EcoStruxure™

Control Expert Asset Link

User Guide

06/2023

EIO0000004195.10

www.se.com

Schneider Electric

Legal Information

The Schneider Electric brand and any trademarks of Schneider Electric SE and its subsidiaries referred to in this guide are the property of Schneider Electric SE or its subsidiaries. All other brands may be trademarks of their respective owners.

This guide and its content are protected under applicable copyright laws and furnished for informational use only. No part of this guide may be reproduced or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), for any purpose, without the prior written permission of Schneider Electric.

Schneider Electric does not grant any right or license for commercial use of the guide or its content, except for a non-exclusive and personal license to consult it on an "as is" basis. Schneider Electric products and equipment should be installed, operated, serviced, and maintained only by qualified personnel.

As standards, specifications, and designs change from time to time, information contained in this guide may be subject to change without notice.

To the extent permitted by applicable law, no responsibility or liability is assumed by Schneider Electric and its subsidiaries for any errors or omissions in the informational content of this material or consequences arising out of or resulting from the use of the information contained herein.

As part of a group of responsible, inclusive companies, we are updating our communications that contain non-inclusive terminology. Until we complete this process, however, our content may still contain standardized industry terms that may be deemed inappropriate by our customers.

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.

Qualification of Personnel

A qualified person is one who has the following qualifications:

- Skills and knowledge related to the construction and operation of electrical equipment and the installation.
- Knowledge and experience in industrial control programming.
- Received safety-related training to recognize and avoid the hazards involved.

The qualified person must be able to detect possible hazards that may arise from parameterization, modifying parameter values and generally from mechanical, electrical, or electronic equipment. The qualified person must be familiar with the standards, provisions, and regulations for the prevention of industrial accidents, which they must observe when designing and implementing the system.

Proper Use

This product is a library to be used together with the automation control systems and is intended solely for the purposes described in the present documentation as applied in the industrial sector.

Always observe the applicable safety-related instructions, the specified conditions, and the technical data.

Perform a risk evaluation concerning the specific use before using the product. Take protective measures according to the result.

Since the product is used as a part of an overall system, you must ensure the safety of the personnel by means of the concept of this overall system (for example, machine concept).

Any other use is not intended and may be hazardous.

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:

(In case of divergence or contradiction between any translation and the English original, the original text in the English language will prevail.)

- 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 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 Asset Link.

Validity Note

This document has been updated for the release of EcoStruxure™ Control Expert Asset Link V4.0.

The characteristics that are described in the present document, as well as those described in the documents included in the Related Documents section below, can be found 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
Modicon Libraries 2022 - General Purpose for AVEVA System Platform (OMI)	EIO0000002094

Technical Support

Visit https://www.se.com/myschneider/ for support, software updates, and latest information.

Product Related Information

⚠WARNING

LOSS OF CONTROL

- Perform a Failure Mode and Effects Analysis (FMEA), or equivalent risk analysis, of your application, and apply preventive and detective controls before implementation.
- Provide a fallback state for undesired control events or sequences.
- Provide separate or redundant control paths wherever required.
- Supply appropriate parameters, particularly for limits.
- Review the implications of transmission delays and take actions to mitigate them.
- Review the implications of communication link interruptions and take actions to mitigate them.
- Provide independent paths for control functions (for example, emergency stop, over-limit conditions, and error conditions) according to your risk assessment, and applicable codes and regulations.
- Apply local accident prevention and safety regulations and guidelines.[1]
- Test each implementation of a system for proper operation before placing it into service.

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

[1] For additional information, refer to NEMA ICS 1.1 (latest edition), *Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control* and to NEMA ICS 7.1 (latest edition), *Safety Standards for Construction and Guide for Selection, Installation and Operation of Adjustable-Speed Drive Systems* or their equivalent governing your particular location.

Examples described in this manual are provided for information only.

⚠WARNING

UNINTENDED EQUIPMENT OPERATION

Adapt examples that are given in this manual to the specific functions and requirements of your industrial application before you implement them.

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

Terminology Derived from Standards

The technical terms, terminology, symbols and the corresponding descriptions in this manual, or that appear in or on the products themselves, are generally derived from the terms or definitions of international standards.

In the area of functional safety systems, drives and general automation, this may include, but is not limited to, terms such as safety, safety function, safe state, fault, fault reset, malfunction, failure, error, error message, dangerous, etc.

Among others, these standards include:

Standard	Description
IEC 61131-2:2007	Programmable controllers, part 2: Equipment requirements and tests.
ISO 13849-1:2015	Safety of machinery: Safety related parts of control systems. General principles for design.

Standard	Description
EN 61496-1:2013	Safety of machinery: Electro-sensitive protective equipment. Part 1: General requirements and tests.
ISO 12100:2010	Safety of machinery - General principles for design - Risk assessment and risk reduction.
EN 60204-1:2006	Safety of machinery - Electrical equipment of machines - Part 1: General requirements.
ISO 14119:2013	Safety of machinery - Interlocking devices associated with guards - Principles for design and selection.
ISO 13850:2015	Safety of machinery - Emergency stop - Principles for design.
IEC 62061:2015	Safety of machinery - Functional safety of safety-related electrical, electronic, and electronic programmable control systems.
IEC 61508-1:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: General requirements.
IEC 61508-2:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: Requirements for electrical/electronic/programmable electronic safety-related systems.
IEC 61508-3:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: Software requirements.
IEC 61784-3:2016	Industrial communication networks - Profiles - Part 3: Functional safety fieldbuses - General rules and profile definitions.
2006/42/EC	Machinery Directive
2014/30/EU	Electromagnetic Compatibility Directive
2014/35/EU	Low Voltage Directive

In addition, terms used in the present document may tangentially be used as they are derived from other standards such as:

Standard	Description
IEC 60034 series	Rotating electrical machines
IEC 61800 series	Adjustable speed electrical power drive systems
IEC 61158 series	Digital data communications for measurement and control – Fieldbus for use in industrial control systems

Finally, the term zone of operation may be used in conjunction with the description of specific hazards, and is defined as it is for a hazard zone or danger zone in the Machinery Directive (2006/42/EC) and ISO 12100:2010.

NOTE: The aforementioned standards may or may not apply to the specific products cited in the present documentation. For more information concerning the individual standards applicable to the products described herein, see the characteristics tables for those product references.

Securing the Environment

Hardening the Computer

Overview

The computers located in the control room are exposed to attacks. Those running EcoStruxure Process Expert, AVEVA Plant SCADA, OPC Factory Server, AVEVA System Platform or OPC UA Server Expert need to be hardened.

For more detailed information, refer to the System Technical Note *How can I... Reduce Vulnerability to Cyber Attacks*.

Hardening Engineering Workstations

Customers may choose from various commercial computer systems for their engineering workstation needs. Key hardening techniques include:

- Strong password management.
- User account management.
- Methods of least privilege applied to applications and user accounts.
- Removal or disabling unneeded services.
- Removing remote management privileges, if not necessary.
- Systematic patch management.

Using Antivirus Software

Use an antivirus software on each computer of the EcoStruxure Process Expert for AVEVA System Platform, Asset Link and AVEVA System Platform infrastructure and keep it up-to-date.

Disabling Unused Network Interface Cards

Verify that network interface cards that are not required by the application are disabled. For example, if your system has two cards and the application uses only one, verify that the other network card is disabled.

Refer to the help of the operating systems for instructions on how to proceed.

Introducing EcoStruxure™ Control Expert Asset Link

About Asset Link

Introduction

Asset Link is used for integration of control and supervisory systems.

EcoStruxure™ Control Expert Asset Link analyzes your control project from EcoStruxure™ Process Expert for AVEVA System Platform or EcoStruxure™ Control Expert or EcoStruxure™ Machine Expert and reads the configuration information (Function Blocks), data structures (Variables) and meta data (Comments and Initial Values). It generate Assets in AVEVA System Platform (ASP), with configured attributes and addresses reducing engineering time.

Product Objectives

Asset Link is a build-time engineering tool that assists you in the engineering of the ASP supervisory application by automating the creation and updating of application based on data extracted from EcoStruxure™ Control Expert or EcoStruxure™ Machine Expert control projects.

These are the main design principles of Asset Link:

- AVEVA System Platform IDE (formerly ArchestrA Integrated Development Environment) drives the process while Asset Link is delivered as an ASP template.
- Asset Link can retrieve and combine multiple data from one or more variables in the control project and use them from AppObjects modeling Assets.
- Asset Link serves the entire lifecycle of an automation system.

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

By delivering mechanisms that facilitate the reuse of information already available from 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 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 risks due to those incremental changes.
- *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

Asset Link is a licensed product. Observe this license activation process:

Stage	Description
1	Install Asset Link. Refer to the Installation, page 22
2	Activate the Asset Link License. Refer to the Activating a License, page 16.

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 Asset Link tool to see the updated status of license.
- Asset Link 1.0 is node-locked license, so it uses the **License Manager** for the license activation. However, Asset Link 2.1 and greater versions have a floating license, so you must activate the license using the **Floating License Manager**.

Activating a License

Activating software licenses is required to use Asset Link. The licensing mechanism involves using two software applications:

- The Floating License Manager (FLM): Allows you to activate licenses on a computer.
- The License Manager (LM): Indicates to the software on which computer the FLM that hosts the required licenses is installed.

Once you receive your license Activation ID from Schneider Electric, activate the license by using the Floating License Manager.

The following methods are available to activate a license:

- By Web: Default method when the local computer has an internet connection.
- By Web portal: Alternate method when no Internet connection is available on the PC where you wish to activate the license.

For a detailed description of each method, refer to the Schneider Electric Floating License Manager help.

Product Overview

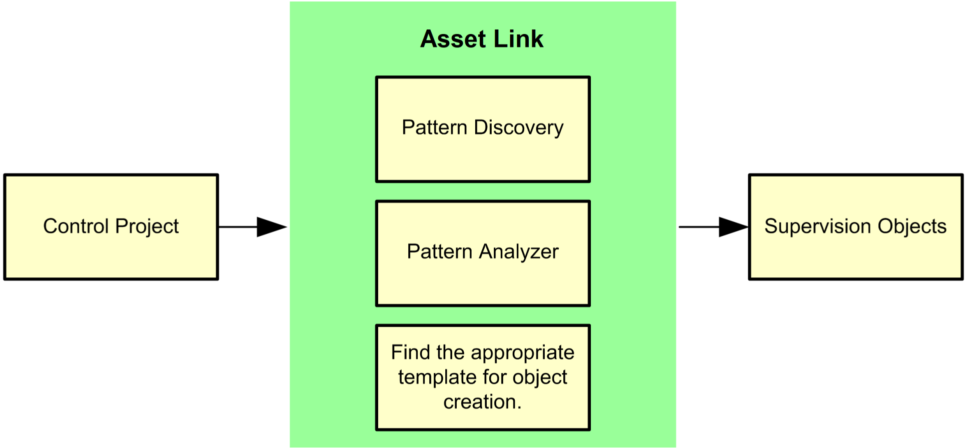
Conversion Overview

Introduction

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

Creation of Supervision Objects

The following illustration shows the process when Asset Link is applied to the source 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 System Platform 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 naming conventions are applied systematically for control variables, Asset Link discovers the variables and applies them for all assets of the same type.

Pattern Discovery

Asset Link provides services that auto-discover the rules for generating AppObjects and storing them in pattern files (.xml) that you can refine:

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

  NOTE: You must 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 the automatic identification of assets and their (optional) associated services are generated and stored in files that are used later in the Bulk Processing. You can adjust the automatically discovered patterns.

Pattern Refinement

You can use this (optional) activity to refine the rules in patterns that Asset Link discovers. You can enhance patterns to increase the automation of Bulk Processing or reuse data from the control project.

Control Project Code

From EcoStruxure Process Expert / EcoStruxure Control Expert / EcoStruxure Machine Expert, variables are exported to `.xsy` or `.xml` files.

Bulk Processing

Asset Link supports the Bulk Processing, to generate or update AVEVA System Platform 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 search for occurrences of defined patterns.
- The Asset Link application shows a list of ASP AppObjects that can be created or updated and allows you to select or deselect them as needed.

Requirements and Prerequisites

Introduction

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

Software Requirements

These are the versions that are required for use with Asset Link V4.0 operations:

- AVEVA System Platform 2020 and later
- EcoStruxure™ Process Expert for AVEVA System Platform 2021 and later
- EcoStruxure™ Control Expert V14.0 and later
- EcoStruxure™ Machine Expert V2.1 and later (formerly SoMachine V4.3 and later)
- OPC DA server v3.61 and later or OPC UA Server v1.0 and later

- An XML editor (For example: Altova XMLSpy for Schemas and XML Documents)
- Schneider Electric License Manager 2.9.0.0
- .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 **Display** settings, select these **Scale and layout** settings to implement the resolution of the Asset Link 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** is a configuration tool for PAC Modicon projects. Your choice of libraries is not limited to the PAC Modicon General Purpose Library (GPL), but Asset Link applies systematic rules that Asset Link applies to other libraries to automatically determine which ASP application objects are created and retrieved from the Control Expert project.
- **EcoStruxure™ Machine Expert** reduces engineering time through intuitive machine programming.
- **OPC Factory Server** (OFS)**:** Asset Link manages application object I/O references that indicate OPC DA Items through the AVEVA DIO OP client when it is connected to the OFS.
- **AVEVA System Platform** (ASP)**:** This is the industrial software platform 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.
- **General Purpose Library - EcoStruxure Process Expert for AVEVA System Platform 2021 General Purpose Library for AVEVA System Platform Hotfix 80887**: Ensure that AVEVA System Platform galaxy is created with GPL templates in ready to use state.

Before you begin the usage of Asset Link tool, export the appropriate source file, page 20.

Exporting Control Projects

Introduction

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

- *.xsy:* Export an .xsy file from Control Expert or EcoStruxure Process Expert for AVEVA System Platform.
- *.xml:* Export an .xml file from Machine Expert.
- *.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	Right 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 appears. 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

Asset Link Installation

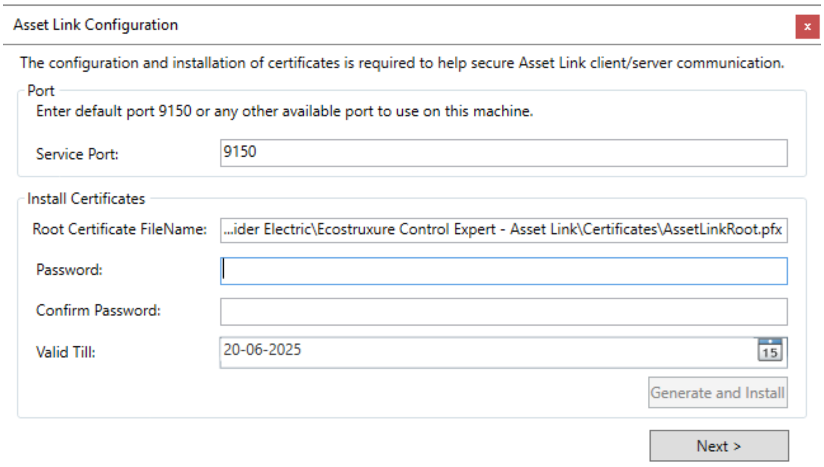
Instructions

Install the EcoStruxure™ Control Expert Asset Link software in the Galaxy Repository (GR) node:

Step	Action
1	Double-click the Asset Link installation file `setup.exe`. **NOTE:** The installation of .Net Framework is a prerequisite for the installation of Asset Link. If .Net Framework is not installed, the installation program installs it automatically. After a successful installation, restart your system and continue with the installation procedure.
2	Wait for the installation wizard to open and click the **Next** button.
3	If you agree, accept the licensing agreement and click the **Next** button.
4	Enter the appropriate information on the **Customer Information** page and click the **Next** button.

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\`
	In the **Program Files Destination Folder** dialog box, click the **Change** button to navigate to an installation file path.
	Alternately, you can accept the default program files destination folder:
	`C:\Program Files\Schneider Electric\Ecostruxure Control Expert - Asset Link\`
	Click the **Next** button.

6	Click the **Install** button. **NOTE: License Manager** and **Floating License Manager** are installed automatically.

Step	Action
7	During installation, Asset Link configuration dialog box gets displayed:  Asset Link Configuration The configuration and installation of certificates is required to help secure Asset Link client/server communication. Port Enter default port 9150 or any other available port to use on this machine. Service Port: 9150 Install Certificates Root Certificate FileName: ...ider Electric\Ecostruxure Control Expert - Asset Link\Certificates\AssetLinkRoot.pfx Password: Confirm Password: Valid Till: 20-06-2025 Enter **Password** and Confirm **Password** and click **Generate and Install** button. This installs certificates for Asset Link server client secure communication. **NOTE: Password set**: At least 8 character and a combination of upper case, lower case, symbols and numbers. **NOTE: Reset Password**: To reset the password of the certificates, use the **Repair** option from the Asset Link installer. The new generated root certificate has to be reinstalled in remote client machines for Asset Link. On the GR node machine, root certificate `AssetLinkRoot.pfx` is installed and this file is copied to `C:\ProgramData\Schneider Electric\Ecostruxure Control Expert - Asset Link\Certificates`.
8	**Service Port**: Default port is 9150 or enter other available port in the computer. This port is used for Asset Link server/client communication. Click on **Next** to proceed with the installation. **NOTE:** Post installation, in the case of certificate expiration, the certificate needs to be reinstalled using the Asset Link installation **Repair** option. Also, if a **Service Port** number is to be modified, then use the **Repair** option.
9	Click on **Finish** button to complete the installation.

Installation Contents

The installation folder contains these sub-folders:

Folder	Description
Asset Link Template and Pattern Schema	This folder contains `$EsxCEAssetLink.aaPKG` Asset Link template and `PACConnectorSchema.xsd` file. The .xml schema in the `PACConnectorSchema.xsd` file can be used by .xml editors to manually create patterns that conform to the appropriate syntax and structure, page 76. When you copy this file to every folder that contains patterns, Asset Link verifies their validity too.
GPL Patterns	The General Purpose Library includes a set of patterns that are copied to folders that can be accessed by Asset Link or used as examples for the creation of new patterns. You can dedicate any folder to this purpose, but the use of a shared folder allows Asset Link to access the patterns from workstations that run System Platform IDE.
GPL ASP Templates	The ASP base template `$EsxCEAssetLink` and its derived template `$aESxPACConnector` are delivered in an ASP object export file (. aaPDF), which is imported to Galaxy where Asset Link is used from, via System Platform IDE from the `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 EcoStruxure Control Expert Asset Link 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. Also contains Application Project XML schema and sample file which can be referenced to create the application project instances data file for sources other than Process Expert, to read the application instance data by configuring the Pattern Action and provide this file as input in Asset Link configuration. For Process Expert this file is generated from Ecostruxure Process Expert for AVEVA System Platform application.
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 Asset Link delivery.

NOTE: For the installation of Modicon Libraries - General Purpose Library refer to Appendix A, page 118

Using Demo Templates

Overview

This section describes how to use demo templates in the Asset Link.

Follow these steps to use demo templates in the Asset Link:

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

Multi Client Architecture

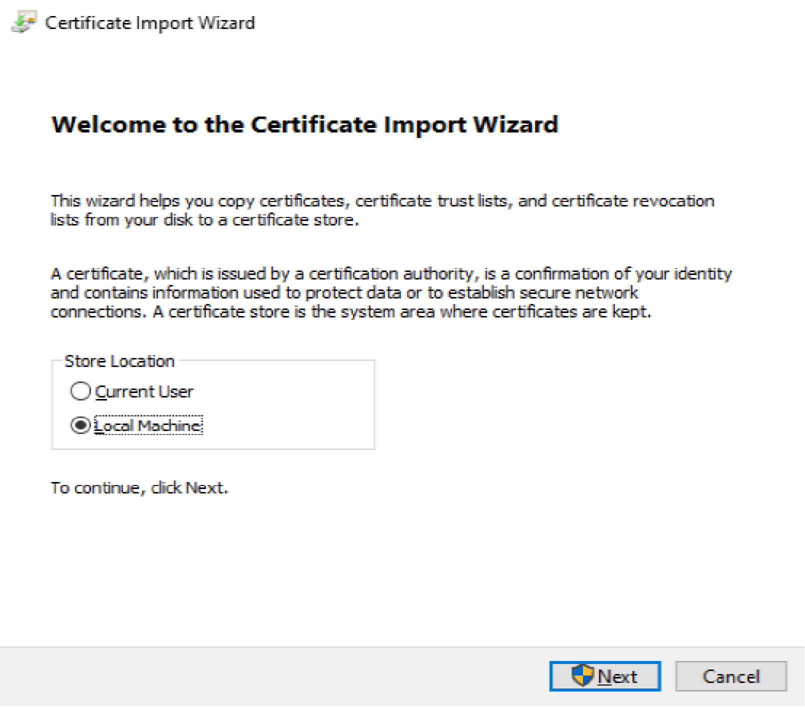
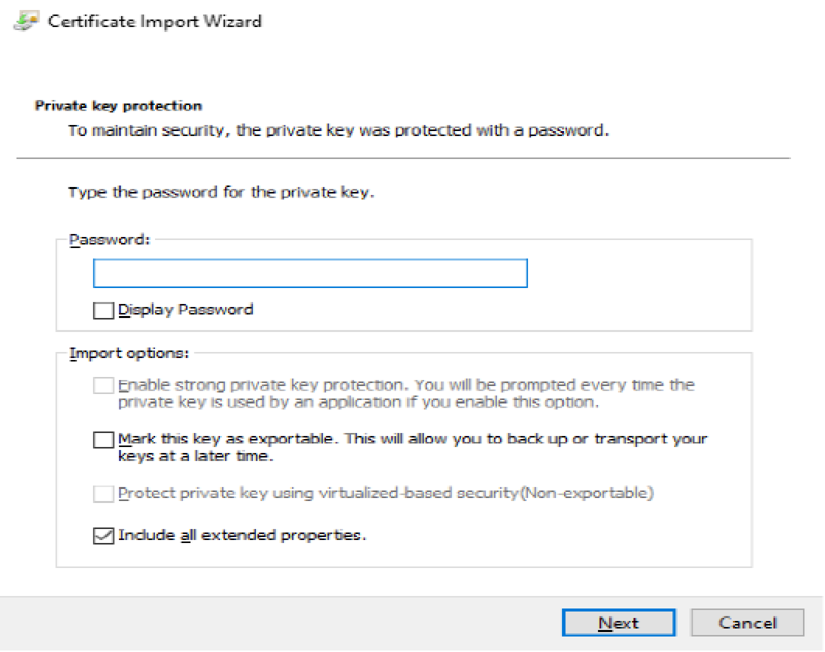
Overview

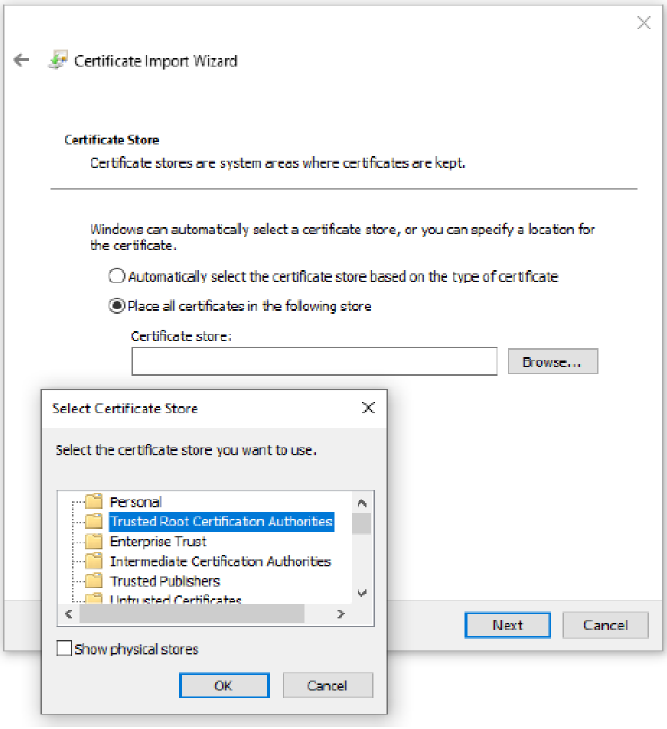
This section describes the functionality for support of multi client architecture in Asset Link for EcoStruxure Control Expert, allowing one or more users to use Asset Link from a remote client of System Platform IDE connecting to same galaxy or different galaxies.

