



Process Expert

Global Templates

Reference Manual

EIO0000001986.07
05/2023



Legal Information

The information provided in this document contains general descriptions, technical characteristics and/or recommendations related to products/solutions.

This document is not intended as a substitute for a detailed study or operational and site-specific development or schematic plan. It is not to be used for determining suitability or reliability of the products/solutions for specific user applications. It is the duty of any such user to perform or have any professional expert of its choice (integrator, specifier or the like) perform the appropriate and comprehensive risk analysis, evaluation and testing of the products/solutions with respect to the relevant specific application or use thereof.

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

This document and its content are protected under applicable copyright laws and provided for informative use only. No part of this document 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 document or its content, except for a non-exclusive and personal license to consult it on an "as is" basis.

Schneider Electric reserves the right to make changes or updates with respect to or in the content of this document or the format thereof, at any time 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 document, as well as any non-intended use or misuse of the content thereof.

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.

© 2023 – Schneider Electric. All rights reserved.

Table of Contents

| | |
|--|----|
| Safety Information | 7 |
| Before You Begin | 7 |
| Start-up and Test | 8 |
| Operation and Adjustments | 9 |
| About the Book | 10 |
| Introduction | 15 |
| Global Templates Creation | 16 |
| Guidelines | 16 |
| Overview of the Template Creation Process | 17 |
| Object Model Description | 19 |
| Global Templates Description | 20 |
| Purpose of Global Templates | 20 |
| Types of Global Templates | 21 |
| Facet Templates | 23 |
| Interface Models | 25 |
| Composite Templates | 30 |
| Template Usage Considerations | 32 |
| Element Selection | 32 |
| Parameter Configuration | 34 |
| Interface Links | 35 |
| Constituent Generation | 36 |
| Consistency Check | 37 |
| Main Template Creation Strategies | 39 |
| Modifying Schneider Electric Templates | 39 |
| Modifying Schneider Electric Templates | 39 |
| Template Composition Strategy | 40 |
| Template Composition Strategy | 40 |
| Template Element Layout | 42 |
| Data Propagation | 43 |
| Data Propagation | 43 |
| Generating and Propagating Parameter and Variable Names | 45 |
| Naming Conventions | 48 |
| Naming Convention for Control | 48 |
| Naming Convention for Supervision | 53 |
| Storage Considerations | 58 |
| Global Templates Library Structure | 58 |
| Content Repository | 59 |
| Process Expert Database | 61 |
| Global Templates Definition | 62 |
| Interface Model Definition | 62 |
| Components of the Interface Model Definition | 62 |
| Interface Model Header | 63 |
| Interface Model Elements | 66 |
| Transformation Functions | 68 |
| Interface Model Element Rules | 72 |
| Facet Template Definition | 73 |
| Components of the Facet Template Definition | 73 |

| | |
|--|-----|
| Facet Template Properties | 74 |
| Composite Template Definition..... | 75 |
| Components of the Composite Template Definition | 75 |
| Composite Template Properties (Header)..... | 76 |
| Composite Template Element Rules | 77 |
| Common Definitions..... | 80 |
| Header and Properties Common Definition..... | 80 |
| Document Outline | 82 |
| Attribute Common Definition | 84 |
| Parameter Common Definition | 85 |
| Interface Rules..... | 88 |
| Elements..... | 92 |
| General Description of Elements | 93 |
| Calculated Variable Tag Supervision Element..... | 93 |
| Disk Variable Tag Supervision Element | 95 |
| Equipment Supervision Element..... | 96 |
| Equipment Generation Examples | 103 |
| Equipment Parameter Supervision Element | 105 |
| Equipment Group of Messages Supervision Element..... | 107 |
| Binding Functions | 108 |
| Concatenation Functions | 108 |
| Comparison Functions..... | 110 |
| Logical Functions | 112 |
| Mathematical Functions..... | 113 |
| Multiplexer Function | 114 |
| Fan-In Function..... | 115 |
| CrDocument Function for Runtime Navigation Services | 115 |
| Url Function for Runtime Navigation Services | 118 |
| Miscellaneous Functions | 119 |
| LocationGet Instance Path Calculation Function | 121 |
| LocationExtract Instance Path Extraction Function | 123 |
| Topological Address Calculation Functions..... | 124 |
| Other Address Calculation Functions | 128 |
| Control And Supervision Resources | 136 |
| Prerequisites..... | 137 |
| Control Constituents | 137 |
| Preparing Control Constituents..... | 137 |
| Supervision Constituents..... | 140 |
| Creating and Modifying Supervision Genies | 140 |
| Preparation | 141 |
| Defining Services Provided by Templates | 142 |
| Function Analysis..... | 142 |
| Making a Sketch | 143 |
| Defining Control Services | 145 |
| Defining Supervision Services | 147 |
| Defining Parameters of Elements..... | 151 |
| Parameter Description..... | 151 |
| Creating and Customizing Parameters | 153 |
| Defining Links and Linking Capabilities..... | 157 |
| Working Principle..... | 157 |
| Types of Interface Links..... | 158 |

| | |
|--|-----|
| Other Interface Link Aspects | 163 |
| Materializing the Template | 167 |
| Reusing Global Templates | 167 |
| Defining a Naming Convention | 168 |
| Defining Instance Appearance | 171 |
| Managing, Modifying, and Creating Templates | 174 |
| Template Creation Wizard | 175 |
| Overview | 176 |
| Template Creation Wizard – Composite Window | 180 |
| Template Creation Wizard Control Facet Windows 1/2 | 182 |
| Template Creation Wizard Control Facet Window 2/2 | 184 |
| Template Creation Wizard Supervision Data Facet Window 1/2 | 190 |
| Template Creation Wizard Supervision Data Facet Window 2/2 | 192 |
| Template Creation Wizard Genie Facet Window 1/2 | 197 |
| Template Creation Wizard Genie Facet Window 2/2 | 199 |
| Template Creation Wizard Composition Summary Window | 201 |
| Identifier Management | 203 |
| Creating Templates by Using the Wizard | 204 |
| Template Creation Example | 207 |
| Template Creation Example – Control Logic Facet 1/2 | 208 |
| Template Creation Example – Control Logic Facet 2/2 | 209 |
| Template Creation Example – Supervision Data Facet 1/2 | 212 |
| Template Creation Example – Supervision Data Facet 2/2 | 212 |
| Template Creation Example – Supervision Genie Facet 1/2 | 216 |
| Template Creation Example – Supervision Genie Facet 2/2 | 217 |
| Template Creation Example – Summary | 218 |
| Global Templates Editors | 220 |
| Opening Global Templates Editors | 220 |
| Opening Global Templates Editors | 220 |
| Interface Editor | 221 |
| Interface Editor | 221 |
| Facet and Composite Template Editors | 227 |
| Facet Editor | 227 |
| Composite Editor | 230 |
| Common Template Editor Components | 233 |
| Common Template Editor Toolbars and Menus | 233 |
| Common Template Editor Panes | 238 |
| Common Template Editor Filters | 240 |
| Using the Radar View | 241 |
| Using the Simplified View in Template Editors | 242 |
| Editing, Creating, and Saving Global Templates | 244 |
| Creating and Managing Bindings | 244 |
| Creating Bindings by Using the Mouse | 245 |
| Creating Bindings by Using Context Menu Commands | 245 |
| Using the Create Bindings Dialog Box | 246 |
| Managing Bindings | 248 |
| Creating and Managing Global Templates | 252 |
| Creating Global Templates | 252 |
| Saving Changes In Global Templates | 254 |
| Copying and Pasting Global Templates and Folders | 256 |
| Configuring Interface Models | 257 |

| | |
|--|-----|
| Configuring Interface Models | 257 |
| Configuring Facet Templates | 260 |
| Encapsulating Control Constituents | 260 |
| Modifying Encapsulated Control Constituents | 270 |
| Genie Creation Workflows | 272 |
| Using, Creating, and Modifying Supervision Animated Graphics | 273 |
| Encapsulating and Configuring Genies | 278 |
| Modifying and Replacing Encapsulated Genies | 281 |
| Configuring Facet Templates | 283 |
| Configuring Composite Templates | 286 |
| Configuring Composite Templates | 286 |
| Configuring Control Module Templates | 289 |
| Configuring Control Module Templates | 289 |
| Updating and Replacing Templates | 291 |
| Updating and Replacing Templates | 292 |
| Template Modification Strategy | 292 |
| Updating Global Templates and Templates of Elements | 294 |
| Updating Global Templates at the Folder Level | 297 |
| Replacing the Template of an Element of a Global Template | 300 |
| Duplicating Global Templates | 305 |
| Replace Dialog Box | 308 |
| Editing and Extending Interfaces In-Place | 308 |
| Updating References in Parent After Editing Child Element – Example | 312 |
| Validating Templates | 317 |
| Validating Templates | 318 |
| Validating Templates | 318 |
| Glossary | 320 |
| Index | 329 |

Safety Information

Important Information

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



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



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

DANGER

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

WARNING

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

CAUTION

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

NOTICE

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

Please Note

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

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

Before You Begin

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

⚠ WARNING

UNGUARDED EQUIPMENT

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

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

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

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

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

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

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

Start-up and Test

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

⚠ WARNING

EQUIPMENT OPERATION HAZARD

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

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

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

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

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

Before energizing equipment:

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

Operation and Adjustments

The following precautions are from the NEMA Standards Publication ICS 7.1-1995:

(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 composition, working principles, and main creation strategies of templates of the EcoStruxure Process Expert General Purpose Library Classic.

It also describes the process and provides guidelines to create and modify Global Templates by using the various template editors and software Participants.

The information contained in this manual applies to the library in general but focuses on the creation or modification of process application templates. As such, it may not cover aspects that are specific to other types and categories of Global Templates (for example, topological templates).

This document does not describe how to use the software and Global Templates to create systems.

This document is written for users with a working knowledge of EcoStruxure Process Expert and its Supervision and Control Participants.

Validity Note

This document has been updated for the release of EcoStruxure Process Expert 2023.

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 |
|--|---------------------|
| EcoStruxure™ Process Expert, Security Deployment Guide | EIO0000004234 (ENG) |
| EcoStruxure™ Process Expert, User Guide | EIO0000001114 (ENG) |
| EcoStruxure™ Process Expert, Foundation Application Templates, User Guide | EIO0000002403 (ENG) |
| EcoStruxure™ Process Expert, General Purpose Library, User Guide | EIO0000004045 (ENG) |
| EcoStruxure™ Process Expert, General Purpose Library Process Templates, Reference Manual | EIO0000004043 (ENG) |
| EcoStruxure™ Process Expert, General Purpose Library Device Templates, Reference Manual | EIO0000004044 (ENG) |
| EcoStruxure™ Process Expert, General Purpose Library Classic Process Templates, Reference Manual | EIO0000000987 (ENG) |
| EcoStruxure™ Process Expert - General Purpose Library Classic Device Templates Reference Manual | EIO0000001308 (ENG) |
| EcoStruxure™ Process Expert, General Purpose Library Classic Communication Templates, Reference Manual | EIO0000001311 (ENG) |
| EcoStruxure™ Process Expert, General Purpose Library Classic Diagnostic Templates, Reference Manual | EIO0000001526 (ENG) |
| EcoStruxure™ Process Expert, Control Participant Services, User Guide | EIO0000001524 (ENG) |

| Title of documentation | Reference number |
|---|---------------------|
| EcoStruxure™ Process Expert, Supervision Participant Services, User Guide | EIO0000001525 (ENG) |
| EcoStruxure™ Process Expert, Installation and Configuration Guide | EIO0000001255 (ENG) |

mySchneider Support Portal

Visit <https://www.se.com/myschneider> for support, software updates, and latest information on EcoStruxure Process Expert.

Product Related Information

| ⚠ WARNING |
|--|
| <p>LOSS OF CONTROL</p> <ul style="list-style-type: none"> • 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. <p>Failure to follow these instructions can result in death, serious injury, or equipment damage.</p> |

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.

| ⚠ WARNING |
|---|
| <p>UNINTENDED EQUIPMENT OPERATION</p> <ul style="list-style-type: none"> • Only use software approved by Schneider Electric for use with this equipment. • Update your application program every time you change the physical hardware configuration. <p>Failure to follow these instructions can result in death, serious injury, or equipment damage.</p> |

The examples in this manual are given 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.

