributes and graphics can either be enabled or disabled depending on the choices and options in the instances using the **Object Wizard** feature. The primary requirements to use this feature are:

- Template must be upgraded with the **Object Wizard**
- The **Patterns** must be upgraded with the **Object Wizard** parameters.

Patterns Tab

Introduction

Each row in the **Patterns** table represents a single .xml file in the configured folder. You can select or deselect patterns in this list.

License Availability

The availability of patterns depends on the license activation status:

Status	Behavior
License is not activated.	Only three patterns are loaded. Only instances related to those templates are available for generation.
License is activated.	All patterns are loaded. Instances related to all respective templates are available for generation.

For license activation process, refer to Product Licensing, page 10.

NOTE: If “vendor daemon” error message from the license manager is displayed while opening the AssetLink editor or while performing any operation in the editor, restart the computer and the AssetLink tool.

Buttons

These buttons appear on the **Patterns** tab:

Button	Description
Refresh Patterns	Click this button to reload all configured pattern files.
Select All	Click this button to select all patterns.
Unselect All	Click this button to deselect all patterns.
Save Pattern	Click this button to save the changes you made for the active pattern.
Create Pattern	Click the **Create Pattern** button to open the **Create Pattern** dialog box and define these parameters: • **TagName:** Enter the name of the instance for which the pattern is created. (The prerequisite example above uses the instance name `AnalogInput1_1`.) • **Prefix in the Tagname:** Assign a tagname prefix to the new pattern. Click the **Generate Pattern** button in the **Create Pattern** dialog box to apply the changes you made in the **Generate Pattern** dialog box. Another dialog box **Select Variable(s) for Creation Rule** will appear. Once the variable is selected the pattern opens in the Pattern Editor, page 26. Observe that the **Save** button is enabled only after the pattern is saved in the respective folder with a new name. **NOTE:** Refer to the pattern creation prerequisites above.

Button	Description
Add Pattern	Click **Add Pattern** to open the **Add Pattern file(s)** dialog box. In the add pattern file dialog box, select the pattern files (.xml format) and click **Open**. **Result**: The pattern files will be copied to the configured location and gets added in the patterns grid.
Update Pattern	Click the **Update Pattern** button to open the **Update Pattern** dialog box and define these parameters: • **Version:** This field shows the implemented pattern version. • **Updated:** This field shows the time of the last pattern update. • **Tagname:** Assign a tagname to the new pattern. • **Prefix in the Tagname:** Assign a tagname prefix to the new pattern. Click the **Update Pattern** button to apply the changes you made in the **Update Pattern** dialog box. Another dialog box **Source Pattern Found** will appear which indicates the detection of the source pattern *.xml* file. • Clicking on **Keep Changes** will open the pattern in the Pattern Editor, page 26. **NOTE:** Observe that the **Save** button is enabled only if the **Version** is changed or changes are made to the pattern. • Clicking on **New Pattern** will open the pattern in the Pattern Editor, page 26. **NOTE:** Observe that the **Save** button is enabled only after a new name is provided to the pattern. **NOTE:** Refer to the prerequisites above.

Table Columns

These columns appear on the **Patterns** tab:

Column	Description
Pattern File	This column displays the names of the available pattern files.
Version	This column displays the version of the attribute of the **Pattern File**.
Last Modification Date Time	This column displays the file saved time stamp for the **Pattern File**. Applies when the pattern file generated or modified by the AssetLink tool, or manually by the user.
Valid	This box is selected (checked) only when the pattern has passed the validation process, including the alignment to the expected pattern schema.
Selected	*selected:* Check this box to scan the **Pattern File**.
	deselected: Uncheck this box to remove the file from the scan.
Exists	This box is checked only when the pattern is associated with a Galaxy Template ID.
Template ID	This ASP template identifier is associated with the attribute in the **Pattern File** that corresponds to the pattern found in the file.
View	Click **View** to open the pattern file in the read-only mode. The content of pattern file will be displayed in the Pattern Editor, page 26.
Edit	Click **Edit** to open the pattern file in the edit mode. The content of pattern file will be displayed in the Pattern Editor, page 26. **NOTE:** • If you have opened a pattern file in the edit mode and try to edit the same pattern file in another derived template instance, it will open in read-only mode. • To edit the pattern file in the derived template instance, you have to close it in the first instance and then reopen the pattern file in the derived template instance.

NOTE:

These columns contain values only after you enter a valid path in the **Patterns Path** field on the **Configuration** tab, page 23.

You can create a copy of existing pattern file or delete it.

- To create a copy of pattern file, right click on the required pattern file and click **Duplicate**. A message is displayed with the pattern file name having a suffix "_Copy.xml". Clicking **OK** will create a copy of pattern file which is then displayed in the grid.
- To delete a pattern file, right click on the required pattern file and click **Delete**. A message will be displayed asking for confirmation, click **Yes** to delete.

Pattern Editor

When you click **View** or **Edit**, the pattern editor will display the content of pattern file in the **Patterns** tab itself.

NOTE: You can click on  to change read-only mode to edit mode.

The pattern editor has the following sections:

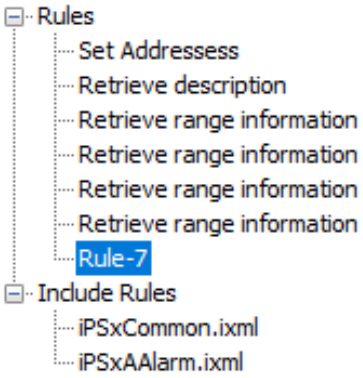
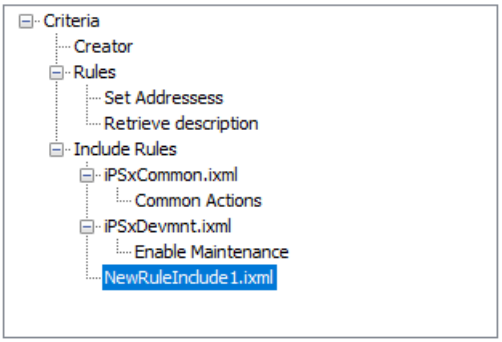
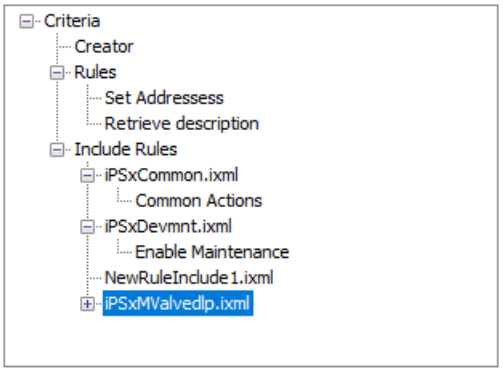
Overview

Section		Description
Template		Displays the name of the pattern file.
Other Details	**Version**	Displays the version of the pattern file.
	Created using	Displays the **TagName** used in the XSY file, prefix of which is used in pattern variable creation.
	Last Modified	Displays the most recent date and time stamp of the modified pattern file.
	Prefix	Displays the number of characters of **TagName** that has to be used for pattern variable creation.

Description

Displays the description of the different options in the **Pattern Editor** when the mouse pointer is placed on them.

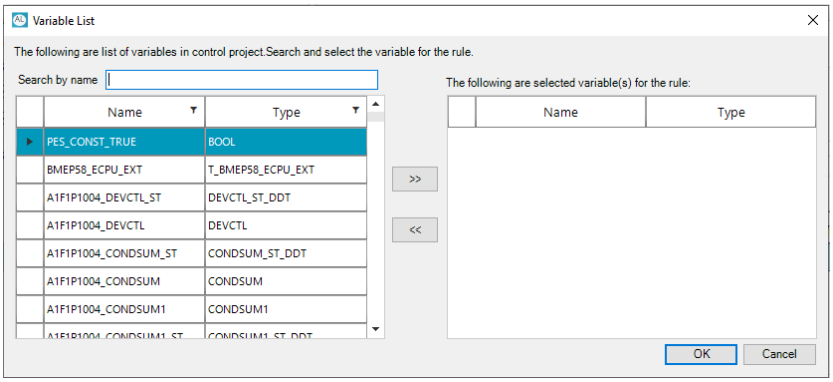
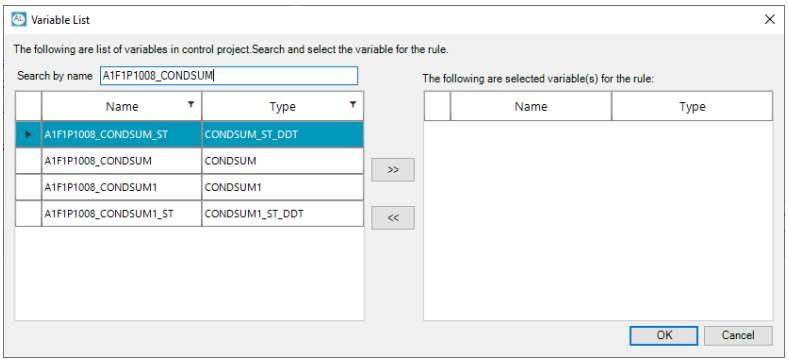
Pattern Rules

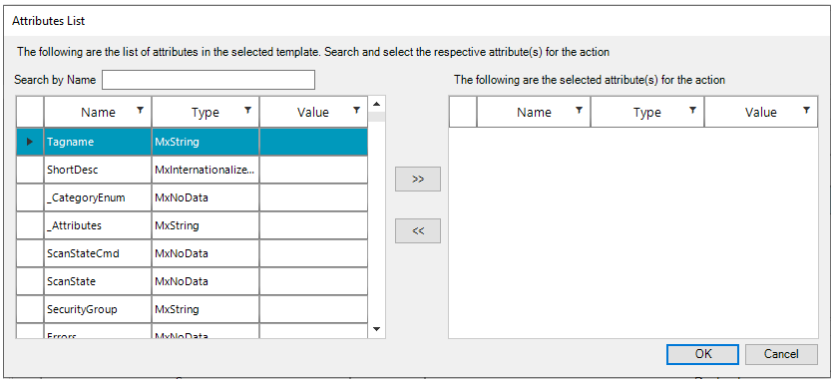
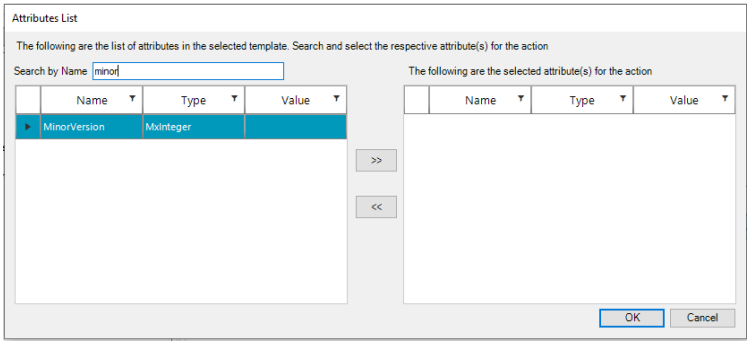
Section	Description
Add Rule	Click **Add Rule** to add a new rule in the pattern file. The newly added rule will be displayed under the **Rules** node.  When you click on rule, the associated details will be displayed in the **Rule Details** section. You can also perform the following actions: • **Duplicate**: To create a copy of the existing rule, right click on the rule and click **Duplicate**. • **Delete**: To delete the existing rule, right click on the rule and click **Delete**. A message will be displayed asking for confirmation, click **Yes** to delete. For more details, refer to Pattern Files, page 47.
Add Include Rule	Click **Add Include Rule** to add a new include rule in the pattern file. A pop-up will be displayed asking to choose if the rule has to be added as **New** or **Existing**. • **New**: This option asks you to provide a name for the rule and then the rule with the given name is created as illustrated in the graphic below.  • **Existing**: This option allows you to select an existing **Include Rule**rule from the respective location it is saved in your system. The selected rule is added as illustrated in the graphic below.  When you click on include rule, the associated details will be displayed in the **Rule Details** section.

Section	Description
	You can also perform the following actions: • **Duplicate**: To create a copy of the existing include rule, right click on the rule and click **Duplicate**. • **Delete**: To delete the existing include rule, right click on the rule and click **Delete**. A message will be displayed asking for confirmation, click **Yes** to delete. For more details, refer to Pattern Files, page 47.
Criteria	Expand **Criteria** to view • **Creator:** This will display the creation rule of the pattern file. The creation rule supports Token based Asset creation and identification. See Token Mechanism, page 31 for more details. • **Rules:** This will display the existing rules of the pattern file. • **Include Rules:** This will display the list of existing include rules of the pattern file.

Rule Details

Section		Description
Rule Name		Allows you to provide a unique name for the rule.
Settings	**Enable the rule during Browse Control Project**	Enable this option to activate and consider the rule during **Browse Control Project**. By default, this option is enabled.
	No Negate	• Enabled: If the criteria for the rule element is not satisfied, all the elements and the rules are executed in the opposite way. • Disabled: If the criteria for the rule element is not satisfied, the elements and the rules are not executed. By default, this option is enabled.
Select variable from control project to identify *** **NOTE:** *** refers to the template name.	**Variables** tab	This tab displays the following details: • The first column is a read-only field • **Variable Name**: This column displays the variable name. • **Variable Type**: This column displays the variable type.
		You can add a new row or a sub element in the grid:

Section		Description
		• **Select Vaiables**: Click on Select Variables to display the list of variables for the selected template.  ◦ You can either type the variable name in the **Search** bar or scroll down to find the respective attribute. ◦ Select the variable and click >> to add it to the list of selected variables. ◦ Select the variable and click << to remove it from the list of the selected variables. **NOTE:** The variables are listed based on its **Name** and **Type**  • **Add variable**: Click button located at the top right corner of the variable tab to add a new row to the grid. Enter the name and type of variable. • **Add sub element**: Click button located at the top right corner of the variable tab to add a sub element to the existing variable. Enter the **Attribute Name**, **Condition Name** and **Value**.
		You can perform the following actions on the variables grid or sub elements grid: • **Copy**: To copy the selected row from the grid, right click on the row and click **Copy**. • **Paste**: To paste the copied row, right click on the row and click **Paste**. • **Edit**: To edit the values of the selected row in the grid, right click on the row and click **Edit**. • **Clear Value**: To clear the existing values of the selected row in the grid, right click on the row and click **Clear Value**. • **Delete Row**: To delete the row from the grid, right click on the row and click **Delete**. A message will be displayed asking for confirmation, click **Yes** to delete.
Select variable from control project to identify *** **NOTE:** *** refers to the template name.	**Action** tab	This tab displays the following details: • **Source**: ◦ **Control Expert Variable:**: – **Name**: This column displays the attribute name of the control expert variable. – **Value**: This column displays the value of the control expert variable.

Section		Description
		• **Action** : This column is a pull-down menu which allows you to select the type of action as either **Action Set** or **Action Retrieve**. ◦ **Action Set**: For this option the source is not applicable since there is a direct value to be set and there are no values to be fetched from the **Control Expert Variable** and fulfilled in **System Platform Attribute**. ◦ **Action Retrieve**: This option will retrieve the value from the **Control Expert Variable** for a particular attribute **Name** and fulfill the respective values in the **System Platform Attribute**. **NOTE:** For more details refer , page 34 • **Destination**: ◦ **System Platform Attribute**: – **Name**: This column displays the name of the system platform attribute. – **Type**: This column displays the type of system platform attribute. – **Contained Name**: This column represents the Contained Name of the instance. Contained name is the name of the parent instance. – **Value**: This column displays the value of the system platform attribute.
		You can add a new row in the grid: • **Select Attributes**: Click on Select Attributes to display the list of attributes for the selected template.  ◦ You can either type the attribute in the **Search** bar or scroll down to find the respective attribute. ◦ Select the attribute and click >> to add it to the list of selected attributes. ◦ Select the attribute and click << to remove it from the list of the selected attributes. **NOTE:** ◦ The attributes are listed based on its **Name**, **Type** and **Value**  ◦ It should be ensured that the right services are enabled so that the correct outputs are obtained for the objects. • **Add new Action**: Click button located at the top right corner of the variable tab to add a new row to the grid. Enter the name and type of variable.

Section		Description
		You can perform the following actions in the grid: • **Copy**: To copy the selected row from the grid, right click on the row and click **Copy**. • **Paste**: To paste the copied row, right click on the row and click **Paste**. • **Edit**: To edit the values of the selected row in the grid, right click on the row and click **Edit**. • **Clear Value**: To clear the existing values of the selected row in the grid, right click on the row and click **Clear Value**. • **Delete Row**: To delete the row from the grid, right click on the row and click **Delete**. A message will be displayed asking for confirmation, click **Yes** to delete.

NOTE: You can perform the following actions on the header rows of **Variables** and **Actions**: tabs.

- **Sort Ascending**: This option allows you to arrange the **Variables** or **Actions** in ascending order.
- **Sort Descending**: This option allows you to arrange the **Variables** or **Actions** in descending order.
- **Clear Sorting**: This option is enabled only if you have selected either **Sort Ascending** or **Sort Descending**.
- **Conditional Formatting**: This options allows you to format the different cells of the column based on their values.
- **Column Chooser**: This option will open another pop-up window which will display the hidden columns which can be dragged and dropped back to the grid.
- **Hide Column**: This option allows you to hide the selected column.
- **Pinned State**: There are three options available:
  - **Unpin Column**: This option allows you to move the column to its default position.
  - **Pin at Left**:This option allows you to move the column to the left
  - **Pin at Right**:This option allows you to move the column to the right
- **Best Fit**: This option will resize all the columns depending on their contents.

Save Changes

Click  to save the changes you made in the pattern editor. A message will be displayed asking for confirmation, click **Yes** to save. All the changes made in the pattern editor will be saved to the pattern files.

Token Mechanism

Token mechanism allows the user to configure in pattern for detection of multiple variable for single asset using Pattern editor. Once the pattern is configured, clicking on **Browse Control Project** will identify the assets within same variable based on input from variable file and the token which matches with the pattern file. The table below with an example explains the identification of assets using the token mechanism.

Step	Action
1	The variable from the .xsy file is Variable name =`FI1001_AINPUT1_AI4_AINPUT1_ST` TypeName = `AINPUT1_ST_DDT`
2	The criteria in the pattern file would be

	<CriterionFound Id="1"> <Value>%%_%%_%%_AINPUT1_ST</Value> </CriterionFound> <CriterionLike Id="2"> <Subelement name="">> <VariableAttribute name="typeName" value="AINPUT1_ST_DDT"/> </Subelement> </CriterionLike>
3	If the variable satisfies the criteria from the pattern then the tokens %1% = FI1001 %2% = AINPUT1 %3% = AI4
4	With the use of the token values in the previous step the asset is identified from the **Actions** in the Creation Rule. <Actions> <ActionCreate Id="1">> <Value>%3%</Value> </ActionCreate> </Actions>
5	Assets are identified based on the input provided in **ActionCreate** value. The asset of the above variable is **AI4** which is determined by fetching the value of **%3%** from the Token.

The various scenarios for action create, local and global tokens are explained below.

1. **Action Create**: The **Action Create** in the pattern file is modified based on the scenarios mentioned in the table below. The inputs considered for the creation rule are:

   Name =`FI1001_AINPUT1_ST`

   TypeName = `AINPUT1_ST_DDT`

Scenario	Pattern Condition		Object ID
	Action Create	Criterion Found	
Prefix as a pre-defined text.	`Test_%1%`	`%%_AINPUT1_ST`	`Test_FI1001`
Suffix as a pre-defined text.	`%1%_Test`	`%%_AINPUT1_ST`	`FI1001_Test`
Pre-defined text for both prefix and suffix.	`Test_%1%_Test`	`%%_AINPUT1_ST`	`Test_FI1001_Test`
Only the prefix as a pre-defined text (With attribute filled in generated instance)	`Test_%1%`	`%%_AINPUT1_ST`	`Test_FI1001`

2. **Local Tokens**: The inputs considered for the creation rule in the below mentioned scenarios are:

   Name =`FI1001_ONE_AINPUT1_ST`

   TypeName = `AINPUT1_ST_DDT`

Scenario	Pattern Condition		Object ID
	Tag Name	Value	
Support multiple tokens in	**CriterionFound**	`%%_%%_AINPUT1_ST`	`FI1001_ONE`

	ActionCreate	%1%_%2%	
Support multiple tokens in typename *** of **CriterionLike** of CreationRule (*** The typename considered in this case is TEST_ST_DDT	**Variable Attribute**	%%_ST_%%	Test_DDT
Support tokens found in **CriterionFound** to be used in **CriterionLike** of CreationRule	**Variable Attribute**	%2%_ST_%%	FI1001_ AINPUT1_DDT
Support tokens along with prefix or suffix in **Action create** of CreationRule	**ActionCreate**	%1%_test%2%_%3%	FI1001_ testAINPUT1_ DDT

3. **Global Token**: The table below explains the support of global tokens for the regular rules.