System Platform Client Machine Setup

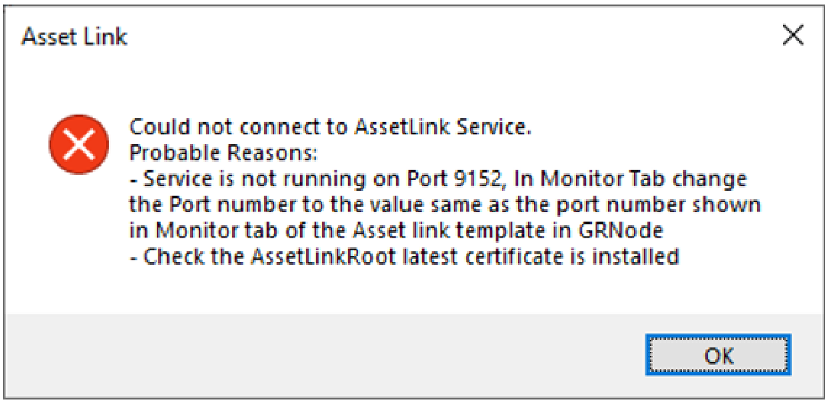
First, follow the steps in the table below to use Asset Link on a System Platform client machine.

Step	Action
1	Copy the Asset Link root certificate `AssetLinkRoot.pfx` to the client machine or share the certificates folder location in GR Node and access it from the client machine. **NOTE:** The default location is `C:\ProgramData\Schneider Electric\Ecostruxure Control Expert -Asset Link\Certificates`.
2	Double **click** on the `AssetLinkRoot.pfx` file and select **Install Certificate**.

3	Select Store Location as **Local Machine**.  Certificate Import Wizard **Welcome to the Certificate Import Wizard** This wizard helps you copy certificates, certificate trust lists, and certificate revocation lists from your disk to a certificate store. A certificate, which is issued by a certification authority, is a confirmation of your identity and contains information used to protect data or to establish secure network connections. A certificate store is the system area where certificates are kept. Store Location ○ Current User ● Local Machine To continue, click Next.
4	In the **password** field, enter the **password** created during certificate installation from Asset Link installer in server GR node machine.  Certificate Import Wizard **Private key protection** To maintain security, the private key was protected with a password. Type the password for the private key. Password: □ Display Password Import options: □ Enable strong private key protection. You will be prompted every time the private key is used by an application if you enable this option. □ Mark this key as exportable. This will allow you to back up or transport your keys at a later time. □ Protect private key using virtualized-based security(Non-exportable) ☑ Include all extended properties.

5	In Certificate store, click **Browse** and select the **Trusted Root Certificate Authorities** as shown.  Certificate Import Wizard Certificate Store Certificate stores are system areas where certificates are kept. Windows can automatically select a certificate store, or you can specify a location for the certificate. ○ Automatically select the certificate store based on the type of certificate ● Place all certificates in the following store Certificate store: Select Certificate Store Select the certificate store you want to use. Personal Trusted Root Certification Authorities Enterprise Trust Intermediate Certification Authorities Trusted Publishers Untrusted Certificates □ Show physical stores
6	Click **Next** and **Finish** for a complete certificate installation.

Then, post installing certificates open System Platform IDE then open Asset Link template. At opening of Asset Link template it tries to connect to Asset Link service running in GR Node. If unsuccessful, Asset Link detects an error and displays a message with possible causes. Open the **Monitor Tab** and verify whether the value in the port number is same as the value entered during the Asset Link installation in GR node (This installation time entered port number is displayed in the Asset Link template **Monitor Tab** in GR node machine). Enter the correct port number and click on Test Connection button.



Asset Link

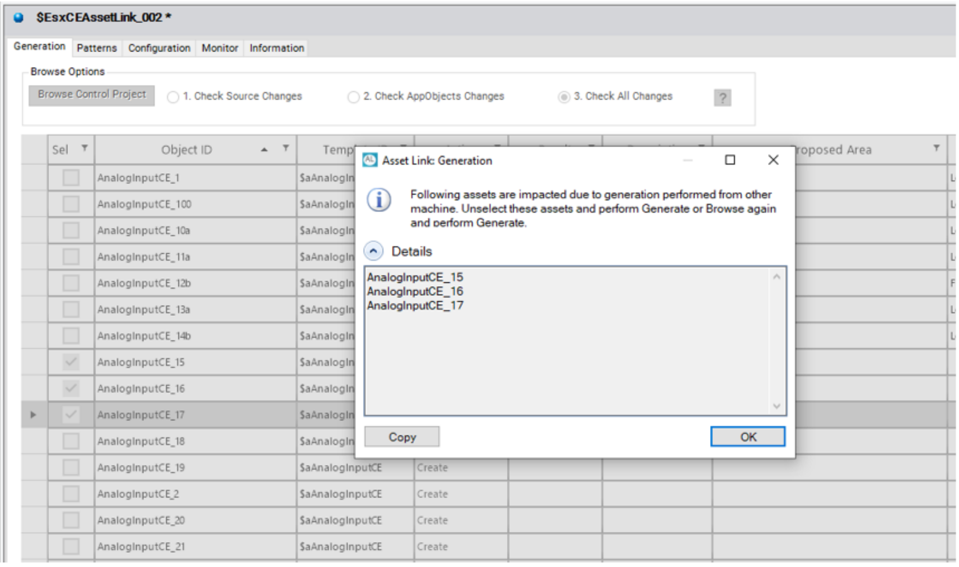
Could not connect to AssetLink Service.
Probable Reasons:
- Service is not running on Port 9152, In Monitor Tab change the Port number to the value same as the port number shown in Monitor tab of the Asset link template in GRNode
- Check the AssetLinkRoot latest certificate is installed

NOTE: If you reinstalled the Asset Link or if you had to repair Asset Link in GR node machine, then the certificates are reinstalled. Whenever the certificates are reinstalled, it is required to reinstall the new generated `AssetLinkRoot.pfx` in the remote client machine for Asset Link in remote System Platform machine to communicate with the server.

Browse and Generation operations in Multi Client Environment

To work with Asset Link in multiple machines, open the Asset Link derived template $EsxCEAssetLink_001 in one client machine (such as a GR node) and the other derived template $EsxCEAssetLink_002 in another machine, and configure either the same source control project xsy or a different xsy file in the Asset Link template **Configuration Tab**.

If the same asset has been selected on multiple machines and both Generate actions have been initiated simultaneously, a message is displayed prompting you to either browse again to refresh the asset action status or unselect the overlapping asset and proceed with the generation.



○ 1. Check Source Changes
○ 2. Check AppObjects Changes
● 3. Check All Changes

Sel	Object ID	Temp		roposed Area
□	AnalogInputCE_1	$aAnalogIn		
□	AnalogInputCE_100	$aAnalogIn		
□	AnalogInputCE_10a	$aAnalogIn		
□	AnalogInputCE_11a	$aAnalogIn		
□	AnalogInputCE_12b	$aAnalogIn		
□	AnalogInputCE_13a	$aAnalogIn		
□	AnalogInputCE_14b	$aAnalogIn		
✓	AnalogInputCE_15	$aAnalogIn		
✓	AnalogInputCE_16	$aAnalogIn		
✓	AnalogInputCE_17	$aAnalogIn		
□	AnalogInputCE_18	$aAnalogIn		
□	AnalogInputCE_19	$aAnalogInputCE	Create	
□	AnalogInputCE_2	$aAnalogInputCE	Create	
□	AnalogInputCE_20	$aAnalogInputCE	Create	
□	AnalogInputCE_21	$aAnalogInputCE	Create	

Following assets are impacted due to generation performed from other machine. Unselect these assets and perform Generate or Browse again and perform Generate.

AnalogInputCE_15
AnalogInputCE_16
AnalogInputCE_17

Troubleshooting Tips:

- During **Generation** from the System Platform GR node PC or remote client PC, if it could not communicate with the Asset Link service, then an error is displayed at the end of the **Generation**. Restart the PC and open Asset Link. Further, verify if the service port is blocked.
- If the error message "Could not connect to Asset Link Service" is displayed at the opening of Asset Link, or during **Browse** or **Generation** operations, verify whether the Asset Link Service is running on the GR Node PC and that the Service Port, by default 9150 configured during installation, is not blocked. If the Asset Link Service is stopped, restart the service, and if still it does not run, contact your system administrator.

Work Flow

Process Overview

Getting Started

Follow these steps in System Platform IDE:

Step	Action
1	Import the ASP templates you want to use: • AppObjects and Infrastructure Templates as needed • Asset Link 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 Asset Link tool.
2	Create the Galaxy communications infrastructure: • Create the required number of instances for ASP OPCClient DIOs (device integration objects) to communicate with PAC Modicon controllers through OFS. **NOTE:** Refer to the *PlantStruxure General Purpose Library for ASP Supervision Services User Guide* for more information. • In high-availability systems, create the required number of instances of ASP Redundant DIOs. **NOTE:** Redundancy is the duplication of critical components to increase the availability of the system. The servers maintain synchronized data through a dedicated network. If the primary server is inoperable, the backup server takes over where primary server left off. Configuration of redundancy is established through a check box on the AppEngine.
3	Create Asset Link objects in Galaxy: • Use Asset Link from the derived templates in $EsxCEAssetLink. (You do not need to create AppObject instances, as they are only used during the build process, and are not required for deployment or execution in the runtime.) • In most of cases, you can create a single connector for each controller.
4	Open the Asset Link object from the System Platform IDE: • Asset Link automatically connects to the Galaxy from which it was instantiated. • The login credentials are requested if Galaxy is secured by password.

Configuration Process

Select the configuration options:

Step	Action
1	Open the Asset Link **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 **Machine Expert**. • The protocol **Dynamic DNP3** is applicable only for **SCADAPack**.

Step	Action
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	The **Configure Scan Groups** option provides the capability to configure the I/O source attributes of an asset with different scan group, so that these attributes can be scanned in different frequency based on your requirement. Click **Configure Scan Groups** to display the scan groups used in the patterns. For more details refer to Configure Scan Groups, page 37.
8	Enter an address in the **OI Address Reference** field. **NOTE:** Refer to the parameter descriptions for the **Configuration** tab, page 34.
9	In the **Optional Prefix of the AppObject Tagname** field, enter a prefix. **NOTE:** Refer to the parameter descriptions for the **Configuration** tab, page 34.
10	Click the browse button (**...**) associated with the **Patterns Path** field and navigate to the folder that contains the respective Pattern.
11	Click the browse button (**...**) associated with the **Pattern Project** field to view control projects that were exported with the updated variables. **NOTE:** The selection of a pattern project is required only when you need new patterns for generation or obsolete patterns that need to be updated.
12	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:** • If **Process Expert** is selected as a source, associate only one System in **Ecostruxure Process Expert** for AVEVA System Platform (EPEforASP) to one galaxy in the AVEVA System Platform. If multiple systems from EPEforASP are associated to the same galaxy, any identically named objects coming from two different systems from EPEforASP create issues in the AVEVA System Platform.
13	Click the browse button (**...**) associated with the **Application Project (Optional)** field to configure the application .xml file. **NOTE:** • This field can be edited for all **Sources** except for **Process Expert**. When you configure the control project for process expert, by default, the application XML is present in the same location as the control project, hence it is automatically filled. For more details refer to Application Project, page 59. • Ensure that the source files have the necessary inputs that must be generated into ASP attributes.
14	Navigate to the **Patterns** tab. Open a pattern for editing and modify as required. Save and close the pattern. **NOTE:** For more details refer to Pattern Editor, page 46.
15	Navigate to the **Generation** tab and click **Browse Control Project**. Once the object are loaded, select and generate the required objects by **clicking Generate Objects**. **NOTE:** For more details refer to Generation tab, page 61.

DNP3 Configuration

The **DNP3 Protocol** has nine parameters out of which three parameters **Telemetry Server Name, Outstation** (refer to Configuration Tab, page 34), and **Hierarchy** refer to Configure Hierarchy, page 64) must be configured by you and the remaining are configured internally by Asset Link.

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

Step	Action
1	Open the Object Editor by double clicking on the generated object in the Model tab of **ASP**.
2	Open the **Attributes Tab** in the Object Editor.
3	Click ▼ and then click All.
4	Enable the **I/O Read** or **I/O Read/Write** under the **Enabled Features** list.
5	Select an **Attribute** and view its details. This **Attribute** contains 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 updated 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 Asset Link **Patterns** tab, click the **Create Pattern** button to open the **Create Pattern** dialog box.
3	Fill in 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 Asset Link **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 Asset Link 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.

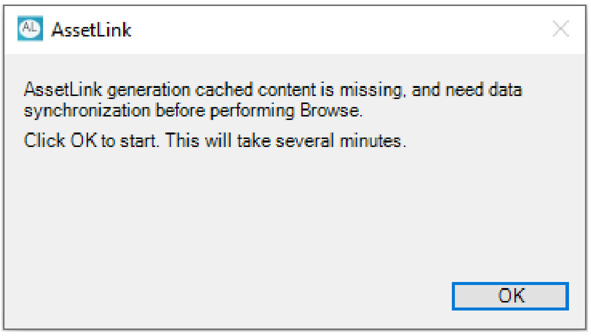
Update Galaxy

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

Step	Action
1	Open the Asset Link **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.

Restore Galaxy

Restoring a backup Galaxy having Asset Link generated assets using the Asset Link version 4.0 requires the synchronization of the Asset Link generation cache content. Hence, for the first time when you open Asset Link and perform **Browse**, the following message is shown to synchronize the generation cache content.



AssetLink generation cached content is missing, and need data synchronization before performing Browse.

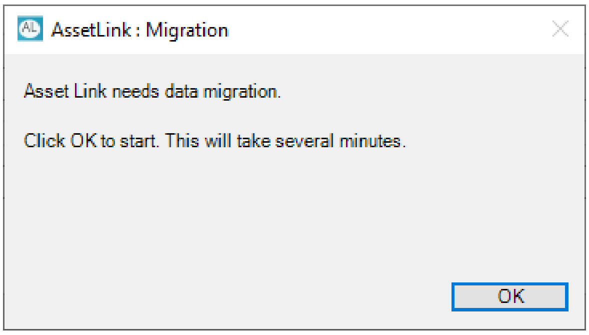
Click OK to start. This will take several minutes.

Migration

Upgrade to Asset Link 4.0	Asset Link 1.0.0/ 1.0.1 user	Asset Link 1.0.2 SP2 user (Imported $EsxCEAsset-Link without deleting existing 1.0.1 template)	Asset Link 1.0.2 SP2 user (Imported $EsxCEAsset-Link after deleting existing 1.0.1 template)	Asset Link 1.0.2 SP2 user (Imported $EsxCEAsset-Link without deleting existing 1.0.2 SP2 template)	Asset Link 2.0.0/ 2.0.1/ 2.0.2 user	Asset Link 2.1/ 2.2 user	Asset Link 3.0/ 3.0.1 user
Installation Procedure	Uninstall Asset Link 1.0.0/ 1.0.1 and install Asset Link 4.0.	Uninstall Asset Link 1.0.2 SP2 and install Asset Link 4.0.	Uninstall Asset Link 1.0.2 SP2 and install Asset Link 4.0.	Uninstall Asset Link 1.0.2 SP2 and install Asset Link 4.0.	Uninstall Asset Link 2.0.0/ 2.0.1/ 2.0.2 and install Asset Link 4.0.	Install Asset Link 4.0 for upgrading.	Install Asset Link 4.0 for upgrading.
Using Asset Link-Template	Re-create Asset Link object in System Platform IDE. Delete existing Asset Link 1.0.0/ 1.0.1 template and Import Asset Link 4.0 template - `$EsxCEAsset-Link.aaPKG`.	Re-create Asset Link object in System Platform IDE. Delete existing Asset Link 1.0.1 template and Import Asset Link 4.0 template - `$EsxCEAsset-Link.aaPKG`.	Import Asset Link 4.0 template - `$EsxCEAs-setLink.aaPKG`. (above existing 1.0.2 SP2 template)	Import Asset Link 4.0 template - `$EsxCEAs-setLink.aaPKG`. (above existing 1.0.2 SP2 template)	Import Asset Link 4.0 template - `$EsxCEAs-setLink.aaPKG`. (Above the existing 2.0.0/ 2.0.1/ 2.0.2 template)	Import Asset Link 4.0 template - `$EsxCEAs-setLink.aaPKG`. (Above the existing 2.1/ 2.2 template)	Import Asset Link 4.0 template - `$EsxCEAs-setLink.aaPKG`. (Above the existing 3.0/ 3.0.1 template)
Impact	Since previous version of Asset Link template is deleted, re-enter settings of Asset Link. No impact in already generated ASP AppObjects.	Since previous version of Asset Link template is deleted, re-enter settings of Asset Link. No impact in already generated ASP AppObjects.	Since it is new import of Asset Link, there is no impact. No impact in already generated ASP AppObjects.	Since it is imported from a previous version of Asset Link templates, there is no impact. No impact in already generated ASP AppObjects.			

Migrating to Asset Link V4.0

Importing the new `aapkg` delivered with Asset Link 4.0, on top of versions prior to Asset Link 2.2.2 (template code base version 5888) requires migration. Opening of the Asset link template object prompts to do the migration.



Asset Link needs data migration.

Click OK to start. This will take several minutes.

Patterns migration: The patterns files migration takes place while migrating from any of the previous versions of Asset Link to Asset Link 4.0. In case there are files that are not migrated then refer to the log file to identify these files.

> **NOTE:** Backward compatibility: If you want to use the pattern files of the previous versions then use the backup files created during migration of patterns.

Asset Link Operations

Introduction

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

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 36. **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 are discarded as the entire area is 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 36. **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.

Parameter		Description
DIO Name[**]		This is the name of the OPCClient template instance.
Scan Group[**]		This option provides the capability to configure the I/O source attributes of an asset with different scan groups, so that these attributes can be scanned in different time intervals based on your requirement. Click **Configure Scan Groups** to display the scan groups used in the patterns. For more details refer to **Configure Scan Groups**, page 37.
OI Address Reference[**]		The referenced address combines these components: • *OI gateway:* This is the Operation Integration (OI) gateway that you configure 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 39.
DNP3 Configuration[***]	**Telemetry Server Name**	It is an editable field in which you enter the name of the Telemetry server.
	Outstation Full Name	It is an editable field in which you 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 Asset Link 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:

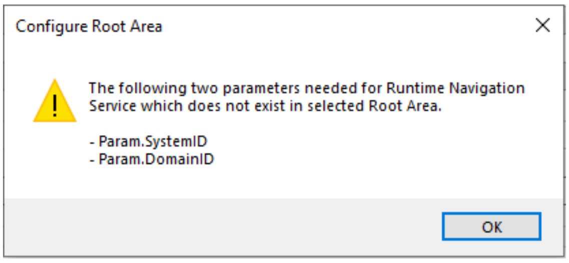
Step	Action
1	Configure the **Galaxy Settings** in the **Configuration** tab by providing the pattern folder location and variable file.
2	Select **Process Expert** in the **Source** and select the type of **Root Area**.
3	Open **Generation** tab and click **Browse Control Project** to update the objects 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 there is already a **Root Area** in the galaxy, but the selected **Root Area** is different from the existing one then the hierarchy is deleted and a new one is created depending on the plant hierarchy. A pop-up message displays the elapsed time for recreating the hierarchy.

Area:

Step	Action
1	Configure the **Galaxy Settings** in the **Configuration** tab by providing the pattern folder location and 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** are created. • If there is already an **Area** in the galaxy, but the selected **Area** is different from the existing **Area** then the 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.



Configure Root Area

The following two parameters needed for Runtime Navigation Service which does not exist in selected Root Area.

- Param.SystemID
- Param.DomainID

- If the area is renamed in the source, then a new area is created.

Update/ Move

Root Area:

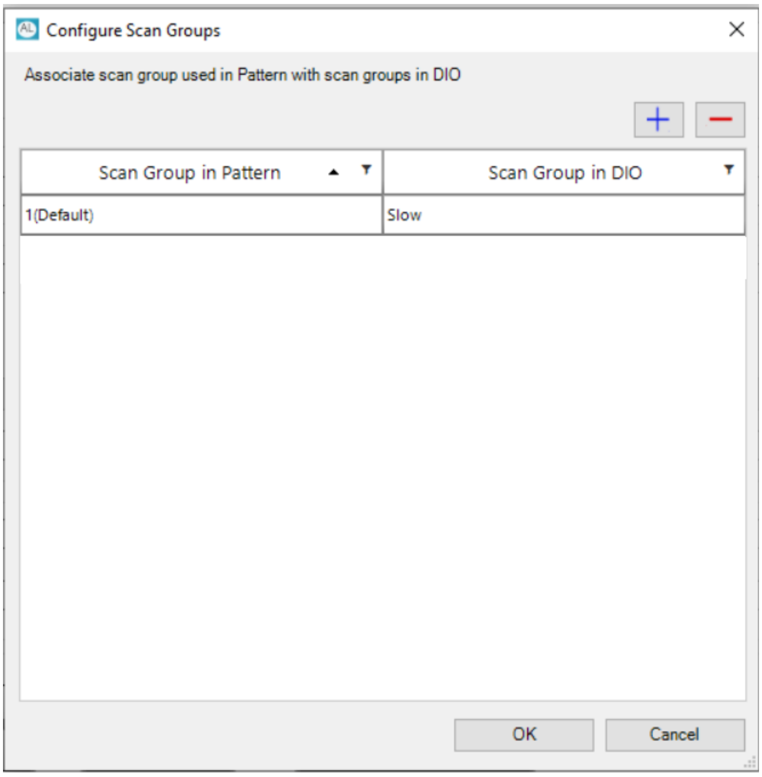
Step	Action
1	Configure the **Galaxy Settings** in the **Configuration** tab by providing the pattern folder location and variable file.
2	Select **Process Expert** in the **Source** and select the type of **Root Area**.
3	Open **Generation** and click **Browse Control Project** to update the objects 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** are moved to the respective folders.

Configure Scan Groups

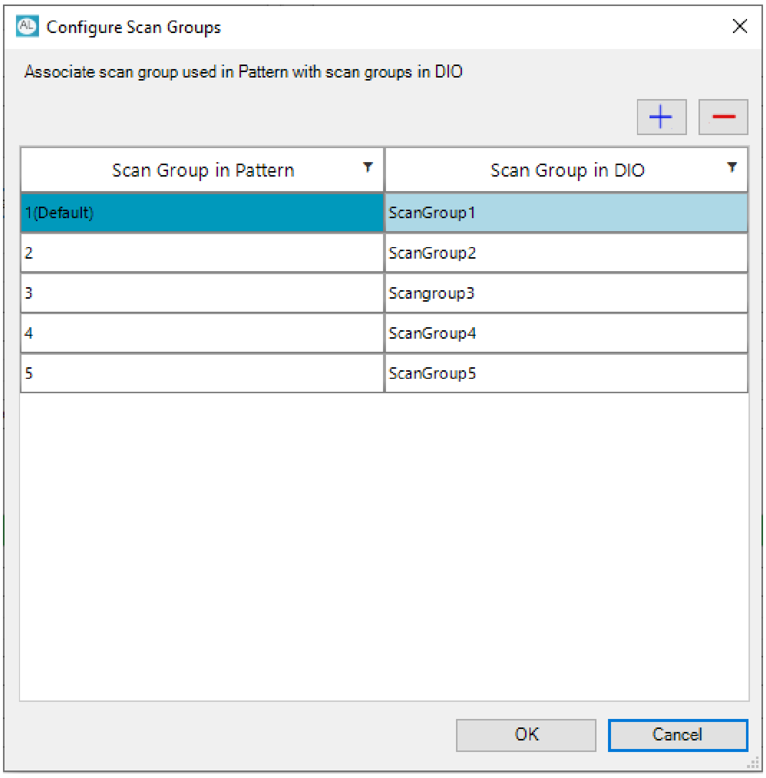
System Platform attributes can be generated from Asset Link to read/ write their values in different time intervals using multiple scan groups. The **Configure Scan Groups** helps to assign the multiple scan groups of DIO gateway into different attributes of an asset. Clicking on **Configure Scan Groups** in the configuration tab opens the pop-up as shown in the following figure.