NOTE: Templates shown in examples throughout this manual may differ from the actual templates contained in the supplied Schneider Electric libraries.

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. |
| 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.

Introduction

What’s in This Part

Global Templates Creation 16

Global Templates Creation

What's in This Chapter

Guidelines 16

Overview of the Template Creation Process 17

Guidelines

Overview

Modeling functions of an object by using a control module template is a complex task that can be completed in various ways.

The performance, reusability, and ease of use of a template depends on how it is created.

To take advantage of the capabilities of the EcoStruxure Process Expert object model, you need to understand the underlying mechanisms of Global Templates.

Explore the EcoStruxure Process Expert General Purpose Library (Classic) and use its templates as examples.

For more information on the object model, refer to Object Model Concept (see *EcoStruxure Process Expert, User Guide*).

Before Creating Templates

The Foundation and General Purpose libraries contain many templates, which are designed to cover a wide range of automation applications.

In addition, many specialized libraries are available, which target applications in specific segments, such as water, food and beverage, mining, and so on.

Before you start with the creation of templates, verify whether one of these libraries contains a suitable template.

Template Creation Strategies

The strategies that are described in this manual are the fundamental principles by which templates of the EcoStruxure Process Expert General Purpose Library (Classic) are created.

By applying these strategies when you create templates, the following are facilitated:

- The integration of your templates into existing systems.
- The reuse of existing Global Templates.
- The engineering effort required during template and system creation.

Template Design Guidelines and Key Aspects

By using the template editing and creation tools of the software, you can edit and create Global Templates in different ways.

Throughout this manual, guidelines provide you with methods, tips, and other useful information. They may help you create templates for your specific

application by using the same standards as those used for Schneider Electric Global Templates.

The following are key aspects to be considered during the template creation or modification process:

- The position of Functions and Function Blocks (FFBs) inside sections, page 266 of Control resources, page 137 that are encapsulated in facet templates.
- The position of Control facets (elements), page 287 inside composite templates.
- Selecting only the variables that are required, page 137 when creating Control resources that are encapsulated.
- The correct layout of elements inside the template, page 42.
- Using the *\$InstanceID* system parameter, page 48.
- Entering a description for parameters, page 86.
- Complying with the character length limitation for names, page 52.
- Using the @ special character to use the local language in Supervision pages in runtime. For details, refer to *Mark text for translation* in the Supervision Participant help.
- Using the HMI attribute, page 137 for variables that you encapsulate and that are exchanged between the controller and Supervision to optimize performance.
- The correct configuration of composite template element rules, page 77 and interface rules, page 88.

Technical Support

When you create templates by following the creation strategies and design guidelines described in this manual, troubleshooting is facilitated.

If you need assistance, contact your local Schneider Electric service representative.

Overview of the Template Creation Process

Steps to Create Control Module Templates

The table outlines the control module template creation process.

| Step | Task description |
|------|---|
| 1 | Understand template principles and creation strategies. |
| 2 | Understand how Global Templates of the Foundation library and frequently used templates of the General Purpose Library (Classic) work. For example, hardware abstraction layer (HAL) facet templates (such as <i>\$DISignal_UL</i>) and <i>CONDSUM1_UC</i> composite template (management of interlock conditions). |
| 3 | Make a functional analysis of the control module. |
| 4 | Prepare Control and Supervision resources. |
| 5 | Make a sketch of the control module template. |
| 6 | Define Control and Supervision functions. |
| 7 | Define parameters. |
| 8 | Define links and linking capabilities. |
| 9 | Define names of references and elements. |
| 10 | Define the appearance of the instance. |

| Step | Task description |
|------|--|
| 11 | Prepare constituent files. |
| 12 | Create a folder structure in the Global Templates library. |
| 13 | Select Schneider Electric templates to be used and duplicate them. |
| 14 | Create interfaces. |
| 15 | Create Control templates. |
| 16 | Create Supervision templates. |
| 17 | Update templates. |
| 18 | Validate templates. |

Object Model Description

What’s in This Part

| | |
|---|----|
| Global Templates Description | 20 |
| Template Usage Considerations | 32 |
| Main Template Creation Strategies | 39 |
| Storage Considerations | 58 |
| Global Templates Definition..... | 62 |

Overview

This part describes the main characteristics of Schneider Electric Global Templates, usage considerations, and strategies that are applied to create them.

Understanding these aspects allows you to create efficient templates and optimize engineering time.

It also facilitates the integration of your templates into systems that use Schneider Electric templates.

Global Templates Description

What's in This Chapter

| | |
|-----------------------------------|----|
| Purpose of Global Templates | 20 |
| Types of Global Templates | 21 |
| Facet Templates | 23 |
| Interface Models | 25 |
| Composite Templates | 30 |

Overview

This chapter outlines the main characteristics of the different types of Global Templates, which you need to consider when creating templates.

Purpose of Global Templates

Terminology

To make reading of this manual easier, Global Templates are referred to as *templates* from here on.

Purpose of Templates

The purpose of *control module* templates (for example *\$Motor*) is to model Control and Supervision functions of objects of a system in a generic way so that they can be used in a wide range of applications. This is achieved by encapsulating into templates resources, which make abstraction of what is specific to the modeled object in a given installation.

Templates are designed to create instances, to be used in one or many systems, possibly over a long time.

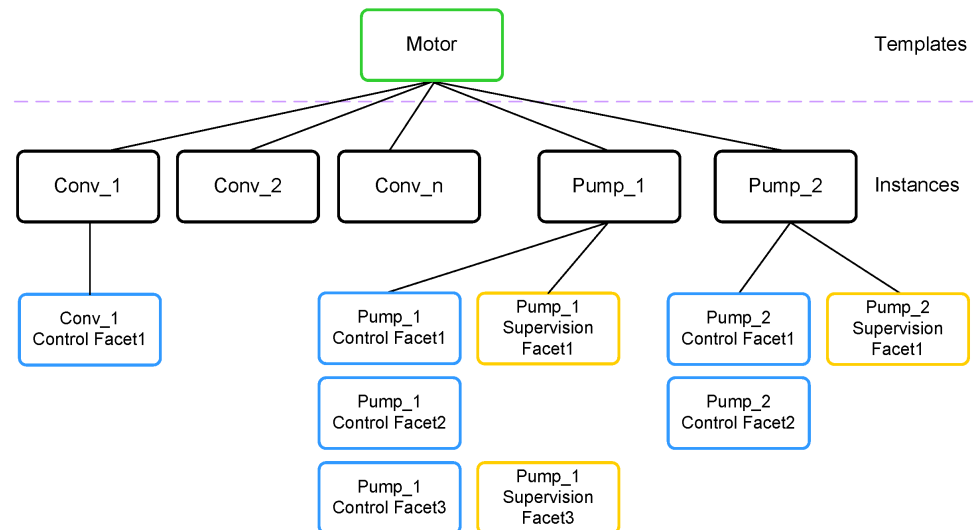
Because templates are generic and encompass a wide scope of use cases, their instances are customizable to meet the requirements of a given system while leaving the template unaltered. Customizing instances reduces the amount of data that is generated, simplifies engineering, and maintenance. Yet, the generated data is consistent with other instances of the same template and with instances of other templates.

For example, from the same *\$Motor* template, you can create two instances, which are quite different:

- Pump_1: Configured to use interlock and diagnostic information management, and associated Supervision services. The motor of the pump is connected to a drive.
- Conveyor_1: Configured to use only the minimum on/off logic without Supervision services, and hardwired to an I/O module.

Finally, templates can be [updated](#), [page 291](#), for example, to keep up with the evolution of technology or to encompass new functions. In turn, their instances can be updated as well.

The following figure illustrates how you can create many different instances from one control module template.



Types of Global Templates

Overview

To provide consistency, engineering efficiency, and reusability, control module templates consist of a hierarchical organization of templates of different types:

- *Facet* templates, page 23: They encapsulate resources that model a functionality of one of the software Participants, such as the Control or Supervision Participant. They are the smallest type of template.
- *Interface* models, page 25: They are a core mechanism to share data among facet templates and between instances.
- *Composite* templates, page 30: They are organizational components, which regroup multiple objects into a single entity.

The complete description of templates and interface models is provided through their definition, page 62.

Toolbox

The **Toolbox** pane of the **Global Templates Explorer** contains empty templates, page 252 of each type.

Type/Category Matrix

The table shows the types of templates and interfaces that exist for the application and topological categories.

| Type of template | Application | | Topological |
|---|---|-------------|--|
| Facet template | Control | Supervision | N/A |
| Interface model | Application | | Physical Communication Mapping |
| Control module templates and their composite elements | Process Device Communication Diagnostic Equipment Modules | | Platform (modeling the hardware and software infrastructure) |

For more information, refer to the respective templates user guides, which are available in the help of the software.

Template Identifier and Version

For each template of the Global Templates library, the combination of the following parameters is key to maintain the integrity of system data:

- Template identifier: The software allows only unique identifiers.
- Template version: Allows various versions of the same template to co-exist and to be used. The template version consists of three components, page 81, which makes it possible to manage changes precisely.

NOTE: A usability state property allows you to manage usage restrictions for each version of a template.

Template Naming Convention

Identifiers of templates provided by Schneider Electric start with the \$ prefix. Some of these templates may not have the \$ prefix to indicate that you can modify them to meet your specific requirements.

In addition, suffixes are used to identify types of templates. Control module templates have no suffix.

NOTE: You cannot use the \$ prefix for templates that you create or modify, page 39.

The table describes the suffixes that are used to identify types of Global Templates.

| Suffix | Description |
|--------|--|
| _UL | Control logic facet template. |
| _UH | Control HMI facet template. |
| _UC | Control logic and HMI composite template. |
| _CD | Supervision data composite and facet templates. |
| _CG | Supervision genie composite and facet templates. |
| _CS | Supervision composite template. |
| _CR | Supervision server composite template. |
| _CC | Supervision client composite template. |

NOTE: Other naming conventions applicable to Schneider Electric templates are described further in this chapter.

Facet Templates

Overview

This topic highlights some of the main characteristics of facet templates that you need to consider when creating templates.

Facet templates are the corner stone of the Global Templates library and the control module templates that you create. They are key to making templates reusable and their instances customizable.

Each facet of a control module template delivers a specific subset (also called *service*) of the complete functionality that is expected from the template. The entirety of the facets assigned to a project creates the logical Participant project.

You can see the same facet template used in several control module templates.

For example, the `$CONDSUM1_UL` facet template generates the `CONDSUM1` DFB, which provides interlock management functions.

For more information on the facet template definition, refer to *Global Templates Definition*, page 62.

Participant

Each facet template is associated to only one software Participant because during the system engineering life cycle, you need to assign facets to the matching Participant project.

Therefore, when you select a facet template from the **Toolbox** of the **Global Templates Explorer**, verify that you select the facet template from the correct category.

For example, to create a Control facet template, select a template of the **Control** category.

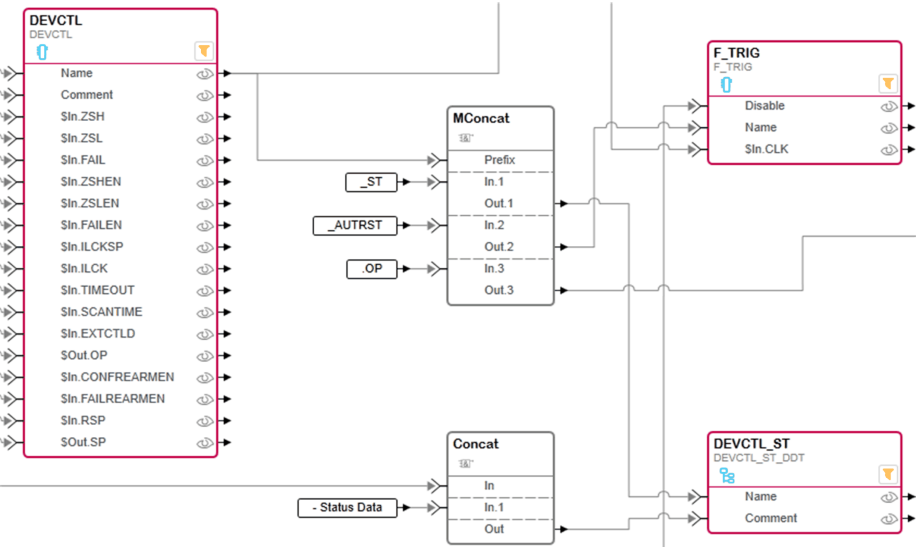
Constituents

Typically, a Control facet template encapsulates one DFB and/or one or more variables.