Scenario		Input		Pattern Condition		Object ID
		Name	**TypeName**	**Tag Name**	**Value**	
Support multiple tokens in	**Criterion-Found**	FI1001_ ONE_ AINPUT1	AINPUT1	**Criter-ion-Found**	%Tagname %_%#2%	FI1001_ ONE
	Criterion-Like	FI1001_ ONE_ AINPUT1	AINPUT1	**Variable Attrib-ute**	%#2%	
	ActionSet	FI1001_ ONE_ AINPUT1_ ST_CFGW	AINPUT1_ ST_DDT	**Action-Set**	%Date-Source%% Tagname% _AIN-PUT1_ST. PV%#3%	FI1001_ ONE
	ActionRe-trieve	FI1001_ ONE_ AINPUT1	AINPUT1_ ST_DDT	**Variable Attrib-ute**	%#2%	FI1001_ ONE
Support global tokens along with local tokens of regular rule		**Name**: FI1001_ONE_ AINPUT1 **TypeName**: AINPUT1_ST_ DDT **Attribute Name**: AINPUT1_ RANGE		**Variable Attrib-ute**	%#2%_%1%	FI1001_ ONE

Object Wizard Support

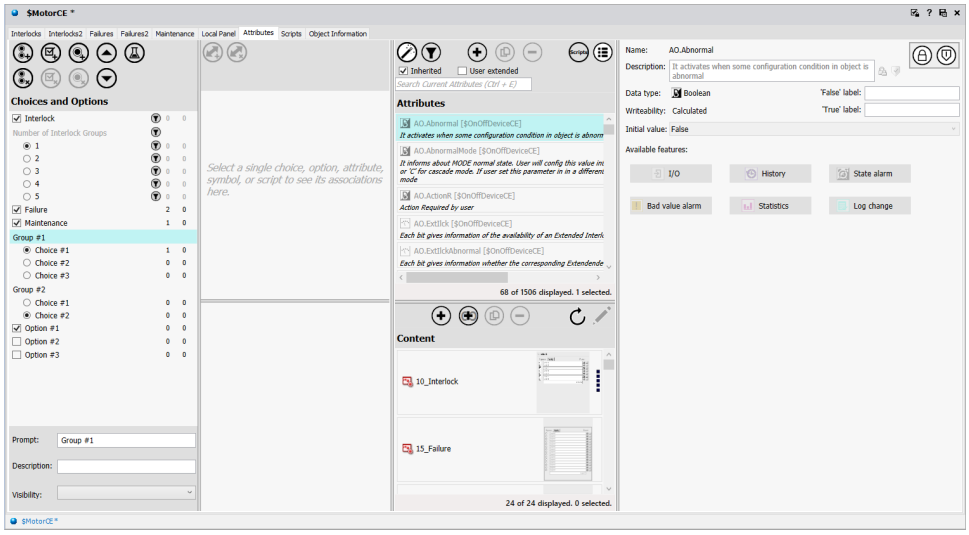
Follow the steps in the table below to view the **Object Wizard** supported asset

Step	Action
1	In the **Configuration** tab and add a Variable file having multiple variables with one asset existing in galaxy with different wizard configuration.
2	Navigate to the **Patterns** tab and edit the Patterns in the **Patterns editor**.
3	Add the **Options** and **Choices** for the selected rule (For more details see, page 34) and click in the **Patterns** tab to save the changes.
4	Navigate to the **Generation** tab and click **Browse Control Project**.
5	Select the assets and generate them.
6	Open the generated asset to view the wizard configuration.

Object Wizard Editor

Object Wizard Editor is a user interface to configure the assets from the templates. To open the**Object Wizard Editor**

- Open the template in the **Object Editor**
- The template by default will open in **Interlocks Tab**, hence select **Attributes Tab** and open it.



The **Object Wizard** consists of two components the **Choice Groups** and **Options**.

- **Choice Group** consists of a prompt and a set of two or more possible responses (**Choice Group**).You can add or remove **Choice Group** by clicking on or . It contains a minimum of two **Choices**. By default the first **Choice** is selected which can be changed manually.
  - **Choice** is an element of **Choice Group** and not more than one **Choice** can be selected at a time. You can add or remove **Choices** by clicking on or respectively.
- **Option** represents a binary choice (true if check box is checked and false if unchecked) and are not mutually exclusive. Unlike **Choices** there is no limitation on the number of **Options** that can be selected. You can add or remove **Options** by clicking on or respectively.

Click to save and close the object editor once the necessary modifications of **Choices** and **Options** are complete.

Data Grid Services

The table may sometimes display multiple rows for the same **Template ID** or **Version**. When there is more than one pattern for the **Template ID**, the highest corresponding **Version** is selected by default.

You can select one row at a time for each **Template ID**. When make the selection, any other pattern that uses that specific **Template ID** is deselected.

When you select a column heading, the table is resorted according to the column function. (By default, the table is arranged alphabetically according to the **Pattern File** name.)

These shortcuts are available when you select multiple rows in the **Pattern File** table:

- Press the space bar to toggle the check mark in the **Selected** column for all selected rows.
- Press the *S* key to check the box in the **Selected** column for all selected rows.
- Press the *U* key to uncheck the box in the **Selected** column for all selected rows.

Pattern Creation Prerequisites

Adhere to these prerequisites before you create a new pattern:

Instance Creation Create an instance of a ASP template and use attributes with syntax that conforms to these examples: • **Instance:** `AnalogInput1_1` • **Attribute:** `ST.STW` • **Syntax:** `Address format with respect to protocol.`
Required Files These files are required for creating, updating, or validating patterns. Add these files to the folder in which new patterns are created or updated: • `PACConnectorIncludeSchema.xml` • `PACConnectorSchema.xml` **NOTE:** By default, these files are located in this installation folder: **AssetLink > GPL Patterns**

Generation Tab

Introduction

Use the features on the **Generation** tab to browse variables in the configured control project file to find matches for all selected patterns

NOTE: By default, the **Generation** tab is available only when valid configuration data is configured and saved on the **Configuration** tab, page 19.

Overview

Generate and update ASP objects:

Stage	Description
1	Browse the configured control project variables.
2	Select ASP AppObjects for processing.
3	Create and update ASP objects. The browsed list in the **Generation** tab is selectable. Only selected objects instances are created and updated.

Browse Variables

Click the **Browse Control Project** button to start the process of exploring variables in the control project to find those that match the modeling that is described in the currently described patterns. Then the tool automatically explores Galaxy to find the appropriate ASP AppObjects to create, update, or re-create.

This table shows the multiple combinations for which AssetLink impacts the generation of instances:

XSY / XML	Pattern	ASP Template	Expected Behavior
Asset available - with 5 variables	Pattern match for all 5 variables	Available	Tool will detect this object to create.
Asset available - with 5 variables	Pattern is available for matching all 5 variables	Not Available	Tool will not find this object and hence will not be listed in the Generation Tab. This ignored object (if it is HMI variable), it will be available in log file exist in the following location: ""C:\ProgramData \Schneider Electric\Ecostruxure Control Expert - Asset Link""
Asset available - with 5 variables	No match with the pattern available for all 5 variables	Available	Tool will not find this object and hence will not be listed in the Generation Tab. This ignored object (if it is HMI variable), it will be available in log file exist in the following location: ""C:\ProgramData \Schneider Electric\Ecostruxure Control Expert - Asset Link\Logs \BrowseControlProject_Log_ datetimestamp.txt""
Asset available - with 5 variables	Pattern matches with 3 variables out of 5 variable	Available	Tool will not find this object and hence will not be listed in the Generation Tab. This ignored object (if it is HMI variable), it will be available in log file exist in the following location: ""C:\ProgramData \Schneider Electric\Ecostruxure Control Expert - Asset Link\Logs \BrowseControlProject_Log_ datetimestamp.txt""
Asset available - with 5 variables	Pattern has criteria for 2 variables out of 5 and this 2 variable matches	Available	Tool will detect this object to create.
Asset available - with 5 variables	Pattern has criteria for only 2 variables, But variable type name does not match.	Available	Tool will not find this object and hence will not be listed in the Generation Tab. This ignored object (if it is HMI variable), it will be available in log file exist in the following location: ""C:\ProgramData \Schneider Electric\Ecostruxure Control Expert - Asset Link\Logs \BrowseControlProject_Log_ datetimestamp.txt""
Asset available - with 5 variables	Pattern does not exist	Available	Tool will not find this object and hence will not be listed in the Generation Tab. This ignored object (if it is HMI variable), it will be available in log file exist in the following location: ""C:\ProgramData \Schneider Electric\Ecostruxure Control Expert - Asset Link\Logs \BrowseControlProject_Log_ datetimestamp.txt""

Check Changes

When the **Check Changes** check box is selected, it finds the ASP AppObjects that are potentially updated. Otherwise, all ASP AppObjects that already existing with a match to variables in the control project variables are proposed as candidates to be updated. Early in the engineering process, when most objects are changing, it could be convenient to leave this box empty (no checkmark) to run the process faster.

After even small changes are performed (for instance, in production time), we recommend that you select this box to clearly identify the changes and any impact they have on the supervisory application.

This function creates a list of selectable objects that are found when it scans the source file and shows the **Action** that applies to the ASP object (**Create**, **Update**, **To be Resolved**, **Re-Create**, or **No Action**).

Considerations:

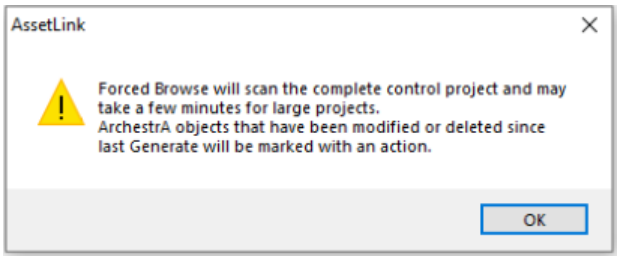
- The progress bar on the **Generation** tab indicates the patterns that remain to be checked and the ASP AppObject that is searched in Galaxy.
- When you select a column heading, the table is resorted according to the column function. (By default, the table is arranged alphabetically according to the **Object ID**, **Template ID**, and **Action**.
- It is possible to display more than one row for the same ASP AppObject if more than one pattern matches the same object.

  NOTE: User needs to ensure RTU file is updated while updating the control variable file.

Forced Browse

The **Forced Browse** allows you to recover any manually deleted AssetLink generated object in the ArchestrA IDE.

When the **Forced Browse** check box is selected, it will display the following message:



Click **OK** and then click **Browse Control Project** button to browse the control project.

Considerations:

- If **Forced Browse** check box is not selected, then the **Browse Control Project** will only display the ASP AppObject which is in **Create**, **Re-Create**, or **Update** status.
- If **Forced Browse** check box is not selected and configuration parameter is modified, then the **Browse Control Project** will display the ASP AppObjects as a difference in settings.

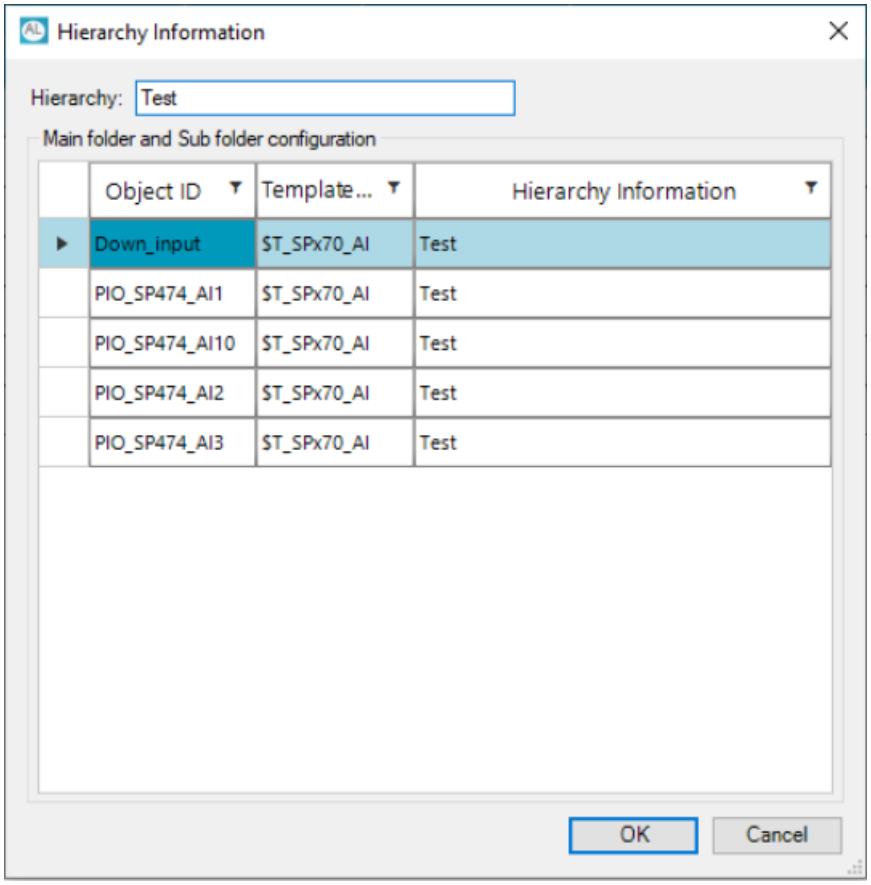
  Only the immediate browse operation will display this change, two consecutive browse operation will not display the ASP APPObject with difference in settings.

  In this scenario, we suggest you to enable **Forced Browse** and then **Browse Control Project**. Also, **Select All Candidates** and **Generate**.
- If **Forced Browse** check box is selected, then the **Browse Control Project** will only display all the ASP AppObjects which is in **Create**, **Re-Create**, **Update**, and **No Action** status.

NOTE: If there is any configuration change, then all the ASP AppObjects will be displayed.

Configure Hierarchy

Clicking on Hierarchy will open a dialog box using which the location can be set for the selected objects.



- **Hierarchy**: It is an editable field allowing the user to enter the Asset location for all the selected rows.
- **Object ID, Template ID, Hierarchy Information**: This is a read-only field displaying the rows which are selected in the generation table.

  NOTE: Clicking on OK will update the **Hierarchy Information** in the generation table for the selected rows.

Data Grid Services

When the table is populated with current proposals to create, update, or re-create ASP AppObjects, you can select or deselect them as required:

- You can select a single row with the same value as the one in the **Object ID** column.
- Click the **Select All Candidates** button to select all instances. If more than one action is suggested for the same **Object ID** row, the **Update** action is implemented by default.

These shortcuts are available when you select multiple rows in the **Generation** tab table:

- Press the space bar to toggle the check mark in the **Selected** column for all selected rows.
- Press the *S* key to select the box in the **Selected** column for all selected rows.

- Press the *U* key to deselect the box in the **Selected** column for all selected rows.

Selecting an asset will auto select the respective area. This is applicable if you have area information available in the **Generation** tab table.

NOTE: Auto Select of an area will happen only if the area is not deployed.

Table Columns

These columns appear on the **Generation** tab:

Column	Description
Select	Select the check box in this column to select the instance in the corresponding row.
Object ID	This column reports the name of the ASP object that will be created, updated, or re-created.
Template ID	This column reports the name of the ASP template that will be applied.
Action	This table reports the action that applies to the ASP object: • **Create:** If the ASP AppObject does not already exist in Galaxy, it is created. • **Update:** Update a ASP AppObject in Galaxy if it is derived from the same ASP template that is defined in the pattern that matches that object. • **Re-Create:** Sometimes, a ASP object in Galaxy is derived from a ASP template that is different than the one defined in the pattern that matches the object. In such cases you can use the **Re-Create** action to delete the original object and create one that is based on the appropriate template. • **No Action:** When a ASP object in Galaxy is derived from the template that is defined in the pattern that matched that object, the **No Action** confirms that the object does not change. • **To be Resolved:** If any conflict has occurred in the asset name (comes from control project file), **To be Resolved** is displayed. To resolve the conflict, click **Resolve** button, page 44. • **Rename**: When the Process Expert folder is renamed then the **Action** column for that folder is updated as “Update”, see, page 45 **NOTE:** The above options are enabled only when the browsing of a control project discovers at least one instance to generate.
Result	This column reports the result of action that applies to the ASP object.
Description	This column describes the objects that are found in Galaxy.
Proposed Area	This column reports the area in which the respective asset will be assigned.
Area	This column reports the Assigned Area that corresponds to the ASP AppObject (if the object is found in Galaxy).
Container	This table reports the container of the ASP AppObject if the object is found in Galaxy and is contained in another ASP AppObject.
Hierarchical Name	For ASP AppObjects that are found in Galaxy, this table reports the hierarchical object name in this format: `<Container>.<ContainedName>`
Derived From	This column reports the ASP template that was used to create the object if the object is found in Galaxy. Use this column to identify the template that was previously used to propose the re-creation of the ASP AppObject.
Hierarchy Information	This column provides the Asset location. The Asset location can be manually entered for each of the objects separately or the user can use the **Hierarchy** option to provide the same information for multiple objects. **NOTE:** User need to ensure the hierarchy information provided here is same as what is configured in the Telemetry Server.

Buttons

These buttons appear on the **Generation** tab:

Button	Description
Select All Candidates	Click this button to select all objects that require processing (**Create**, **Update**, **Re-create**) and those that do not (**Unchanged**). The default actions are taken into consideration, what means that if the same object could be updated or re-created, only the row related to update is selected.
Unselect All	Click this button to deselect all objects for processing.
Resolve	Click this button to resolve the conflicts of objects and folders. For more details, refer to resolve conflicts, page 44 section.

Check Boxes

These check boxes appear on the **Generation** tab:

Check Box	Description
Select Create	Click this button to add to the list of selected objects for which the **Create** action is proposed.
Select Re-Create	Click this button to add to the list of selected objects for which the **Re-Create** action is proposed.
Select Update	Click this button to add to the list of selected objects for which the **Update** action is proposed.
Select Removed	Click this button to add to the list of selected objects for which the **Removed** action is proposed. **NOTE:** This option is not supported.
Select No Action	Click this button to add to the list of selected objects for which the **Unchanged** action is proposed.

NOTE: You can select (check) multiple options at one time.

Generate Objects

Use these buttons to stop or start the generation process:

Button	Description
Generate Objects	Click this button to process the selected rows and update Galaxy accordingly. • **Generate Objects** button will be enabled only if at least one selected candidate available in the data grid. • **Generate Objects** button will be in disabled state while generation is in progress. **NOTE:** Accordingly, a summary of the number of selected objects per action appears.
Stop Generation	Click this button to stop the generation process. • **Stop Generation** button will be enabled only when you trigger the **Generate Objects**. • **Stop Generation** button will be enabled only if more than 10 selected candidates start to generate. • **Stop Generation** button will be in disabled state while stop generation is in progress.

NOTE: After the object is generated, uncheck all generated objects.

The status for each processed object appears in the **Result** column:

Action	Status	Result
Select Create	**OK**	The object was created successfully.

Action	Status	Result
	NOK	The object was not created.
Select Re-Create	**OK**	The object was created successfully.
	NOK	The object was not created.
Select Update	**OK**	The object was updated successfully.
	NOK	The object was not updated.
Select No Action	**OK**	The object was updated successfully.
	NOK	The object was not updated.
Stop Generation	**OK**	The object was not created.
	NOK	The object was not updated.

An on-screen message reports the number of selected objects.

The **Message Details** dialog box confirms that the ArchestrA log records these events:

- The bulk process starts.
- AppObjects are being managed.
- The bulk process concludes.

NOTE:

- The **Message Details** dialog box categorizes events related to warning messages and detected errors.
- You can manage these events from the Log Viewer in the ArchestrA System Management Console.
- Sometimes, an area gets generated in a different location than expected and this information is not displayed in the **Generation** tab. This generated area location will be available in the log file (*C:\ProgramData \Schneider Electric\Ecostruxure Control Expert - AssetLink\Logs*).
- If the object generation gets stopped and the ArchestrA IDE has restarted, there could be some objects left checked out and this is not identified by the AssetLink tool. Check-in if any objects are checked out and generate again.
- If the ArchestrA IDE has restarted during the process of object generation, there might be some objects left checked-out and this is not identified by the AssetLink tool. Check-in if any objects are checked-out and generate again.
- If Archestra IDE closes in-between the generation process, it is suggested to restart the Archestra services and reopen Asset Link and then proceed with the generation process.