Associate scan group used in Pattern with scan groups in DIO

Scan Group in Pattern	Scan Group in DIO
1(Default)	Slow

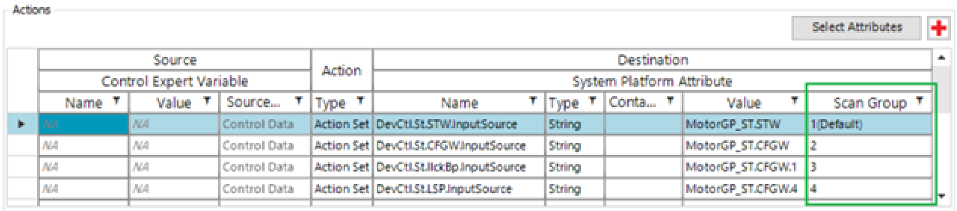
By default it displays the first available associated scan group. By default the first available scan group in DIO is associated with the default scan group in Pattern which is 1. Additionally you can associate the same DIO's scan group with more than one Pattern's scan group. Click + to add a new scan group association and − to delete an existing scan group. You configure up to ten scan groups and associate them in **OPC Client**.



Associate scan group used in Pattern with scan groups in DIO

Scan Group in Pattern	Scan Group in DIO
1(Default)	ScanGroup1
2	ScanGroup2
3	Scangroup3
4	ScanGroup4
5	ScanGroup5

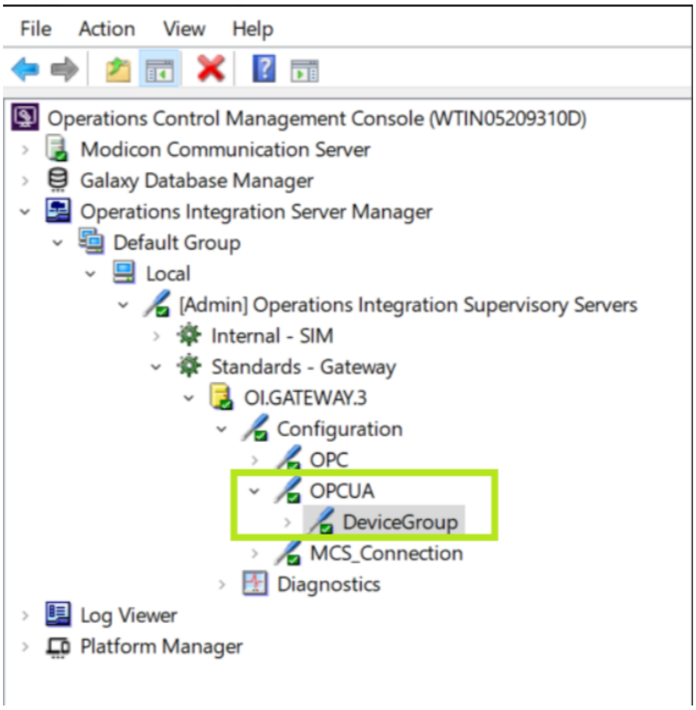
Based on the association done in **Configure Scan Groups** and the scan group position set in the pattern as shown in the figure below, perform **Browse** and **Generate**, then the generated attribute values has the configured scan groups, so that these attributes are read at different intervals.



Source			Action	Destination				
Control Expert Variable				System Platform Attribute				
Name	Value	Source...	Type	Name	Type	Conta...	Value	Scan Group
NA	NA	Control Data	Action Set	DevCtl.St.STW.InputSource	String		MotorGP_ST.STW	1(Default)
NA	NA	Control Data	Action Set	DevCtl.St.CFGW.InputSource	String		MotorGP_ST.CFGW	2
NA	NA	Control Data	Action Set	DevCtl.St.IlckBp.InputSource	String		MotorGP_ST.CFGW.1	3
NA	NA	Control Data	Action Set	DevCtl.St.LSP.InputSource	String		MotorGP_ST.CFGW.4	4

OI Address Reference Example

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

Step	Action
1	Using the Windows search utility (found either in the Start menu or in the Task bar), find and open the **Operation Control Management Console**.
2	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:  File Action View Help Operations Control Management Console (WTIN05209310D) Modicon Communication Server Galaxy Database Manager Operations Integration Server Manager Default Group Local [Admin] Operations Integration Supervisory Servers Internal - SIM Standards - Gateway OI.GATEWAY.3 Configuration OPC OPCUA DeviceGroup MCS_Connection Diagnostics Log Viewer Platform Manager
3	In the navigation tree, click **DeviceGroup** to open the **Node Type** dialog box.
4	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.

  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, 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 Asset Link 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 Asset Link 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 the trial period has expired, it displays a message “Asset Link trial period expired. Activate a license and restart Asset Link Tool ”. For more details, refer to Product Licensing, page 16.

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 Asset Link scans for variables: • *.xsy* : Files with this extension correspond to the selection of a Control Expert project in the **Source** field. • *.xml* file of same name as *.xsy* file has to be available in the same path for Plant Model creation, page 71. • *.xml* : Files with this extension correspond to the selection of a Machine Expert project in the **Source** field .
Application Project (Optional):	This field contains the path where the *.xml* application file is present. This file is used during **Browse** operation to **Generate** the system platform attribute values based on application data, as per the configuration of the **Action Retrieve** elements of the pattern. For more details refer to Application Project, page 59.

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:

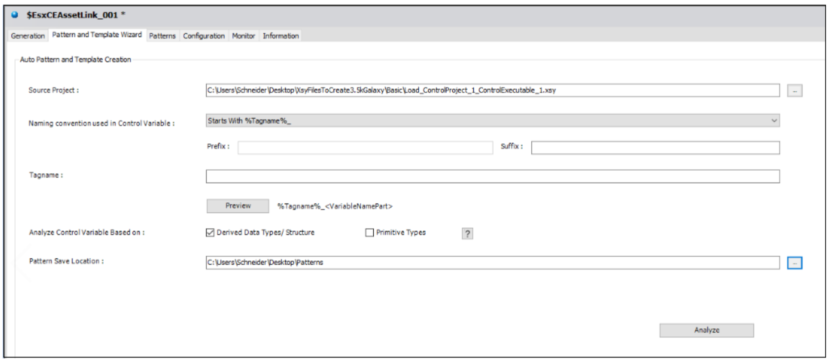
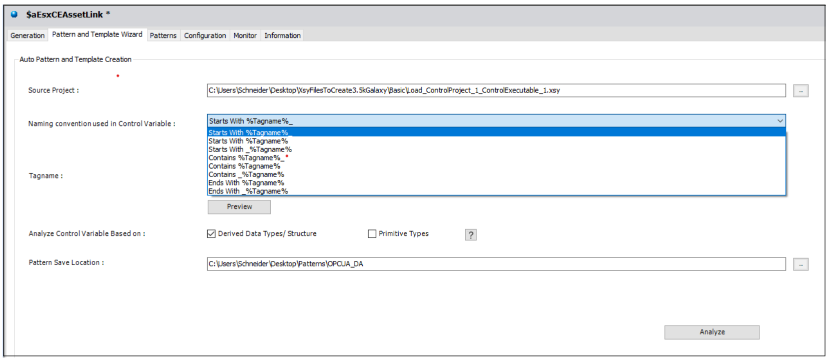
- Template must be upgraded with the **Object Wizard**
- The **Patterns** must be upgraded with the **Object Wizard** parameters.

Pattern and Template Wizard

This section describes the functionality for auto creation of AVEVA System Platform templates and patterns based on DDT / Structure variable or Primitive Type in Control logic.

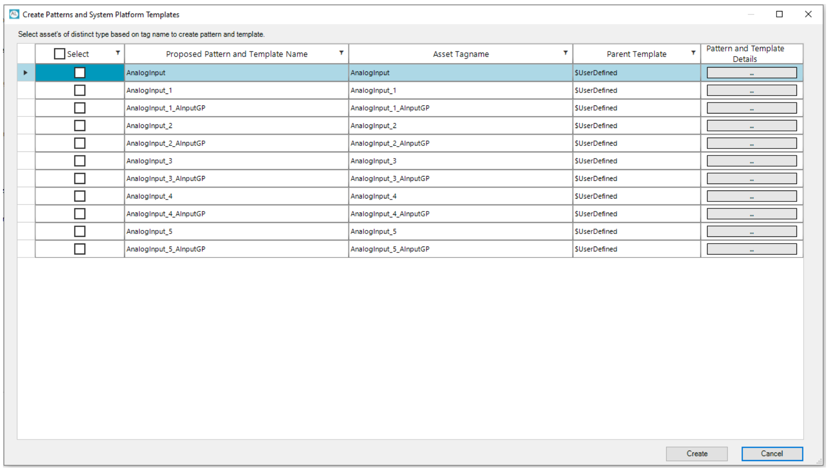
The following table describes how to configure inputs for creation of system template and patterns:

Steps	Action
1	Open Asset Link for EcoStruxure Control Expert.
2	Navigate to **Pattern and Template Wizard** Tab.

3	Browse for **Source Project** and select Source file. Asset Link provides the flexibility to browse the source project from different source type. By default, the **Source Project** is prepopulated based on **Configuration Tab** inputs. You can still change these if necessary.  $EsxCEAssetLink_001 * Auto Pattern and Template Creation Source Project : C:\Users\Schneider\Desktop\XsyFilesToCreate3.5kGalaxy\Basic\Load_ControlProject_1_ControlExecutable_1.xsy Naming convention used in Control Variable : Starts With %Tagname%_ Prefix : Suffix : Tagname : %Tagname%_<VariableNamePart> Analyze Control Variable Based on : ☑ Derived Data Types/ Structure ☐ Primitive Types Pattern Save Location : C:\Users\Schneider\Desktop\Patterns Generation Pattern and Template Wizard Patterns Configuration Monitor Information Preview ? Analyze
4	Configure the **Naming Convention** of control variable as per the naming convention followed in the control project, to identify the assets from the control project. Select the naming convention used in control variable.  $aEsxCEAssetLink * Auto Pattern and Template Creation Source Project : C:\Users\Schneider\Desktop\XsyFilesToCreate3.5kGalaxy\Basic\Load_ControlProject_1_ControlExecutable_1.xsy Naming convention used in Control Variable : Starts With %Tagname%_ Tagname : Analyze Control Variable Based on : ☑ Derived Data Types/ Structure ☐ Primitive Types Pattern Save Location : C:\Users\Schneider\Desktop\Patterns\OPCUA_DA Generation Pattern and Template Wizard Patterns Configuration Monitor Information Starts With %Tagname%_ Starts With %Tagname% Starts With _%Tagname% Contains %Tagname%_ Contains %Tagname% Contains _%Tagname% Ends With %Tagname% Ends With _%Tagname% Preview ? Analyze
5	`Prefix` and `Suffix` text input controls are enabled based on the naming conventions control variable requirements. `Prefix` and `Suffix` are case sensitive. • The control variable name **Starts** with Tagname then the suffix is enabled. • The control variable name **Ends** with Tagname then the prefix is enabled. • The control variable name **Contains** with Tagname then the prefix and suffix are enabled. **NOTE:** If the control variable name **Contains** %Tagname% or %Tagname%_, then the prefix is mandatory. If the variable name **Contains** _%Tagname% then the `suffix` is mandatory.
6	The *TagName* must be configured for creating the pattern and template based on well-known tags or user preferred tags. The *TagName* is optional field if the selected **Naming Convention** used in the control variable contains an underscore, otherwise the field is mandatory. More than one `TagName` can be configured by using comma (,) separator. *Tagname* is case sensitive. For example, if the control variable name for one of the analog input assets is AnalogInput1_AInputGP_ST, then enter the TagName as AnalogInput1. The system finds this asset AnalogInput1 from the .XSY file and displays it in the **Create Patterns and Templates** dialog box on click of the **Create** button.
7	A **Preview** is available to visualize the configured naming convention of control variables in source file.
8	Analyze the control variable based on Derived Data Type / Structure or Primitive Type or both. Based on the selection, the control variables of these types are analyzed to find the assets to create pattern and template. **NOTE:** By default DDT / Structure type is selected.
9	The **Pattern Save Location** is pre-populated based on the inputs from the **Configuration Tab**. Patterns created are saved to this configured location. You can modify these if necessary.

10

Click on **Analyze**.



Create Patterns and System Platform Templates

Select asset's of distinct type based on tag name to create pattern and template.

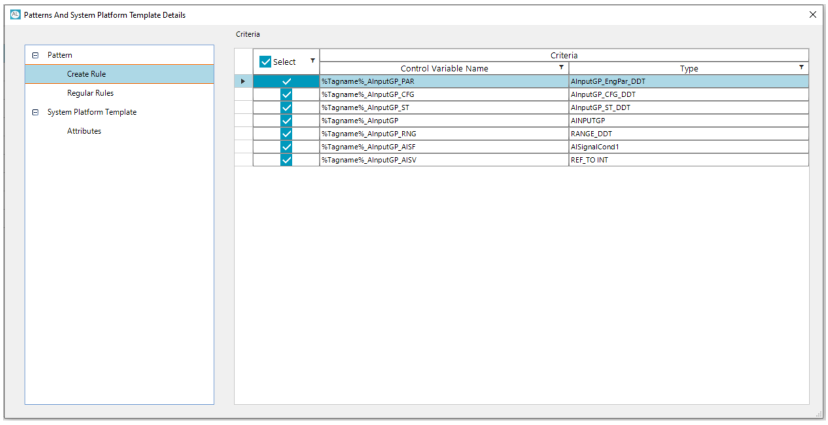
□ Select	Proposed Pattern and Template Name	Asset Tagname	Parent Template	Pattern and Template Details
□	AnalogInput	AnalogInput	$UserDefined	...
□	AnalogInput_1	AnalogInput_1	$UserDefined	...
□	AnalogInput_1_AInputGP	AnalogInput_1_AInputGP	$UserDefined	...
□	AnalogInput_2	AnalogInput_2	$UserDefined	...
□	AnalogInput_2_AInputGP	AnalogInput_2_AInputGP	$UserDefined	...
□	AnalogInput_3	AnalogInput_3	$UserDefined	...
□	AnalogInput_3_AInputGP	AnalogInput_3_AInputGP	$UserDefined	...
□	AnalogInput_4	AnalogInput_4	$UserDefined	...
□	AnalogInput_4_AInputGP	AnalogInput_4_AInputGP	$UserDefined	...
□	AnalogInput_5	AnalogInput_5	$UserDefined	...
□	AnalogInput_5_AInputGP	AnalogInput_5_AInputGP	$UserDefined	...

Result: It displays a dialog box with the assets from the source control project file based on configuration inputs and displays these assets in the summary to create patterns and templates.

In above screenshot, it displays AnalogInput_1, AnalogInput_2 and AnalogInput_3, in which only one asset has to be selected as all are of the same `Analog Input` type asset and therefore only one pattern and template are created. After creation of pattern and template, performing the **Browse** operation identifies all the assets based on the created pattern rules. **Generation** creates instances of that template type in the System Platform.

11

Select the asset and click on **Pattern and Template Details**, to view and edit the pattern and template content before the creation of the pattern and template.



Patterns And System Platform Template Details

Criteria

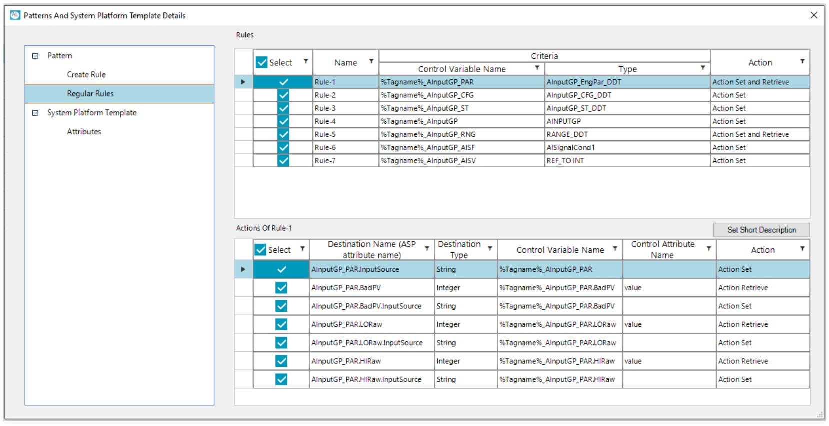
☑ Select	Criteria	
	Control Variable Name	Type
✓	%Tagname%_AInputGP_PAR	AInputGP_EngPar_DDT
✓	%Tagname%_AInputGP_CFG	AInputGP_CFG_DDT
✓	%Tagname%_AInputGP_ST	AInputGP_ST_DDT
✓	%Tagname%_AInputGP	AINPUTGP
✓	%Tagname%_AInputGP_RNG	RANGE_DDT
✓	%Tagname%_AInputGP_AISF	AISignalCond1
✓	%Tagname%_AInputGP_AISV	REF_TO INT

Result: This displays the pattern content which are rules to identify the assets based on control variable names in the source file, and set the Actions in each rule to assign the control variable data to the corresponding System Platform attributes.

Patterns and Template Details - **Create Rule** :

In **Create Rule** to exclude or include the control variable. Based on condition this control variable exists in the source project, assets are identified.

Patterns and Template Details - **Regular Rules** :



Patterns And System Platform Template Details

Rules

☑ Select	Name	Criteria		Action
		Control Variable Name	Type	
✓	Rule-1	%Tagname%_AInputGP_PAR	AInputGP_EngPar_DDT	Action Set and Retrieve
✓	Rule-2	%Tagname%_AInputGP_CFG	AInputGP_CFG_DDT	Action Set
✓	Rule-3	%Tagname%_AInputGP_ST	AInputGP_ST_DDT	Action Set
✓	Rule-4	%Tagname%_AInputGP	AINPUTGP	Action Set
✓	Rule-5	%Tagname%_AInputGP_RNG	RANGE_DDT	Action Set and Retrieve
✓	Rule-6	%Tagname%_AInputGP_AISF	AISignalCond1	Action Set
✓	Rule-7	%Tagname%_AInputGP_AISV	REF_TO INT	Action Set

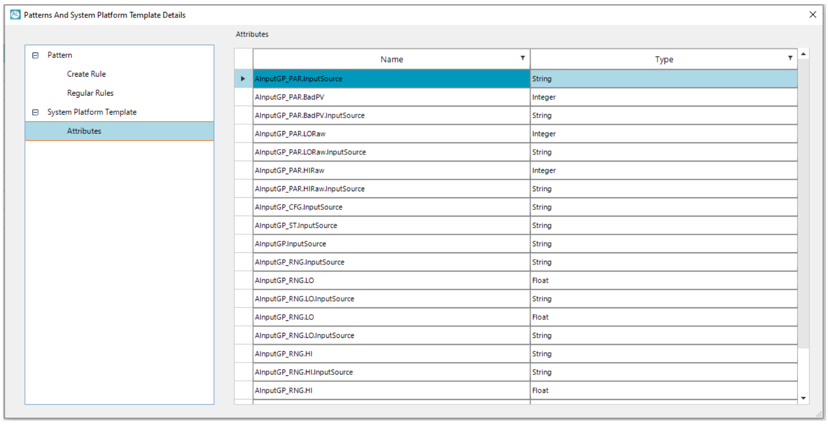
Actions Of Rule-1

☑ Select	Destination Name (ASP attribute name)	Destination Type	Control Variable Name	Control Attribute Name	Action
✓	AInputGP_PAR.InputSource	String	%Tagname%_AInputGP_PAR		Action Set
✓	AInputGP_PAR.BadPV	Integer	%Tagname%_AInputGP_PAR.BadPV	value	Action Retrieve
✓	AInputGP_PAR.BadPV.InputSource	String	%Tagname%_AInputGP_PAR.BadPV		Action Set
✓	AInputGP_PAR.LORaw	Integer	%Tagname%_AInputGP_PAR.LORaw	value	Action Retrieve
✓	AInputGP_PAR.LORaw.InputSource	String	%Tagname%_AInputGP_PAR.LORaw		Action Set
✓	AInputGP_PAR.HIRaw	Integer	%Tagname%_AInputGP_PAR.HIRaw	value	Action Retrieve
✓	AInputGP_PAR.HIRaw.InputSource	String	%Tagname%_AInputGP_PAR.HIRaw		Action Set

<table>
<tr><td></td><td>

- In **Regular Rules**, exclude or include one or more regular rules.
- Exclude or include the **Action set** or **Action Retrieve** for each regular rule. This also excludes or includes the system platform attribute.
- Set the destination short description attribute by clicking on **Set Short Description** button in the actions grid, to add an **Action Retrieve** from source control variable.
- Edit the destination system platform attribute name. Edit the system platform attribute types from string to other AVEVA System Platform data types, except choice and options types, in the rule actions grid.

Patterns and Template Details - **Template Attributes** :



Patterns And System Platform Template Details

Attributes

<table>
<tr><th>Name</th><th>Type</th></tr>
<tr><td>AInputGP_PAR.InputSource</td><td>String</td></tr>
<tr><td>AInputGP_PAR.BadPV</td><td>Integer</td></tr>
<tr><td>AInputGP_PAR.BadPV.InputSource</td><td>String</td></tr>
<tr><td>AInputGP_PAR.LORaw</td><td>Integer</td></tr>
<tr><td>AInputGP_PAR.LORaw.InputSource</td><td>String</td></tr>
<tr><td>AInputGP_PAR.HIRaw</td><td>Integer</td></tr>
<tr><td>AInputGP_PAR.HIRaw.InputSource</td><td>String</td></tr>
<tr><td>AInputGP_CFG.InputSource</td><td>String</td></tr>
<tr><td>AInputGP_ST.InputSource</td><td>String</td></tr>
<tr><td>AInputGP.InputSource</td><td>String</td></tr>
<tr><td>AInputGP_RNG.InputSource</td><td>String</td></tr>
<tr><td>AInputGP_RNG.LO</td><td>Float</td></tr>
<tr><td>AInputGP_RNG.LO.InputSource</td><td>String</td></tr>
<tr><td>AInputGP_RNG.LO</td><td>Float</td></tr>
<tr><td>AInputGP_RNG.LO.InputSource</td><td>String</td></tr>
<tr><td>AInputGP_RNG.HI</td><td>String</td></tr>
<tr><td>AInputGP_RNG.HI.InputSource</td><td>String</td></tr>
<tr><td>AInputGP_RNG.HI</td><td>Float</td></tr>
</table>

The attributes listed in template are the attributes selected in the **Action set** or **Action Retrieve** actions of each regular rule. Auto creation of template will create with the attributes displayed in this dialog box.

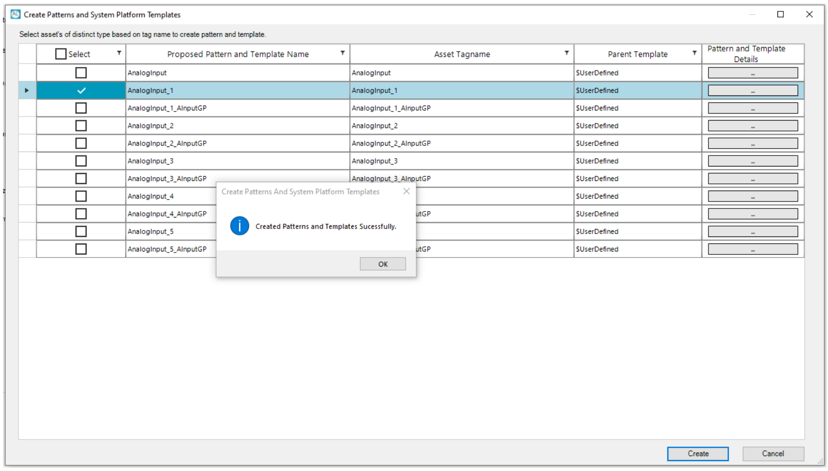
</td></tr>
<tr><td>12</td><td>

Edit the proposed template and pattern name to a unique name. The proposed pattern and template name column is editable, and when a new name is entered, the system validates that the name is unique within the grid and also verifies whether the pattern name and the template name already exist.

The proposed pattern and template name is derived from the asset name. Select the **Parent Template** from which to create the derived template. By default, **$UserDefined** is the parent template.

</td></tr>
<tr><td>13</td><td>

Select the desired **Proposed Pattern Name and Template**, and then click on **Create.**



Create Patterns and System Platform Templates

Select asset's of distinct type based on tag name to create pattern and template.