The constituent, page 137 that is encapsulated in a facet template is not the resource itself (for example, the DFB). It is the definition of the resource that you encapsulate. The resource itself is stored in the content repository, page 59 in the form of a Participant project file.

It is the combination of the resource (from the content repository) and its definition (encapsulated in the facet) that creates the required data in the corresponding project during generation, page 36.

For example, the following figure shows the constituents encapsulated in the *\$DEVCTL_UL* facet template, which are the *DEVCTL* function block and two variables.



NOTE: Certain facets do not encapsulate constituents. For example, Supervision data facets merely create tag data (string) in the databases of the Supervision Participant.

Elements

During instantiation, facet templates create facets, which are elements of the instance. You can view elements, [select](#), [page 32](#) them, and configure their parameters during instantiation.

Elements can be optional and if not selected, they do not generate data.

You can assign rules, [page 33](#) to elements.

The **Instance Editor** allows you to view, for example, the Control facet element **Logic**, which is created by the *\$DEVCTL_UL* facet template. The figure also shows four additional optional facet elements, and the corresponding facet templates (*\$DISignal_UL* and *\$DOSignal_UL*). The version of the templates is indicated.

Motor_1: Instance Editor Status: Valid

| Name | Description | Template | Version |
|---------------------|---------------------------------------|---------------|---------|
| Motor_1 (0/280) | ON / OFF Motor | \$Motor | 2.4.9 |
| Control (0/24) | ON/OFF Motor - Unity Control | \$Motor_UC | 2.2.10 |
| Motor (0/4) | ON/OFF Device Control - Unity Control | \$DEVCTL_UC | 1.0.8 |
| Logic (0) | ON/OFF Device Control Unity Logic | \$DEVCTL_UL | 1.0.8 |
| Running (0) | Digital Input Signal | \$DISignal_UL | 5.3.4 |
| Fail (0) | Digital Input Signal | \$DISignal_UL | 5.3.4 |
| ExternalControl (0) | Digital Input Signal | \$DISignal_UL | 5.3.4 |
| OPDOSignal (0) | Digital Output Signal | \$DOSignal_UL | 5.1.9 |

Identifiers

Facet templates provided by Schneider Electric use specific suffixes to identify them. For example, identifiers of Control facets end with the *_UL* suffix.

Typically, identifiers of Control facets correspond to the name of the resource type that they encapsulate. For example, the facet template that encapsulates the *DEVCTL* DFB type is *\$DEVCTL_UL*.

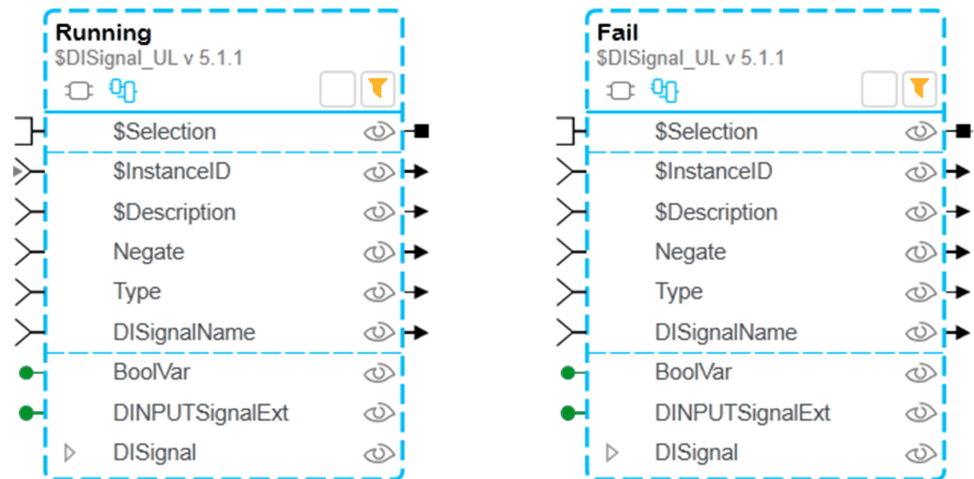
Identifiers of Facet References

References of facet templates, which are the templates that are used inside control module templates have their own identifier.

In the scope of a given control module:

- It allows using the same facet template several times.
- It describes the specific purpose of the facet template.
- The identifier of the reference is displayed in the **Instance Editor** of the instance to identify the service that the facet provides.

For example, within control module *\$Motor*, facet template *\$DISignal_UL* is referenced several times. The identifiers of the references are *Running* and *Fail* among others.



NOTE: A facet, which is an instance of a facet template is associated to the instance it belongs to through the instance identifier, following a naming convention, page 48.

Interface Models

Overview

This topic highlights some of the main characteristics of interface models that you need to consider when creating templates.

Interface models implement a mechanism that allows propagating in one or two directions names of variables, strings, and logical references. This makes it possible to keep separate the services that are provided by the various Participants within the structure of control module templates. By sharing data, interface models allow these services to work together.

For example, they allow propagating the name of a variable from a Control facet to a Supervision facet where this variable name becomes the address of a variable tag. The value held by the variable is resolved in the Participant project.

Data that is propagated can be modified by using transformation functions.

When exposed, interface models allow linking instances.

Finally, interface models make it possible to reuse standard facet templates in many composite templates, making a library versatile, yet consistent.

For more information, refer to the topic describing the interface model definition, page 62.

Shared Data

In the scope of the interface model the data that it shares is called *element*.

The table describes the data that can be shared by interfaces depending on the data source:

| Source of the interface | Shared data | Example |
|-------------------------|-------------------|--|
| Pin of a DFB | Logical reference | Interface <i>\$Real</i> propagating logical reference <i>InstanceID_AINPUT.PV</i> where <i>AINPUT</i> is the name of the DFB and <i>PV</i> the formal parameter. |
| Variable | Variable name | Interface <i>\$AINPUTSignal</i> propagating variable <i>InstanceID_AINPUT_AISV</i> by using data type String. |
| Parameter | Literals | Interface <i>\$Bool</i> propagating value <i>TRUE</i> |
| | Descriptions | Interface <i>\$CondDesc15</i> propagating the diagnostic information from the instance properties to the DFB by using data type String. |

Types

There are different types of interface models, depending on the type of instances or template references they link.

The following types of interface models are predefined:

- **Physical:** Allow data exchange at the physical level (modules) between:
 - Topological references
 - Topological instances
- **Communication:** Allow data exchange at the logical level (I/O scanner, OPC Factory Server software, Supervision I/O devices) between:
 - Topological references
 - Topological instances
- **Application:** Allow data exchange between:
 - Application references/instances
 - Topological references/instances
- **Mapping:** Allow data exchange during the mapping stage between application objects (representing the logical projection of the hardware) and topological objects (representing the actual hardware defined in the topology).

Categories

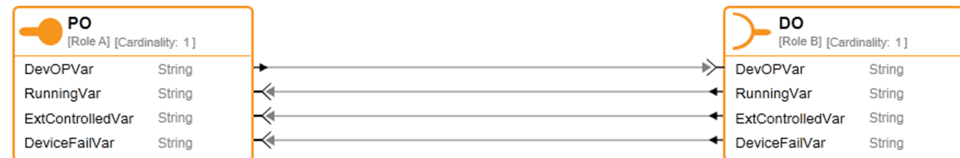
In addition to the four types of interface models, three categories exist, which indicate what data the interface model propagates:

- **Elementary:** To share elementary data types (for example, the name of one boolean variable or its logical reference).
- **Single-element:** To share one element (for example, the name of one structured variable or a string.)
- **Multi-element:** To share several elements of different types. One or more of these elements can be nested interfaces.

The following figure shows the elementary interface *\$Bool*, which is designed to propagate the name of a boolean variable:



The following figure shows the multi-element interface *\$DEV1S1D*, which is designed to link a process object (PO) and a device object (DO). It allows sharing the names of variables, for example, between a motor (PO) and a drive (DO) instance:



Nested Interfaces

An interface model can contain an interface as element, which is called a *nested interface*.

The link between roles of a nested interface is represented like an interface link.

You can view and edit the nested interface by right-clicking it and selecting the corresponding command.

The following figure shows the multi-element interface *\$SDDEVCTLStatus*, which is designed to share, among others, the names of structured variables between a Control and a Supervision facet. It contains a nested interface *Range*.



NOTE: The identifiers of the roles of *\$SDDEVCTLStatus* are the same as those of the nested interface *Range*.

Roles

Interface models have two ends, each one connecting to a different element, allowing them to share data. Each end has a *role*, page 63. By default, roles are defined as *A* and *B* to distinguish them. A different icon for each role serves as a visual indicator.

Only default role *A* can connect to default role *B* of the same interface model.

You can *customize*, page 29 the role to be more explicit. For example, it can indicate the function of the end where role *Def* identifies the end providing data, *Ref* the one receiving data.

In case of multi-element interfaces which may propagate data in both directions, the concept of *Def* and *Ref* roles is not applicable. For these interfaces, the role is used only to differentiate one end from the other because two ends with the same role cannot connect. For example, roles can use the default convention, *A* and *B*, or *PO* and *DO*, which stand for process object and device object respectively.

NOTE: The default execution order is from *A* to *B* but you can change it.

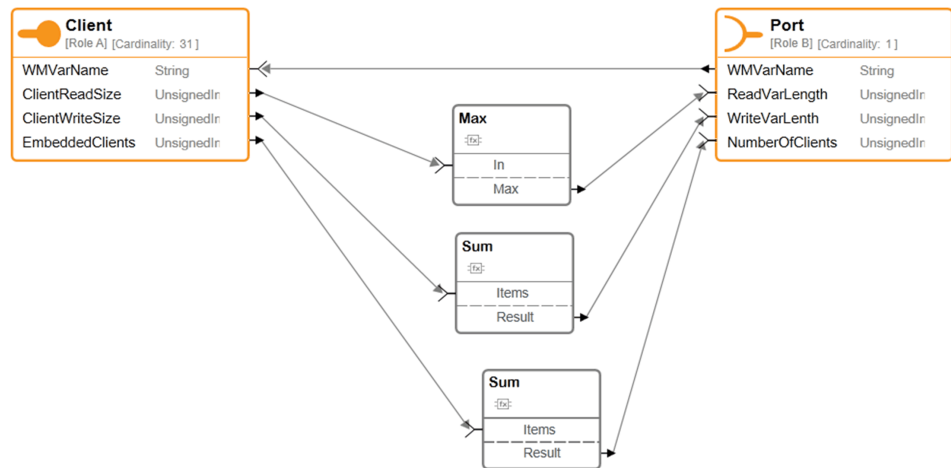
Cardinality

Cardinality, page 64 is a property of interface models that defines the number of connections that each role supports.

Transformation Functions

You can select among several transformation functions, page 68 to be used when you create interface models. They allow transforming an element between role *A* and role *B*.

The following figure shows the *\$MBWorkMemory* interface, which is used to calculate the work memory when serializing client requests in Modbus TCP communications. The cardinality of role *A* (*Client*) allows it to be connected several times. In such case, the transformation functions *Max* and *Sum*, use the values that are provided through each connection to output the result in the corresponding elements of role *B* (*Port*):



Rules

Rules, page 88 are defined for interfaces. They allow, for example, making interface links mandatory, optional, or prohibited. They can also be used to disable elements of an instance or the generation of resources if an interface link is not established.

Rules can also be defined for elements of interfaces, page 72.

NOTE: For nested interfaces, the meaning of interface element rules is different than for regular interface elements.

Exposed Interfaces

When the interface of an instance is exposed at the highest level of the template composition, it allows you to connect it to other instances of the application. Such links, page 35 allow creating a link between two function blocks or a logical reference in the Control Participant project.

You can expose interfaces, by deferring unbound ones. Unbound interfaces are those that have one role that is not connected.

Identifiers

Identifiers of interface models provided by Schneider Electric follow a naming convention that depends on their type.