Monitor Tab

Overview

The **Monitor** tab allows you to notify for any modifications in the **Control Project** file (.xsy) (such as file replaced or edited) and (.xml) file while it is in use based on the time interval.

Configuration

Configure the following settings:

Settings	Description
Enable	Select this to enable the monitoring feature. Default: Enabled
Disable	Select this to disable the monitoring feature.
Time Interval	Set the time interval between 1 to 1440 in minutes. Default time: 15 minutes

After configuring, Click **Save** to activate the monitoring feature.

When any modification is detected in the:

- **Control Project** file, it will be notified in the **Generation** tab as **Modified version of Control Project file is detected**.
- **XML** file, it will be notified in the **Generation** tab as **Modified Version of Plant Hierarchy file is detected**.
- **Control Project** file and **XML** file, it will be notified in the **Generation** tab as **Modified Version of Control Project and Plant Hierarchy file is detected**.

Click **Browse Control Project** to update the control project variables.

Working with ASP Plant Model

Creating Plant Model in ASP

Overview

This chapter describes the working of AssetLink tool for creation of Plant Model in ASP using the data defined in the EcoStruxure™ Process Expert Plant Model for AVEVA System Platform.

You can perform actions such as **Create**, page 44, **Move/Update**, page 44, and **Resolve Conflicts**, page 44 of objects based on the hierarchy provided in the . xml file.

For configuration of plant model, refer to Plant Model Configuration, page 42.

Plant Model Configuration

ASP Plant Model Configuration

This section describes how to create Plant Model in ASP based on tha data which is defined in the EcoStruxure™ Process Expert.

For details about the work flow of ArchestrA IDE, refer to the work flow, page 15 chapter.

Configuration Tab

Configure the following settings:

Section	Parameter	Description
Galaxy Settings	**Source**	Select **Process Expert** (*.xsy* files) from the pull-down menu.
	Protocol	For details, refer to configuration, page 19 chapter.
	Device Name	
	DIO Name	
	Scan Group	
	OI Address Reference	

Section	Parameter	Description
	Optional Prefix of the AppObject Tagname	
Patterns Settings	**Patterns Path**	Enter the path to the .xml pattern files that AssetLink applies. For example: *C:\ProgramData\Schneider Electric\Ecostruxure Control Expert - Asset Link\GPL Patterns* **NOTE:** The common pattern schema .xsd files has to be in this same folder. These patterns are scanned each time and this path changes after you reopen AssetLink or press the **Refresh Patterns** button in the **Patterns** tab.
	Pattern Project	This is the .xsy file that is used to create and update the pattern. For example: *C:\ProgramData\Schneider Electric\Ecostruxure Control Expert - Asset Link\Control Expert Variable File \PlantModelXSY.xsy* **NOTE:** The plant model .xsy and .xml files has to be in this same folder.
Control Project Settings	**Control Project**	This field contains the full name of the file that AssetLink scans for variables. For example: *C:\ProgramData\Schneider Electric\Ecostruxure Control Expert - Asset Link\Control Expert Variable File \PlantModelXSY.xsy* For more details, refer to control project settings, page 23.

Patterns Tab

For details about this tab, refer to the patterns tab, page 24 chapter.

Generation Tab

For more details about this tab, refer to the generation tab, page 35 chapter.

These columns appear on the **Generation** tab:

Column	Description
Select	For details, refer to configuration, page 39 table columns.
Object ID	This column displays the asset and area name which is configured in the input file (.xsy). • If you have resolved asset name generated in ASP AppObject, then the **Object ID** will display “Original Name(Resolved Name)”. • If you have resolved area name generated in ASP AppObject, then the **Object ID** will display “Original Name(–>)” to refer the resolve area name in **Area** column.
Template ID	This column displays the respective derived type of template for asset and area. For system name comes from EcoStruxure™ Process Expert and it is of type root area. For child hierarchy, it is of type area ($aPSxAreaGP).
Action	For details, refer to configuration, page 39 table columns.
Result	
Description	
Proposed Area	This column reports the hierarchical area name that corresponds to the EcoStruxure™ Process Expert .xml file. **NOTE:** • Areas will be modified based on hierarchy given in the .xml file. • Areas which are modified with latest hierarchy will be displayed in the **Model** tab of the ASP. • If area of object already deployed, it will un-deploy and regenerates the area.

Column	Description
Area	For details, refer to configuration, page 39 table columns.
Container	
Hierarchical Name	
Derived From	

NOTE: The **Area** object listed here are of derived template type $aPSxAreaGP. If any area object created by the user using derived template type $PSxAreaGP, then it will not be managed by AssetLink. You have to select the respective area along with the asset in order to generate the asset at the right location.

Create

You can create the objects based on the hierarchy of EcoStruxure™ Process Expert Plant Model provided in the .xml file.

To create:

Step	Action
1	In the **Generation** tab, select the objects with status displayed as **Create** in the **Action** column.
2	Click **Generate Objects**, page 40. **Result**: The **Result** column displays the status as **Created** when the objects are successfully created.
3	Click **Browse Control Project** to update the objects list. **Result**: The **Proposed Area** column displays the hierarchy of created objects with corresponds to the Process Expert Plant Model .xml file and **Area** column displays the ASP hierarchy. **NOTE:** • For the object of type “Area”, the **Proposed Area** column will be empty. • If the object of type “Area” is selected to generate, the **Proposed Area** column displays the hierarchy of area. • If only instance is selected without respective area, only instance will be created under unassigned area.

You can see the created objects in the **Model** tab of the ASP.

Move/Update

You can move the objects to other folders in the **Model** tab of the ASP hierarchy.

To move the objects:

Step	Action
1	In the **Model** tab, drag and drop the objects to the required folder.
2	In the **Generation** tab, click **Browse Control Project** to update the objects list. **Result**: The **Area** column displays the ASP hierarchy of the newly moved object and **Proposed Area** column displays the hierarchy corresponds to the Process Expert Plant Model .xml file.

To retain the hierarchy of the Process Expert Plant Model in the **Model** tab of ASP, click **Generate Objects**.

Resolve Conflicts

You can resolve the conflicts of objects and folders. Conflict occurs due to identical names of objects and folders, incorrect naming format, or exceeding 32 characters limit.

The status of conflict is displayed as **To be Resolved** in the **Action** column.

NOTE:

- You cannot generate the objects for those in **To be Resolved** status.
- If renamed object has a conflict, it will be detected as **To be Resolved** status. Post resolve, it will process as a new object with Create action.

Step	Action
1	In the **Generation** tab, click **Resolve** button. **Result**: The **Resolve** window opens.
2	In **Select** column, select the respective object or click **Select All**.
3	Click **OK**. **Result**: The selected conflicts are resolved and the status is displayed as **Resolved** in **Action** column and the respective item will be selected in the **Generation** tab.

Resolve Window

These columns appears on the **Resolve** window:

Column	Description
Select	Select the check box in this column to select the object to resolve the conflict.
Object ID	This column displays the name of the ASP object that is in conflict status.
Template ID	This column displays the name of the ASP template that is in conflict status.
Type	This column displays the type as **Asset** or **Area**.
Proposed Name	This column displays the proposed name of object. You can edit and provide your own name.
Description	This column displays the description of conflict of the respective objects.
Select All	Select this check box to select all the objects to resolve the conflict.
Unselect All	Select this check box to unselect all the objects.

Rename

In the System Platform you can handle the EPE folder rename by renaming the respective areas.

Step	Action
1	Navigate to the **Configuration** tab and provide the folder path for the Variables and the Patterns files.
2	Navigate to the **Generation** tab and click **Browse Control Project**.
3	The **Action** column of the renamed folder is updated as “Update” and the **Result** column will display "**Diff: Renamed**". **NOTE:** • If the unique identifier occurs same for two different objects from the**Process Expert** source, then, the object will not detect as "**Diff: Renamed**". In this case, "**Create**" action will be performed. • If the respective object is in deployed state, the **Result** column will display “**Source object rename/ location change detected. Undeploy application object to continue**”.
4	Select the folder and click **Generate objects**.
5	The selected folder is renamed and this can be viewed in the Modal tab of the System Platform.

The actual **Root Area** coming from EPE is named as “Root” and converted into system name when you execute using Asset Link. This will be notified to you in the **Result** column of the **Generation** tab.

Step	Action
1	Navigate to the **Configuration** tab and provide the folder path for the Variables and the Patterns files.
2	Navigate to the **Generation** tab and click **Browse Control Project**.
3	If the **Action** column of the **Root Area** which has been replaced with System Name displays “Create”, then **Result** column will display “Root from source is replaced with System Name”.

Pattern Files

Introduction to Syntax and Structure

Pattern File Syntax

Introduction

The source pattern file is an exported control project file that serves these functions:

- The file defines the method for extracting control project variables that correspond to specific asset types.
- The file defines the reprocessing this information to create or update the corresponding ASP AppObjects through the use of ASP templates.
- The file determines the presence of asset instances in the control project.
- The file indicates the ASP template to be used in the supervision and control of such asset types from the ASP.
- The file satisfies the ASP AppObject attributes according to data found in the control project.

Structure

This table describes the main components of the pattern file:

Component	Description
Pattern Header	The header describes the main data of the pattern, like the ASP template that is associated with it, the ASP AppObject that was used to create it, and the date and time of the most recent modification.
List of Rules	This list describes the exploration of control project variables to determine the presence of an asset that needs to be projected as a ASP AppObject in the ASP Galaxy. The list determines the data to be retrieved from control project variables (not only ASP AppObject IO references, but also descriptions, initial values, etc.). The list also satisfies ASP AppObject user-defined attributes so you do not have to enter the same information separately for control and supervisory purposes. Each rule is defined by these components: • **Rule Header:** This header identifies the rule, provides details about the rule's creation, indicates if the rule was manually modified. etc. • **List of Criteria:** These criteria determine the presence of data that satisfy ASP AppObjects. The syntax of the criteria allows you to define naming conventions that are expected from control project variables to determine the presence of the asset and collect data that can be used later from the list of actions that satisfy the ASP AppObject UDAs. • **List of Actions:** The actions in this list are executed when the criteria that is defined for the rule is satisfied. The list also provides the creation of ASP AppObjects and the manner in which they satisfy ASP AppObject UDAs with information found within the criteria.

Syntax Components

This is a simple example of the syntax that is used in the patterns:

```
<variables name="RUB1_DEVCTL_ST" typeName="DEVCTL_ST_DDT">

<comment>RUB1_DEVCTL DDT Comment</comment>

<information name="IsVariableHMI" value="-1"/>

</variables>
```

Below, we describe the components of the syntax (tags, elements, and attributes).

Tag. A tag is a markup construct. The tag content is framed by less-than (<) and greater than (>) signs. Use these tags:

Tag Type	Tag Syntax
start-tag	`<section>`
end-tag	`</section>`
empty-element tag	`<line-break />`

Element. An element is a logical document component that conforms one of these formats:

- The element tag is empty.
- The content of the element is between a start-tag and a matching end-tag. The content itself may contain markup that includes child elements. Examples:
  - `<comment>RUB1_DEVCTL DDT Comment</comment>`
  - `<information name="IsVariableHMI" value="-1"/>`

Attribute. An attribute is a markup construct that consists of a name-value pair within the start-tag or empty-element tag:

- In this example, the attributes `src` and `alt` have the respective values `madonna.jpg` and `Madonna`:

  `<img src="madonna.jpg" alt="Madonna" />`
- In this example, the name of the attribute number carries the value `3`:

  `<step number="3">Connect A to B.</step>`

  NOTE: An .xml attribute can have only one value, and each attribute can appear only once in each element.

XSY File Structure

Syntax Structure

A familiarity with the pattern syntax helps you to evaluate the structure and information in the .xsy file. This is a sample structure from a valid pattern file:

```
<VariablesExchangeFile>
<fileHeader company="Schneider Automation" …></fileHeader>
<contentHeader name="Project" version="0.0.000"></contentHeader>
<dataBlock>
<variables name="" typeName="T_BMEP58_ECPU_EXT">
[…]
</variables>
<variables name="RUB1_DEVCTL" typeName="DEVCTL">
[…]
</variables>
[…]
<variables name="xxxx" typeName="YYYY">
[…]
</variables>
</dataBlock>
</VariablesExchangeFile>
```

Notice that the set of `variables` elements contain two attributes, `name` and `typeName`.

Variables Element

Each `variables` element contains a set of different elements that define more information about the variable:

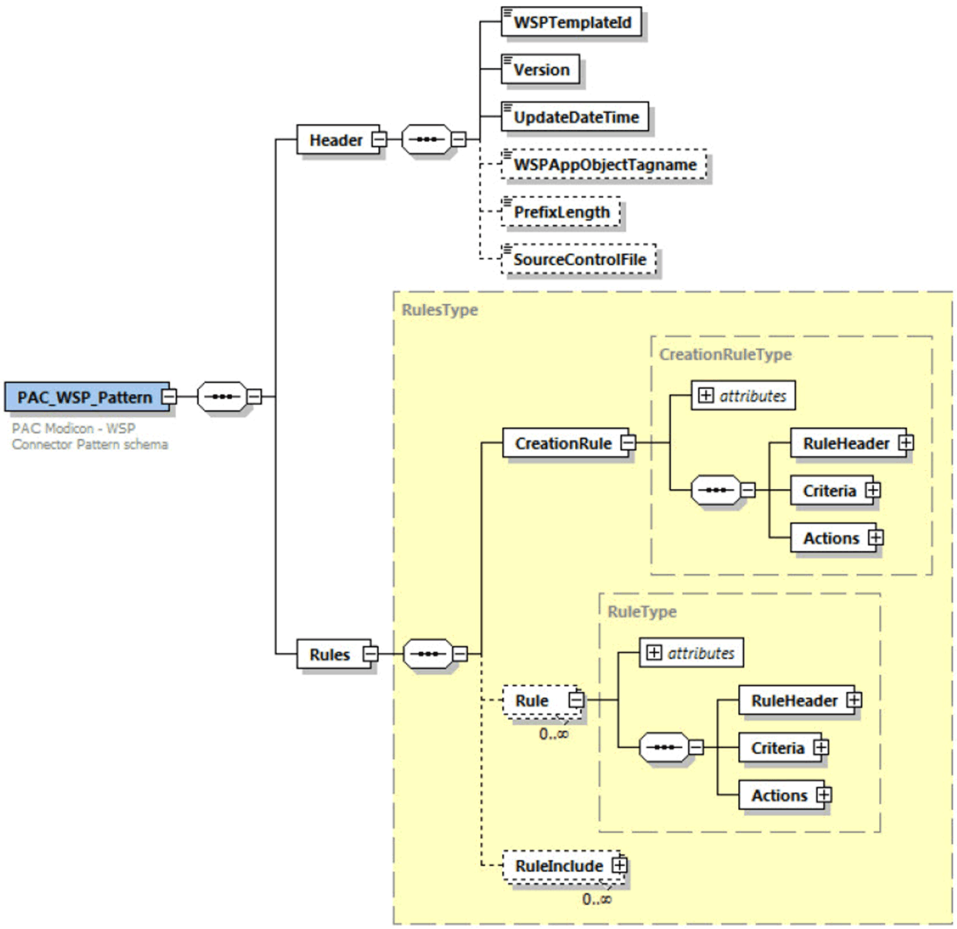
```
<variables name="RUB1_CONDSUM1" typeName="CONDSUM1">
<comment>RUB1 COMMENT</comment>
<instanceElementDesc name="COND01">
<comment>Condition 1 (higher priority) Rub</comment> </instanceElementDesc>
<instanceElementDesc name="REQREARM01">
<comment>Condition 1 requires rearm Rub</comment>
</instanceElementDesc>
<instanceElementDesc name="SAFEPOS01">
<comment>Safe Position 1 Rub</comment>
</instanceElementDesc>
<instanceElementDesc name="BYPASSDIS01">
<comment>Disable Bypass 1 Rub</comment>
</instanceElementDesc>
</variables>
```

Pattern Definition

About Pattern Definitions

Pattern Definition Elements and Sub-elements

Access the pattern definition elements and their sub-elements by expanding any visible element:



The main element (`PAC_ASP_Pattern`) has two child elements:

- Header, page 50
- Rules, page 51

Pattern Definition Header

Introduction

This topic describes the functionality of the elements and sub-elements that you see when you expand (**+**) the `Header` element in the pattern definition flowchart, page 50.

Path: `PAC_ASP_Pattern/Header`

Header Element

The `Header` element contains these sub-elements:

Element	Description
ASPTemplateId	This identifier is the exact name of the ASP Template to be used for generating new instances when this pattern is matched.
Version	This sub-element contains the currently implemented version of the pattern.
UpdateDateTime	This sub-element reports the date and time of the last change to the pattern.
ASPAppObjectTag-name	If the Converter discovered this pattern automatically, this sub-element contains the tag name of the AppObject that was used to create the pattern.
PrefixLength	If the **ASPAppObjectTagname** has a prefix, it is shown in this sub-element.
SourceControlFile	If the Converter discovered this pattern automatically, this sub-element contains the path to the .xsy file that was used to create the pattern.

This is an example of a `Header` element:

```
<Header>
<ASPTemplateId>$aPSxMotor</ASPTemplateId>
<Version>1.0</Version>
<UpdateDateTime>2017-10-18 12:00:00</UpdateDateTime>
<ASPAppObjectTagname>MOTOR1</ASPAppObjectTagname>
<PrefixLength>0</PrefixLength>
<SourceControlFile/>
</Header>
```

Pattern Definition Rules

Introduction

This topic describes the functionality of the elements and sub-elements that you see when you expand (**+**) the `Rules` element in the pattern definition flowchart, page 50.

Path: `PAC_ASP_Pattern/Rules`

The sub-elements of the **Rules** flowchart are discussed below:

- `Rules, page 51`
- `CreationRule, page 52`
- `Rule, page 58`
- `RuleInclude, page 65`

Rules Element

Rules Element

This topic describes the functionality of the elements and sub-elements that you see when you expand (**+**) the `Rules` element in the pattern definition flowchart, page 50 through this path:

`PAC_ASP_Pattern/Rules`

Expand (**+**) the `Rules` child element to see these sub-elements:

Element	Description
CreationRule, page 52	AssetLink constantly attempts to execute the CreationRule element.
Rule, page 58	0 ... *n*
RuleInclude, page 65	0 ... *n*

This is an example of a Rules element:

```
<Rules>

<CreationRule Id="0" Comment="Creator">[…]</CreationRule>

<Rule Id="1" Comment="Set Addressess">[…]</Rule>

<Rule Id="2" Comment="Retrieve Description">[…]</Rule>

<RuleInclude file="iPSxCondsum1.ixml"/>

</Rules>
```

CreationRule

Introduction

This topic describes the functionality of the elements and sub-elements that you see when you expand (**+**) the Rules element in the pattern definition flowchart, page 50 through this path:

PAC_ASP_Pattern/Rules/CreationRule

CreationRule Element

CreationRule element descriptions:

Element(s)		Description
attrib-utes	Id	The attributes for this element provide practical information about the element.
	Comment	
RuleHeader		This element establishes information for the rule.
Criteria		This element contains a set of criteria that are satisfied to execute the rule.
Actions		This element contains actions that are executed when the Criteria are satisfied.

These sub-elements of the CreationRule are described below:

- RuleHeader, page 52
- Criteria, page 53
- CriterionFound, page 53
- CriterionLike, page 54
- Actions, page 57

RuleHeader Element

Follow this path to access this element:

PAC_ASP_Pattern/Rules/CreationRule/RuleHeader

Expand the RuleHeader element to access these elements in the RuleHeaderType area:

- `Auto`: This Boolean determines the way that the rule was created:
  - *true:* The rule is generated automatically through the pattern discovery process.
  - *false:* The user creates the rule manually.
- `Updated`: This Boolean value has these values:
  - *true:* The value is always `true` for manual rules and for automatically generated rules that you can modify.
  - *false:* You can not update the rule with AssetLink.
- `Enabled`: This Boolean controls the application of the rule during bulk processing:
  - `true`: The rule is applied during bulk processing.
  - `false`: The rule is ignored during bulk processing but retained in the pattern file.