<table>
<tr><th>□ Select</th><th>Proposed Pattern and Template Name</th><th>Asset Tagname</th><th>Parent Template</th><th>Pattern and Template Details</th></tr>
<tr><td>□</td><td>AnalogInput</td><td>AnalogInput</td><td>$UserDefined</td><td>...</td></tr>
<tr><td>✓</td><td>AnalogInput_1</td><td>AnalogInput_1</td><td>$UserDefined</td><td>...</td></tr>
<tr><td>□</td><td>AnalogInput_1_AInputGP</td><td>AnalogInput_1_AInputGP</td><td>$UserDefined</td><td>...</td></tr>
<tr><td>□</td><td>AnalogInput_2</td><td>AnalogInput_2</td><td>$UserDefined</td><td>...</td></tr>
<tr><td>□</td><td>AnalogInput_2_AInputGP</td><td>AnalogInput_2_AInputGP</td><td>$UserDefined</td><td>...</td></tr>
<tr><td>□</td><td>AnalogInput_3</td><td>AnalogInput_3</td><td>$UserDefined</td><td>...</td></tr>
<tr><td>□</td><td>AnalogInput_3_AInputGP</td><td>AnalogInput_3_AInputGP</td><td>$UserDefined</td><td>...</td></tr>
<tr><td>□</td><td>AnalogInput_4</td><td></td><td>$UserDefined</td><td>...</td></tr>
<tr><td>□</td><td>AnalogInput_4_AInputGP</td><td>utGP</td><td>$UserDefined</td><td>...</td></tr>
<tr><td>□</td><td>AnalogInput_5</td><td></td><td>$UserDefined</td><td>...</td></tr>
<tr><td>□</td><td>AnalogInput_5_AInputGP</td><td>utGP</td><td>$UserDefined</td><td>...</td></tr>
</table>

Create Patterns And System Platform Templates

Created Patterns and Templates Sucessfully.

Result: The Pattern and Templates are created .

</td></tr>
<tr><td>14</td><td>

Navigate to **Generation Tab.** Perform a **Browse** to identify assets based on the auto created pattern and template. Then you can **Generate** assets. For more details, refer to Generation Tab, page 61.

</td></tr>
</table>

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 available patterns are loaded. Instances related to the templates are available for generation.

For license activation process, refer to Product Licensing, page 16.

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

Buttons

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

Button	Description
Refresh Patterns	Click this button to reload the configured pattern files.
Select All	Click this button to select all patterns.
Unselect All	Click this button to deselect all patterns.
Save Button	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. (For example:`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 **Create Pattern** dialog box. Another dialog box **Select Variable(s) for Creation Rule** appears. Once the variable is selected the pattern opens in the Pattern Editor, page 46.

Button	Description
Add Pattern	Click **Add Pattern** to open the **Add Pattern file(s)** dialog box. Select the pattern files (.xml format) and click **Open**. **Result**: The pattern files are copied to the configured location and is 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** appears which indicates the detection of the source pattern *.xml* file. • Clicking on **Keep Changes** opens the pattern in the Pattern Editor, page 46. **NOTE:** The **Save** button is enabled only if the **Version** is changed or changes are made to the pattern. • Clicking on **New Pattern** opens the pattern in the Pattern Editor, page 46. **NOTE:** The **Save** button is enabled only after a new name is provided to the pattern.

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 **Pattern File**.
Last Modification Date Time	This column displays the file saved time stamp for the **Pattern File**. Applies when the pattern file is generated or modified by the Asset Link tool or manually saved.
Valid	This box is selected (checked) only when the pattern has passed the validation process, including the alignment to the expected pattern schema.
Select	Select this check box to scan the **Pattern File**.
	Clear this check box to remove the file from the scan.
Exists	This box is activated 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**.
View	Click **View** to open the pattern file in the read-only mode. The content of the pattern file is displayed in the Pattern Editor, page 46.
Edit	Click **Edit** to open the pattern file in the edit mode. The content of the pattern file is displayed in the Pattern Editor, page 46. **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 opens 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.

These columns contain values only after you enter a valid path in the **Patterns Path** field on the **Configuration** tab, page 40.

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** creates 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 is displayed asking for confirmation, click **Yes** to delete.

Pattern Editor

When you click **View** or **Edit**, the pattern editor displays 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:

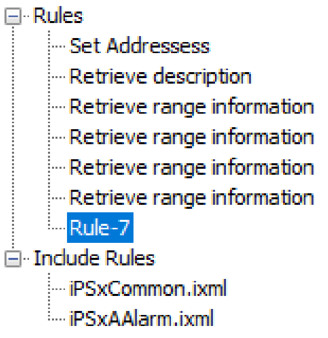
Overview

Section		**Description**
Template		Displays the name of the pattern file.
click on Other details button	**Version**	Displays the version of the pattern file.
	Created using	Displays the **TagName** used in the XSY file.
	Last Modified	Displays the most recent time stamp of the modified pattern file.
	Prefix	Displays the number of characters of **TagName** that has to be used for pattern variable creation.

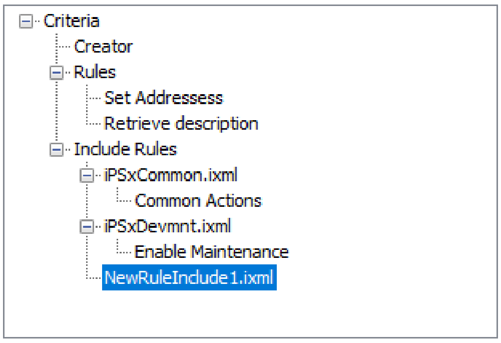
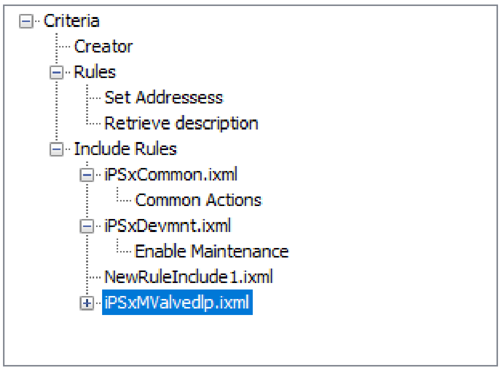
Description

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

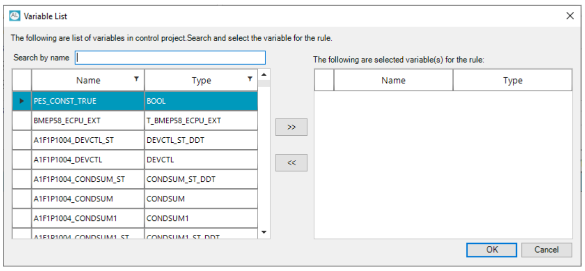
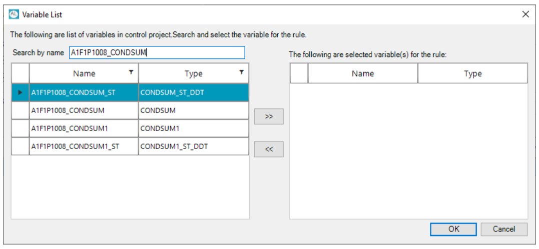
Pattern Rules

Section	Description
Add Rule	Click **Add Rule** to add a new rule in the pattern file. The newly added rule is displayed under the **Rules** node.  Rules Set Addressess Retrieve description Retrieve range information Retrieve range information Retrieve range information Retrieve range information Rule-7 Include Rules iPSxCommon.ixml iPSxAAlarm.ixml When you click on rule, the associated details are 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 is displayed asking for confirmation, click **Yes** to delete. For more details, refer to Pattern Files, page 76.

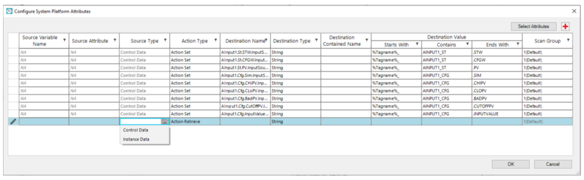
Pattern Rules (Continued)

Section	Description
Add Include Rule	Click **Add Include Rule** to add a new include rule in the pattern file. **Include Rules** are common rules which are created in one pattern, can be included in the multiple patterns. A pop-up is displayed asking to choose if the rule has to be added as **New** or **Existing**. • **New**: Provide a name for the rule and then the rule is created as illustrated in the graphic below.  Criteria Creator Rules Set Addressess Retrieve description Include Rules iPSxCommon.ixml Common Actions iPSxDevmnt.ixml Enable Maintenance NewRuleInclude1.ixml • **Existing**: Select an existing **Include Rule** from the location, it is saved in your system. The selected rule is added as illustrated in the graphic below.  Criteria Creator Rules Set Addressess Retrieve description Include Rules iPSxCommon.ixml Common Actions iPSxDevmnt.ixml Enable Maintenance NewRuleInclude1.ixml iPSxMValvedlp.ixml When you click on **Include Rule**, the associated details are displayed in the **Rule Details** section. 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 is displayed asking for confirmation, click **Yes** to delete. For more details, refer to Pattern Files, page 76.
Criteria	Expand **Criteria** to display: • **Creator:** Creation rule of the pattern file. The creation rule supports Token based Asset creation and identification. See Token Mechanism, page 55 for more details. • **Rules:** Existing rules of the pattern file. • **Include Rules:** List of existing include rules of the pattern file.

Rule Details

Section		Description
Rule Name		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.
	Negate	• Enabled: If the criteria for the rule element is not satisfied, 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 <template name>	**Variables** tab	This tab displays the following details: • **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: • **Select Vaiables**: Click on Select Variables to display the list of variables for the selected template. Variable List The following are list of variables in control project.Search and select the variable for the rule. Search by name Name · Type PES_CONST_TRUE · BOOL BMEP58_ECPU_EXT · T_BMEP58_ECPU_EXT A1F1P1004_DEVCTL_ST · DEVCTL_ST_DDT A1F1P1004_DEVCTL · DEVCTL A1F1P1004_CONDSUM_ST · CONDSUM_ST_DDT A1F1P1004_CONDSUM · CONDSUM A1F1P1004_CONDSUM1 · CONDSUM1 A1F1P1004_CONDSUM1_ST · CONDSUM1_ST_DDT The following are selected variable(s) for the rule: Name · Type ◦ 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** Variable List The following are list of variables in control project.Search and select the variable for the rule. Search by name A1F1P1008_CONDSUM Name · Type A1F1P1008_CONDSUM_ST · CONDSUM_ST_DDT A1F1P1008_CONDSUM · CONDSUM A1F1P1008_CONDSUM1 · CONDSUM1 A1F1P1008_CONDSUM1_ST · CONDSUM1_ST_DDT The following are selected variable(s) for the rule: Name · 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**.

Rule Details (Continued)

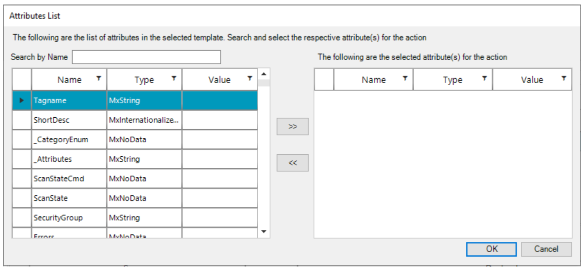
Section		Description
		• **Clear Value**: To clear the existing values of the selected row in the grid, right click on the row and click **Clear Value**. • **Delete Row**: To delete the row from the grid, right click on the row and click **Delete**. A message is displayed asking for confirmation, click **Yes** to delete.
Select variable from control project to identify <template name>	**Action** tab	This tab displays the following details: • **Source Variable** :  ◦ **Source Variable Name**: This column allows you to enter the attribute name of the source variable (for example, EcoStruxure Control Expert, EPEforASP, EcoStruxure Machine Expert and RTU). ◦ **Source Attribute**: This column allows you to enter values in this field when the **Action Type** is set to **Action Retrieve**. Example: If Source Variable is "HI" the source attribute can be "comment". **Source Type**: This column allows you to set the source type to either **Control Data** or **Instance Data**. The **Source Type** can be set to **Control Data** to retrieve the data from the control project. It can be set to **Instance Data** to retrieve the data from the application project. This column/field can be edited only when the **Action Type** is set to **Action Retrieve**. • **Action Type**: 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 columns (**Source Variable Name**, **Source Attribute**, **Source Type**) are not applicable since there is a direct value to be set. ◦ **Action Retrieve**: This option retrieves the value from the Source Variable .xsy file if the **Source Type** is set to **Control Data**, where as it retrieves the value from the Application project xml if the **Source Type** is set to **Instance Data**, for a particular attribute **Name** and fill the respective values in the **System Platform Attribute**. For more details on retrieving data from application project .*xml* refer to Application Project, page 59. **NOTE:** For more details refer to Object Wizard Support, page 58 • **Destination**: ◦ **System Platform Attribute**: – **Destination Name**: This column displays the name of the system platform attribute. – **Destination Type**: This column displays the type of system platform attribute. – **Destination Contained Name**: This column represents the Contained Name of the instance. Contained name is the name of the parent instance. – **Destination Value**: This column displays the value of the system platform attribute. The **Destination Value** contains **Start With** , **Contains** and **Ends With** provides the naming convention flexibility when entering the control variable name for **Action Type** set to **Action Set** and **Destination Type** set to **String**. For **Destination Type** set to **Boolean**, the value in this column "TRUE" or "FALSE" can be entered in any one of these 3 fields of **Destination Value**. – **Scan Group**: This column displays the associated scan group for the respective system platform attribute. You can set the scan group only when the **Action Type** is **Action Set**. In case of **Action**

Configure System Platform Attributes

Source Variable Name	Source Attribute	Source Type	Action Type	Destination Name	Destination Type	Destination Contained Name	Destination Value			Scan Group
							Starts With	Contains	Ends With	
N/A	N/A	Control Data	Action Set	AInput1.St.STW.InputS...	String		%Tagname%_	AINPUT1_ST	.STW	1(Default)
N/A	N/A	Control Data	Action Set	AInput1.St.CFGW.Input...	String		%Tagname%_	AINPUT1_ST	.CFGW	1(Default)
N/A	N/A	Control Data	Action Set	AInput1.St.PV.InputSou...	String		%Tagname%_	AINPUT1_ST	.PV	1(Default)
N/A	N/A	Control Data	Action Set	AInput1.Cfg.Sim.InputS...	String		%Tagname%_	AINPUT1_CFG	.SIM	1(Default)
N/A	N/A	Control Data	Action Set	AInput1.Cfg.CHiPV.Inp...	String		%Tagname%_	AINPUT1_CFG	.CHIPV	1(Default)
N/A	N/A	Control Data	Action Set	AInput1.Cfg.CLoPV.Inp...	String		%Tagname%_	AINPUT1_CFG	.CLOPV	1(Default)
N/A	N/A	Control Data	Action Set	AInput1.Cfg.BadPV.Inp...	String		%Tagname%_	AINPUT1_CFG	.BADPV	1(Default)
N/A	N/A	Control Data	Action Set	AInput1.Cfg.CutOffPV.I...	String		%Tagname%_	AINPUT1_CFG	.CUTOFFPV	1(Default)
N/A	N/A	Control Data	Action Set	AInput1.Cfg.InputValue...	String		%Tagname%_	AINPUT1_CFG	.INPUTVALUE	1(Default)
			Action Retrieve		String					1(Default)

Rule Details (Continued)

<table>
<tr><th colspan="2">Section</th><th>Description</th></tr>
<tr><td></td><td></td><td><b>Retrieve</b>, the <b>Scan Group</b> is disabled since it is not applicable. For scan group number in pattern and its association with scan group in DIO, refer to the Configure Scan Groups, page 37. If the scan group in the pattern is not associated with any scan group name in the configuration tab then it uses the default scan group during generation.</td></tr>
<tr><td></td><td></td><td>
You can add a new row in the grid:
<ul>
<li><b>Select Attributes</b>: Click on Select Attributes to display the list of attributes for the selected template.



Attributes List

The following are the list of attributes in the selected template. Search and select the respective attribute(s) for the action

Search by Name

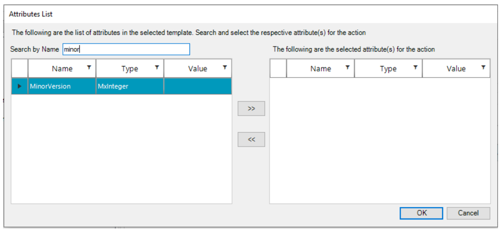
<table>
<tr><th>Name</th><th>Type</th><th>Value</th></tr>
<tr><td>Tagname</td><td>MxString</td><td></td></tr>
<tr><td>ShortDesc</td><td>MxInternationalize...</td><td></td></tr>
<tr><td>_CategoryEnum</td><td>MxNoData</td><td></td></tr>
<tr><td>_Attributes</td><td>MxString</td><td></td></tr>
<tr><td>ScanStateCmd</td><td>MxNoData</td><td></td></tr>
<tr><td>ScanState</td><td>MxNoData</td><td></td></tr>
<tr><td>SecurityGroup</td><td>MxString</td><td></td></tr>
<tr><td>Errors</td><td>MxNoData</td><td></td></tr>
</table>

The following are the selected attribute(s) for the action

<table>
<tr><th>Name</th><th>Type</th><th>Value</th></tr>
</table>

<ul>
<li>You can either type the attribute in the <b>Search</b> bar or scroll down to find the attribute.</li>
<li>Select the attribute and click >> to add it to the list of selected attributes.</li>
<li>Select the attribute and click << to remove it from the list of the selected attributes.

<b>NOTE:</b>
<ul>
<li>The attributes are listed based on the <b>Name</b>, <b>Type</b> and <b>Value</b>



Attributes List

The following are the list of attributes in the selected template. Search and select the respective attribute(s) for the action

Search by Name minor

<table>
<tr><th>Name</th><th>Type</th><th>Value</th></tr>
<tr><td>MinorVersion</td><td>MxInteger</td><td></td></tr>
</table>

The following are the selected attribute(s) for the action

<table>
<tr><th>Name</th><th>Type</th><th>Value</th></tr>
</table>

</li>
<li>Ensure that the necessary services are enabled so that the correct outputs are obtained for the objects.</li>
</ul>
</li>
</ul>
</li>
<li><b>Add new Action</b>: 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.</li>
</ul>
</td></tr>
<tr><td></td><td></td><td>
You can perform the following actions in the grid:
<ul>
<li><b>Copy</b>: To copy the selected row from the grid, right click on the row and click <b>Copy</b>.</li>
<li><b>Paste</b>: To paste the copied row, right click on the row and click <b>Paste</b>.</li>
<li><b>Edit</b>: To edit the values of the selected row in the grid, right click on the row and click <b>Edit</b>.</li>
<li><b>Clear Value</b>: To clear the existing values of the selected row in the grid, right click on the row and click <b>Clear Value</b>.</li>
<li><b>Delete Row</b>: To delete the row from the grid, right click on the row and click <b>Delete</b>. A message is displayed asking for confirmation, click <b>Yes</b> to delete.</li>
</ul>
</td></tr>
</table>

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 opens another pop-up window which displays the hidden columns which can be dragged and dropped 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 resizes the columns depending on their contents.

Save Changes

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

Naming Convention Flexibility

This feature describes the flexibility in the **Pattern Editor** to:

- Configure the naming convention of control variable as per the control project naming convention standards, to identify the assets from the control project source file.
- Configure the system platform input source attribute with support to add prefix/suffix for the control variable name.

The scenarios below illustrate the naming convention variable for different Control Project:

Examples for different control project variable naming conventions and their configuration in Asset Link pattern naming convention fields of **Starts With**, **Contains**, **Ends With**.

For example in editor of the Analog Input pattern, configure as follows in the default naming convention grid.

1. If asset is **Analog Input** and the control variable name has `Tagname` at **start**

   Example: AI1_AInputGP_ST where AI1 is the Tagname.

    - **Starts With** as `%Tagname%_`
    - **Contains** as empty field
    - **Ends With** as empty field

2. If asset is **Analog Input** and Control variable name has `Tagname` at **End.**

   Example: AInputGP_ST_AI1 where AI1 is the Tagname.

    - **Starts With** as empty field
    - **Contains** as empty field
    - **Ends With** as `_%Tagname%`

3. Control variable name contains prefix and suffix .

   Example: Plant_AI1_AInputGP_ST_Area where AI1 is the Tagname.

   The `Plant` and `Area` are the prefix and suffix of the variable.

   - **Starts With** as `Plant_`
   - **Contains** as `%Tagname%_`
   - **Ends With** as `_Area`

4. Control variable name has `Tagname` at **start** and has suffix .

   Example: AI1_AInputGP_ST_Area where AI1 is the Tagname.

   `Area` is the suffix of the variable.

   - **Starts With** as `%Tagname%_`
   - **Contains** as empty field
   - **Ends With** as `_Area`

5. Control variable name has prefix

   Example: Plant_AInputGP_ST_AI1 where AI1 is the Tagname.

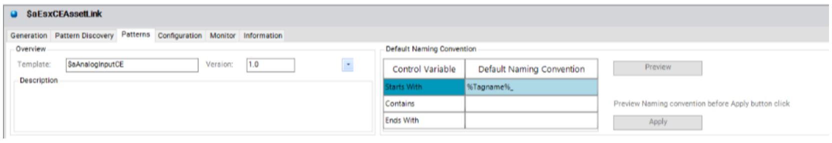
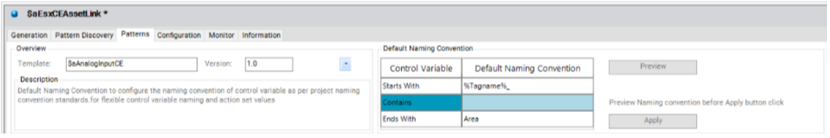
   - **Starts With** as `Plant_`
   - **Contains** as empty field
   - **Ends With** as `_%Tagname%`

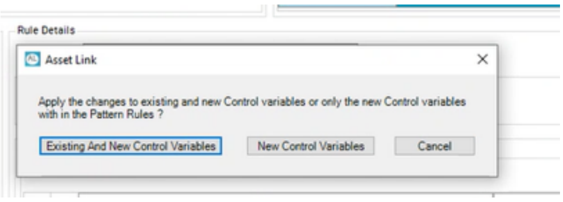
Default Naming Convention validation rules:

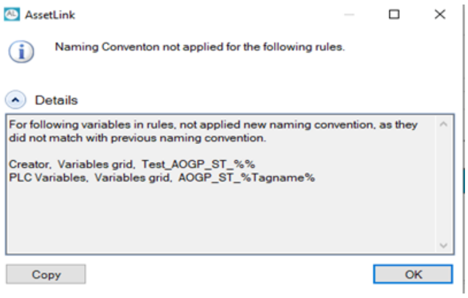
- Naming Convention must have `%Tagname%` in one of the Control Variable fields.

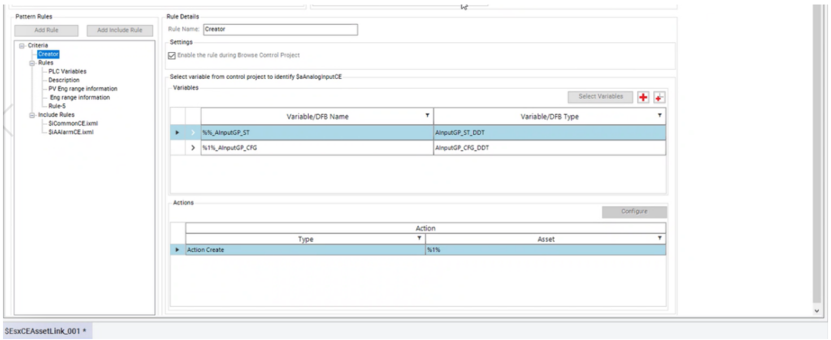
  NOTE: Click on **Preview** or **Apply** to validate the control variable naming convention configuration. If the edited naming convention does not comply with the validation rules, it displays an error message.

The following table describes the naming convention flexibility .

Step	Action
1	Open Asset Link Template in System Platform.
2	Navigate to the **Patterns** tab. **NOTE:** Based on the naming convention in the source .xsy file, the other patterns should be formed using the naming convention described in the steps below.
3	Select one of the **Pattern** then Click on **Edit** and open the **Pattern Editor**.
4	Configure the default naming convention for the control variable as required. By default, the `%Tagname%_` is configured under **Starts With**.  $aEsxCEAssetLink Template: $aAnalogInputCE Version: 1.0 Default Naming Convention Control Variable / Default Naming Convention Starts With: %Tagname%_ Contains: Ends With: Preview Naming convention before Apply button click Generation Pattern Discovery Patterns Configuration Monitor Information Overview Description Preview Apply
5	Configure the default naming convention as per requirements. For more details refer to Naming Convention.  $aEsxCEAssetLink * Template: $aAnalogInputCE Version: 1.0 Description Default Naming Convention to configure the naming convention of control variable as per project naming convention standards.for flexible control variable naming and action set values Default Naming Convention Control Variable / Default Naming Convention Starts With: %Tagname%_ Contains: Ends With: Area Preview Naming convention before Apply button click Generation Pattern Discovery Patterns Configuration Monitor Information Overview Preview Apply
6	Click on **Preview** to visualize the configured default naming convention.