The table describes the naming convention that is applied to the interface model identifier, the role, and the element identifier:

| Interface model type | Interface model identifier | Role | Element identifier |
|--------------------------|--|--|---|
| Elementary interface | Name of the data type that is propagated. For example, <i>\$Bool</i> . | <i>Def</i> and <i>Ref</i> | <i>Name</i> |
| Single-element interface | Name of the type that is propagated. For example, <i>\$DEVCTL_ST</i> , which is a data structure. | | |
| Multi-element interface | Descriptive name of the set of data that is propagated. For example, <i>\$DEVCTLStatus</i> , which propagates the names of the parameters of the DEVCTL_ST.STW word. | If data is exchanged both ways: <i>A</i> and <i>B</i> or <i>PO</i> and <i>DO</i> ; otherwise <i>Def</i> and <i>Ref</i> . | Descriptive name for each element (for example, <i>Device FailVar</i>), or name of a nested interface. |
| Mapping interface | Descriptive name of the purpose of the interface model. For example, <i>\$AIChannel</i> , which allows reading a variable from an analog input channel. | <i>CO</i> and <i>HO</i> , <i>DO</i> and <i>HO</i> , or <i>SO</i> and <i>HO</i> . | Descriptive name of the channel. For example, <i>MBChannel</i> . |

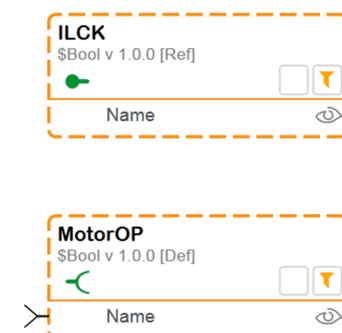
Identifiers of Interfaces

Interfaces, which are instances of interface models that are used inside other templates have their own identifier.

In the scope of a given template:

- It allows using the same interface model several times.
- It describes the specific purpose of the interface.
- The identifier of the interface is visible in the instance in the **Asset Workspace Editor** and **Links Editor**.

For example, within control module *\$Motor*, interface model *\$Bool* is referenced several times. The identifiers of the interfaces are *ILCK* and *MotorOP* among others.



Interface Properties

Interface models have properties that are included by default. They provide the following information.

| Property | Description |
|-----------------------|---|
| \$Disable | When the boolean value provided to the property through a binding, page 244 is TRUE, the interface is disabled, page 165. |
| \$IsConnected | Outputs a boolean value TRUE when the interface link is successfully established, page 165. |
| \$InstanceID | Identifier of the instance linked to the other role of the interface. |
| \$TemplateID | Identifier of the template of the instance linked to the other role of the interface. |
| \$InstanceDesc | Description of the instance linked to the other role of the interface. |

Composite Templates

Overview

This topic describes the main characteristics of composite templates that you need to consider when creating templates.

Composite templates are organizational templates, which help create the structure of control module templates by grouping the services they provide.

For more information on the composite template definition, refer to *Global Templates Definition*, page 62.

Participant Services

Composite templates are associated to a single Participant by containing only facets that provide services of that Participant (for example, the Control Participant).

Control Module Templates

In a template, the highest-level composite template is called the *control module* template. Only composite templates that occupy this position provide services of several Participants (for example, Control and Supervision Participants).

Template Composition

Composite templates are key to implement the *composition strategy*, page 40 when you create templates. They allow grouping services inside templates, and impact the way elements are displayed in the **Instance Editor** and **View Assignments** windows of the **Application Explorer**.

Composite templates can contain facet templates and/or other composite templates.

For example, a composite template allows you to create groups **Control**, **Failures**, and **LocalPanel** to organize the Control logic facets they contain:

| Motor_1 (\$Motor) : Assignment Viewer | | | |
|---------------------------------------|-------------|--------------------|---------------|
| Name | Type | Facet | FacetTemplate |
| Motor_1 | | | |
| Control | | | |
| Fail | Unity Logic | Motor_1_Motor_FAIL | \$DISignal_UL |
| Failures | | | |
| Logic | Unity Logic | Motor_1_CONDSUM | \$CONDSUM_UL |
| LocalPanel | | | |
| Logic | Unity Logic | Motor_1_DEVL | \$DEVL_UL |
| ZERODISignal | Unity Logic | Motor_1_DEVL_ZS | \$DISignal_UL |

Template Identifiers

Schneider Electric composite templates use specific suffixes to identify them. For example, identifiers of Control composite templates end with the `_UC` suffix.

NOTE: Identifiers of control module templates have no suffix. When creating control module templates, choose an identifier that is representative of the function or device that the control module models. For example, `$TeSysTEM`, a Schneider Electric control module that models the functions of a *TeSys T* device communicating through *Explicit Messaging*.

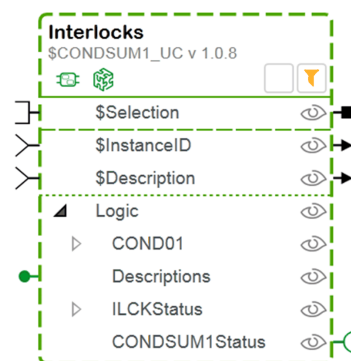
Identifiers of Composite References

References of composite templates, which are instances of composite templates used inside a control module template, have their own identifier.

In the context of a given control module:

- The identifier describes the purpose of the composite template.
- The identifier of the reference is displayed in the **Instance Editor** of the instance to identify a service or a group of services.

For example, within control module `$Motor`, composite template `$CONDSUM1_UC` is referenced with the identifier *Interlocks*.



Template Usage Considerations

What's in This Chapter

| | |
|-------------------------------|----|
| Element Selection | 32 |
| Parameter Configuration | 34 |
| Interface Links | 35 |
| Constituent Generation | 36 |
| Consistency Check..... | 37 |

Overview

This chapter describes the aspects related to the instantiation of templates and the generation of their constituents. They have a direct impact on how you create a template and on how it will be used.

A template designed properly helps reducing engineering time throughout the system engineering life cycle.

Element Selection

Overview

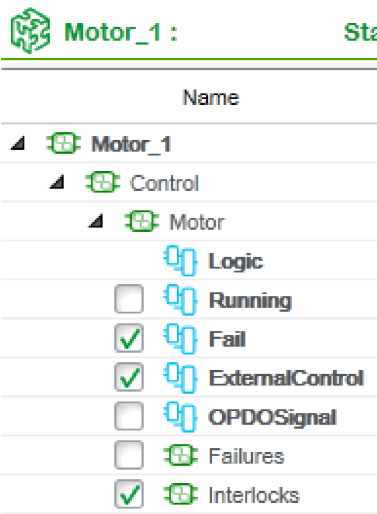
This topic explains the strategy to make elements optional and group them.

To provide flexibility and reusability, a template needs to be built to cover a wide range of applications. Thus, the template needs to contain many facet templates, each one providing a service in the form of an element. By selecting which elements are enabled, each instance of a template can be uniquely configured to provide specific services.

The element selection is the process in which the user selects and clears elements to customize an instance to meet the requirements of the system.

When an element is not selected, its facet is not created and no constituents are generated.

The following figure shows some elements of *Motor_1*, which is an instance of template *\$Motor*. Elements **Fail**, **ExternalControl**, and **Interlocks** are selected.



NOTE: Core elements that have no configurable parameters do not appear in the **Instance Editor** of the **Application Explorer**.

Core and Optional Elements

Unnecessary services should not overload controllers and servers, nor should instances be more complex than necessary.

Therefore, you need to distinguish between the following services:

- **Core:** They are required for the instance to function properly. Constituents of core elements are generated by default. For example, the logic element of a motor template.
- **Optional:** Services that provide additional functions but that are not necessary for the instance to function properly. For example, the maintenance information service of a motor template.

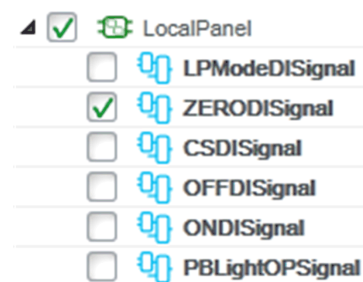
Optional services can be selected as needed during instantiation.

Grouping Elements

Composite templates allow you to group services. The facets that are referenced by a composite template are displayed as elements of that group.

Like you can make a service optional, you can make [groups of services, page 40](#) optional. This is helpful to reduce engineering time during instantiation.

An example of an optional group of services for Schneider Electric templates is **Local Panel**, which is implemented by the composite template `$DEVLP_UC`:



Automatic Element Selection

To reduce the engineering effort during instantiation, it may be convenient to create a mechanism that automatically selects the corresponding Supervision service when the user selects an optional Control service.

The opposite should also be true when the user clears the optional Control service.

For example, in the `$Motor` template, by default, when you select the **Interlocks** Control service, the corresponding Supervision service is automatically selected.

The Control service generates the `CONDSUM1` DFB and the Supervision service creates the necessary tags and enables the Interlocks tab of the motor management faceplate.

Element Rules

Because your template may contain more elements than are required for each instance, you can create [rules for element selection, page 77](#) that help selecting only valid combinations. It allows, for example, making the selection of an element mandatory or a conflicting combination of elements unavailable. This feature helps configuring more complex instances correctly.

NOTE: When an element rule is not satisfied, the **Data** status of the instance becomes **Invalid**.

Parameter Configuration

Overview

This topic explains the aspect of configuring parameters at the instance level and how it impacts template creation.

Constituents that you encapsulate may require parameters to function properly. Some of these parameters may be different for each instance of a template, depending on the system and on the element selection.

Parameter values are generated with the constituents in the logical Participant projects.

The configuration of parameters in the **Instance Editor** of the instance is the process in which the user provides values for selected elements during instantiation. The configuration needs to be flexible to adapt to the various use cases.

NOTE: Only elements that have configurable parameters appear in the **Instance Editor** of the **Application Explorer**. The complete set of parameters of an instance is visible in the **Inspect Instance** window.

Types of Parameters

When creating templates, you can choose among the following methods to provide values to resources and elements:

- Platform inputs: They are part of the template definition and are not configurable at the instance level. (For example, the data type for the **Type** parameter of a variable tag element.)
- User configurable: They can be configured in the **Instance Editor** of the instance. You can assign them a default value; otherwise their value is NULL. (For example, the high and low ends of the range for an analog input instance.)

The method depends on the source that you have identified for the parameter value and the treatment that the value may undergo during the life cycle of the instance.

NOTE: For optional elements of an instance, parameters are editable only if the element is selected.

Other Sources of Values

You can also provide values to resources and elements through [interfaces](#), [page 25](#) and [interface links](#), [page 35](#).

Parameter Definition

For each parameter, you can define a set of [properties](#), [page 85](#) that help entering correct values during instantiation.

Typically, the parameters of each element are grouped by category, which makes it easier to configure them when the number of parameters is high.

For information on the properties of parameters and how to group them, refer to the topic describing how to *define parameters of elements*, [page 85](#).

Interface Links

Overview

This topic explains the aspect of interface links at the instance level and how it impacts template creation.

At the instance level, exposed interfaces, page 25 of selected elements allow interface links to be made in the **Asset Workspace Editor** or **Links Editor**. This mechanism makes it possible to propagate values between Control facets of instances.

For example, an interface link allows an analog input instance to provide the process value to a PID instance. Such a link results in a link between the corresponding pins of the DFBs generated by the facets.

In addition, during hardware mapping, mapping interfaces allow linking facets of application instances to facets of topological instances in order to assign addresses to variables.