This is an example of a `RuleHeader` element:

```
<RuleHeader>

<Auto>true</Auto>

<Updated>false</Updated>

<Enabled>true</Enabled>

</RuleHeader>
```

Criteria Element

Follow this path to access this element:

`PAC_ASP_Pattern/Rules/CreationRule/Criteria`

The `Criteria` element contains a set of criteria that are satisfied for the execution of actions. Every criterion in the `Criteria` element resolves to `true` before the execution of an action.

Expand the `Criteria` element to access these elements in the `CriteriaCreateType` area

- `CriterionFound`: This element tries to locate specific kind of name for variables in the control project.
- `CriterionLike`: For each match found by `CriterionFound`, this element assesses the validity of the match.

The `Criteria` element represents a sequence from 1 to *n* of:

- *1*: `CriterionFound`
- a set (0 to *x*: `CriterionLike`

CriterionFound Element

Follow this path to access this element:

`PAC_ASP_Pattern/Rules/CreationRule/Criteria/CriterionFound`

Expand the `CriterionFound` element to access these elements:

- `attributes`: The `Id` attribute provides practical information about the element.
- `Value`: This element corresponds to the variable name.

This is an example of a CriterionFound element that contains `attribute` and a `Value` sub-element:

```
<CriterionFound Id="1" >

<Value>%%_DEVCTL_ST</Value>

</CriterionFound>
```

The `CriterionFound` element tries to find matches in control project file by searching the `name` attributes that correspond to `variable` elements in the file.

In this example from a control project, the `CriterionFound` element discovers that the type name `DEVCTL_ST_DDT` matches both `MOTOR01_DEVCTL_ST` and `MOTOR02_DEVCTL_ST`:

```
<variables name="MOTOR01_DEVCTL" typeName="DEVCTL">

<comment>MOTOR1</comment>

</variables>

<variables name="MOTOR01_DEVCTL_ST" typeName="DEVCTL_ST_DDT">

<attribute name="IsVariableHMI" value="-1"></attribute>

</variables>

<variables name="MOTOR02_DEVCTL" typeName="DEVCTL">

<comment>MOTOR2</comment>

</variables>

<variables name="MOTOR02_DEVCTL_ST" typeName="DEVCTL_ST_DDT">

<attribute name="IsVariableHMI" value="-1"></attribute>

</variables>
```

CriterionLike Element

Follow this path to access this element:

`PAC_ASP_Pattern/Rules/CreationRule/Criteria/CriterionLike`

Expand the `CriterionLike` element to access these elements:

- `attributes`: The `name` attribute provides practical information about the element.
- `Subelement`: This element corresponds to the components of the `SubelementType` area:
  - `VariableAttribute`: This element is a component of the definition of the search. The criterion is satisfied when all instances of `VariableAttribute` are found. The `VariableAttribute` element searches for a specific value for an attribute inside an element (specified in `Subelement`) in the control project, always inside the `Variables` selected in the `CriterionFound` element. The attributes for this element represent the `name` and `value` of the attribute.
  - `ElementValue`: This element contributes to the search definition. The criterion is satisfied when all instances of `ElementValue` are found. The attribute for this element represents the `value` of the attribute.
  - `Subelement`: This element indicates whether the search takes place in the same variable element or in one of the sub-elements in the control project file (separated by a period [`.`]). This element contains the `VariableAttribute` and `ElementValue` sub-elements.

This is an example of a CriterionLike element that contains `attribute` and a `Value` sub-elements:

```
<CriterionLike Id="2">

<Subelement name="">

<VariableAttribute name="typeName" value="DEVCTL_ST_DDT"></
VariableAttribute>

</Subelement>

</CriterionLike>

<CriterionLike Id="3">

<Subelement name="attribute">

<VariableAttribute name="name" value="IsVariableHMI"></VariableAttribute>

<VariableAttribute name="value" value="-1"></VariableAttribute>

</Subelement>

</CriterionLike>
```

The `CriterionLike` element looks in the control project file for specific values for `variables` elements that were discovered by the `CriterionFound` element.

This is an example of a CriterionLike element in the control project file:

```
<variables name="MOTOR01_DEVCTL" typeName="DEVCTL">

<comment>MOTOR1</comment>

</variables>

<variables name="MOTOR01_DEVCTL_ST" typeName="DEVCTL_ST_DDT">

<attribute name="IsVariableHMI" value="-1"></attribute>

</variables>

<variables name="MOTOR02_DEVCTL_ST" typeName="DEVCTL_ST_DDT">

<attribute name="IsVariableHMI" value="-1"></attribute>

</variables>
```

The recursive `Subelement` is found inside other sub-elements to refine the search when possible. For example, the same functionality can be expressed in two ways. Refer to the two examples that follow.

Example (non-recursive subelement)**:**

```
<CriterionLike Id="2">

<Subelement name="">

<VariableAttribute name="typeName" value="DEVCTL_ST_DDT"></
VariableAttribute>

</Subelement>

</CriterionLike>

<CriterionLike Id="3">

<Subelement name="attribute">

<VariableAttribute name="name" value="IsVariableHMI"></VariableAttribute>

<VariableAttribute name="value" value="-1"></VariableAttribute>

</Subelement>

</CriterionLike>
```

Example (recursive subelement)**:**

```
<CriterionLike Id="2">

<Subelement name="">

<VariableAttribute name="typeName" value="DEVCTL_ST_DDT"></
VariableAttribute>

<Subelement name="attribute">

<VariableAttribute name="name" value="IsVariableHMI"></VariableAttribute>

<VariableAttribute name="value" value="-1"></VariableAttribute>

</Subelement>

</Subelement>

</CriterionLike>
```

The `Criteria` for a `CreationRule` can consist of more than one `CriteriaFound` element, as shown in this example:

```
<Criteria>

<CriterionFound Id="1">

<Value>%%_DEVCTL_ST</Value>

</CriterionFound>

<CriterionLike Id="2">

<Subelement name="">

<VariableAttribute name="typeName" value="DEVCTL_ST_DDT"/>

<Subelement name="attribute">

<VariableAttribute name="name" value="IsVariableHMI"/>

<VariableAttribute name="value" value="-1"/>

</Subelement>

</Subelement>

</CriterionLike>

<CriterionFound Id="4">

<Value>%1%_DEVCTL</Value>

</CriterionFound>

<CriterionLike Id="5">

<Subelement name="">

<VariableAttribute name="typeName" value="DEVCTL"/>

</Subelement>

</CriterionLike>

</Criteria>
```

Notice that `Criteria` in the above example requests two different variables (`_DEVCTL_ST` and _DEVCTL) with the same prefix.

In another example, only the `MOTOR1` object is created because `MOTOR2` does not satisfy the requirements of the second `CriteriaFound` element in the rule:

```
<variables name="MOTOR01_DEVCTL" typeName="DEVCTL">

<comment>MOTOR1</comment>

</variables>

<variables name="MOTOR01_DEVCTL_ST" typeName="DEVCTL_ST_DDT">

<attribute name="IsVariableHMI" value="-1"></attribute>

</variables>

<variables name="MOTOR02_DEVCTL_ST" typeName="DEVCTL_ST_DDT">

<attribute name="IsVariableHMI" value="-1"></attribute>

</variables>
```

This is an example, we can have more than one condition within `CriteriaLike` as shown below.

Type name "`AALARM_CFG_DDT`" and `Comment` value is "0" are two conditions.

```
<Criteria>

<CriterionFound Id="1">

<Value>%Tagname%_AALARM_CFG</Value>

</CriterionFound>

<CriterionLike Id="2">

<Subelement name="">

<VariableAttribute name="typeName" value="AALARM_CFG_DDT"></
VariableAttribute>

<Subelement name="instanceElementDesc">

<VariableAttribute name="name" value="SPHH"></VariableAttribute>

<Subelement name="comment">

<ElementValue value="0"></ElementValue>

</Subelement>

</Subelement>

</Subelement>

</CriterionLike>

</Criteria>
```

Actions Element

Follow this path to access this element:

`PAC_ASP_Pattern/Rules/CreationRule/Actions`

Expand the `Actions` element to access these elements:

- `attributes`: The `Id` attribute provides practical information about the element.
- `Value`: This element corresponds to the variable name.

In a `CreationRule`, the `Actions` element instantiates an object from the template that is defined in `Header.ASPTemplateID`. The instance name is provided in the `Value` element:

```
<Actions>

<ActionCreate Id="1" >

<Value>%1%</Value>

</ActionCreate>

</Actions>
```

The value %1% is the value for the first token (%%) that is satisfied by the CriterionFound element (MOTOR01):

```
<CriterionFound Id="1" >

<Value>%%_DEVCTL_ST</Value>

</CriterionFound>
```

```
<variables name="MOTOR01_DEVCTL_ST" typeName="DEVCTL_ST_DDT">

<attribute name="IsVariableHMI" value="-1"></attribute>

</variables>
```

When the CreationRule is executed, the new instance name is available for reference in other rules as %Tagname%.

Rule

Introduction

This topic describes the functionality of the elements and sub-elements that you see when you expand (**+**) the Rule element in the pattern definition flowchart, page 50 through this path:

PAC_ASP_Pattern/Rules/Rule

This set of 0 to *n* rule elements follows the CreationRule, page 52.

Rule Element

Expand the Rule element to access these sub-elements in the RuleType area:

Element(s)		Description
attributes	Id	The attributes for this element provide practical information about the element.
	Comment	
	NoNegated	
RuleHeader		This element establishes information for the rule.
Criteria		This element contains a set of criteria that are satisfied to execute the rule.
Actions		This element contains actions that are executed, even when the Criteria are not satisfied.

An execution of the CreationRule element creates new instances in ASP. Then the criteria for each Rule element can be satisfied or not, depending on its own defined criteria:

- If the criteria for the Rule element are satisfied, the rule is executed, which activates all of its Actions elements.
- If the criteria for the Rule element are not satisfied while the NoNegated attribute is FALSE, the rule executes in the opposite way and all Actions elements in the opposite way.
- If the criteria for the Rule element are not satisfied while the NoNegated attribute is TRUE, the rule does not execute and no Actions elements are executed.

These sub-elements of the Rule are described below:

- RuleHeader, page 59
- Criteria, page 59
- Actions, page 59
- ActionSet, page 60
- ASPAppObjectAttribute, page 61

RuleHeader

Follow this path to access the sub-elements for the RuleHeader element in the RuleHeaderType area:

PAC_ASP_Pattern/Rules/Rule/RuleHeader

This is an example of a RuleHeader element:

```
<RuleHeader>

<Auto>true</Auto>

<Updated>false</Updated>

<Enabled>true</Enabled>

</RuleHeader>
```

Expand the RuleHeader element to access these elements:

- Auto: This Boolean reports that the rule was created with an automatic process (TRUE) or manually (FALSE).
- Updated: This Boolean indicates that the rule was updated manually after its automatic creation.
- Enabled: This Boolean indicates whether AssetLink analyzes rule or not.

Criteria

Follow this path to access the sub-elements for the Criteria element in the CriteriaType area:

PAC_ASP_Pattern/Rules/Rule/Criteria

The Criteria element is a set of criteria that are satisfied before the Actions element is executed. Each criterion in the element resolve to TRUE before the Actions elements can execute normally.

Expand the Criteria element to access these elements:

- CriterionAlways: This element indicates that the rule is executed under any conditions.
- CriterionFound: This element looks for specific kinds of names for variables in a control project file.
- CriterionLike: For each match found by CriterionFound, this element assesses the validity of the match.

The CriterionFound and CriterionLike elements are equivalent to the definitions for CreationRule.Criteria, but in that case a sequence of only one CriterionFound element and *n* CriterionLike elements can be present.

Actions

Follow this path to access the sub-elements for the Actions element in the ActionsType area:

PAC_ASP_Pattern/Rules/Rule/Actions

The value of the `Actions` element represents the combined number of `ActionSet` and `ActionRetrieve` elements.

Element execution:

- When the required criteria are satisfied, the `Actions` elements execute normally.
- When the required criteria are not fully satisfied, the `Actions` elements execute inversely.

These different executions of the `Actions` element write values to the UDAs of the newly created instance:

Expand the `Actions` element to access these elements:

- `ActionSet, page 60`: This element writes a specific constant value on one UDA.
- `ActionRetrieve, page 61`: This element writes a value that was obtained from the control project in a specific UDA.

ActionSet

Follow this path to access the sub-elements for the ActionSet element in the `ActionSetType` area:

`PAC_ASP_Pattern/Rules/Rule/Actions/ActionSet`

Expand the `ActionSet` element to access these elements:

- `attributes` (`Id`)
- `ASPAppObjectAttribute, page 62`:
  - `attributes` (`type`)
  - `attributes` (`ContainedName`)
- `Value, page 61`

Element execution:

- When the required criteria are satisfied, the `Actions` element writes the value specified in `Action`.
- When the required criteria are not fully satisfied, the `Actions` element writes the opposite value in the UDA (if it exists):
  - Boolean: opposite value
  - integer, double, float: 0
  - string: “”.

This is an example of an `ActionSet` element::

```
<ActionSet Id="0">

<ASPAppObjectAttribute type="Boolean">Config.Ref.Disable</
ASPAppObjectAttribute>

<Value>True</Value>

</ActionSet>



<ActionSet Id="1">

<ASPAppObjectAttribute type="String">DevCtl.St.CFGW.InputSource</
ASPAppObjectAttribute>

<Value>%DataSource%%Tagname%_DEVCTL_ST.CFGW</Value>

</ActionSet>
```

Expand the `ActionSet` element to access these elements:

- `ASPAppObjectAttribute, page 61`
- `Value, page 61`

ASPAppObjectAttribute

Follow this path to access the sub-elements for the ASPAppObjectAttribute element in the `ASPAppObjectAttributeType` area:

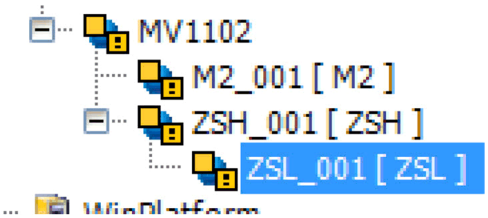
`PAC_ASP_Pattern/Rules/Rule/Actions/ActionSet/ASPAppObjectAttribute`

The `ASPAppObjectAttribute` element defines the UDA of newly created instances in ASP.

Expand the `ASPAppObjectAttribute` element to access these elements:

- `type`: This attribute defines the UDA's datatype (Boolean, integer, string, double, float).
- `ArrayIndex`: This attribute defines the UDA Array Index value.
- `ContainedName`: When present, this element indicates that the UDA is not from the created instance. Instead, the UDA is from a child instance that was created when the template was instantiated. The name is separated by '.' of each `ContainedName` element.

This graphic shows different values assigned to `ContainedName`:



- `M2`: This `ContainedName` refers to UDAs inside `MS_001`.
- `ZSH`: This `ContainedName` refers to UDAs inside `ZSH_001`.
- `ZSH.ZSL`: This `ContainedName` refers to UDAs inside `ZSL_001`.

Value

Follow this path to access the sub-elements for the ASPAppObjectAttribute element in the `ASPAppObjectAttributeType` area:

`PAC_ASP_Pattern/Rules/Rule/Actions/ActionSet/Value`

The `Value` element defines the value that is written to the UDA.

Example:

- `%1%`: This `Value` refers to the first satisfied `%%` that is found in the rule.
- `%Tagname%`: This `Value` refers to the instance name.
- `%DataSource%`: This `Value` refers to the prefix information used for the connection in the ASP.
- composed: `%Tagname%_Text_whatever`:
  - `%Tagname%_Text_whatever`

ActionRetrieve

Follow this path to access the sub-elements for the ASPAppObjectAttribute element in the `ActionRetrieveType` area:

`PAC_ASP_Pattern/Rules/Rule/Actions/ActionRetrieve`

When an `ActionRetrieve` element is executed, a value that was extracted from the variables in the control project file and matched with the required criteria is written to a UDA of the newly created instance.

Element execution:

- When the required criteria are satisfied, the `ActionRetrieve` element writes the value specified in `Action`.
- When the required criteria are not fully satisfied, the `ActionRetrieve` element writes the opposite value in the UDA (if it exists):
  - Boolean: opposite value
  - integer, double, float: 0
  - string: “”.

Expand the `ActionRetrieve` element to access these elements:

- `ASPAppObjectAttribute`, page 62: This element defines the UDA to be written.
- `Subelement`: This element defines new criteria (if any) that is examined in sub-elements of the `variables(xsy)` element to point exactly to the value to be written.
- `GetElementValue`, page 63: Choose this element to define an obtained value that corresponds to an element.
- `GetVariableAttribute`, page 63: Choose this element to define an obtained value that corresponds to an attribute.

This is an example of an `ActionRetrieve` element:

```
<ActionSet Id="0">

<ASPAppObjectAttribute type="Boolean">Config.Ref.Disable</
ASPAppObjectAttribute>

<Value>True</Value>

</ActionSet>



<ActionSet Id="1">

<ASPAppObjectAttribute type="String">DevCtl.St.CFGW.InputSource</
ASPAppObjectAttribute>

<Value>%DataSource%%Tagname%_DEVCTL_ST.CFGW</Value>

</ActionSet>
```

ASPAppObjectAttribute

Follow this path to access the sub-elements for the ASPAppObjectAttribute element in the `ASPAppObjectAttributeType` area:

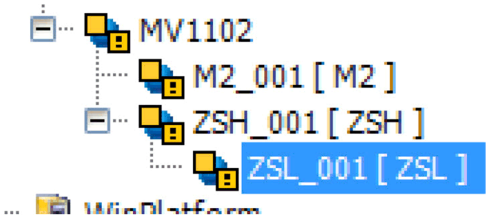
`PAC_ASP_Pattern/Rules/Rule/Actions/ActionRetrieve/ASPAppObjectAttribute`

The `ASPAppObjectAttribute` element defines the UDA of newly created instances in ASP.

Expand the `ASPAppObjectAttribute` element to access these elements:

- `type`: This attribute defines the UDA's datatype (Boolean, integer, string, double, float).
- `ContainedName`: When present, this element indicates that the UDA is not from the created instance. Instead, the UDA is from a child instance that was created when the template was instantiated. The name is separated by '.' of each `ContainedName` element.

This graphic shows different values assigned to `ContainedName`:



- M2: This ContainedName refers to UDAs inside MS_001.
- ZSH: This ContainedName refers to UDAs inside ZSH_001.
- ZSH.ZSL: This ContainedName refers to UDAs inside ZSL_001.

Subelement

Follow this path to access the sub-elements for the Subelement element in the SubelementType area:

PAC_ASP_Pattern/Rules/Rule/Actions/ActionRetrieve/Subelement

Expand the Subelement element to access these elements:

- attribute (name): The name attribute contains a set of VariableAttribute and ElementValue elements.
- VariableAttribute:
  - attribute (name)
  - attribute (value)
- ElementAttribute:
  - attribute (value)

When Subelement is present, it defines new criteria that are applied to sub-elements inside the variables(xsy) element selected in the CriterionFound of the executed Rule:

Subelement: present	Subelement defines the conditions for selecting a specific sub-element in the variables element in the .xsy file in the CriterionFound element of the executed Rule. The one that accomplish all of them is used to obtain the value that is written to the UDA.
Subelement: not present	The selected value is the first one selected from the GetElementValue, page 63 or GetVariableAttribute, page 63 elements.

NOTE: Refer to section 0 CriterionLike for a more complete Subelement description, page 54.

GetElementValue

Follow this path to access the sub-elements for the GetElementValue element in the GetElementAttType area:

PAC_ASP_Pattern/Rules/Rule/Actions/ActionRetrieve/Subelement/GetElementValue

When GetElementValue is defined in the ActionRetrive element, the value that is written to the UDA is the value of the element defined in this subelement attribute.

The path defined in this attribute always starts in the element that is selected by the Subelement that is defined above.

GetVariableAttribute

Follow this path to access the sub-elements for the GetElementValue element in the GetElementAttributeType area:

PAC_ASP_Pattern/Rules/Rule/Actions/ActionRetrieve/Subelement/GetVariableAttribute

Expand the GetVariableAttribute element to access the subelement and attribute attributes. When GetVariableAttribute is defined in the ActionRetrive element, the value that is written to the UDA is the value of the attribute defined in attribute of the element that is defined by the subelement attribute of GetVariableAttribute.