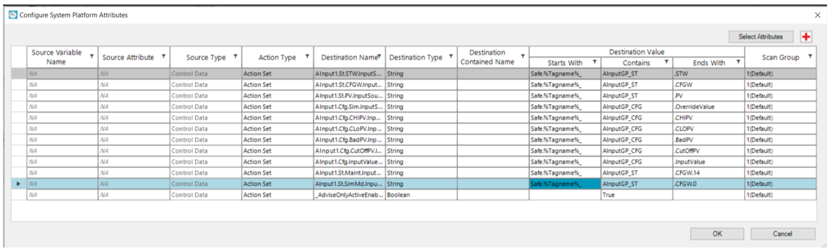
<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>7</td><td>
Click on <b>Apply</b> to apply this naming convention to the control variables used in this <b>pattern</b> rules.
The naming convention is to be applied in the pattern rules for both the new and existing control variables, or exclusively for the new variable.

Rule Details
Asset Link
Apply the changes to existing and new Control variables or only the new Control variables with in the Pattern Rules ?

</td></tr>
<tr><td>8</td><td>
Click on <b>Existing And New Control Variables</b> to apply the configured naming convention to the existing control variables in the pattern rules and also applies to the new control variables added in existing rules or in new rules.
<b>NOTE:</b> If the variables could not be modified with new naming convention as they do not match with previous configured naming convention, then it is displayed as follows

AssetLink
Naming Conventon not applied for the following rules.
Details
For following variables in rules, not applied new naming convention, as they did not match with previous naming convention.
Creator, Variables grid, Test_AOGP_ST_%%
PLC Variables, Variables grid, AOGP_ST_%Tagname%

<b>Result:</b> The control variables are updated with the new name standard after the modifications are applied.
</td></tr>
<tr><td>9</td><td>
Navigate to one of the <b>Pattern Rule</b> and click on <b>Configure</b> in Action Grid.

Pattern Rules

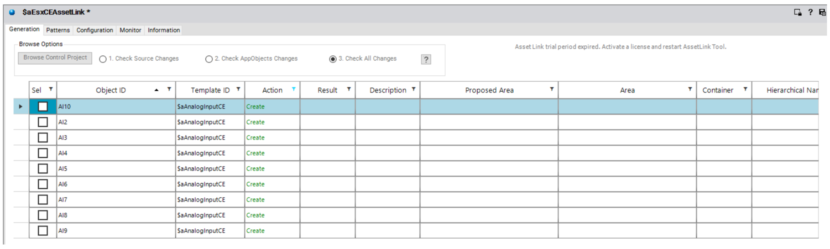
Criteria
- Creator
- Rules
  - PLC Variables
  - Description
  - PV Eng range information
  - Eng range information
  - Rule-5
- Include Rules
  - $iCommonCE.ixml
  - $iAAlarmCE.ixml
Rule Details
Rule Name: Creator
Settings
☑ Enable the rule during Browse Control Project
Select variable from control project to identify $aAnalogInputCE
Variables

<table>
<tr><th>Variable/DFB Name</th><th>Variable/DFB Type</th></tr>
<tr><td>%%_AInputGP_ST</td><td>AInputGP_ST_DDT</td></tr>
<tr><td>%1%_AInputGP_CFG</td><td>AInputGP_CFG_DDT</td></tr>
</table>
Actions

<table>
<tr><th colspan="2">Action</th></tr>
<tr><th>Type</th><th>Asset</th></tr>
<tr><td>Action Create</td><td>%1%</td></tr>
</table>
$EsxCEAssetLink_001 *
<b>Result:</b> The System Platform attributes are displayed in a new pop-up window for action configuration with the default naming convention values already applied.
</td></tr>
<tr><td>10</td><td>
Optionally, with Action type set to <b>Action Set</b>, you can prefix/suffix the destination value as per the control variable names in the Destination Value columns. For example, to access safety-related control variables, <b>Starts With</b> column value can be prefixed with "Safe."

Configure System Platform Attributes

<table>
<tr><th>Source Variable Name</th><th>Source Attribute</th><th>Source Type</th><th>Action Type</th><th>Destination Name</th><th>Destination Type</th><th>Destination Contained Name</th><th colspan="3">Destination Value</th><th>Scan Group</th></tr>
<tr><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th>Starts With</th><th>Contains</th><th>Ends With</th><th></th></tr>
<tr><td>NA</td><td>NA</td><td>Control Data</td><td>Action Set</td><td>AInput1.St.STW.InputS...</td><td>String</td><td></td><td>Safe.%Tagname%_</td><td>AInputGP_ST</td><td>.STW</td><td>1(Default)</td></tr>
<tr><td>NA</td><td>NA</td><td>Control Data</td><td>Action Set</td><td>AInput1.St.CFGW.Input...</td><td>String</td><td></td><td>Safe.%Tagname%_</td><td>AInputGP_ST</td><td>.CFGW</td><td>1(Default)</td></tr>
<tr><td>NA</td><td>NA</td><td>Control Data</td><td>Action Set</td><td>AInput1.St.PV.InputSou...</td><td>String</td><td></td><td>Safe.%Tagname%_</td><td>AInputGP_ST</td><td>.PV</td><td>1(Default)</td></tr>
<tr><td>NA</td><td>NA</td><td>Control Data</td><td>Action Set</td><td>AInput1.Cfg.Sim.InputS...</td><td>String</td><td></td><td>Safe.%Tagname%_</td><td>AInputGP_CFG</td><td>.OverrideValue</td><td>1(Default)</td></tr>
<tr><td>NA</td><td>NA</td><td>Control Data</td><td>Action Set</td><td>AInput1.Cfg.CHiPV.Inp...</td><td>String</td><td></td><td>Safe.%Tagname%_</td><td>AInputGP_CFG</td><td>.CHIPV</td><td>1(Default)</td></tr>
<tr><td>NA</td><td>NA</td><td>Control Data</td><td>Action Set</td><td>AInput1.Cfg.CLoPV.Inp...</td><td>String</td><td></td><td>Safe.%Tagname%_</td><td>AInputGP_CFG</td><td>.CLOPV</td><td>1(Default)</td></tr>
<tr><td>NA</td><td>NA</td><td>Control Data</td><td>Action Set</td><td>AInput1.Cfg.BadPV.Inp...</td><td>String</td><td></td><td>Safe.%Tagname%_</td><td>AInputGP_CFG</td><td>.BadPV</td><td>1(Default)</td></tr>
<tr><td>NA</td><td>NA</td><td>Control Data</td><td>Action Set</td><td>AInput1.Cfg.CutOffPV.I...</td><td>String</td><td></td><td>Safe.%Tagname%_</td><td>AInputGP_CFG</td><td>.CutOffPV</td><td>1(Default)</td></tr>
<tr><td>NA</td><td>NA</td><td>Control Data</td><td>Action Set</td><td>AInput1.Cfg.InputValue...</td><td>String</td><td></td><td>Safe.%Tagname%_</td><td>AInputGP_CFG</td><td>.InputValue</td><td>1(Default)</td></tr>
<tr><td>NA</td><td>NA</td><td>Control Data</td><td>Action Set</td><td>AInput1.St.Maint.Input...</td><td>String</td><td></td><td>Safe.%Tagname%_</td><td>AInputGP_ST</td><td>.CFGW.14</td><td>1(Default)</td></tr>
<tr><td>NA</td><td>NA</td><td>Control Data</td><td>Action Set</td><td>AInput1.St.SimMd.Inpu...</td><td>String</td><td></td><td>Safe.%Tagname%_</td><td>AInputGP_ST</td><td>.CFGW.0</td><td>1(Default)</td></tr>
<tr><td>NA</td><td>NA</td><td>Control Data</td><td>Action Set</td><td>_AdviseOnlyActiveEnab...</td><td>Boolean</td><td></td><td></td><td>True</td><td></td><td>1(Default)</td></tr>
</table>

</td></tr>
<tr><td>11</td><td>To save these modifications, click on <b>Save</b>.</td></tr>
</table>

<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>12</td><td>Go to <b>Generation</b> tab and click <b>Browse Control Project.</b>


○ 1. Check Source Changes ○ 2. Check AppObjects Changes ● 3. Check All Changes

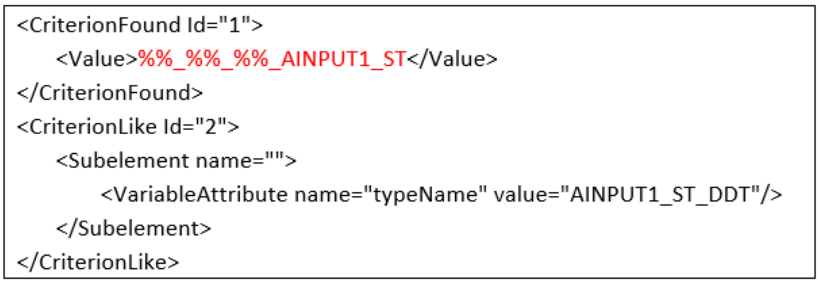
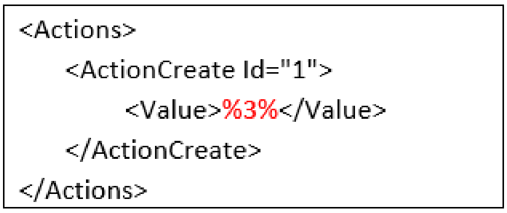
Asset Link trial period expired. Activate a license and restart AssetLink Tool.

<table>
<tr><th>Sel</th><th>Object ID</th><th>Template ID</th><th>Action</th><th>Result</th><th>Description</th><th>Proposed Area</th><th>Area</th><th>Container</th><th>Hierarchical Nar</th></tr>
<tr><td>□</td><td>AI10</td><td>$aAnalogInputCE</td><td>Create</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>□</td><td>AI2</td><td>$aAnalogInputCE</td><td>Create</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>□</td><td>AI3</td><td>$aAnalogInputCE</td><td>Create</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>□</td><td>AI4</td><td>$aAnalogInputCE</td><td>Create</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>□</td><td>AI5</td><td>$aAnalogInputCE</td><td>Create</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>□</td><td>AI6</td><td>$aAnalogInputCE</td><td>Create</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>□</td><td>AI7</td><td>$aAnalogInputCE</td><td>Create</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>□</td><td>AI8</td><td>$aAnalogInputCE</td><td>Create</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>□</td><td>AI9</td><td>$aAnalogInputCE</td><td>Create</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

<b>NOTE: Browse of Control Project</b> detects and displays the assets and with proposing the asset name, as per control variable condition in the pattern creation rule.</td></tr>
<tr><td>13</td><td><b>Select</b>/<b>Select All</b> the necessary assets and perform <b>Generate</b>.</td></tr>
</table>

Token Mechanism

Token mechanism allows you to configure the pattern for detection of multiple variable for single asset using the Pattern editor. Once the pattern is configured, clicking on **Browse Control Project** will identify the assets within same variable based on the input from the 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.

<table>
<tr><th>Stage</th><th>Description</th></tr>
<tr><td>1</td><td>The variable from the .xsy file is
Variable name =<code>FI1001_AINPUT1_AI4_AINPUT1_ST</code>
TypeName = <code>AINPUT1_ST_DDT</code></td></tr>
<tr><td>2</td><td>The criteria in the pattern file would be

<pre>
<CriterionFound Id="1">
    <Value>%%_%%_%%_AINPUT1_ST</Value>
</CriterionFound>
<CriterionLike Id="2">
    <Subelement name="">
        <VariableAttribute name="typeName" value="AINPUT1_ST_DDT"/>
    </Subelement>
</CriterionLike>
</pre></td></tr>
<tr><td>3</td><td>If the variable satisfies the criteria from the pattern then the tokens are:
%1% = FI1001
%2% = AINPUT1
%3% = AI4</td></tr>
<tr><td>4</td><td>With the use of the token values in the previous stage, the asset is identified from the <b>Actions</b> in the Creation Rule.

<pre>
<Actions>
    <ActionCreate Id="1">
        <Value>%3%</Value>
    </ActionCreate>
</Actions>
</pre></td></tr>
<tr><td>5</td><td>Assets are identified based on the input provided in <b>ActionCreate</b> value. The asset of the above variable is <b>AI4</b> which is determined by fetching the value of <b>%3%</b> from the Token.</td></tr>
</table>

The various scenarios for **Action Create**, **Local Tokens** 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 <typename> of **CriterionLike** of CreationRule **NOTE:** 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 Tokens**: 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	**Criterion-Found**	%Tagname%_%#2%	FI1001_ONE
	Criterion-Like	FI1001_ONE_AINPUT1	AINPUT1	**Variable Attribute**	%#2%	
	ActionSet	FI1001_ONE_AINPUT1_ST_CFGW	AINPUT1_ST_DDT	**Action-Set**	%DateSource%%Tagname%_AINPUT1_ST.PV%#3%	FI1001_ONE
	ActionRetrieve	FI1001_ONE_AINPUT1	AINPUT1_ST_DDT	**Variable Attribute**	%#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 Attribute**	%#2%_%1%	FI1001_ONE

Object Wizard Supported Asset

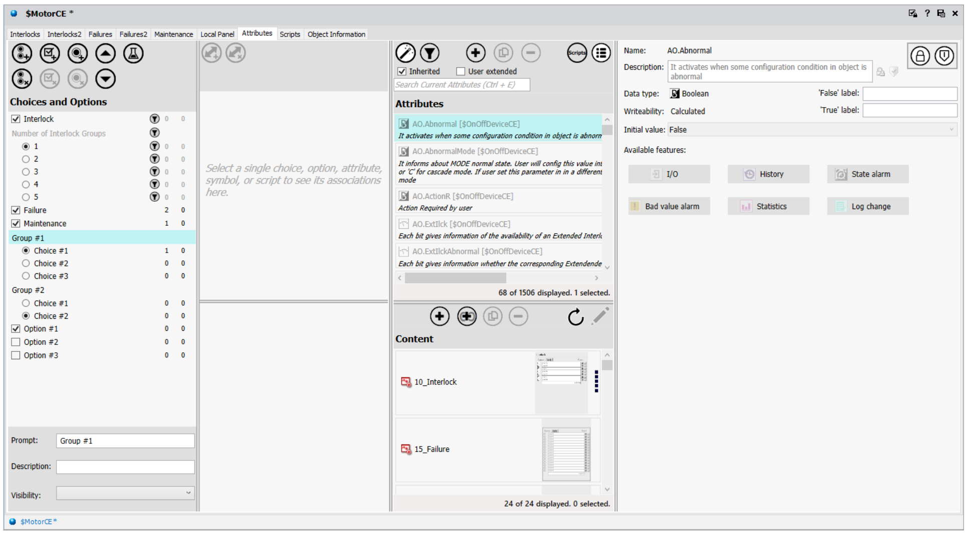
Follow the steps in the table below to view the **Object Wizard** supported asset

Step	Action
1	In the **Configuration** tab, 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 58) 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**

1. Open the template in the **Object Editor**.
2. The template by default opens in **Interlocks Tab**, select **Attributes Tab**.



☑ Interlock	0	0
Number of Interlock Groups		
● 1	0	0
○ 2	0	0
○ 3	0	0
○ 4	0	0
○ 5	0	0
☑ Failure	2	0
☑ Maintenance	1	0
Group #1		
● Choice #1	1	0
○ Choice #2	0	0
○ Choice #3	0	0
Group #2		
○ Choice #1	0	0
● Choice #2	0	0
☑ Option #1	0	0
□ Option #2	0	0
□ Option #3	0	0

Prompt: Group #1

Description:

Visibility:

Select a single choice, option, attribute, symbol, or script to see its associations here.

☑ Inherited □ User extended

AO.Abnormal [$OnOffDeviceCE]
It activates when some configuration condition in object is abnorm

AO.AbnormalMode [$OnOffDeviceCE]
It informs about MODE normal state. User will config this value int or 'C' for cascade mode. If user set this parameter in in a different mode

AO.ActionR [$OnOffDeviceCE]
Action Required by user

AO.ExtIlck [$OnOffDeviceCE]
Each bit gives information of the availability of an Extended Interl

AO.ExtIlckAbnormal [$OnOffDeviceCE]
Each bit gives information whether the corresponding Extendende

68 of 1506 displayed. 1 selected.

10_Interlock

15_Failure

24 of 24 displayed. 0 selected.

Name: AO.Abnormal

Description: It activates when some configuration condition in object is abnormal

Data type: Boolean

'False' label:

Writeability: Calculated

'True' label:

Initial value: False

Available features:

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 only 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 selected and FALSE if unselected) and the various options 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 you make the selection, any other pattern that uses that specific **Template ID** is deselected.

When you select a column heading, the table is sorted 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 **Select** column for selected rows.
- Press **S** to check the box in the **Select** column for selected rows.
- Press **U** to uncheck the box in the **Select** column for selected rows.

Pattern Creation Prerequisites

Adhere to these prerequisites before you create a new pattern:

Instance Creation

Create an instance of an 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`

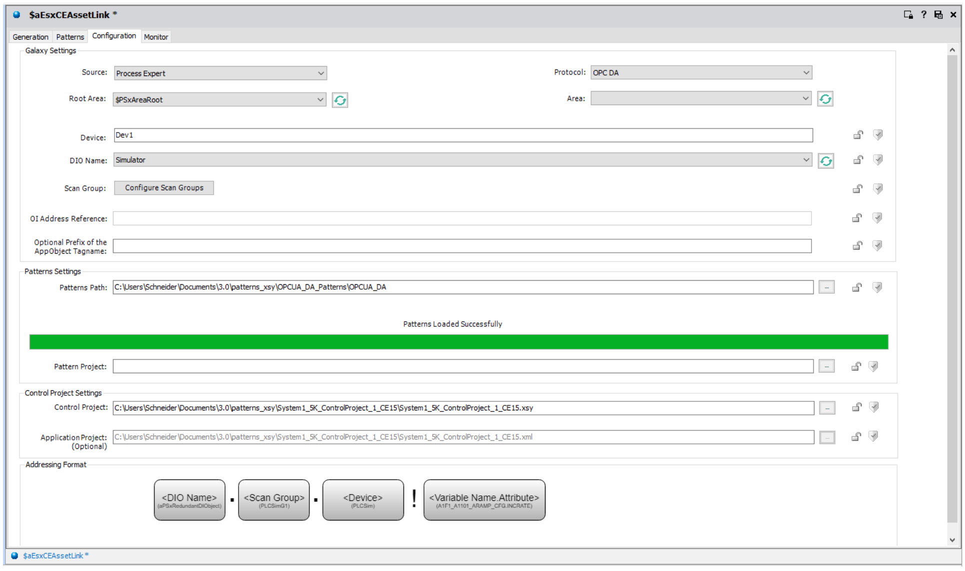
  NOTE: By default, these files are located in this installation folder: **Asset Link > GPL Patterns**

Application Project

The **Application Project** allows to generate the SCADA application related data based on supervision values present in Application Project file. You have the capability to configure the attributes of an asset to fetch the data from the application project file (*.xml*) like that of the control project file (*.xsy*).

In the Pattern Editor, page 46 the Regular/ Include rules refer to the application file (*.xml*) for setting the System Platform attributes. In the Actions tab, page 50 of the pattern editor the **Source Type** by default is set to **Control Data** which means that the data is retrieved from the control project file. You can select **Instance Data** if the data has to be retrieved from Application project. This is allowed only when the **Action** is set to **Action Retrieve**.

In case the **Source Type**, is **Process Expert**, the application project *.xml* is generated during the EPEforASP build operation. When you select the *.xsy* in the Asset Link **Configuration** tab, the application project *.xml* is automatically filled in the **Application Project** field as shown in the following figure (it is a read-only field in this case).



Galaxy Settings

Source: Process Expert

Protocol: OPC DA

Root Area: $PSxAreaRoot

Area:

Device: Dev1

DIO Name: Simulator

Scan Group:

OI Address Reference:

Optional Prefix of the AppObject Tagname:

Patterns Settings

Patterns Path: C:\Users\Schneider\Documents\3.0\patterns_xsy\OPCUA_DA_Patterns\OPCUA_DA

Patterns Loaded Successfully

Pattern Project:

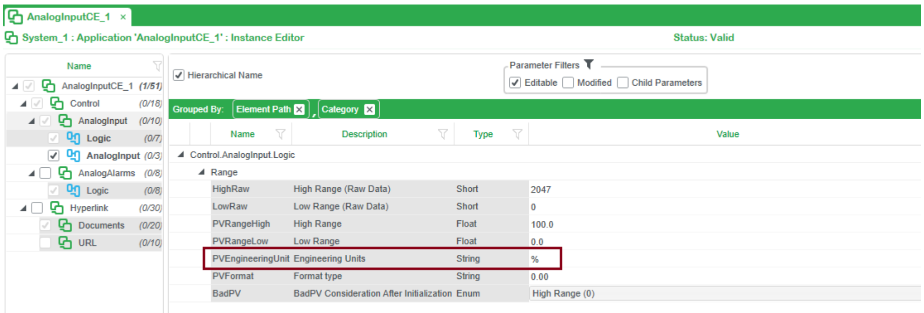
Control Project Settings

Control Project: C:\Users\Schneider\Documents\3.0\patterns_xsy\System1_5K_ControlProject_1_CE15\System1_5K_ControlProject_1_CE15.xsy

Application Project: (Optional) C:\Users\Schneider\Documents\3.0\patterns_xsy\System1_5K_ControlProject_1_CE15\System1_5K_ControlProject_1_CE15.xml

Addressing Format

NOTE: If the source selected is **Process Expert**, to read the parameter *Engineering Units* from the application instance `AnalogInputCE_1` of the template type `$AnalogInputCE`, refer the complete name in the process expert generated application project xml as shown in the following figure.



System_1 : Application 'AnalogInputCE_1' : Instance Editor

Status: Valid

☑ Hierarchical Name

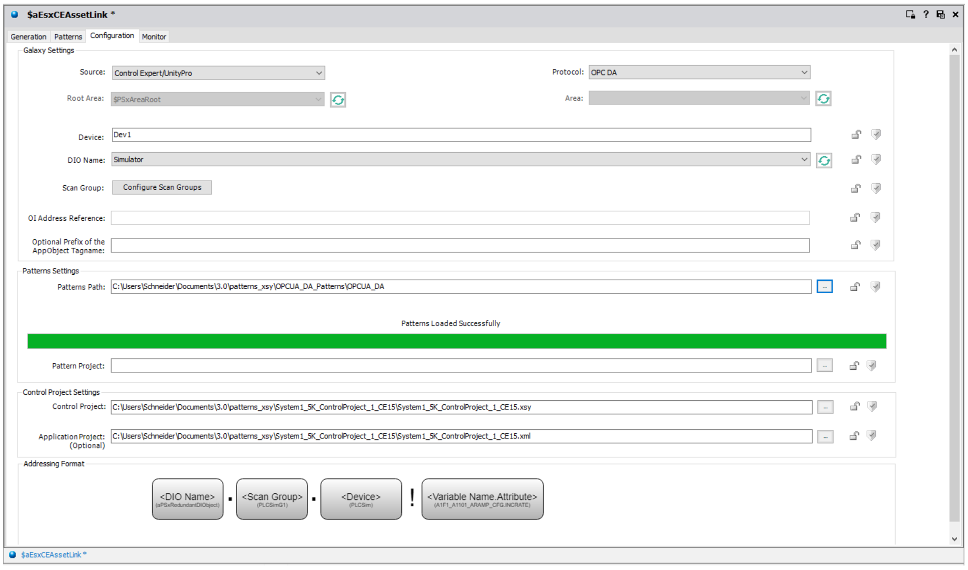
☑ Editable □ Modified □ Child Parameters

Control.AnalogInput.Logic

Range

Name	Description	Type	Value
HighRaw	High Range (Raw Data)	Short	2047
LowRaw	Low Range (Raw Data)	Short	0
PVRangeHigh	High Range	Float	100.0
PVRangeLow	Low Range	Float	0.0
PVEngineeringUnit	Engineering Units	String	%
PVFormat	Format type	String	0.00
BadPV	BadPV Consideration After Initialization	Enum	High Range (0)

For the other sources, you must browse for the location where the application project *.xml* is present. Refer to the sample *.xml* present in the following path `C:\ProgramData\Schneider Electric\Ecostruxure Control Expert - Asset Link\Demo Template and Patterns\Application Project Schema` to create this file.