For example, the following figure shows highlighted mapping interface *DIChannel* of instance *Valve_1*, which allows propagating the address of a digital input channel. This allows the DFB to read the value of the hardwired low limit switch signal:

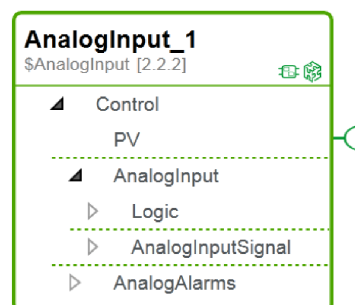
| HWInstance | HWTemplate | HWMappingInterface | HWInterfaceType | AppInstance | AppFacet | AppTemplate | AppMappingInterface | AppInterfaceType |
|--|-------------|---------------------|-----------------|-------------|--------------------|-------------|---------------------|------------------|
| Controller_1 1:Prilocal 1:D 1:R 05:140DDI35300 | \$QDInput32 | DinputCh_01 QDInput | SDIChannel/HO | Valve_1 | Valve_1_ZSL | SValve | DIChannel | SDIChannel/SO |
| Controller_1 1:Prilocal 1:D 1:R 05:140DDI35300 | \$QDInput32 | DinputCh_02 QDInput | SDIChannel/HO | Valve_1 | Valve_1_VALVE_EXTC | SValve | DIChannel | SDIChannel/SO |
| Controller_1 1:Prilocal 1:D 1:R 05:140DDI35300 | \$QDInput32 | DinputCh_03 QDInput | SDIChannel/HO | Valve_1 | Valve_1_VALVE_OP | SValve | DOChannel | SDOChannel/SO |

Interface Definition

The following aspects of the interface definition come into play:

- Interface category.
- Interface model and role for unbound interfaces. Only interfaces using the same interface model and opposite roles can be linked.
- Whether such unbound interfaces are exposed (deferred).
- The side on which the interface appears on the instance.

The following figure shows an instance of a control module template in the **Links Editor** with exposed interface *PV*:



NOTE: The **Interfaces** section of the **Inspect Instance** window provides detailed information on the interfaces that are used.

Interface Rules

To cover the different use cases in which control module templates are used, instances often provide many possibilities to make interface links, some of which may not be compatible. During template creation, you need to foresee links that

affect the proper operation of instances and create rules, page 88 that allow making only valid links.

In addition, you need to create rules, page 72 to define which elements, page 66 of multi-element interfaces are required and which ones are optional.

In the following example, instances of *\$AnalogOutput* are created with an **invalid Link** status because a rule requires either **SPRange**, **RSPRanged**, or **RSPRangedwTrack** interface to be linked to receive the necessary setpoint and range data:

| Identifier | Template | Version | Data | Link |
|----------------|----------------|---------|-------|---------|
| System_1 | | | | |
| Folder_1 | | | | |
| AnalogOutput_1 | \$AnalogOutput | 4.1.16 | Valid | Invalid |

AnalogOutput_1

\$AnalogOutput [3.4.1]

Control

ILCK

AnalogOutput

Logic

RSP

SPRange

RSPRanged

RSPRangedwTrack

CHOUT

OP

AORange

AOSignal

IntVar

AOUTPUTSignalExt

LocalPanel

NOTE: When an interface rule is not satisfied, the **Link** status of the instance is **Invalid**.

Constituent Generation

Overview

This topic explains the aspects of generating data based on encapsulated constituents and how it impacts template creation.

After you have assigned Control and Supervision facets to their respective Participant projects, you can proceed to the generation stage.

During generation, the following create the logical Participant project:

- The definition of resources that are encapsulated in selected elements of the instance.
- The parameter configuration of these elements.
- The actual resources stored in the content repository.

This process takes place for Control and Supervision Participant projects.

Participant Compliant Data

For the generation process to complete successfully, data that is generated needs to satisfy Participant-specific rules. For example, for the Supervision Participant, fields of tags cannot exceed a certain length.

Although some of these rules are verified at the platform level (see *EcoStruxure Process Expert, User Guide*), ensure during template creation that the data that is generated satisfies the rule; otherwise during instantiation, data may need to be modified, which increases engineering time.

For example, the software verifies that the names of variables that are generated by the instance of a template do not exceed the [maximum name length, page 52](#) accepted by the Control Participant. If the mechanism that generates variable names in your template does not take this limitation into account, you may not be able to generate the facet if a user enters a long instance identifier.

Part of the rule that is described in the above example can be implemented through properties of element [parameters, page 52](#).

Version Compatibility

There are restrictions related to versions of resources.

The Control Participant, for example, does not support two versions of the same EFB. Therefore, if you encapsulate an EFB with a version that is different from the version of the EFB that is in the library of the Control Participant, generation does not complete successfully.

Other version-related restrictions can come from the platform. For example, the software does not support two versions of the same Supervision Include project. Therefore, if you encapsulate a genie into a template by adding a later version of the current included project to the content repository, this current included project is replaced with the new version. This may impact existing templates and their instances.

NOTICE

LOSS OF DATA

Resolve a project file conflict that arises when adding a project file to the content repository by selecting the correct file version.

Failure to follow these instructions can result in unusable Supervision projects.

Consistency Check

Overview

This topic explains the aspects of checking the consistency of Control logic that is generated by templates and how it impacts template creation.

The software provides a mechanism, called consistency check (see *EcoStruxure Process Expert, User Guide*), which allows you to identify which modifications have been made to the logic of a Control project during refinement compared to the logic that the software had generated originally.

For example, if during refinement, you delete the link between two DFBs that you had created in the **Asset Workspace Editor** or **Links Editor**, the software informs you of such change when you perform a consistency check.

For this mechanism to provide helpful information, you need to keep this capability in mind when designing templates. Its purpose is to allow you to restore the original logic while keeping refinements on code that the software has not generated.

Examples of such refinements are:

- Setting the pin of a DFB to which no value was assigned to true.

- Entering a comment in a field for which no value was configured at the instance level (the value of the parameter of the instance is NULL).

None of these actions is detected as an inconsistency.

Main Template Creation Strategies

What's in This Chapter

| | |
|--|----|
| Modifying Schneider Electric Templates | 39 |
| Template Composition Strategy | 40 |
| Data Propagation | 43 |
| Naming Conventions | 48 |

Overview

This chapter describes the main strategies that are applied to create Schneider Electric templates and that you should follow when creating your own templates.

Modifying Schneider Electric Templates

Modifying Schneider Electric Templates

Objective

The objective of this strategy, page 292 is to maintain the integrity of the original Global Templates library when you modify templates referenced by control modules and to identify your control modules and referenced templates.

By using different identifiers, it also keeps Schneider Electric templates from being updated with yours.

Guidelines

- You cannot use the \$ prefix for templates that you modify.
- When you modify a template and update a control module template with such an updated reference, save them with an identifier that is different from the original one.
- Store modified and updated templates in your own folder structure, page 58.
- Use the **Duplicate** command, page 305 to automate steps.

Example

If, for example, a Schneider Electric control module template *\$Motor* references facet templates *\$DEVCTL_UL* and *\$DISignal_UL*.

If you decide to modify facet template *\$DEVCTL_UL*, first save it as, for example, *MyDEVCTL_UL*.

Open *\$Motor* and save it as, for example, *MyMotor*. Then, replace *\$DEVCTL_UL* with *MyDEVCTL_UL*. This new control module references *\$DISignal_UL*.

\$Motor and its facet references *\$DEVCTL_UL*, and *\$DISignal_UL* remain unaltered.

NOTE: The identifiers of elements in the scope of the control module composition do not matter. As such, *\$DEVCTL_UL* with *MyDEVCTL_UL* can both have the same identifier, for example *Logic*.

Template Composition Strategy

Overview

This topic explains the aspects related to the template composition and the layout of elements inside facet and composite templates. It describes how they impact template creation and provides guidelines.

Template Composition Strategy

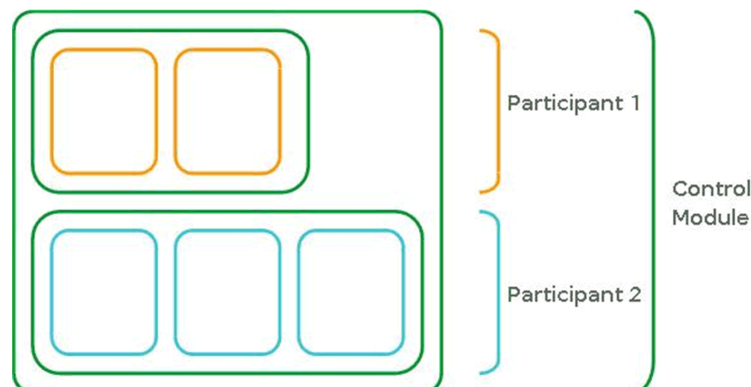
Overview

You can create a control module template by using only one composite template, which references one facet template that contains both the Control and Supervision constituents. However, by following the composition strategy described in this topic, your templates are consistent and compatible with Schneider Electric templates. This makes it possible to integrate duplicates of templates from the Foundation and General Purpose libraries into your own control module templates.

Guidelines

- The first level of granularity is by Participant.
- Position similar functions in different templates of a same level.
- For each group of Control services, create a group that contains the corresponding Supervision services so that they can be selected/unselected together.
- Use interfaces to share data between facets of different Participants.

In the following figure, Participant 1 is the Supervision and Participant 2 the Control Participant.



Representation of the Composition

The composition of your template affects the way the elements of instances thereof are displayed, page 171 in the **Instance Editor**.

The Schneider Electric composition strategy helps users understand how instances function.

Modularity of Services

The objective of highest-level composite templates is to provide functions of control modules as defined in the ISA-S88.01-1995 standard. These functions are delivered by several Participants.

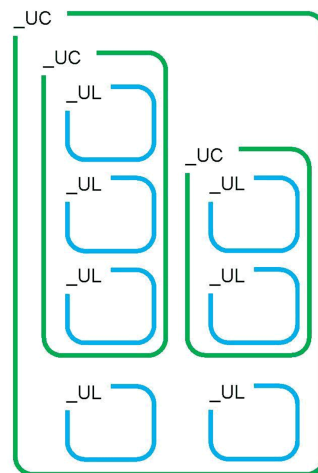
Group services (elements) of each Participant by function with a granularity that provides enough flexibility to adapt to most use cases.

Decoupling of Services

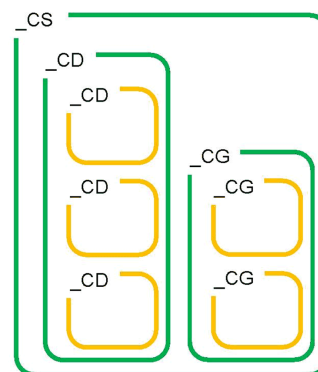
Keep Participant services separate in their respective facet and composite templates. This helps ensure modularity and reusability.

For example, changes in a Control element may not affect the Supervision element because the Control data that the Supervision facet requires is exchanged by using an interface (such as, the variable address in a variable tag). Other data is created inside the facet either through parameters or platform inputs and matches the Supervision components (such as, the name of the variable tag).

The following figure shows how to group Control templates based on their type, page 22:



The following figure shows how to group Supervision templates based on their type, page 22:



NOTE: Supervision data composite and facet templates have the same suffix (`_CD`), and Genie composite and facet templates have the same suffix (`_CG`). In the template editors, they are represented with a blue outline.

Categories of Templates

Avoid mixing inside the same control module template services pertaining to different categories of templates, page 21. For example, do not include in a motor template facets modeling an I/O device, such as, a drive.

However, you can create a template that contains more than one control module. For example, you can integrate into a composite template, a motor, a drive, and a communication port control module template.

Template Element Layout

Overview

Templates can contain many elements of various kinds. Keeping the layout organized helps the person who views or edits the template have a better understanding of the template construction and function.

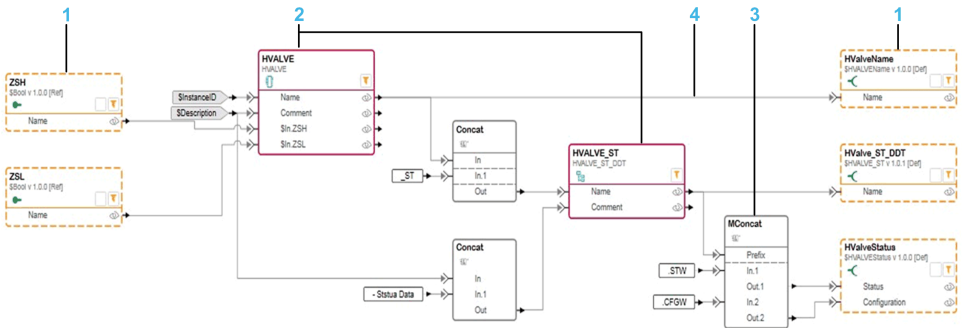
For facet templates, the layout has no functional impact but for Control composite templates, the layout also impacts the execution order of elements, page 287.