The path defined in this `subelement` always starts in the element that is selected by the `Subelement` that is defined above:

```
<ActionRetrieve Id="1" >

<ASPAppObjectAttribute type="String">ShortDesc</ASPAppObjectAttribute>

<GetElementValue subelement="comment"/>

</ActionRetrieve>


#Variable selected in XSY

<variables name="RUB1_DEVCTL" typeName="DEVCTL">

<comment>RUB1 COMMENT</comment>

</variables>


ShortDesc <- "RUB1 COMMENT"
```

Features of the above example:

- The `ActionRetrive` element does not include a defined `Subelement`, so the selected value points from the `Variables` element itself.
- A `subelement` is defined as `comment` in the `GetElementValue` element (**bold** type), so you have to get the value from the `Variables.comment` element (`instanceElementDesc`).

```
<ActionRetrieve Id="1" >

<ASPAppObjectAttribute type="String">Ilck.Legend1</ASPAppObjectAttribute>

<Subelement name="instanceElementDesc">

<VariableAttribute name="name" value="COND01"></VariableAttribute>

</Subelement>

<GetElementValue subelement="comment"/>

</ActionRetrieve>


#Variable selected in XSY

<variables name="RUB1_CONDSUM1" typeName="CONDSUM1">

<comment>RUB1 COMMENT</comment>

<instanceElementDesc name="COND01" property="PR01">

<comment writeBy="Charles Xavier">Condition 1 (higher priority) Rub</
comment>

</instanceElementDesc>

<instanceElementDesc name="COND02" property="PR01">

<comment writeBy="James Logan">Condition 2 Rub</comment>

</instanceElementDesc>

</variables>

Ilck.Legend1 <- "Condition 1 (higher priority) Rub"
```

Features of the above example:

- The `ActionRetrive` element includes a defined `Subelement`, so the selected value points from an `instanceElementDesc` in the `Variables` element.
- There are criteria that has to satisfy the `instanceElementDesc` sub-element that includes a `name` attribute with the value `COND01`.

- In the `GetElementValue` element, `subelement` is defined as `comment`, so you have to get the value of the `Variables.comment` element (`instanceElementDesc`).
- This `variables` element has two `instanceElementDesc` subelements, but only one satisfies the requirements of the `name` attribute with the value `COND01`

```
<ActionRetrieve Id="1" >

<ASPAppObjectAttribute type="String">Ilck.Legend1</ASPAppObjectAttribute>

<Subelement name="instanceElementDesc">

<VariableAttribute name="property" value="PR01"></VariableAttribute>

<VariableAttribute name="name" value="COND02"></VariableAttribute>

</Subelement>

<GetVariableAttribute subelement="comment" attribute="writeBy"/>

</ActionRetrieve>


#Variable selected in XSY

<variables name="RUB1_CONDSUM1" typeName="CONDSUM1">

<comment>RUB1 COMMENT</comment>

<instanceElementDesc name="COND01" property="PR01">

<comment writeBy="Charles Xavier">Condition 1 (higher priority) Rub</
comment>

</instanceElementDesc>

<instanceElementDesc name="COND02" property="PR01">

<comment writeBy="James Logan">Condition 2 Rub</comment>

</instanceElementDesc>

</variables>
```

Features of the above example:

- The `ActionRetrive` element includes a defined `Subelement`, so the selected value points from an `instanceElementDesc` in the `Variables` element.
- There are criteria that has to satisfy the `instanceElementDesc` sub-element that includes attributes with these values:
  - `property`: `PR01`
  - `name`: `COND02`
- The`GetVariableAttribute` element includes these definitions:
  - `subelement`: `comment`
  - `attribute`: `writeby` (Get the value for the `writeBy` attribute from the `instanceElementDesc.comment` element.)
- These `variables` element has two `instanceElementDesc` subelements, but only one satisfies the requirements of the `name` attribute with the value `COND02`.

RuleInclude

Introduction

Follow this path to access the RuleInclude element in the `RuleIncludeType` area:

`PAC_ASP_Pattern/Rules/Rule/RuleInclude`

This set of 0 to *n* rule elements follows the Rules.

iXML File

The RuleInclude element one attribute named `file`. The `file` attribute contains the name of an iXML file.

The iXML files contains rules that are added to the rules of the current pattern file.

The and `RuleInclude` elements follow the same rules for patterns. The use of these files facilitates the sharing of rules among multiple patterns.

GPL Patterns for AssetLink

General Purpose Library

Introduction

This section describes the General Purpose Library (GPL) that is available for use in PAC Modicon and ASP.

Use the information in this section to use the set of pre-built AssetLink patterns in the GPL. Such patterns are built to connect the 2018 R3 release these GPL libraries for both PAC Modicon and ASP environments.

Only the GPL patterns for the more commonly used asset types are described in this section.

Prerequisites

Readers of this section should have a working familiarity with these products:

- PAC Modicon
- Control Expert (formerly Unity Pro)
- AVEVA System Platform (ASP)
- EcoStruxure™ Control Expert AssetLink
- General Purpose Library for Modicon and ASP

Project Engineering

Control Projects

The pre-built set of GPL patterns expect a concrete naming convention to allow AssetLink to recognize the assets that are automated in the control project. Such a naming convention, therefore, is applied to variables when the control project code is written. This means you do not have to adjust the patterns later.

Of course, you can apply a different naming convention, but in that case the pattern files require editing to conform to any adjustments. This case might also require the manual refinement of the sophisticated GPL patterns because all rules for cannot necessarily be found during the pattern discovery process.

Supervisory Projects

The GPL patterns use the GPL application templates that start with this prefix: `$aPSx`

The GPL patterns are in the folders that are configured from the AssetLink objects in the ASP Galaxy.

AssetLink supports these GPL pattern types:

- *XML:* Use XML pattern files when you model asset types.

- *iXML:* Some pattern files reference these iXML rule include files, page 65 to define rules that are used by multiple patterns to minimize maintenance. These files are often used to model the optional asset services (interlocks, detected failures, local panels, etc.) from the GPL.
- *XSD:* XSD files contain the XML schema of the pattern and rule include files.

  NOTE: Refer to the introduction to syntax and structure, page 47.

Copy such files to the appropriate folders.

NOTE: You can use the same folder location for multiple connector objects.

Scope and Naming Conventions

Introduction

The tables below describe these concepts:

- the scope of the current GPL patterns release
- the naming conventions that are applied to the control project code
- the relationship to the supervisory application

Common Asset Services

This table describes the contents of the include rules files (.iXML) that are referenced by pattern files:

Include Rules File	Service	DFB Type	Variable Naming Convention	Variable Data Type	Comments
iPSxAAlarm	Analog Alarms	AALARM	<obj>_AALARM_CFG	AALARM_CFG_DDT	All alarm setpoints are enabled. Customize other types of combinations as the actual setpoints to be managed cannot be inferred control project variables.
iPSxAoutputlp	Analog Output Local Panel	AOUTPUTLP	<obj>_AOUTPUTLP _ST	AOUTPUTLP_ST_DDT	
iPSxCommon	n/a	n/a	n/a	n/a	This Include Rules File is included in all patterns as it disables the auto-binding of references of the ASP AppObjects by setting up the **Config.Ref.Disable** UDA.
iPSxCondsum	Failure Conditions Summary	CONDSUM	<obj>_CONDSUM	CONDSUM	The detected failure condition descriptions are retrieved from the DFB CONDSUM pin descriptions **COND##**.
			<obj>_CONDSUM_ST	CONDSUM_ST_DDT	
iPSxCondsum1Ana-log	Interlock Conditions Summary (for Analog Assets)	CONDSUM1	<obj>_CONDSUM1	CONDSUM1	The interlock condition descriptions are retrieved from the DFB CONDSUM1 pin descriptions **COND##**.
			<obj>_CONDSUM1_ST	CONDSUM1_ST_DDT	
iPSxCondsum1-Discr	Interlock Conditions Summary (for Discrete Assets)	CONDSUM1	<obj>_CONDSUM1	CONDSUM1	The interlock condition descriptions are retrieved from the DFB CONDSUM1 pin descriptions **COND##**.
			<obj>_CONDSUM1_ST	CONDSUM1_ST_DDT	

Include Rules File	Service	DFB Type	Variable Naming Convention	Variable Data Type	Comments
iPSxCondsumF2	Failure Conditions Summary (for 2 direction/speed)	CONDSUM	<obj>_RC_ CONDSUM	CONDSUM	The detected failure condition descriptions are retrieved from the DFB CONDSUM pin descriptions **COND##**.
			<obj>_RC_ CONDSUM_ST	CONDSUM_ST_ DDT	
iPSxCondsumFC	Failure Conditions Summary (for sequences and equipment modules)	CONDSUM	<obj>_FC_ CONDSUM	CONDSUM	The detected failure condition descriptions are retrieved from the DFB CONDSUM pin descriptions **COND##**.
			<obj>_FC_ CONDSUM_ST	CONDSUM_ST_ DDT	
iPSxCondsumIC	Initial Conditions Summary (for sequences and equipment modules)	CONDSUM	<obj>_IC_ CONDSUM	CONDSUM	The initial condition descriptions are retrieved from the DFB CONDSUM pin descriptions **COND##**.
			<obj>_IC_ CONDSUM_ST	CONDSUM_ST_ DDT	
iPSxDevlp	On/Off Device Local Panel	DEVLP	<obj>_DEVLP_ST	DEVLP_ST_DDT	
iPSxDiPSx-Devlpvmnt	On/Off Device Maintenance	DEVMNT	<obj>_DEVMNT_ST	DEVMNT_ST_DDT	
iPSxMotor2lp	On/Off Device Local Panel	MOTOR2LP	<obj>_MOTOR2LP_ ST	MOTOR2LP_ST_ DDT	
iPSxMValvedlp	Discrete Motorized Valve Local Panel	MVALVEDLP	<obj>_ MVALVEDLP_ST	MVALVEDLP_ST_ DDT	

Asset Types

NOTE: The variables that appear in **bold type** in the table below are required from the AppObject creation rule in the pattern.

Process Patterns

This table describes the contents of the pattern files (.XML) and the asset types for which they help to automate their supervisory responsibilities:

Pattern File (& ref. to specific Include Rules)	Asset Type	DFB Type	Variable Naming Convention[1]	Variable Data Type	ASP Template	Comments
aPSxAlarmSummary *iPSxCondsum	Alarms Summa-ry	DINPUT DA-LARM	<obj>_DINPUT	DINPUT	$aPSxA-larmSumma-ry	The description of the Asset is retrieved from the description of the DFB.
			<obj>_DI_ST	DINPUT_ST_DDT		
			<obj>_DALARM_ ST	DALARM_ST_ DDT		
aPSxAnalogInput *iPSxAAlarm	Analog Input (legacy)	AINPUT	<obj>_AINPUT	AINPUT	$aPSxAna-logInput	The description of the Asset is retrieved from the description of the DFB. From the constant Range variable: Engineering Units is extracted from the comment of the field HI if fulfilled. Numeric Format is extracted from the comment of the field LO if fulfilled. The High and Low range is retrieved from the HI and LO field initial values.
			<obj>_AINPUT_ ST	AINPUT_ST_DDT		
			<obj>_AINPUT_ CFG	AINPUT_CFG_ DDT		
			<obj>_PV_RNG	RANGE_DDT		
aPSxAnalogInput1 *iPSxAAlarm	Analog Input	AIN-PUT1	<obj>_AINPUT1	AINPUT1	$aPSxAna-logInput1	The description of the Asset is retrieved from the description of the DFB.
			<obj>_AINPUT1_ ST	AINPUT1_ST_ DDT		

Pattern File (& ref. to specific Include Rules)	Asset Type	DFB Type	Variable Naming Convention[1]	Variable Data Type	ASP Template	Comments
			<obj>_AINPUT1_CFG	AINPUT1_CFG_DDT		From the constant Range variable: Engineering Units is extracted from the comment of the field HI if fulfilled. Numeric Format is extracted from the comment of the field LO if fulfilled.
			<obj>_PV_RNG	RANGE_DDT		
aPSxAnalogOutput *iPSxCondsum *iPSxAoutputlp	Analog Output	AOUT-PUT	<obj>_AOUTPUT	AOUTPUT	$aPSxAna-logOutput	The description of the Asset is retrieved from the description of the DFB. From the constant Range variable: Engineering Units is extracted from the comment of the field HI if fulfilled. Numeric Format is extracted from the comment of the field LO if fulfilled. The High and Low range is retrieved from the HI and LO field initial values.
			<obj>_AOUTPUT_ST	AOUTPUT_ST_DDT		
			<obj>_AOUTPUT_CFG	AOUTPUT_CFG_DDT		
			<obj>_PV_RNG	RANGE_DDT		
aPSxSelect1	Analog Selector	ASE-LECT1	<obj>_ASELECT1	ASELECT1	$aPSxASe-lect1	The description of the Asset is retrieved from the description of the DFB. The description of each analog input is retrieved from the descriptions of the pins 'SP#'
			<obj>_ASELECT1_ST	ASELECT1_ST_DDT		
			<obj>_ASELECT1_CFG	ASELECT1_CFG_DDT		
aPSxControlValve *iPSxCondsum1Ana-log	Control Valve	CVALVE	<obj>_CVALVE	CVALVE	$aPSxCon-trolValve	The description of the Asset is retrieved from the description of the DFB.
			<obj>_CVALVE_ST	CVALVE_ST_DDT		
			<obj>_CVALVE_CFG	CVALVE _CFG_DDT		
			<obj>_CVALVELP_ST	CVALVELP_ST_DDT		
aPSxDigitalInput *iPSxDevmnt	Digital Input	DINPUT	<obj>_DINPUT	DINPUT	$aPSxDigi-talInput	The description of the Asset is retrieved from the description of the DFB.
			<obj>_DINPUT_ST	DINPUT_ST_DDT		
aPSxDigitalOutput *iPSxCondsum1Discr *iPSxDevmnt	Digital Output	DOUT-PUT	<obj>_DOUTPUT	DOUTPUT	$aPSxDigi-talOutput	The description of the Asset is retrieved from the description of the DFB.
			<obj>_DOUTPUT_ST	DOUTPUT_ST_DDT		
aPSxDiscreteSP	Discrete Setpoint	n/a	**<obj>_DISCRETESP**	BOOL	$aPSxDis-creteSP	The description of the Asset is retrieved from the description of the variable.
aPSxDualOutput-Valve *iPSxCondsum1Discr *iPSxCondsum *iPSxDevmnt	Dual Output Valve	DVALVE	<obj>_DVALVE	DVALVE	$aPSxDua-lOutputValve	The description of the Asset is retrieved from the description of the DFB.
			<obj>_DVALVE_ST	DVALVE_ST_DDT		
aPSxDurationSP	Duration Setpoint	n/a	**<obj>_DURATIONSP**	TIME	$aPSxDura-tionSP	The description of the Asset is retrieved from the description of the variable.
aPSxEquipmentMod-ule *iPSxCondsumIC	Equip-ment Module	EMCTL	<obj>_EMCTL	EMCTL	$aPSxE-quipment-Module	The description of the Asset is retrieved from the description of the DFB.
			<obj>_EMCTL_ST	EMCTL_ST_DDT		

Pattern File (& ref. to specific Include Rules)	Asset Type	DFB Type	Variable Naming Convention[1]	Variable Data Type	ASP Template	Comments
*iPSxCondsumFC			**<obj>_EMCTL_ CFG**	EMCTL_CFG_ DDT		
			<obj>_IC_ CONDSUM_ST	CONDSUM_ST_ DDT		
			<obj>_FC_ CONDSUM_ST	CONDSUM_ST_ DDT		
aPSxHandValve	Hand Valve	HVALVE	<obj>_HVALVE	HVALVE	$aPSxHand-Valve	The description of the Asset is retrieved from the description of the DFB.
			<obj>_HVALVE_ ST	HVALVE_ST_ DDT		
aPSxIBPhase *iPSxCondsumIC *iPSxCondsumFC	InBatch Phase	IB-PHASE	<obj>_IBPHASE	IBPHASE	$aPSxIB-Phase	The description of the Asset is retrieved from the description of the DFB. Maximum one of the IBPAR05, IBPAR10 and IBPAR16 variables are expected at same time. The description of each parameter is retrieved from the description of the related field 'IP##' and 'OP##'.
			<obj>_IBPHASE_ ST	IBPHASE_ST_ DDT		
			<obj>_IBPHASE_ CFG	IBPHASE_CFG_ DDT		
			<obj>_IBPAR05_ ST	IBPAR05_ST_ DDT		
			<obj>_IBPAR10_ ST	IBPAR10_ST_ DDT		
			<obj>_IBPAR16_ ST	IBPAR16_ST_ DDT		
aPSxIC	Extended Initial Condition	CON-DSUM	<obj>_IC	CONDSUM		The description of the Asset is retrieved from the description of the DFB. Once the AppObjects are generated, the user has to contain them in their related AppObject Container manually. The detected failure condition descriptions are retrieved from the DFB CONDSUM pin descriptions 'COND##'.
			<obj>_IC_ST	CONDSUM_ST_ DDT		
aPSxIlck	Extended Interlock Condition	CON-DSUM	<obj>_ILCK	CONDSUM		The description of the Asset is retrieved from the description of the DFB. Once the AppObjects are generated, the user has to contain them in their related AppObject Container manually. The detected failure condition descriptions are retrieved from the DFB CONDSUM pin descriptions 'COND##'.
			<obj>_ILCK_ST	CONDSUM_ST_ DDT		
aPSxIMCtl *iPSxCondsum1Analog	Internal Model Controller	IMCTL	<obj>_IMCTL	IMCTL	$aPSxIMCtl	The description of the Asset is retrieved from the description of the DFB. From the constant Range variables: Engineering Units is extracted from the comment of the field HI if fulfilled. Numeric Format is extracted from the comment of the field LO if fulfilled. The High and Low range is retrieved from the HI and LO field initial values.
			<obj>_IMCTL_ST	IMCTL_ST_DDT		
			<obj>_IMCTL_ DDT	IMCTL_CFG_ DDT		
			<obj>_PV_RNG	RANGE_DDT		
			<obj>_OP_RNG	RANGE_DDT		
aPSxIntegerSP	Integer Setpoint	n/a	**<obj>_ DINTEGERSP**	INT	$aPSxInte-gerSP	The description of the Asset is retrieved from the description of the variable.