Galaxy Settings

Source: Control Expert/UnityPro

Protocol: OPC DA

Root Area: $PSxAreaRoot

Area:

Device: Dev1

DIO Name: Simulator

Scan Group:

OI Address Reference:

Optional Prefix of the AppObject Tagname:

Patterns Settings

Patterns Path: C:\Users\Schneider\Documents\3.0\patterns_xsy\OPCUA_DA_Patterns\OPCUA_DA

Patterns Loaded Successfully

Pattern Project:

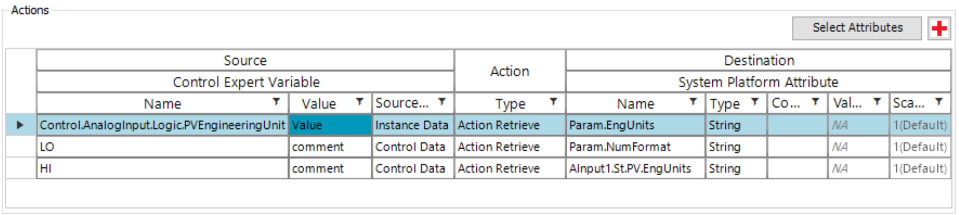
Control Project Settings

Control Project: C:\Users\Schneider\Documents\3.0\patterns_xsy\System1_5K_ControlProject_1_CE15\System1_5K_ControlProject_1_CE15.xsy

Application Project: (Optional) C:\Users\Schneider\Documents\3.0\patterns_xsy\System1_5K_ControlProject_1_CE15\System1_5K_ControlProject_1_CE15.xml

Addressing Format

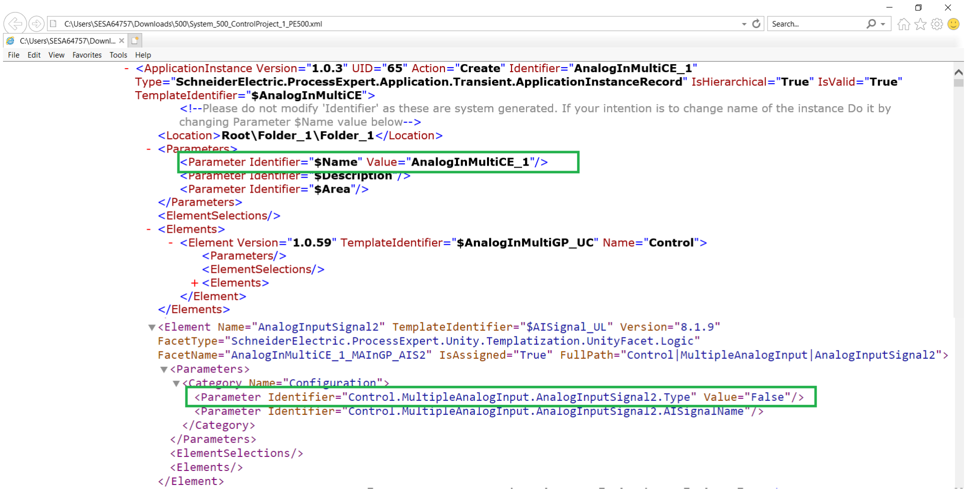
This Application Project *.xml* can be used to configure the **Source Variable** details, for example the **Name** and **Value** in the Actions tab of the pattern editor.



Actions

	Source			Action	Destination				
	Control Expert Variable				System Platform Attribute				
	Name	Value	Source...	Type	Name	Type	Co...	Val...	Sca...
▶	Control.AnalogInput.Logic.PVEngineeringUnit	Value	Instance Data	Action Retrieve	Param.EngUnits	String		NA	1(Default)
	LO	comment	Control Data	Action Retrieve	Param.NumFormat	String		NA	1(Default)
	HI	comment	Control Data	Action Retrieve	AInput1.St.PV.EngUnits	String		NA	1(Default)

The details can be found under the **Parameter Identifier** present in the Application Project *.xml*.



C:\Users\SESA64757\Downloads\500\System_500_ControlProject_1_PE500.xml

Search...
C:\Users\SESA64757\Downl...
File Edit View Favorites Tools Help

```
- <ApplicationInstance Version="1.0.3" UID="65" Action="Create" Identifier="AnalogInMultiCE_1"
  Type="SchneiderElectric.ProcessExpert.Application.Transient.ApplicationInstanceRecord" IsHierarchical="True" IsValid="True"
  TemplateIdentifier="$AnalogInMultiCE">
      <!--Please do not modify 'Identifier' as these are system generated. If your intention is to change name of the instance Do it by
      changing Parameter $Name value below-->
    <Location>Root\Folder_1\Folder_1</Location>
  - <Parameters>
      <Parameter Identifier="$Name" Value="AnalogInMultiCE_1"/>
      <Parameter Identifier="$Description"/>
      <Parameter Identifier="$Area"/>
    </Parameters>
    <ElementSelections/>
  - <Elements>
    - <Element Version="1.0.59" TemplateIdentifier="$AnalogInMultiGP_UC" Name="Control">
          <Parameters/>
          <ElementSelections/>
        + <Elements>
      </Element>
    </Elements>

 ▼<Element Name="AnalogInputSignal2" TemplateIdentifier="$AISignal_UL" Version="8.1.9"
   FacetType="SchneiderElectric.ProcessExpert.Unity.Templatization.UnityFacet.Logic"
   FacetName="AnalogInMultiCE_1_MAInGP_AIS2" IsAssigned="True" FullPath="Control|MultipleAnalogInput|AnalogInputSignal2">
   ▼<Parameters>
     ▼<Category Name="Configuration">
        <Parameter Identifier="Control.MultipleAnalogInput.AnalogInputSignal2.Type" Value="False"/>
        <Parameter Identifier="Control.MultipleAnalogInput.AnalogInputSignal2.AISignalName"/>
      </Category>
    </Parameters>
    <ElementSelections/>
    <Elements/>
  </Element>
```

During **Browse** operation, for the assets whose tag name matches with the Application Project *.xml* **ApplicationInstance** element identifier, the attributes are fetched from the instance data depending on the **Parameter** identifier configured in the **Name** and **Value** fields in the Pattern and then the attribute values are generated.

Generation Tab

Introduction

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

> **NOTE:** By default, the **Generation** tab is available only when valid configuration data is configured and saved on the **Configuration** tab, page 34.

Overview

Generate and update an ASP objects:

Stage	Description
1	Browse the configured control project variables.
2	Select the assets for processing.
3	Create and update an ASP AppObjects. 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 an ASP AppObjects to create, update, or re-create.

This table shows the multiple combinations for which Asset Link 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 detects this object to create.
	Pattern is available for matching all 5 variables	Not Available	Tool does not find this object and hence it is not listed in the Generation Tab. This object (if it is HMI variable) is ignored, it is available in the log file at the following location: *C:\ProgramData\Schneider Electric\Ecostruxure Control Expert - Asset Link*
	No match with the pattern available for all 5 variables	Available	Tool does not find this object and hence it is not be listed in the Generation Tab. This object (if it is HMI variable) is ignored, it is available in the log file at the following location: *C:\ProgramData\Schneider Electric\Ecostruxure Control Expert - Asset Link\Logs\BrowseControlProject_Log_datetimestamp.txt*
	Pattern matches with 3 variables out of 5 variables	Available	Tool does not find this object and hence it is not listed in the Generation Tab. This object (if it is HMI variable) is ignored, it is available in the log the file at the following location: *C:\ProgramData\Schneider Electric\Ecostruxure Control Expert - Asset Link\Logs\BrowseControlProject_Log_datetimestamp.txt*
	Pattern has criteria for 2 variables out of 5 and these 2 variables match	Available	Tool detects this object to create.
	Pattern has criteria for only 2 variables, but variable type name does not match.	Available	Tool does not find this object and hence it is not listed in the Generation Tab. This object (if it is HMI variable) is ignored, it is available in the log file at the following location: *C:\ProgramData\Schneider Electric\Ecostruxure Control Expert - Asset Link\Logs\BrowseControlProject_Log_datetimestamp.txt*
	Pattern does not exist	Available	Tool does not find this object and hence it is not listed in the Generation Tab. This object (if it is HMI variable) is ignored, it is available in the log file at the following location: *C:\ProgramData\Schneider Electric\Ecostruxure Control Expert - Asset Link\Logs\BrowseControlProject_Log_datetimestamp.txt*

Check Source Changes

The **Check for Source Changes**, a radio button is by default selected when you open Asset Link for the first time, it identifies if there is any change from the source for the already generated assets from Asset Link in the same destination. This option helps in detecting the incremental source changes.

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 sorted 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:

  - Ensure that the Remote Terminal Unit file is updated while updating the control variable file.
  - Only the changes from the Control expert and Machine expert sources are detected.

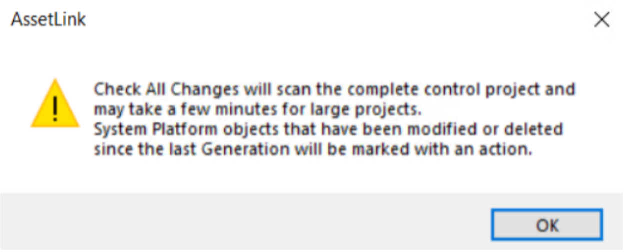
Check AppObjects Changes

You can identify if any of the attribute values are modified in the System Platform for the assets that are already generated. When the **Check AppObjects Changes** radio button option is selected, it also lists the assets for which source side changes are done in the **Browse** grid. This option can be selected only if the generation has been performed at least once.

Check All Changes

Enabling this option will identify the modifications related to the plant model area in the target for the assets that are already generated and also it allows the recovery of any manually deleted Asset Link generated object in the System Platform IDE.

When this option is selected, it displays the following message:



Check All Changes will scan the complete control project and may take a few minutes for large projects.
System Platform objects that have been modified or deleted since the last Generation will be marked with an action.

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

Considerations:

- If this option is not selected, then the **Browse Control Project** only displays the ASP AppObject which is in **Create**, **Re-Create**, or **Update** status.

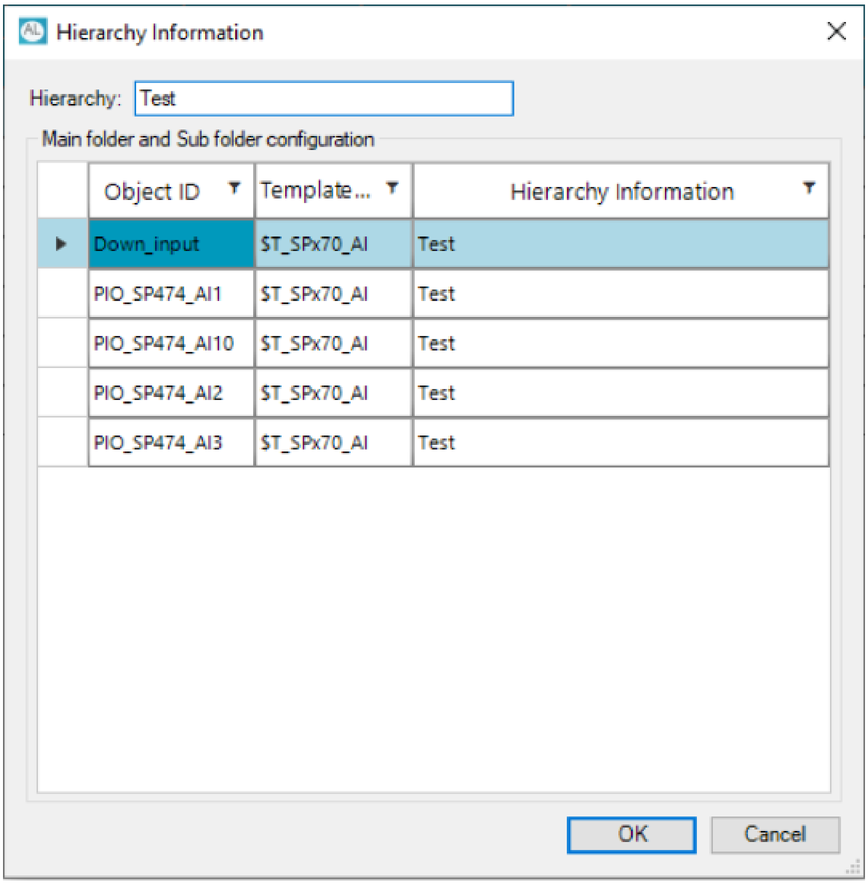
- If this option is not selected and configuration parameter(s) in configuration tab is modified, then the **Browse Control project** will display the ASP AppObjects with result column showing difference in settings. Only the immediate browse operation after configuration parameter(s) change displays this change, two consecutive browse operation do not display the ASP AppObject with difference in settings. In this scenario, you should select **Check for All Changes** and then click **Browse Control Project**. Then **Select All Candidates** and **Generate**.
- If this option is selected, then the **Browse Control Project** only displays all the ASP AppObjects which are in **Create**, **Re-Create**, **Update**, and **No Action** status.

  NOTE:

  - If there is any configuration change, then all the ASP AppObjects are displayed.
  - This option can be selected only if the generation is performed at least once.
  - Modifications at both the source and the destination are detected.
  - Asset hierarchy changes done in target System Platform, are identified only when this option is selected.

Configure Hierarchy

In the **Generation Tab**, click on Hierarchy to open a dialog box in which the location can be set for the selected objects.



Hierarchy: Test

Main folder and Sub folder configuration

	Object ID	Template...	Hierarchy Information
▶	Down_input	$T_SPx70_AI	Test
	PIO_SP474_AI1	$T_SPx70_AI	Test
	PIO_SP474_AI10	$T_SPx70_AI	Test
	PIO_SP474_AI2	$T_SPx70_AI	Test
	PIO_SP474_AI3	$T_SPx70_AI	Test

Hierarchy: Editable field allowing you to enter the Asset location for all the selected rows.

Object ID, Template ID, Hierarchy Information: Read-only fields displaying the rows which are selected in the generation table.

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

Data Grid Services

When the table is populated with the 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 **Select** column for all selected rows.
- Press **S** to select the box in the **Select** column for all selected rows.
- Press **U** to deselect the box in the **Select** column for all selected rows.

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

NOTE: This automatic selection of an area happens 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 an 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, an ASP object in Galaxy is derived from an 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 an 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. **No action** is displayed usually in grey color. When displayed in red color, it indicates that the template or pattern needs correction as some of the AVEVA System Platform template attributes mentioned in Pattern Actions are not found in the template or are not writable, the details are found in the CSV file path displayed below the progress bar in the **Generation tab**. • **To be Resolved:** If any conflict has been detected in the asset name (comes from control project file), **To be Resolved** is displayed. To resolve the conflict, click the **Resolve** button, page 74. • **Rename**: When the Process Expert folder is renamed then the **Action** column for that folder is updated as “Update”, see, page 75 **NOTE:** The above options are enabled only when the browsing of a control project discovers at least one instance to generate.

Column	Description
Result	This column reports the result of the 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 suggest 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 you can use the **Hierarchy** option to provide the same information for multiple objects. **NOTE:** 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 the objects that require processing (**Create**, **Update**, **Re-create**), objects with **No Action** will not be selected. The default actions are taken into consideration, which 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 74.

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 No Action	Click this button to add to the list of selected objects for which **No Action** is proposed.

NOTE: You can select several options.

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 is enabled only if at least one selected candidate available in the data grid. • **Generate Objects** button is 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 is enabled only after you trigger the **Generate Objects**. • **Stop Generation** button is enabled only if more than 10 selected candidates start to generate. • **Stop Generation** button is in a disabled state while stop generation is already in progress.

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

Action	Result
Select Create	Displays **Created** when the object was successfully created. If unsuccessful, it displays the reason for not being created.
Select Re-Create	Deletes the existing object and recreates the object. Displays **Created** when the object was successfully created. If unsuccessful, it displays the reason for not being created.
Select Update	Displays **Updated** when the object was successfully updated. If unsuccessful, it displays the reason for not being updated.
Select No Action	No changes were made to the object. Displays **Updated** to inform that it is up to date.
Stop Generation	The object was not created/updated.

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

The **Message Details** dialog box confirms that the System Platform IDE 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 alert messages and detected errors.
- You can manage these events from the Log Viewer in the Operations Control Management Console.
- If the object generation gets stopped and the System Platform IDE has restarted, there could be some objects left checked out and this is not identified by the Asset Link tool. Check-in any checked out object(s) and generate again.
- If the System Platform IDE has restarted during the process of object generation, there might be some objects left checked-out and this is not identified by the Asset Link tool. Check-in any checked-out object(s) and generate again.
- If System Platform IDE closes during the generation process, restart the System Platform IDE 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 minutes. Default time: 1 minute

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

When a modification is detected in the:

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

Auto Browse and Generation

The **Auto Browse and Generation** describes the functionality for support of **Auto Browse** and **Auto Generate** based on the control project source file changes and provides flexibility in Asset Link for EcoStruxure Control Expert.

The Asset Link for EcoStruxure Control Expert offers you the choice of enabling or disabling the **Auto Browse** and **Auto Generate** actions. By default **Auto Browse** and **Auto Generate** are disabled.

NOTE:

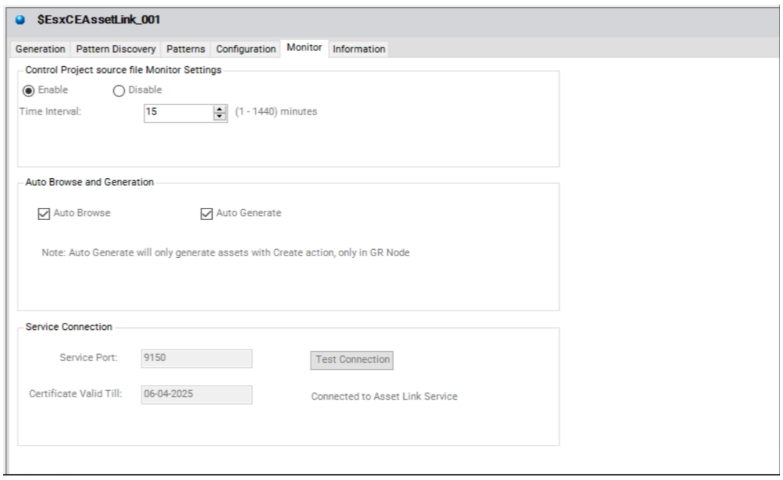
- The **Monitor Tab** must be enabled with mandatory **Time Interval** settings to enable **Auto Browse** and **Auto Generate**.
- **Auto Generate** is applied to assets with the **Create action**, only in GR node.
- If **Auto Generate** is enabled, **Auto Browse** is enabled automatically.

The following tables describe use cases with different conditions:

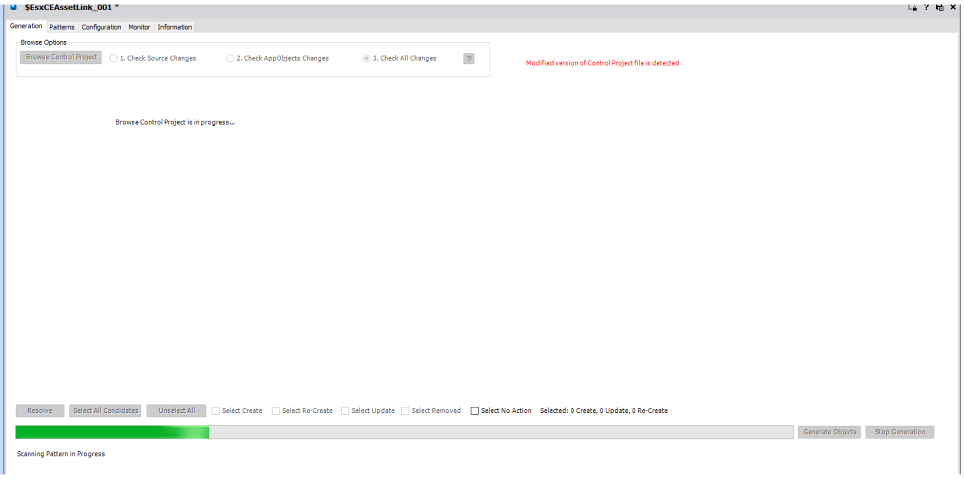
Asset Link for EcoStruxure Control Expert triggers the **Auto Browse** and **Auto Generate** actions, if the source file is detected as modified. The control project source file changes are monitored at time interval set in the **Monitor Tab**.

Step	Auto Browse	Auto Generate	Result
1	Enabled	Disabled	**Auto Browse** happens in Asset Link for EcoStruxure Control Expert.
2	Enabled	Enabled	**Auto Browse** and **Auto Generate** is performed in Asset Link for EcoStruxure Control Expert.
3	Disabled	Disabled	No **Auto Browse** or **Auto Generate** is performed. You must perform manual **Browse** and **Generate**.

The following table describes the **Auto Browse** and **Auto Generate** procedures:

Step	Action
1	Open Asset Link for EcoStruxure Control Expert.
2	Navigate to **Monitor Tab**.
3	Enable the **Auto Browse** and **Auto Generate** actions as needed.  $EsxCEAssetLink_001 Generation Pattern Discovery Patterns Configuration Monitor Information Test Connection Control Project source file Monitor Settings ● Enable ○ Disable Time Interval: 15 (1 - 1440) minutes Auto Browse and Generation ☑ Auto Browse ☑ Auto Generate Note: Auto Generate will only generate assets with Create action, only in GR Node Service Connection Service Port: 9150 Certificate Valid Till: 06-04-2025 Connected to Asset Link Service **Result**: When the configured control project source file is modified, and if this option **Auto Browse** is enabled, it automatically triggers the browse operation. Also if **Auto Generate** is enabled, after browse, it automatically performs the **Generation**. **NOTE: Auto Browse** and **Auto Generate** actions are triggered only in the Asset Link opened in GR node. If working with Asset Link template opened on more than one computer, the **Auto Browse** and **Auto Generate** actions are skipped.

When the configured control project source file is modified, if the option **Auto Browse** is enabled , it automatically triggers the browse operation. The **Check Source Changes** option is enabled by default. However, you can choose to modify the browse option to **Check All Changes** or **Check AppObjects Changes** and the **Auto Browse** is still functional.



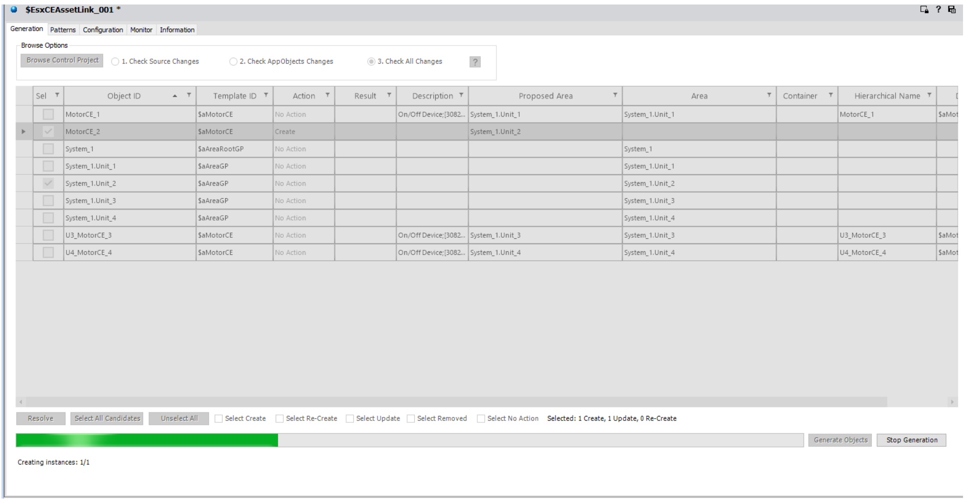
Modified version of Control Project file is detected

Browse Control Project is in progress...

Selected: 0 Create, 0 Update, 0 Re-Create

Scanning Pattern in Progress

Auto Generation is triggered after the browse action if **Auto Generate** is enabled. **Auto Generate** happens only for assets of **Create action**. Assets with **Update**, **Re-create**, **No Action** or **Resolved** are not considered for automatic generation.