Guidelines

- Arrange elements according to the flow of data, from left to right.
- On the outer left-hand side, position interfaces receiving data.
- On the outer right-hand side, position interfaces providing data.
- In the center, arrange parameters, platform inputs, binding functions, facet templates, composite templates, or constituents so that they do not overlap.

Layout Example

The following figure shows an example of layout of elements of a Control facet template as you can see it in the **Facet Editor**.



| Item | Description |
|------|---|
| 1 | Interfaces. On the left-hand side, interfaces that receive data. On the right-hand side, interfaces that propagate data to other elements. |
| 2 | DFB and variable constituents that are encapsulated in the template. |
| 3 | Transformation functions. |
| 4 | Grey lines represent bindings that are used to propagate data, page 43 between constituents, transformation functions, interfaces, parameters, and platform inputs within the template. |

Data Propagation

Overview

This topic explains the mechanisms that are available to generate data and propagate it to and from the following elements:

- Properties of FFBs, variables, Supervision data and genie elements.
- Facet templates.
- Composite templates.
- Interfaces.

Data Propagation

Guidelines

- Use the output pin of an element to propagate data to other elements rather than connecting the source directly to each element.
- Use the *\$FullName* property, page 45 of the pin of a Control element (such as a DFB) to reduce the use of concatenation functions.

Working Principle

The input and output pins, page 93 of elements allow propagating data in a consistent way.

The same data that is supplied through the input pin is made available on the output pin. No transformation occurs in between.

The table describes the various possibilities to provide data to the input pin of an element.

| Data provided through | Purpose | Example use case |
|--|---|--|
| Parameter | Provide a numerical value or a string during instantiation of the template in the Application Explorer . | Setpoint value for a function block. For example, 80. |
| Platform input | Provide a constant numerical value or a string at the template level (not editable during instantiation). | Comment field of an encapsulated DFB. For example, Control logic for motor. |
| Output pin from another element | Share the same data between several elements (may be combined with binding functions). | Sharing the <i>\$InstanceID</i> parameter value across several elements or providing variable names to interfaces. |
| Role <i>Ref</i> of an interface (consumer end) | Provide data from another element within the template or another instance. | Tag addresses received from Control facets through a <i>xStatus</i> multi-element interface. |
| None | Allows providing data during refinement (applies only to constituents). | - |

NOTE: The absence of connection on an input pin results in NULL value at the opposite output pin.

Links Between Elements

So that data can be propagated, links must exist between the source and the destination of the data.

Within a template, use [bindings](#), page 244.

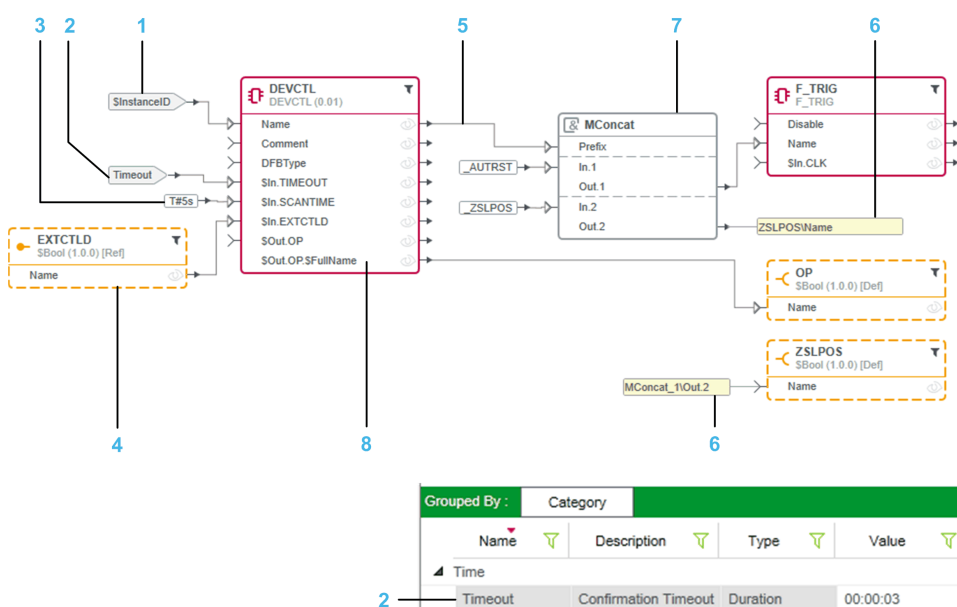
You may be able to create several bindings originating from one output pin.

Various methods are available that let you create bindings and locate their source and destination.

NOTE: Within a composite template, input and output pins of facet templates are readily available for connection by using bindings. However, when you want to connect the parameter of an element that is referenced by another composite template, you need to defer it first to the current template so that its input/output pins become available.

Providing and Propagating Data

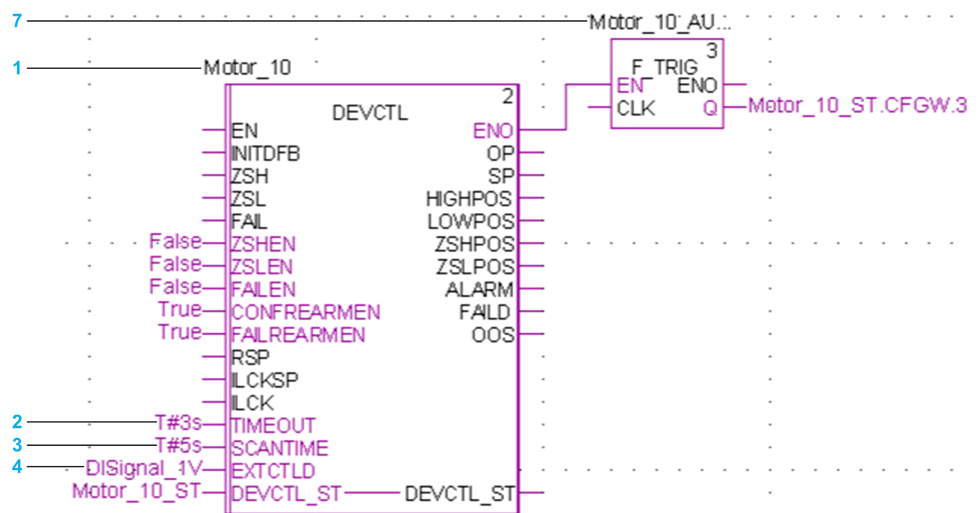
The following example illustrates different possibilities to connect input an output pins of elements to propagate data by using [bindings](#), page 244.



| Item | Description |
|------|---|
| 1 | <i>\$InstanceID</i> : The parameter, page 85 value is assigned to the <i>Name</i> property of the DEVCTL DFB and propagated to other elements of the template. To provide consistency and simplify routing of bindings, the value is taken from the output pin of the <i>Name</i> property rather than connecting the other elements to <i>\$InstanceID</i> . |
| 2 | <i>Timeout</i> : The value is provided through a parameter, page 34 that is editable during instantiation in the Instance Editor of the instance; otherwise, the default value is used if one is configured in the properties of the parameter. The same value is available on the opposite output pin. |
| 3 | <i>Scantime</i> : The value is provided through a platform input, page 34 as a constant during template creation. The same value is available on the opposite output pin. |
| 4 | <i>EXTCTLD</i> : A variable or logical reference is created on the <i>EXTCTLD</i> input pin of the DEVCTL DFB through this interface, page 25 when its other role (<i>Def</i>) is connected. The same value (a string) is available on the opposite output pin and can be propagated further by using, for example, another <i>\$Bool</i> interface. |
| 5 | Binding shown in line style, page 248. |
| 6 | Binding shown in connector style, page 248. |

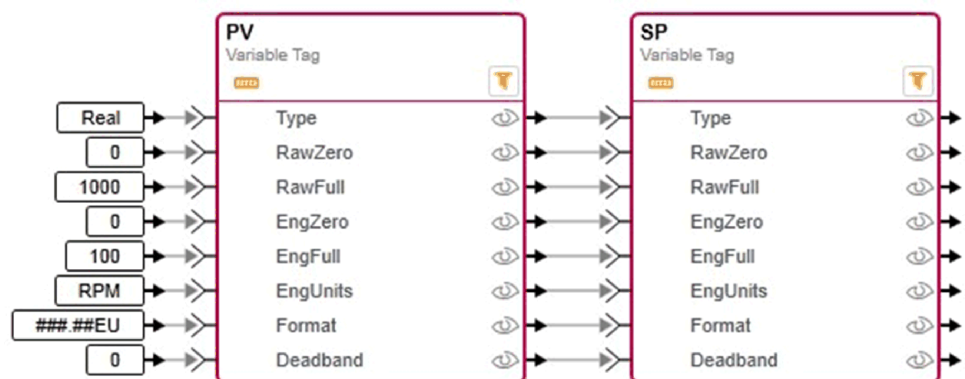
| Item | Description |
|------|--|
| 7 | Before propagating the value of the <i>\$InstanceID</i> parameter to the <i>Name</i> attribute of the F_TRIG DFB, it is modified by concatenating it with the <i>_AURST</i> string (fixed platform input). This is made possible by using a binding function, page 108 (for example, <i>Concat</i> or <i>MConcat</i>). |
| 8 | <p>The <i>\$FullName</i> property, page 45 lets you generate the name of an input or output parameter or DDT child variable. In this example, output parameter <i><InstanceID>.OP</i>. This reduces the need for concatenation functions. The property has only an output pin and its value does not depend on the connection that may exist on the pin of the corresponding input/output (in the example, the <i>OP</i> output pin).</p> <p>The value (string) is propagated to the <i>OP</i> interface by using a binding. When the other role of the interface (<i>Ref</i>) is connected, it can propagate the value further.</p> |

Based on the previous example, the figure shows the result of the five data connections (items 1, 2, 3, 4, and 7) in the Control Participant project. The values that are provided through the *\$InstanceID* parameter and the *EXTCTLD* interface are respectively *Motor_10* and *DISignal_1V*.



Propagating Data to Several Elements

This example shows an effective method to propagate data to several Supervision variable tag properties.



Generating and Propagating Parameter and Variable Names

Overview

The name of a parameter consists of the following components:

- The identifier of the instance in the application that encapsulates the logic.

- The name of the control constituent instance.
- The name of the parameter in case of an FFB or the variable hierarchy in case of a DDT.

For example, *Motor_1_DEVCTL.OP*, which is the name of the *OP* output parameter of the *DEVCTL* DFB generated by the *Motor_1* instance.

To generate the parameter or variable name, you can use either solution:

- A concatenation function, page 108, which requires some manual configuration but gives you more flexibility as to the name that you can generate.
- The *\$FullName* property of an input or output pin of a Control element.

NOTE: You can view the resulting name by using the **Inspect Instance** functionality (see *EcoStruxure Process Expert, User Guide*) for an instance.

Generating and Propagating Parameter Names by Using the *\$FullName* Property

To reduce the use of concatenation functions and the amount of manual configuration, you can use the *\$FullName* property.

You must select the property for the corresponding parameter or variable in the **Select Variables** window, page 261 first.

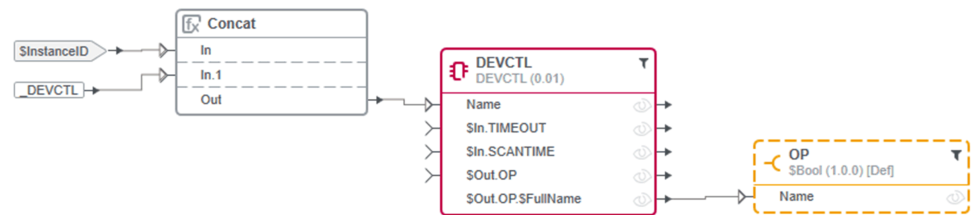
Its output is equal to the concatenation of the following:

- The value of the *Name* property of the Control element.
- A dot separator.
- The name of the parameter (pin name) or variable.

The table describes the format and the output of the *\$FullName* property for various types of Control constituents.