Pattern File (& ref. to specific Include Rules)	Asset Type	DFB Type	Variable Naming Convention[1]	Variable Data Type	ASP Template	Comments
aPSxLeadLagCtl *iPSxCondsum1Analog	Lead Lag Controller	LDLGCTL	<obj>_LDLGCTL	LDLGCTL	$aPSxLeadLagCtl	The description of the Asset is retrieved from the description of the DFB. From the constant Range variables: Engineering Units is extracted from the comment of the field HI if fulfilled. Numeric Format is extracted from the comment of the field LO if fulfilled. The High and Low range is retrieved from the HI and LO field initial values.
			<obj>_LDLGCTL_ST	LDLGCTL_ST_DDT		
			<obj>_LDLGCTL_DDT	LDLGCTL_CFG_DDT		
			<obj>_SP_RNG	RANGE_DDT		
			<obj>_OP_RNG	RANGE_DDT		
aPSxMAnalogInput1	Multiple Analog Input	MAINPUT1	<obj>_MAINPUT1	MAINPUT1	$aPSxMAnalogInput1	The description of the Asset is retrieved from the description of the DFB. From the constant Range variable: Engineering Units is extracted from the comment of the field HI if fulfilled. Numeric Format is extracted from the comment of the field LO if fulfilled.
			<obj>_MAINPUT1_ST	MAINPUT1_ST_DDT		
			<obj>_MAINPUT1_CFG	MAINPUT1_CFG_DDT		
			<obj>_PV_RNG	RANGE_DDT		
aPSxMessageBox	Message Box	MSGBOX	<obj>_MSGBOX	MSGBOX	$aPSxMessageBox	The description of the Asset is retrieved from the description of the DFB.
			<obj>_MSGBOX_ST	MSGBOX_ST_DDT		
			<obj>_MSGBOX_CFG	MSGBOX_CFG_DDT		
aPSxMotor *iPSxCondsum1Discr *iPSxCondsum *iPSxDevmnt *iPSxDevlp	On/Off Motor	DEVCTL	<obj>_DEVCTL	DEVCTL	$aPSxMotor	The description of the Asset is retrieved from the description of the DFB.
			<obj>_DEVCTL_ST	DEVCTL_ST_DDT		
aPSxMotor2 *iPSxCondsum1Discr *iPSxCondsum *iPSxCondsumF2 *iPSxDevmnt *iPSxMotor2lp	Motor 2 speeds/ directions	MOTOR2	<obj>_MOTOR2	MOTOR2	$aPSxMotor2	The description of the Asset is retrieved from the description of the DFB.
			<obj>_MOTOR2_ST	MOTOR2_ST_DDT		
aPSxMotorizedValve *iPSxCondsum1Discr	Motorized Valve	MVALVE MVALVELP MOTOR2 CONDSUM DINPUT AINPUT1	<obj>_MVALVE	MVALVE	$aPSxMotorizedValve	This Pattern manages the related Container and its Contained objects. The description of the Asset is retrieved from the description of the DFBs 'MVALVE', 'MOTOR2', 'DINPUT' and 'AINPUT1'. The detected failure condition descriptions are retrieved from the DFBs CONDSUM pin descriptions 'COND##'. Engineering Units is extracted from the comment of the field HI if fulfilled.
			<obj>_MVALVE_ST	MVALVE_ST_DDT		
			<obj>_MVALVE_CFG	MVALVE_CFG_DDT		
			<obj>_MVALVELP_ST	MVALVELP_ST_DDT		
			<obj>_M2_ST	MOTOR2_ST_DDT		
			<obj>_M2_FC	CONDSUM		
			<obj>_M2_FC_ST	CONDSUM_ST_DDT		
			<obj>_M2_RC	CONDSUM		

Pattern File (& ref. to specific Include Rules)	Asset Type	DFB Type	Variable Naming Convention[1]	Variable Data Type	ASP Template	Comments
			<obj>_M2_RC_ST	CONDSUM_ST_DDT		Numeric Format is extracted from the comment of the field LO if fulfilled.
			<obj>_M2_MNT_ST	DEVMNT_ST_DDT		
			<obj>_ZSH_ST	DINPUT_ST_DDT		
			<obj>_ZSL_ST	DINPUT_ST_DDT		
			<obj>_AI_ST	AINPUT1_ST_DDT		
			<obj>_AI_CFG	AINPUT1_CFG_DDT		
			<obj>_PV_RNG	RANGE_DDT		
aPSxMotorized-ValveD *iPSxCondsum1Discr *iPSxMValvedlp	Discrete Motor-ized Valve	MVALV-ED MVAL-VELP MO-TOR2 CON-DSUM DINPUT AIN-PUT1	<obj>_MVALVED	MVALVED	$aPSxMo-torizedValve	This Pattern manages the related Container and its Contained objects. The description of the Asset is retrieved from the description of the DFBs 'MVALVED', 'MOTOR2' and 'DINPUT'. The detected failure condition descriptions are retrieved from the DFBs CONDSUM pin descriptions 'COND##'.
			<obj>_MVALVED_ST	MVALVE_ST_DDT		
			<obj>_MVALVED_CFG	MVALVE_CFG_DDT		
			<obj>_M2_ST	MOTOR2_ST_DDT		
			<obj>_M2_FC	CONDSUM		
			<obj>_M2_FC_ST	CONDSUM_ST_DDT		
			<obj>_M2_RC	CONDSUM		
			<obj>_M2_RC_ST	CONDSUM_ST_DDT		
			<obj>_M2_MNT_ST	DEVMNT_ST_DDT		
			<obj>_ZSH_ST	DINPUT_ST_DDT		
			<obj>_ZSL_ST	DINPUT_ST_DDT		
aPSxMotorVS *iPSxCondsum1Discr *iPSxCondsum *iPSxDevmnt	Motor Variable Speed	SDDEV-CTL	<obj>_SDDEVCTL	SDDEVCTL	$aPSxMo-torVS	The description of the Asset is retrieved from the description of the DFB. From the constant Range variables: Engineering Units is extracted from the comment of the field HI if fulfilled. Numeric Format is extracted from the comment of the field LO if fulfilled. The High and Low range is retrieved from the HI and LO field initial values.
			<obj>_SDDEVCTL_ST	SDDEVCTL_ST_DDT		
			<obj>_SDDEVCTL_CFG	SDDEVCTL_CFG_DDT		
			<obj>_PV_RNG	RANGE_DDT		
			<obj>_OP_RNG	RANGE_DDT		
aPSxPID *iPSxCondsum1Ana-log	PID Control-ler	PIDCTL	<obj>_PIDCTL	PIDCTL	$aPSxIMCtl	The description of the Asset is retrieved from the description of the DFB. From the constant Range variables: Engineering Units is extracted from the comment of the field HI if fulfilled. Numeric Format is extracted from the comment of the field LO if fulfilled. The High and Low range is retrieved from the HI and LO field initial values.
			<obj>_PIDCTL_ST	PIDCTL_ST_DDT		
			<obj>_PIDCTL_DDT	PIDCTL_CFG_DDT		
			<obj>_PV_RNG	RANGE_DDT		
			<obj>_OP_RNG	RANGE_DDT		

Pattern File (& ref. to specific Include Rules)	Asset Type	DFB Type	Variable Naming Convention[1]	Variable Data Type	ASP Template	Comments
aPSxPIDMultiplexer	PID Multi-plexer	PID-MUX	<obj>_PIDMUX	PIDMUX	$aPSxPID-Multiplexer	The description of the Asset is retrieved from the description of the DFB and the description of the multiplexed PIDs is retrieved from the variables PIDMUX_ST_CFG.
			<obj>_PIDMUX_ ST	PIDMUX_ST_ DDT		
			<obj>_PIDMUX_ CFG1	PIDMUX_CFG_ DDT		
			<obj>_PIDMUX_ CFG2	PIDMUX_CFG_ DDT		
			<obj>_OP_RNG	RANGE_DDT		
aPSxPWM *iPSxCondsum1Ana-log	Pulse Width Modula-tor	PWMC-TL	<obj>_PWMCTL	PWMCTL	$aPSxPWM	The description of the Asset is retrieved from the description of the DFB.
			<obj>_PWMCTL_ ST	PWMCTL_ST_ DDT		
			<obj>_PWMCTL_ DDT	PWMCTL_CFG_ DDT		
aPSxRamp	Ramp	ARAMP	<obj>_ARAMP	ARAMP	$aPSxRamp	The description of the Asset is retrieved from the description of the DFB. From the constant Range variable: Engineering Units is extracted from the comment of the field HI if fulfilled. Numeric Format is extracted from the comment of the field LO if fulfilled. The High and Low range is retrieved from the HI and LO field initial values.
			<obj>_ARAMP_ ST	ARAMP_ST_DDT		
			<obj>_ARAMP_ CFG	ARAMP_CFG_ DDT		
			<obj>_SP_RNG	RANGE_DDT		
aPSxRatioCtl	Ratio Control-ler	RA-TIOCTL	<obj>_RATIOCTL	RATIOCTL	$aPSxRa-tioCtl	The description of the Asset is retrieved from the description of the DFB.
			<obj>_ RATIOCTL_ST	RATIOCTL_ST_ DDT		
			<obj>_ RATIOCTL_DDT	RATIOCTL_CFG_ DDT		
aPSxRealSP	Real Setpoint	n/a	**<obj>_REALSP**	REAL	$aPSx-RealSP	The description of the Asset is retrieved from the description of the variable.
aPSxSequentialCon-trol *iPSxCondsumIC *iPSxCondsumFC	Sequen-tial Control	SEQCT-L1	<obj>_SEQCTL1	SEQCTL1	$aPSxSe-quentialCon-trol	The description of the Asset is retrieved from the description of the DFB. Maximum one of the SEQPAR05, SEQPAR10 and SEQPAR16 variables are expected at same time. The description of each parameter is retrieved from the description of the related field 'IP##', 'OP##' and 'RPT##'.
			<obj>_SEQCTL1_ ST	SEQCTL1_ST_ DDT		
			<obj>_SEQCTL1_ CFG	SEQCTL1_CFG_ DDT		
			<obj>_ SEQPAR05_ST	SEQPAR05_ST_ DDT		
			<obj>_ SEQPAR10_ST	SEQPAR10_ST_ DDT		
			<obj>_ SEQPAR16_ST	SEQPAR16_ST_ DDT		
aPSxSplitRangeCtl *iPSxCondsum1Ana-log	Split Range Control-ler	SPLRG-CTL	<obj>_SPLRGCTL	SPLRGCTL	$aPSxSpli-tRangeCtl	The description of the Asset is retrieved from the description of the DFB. From the constant Range variables: OP Engineering Units is extracted from the comment of the field HI if fulfilled. OP Numeric Format is extracted from the comment of the field LO if fulfilled.
			<obj>_ SPLRGCTL_ST	SPLRGCTL_ST_ DDT		
			<obj>_ SPLRGCTL_CFG	SPLRGCTL_ CFG_DDT		
			<obj>_SP_RNG	RANGE_DDT		
			<obj>_OP_RNG	RANGE_DDT		

Pattern File (& ref. to specific Include Rules)	Asset Type	DFB Type	Variable Naming Convention[1]	Variable Data Type	ASP Template	Comments
						The High and Low range is retrieved from the HI and LO field initial values.
aPSxStep3Ctl *iPSxCondsum1Ana-log	3 Steps Control-ler	STEP3-CTL	<obj>_STEP3CTL	STEP3CTL	$aPSx-Step3Ctl	The description of the Asset is retrieved from the description of the DFB. From the constant Range variable: Engineering Units is extracted from the comment of the field HI if fulfilled. Numeric Format is extracted from the comment of the field LO if fulfilled. The High and Low range is retrieved from the HI and LO field initial values.
			<obj>_ STEP3CTL_ST	STEP3CTL_ST_ DDT		
			<obj>_ STEP3CTL_CFG	STEP3CTL_ CFG_DDT		
			<obj>_PV_RNG	RANGE_DDT		
aPSxTotal *iPSxCondsum	Totalizer	TOTAL	<obj>_TOTAL	TOTAL	$aPSxTotal	The description of the Asset is retrieved from the description of the DFB.
			<obj>_TOTAL_ST	TOTAL_ST_DDT		
			<obj>_TOTAL_ CFG	TOTAL_CFG_ DDT		
aPSxValve *iPSxCondsum1Discr *iPSxCondsum *iPSxDevmnt *iPSxDevlp	On/Off Valve	DEVCT-L	<obj>_DEVCTL	DEVCTL	$aPSxValve	The description of the Asset is retrieved from the description of the DFB.
			<obj>_DEVCTL_ ST	DEVCTL_ST_ DDT		
1. **BOLD** indicates the variables are needed from the AppObject creation rule in the Pattern.						

Device Patterns

This table describes the contents of the device pattern files (.XML) and the asset types for which they help to automate their supervisory responsibilities:

Pattern File (& ref. to specific Include Rules)	Asset Type	DFB Type	Variable Naming Convention[1]	Variable Data Type	ASP Template	Comments
aPSxHWCompact *iPSxCommon. ixml*	Circuit Breakers (Description for Hardwired Compact)	HWCIR-CUIT-BREAKER	<Obj>_ HWCIRCUITBREAK-ER	HWCIRCUIT-BREAKER	$PSxHW-Compact	The description of the Asset is retrived from the description of the DFB.
			<Obj>_HWCB_CFG	HWCB_CFG_ DDT		
			<Obj>_HWCB_ST	HWCB_ST_DDT		
aPSxCompact *iPSxCommon. ixml*	Circuit Breakers (COMPACT - Compact NSX Protection Unit)	MBCOM-PACTNSX MBUCOM-PACTNSX	<Obj>_ MBCOMPACTNSX	MBCOM-PACTNSX	$PSxCom-pact	The description of the Asset is retrived from the description of the DFB.
			<Obj>_ MBUCOMPACTNSX	MBUCOM-PACTNSX		
			<Obj>_COMPACT_ CFG	COMPACT_ CFG_DDT		
			<Obj>_COMPACT_ ST	COMPACT_ST_ DDT		
			<Obj>_COMPACT_ MEA	COMPACT_ MEA_DDT		
			<Obj>_COMPACT_ MEAExt	COMPACT_ MEAExt_DDT		
			<Obj>_COMPACT_ MEAExt1	COMPACT_ MEAExt1_DDT		

Pattern File (& ref. to specific Include Rules)	Asset Type	DFB Type	Variable Naming Convention[1]	Variable Data Type	ASP Template	Comments
aPSxHWCircuit-Breaker *iPSxCommon.ixml*	Circuit Breakers (Description for Hardwired Circuit Breaker)	HWCIR-CUIT-BREAKER	<Obj>_ HWCIRCUITBREAK-ER	HWCIRCUIT-BREAKER	PSxHWCir-cuitBreaker	The description of the Asset is retrived from the description of the DFB.
			<Obj>_HWCB_CFG	HWCB_CFG_ DDT		
			<Obj>_HWCB_ST	HWCB_ST_DDT		
aPSxHWMaster-pact *iPSxCommon.ixml*	Circuit Breakers (Description for Hardwired Masterpact)	HWCIR-CUIT-BREAKER	<Obj>_ HWCIRCUITBREAK-ER	HWCIRCUIT-BREAKER	PSxHWM-asterpact	The description of the Asset is retrived from the description of the DFB.
			<Obj>_HWCB_CFG	HWCB_CFG_ DDT		
			<Obj>_HWCB_ST	HWCB_ST_DDT		
aPSxMasterPACT *iPSxCommon.ixml*	Circuit Breakers (MasterPACT Protection Unit with Chassis)	MBMAS-TERPACT	<Obj>_ MBMASTERPACT	MBMASTER-PACT	$PSxMas-terPACT	The description of the Asset is retrived from the description of the DFB.
			<Obj>_ MASTERPACT_CFG	MASTERPACT_ CFG_DDT		
			<Obj>_ MASTERPACT_ST	MASTERPACT_ ST_DDT		
			<Obj>_ MASTERPACT_MEA	MASTERPACT_ MEA_DDT		
			<Obj>_ MASTERPACT_ MEAExt	MASTERPACT_ MEAExt_DDT		
			<Obj>_ MASTERPACT_ MEAExt1	MASTERPACT_ MEAExt1_DDT		
aPSxMaster-pactMTZwoC *iPSxCommon.ixml*	Circuit Breakers (Master-pactMTZ Protection Unit without Chassis)	MBUMAS-TER-PACTMTZ	<Obj>_ MBUMASTER-PACTMTZ	MBUMASTER-PACTMTZ	$PSxMas-ter-pactMTZ-woC	The description of the Asset is retrived from the description of the DFB.
			<Obj>_ MASTERPACT_CFG	MASTERPACT_ CFG_DDT		
			<Obj>_ MASTERPACT_ST	MASTERPACT_ ST_DDT		
			<Obj>_ MASTERPACT_MEA	MASTERPACT_ MEA_DDT		
			<Obj>_ MASTERPACT_ MEAExt	MASTERPACT_ MEAExt_DDT		
			<Obj>_ MASTERPACT_ MEAExt1	MASTERPACT_ MEAExt1_DDT		
aPSxMaster-PACTNxC *iPSxCommon.ixml*	Circuit Breakers (MasterpactNx Protection Unit with Chassis)	MBUMAS-TER-PACTNxC	<Obj>_ MBUMASTER-PACTNxC	MBUMASTER-PACTNxC	$PSxMas-ter-PACTNxC	The description of the Asset is retrived from the description of the DFB.
			<Obj>_ MASTERPACT_CFG	MASTERPACT_ CFG_DDT		
			<Obj>_ MASTERPACT_ST	MASTERPACT_ ST_DDT		
			<Obj>_ MASTERPACT_MEA	MASTERPACT_ MEA_DDT		
			<Obj>_ MASTERPACT_ MEAExt	MASTERPACT_ MEAExt_DDT		
			<Obj>_ MASTERPACT_ MEAExt1	MASTERPACT_ MEAExt1_DDT		
aPSxMasterPACT-woC	Circuit Breakers (Masterpact Protection Unit	MBMAS-TERPACT	<Obj>_ MBMASTERPACT	MBMASTER-PACT	$PSxMas-terPACT-woC	The description of the Asset is retrived from the description of the DFB.

Pattern File (& ref. to specific Include Rules)	Asset Type	DFB Type	Variable Naming Convention[1]	Variable Data Type	ASP Template	Comments
iPSxCommon.ixml	without Chassis)		**<Obj>_ MASTERPACT_CFG**	MASTERPACT_ CFG_DDT		
			<Obj>_ MASTERPACT_ST	MASTERPACT_ ST_DDT		
			<Obj>_ MASTERPACT_MEA	MASTERPACT_ MEA_DDT		
			<Obj>_ MASTERPACT_ MEAExt	MASTERPACT_ MEAExt_DDT		
			<Obj>_ MASTERPACT_ MEAExt1	MASTERPACT_ MEAExt1_DDT		
aPSxMaster-pactMTZC *iPSxCommon.ixml*	Circuit Breakers (Master-pactMTZ Protection Unit with Chassis)	MBUMAS-TER-PACTMT-CZ	<Obj>_ MBUMASTER-PACTMTZC	MBUMASTER-PACTMTZC	$PSxMas-ter-pactMTZC	The description of the Asset is retrived from the description of the DFB.
			<Obj>_ MASTERPACT_CFG	MASTERPACT_ CFG_DDT		
			<Obj>_ MASTERPACT_ST	MASTERPACT_ ST_DDT		
			<Obj>_ MASTERPACT_MEA	MASTERPACT_ MEA_DDT		
			<Obj>_ MASTERPACT_ MEAExt	MASTERPACT_ MEAExt_DDT		
			<Obj>_ MASTERPACT_ MEAExt1	MASTERPACT_ MEAExt1_DDT		
aPSxMasterPACTC *iPSxCommon.ixml*	Circuit Breakers (MasterPACT Protection Unit with Chassis)	MBMAS-TER-PACTC	<Obj>_ MBMASTERPACTC	MBMASTER-PACTC	$PSxMas-terPACTC	The description of the Asset is retrived from the description of the DFB.
			<Obj>_ MASTERPACT_CFG	MASTERPACT_ CFG_DDT		
			<Obj>_ MASTERPACT_ST	MASTERPACT_ ST_DDT		
			<Obj>_ MASTERPACT_MEA	MASTERPACT_ MEA_DDT		
			<Obj>_ MASTERPACT_ MEAExt	MASTERPACT_ MEAExt_DDT		
			<Obj>_ MASTERPACT_ MEAExt1	MASTERPACT_ MEAExt1_DDT		
aPSxMaster-PACTNxwoC *iPSxCommon.ixml*	Circuit Breakers (MasterpactNx Protection Unit without Chassis)	MBUMAS-TER-PACTNx	<Obj>_ MBUMASTER-PACTNx	MBUMASTER-PACTNx	$PSxMas-ter-PACTNx-woC	The description of the Asset is retrived from the description of the DFB.
			<Obj>_ MASTERPACT_CFG	MASTERPACT_ CFG_DDT		
			<Obj>_ MASTERPACT_ST	MASTERPACT_ ST_DDT		
			<Obj>_ MASTERPACT_MEA	MASTERPACT_ MEA_DDT		
			<Obj>_ MASTERPACT_ MEAExt	MASTERPACT_ MEAExt_DDT		
			<Obj>_ MASTERPACT_ MEAExt1	MASTERPACT_ MEAExt1_DDT		
aPSxSepam20CB *iPSxCommon.ixml*	Digital Protection Relays (Digital Protection	MBSE-PAM20CB	<Obj>_ MBSEPAM20CB	MBSEPAM20CB	$PSxSe-pam20CB	The description of the Asset is retrived from the description of the DFB.
			<Obj>_SEPAM_CFG	SEPAM_CFG_ DDT		