Sel	Object ID	Template ID	Action	Result	Description	Proposed Area	Area	Container	Hierarchical Name	
□	MotorCE_1	$aMotorCE	No Action		On/Off Device;[3082...	System_1.Unit_1	System_1.Unit_1		MotorCE_1	$aMot
☑	MotorCE_2	$aMotorCE	Create			System_1.Unit_2				
□	System_1	$aAreaRootGP	No Action				System_1			
□	System_1.Unit_1	$aAreaGP	No Action				System_1.Unit_1			
☑	System_1.Unit_2	$aAreaGP	No Action				System_1.Unit_2			
□	System_1.Unit_3	$aAreaGP	No Action				System_1.Unit_3			
□	System_1.Unit_4	$aAreaGP	No Action				System_1.Unit_4			
□	U3_MotorCE_3	$aMotorCE	No Action		On/Off Device;[3082...	System_1.Unit_3	System_1.Unit_3		U3_MotorCE_3	$aMot
□	U4_MotorCE_4	$aMotorCE	No Action		On/Off Device;[3082...	System_1.Unit_4	System_1.Unit_4		U4_MotorCE_4	$aMot

Selected: 1 Create, 1 Update, 0 Re-Create

Creating instances: 1/1

NOTE: During generation if the System Platform got abruptly closed, then after reopened System Platform and on opening of Asset Link, it shows a message in the **Generation Tab** as last generation was not successful, perform another browse and generate to generate the remaining assets.

Information Tab

The **Information** tab displays the following details of Ecostruxure Control Expert Asset Link.

Field	Description
Asset Link Version	Displays the version of Asset Link.
Log Files	Displays the path where the log files are saved.
User Guide	Displays the path where the user guide is present.
License Details	Displays information regarding the product license.

Working with AVEVA System Platform Plant Model

Creating Plant Model in ASP

Overview

This chapter describes the function of the Asset Link tool in the creation of a Plant Model in ASP using the data defined in the Plant Model for EcoStruxure™ Process Expert AVEVA System Platform.

You can perform actions such as **Create**, page 73, **Move/Update**, page 73, and **Resolve Conflicts**, page 74 of objects based on the hierarchy provided in the . xml file.

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 AVEVA System Platform.

For details about the work flow of System Platform IDE, refer to the work flow, page 29.

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 34.
	Device Name	
	DIO Name	
	Scan Group	
	OI Address Reference	
	Optional Prefix of the AppObject Tagname	
Patterns Settings	**Patterns Path**	Enter the path to the .xml pattern files that Asset Link 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 Asset Link 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 have to be in this same folder.

Section	Parameter	Description
Control Project Settings	**Control Project**	This field contains the full name of the file that Asset Link 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 40.
	Application Project	When you configure the control project for process expert, by default, the application XML is present in the same location as the control project, hence it is automatically filled. For more details refer to Application Project, page 59.

Patterns Tab

For details about this tab, refer to the patterns tab, page 44 chapter.

Generation Tab

For more details about this tab, refer to the generation tab, page 61 chapter.

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

Column	Description
Select	For details, refer to configuration, page 65 table columns.
Object ID	This column displays the asset or the area name which is configured in the input file (.xsy). • If you have resolved asset name generated in ASP AppObject, then the **Object ID** displays “Original Name(Resolved Name)”. • If you have resolved area name generated in ASP AppObject, then the **Object ID** displays “Original Name(–>)” to refer to the resolve area name in **Area** column.
Template ID	This column displays the respective derived type of template for asset and area. EcoStruxure™ Process Expert for AVEVA System Platform system name is displayed as root area which is of **Template** type $aPSxAreaRootGP. For child hierarchy, it is of type area ($aPSxAreaGP).
Action	For details, refer to configuration, page 65 table columns.
Result	
Description	
Proposed Area	This column reports the hierarchical area name that corresponds to the EcoStruxure™ Process Expert .xml file. **NOTE:** • Areas are modified based on hierarchy given in the .xml file. • Areas which are modified with updated hierarchy are displayed in the **Model** tab of the ASP. • If the area of the object is already deployed, it un-deploys and regenerates the area.
Area	For details, refer to configuration, page 65 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 you using derived template type $PSxAreaGP, then it is not managed by Asset Link. 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 for AVEVA System Platform Plant Model provided in the .xml file:

Step	Action
1	In the **Generation** tab, select the objects with status displayed as **Create** in the **Action** column.
2	Click **Generate Objects**, page 67. **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 the **Area** column displays the ASP hierarchy. **NOTE:** • For the object of type “Area”, the **Proposed Area** column is 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 is 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. Conflicts are detected when there are 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 in **To be Resolved** status.
- If renamed object has a conflict, it is 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 is 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 which has a detected conflict to resolve.
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 which have a detected conflict to resolve.
Unselect All	Select this check box to unselect the objects.

Rename

In the System Platform you can handle the EcoStruxure™ Process Expert 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 displays "**Diff: Renamed**". **NOTE:** • If the unique identifier is the same for two different objects from the **Process Expert** source, then, the object does not detect as "**Diff: Renamed**". In this case, "**Create**" action is performed. • If the respective object is in deployed state, the **Result** column displays “**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 EcoStruxure™ Process Expert is named as “Root” and converted into system name when you execute using Asset Link. This is 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 displays **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 of 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 includes 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 created as an 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 includes 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 creation, indicates if the rule was manually modified. etc. • **List of Criteria:** These criteria determine the data that constitutes ASP AppObjects. The syntax of the criteria allows you to define naming conventions that are expected from control project variables to detect the presence of the asset and collect data that can be used later from the list of actions that are required for the ASP AppObject User Defined Attribute(s) (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 contribute to the 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>
```

Here is a description of 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

Knowledge of 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>
```

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:

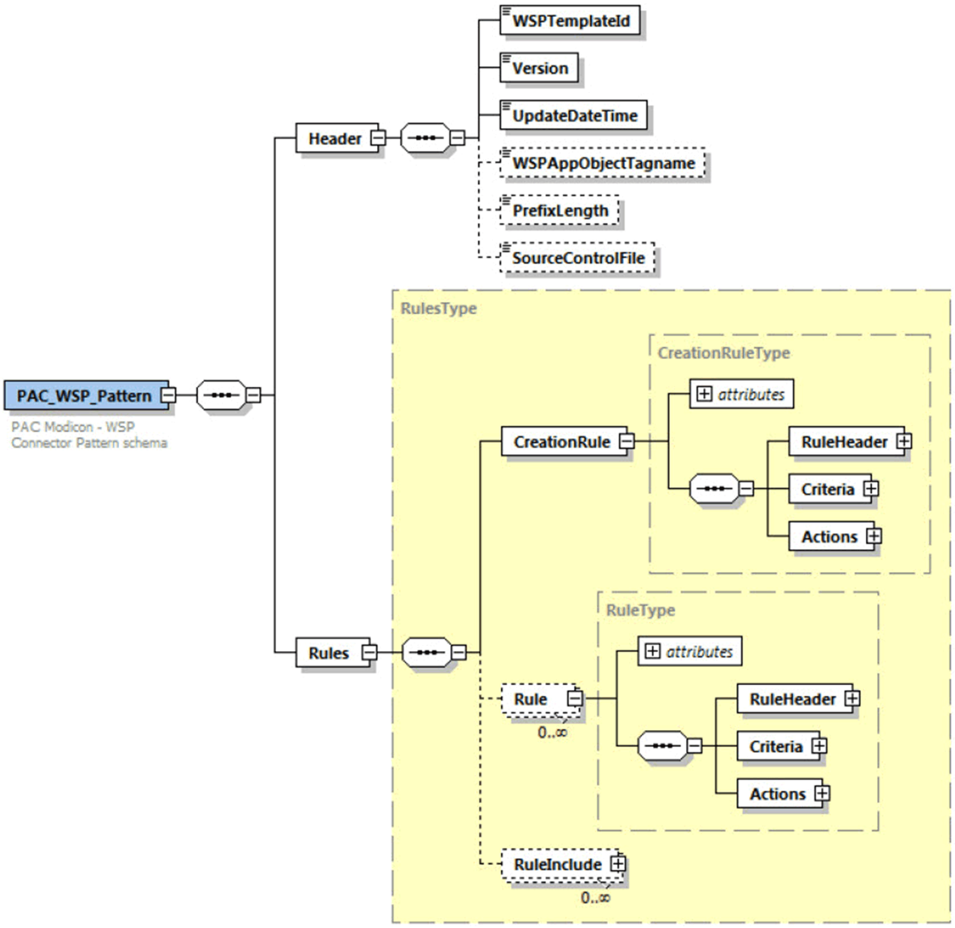
```
<variables name="RUB1_CONDSUM1" typeName="CONDSUM1">
<comment>RUB1 COMMENT</comment>
<instanceElementDesc name="COND01">
<comment>Condition 1 (higher priority) Rub</comment> </instanceElementDesc>
<instanceElementDesc name="REQREARM01">
<comment>Condition 1 requires rearm Rub</comment>
</instanceElementDesc>
<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 a visible element:



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

- Header, page 79
- Rules, page 80

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 79.

Path: `PAC_WSP_Pattern/Header`

Header Element

The `Header` element contains these sub-elements:

Element	Description
ASPTemplateId	This identifier is the exact name of the ASP Template, used for generating new instances when this pattern is matched.
Version	This sub-element contains the implemented version of the pattern.
UpdateDateTime	This sub-element reports the date and time of the last modification to the pattern.
ASPAppObjectTagname	If the pattern is discovered 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 pattern is discovered 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 79.

Path: `PAC_WSP_Pattern/Rules`

The sub-elements of the Rules, page 80 flowchart are discussed below:

- `CreationRule`, page 81
- `Rule`, page 87
- `RuleInclude`, page 95

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 79 through this path:

`PAC_WSP_Pattern/Rules`

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

Element	Description
CreationRule, page 81	Asset Link executes the CreationRule element.
Rule, page 87	0 ... *n*
RuleInclude, page 95	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 79 through this path:

PAC_WSP_Pattern/Rules/CreationRule

CreationRule Element

CreationRule element descriptions:

Element(s)		Description
attributes	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 82
- Criteria, page 82
- CriterionFound, page 82
- CriterionLike, page 83
- Actions, page 86

RuleHeader Element

Follow this path to access this element:

`PAC_WSP_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:* You created the rule manually.
- `Updated`: This Boolean value has these values:
  - *TRUE:* The value is `TRUE` for manually created rules and for automatically generated rules that you can edit.
  - *FALSE:* You cannot edit the rule with Asset Link.
- `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_WSP_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:

- `CriterionFound`
- `CriterionLike`

  The combination of `CriterionFound` and `CriterionLike` is one set of conditions in the Criteria Element.

CriterionFound Element

Follow this path to access this element:

`PAC_WSP_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 searches for 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_WSP_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, 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 an `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 searches 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>
```

`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, there can be 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_WSP_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 79 through this path:

`PAC_WSP_Pattern`/`Rules`/`Rule`

This set of 0 to *n* rule elements follows the `CreationRule`, page 81.

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 88
- Criteria, page 88
- Actions, page 89
- ActionSet, page 89
- WSPAppObjectAttribute, page 90
- Value, page 90
- ActionRetrieve, page 91
- WSPAppObjectAttribute, page 91
- Subelement, page 92
- GetElementValue, page 93
- GetVariableAttribute, page 93

RuleHeader

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

PAC_WSP_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 Asset Link analyzes rule or not.

Criteria

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

PAC_WSP_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 searches 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_WSP_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 **User Defined Attribute** of the newly created instance:

Expand the Actions element to access these elements:

- ActionSet, page 89: This element writes a specific constant value on one UDA.
- ActionRetrieve, page 91: 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_WSP_Pattern/Rules/Rule/Actions/ActionSet

Expand the ActionSet element to access these elements:

- attributes (Id)
- WSPAppObjectAttribute, page 91:
  - attributes (type)
  - attributes (ContainedName)
- Value, page 90

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">

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

<Value>True</Value>

</ActionSet>


<ActionSet Id="1">

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

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

</ActionSet>
```

Expand the `ActionSet` element to access these elements:

- `WSPAppObjectAttribute, page 90`
- `Value, page 90`

WSPAppObjectAttribute

Follow this path to access the sub-elements for the `WSPAppObjectAttribute` element in the `WSPAppObjectAttributeType` area:

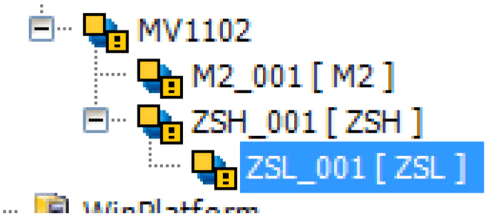
`PAC_WSP_Pattern/Rules/Rule/Actions/ActionSet/WSPAppObjectAttribute`

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

Expand the `WSPAppObjectAttribute` element to access these elements:

- `type`: This attribute defines the UDA 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 `WSPAppObjectAttribute` element in the `WSPAppObjectAttributeType` area:

`PAC_WSP_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 match `%%` 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 `WSPAppObjectAttribute` element in the `ActionRetrieveType` area:

`PAC_WSP_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 match 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:

- `WSPAppObjectAttribute, page 91`: 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 93`: Choose this element to obtain a value that corresponds to an element.
- `GetVariableAttribute, page 93`: Choose this element to obtain a value that corresponds to an attribute.

This is an example of an `ActionRetrieve` element:

```
<ActionSet Id="0">

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

<Value>True</Value>

</ActionSet>



<ActionSet Id="1">

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

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

</ActionSet>
```

WSPAppObjectAttribute

Follow this path to access the sub-elements for the `WSPAppObjectAttribute` element in the `WSPAppObjectAttributeType` area:

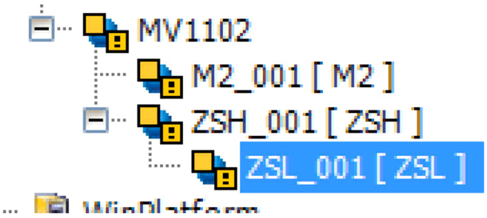
PAC_WSP_Pattern/Rules/Rule/Actions/ActionRetrieve/WSPAppObjectAttribute

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

Expand the WSPAppObjectAttribute element to access these elements:

- type: This attribute defines the UDA 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.
- ScanGroup, page 93: This attribute defines the scan group value between 1 to 10.

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 Subelement element in the SubelementType area:

PAC_WSP_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. It 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 93 or GetVariableAttribute, page 93 elements.

NOTE: Refer to the section CriterionLike for complete Subelement description, page 83.

GetElementValue

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

`PAC_WSP_Pattern`/`Rules`/`Rule`/`Actions`/`ActionRetrieve`/`Subelement`/`GetElementValue`

When `GetElementValue` is defined in the `ActionRetrieve` 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 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_WSP_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` starts in the element that is selected by the `Subelement` that is defined above:

```
<ActionRetrieve Id="1" >

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

<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, so you have to get the value from the `Variables.comment` element (`instanceElementDesc`).

```
<ActionRetrieve Id="1" >
<WSPAppObjectAttribute type="String">Ilck.Legend1</WSPAppObjectAttribute>
<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 have 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" >

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

<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 have 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 have 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_WSP_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 pattern file.

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

GPL Patterns for Asset Link

General Purpose Library

Introduction

This section describes controller code based on General Purpose Library for EcoStruxure™ Control Expert and AVEVA System Platform.

Use the information in this section to use the set of pre-built Asset Link patterns in the GPL. Such patterns are built to establish a connection between the 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
- AVEVA System Platform (ASP)
- EcoStruxure™ Control Expert Asset Link
- General Purpose Library for Modicon and ASP

Project Engineering

Control Projects

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

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 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 Asset Link objects in the ASP Galaxy.

`Default patterns files location is C:\ProgramData\Schneider Electric\Ecostruxure Control Expert - Asset Link\GPL Patterns.`

Asset Link 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 95 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 errors, 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 76.

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 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	Alarm setpoints are enabled. Customize other types of combinations as the actual setpoints to be managed cannot be inferred from 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 Rules file is included in all patterns to prevent the automatic binding of ASP AppObjects references by setting the **Config.Ref.Disable** UDA.
iPSxCondsum	Error Conditions Summary	CONDSUM	<obj>_CONDSUM	CONDSUM	The detected error condition descriptions are retrieved from the DFB CONDSUM pin descriptions **COND##**.
			<obj>_CONDSUM_ST	CONDSUM_ST_DDT	

Include Rules File	Service	DFB Type	Variable Naming Convention	Variable Data Type	Comments
iPSxCondsum1Analog	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	
iPSxCondsumF2	Error Conditions Summary (for 2 direction/speed)	CONDSUM	<obj>_RC_CONDSUM	CONDSUM	The detected error condition descriptions are retrieved from the DFB CONDSUM pin descriptions **COND##**.
			<obj>_RC_CONDSUM_ST	CONDSUM_ST_DDT	
iPSxCondsumFC	Error Conditions Summary (for sequences and equipment modules)	CONDSUM	<obj>_FC_CONDSUM	CONDSUM	The detected error 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

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:

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

Pattern File (& ref. to specific Include Rules)	Asset Type	DFB Type	Variable Naming Convention[1]	Variable Data Type	ASP Template	Comments
aPSxAlarmSummary *iPSxCondsum	Alarms Summary	DINPUT	<obj>_DINPUT	DINPUT	$aPSxAlarmSummary	The description of the Asset is retrieved from the description of the DFB.
		DALARM	**<obj>_DI_ST**	DINPUT_ST_DDT		
			<obj>_DALARM_ST	DALARM_ST_DDT		

Pattern File (& ref. to specific Include Rules)	Asset Type	DFB Type	Variable Naming Convention[1]	Variable Data Type	ASP Template	Comments
aPSxAnalogInput *iPSxAAlarm	Analog Input (legacy)	AINPUT	<obj>_AINPUT	AINPUT	$aPSxAnalogInput	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	AINPUT1	<obj>_AINPUT1	AINPUT1	$aPSxAnalogInput1	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>_AINPUT1_ST	AINPUT1_ST_DDT		
			<obj>_AINPUT1_CFG	AINPUT1_CFG_DDT		
			<obj>_PV_RNG	RANGE_DDT		
aPSxAnalogOutput *iPSxCondsum *iPSxAoutputlp	Analog Output	AOUTPUT	<obj>_AOUTPUT	AOUTPUT	$aPSxAnalogOutput	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	ASELECT1	<obj>_ASELECT1	ASELECT1	$aPSxASelect1	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 *iPSxCondsum1Analog	Control Valve	CVALVE	<obj>_CVALVE	CVALVE	$aPSxControlValve	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	$aPSxDigitalInput	The description of the Asset is retrieved from the description of the DFB.
			<obj>_DINPUT_ST	DINPUT_ST_DDT		
aPSxDigitalOutput *iPSxCondsum1Discr *iPSxDevmnt	Digital Output	DOUTPUT	<obj>_DOUTPUT	DOUTPUT	$aPSxDigitalOutput	The description of the Asset is retrieved from the description of the DFB.
			<obj>_DOUTPUT_ST	DOUTPUT_ST_DDT		

Pattern File (& ref. to specific Include Rules)	Asset Type	DFB Type	Variable Naming Convention[1]	Variable Data Type	ASP Template	Comments
aPSxDiscreteSP	Discrete Setpoint	n/a	**<obj>_DISCRETESP**	BOOL	$aPSxDiscreteSP	The description of the Asset is retrieved from the description of the variable.
aPSxDualOutputValve *iPSxCondsum1Discr *iPSxCondsum *iPSxDevmnt	Dual Output Valve	DVALVE	<obj>_DVALVE	DVALVE	$aPSxDualOutputValve	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	$aPSxDurationSP	The description of the Asset is retrieved from the description of the variable.
aPSxEquipmentModule *iPSxCondsumIC *iPSxCondsumFC	Equipment Module	EMCTL	<obj>_EMCTL	EMCTL	$aPSxEquipmentModule	The description of the Asset is retrieved from the description of the DFB.
			<obj>_EMCTL_ST	EMCTL_ST_DDT		
			<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	$aPSxHandValve	The description of the Asset is retrieved from the description of the DFB.
			<obj>_HVALVE_ST	HVALVE_ST_DDT		
aPSxIBPhase *iPSxCondsumIC *iPSxCondsumFC	InBatch Phase	IBPHASE	<obj>_IBPHASE	IBPHASE	$aPSxIBPhase	The description of the Asset is retrieved from the description of the DFB. A maximum of one of the IBPAR05, IBPAR10 and IBPAR16 variables are expected at a 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	CONDSUM	<obj>_IC	CONDSUM		The description of the Asset is retrieved from the description of the DFB. Once the AppObjects are generated, you have to contain them in their related AppObject Container manually. The detected initial condition descriptions are retrieved from the DFB CONDSUM pin descriptions 'COND##'.
			<obj>_IC_ST	CONDSUM_ST_DDT		
aPSxIlck	Extended Interlock Condition	CONDSUM	<obj>_ILCK	CONDSUM		The description of the Asset is retrieved from the description of the DFB. Once the AppObjects are generated, you have to contain them in their related AppObject Container manually. The interlock condition descriptions are retrieved from the DFB CONDSUM pin descriptions 'COND##'.
			<obj>_ILCK_ST	CONDSUM_ST_DDT		

Pattern File (& ref. to specific Include Rules)	Asset Type	DFB Type	Variable Naming Convention[1]	Variable Data Type	ASP Template	Comments
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	$aPSxIntegerSP	The description of the Asset is retrieved from the description of the variable.
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		

Pattern File (& ref. to specific Include Rules)	Asset Type	DFB Type	Variable Naming Convention[1]	Variable Data Type	ASP Template	Comments
aPSxMotor2 *iPSxCondsum1Discr *iPSxCondsum *iPSxCondsumF2 *iPSxDevmnt *iPSxMotor2Ip	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 error condition descriptions are retrieved from the DFBs CONDSUM pin descriptions 'COND##'. 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>_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		
			<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		
			<obj>_AI_ST	AINPUT1_ST_DDT		
			<obj>_AI_CFG	AINPUT1_CFG_DDT		
			<obj>_PV_RNG	RANGE_DDT		
aPSxMotorizedValveD *iPSxCondsum1Discr *iPSxMValvedIp	Discrete Motorized Valve	MVALVED MVALVELP MOTOR2 CONDSUM DINPUT AINPUT1	<obj>_MVALVED	MVALVED	$aPSxMotorizedValve	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 error 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	Motor Variable Speed	SDDEVCTL	<obj>_SDDEVCTL	SDDEVCTL	$aPSxMotorVS	The description of the Asset is retrieved from the description of the DFB. From the constant Range variables:
			<obj>_SDDEVCTL_ST	SDDEVCTL_ST_DDT		

Pattern File (& ref. to specific Include Rules)	Asset Type	DFB Type	Variable Naming Convention[1]	Variable Data Type	ASP Template	Comments
*iPSxDevmnt			**<obj>_SDDEVCTL_CFG**	SDDEVCTL_CFG_DDT		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>_PV_RNG	RANGE_DDT		
			<obj>_OP_RNG	RANGE_DDT		
aPSxPID *iPSxCondsum1Analog	PID Controller	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		
aPSxPIDMultiplexer	PID Multiplexer	PIDMUX	<obj>_PIDMUX	PIDMUX	$aPSxPIDMultiplexer	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 *iPSxCondsum1Analog	Pulse Width Modulator	PWMCTL	<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 Controller	RATIOCTL	<obj>_RATIOCTL	RATIOCTL	$aPSxRatioCtl	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	$aPSxRealSP	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
aPSxSequentialControl *iPSxCondsumIC *iPSxCondsumFC	Sequential Control	SEQCTL1	<obj>_SEQCTL1	SEQCTL1	$aPSxSequentialControl	The description of the Asset is retrieved from the description of the DFB. A maximum of one of the SEQPAR05, SEQPAR10 and SEQPAR16 variables are expected at a 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 *iPSxCondsum1Analog	Split Range Controller	SPLRGCTL	<obj>_SPLRGCTL	SPLRGCTL	$aPSxSplitRangeCtl	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. The High and Low range is retrieved from the HI and LO field initial values.
			<obj>_SPLRGCTL_ST	SPLRGCTL_ST_DDT		
			<obj>_SPLRGCTL_CFG	SPLRGCTL_CFG_DDT		
			<obj>_SP_RNG	RANGE_DDT		
			<obj>_OP_RNG	RANGE_DDT		
aPSxStep3Ctl *iPSxCondsum1Analog	3 Steps Controller	STEP3CTL	<obj>_STEP3CTL	STEP3CTL	$aPSxStep3Ctl	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	DEVCTL	<obj>_DEVCTL	DEVCTL	$aPSxValve	The description of the Asset is retrieved from the description of the DFB.
			<obj>_DEVCTL_ST	DEVCTL_ST_DDT		

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)	HWCIRCUITBREAKER	<Obj>_HWCIRCUITBREAKER	HWCIRCUITBREAKER	$PSxHWCompact	The description of the Asset is retrieved 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)	MBCOMPACTNSX MBUCOMPACTNSX	<Obj>_MBCOMPACTNSX	MBCOMPACTNSX	$PSxCompact	
			<Obj>_MBUCOMPACTNSX	MBUCOMPACTNSX		
			<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		
aPSxHWCircuitBreaker *iPSxCommon.ixml*	Circuit Breakers (Description for Hardwired Circuit Breaker)	HWCIRCUITBREAKER	<Obj>_HWCIRCUITBREAKER	HWCIRCUITBREAKER	PSxHWCircuitBreaker	
			<Obj>_HWCB_CFG	HWCB_CFG_DDT		
			<Obj>_HWCB_ST	HWCB_ST_DDT		
aPSxHWMasterpact *iPSxCommon.ixml*	Circuit Breakers (Description for Hardwired Masterpact)	HWCIRCUITBREAKER	<Obj>_HWCIRCUITBREAKER	HWCIRCUITBREAKER	PSxHWMasterpact	
			<Obj>_HWCB_CFG	HWCB_CFG_DDT		
			<Obj>_HWCB_ST	HWCB_ST_DDT		
aPSxMasterPACT *iPSxCommon.ixml*	Circuit Breakers (MasterPACT Protection Unit with Chassis)	MBMASTERPACT	<Obj>_MBMASTERPACT	MBMASTERPACT	$PSxMasterPACT	
			<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		
aPSxMasterpactMTZwoC *iPSxCommon.ixml*	Circuit Breakers (MasterpactMTZ Protection Unit without Chassis)	MBUMASTERPACTMTZ	<Obj>_MBUMASTERPACTMTZ	MBUMASTERPACTMTZ	$PSxMasterpactMTZwoC	
			<Obj>_MASTERPACT_CFG	MASTERPACT_CFG_DDT		
			<Obj>_MASTERPACT_ST	MASTERPACT_ST_DDT		
			<Obj>_MASTERPACT_MEA	MASTERPACT_MEA_DDT		
			<Obj>_MASTERPACT_MEAExt	MASTERPACT_MEAExt_DDT		

Pattern File (& ref. to specific Include Rules)	Asset Type	DFB Type	Variable Naming Convention[1]	Variable Data Type	ASP Template	Comments
			<Obj>_MASTERPACT_MEAExt1	MASTERPACT_MEAExt1_DDT		
aPSxMasterPACTNxC *iPSxCommon.ixml*	Circuit Breakers (MasterpactNx Protection Unit with Chassis)	MBUMASTERPACTNxC	<Obj>_MBUMASTERPACTNxC	MBUMASTERPACTNxC	$PSxMasterPACTNxC	
			<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		
aPSxMasterPACTwoC *iPSxCommon.ixml*	Circuit Breakers (Masterpact Protection Unit without Chassis)	MBMASTERPACT	<Obj>_MBMASTERPACT	MBMASTERPACT	$PSxMasterPACTwoC	
			<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		
aPSxMasterpactMTZC *iPSxCommon.ixml*	Circuit Breakers (MasterpactMTZ Protection Unit with Chassis)	MBUMASTERPACTMTCZ	<Obj>_MBUMASTERPACTMTZC	MBUMASTERPACTMTZC	$PSxMasterpactMTZC	
			<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)	MBMASTERPACTC	<Obj>_MBMASTERPACTC	MBMASTERPACTC	$PSxMasterPACTC	
			<Obj>_MASTERPACT_CFG	MASTERPACT_CFG_DDT		
			<Obj>_MASTERPACT_ST	MASTERPACT_ST_DDT		
			<Obj>_MASTERPACT_MEA	MASTERPACT_MEA_DDT		
			<Obj>_MASTERPACT_MEAExt	MASTERPACT_MEAExt_DDT		

Pattern File (& ref. to specific Include Rules)	Asset Type	DFB Type	Variable Naming Convention[1]	Variable Data Type	ASP Template	Comments
			<Obj>_MASTERPACT_MEAExt1	MASTERPACT_MEAExt1_DDT		
aPSxMasterPACTNxwoC *iPSxCommon.ixml*	Circuit Breakers (MasterpactNx Protection Unit without Chassis)	MBUMASTERPACTNx	<Obj>_MBUMASTERPACTNx	MBUMASTERPACTNx	$PSxMasterPACTNxwoC	
			<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 Relays Sepam 20C Modbus Serial; Sepam 20C MB TCP I/O Scanning)	MBSEPAM20CB	<Obj>_MBSEPAM20CB	MBSEPAM20CB	$PSxSepam20CB	
			<Obj>_SEPAM_CFG	SEPAM_CFG_DDT		
			<Obj>_SEPAM_ST	SEPAM_ST_DDT		
			<Obj>_SEPAM_VMEA	SEPAM_VMEA_DDT		
			<Obj>_SEPAM_IO20	SEPAM_IO20_DDT		
aPSxSepam20CSTM *iPSxCommon.ixml*	Digital Protection Relays (Digital Protection Relays Sepam 20C STM Modbus Serial; Sepam 20C STM TCP I/O Scanning)	MBSEPAM20CSTM	<Obj>_MBSEPAM20CSTM	MBSEPAM20CSTM	$PSxSepam20CSTM	
			<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)	MBSEPAM40C	<Obj>_MBSEPAM40C	MBSEPAM40C	$PSxSepam40C	
			<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)	MBSEPAM40C ESEPAM80C	<Obj>_MBSEPAM80C	MBSEPAM80C	$PSxSepam80C	
			<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		

Pattern File (& ref. to specific Include Rules)	Asset Type	DFB Type	Variable Naming Convention[1]	Variable Data Type	ASP Template	Comments
aPSxAccuSine *iPSxCommon.ixml*	Harmonic Filters (AccuSine)	EACCUSINE	<Obj>_EACCUSINE	EACCUSINE	$PSxAccuSine	
			<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)	MBPM1200	<Obj>_MBPM1200	MBPM1200	$PSxPM1200	
			<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)	MBPM5350	<Obj>_MBPM5350	MBPM5350	$PSxPM5350	
			<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 Explicit Messaging)	EMPM53xx	<Obj>_EMPM53xx	EMPM53xx	$PSxPM53xx	
			<Obj>PM_CFG	PM_CFG_DDT		
			<Obj>_PM_ST	PM_ST_DDT		
			<Obj>_PM_MEA	PM53xx_MEA_DDT		
aPSxPM82xx *iPSxCommon.ixml*	Power Meters (Power Meter PM82xx MBTCP Explicit Messaging)	EMPM82xx	<Obj>_EMPM82xx	EMPM82xx	$PSxPM82xx	
			<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	$PSxPM710	
			<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	<Obj>_MBPM800	MBPM800	$PSxPM800	
		EPM800	<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	$PSxPM9C	
			<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	$PSxATS22	
			<Obj>_ATS22_CFG	ATS22_CFG_DDT		
			<Obj>_ATS22_ST	ATS22_ST_DDT		

Pattern File (& ref. to specific Include Rules)	Asset Type	DFB Type	Variable Naming Convention[1]	Variable Data Type	ASP Template	Comments
aPSxATS48 *iPSxCommon.ixml*	Soft Starters (ATS - Altistart 48 Progressive Starter)	MBATS48	<Obj>_MBATS48	MBATS48	$PSxATS48	
			<Obj>_ATS_CFG	ATS_CFG_DDT		
			<Obj>_ATS_ST	ATS_ST_DDT		
aPSxATV212 *iPSxCommon.ixml*	Speed Drives (ATV - Speed Drive)	MBATV212	<Obj>_MBATV212	MBATV212	$PSxATV212	
			<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 MBATV7161 PBATV7161 EMESATV7161 EATV7161 EATV32 ASATV7161	<Obj>_MBATV	MBATV	$PSxATVMainData	
			<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		
aPSxATVAllData *iPSxCommon.ixml*	Speed Drives (Speed drives ATV61 & ATV71 & ATV32 MB TCP IO Scanner; ATV61 & ATV71 CANopen)	ATV7161 EATV32	<Obj>_ATV7161	ATV7161	$PSxATVAllData	
			<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		
aPSxATV6xxAllData *iPSxCommon.ixml*	Speed Drives (Altivar Process Variable Speed Drives ATV6xx)	ATV6xx	<Obj>_ATV6xx	ATV6xx	$PSxATV6xxAllData	
			<Obj>_ATV_CFG	ATV6xx_CFG_DDT		
			<Obj>_ATV_ST	ATV6xx_ST_DDT		
			<Obj>_ATV_IO	ATV6xx_IO_DDT		
			<Obj>_ATV_IOEXT	ATV6xx_IOEXT_DDT		
aPSxATV9xxAllData *iPSxCommon.ixml*	Speed Drives (Altivar Process Variable Speed Drives ATV9xx)	ATV9xx	<Obj>_ATV9xx	ATV9xx	$PSxATV9xxAllData	
			<Obj>_ATV_CFG	ATV9xx_CFG_DDT		
			<Obj>_ATV_ST	ATV9xx_ST_DDT		
			<Obj>_ATV_IO	ATV9xx_IO_DDT		
			<Obj>_ATV_IOEXT	ATV9xx_IOEXT_DDT		
aPSxATV6xxAllDataandWarnings *iPSxCommon.ixml*	Speed Drives (Altivar Process Variable Speed Drives ATV6xx with Process Warnings)	ATV6xx	<Obj>_ATV6xx	ATV6xx	aPSxATV6xxAllDataandWarnings	
			<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		

Pattern File (& ref. to specific Include Rules)	Asset Type	DFB Type	Variable Naming Convention[1]	Variable Data Type	ASP Template	Comments
			<Obj>_EMATVWARN_ST	EMATVWARN_ST_DDT		
aPSxATV6xxxAllDataandWarnings *iPSxCommon.ixml*	Speed Drives (Altivar Process Variable Speed Drives ATV6xxx)	ATV6xxxWarn	<Obj>_ATV6xxxWarn	ATV6xxxWarn	$PSxATV6xxxAllDataandWarnings	
			<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		
aPSxATV9xxAllDataandWarnings *iPSxCommon.ixml*	Speed Drives (Altivar Process Variable Speed Drives ATV9xx with Process Warnings)	ATV9xx	<Obj>_ATV9xx	ATV9xx	$PSxATV9xxAllDataandWarnings	
			<Obj>_ATV_CFG	ATV9xx_CFG_DDT		
			<Obj>_ATV_ST	ATV9xx_ST_DDT		
			<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	
			<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	
			<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	Motor Controllers and Starters	ETESYST TESYSTCTL	<Obj>_ETESYST	ETESYST	$PSxTesysTMainData	
			<Obj>_TESYSTCTL	TESYSTCTL		

Pattern File (& ref. to specific Include Rules)	Asset Type	DFB Type	Variable Naming Convention[1]	Variable Data Type	ASP Template	Comments
iPSxCommon.ixml	(TesysT - Motor Controllers and Starters)		**<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	
			<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	
			<Obj>_TESYSUSCST	TESYSUSCST		
			<Obj>_TESYSUCTL	TESYSUCTL		
			<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))	MBTESYSUSC TESYSUSC	<Obj>_MBTESYSUSC	MBTESYSUSC	$PSxTesysUMEC	
			<Obj>_TESYSUSC	TESYSUSC		
			<Obj>_TESYSU_CFG	TESYSU_CFG_DDT		
			<Obj>_TESYSU_ST	TESYSU_ST_DDT		
			<Obj>_TESYSU_MEC	TESYSU_MEC_DDT		
aPSxTesysTMainDataPB *iPSxCommon.ixml*	Motor Controllers and Starters (TesysT on Profibus - Motor Controllers and Starters)	PBTESYST	<Obj>_PBTESYST	PBTESYST	$PSxTesysTMainDataPB	
			<Obj>_TESYSTPB_CFG	TESYST_CFG_DDT		
			<Obj>_TESYSTPB_SST	TESYST_ST_DDT		
aPSxATV6xxxAllData *iPSxCommon.ixml*	Speed Drives (Altivar Process Variable Speed Drives ATV6xxx)	ATV6xxx	<Obj>_ATV6xxx	ATV6xxx	$PSxATV6xxxAllData	
			<Obj>_ATV_CFG	ATV6xxx_CFG_DDT		
			<Obj>_ATV_ST	ATV6xxx_ST_DDT		
			<Obj>_ATV_IO	ATV6xxx_IO_DDT		
			<Obj>_ATV_IOEXT	ATV6xxx_IOEXT_DDT		
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 Asset Link pattern names.

NOTE: The patterns describe in this section are available after you install Asset Link 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 Asset Link process pattern names:

Num-ber	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
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

Number	Sub-family	ASP template	Pattern name
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 Asset Link device pattern names:

Number	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

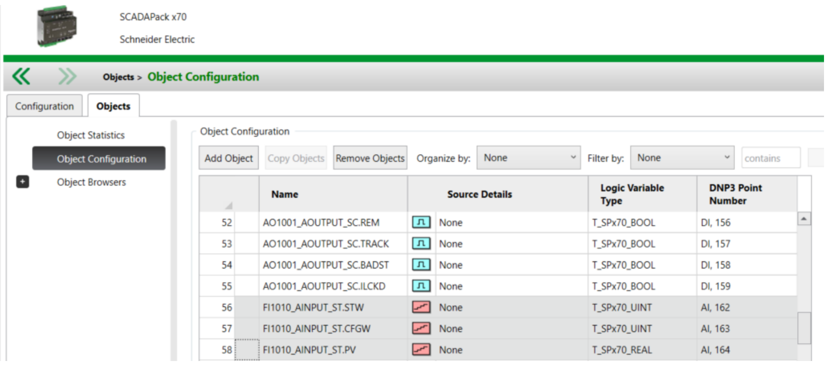
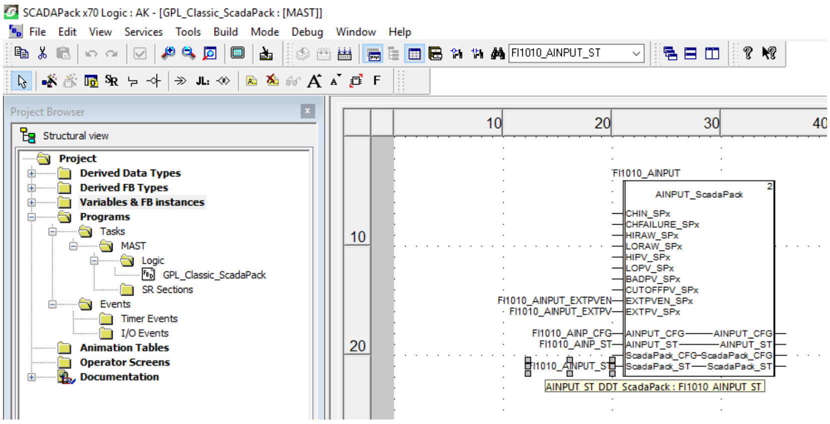
Number	Sub-family	ASP template	Pattern name
13	Digital Protection Relays	$PSxSepam20CSTM	aPSxSepam20CSTM
14	Digital Protection Relays	$aPSxSepam40C	aPSxSepam40C
15	Digital Protection Relays	$aPSxSepam80C	aPSxSepam80C
16	Accusine	$aPSxAccuSine	aPSxAccuSine
17	Power Meters	$aPSxPM1200	aPSxPM1200
18	Power 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
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 for demo Templates.

NOTE: Library set for SCADAPack is not included. The demo template and patterns are in the installation default location `C:\ProgramData\Schneider Electric\Ecostruxure Control Expert - Asset Link\SCADAPack Demo Templates and Patterns`.

Step Number	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. 

SCADAPack x70

Schneider Electric

Objects > Object Configuration

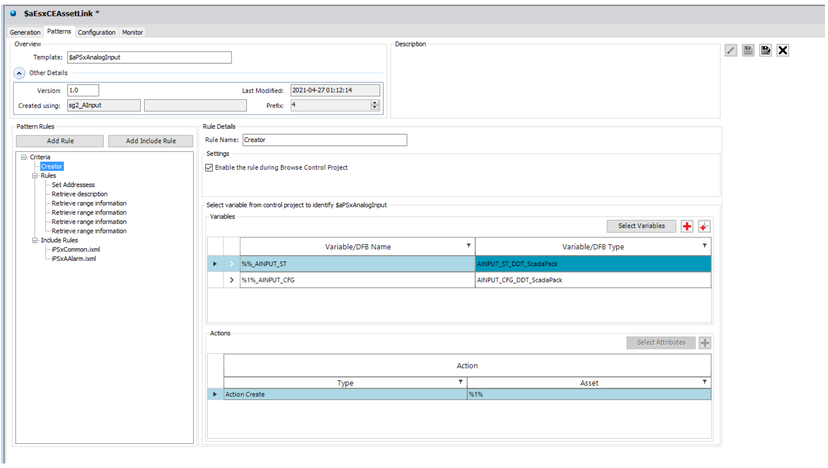
Organize by: None

Filter by: None

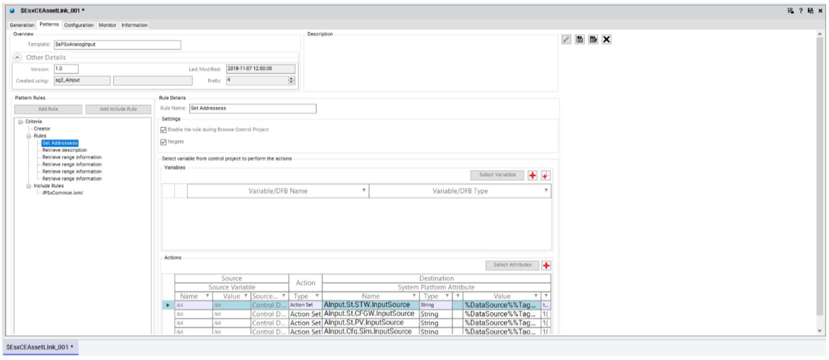
	Name	Source Details	Logic Variable Type	DNP3 Point Number
52	AO1001_AOUTPUT_SC.REM	None	T_SPx70_BOOL	DI, 156
53	AO1001_AOUTPUT_SC.TRACK	None	T_SPx70_BOOL	DI, 157
54	AO1001_AOUTPUT_SC.BADST	None	T_SPx70_BOOL	DI, 158
55	AO1001_AOUTPUT_SC.ILCKD	None	T_SPx70_BOOL	DI, 159
56	FI1010_AINPUT_ST.STW	None	T_SPx70_UINT	AI, 162
57	FI1010_AINPUT_ST.CFGW	None	T_SPx70_UINT	AI, 163
58	FI1010_AINPUT_ST.PV	None	T_SPx70_REAL	AI, 164

SCADAPack x70 Logic : AK - [GPL_Classic_ScadaPack : [MAST]]

<table>
<tr><th>Step Number</th><th>Description</th></tr>
<tr><td>3</td><td>
<p>The figure below is a screenshot of the Pattern Editor wherein the pattern is designed to match the respective function block.</p>



<p>$aEsxCEAssetLink *</p>
<p>Generation | Patterns | Configuration | Monitor</p>
<p>Overview
Template: $aPSxAnalogInput
Other Details
Version: 1.0
Last Modified: 2021-04-27 01:12:14
Created using: sg2_AInput
Prefix: 4</p>
<p>Description</p>
<p>Pattern Rules
Add Rule | Add Include Rule</p>
<p>Criteria
- Creator
- Rules
-- Set Addressess
-- Retrieve description
-- Retrieve range information
-- Retrieve range information
-- Retrieve range information
-- Retrieve range information
- Include Rules
-- iPSxCommon.ixml
-- iPSxAAlarm.ixml</p>
<p>Rule Details
Rule Name: Creator
Settings
☑ Enable the rule during Browse Control Project</p>
<p>Select variable from control project to identify $aPSxAnalogInput
Variables
Select Variables</p>
<table>
<tr><th>Variable/DFB Name</th><th>Variable/DFB Type</th></tr>
<tr><td>%%_AINPUT_ST</td><td>AINPUT_ST_DDT_ScadaPack</td></tr>
<tr><td>%1%_AINPUT_CFG</td><td>AINPUT_CFG_DDT_ScadaPack</td></tr>
</table>
<p>Actions
Select Attributes</p>
<table>
<tr><th colspan="2">Action</th></tr>
<tr><th>Type</th><th>Asset</th></tr>
<tr><td>Action Create</td><td>%1%</td></tr>
</table>
</td></tr>
<tr><td>4</td><td>
<p>The figure below shows the Action set parameters for the respective RTU object parameter named <code>STW</code>, <code>CFGW</code> and <code>PV</code>. The respective System Platform attribute must be named as shown below. For example, the input source STW named as <code>AInput.St.STW.InputSource</code>, wherein, <code>STW</code> mapped with RTU object parameter is same as <code>STW</code>. Similarly, it is applicable for other parameters like <code>CFGW</code>, <code>PV</code>.</p>



<p>$EsxCEAssetLink_001 *</p>
<p>Generation | Patterns | Configuration | Monitor | Information</p>
<p>Overview
Template: $aPSxAnalogInput
Other Details
Version: 1.0
Last Modified: 2018-11-07 12:00:00
Created using: sg2_AInput
Prefix: 4</p>
<p>Description</p>
<p>Pattern Rules
Add Rule | Add Include Rule</p>
<p>Criteria
- Creator
- Rules
-- Set Addressess
-- Retrieve description
-- Retrieve range information
-- Retrieve range information
-- Retrieve range information
-- Retrieve range information
- Include Rules
-- iPSxCommon.ixml</p>
<p>Rule Details
Rule Name: Set Addressess
Settings
☑ Enable the rule during Browse Control Project
☑ Negate</p>
<p>Select variable from control project to perform the actions
Variables
Select Variables</p>
<table>
<tr><th>Variable/DFB Name</th><th>Variable/DFB Type</th></tr>
</table>
<p>Actions
Select Attributes</p>
<table>
<tr><th colspan="3">Source</th><th>Action</th><th colspan="4">Destination</th></tr>
<tr><th colspan="3">Source Variable</th><th></th><th colspan="4">System Platform Attribute</th></tr>
<tr><th>Name</th><th>Value</th><th>Source...</th><th>Type</th><th>Name</th><th>Type</th><th>Value</th><th></th></tr>
<tr><td>NA</td><td>NA</td><td>Control D...</td><td>Action Set</td><td>AInput.St.STW.InputSource</td><td>String</td><td>%DataSource%%Tag...</td><td>1...</td></tr>
<tr><td>NA</td><td>NA</td><td>Control D...</td><td>Action Set</td><td>AInput.St.CFGW.InputSource</td><td>String</td><td>%DataSource%%Tag...</td><td>1(</td></tr>
<tr><td>NA</td><td>NA</td><td>Control D...</td><td>Action Set</td><td>AInput.St.PV.InputSource</td><td>String</td><td>%DataSource%%Tag...</td><td>1(</td></tr>
<tr><td>NA</td><td>NA</td><td>Control D...</td><td>Action Set</td><td>AInput.Cfg.Sim.InputSource</td><td>String</td><td>%DataSource%%Tag...</td><td>1(</td></tr>
</table>
<p>$EsxCEAssetLink_001 *</p>
</td></tr>
</table>

Appendices

What’s in This Part

Library Installation

What’s in This Chapter

Overview

You can install the Modicon Libraries - General Purpose for AVEVA System Platform in System Platform IDE. Default installation location for library template is `C:\ProgramData\Schneider Electric\Ecostruxure Control Expert - Asset Link\GPL WSP Templates`.

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

- Script function libraries.
- The Galaxy style library.

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 Process Expert for AVEVA System Platform 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 System Platform 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**.

Step	Action
14	Click **Galaxy** > **Import** > **Object(s)**.
15	Select the *GPL for ASP Application Templates yyyymmdd.aaPKG* file and click **Open**.

Glossary

A

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™ 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.

S

System Platform IDE:

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

Index

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.