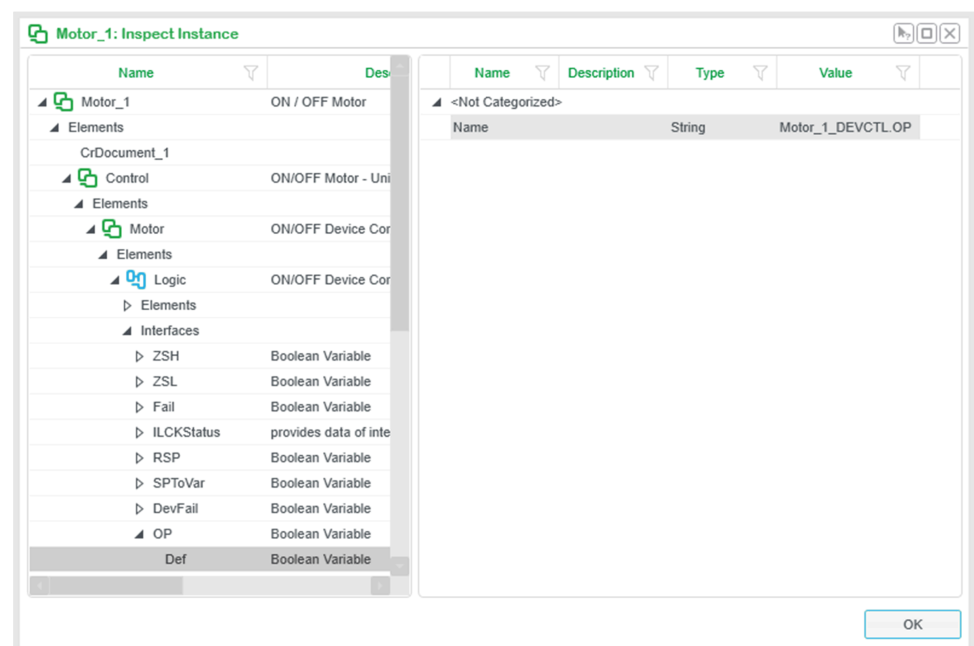
| Control constituent | Format of the output of the <i>\$FullName</i> property | Examples | |
|--------------------------|---|---|---|
| | | Name of the property in the element | Output of the property |
| EFB or DFB | <Value of <i>Name</i> of the element>.<parameter name> | <i>\$Out.OP.\$FullName</i> | <i>Motor_1_DEVCTL.OP</i> Where <i>Motor_1_DEVCTL</i> is the value that is provided to the <i>Name</i> property by using a concatenation function at a higher level of the template composition, page 49. |
| Public variable of a DFB | <Value of <i>Name</i> of the element>.<variable name> Or, <value of <i>Name</i> of the element>.<hierarchical child variable name> | <i>\$Public.Var1.\$Full Name</i> Or, <i>\$Public.Var1.C1.\$Full Name</i> | <i>Motor_1_DEVCTL.Var1</i> Where <i>Var1</i> is the public variable whose parameter name appears as <i>\$Public.Var1</i> for the <i>DEVCTL</i> DFB in the Select Variables window. Or, <i>Motor_1_DEVCTL.Var1.C1</i> where <i>C1</i> is a child variable of <i>Var1</i> . |
| Child variable of a DDT | <Value of <i>Name</i> of the element>.<hierarchical child variable name> | <i>\$STW.\$FullName</i> | <i>Motor_1_HVALVE_ST.STW</i> Where: <ul style="list-style-type: none"> • <i>Motor_1_HVALVE_ST</i> is the value that is provided to the <i>Name</i> property. • <i>HVALVE_ST</i> is the name of the DDT variable element and <i>STW</i> one of its child variables in the Select Variables window. |
| | | <i>Run.CH1.\$FullName</i> | <i>Motor_1_Run.CH1</i> Where: <ul style="list-style-type: none"> • <i>Motor_1_Run</i> is the value that is provided to the <i>Name</i> property. • <i>Run</i> is the name of the DDT variable and <i>CH1</i> one of its child variables in the Select Variables window. |

The following figure shows an example where the $\$FullName$ property of the *OP* output of the *DEVCTL* Control element (DFB) is used to generate the output parameter name *Motor_1_DEVCTL.OP*, which is passed on to the *OP* interface by using a binding. The value that is provided to the *Name* input, page 49 is *Motor_1_DEVCTL* where *Motor_1* is the instance identifier.



NOTE: The interface allows propagating the parameter name within or outside the template.

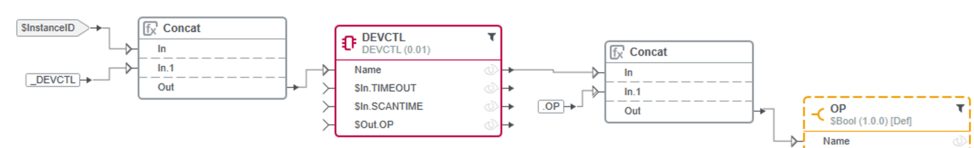
Continuing with the previous example, once instance *Motor_1* has been created, the following figures show how you can use the **Inspect Instance** window (see *EcoStruxure Process Expert, User Guide*) to view the value that is propagated by the *OP* interface.



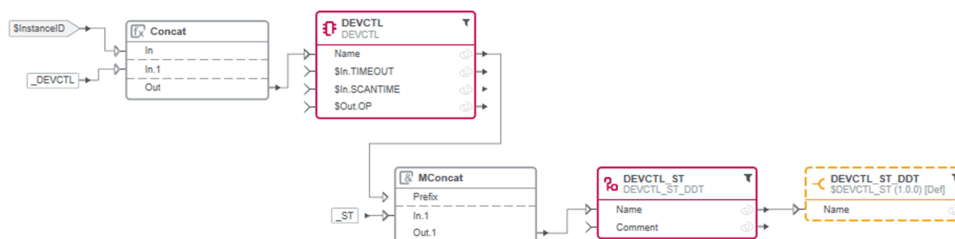
Generating and Propagating Parameter and Variable Names by Using Concatenation Functions

Alternatively, you can use a concatenation function, page 108 to create a parameter or variable name. The *Concat* function concatenates the output pin of the *Name* property of the *DEVCTL* element and a platform input, which is manually configured with the name of the parameter or variable. The output of the function lets you propagate the generated name through a binding.

In this example, if the instance identifier is *Motor_1*, the value that is provided to the *Name* input, page 49 is *Motor_1_DEVCTL* and the name of the output parameter that is generated by the *Concat* function is *Motor_1_DEVCTL.OP*.



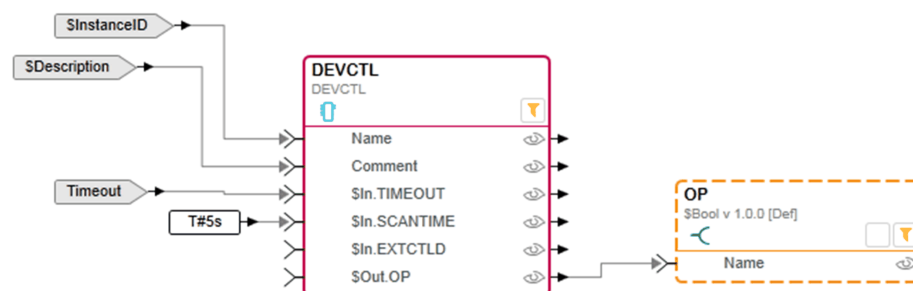
In this example, the concatenation function is used to generate the name of the DDT variable *Motor_1_DEVCTL_ST* (where *Motor_1* is the name of the instance) and propagate it by using an interface. The *DEVCTL* element is not required to generate the name but is only used to propagate the value that is provided to its *Name* input.



NOTE: In both examples above, by using the output pin of the *Name* property, a change that is made on the input pin side of this property is automatically propagated to its output pin.

Incorrect Parameter Name Propagation

The following figure is an example of an incorrect method to propagate the name of the *OP* output parameter of the *DEVCTL* DFB. Because no connection is made with the input pin of the *\$Out.OP* property, no data is available on the output pin of *\$Out.OP* to which the *OP* interface is connected.



Naming Conventions

Overview

This topic explains the aspect of configuring names of constituents that are generated in logical Participant projects, how it impacts template creation, and provides guidelines, page 16.

Global Templates provided by Schneider Electric follow a naming convention.

Applying this convention to your templates helps avoid name conflicts among resources generated by your templates and by Schneider Electric templates.

Naming Convention for Control

Guidelines

For Control templates:

- To generate Control resources that have a unique name in the logical Participant project, concatenate, page 49 the Control resource type with the instance identifier at either of the following levels of the composition:
 - The second composite template (*_UC*) starting from the facet template.
 - The first composite template of the control module template.

Use the *Concat* or *MConcat* binding function.

- The *\$InstanceID* each facet is to be unique (name that appears in the **Facet** column); otherwise the instance **Data** status is set to *Invalid*.

NOTE: For facets generating more than one variable, use an additional level of concatenation, page 51 at the constituent level.

Objective of the Convention

The Schneider Electric naming convention has the following objectives:

- Associate the name of a facet to the instance referencing it and to the constituent that it encapsulates. This makes it easier to identify facets. For example, the facet identifier *Motor_1_DEVCTL* indicates that the facet is referenced by instance *Motor_1* and encapsulates a constituent of type *DEVCTL*.
- Create unique names for resources generated by facets so that no name conflicts are created. This is required because many instances of one template can exist in the application, which may generate facets and resources with the same name. Likewise, many templates can reference the same facet, which also leads to the generation of identical facets and resources.

For example, the *CONDSUM1* DFB, which allows managing interlock conditions is referenced by *\$Motor*, *\$Valve*, and many other templates.

NOTE: If facets do not have a unique name in the application (identifier and path in the composition of the template) the instance **Data** status becomes *Invalid*. Further, in a Control Participant project, generation cannot complete successfully.

Working Principle

The Schneider Electric naming convention is implemented through a mechanism that consists in using the instance identifier (*\$InstanceID* parameter) at each level of the composition by using bindings, page 244. When necessary, at lower levels of the template composition, the parameter is concatenated with the name of the constituent that is encapsulated in the facet element.

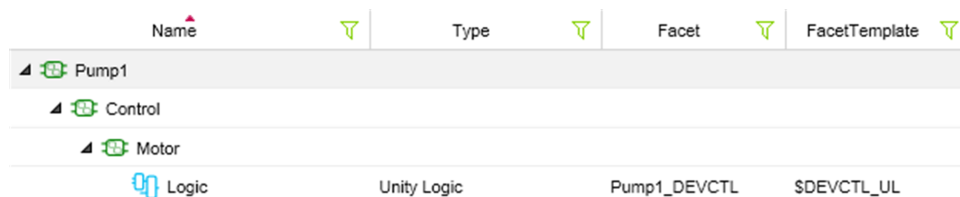
For Control resources the constituent name is usually the type of the DFB or the name of a variable.

From the moment, the *\$InstanceID* parameter is concatenated at a composite or facet template level, the concatenated string becomes part of the *\$InstanceID* for the child elements of the sublevels.

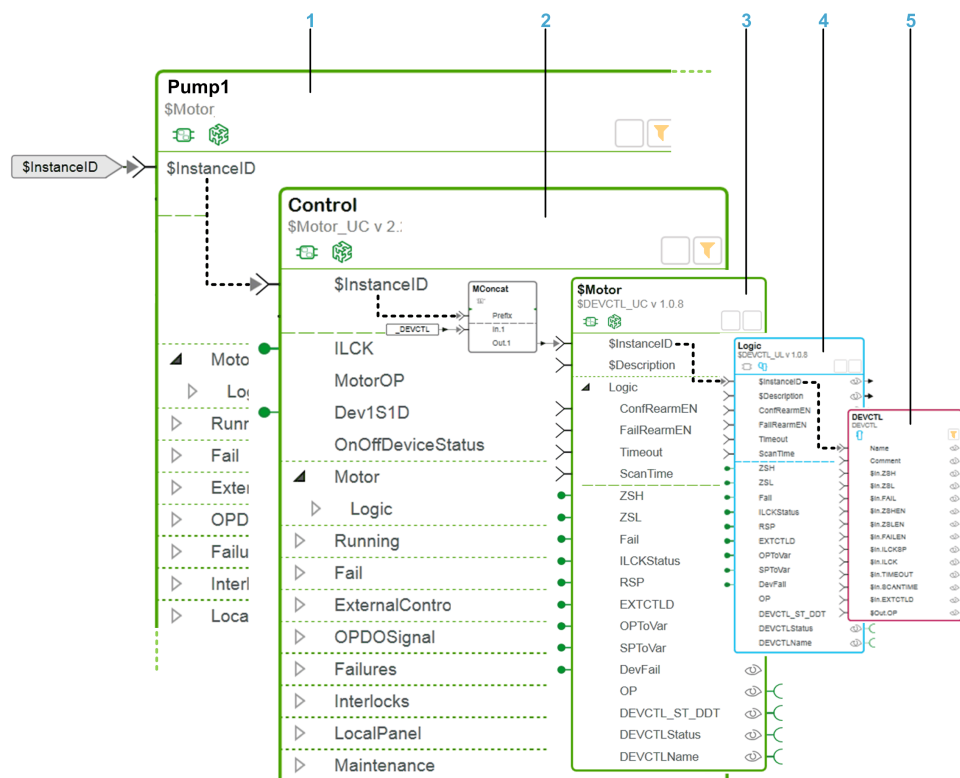
During instantiation, bindings link the resulting identifier to the *\$InstanceID* parameter of the facet templates to create the **Facet** identifier.