Pattern File (& ref. to specific Include Rules)	Asset Type	DFB Type	Variable Naming Convention[1]	Variable Data Type	ASP Template	Comments
	Relays Sepam 20C Modbus Serial; Sepam 20C MB TCP I/O Scanning)		**<Obj>_SEPAM_ST**	SEPAM_ST_DDT		
			<Obj>_SEPAM_ VMEA	SEPAM_VMEA_ DDT		
			<Obj>_SEPAM_IO20	SEPAM_IO20_ DDT		
aPSxSe-pam20CSTM *iPSxCommon. ixml*	Digital Protection Relays (Digital Protection Relays Sepam 20C STM Modbus Serial; Sepam 20C STM TCP I/O Scanning)	MBSE-PAM20CS-TM	<Obj>_ MBSEPAM20CSTM	MBSE-PAM20CSTM	$PSxSe-pam20CS-TM	The description of the Asset is retrived from the description of the DFB.
			<Obj>_SEPAM_CFG	SEPAM_CFG_ DDT		
			<Obj>_SEPAM_ST	SEPAM_ST_DDT		
			<Obj>_SEPAM_ AMEA	SEPAM_AMEA_ DDT		
			<Obj>_SEPAM_IO20	SEPAM_IO20_ DDT		
aPSxSepam40C *iPSxCommon. ixml*	Digital Protection Relays (Digital Protection Relays Sepam 40C Modbus Serial; Sepam 40C MB TCP I/O Scanning)	MBSE-PAM40C	<Obj>_ MBSEPAM40C	MBSEPAM40C	$PSxSe-pam40C	The description of the Asset is retrived from the description of the DFB.
			<Obj>_SEPAM_CFG	SEPAM_CFG_ DDT		
			<Obj>_SEPAM_ST	SEPAM_ST_DDT		
			<Obj>_SEPAM_IO40	SEPAM_IO40_ DDT		
			<Obj>_SEPAM_MEA	SEPAM_MEA_ DDT		
aPSxSepam80C *iPSxCommon. ixml*	Digital Protection Relays (Digital Protection Relays Sepam 80C Modbus Serial; Sepam 80C MB TCP I/O Scanning)	MBSE-PAM40C ESE-PAM80C	<Obj>_ MBSEPAM80C	MBSEPAM80C	$PSxSe-pam80C	The description of the Asset is retrived from the description of the DFB.
			<Obj>_ESEPAM80C	ESEPAM80C		
			<Obj>_SEPAM_CFG	SEPAM_CFG_ DDT		
			<Obj>_SEPAM_ST	SEPAM_ST_DDT		
			<Obj>_SEPAM_IO80	SEPAM_IO80_ DDT		
			<Obj>_SEPAM_MEA	SEPAM_MEA_ DDT		
aPSxAccuSine *iPSxCommon. ixml*	Harmonic Filters (AccuSine)	EACCU-SINE	<Obj>_EACCUSINE	EACCUSINE	$PSxAccu-Sine	The description of the Asset is retrived from the description of the DFB.
			<Obj>_ACCUSINE_ CFG	ACCUSINE_ CFG_DDT		
			<Obj>_ACCUSINE_ ST	ACCUSINE_ST_ DDT		
			<Obj>_ACCUSINE_ MEA	ACCUSINE_ MEA_DDT		
aPSxPM1200 *iPSxCommon. ixml*	Power Meters (Power Meter PM1200 MB Serial)	MBP-M1200	<Obj>_MBPM1200	MBPM1200	$PSxP-M1200	The description of the Asset is retrived from the description of the DFB.
			<Obj>PM_CFG	PM1200_CFG_ DDT		
			<Obj>_PM_ST	PM_ST_DDT		
			<Obj>_PM_MEA	PM1200_MEA_ DDT		
aPSxPM5350 *iPSxCommon. ixml*	Power Meters (Power Meter PM5350 MB Serial)	MBP-M5350	<Obj>_MBPM5350	MBPM5350	$PSxP-M5350	The description of the Asset is retrived from the description of the DFB.
			<Obj>PM_CFG	PM_CFG_DDT		
			<Obj>_PM_ST	PM_ST_DDT		
			<Obj>_PM_MEA	PM_MEA_DDT		
aPSxPM53xx *iPSxCommon. ixml*	Power Meters (Power Meter PM53xx MBTCP	EMPM53x-x	<Obj>_EMPM53xx	EMPM53xx	$PSxP-M53xx	The description of the Asset is retrived from the description of the DFB.
			<Obj>PM_CFG	PM_CFG_DDT		

Pattern File (& ref. to specific Include Rules)	Asset Type	DFB Type	Variable Naming Convention[1]	Variable Data Type	ASP Template	Comments
	Explicit Messaging)		**<Obj>_PM_ST**	PM_ST_DDT		
			<Obj>_PM_MEA	PM53xx_MEA_DDT		
aPSxPM82xx *iPSxCommon.ixml*	Power Meters (Power Meter PM82xx MBTCP Explicit Messaging)	EMPM82x-x	<Obj>_EMPM82xx	EMPM82xx	$PSxP-M82xx	The description of the Asset is retrived from the description of the DFB.
			<Obj>PM_CFG	PM82xx_CFG_DDT		
			<Obj>_PM_ST	PM_ST_DDT		
			<Obj>_PM_MEA	PM82xx_MEA_DDT		
aPSxPM710 *iPSxCommon.ixml*	Power Meters (Power Meter PM710 MB Serial)	MBPM700	<Obj>_MBPM700	MBPM700	$PSxP-M710	The description of the Asset is retrived from the description of the DFB.
			<Obj>PM_CFG	PM_CFG_DDT		
			<Obj>_PM_ST	PM_ST_DDT		
			<Obj>_PM_MEA	PM_MEA_DDT		
aPSxPM800 *iPSxCommon.ixml*	Power Meters (Power Meters: PM800 MB TCP Explicit Messaging; PM800 MB Serial)	MBPM800 EPM800	<Obj>_MBPM800	MBPM800	$PSxP-M800	The description of the Asset is retrived from the description of the DFB.
			<Obj>_EPM800	EPM800		
			<Obj>PM_CFG	PM_CFG_DDT		
			<Obj>_PM_ST	PM_ST_DDT		
			<Obj>_PM_MEA	PM_MEA_DDT		
aPSxPM9C *iPSxCommon.ixml*	Power Meters (Power Meter PM9C MB Serial)	MBPM9C	<Obj>_MBPM9C	MBPM9C	$PSxPM9-C	The description of the Asset is retrived from the description of the DFB.
			<Obj>PM_CFG	PM9C_CFG_DDT		
			<Obj>_PM_MEA	PM9C_MEA_DDT		
			<Obj>_PM_ST	PM_ST_DDT		
aPSxATS22 *iPSxCommon.ixml*	Soft Starters (ATS - Altistart 22 Progressive Starter)	MBATS22	<Obj>_MBATS22	MBATS22	$PSxAT-S22	The description of the Asset is retrived from the description of the DFB.
			<Obj>_ATS22_CFG	ATS22_CFG_DDT		
			<Obj>_ATS22_ST	ATS22_ST_DDT		
aPSxATS48 *iPSxCommon.ixml*	Soft Starters (ATS - Altistart 48 Progressive Starter)	MBATS48	<Obj>_MBATS48	MBATS48	$PSxAT-S48	The description of the Asset is retrived from the description of the DFB.
			<Obj>_ATS_CFG	ATS_CFG_DDT		
			<Obj>_ATS_ST	ATS_ST_DDT		
aPSxATV212 *iPSxCommon.ixml*	Speed Drives (ATV - Speed Drive)	MBATV212	<Obj>_MBATV212	MBATV212	$PSxAT-V212	The description of the Asset is retrived from the description of the DFB.
			<Obj>_ATV_CFG	ATV_CFG_DDT		
			<Obj>_ATV_ST	ATV_ST_DDT		
aPSxATVMainData *iPSxCommon.ixml*	Speed Drives (ATV Speed Drives: ATV61 & ATV71 MB TCP Explicit Messaging; ATV12 & ATV212 & ATV312 & ATV61 & ATV71 MB Serial; ATV61 & ATV71 Advantys; ATV61 & 71 on Profibus DP)	MBATV MBAT-V7161 PBAT-V7161 EME-SATV7161 EATV7161 EATV32 ASAT-V7161	<Obj>_MBATV	MBATV	$PSxATV-MainData	The description of the Asset is retrived from the description of the DFB.
			<Obj>_MBATV7161	MBATV7161		
			<Obj>_PBATV7161	PBATV7161		
			<Obj>_EMESATV7161	EMESATV7161		
			<Obj>_EATV7161	EATV7161		
			<Obj>_EATV32	EATV32		
			<Obj>_ASATV7161	ASATV7161		
			<Obj>_ATV_CFG	ATV_CFG_DDT		
			<Obj>_ATV_ST	ATV_ST_DDT		

Pattern File (& ref. to specific Include Rules)	Asset Type	DFB Type	Variable Naming Convention[1]	Variable Data Type	ASP Template	Comments
aPSxATVAllData *iPSxCommon.ixml*	Speed Drives (Speed drives ATV61 & ATV71 & ATV32 MB TCP IO Scanner; ATV61 & ATV71 CANopen)	ATV7161 EATV32	<Obj>_ATV7161	ATV7161	$PSxAT-VAllData	The description of the Asset is retrived from the description of the DFB.
			<Obj>_EATV32	EATV32		
			<Obj>_ATV_CFG	ATV_CFG_DDT		
			<Obj>_ATV_ST	ATV_ST_DDT		
			<Obj>_ATV_IOEXT	ATV_IOEXT_DDT		
			<Obj>_ATV_IO	ATV_IO_DDT		
aPSxATV6xxAllDa-ta *iPSxCommon.ixml*	Speed Drives (Altivar Process Variable Speed Drives ATV6xx)	ATV6xx	<Obj>_ATV6xx	ATV6xx	$PSxAT-V6xxAllDa-ta	The description of the Asset is retrived from the description of the DFB.
			<Obj>_ATV_CFG	ATV6xx_CFG_DDT		
			<Obj>_ATV_ST	ATV6xx_ST_DDT		
			<Obj>_ATV_IO	ATV6xx_IO_DDT		
			<Obj>_ATV_IOEXT	ATV6xx_IOEXT_DDT		
aPSxATV9xxAllDa-ta *iPSxCommon.ixml*	Speed Drives (Altivar Process Variable Speed Drives ATV9xx)	ATV9xx	<Obj>_ATV9xx	ATV9xx	$PSxAT-V9xxAllDa-ta	The description of the Asset is retrived from the description of the DFB.
			<Obj>_ATV_CFG	ATV9xx_CFG_DDT		
			<Obj>_ATV_ST	ATV9xx_ST_DDT		
			<Obj>_ATV_IO	ATV9xx_IO_DDT		
			<Obj>_ATV_IOEXT	ATV9xx_IOEXT_DDT		
aPSxATV6xxAllDa-taandWarnings *iPSxCommon.ixml*	Speed Drives (Altivar Process Variable Speed Drives ATV6xx with Process Warnings)	ATV6xx	<Obj>_ATV6xx	ATV6xx	aPS-xATV6x-xAllDa-taand-Warnings	The description of the Asset is retrived from the description of the DFB.
			<Obj>_ATV_CFG	ATV6xx_CFG_DDT		
			<Obj>_ATV_ST	ATV6xx_ST_DDT		
			<Obj>_ATV_IO	ATV6xx_IO_DDT		
			<Obj>_ATV_IOEXT	ATV6xx_IOEXT_DDT		
			<Obj>_EMATVWARN_CFG	EMATVWARN_CFG_DDT		
			<Obj>_EMATVWARN_ST	EMATVWARN_ST_DDT		
aPSxATV6xxxAll-DataandWarnings *iPSxCommon.ixml*	Speed Drives (Altivar Process Variable Speed Drives ATV6xxx)	ATV6xxx-Warn	<Obj>_ATV6xxxWarn	ATV6xxxWarn	$PSxAT-V6xxxAll-Dataand-Warnings	The description of the Asset is retrived from the description of the DFB.
			<Obj>_ATV_CFG	ATV6xxx_CFG_DDT		
			<Obj>_ATV_ST	ATV6xxx_ST_DDT		
			<Obj>_ATV_IO	ATV6xxx_IO_DDT		
			<Obj>_ATV_IOEXT	ATV6xxx_IOEXT_DDT		
			<Obj>_EMATVWARN_CFG	EMATVWARN1_CFG_DDT		
			<Obj>_EMATVWARN_ST	EMATVWARN1_ST_DDT		
aPSxATV9xxAllDa-taandWarnings *iPSxCommon.ixml*	Speed Drives (Altivar Process Variable Speed Drives ATV9xx with Process Warnings)	ATV9xx	<Obj>_ATV9xx	ATV9xx	$PSxAT-V9xxAllDa-taand-Warnings	The description of the Asset is retrived from the description of the DFB.
			<Obj>_ATV_CFG	ATV9xx_CFG_DDT		
			<Obj>_ATV_ST	ATV9xx_ST_DDT		

Pattern File (& ref. to specific Include Rules)	Asset Type	DFB Type	Variable Naming Convention[1]	Variable Data Type	ASP Template	Comments
			<Obj>_ATV_IOEXT	ATV9xx_IOEXT_DDT		
			<Obj>_ATV_IO	ATV9xx_IO_DDT		
			<Obj>_EMATVWARN_CFG	EMATVWARN_CFG_DDT		
			<Obj>_EMATVWARN_ST	EMATVWARN_ST_DDT		
aPSxTesysTMEA *iPSxCommon.ixml*	Motor Controllers and Starters (TesysT - Motor Controllers and Starters)	EIOSTESYST EMETESYST MBTESYST	<Obj>_EIOSTESYST	EIOSTESYST	$PSxTesysTMEA	The description of the Asset is retrived from the description of the DFB.
			<Obj>_EMETESYST	EMETESYST		
			<Obj>_MBTESYST	MBTESYST		
			<Obj>_TESYST_CFG	TESYST_CFG_DDT		
			<Obj>_TESYST_ST	TESYST_ST_DDT		
			<Obj>_TESYST_MEA	TESYST_MEA_DDT		
aPSxTesysTAllData *iPSxCommon.ixml*	Motor Controllers and Starters (Motor controllers TesysT Ethernet MB TCP I/O Scanning; TesysT Ethernet MB TCP Explicit Messaging; TesysT MB Serial)	EMESTESYST EIOSTESYST EMETESYST MBTESYST	<Obj>_EMESTESYST	EMESTESYST	$PSxTesysTAllData	The description of the Asset is retrived from the description of the DFB.
			<Obj>_EIOSTESYST	EIOSTESYST		
			<Obj>_EMETESYST	EMETESYST		
			<Obj>_MBTESYST	MBTESYST		
			<Obj>_TESYST_CFG	TESYST_CFG_DDT		
			<Obj>_TESYST_ST	TESYST_ST_DDT		
			<Obj>_TESYST_MEA	TESYST_MEA_DDT		
			<Obj>_TESYST_MEAEV40	TESYST_MEAEV40_DDT		
			<Obj>_TESYST_MEAEXT	TESYST_MEAEXT_DDT		
aPSxTesysTMainData *iPSxCommon.ixml*	Motor Controllers and Starters (TesysT - Motor Controllers and Starters)	ETESYST TESYSTCTL	<Obj>_ETESYST	ETESYST	$PSxTesysTMainData	The description of the Asset is retrived from the description of the DFB.
			<Obj>_TESYSTCTL	TESYSTCTL		
			<Obj>_TESYST_CFG	TESYST_CFG_DDT		
			<Obj>_TESYST_ST	TESYST_ST_DDT		
aPSxTesysUIO *iPSxCommon.ixml*	Motor Controllers and Starters (TesysU - Motor Controllers and Starters)	MBTESYSUC MBTESYUS TESYSUC	<Obj>_MBTESYSUC	MBTESYSUC	$PSxTesysUIO	The description of the Asset is retrived from the description of the DFB.
			<Obj>_MBTESYUS	MBTESYUS		
			<Obj>_TESYSUC	TESYSUC		
			<Obj>_TESYSU_CFG	TESYSU_CFG_DDT		
			<Obj>_TESYSU_ST	TESYSU_ST_DDT		
			<Obj>_TESYSU_IO	TESYSU_IO_DDT		
aPSxTesysUMainData *iPSxCommon.ixml*	Motor Controllers and Starters (Motor Controllers and Starters TesysUSCST; TesysUSCAD (Advantys))	MBTESYSUSCST TESYSUSCST TESYSUCTL	<Obj>_MBTESYSUSCST	MBTESYSUSCST	$PSxTesysUMainData	The description of the Asset is retrived from the description of the DFB.
			<Obj>_TESYSUSCST	TESYSUSCST		
			<Obj>_TESYSUCTL	TESYSUCTL		

Pattern File (& ref. to specific Include Rules)	Asset Type	DFB Type	Variable Naming Convention[1]	Variable Data Type	ASP Template	Comments
			<Obj>_TESYSU_CFG	TESYSU_CFG_DDT		
			<Obj>_TESYSU_ST	TESYSU_ST_DDT		
aPSxTesysUMEC *iPSxCommon.ixml*	Motor Controllers and Starters (Motor Controllers and Starters TesysUSCST; TesysUSCAD (Advantys))	MBTESY-SUSC TESY-SUSC	<Obj>_MBTESYSUSC	MBTESYSUSC	$PSxTesy-sUMEC	The description of the Asset is retrived from the description of the DFB.
			<Obj>_TESYSUSC	TESYSUSC		
			<Obj>_TESYSU_CFG	TESYSU_CFG_DDT		
			<Obj>_TESYSU_ST	TESYSU_ST_DDT		
			<Obj>_TESYSU_MEC	TESYSU_MEC_DDT		
aPSxTesysTMain-DataPB *iPSxCommon.ixml*	Motor Controllers and Starters (TesysT on Profibus - Motor Controllers and Starters)	PBTE-SYST	<Obj>_PBTESYST	PBTESYST	$PSxTe-sysTMain-DataPB	The description of the Asset is retrived from the description of the DFB.
			<Obj>_TESYSTPB_CFG	TESYST_CFG_DDT		
			<Obj>_TESYSTPB_SST	TESYST_ST_DDT		
aPSxATV6xxxAll-Data *iPSxCommon.ixml*	Speed Drives (Altivar Process Variable Speed Drives ATV6xxx)	ATV6xxx	<Obj>_ATV6xxx	ATV6xxx	$PSxAT-V6xxxAll-Data	The description of the Asset is retrived from the description of the DFB.
			<Obj>_ATV_CFG	ATV6xxx_CFG_DDT		
			<Obj>_ATV_ST	ATV6xxx_ST_DDT		
			<Obj>_ATV_IO	ATV6xxx_IO_DDT		
			<Obj>_ATV_IOEXT	ATV6xxx_IOEXT_DDT		

1. **BOLD** indicates the variables are needed from the AppObject creation rule in the Pattern.

ASP Templates and Pattern Names

Introduction

This section shows the correspondence between the AVEVA System Platform template names and the AssetLink pattern names.

NOTE: The patterns describe in this section are available after you install AssetLink through its installation file (`setup.exe`).

Process Patterns

Mapping Table

This table shows the correspondence for the sequence of AVEVA System Platform templates and the AssetLink process pattern names:

No.	Sub-family	ASP Template	Pattern Name
1	Analog Device Control	aPSxControlValve	aPSxControlValve
2	Analog Device Control	aPSxMotorizedValve	aPSxMotorizedValve
3	Analog Device Control	aPSxMotorVS	aPSxMotorVS
4	Auxiliary Functions	aPSxAlarmSummary	aPSxAlarmSummary
5	Auxiliary Functions	aPSxMessageBox	aPSxMessageBox

No.	Sub-family	ASP Template	Pattern Name
6	Equipment Module	aPSxEquipmentModule	aPSxEquipmentModule
7	InBatch Phase	aPSxIBPhase	aPSxIBPhase
8	On/Off Device Control	aPSxDualOutputValve	aPSxDualOutputValve
9	On/Off Device Control	aPSxHandValve	aPSxHandValve
10	On/Off Device Control	aPSxMotor	aPSxMotor
11	On/Off Device Control	aPSxMotor2	aPSxMotor2
12	On/Off Device Control	aPSxMotorizedValveD	aPSxMotorizedValveD
13	On/Off Device Control	aPSxValve	aPSxValve
14	Process Control	aPSxIMCtl	aPSxIMCtl
15	Process Control	aPSxLeadLagCtl	aPSxLeadLagCtl
16	Process Control	aPSxPID	aPSxPID
17	Process Control	aPSxPIDMultiplexer	aPSxPIDMultiplexer
18	Process Control	aPSxPWM	aPSxPWM
19	Process Control	aPSxRamp	aPSxRamp
20	Process Control	aPSxRatioCtl	aPSxRatioCtl
21	Process Control	aPSxSplitRangeCtl	aPSxSplitRangeCtl
22	Process Control	aPSxStep3Ctl	aPSxStep3Ctl
23	Sequential Control	aPSxSequentialControl	aPSxSequentialControl
24	Signal Processing	aPSxAnalogInput	aPSxAnalogInput
25	Signal Processing	aPSxAnalogInput1	aPSxAnalogInput1
26	Signal Processing	aPSxAnalogOutput	aPSxAnalogOutput
27	Signal Processing	aPSxASelect1	aPSxASelect1
28	Signal Processing	aPSxDigitalInput	aPSxDigitalInput
29	Signal Processing	aPSxDigitalOutput	aPSxDigitalOutput
30	Signal Processing	aPSxMAnalogInput1	aPSxMAnalogInput1
31	Signal Processing	aPSxTotal	aPSxTotal

Enineering Units

The following templates are supported in pattern files:

- aPSxAnalogInput
- aPSxAnalogInput1
- aPSxAnalogOutput
- aPSxASelect1
- aPSxControlValve
- aPSxIMCtl
- aPSxLeadLagCtl
- aPSxMAnalogInput1
- aPSxMotorizedValve
- aPSxMotorVS
- aPSxPID
- aPSxPWM
- aPSxRamp
- aPSxRatioCtl
- aPSxSplitRangeCtl

- aPSxStep3Ctl
- aPSxTotal

Device Patterns

Mapping Table

This table shows the correspondence for the sequence of AVEVA System Platform templates and the AssetLink device pattern names:

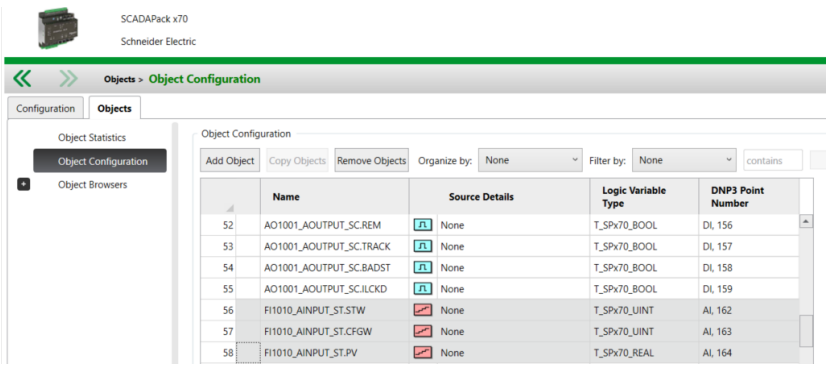
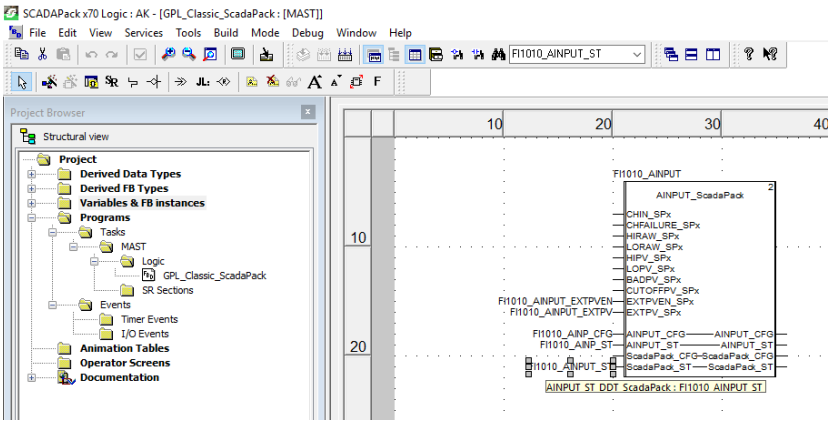
No.	Sub-family	ASP Template	Pattern Name
1	Circuit Breakers	$aPSxHWCompact	aPSxHWCompact
2	Circuit Breakers	$aPSxCompact	aPSxCompact
3	Circuit Breakers	$aPSxHWCircuitBreaker	aPSxHWCircuitBreaker
4	Circuit Breakers	$aPSxHWMasterpact	aPSxHWMasterpact
5	Circuit Breakers	$aPSxMasterPACT	aPSxMasterPACT
6	Circuit Breakers	$aPSxMasterpactMTZwoC	aPSxMasterpactMTZwoC
7	Circuit Breakers	$aPSxMasterPACTNxC	aPSxMasterPACTNxC
8	Circuit Breakers	$aPSxMasterPACTwoC	aPSxMasterPACTwoC
9	Circuit Breakers	$aPSxMasterpactMTZC	aPSxMasterpactMTZC
10	Circuit Breakers	$aPSxMasterPACTC	aPSxMasterPACTC
11	Circuit Breakers	$aPSxMasterpactNxwoC	aPSxMasterpactNxwoC
12	Digital Protection Relays	$aPSxSepam20CB	aPSxSepam20CB
13	Digital Protection Relays	$PSxSepam20CSTM	aPSxSepam20CSTM
14	Digital Protection Relays	$aPSxSepam40C	aPSxSepam40C
15	Digital Protection Relays	$aPSxSepam80C	aPSxSepam80C
16	Accusine	$aPSxAccuSine	aPSxAccuSine
17	Power Meters	$aPSxPM1200	aPSxPM1200
18	Power Meters	$aPSxPM5350	aPSxPM5350
19	Power Meters	$aPSxPM53xx	aPSxPM53xx
20	Power Meters	$aPSxPM82xx	aPSxPM82xx
21	Power Meters	$aPSxPM710	aPSxPM710
22	Power Meters	$aPSxPM800	aPSxPM800
23	Power Meters	$aPSxPM9C	aPSxPM9C
24	Soft Starters	$aPSxATS22	aPSxATS22
25	Soft Starters	$aPSxATS48	aPSxATS48
26	Speed Drivers	$aPSxATV212	aPSxATV212
27	Speed Drivers	aPSxATVMainData	aPSxATVMainData
28	Speed Drivers	$aPSxATVAllData	aPSxATVAllData
29	Speed Drivers	$aPSxATV6xxAllData	aPSxATV6xxAllData
30	Speed Drivers	$aPSxATV9xxAllData	aPSxATV9xxAllData
31	Speed Drivers	$aPSxATV6xxxAllData	aPSxATV6xxxAllData
32	Speed Drivers	$aPSxATV6xxAllData and Warnings	aPSxATV6xxAllData and Warnings

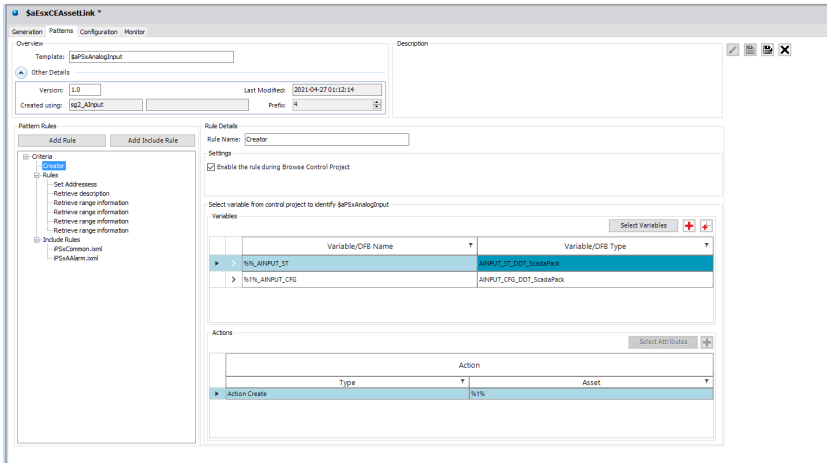
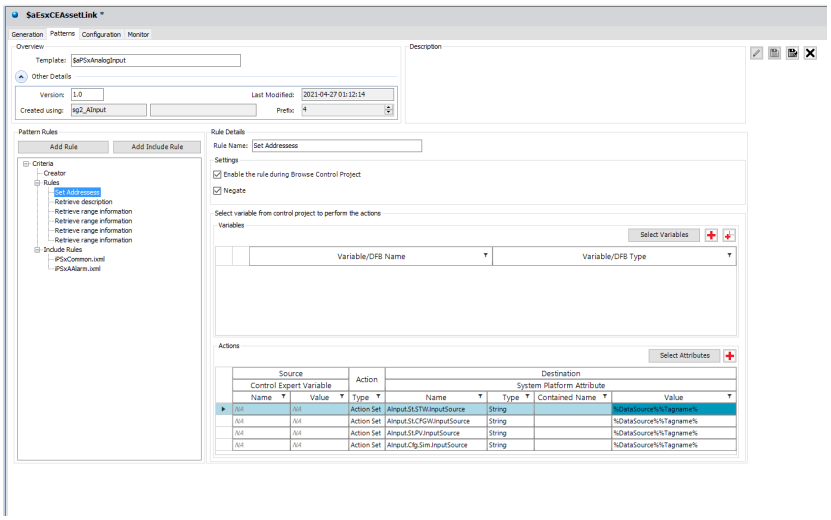
No.	Sub-family	ASP Template	Pattern Name
33	Speed Drivers	$aPSxATV6xxxAllData and Warnings	aPSxATV6xxxAllData and Warnings
34	Speed Drivers	$aPSxATV9xxAllData and Warnings	aPSxATV9xxAllData and Warnings
35	Motor Controllers and Starters	$aPSxTesysTMEA	aPSxTesysTMEA
36	Motor Controllers and Starters	$aPSxTesysTAllData	aPSxTesysTAllData
37	Motor Controllers and Starters	$aPSxTesysTMainData	aPSxTesysTMainData
38	Motor Controllers and Starters	$PSxTesysUIO	aPSxTesysUIO
39	Motor Controllers and Starters	$PSxTesysUMainData	aPSxTesysUMainData
40	Motor Controllers and Starters	$PSxTesysUMEC	aPSxTesysUMEC
41	Motor Controllers and Starters	$PSxTesysTMainDataPB	aPSxTesysTMainDataPB

SCADAPack Patterns

Mapping Table

The table below explains the mapping of SCADAPack Patterns.

Sr . No	Description
1	The figure below illustrates the RTU objects configured in **Remote Connect**. 
2	The figure below illustrates the control logic along with the respective function block. 

Sr . No	Description
3	The figure below is a screenshot of the Pattern Editor wherein the pattern is designed to match the respective function block. 
4	The figure below shows the Action set parameters for the respective RTU object parameter named `STW`, `CFGW` and `PV`. The respective System Platform attribute must be named as shown below. For example, the input source STW named as `AInput.St.STW.InputSource`, wherein, `STW` mapped with RTU object parameter is same as `STW`. Similarly, it is applicable for other parameters like `CFGW`, `PV`. 

Product Overview

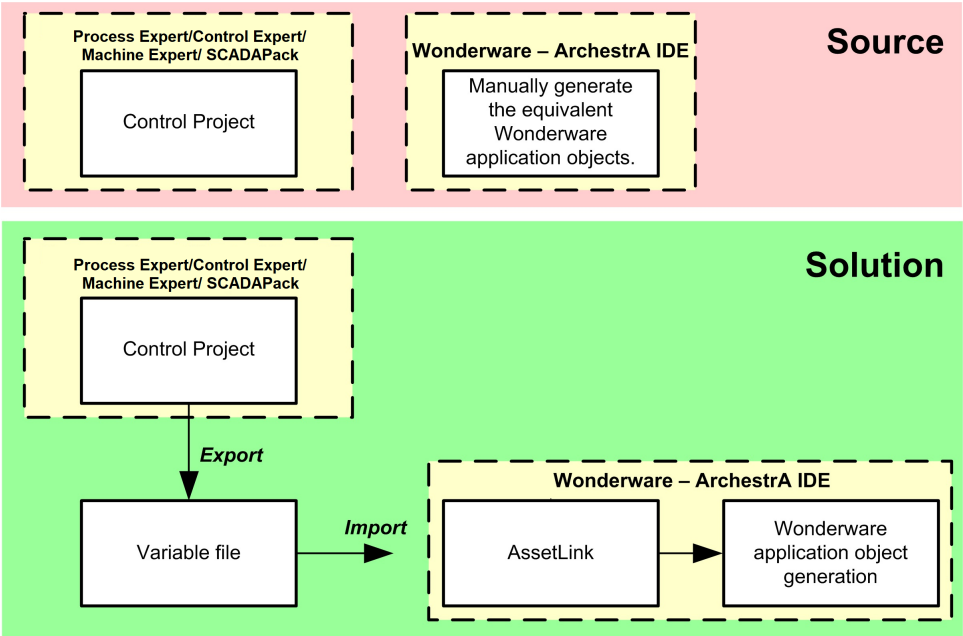
Conversion Overview

Introduction

The graphics below offer high-level views of some of the larger concepts associated with the use of EcoStruxure™ Control Expert AssetLink.

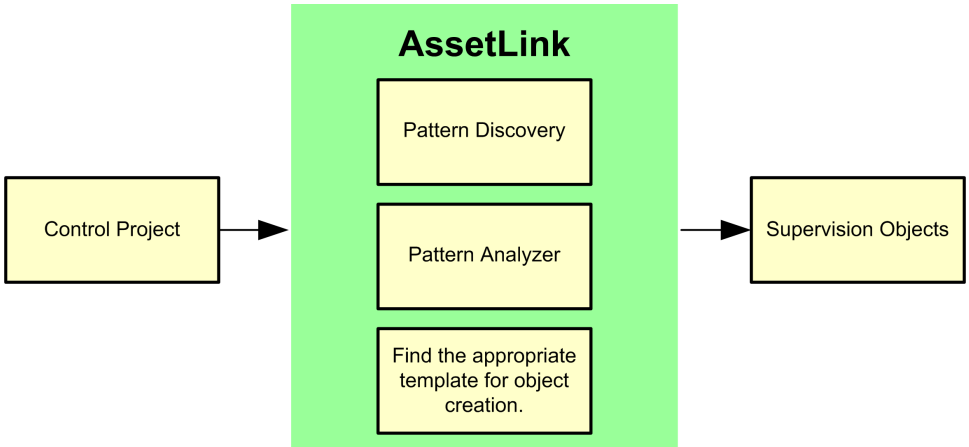
Application Evolution

This illustration shows the application's evolution when AssetLink is applied to the source project:



Supervision Object Extraction

This graphic shows the workflow when AssetLink extracts supervision objects from the source control project:



Preparation Activities

Before the conversion, create and fully test the operational control project logic and the corresponding ASP AppObject for each type of asset (a pump for

example) through the regular services of the PAC Modicon engineering tools and the ArchestrA IDE:

- Create control resources, such as DFB types, DDT, etc.
- Create the Supervisory ASP Templates.
- Test the functionality with examples that can later serve as references for the Pattern Discovery process.

  NOTE: When your naming conventions are applied systematically for control variables, AssetLink discovers the variables and applies them universally for all assets of the same type.

Pattern Discovery

AssetLink provides services that auto-discover the rules for generating AppObjects, so reducing the modeling engineering effort, and storing them in pattern files (.xml) that you can refine:

- Use this implementation, which is based on a ASP template, to launch the engineering tool from the ArchestrA IDE without leaving the ASP environment.

  NOTE: You has to import these templates to proceed with the pattern discovery process.
- Select a control project and ASP AppObject to use for pattern discovery. The rules that are applied to in the automatic identification of assets and their (optional) associated services are generated and stored in files that are used later in the bulk generation process. (You can adjust the automatically discovered patterns.)

Pattern Refinement

You can use this (optional) activity to refine the rules in patterns that AssetLink discovers. You can enhance patterns to increase the automation of bulk processing or reuse data from the control project.

Control Project Code

From Process Expert/Control Expert/Machine Expert, variables that are exported in .xsy or .xml files can be exported during bulk processing. The relevant information for both control and supervision (for example, descriptions and ranges) in entered from Process Expert/Control Expert/Machine Expert as they were entered for the instances.

Bulk Processing

Processes are executed as many times as required to universally generate or update ASP AppObjects based on information found in the control project variables and existing patterns:

- Trigger the exploration of the control project file that describes the variables in the controller and look for occurrences of defined patterns.
- The system shows a list of ASP AppObjects that should be created or updated to allow you to select or deselect them as needed.

Appendices

What's in This Part

What's in This Chapter

Library Installation

Overview

You can install the Modicon Libraries - General Purpose for AVEVA System Platform in ArchestrA IDE.

You can use the following libraries files to install in the ArchestrA IDE:

- Script function libraries.
- The Galaxy style library.

The recommended way to use the library with a new Galaxies is by creating a new Galaxy.

Installing the Library by Using Installation Files

The installation files are composed of:

- Four script function libraries:
  - *PSxLocalize.aaSLIB*
  - *ww.nasc.btl.modeling.aaSLIB*
  - *PSxMessaging.aaSLIB* (for attributes used by EcoStruxure Hybrid DCS runtime navigation services)
  - *System.Windows.Forms.aaSLIB*
- A Galaxy style library:
  - *GalaxyStyles-yyyymmdd.xml*
- Two packages containing the objects:
  - *GPL for ASP Master Templates yymmdd.aaPKG*
  - *GPL for ASP Application Templates yymmdd.aaPKG* (also contains master templates)

Proceed as follows to install the library by using the installation files.

Step	Action
1	Open ArchestrA IDE.
2	Click **Galaxy** > **Import** > **Galaxy Style Library**.
3	Select the `GalaxyStyles-yyyymmdd.xml` file and click **Open**.
4	Click **Galaxy** > **Import** > **Script Function Library**.
5	Select the `PSxLocalize.aaSLIB` file and click **Open**.
6	Click **Galaxy** > **Import** > **Script Function Library**.
7	Select the `PSxMessaging.aaSLIB` file and click **Open**.
8	Click **Galaxy** > **Import** > **Script Function Library**.
9	Select the `ww.nasc.btl.modeling.aaSLIB` file and click **Open**.
10	Click **Galaxy** > **Import** > **Script Function Library**.
11	Select the `System.Windows.Forms.aaSLIB` file and click **Open**.
12	Click **Galaxy** > **Import** > **Object(s)**.
13	Select the *GPL for ASP Master Templates yyyymmdd.aaPKG* file and click **Open**.
14	Click **Galaxy** > **Import** > **Object(s)**.
15	Select the *GPL for ASP Application Templates yyyymmdd.aaPKG* file and click **Open**.

Glossary

A

ArchestrA IDE:

ArchestrA Integrated Development Environment. This framework is incorporated with the Wonderware System Platform to facilitate the building of InTouch OMI ViewApps and managed InTouch HMI applications.

ASP:

AVEVA System Platform. This industrial software platform uses ArchestrA technology for HMI operations management, SCADA supervision, and production and performance management. ASP contains an integrated set of services and an extensible data model to manage plant control and information management systems. ASP supports both the supervisory control layer and the manufacturing execution system layer, presenting them as a single information source. Modular applications sit on top of the ASP Platform.

E

EcoStruxure™ Control Expert:

EcoStruxure™ Control Expert is the programming software for all PACs. The software includes five IEC languages that comply with IEC 61131-3. Depending on requirements, the application may use a mixture of different languages.

EcoStruxure™ Machine Expert:

EcoStruxure™ Machine Expert is a unique solution software for developing, configuring, and commissioning the entire machine in a single software environment, including logic, motion control, robotics/mechatronics, simulation, diagnostics, intelligent motor and load management and drives, HMI (Vijeo Designer), IIoT and related network automation functions.

EcoStruxure™ Process Expert:

EcoStruxure™ Process Expert (formerly named EcoStruxure™ Hybrid DCS) is a single automation system to engineer, operate and maintain the entire plant.

G

Galaxy:

A Galaxy is your entire production environment, including all computers and components that run your application. It is a collection of graphics, objects, engines, templates, and attributes that you define as a set of component parts of an InTouch HMI or OMI application.

O

OFS:

(*OPC Factory Server*) OFS enables real-time SCADA communications with the Control Expert family of PLCs. OFS utilizes the standard OPC data access protocol.

OPC DA:

(*OLE for Process Control Data Access*) The Data Access Specification is the most commonly implemented of the OPC standards that provide specifications for real-time data communications between clients and servers.

OPC UA:

OPC UA (Open Platform Communications United Architecture) is a data exchange standard for industrial communication (machine-to-machine or PC-to-machine communication).

P

PAC:

programmable automation controller. The PAC is the brain of an industrial manufacturing process. It automates a process as opposed to relay control systems. PACs are computers suited to survive the harsh conditions of an industrial environment.

Index

A

I

Schneider Electric
35 rue Joseph Monier
92500 Rueil Malmaison
France

+ 33 (0) 1 41 29 70 00

www.se.com

As standards, specifications, and design change from time to time, please ask for confirmation of the information given in this publication.

EIO0000004195.07