During generation, the same identifier is linked to the *Name* parameter of the facet element to create the constituent name. This string can be used thereafter to generate variable names, page 43.

This example shows an instance of Control facet template $\$DEVCTL_UL$ with the name Pump1_DEVCTL, which is the name of the instance concatenated with the name of the DFB.



Using the sample $\$Motor$ control module template as example, the following figure illustrates how the naming convention is applied to Control templates from the control module down to the constituent level to create a facet and a function block with the name Pump1_DEVCTL.



| Item | Description |
|------|---|
| 1 | At the $\$Motor$ control module level, the user has entered Pump1 as instance name during instantiation, which creates instance identifier Pump1. |
| 2 | In the $\$Motor_UC$ composite, the instance identifier is concatenated with platform input $_DEVCTL$ so that, from this level on, the parameter $\$InstanceID$ becomes Pump1_DEVCTL. To represent the service group $\$Motor_UC$ in the Application Explorer , the instance identifier is not used because the composite is an organizational element only. Control is displayed instead. |
| 3 | In the $\$DEVCTL_UC$ composite, $\$InstanceID$ is Pump1_DEVCTL. To represent the service group $\$DEVCTL_UC$ in the Application Explorer , the instance identifier is not used because the composite is an organizational element only. Control is displayed instead. |
| 4 | In the $\$DEVCTL_UL$ facet, $\$InstanceID$ is also Pump1_DEVCTL, which becomes the identifier of the facet. It is displayed in the Facet column of the View Assignments window and the Assignments pane. To represent the service provided by the facet in the Instance Editor , Logic is used. |
| 5 | Finally, at the constituent level the instance identifier Pump1_DEVCTL is linked to the Name parameter, which leads to the generation of a DFB with the name Pump1_DEVCTL. This name is then used to generate logical references, page 43, |

Multiple Occurrences of the Same Resource

When you use the same facet several times in a composite template, concatenating the instance identifier with the type is not sufficient to generate unique facets and constituents.

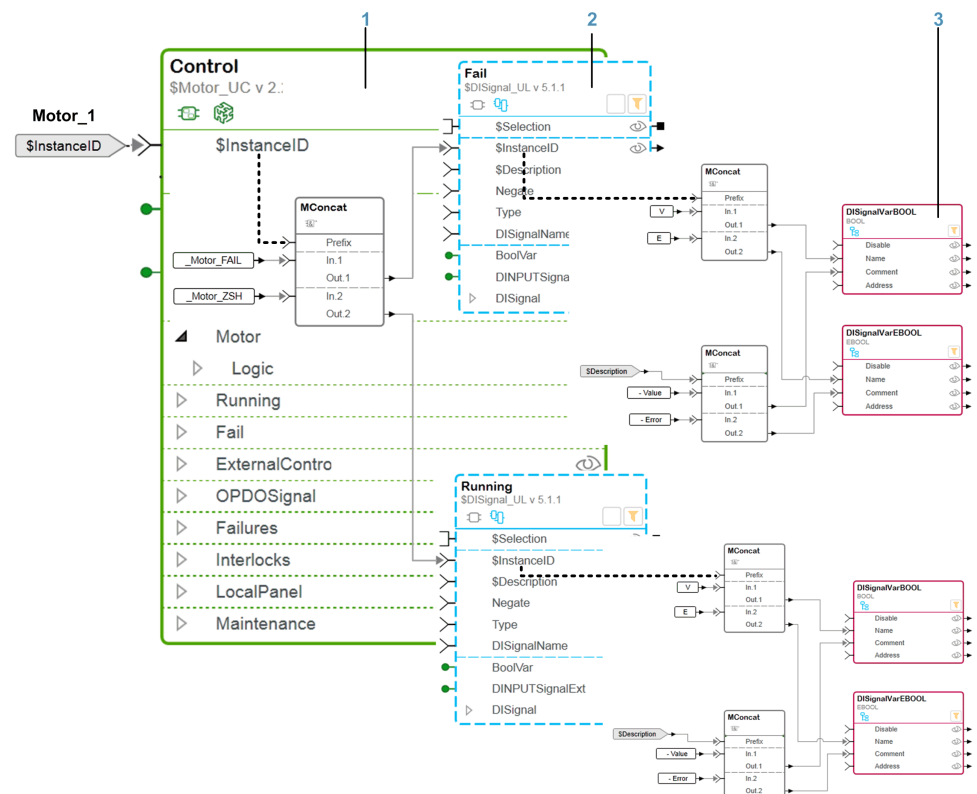
It is often the case for hardware abstraction layer (HAL) templates, for example, the *\$DISignal_UL* facet template.

In such case, the *\$InstanceID* parameter is concatenated with either the type or the template identifier and the name of the corresponding formal parameter (DFB pin).

This example shows two instances of Control facet template *\$DISignal_UL* with names *Motor_1_Motor_ZSH* and *Motor_1_Motor_FAIL*, which are the name of the instance concatenated with the template identifier and the respective names of the formal parameters of the *Motor_1_DEVCTL* DFB.

| Name | Type | Facet | FacetTemplate |
|---------|-------------|--------------------|---------------|
| Motor_1 | | | |
| Control | | | |
| Motor | | | |
| Logic | Unity Logic | Motor_1_DEVCTL | \$DEVCTL_UL |
| Running | Unity Logic | Motor_1_Motor_ZSH | \$DISignal_UL |
| Fail | Unity Logic | Motor_1_Motor_FAIL | \$DISignal_UL |

Using the sample *\$Motor* control module template as example, the following figure illustrates how the naming convention is applied to Control templates from the **Control** composite down to the constituent level to create the variable names in elements *Fail* and *Running*. It shows in particular that a second level of concatenation is required to generate unique variable names.



At the *\$Motor* control module level, the user has entered *Motor_1* as instance name during instantiation, which creates instance identifier *Motor_1*:

| Item | Description |
|------|--|
| 1 | <p>Inside the <i>\$Motor_UC</i> composite (element <i>Control</i>), the instance identifier is concatenated with a specific platform input for each element (<i>Fail</i> and <i>Running</i>) so that, from this level on, the parameter <i>\$InstanceID</i> becomes:</p> <ul style="list-style-type: none"> • <i>Motor_1_Motor_FAIL</i> for element <i>Fail</i> • <i>Motor_1_Motor_ZSH</i> for element <i>Running</i> |
| 2 | <p>Inside element <i>Fail</i>, the <i>\$InstanceID</i> is <i>Motor_1_Motor_FAIL</i> and becomes the identifier of the facet. It is displayed in the Facet column of the View Assignments window and the Assignments pane.</p> <p>The same happens for the other element (<i>Running</i>), where <i>\$InstanceID</i> is <i>Motor_1_Motor_ZSH</i>.</p> |
| 3 | <p>At the constituent level, two constituents are encapsulated in element <i>Fail</i>, which generate one variable each. To create two different variable names, the <i>\$InstanceID</i> is concatenated again before being linked to the <i>Name</i> parameter of each constituent:</p> <ul style="list-style-type: none"> • With platform input <i>V</i> to create variable <i>Motor_1_Motor_FAILV</i>. • With platform input <i>E</i> to create variable <i>Motor_1_Motor_FAILE</i>. <p>The same is applied to constituents of element <i>Running</i> because it uses the same facet template (<i>\$DISignal_UL</i>).</p> <p>NOTE: Because of the variable name length limitation, the variable names are not explicit. To explain the meaning of <i>V</i> and <i>E</i>, a description is provided to the <i>Comment</i> parameter via a concatenation function.</p> |

This example shows that when a facet template generates more than one variable, the facet name does not represent the name of the variables.

Variable Name Length Limitation

In the Control Participant, names of variables are limited to 32 characters. Because variable names are created by concatenating the instance identifier and one or more platform inputs, verify that the resulting name does not exceed the limit.

Section Name Length Limitation

In the Control Participant, names of FBD sections are limited to 32 characters. Because names of sections that the software creates automatically during assignment are created by concatenating the instance identifier and the template identifier, verify that the resulting name does not exceed the limit.

For example, when you assign instance *Motor_1*, which is using template *\$Motor*, the software creates a section named *Motor_1_Motor* (the \$ prefix of Schneider Electric template identifiers is omitted).

Instance Identifiers

The naming convention for Control resources is based on the uniqueness of the instance identifier. This uniqueness is verified by the software during instantiation among instances of the application.

Even if identifiers are identical only because of the hierarchical naming function, the software does not allow it.

NOTE: It is good practice to limit the length of the instance identifier by configuring the **Validation** property of the parameter. This helps staying within the name length limitation of the Control Participant.

For example, if you limit the instance identifier (*\$InstanceID*) to 18 characters and the length of the concatenated platform inputs to create variable names does not exceed 14 characters then, variables names remain within the 32-character limit independently of the instance name entered by the user.

Other Naming Aspects

Naming conventions are also applied to:

- Template identifiers, page 22
- Services, page 25
- Interfaces, page 29
- Variable names and logical references, page 43

NOTE: The software validates certain naming rules. For more information, refer to the topic describing naming rules (see *EcoStruxure Process Expert, User Guide*).

Naming Convention for Supervision

Objective of the Convention

The Schneider Electric naming convention has the following objectives:

- Associate the name of a facet to the instance referencing it.
- Create unique names for facets generated by instances (unique identifier and path in the composition of the template).
- For variable tags:
 - Create a *Tag Name* that corresponds to the property of the genie.
 - Create a tag *Address* that corresponds to the name of the related Control variable that is monitored.
- For advanced alarm tags:
 - Create an *Alarm tag* that correspond to the property of the genie element being animated.
 - Create an *Expressions* that contains the tag name and the expression identifying the bit that is monitored.

The following view of Supervision facets generated by instance *Motor2_2* shows that the identifiers of facets that correspond to the *ForwardFailuresTags* and *ReverseFailuresTags* services have been customized to distinguish them from *Motor2Tags*; the identifiers (*Motor2_2_FC1* and *Motor2_2_FC2* respectively) continue to include the identifier of the instance.

| Name | Type | Facet | FacetTemplate |
|---------------------|----------|--------------|---------------|
| Motor2_2 | | | |
| Supervision | | | |
| Genies | | | |
| Tags | | | |
| ForwardFailuresTags | VJC Data | Motor2_2_FC1 | \$CONDSUM_CD |
| Motor2Tags | VJC Data | Motor2_2 | \$MOTOR2_CD |
| ReverseFailuresTags | VJC Data | Motor2_2_FC2 | \$CONDSUM_CD |

NOTE: A tags facet contains the necessary variable, alarm, trend, and report tags, which are grouped in categories, page 152.

Genie Properties

For genies of the General Purpose Library (Classic), the property of a Genie is composed of the substitution and a fixed string.

During generation, the substitution is replaced with the instance identifier, creating a unique name for the genie.

For example, *%NAME%_FC_CONDSUM_BYPASSW*, where *%NAME%* is the substitution for the instance identifier.

Working Principle

The Schneider Electric naming convention is implemented through a mechanism that consists in using the instance identifier (*\$InstanceID* parameter) at each level of the composition by using bindings, page 244. When necessary, at lower levels of the template composition, the parameter is concatenated with character strings to generate a name that corresponds to the property of the Supervision component.

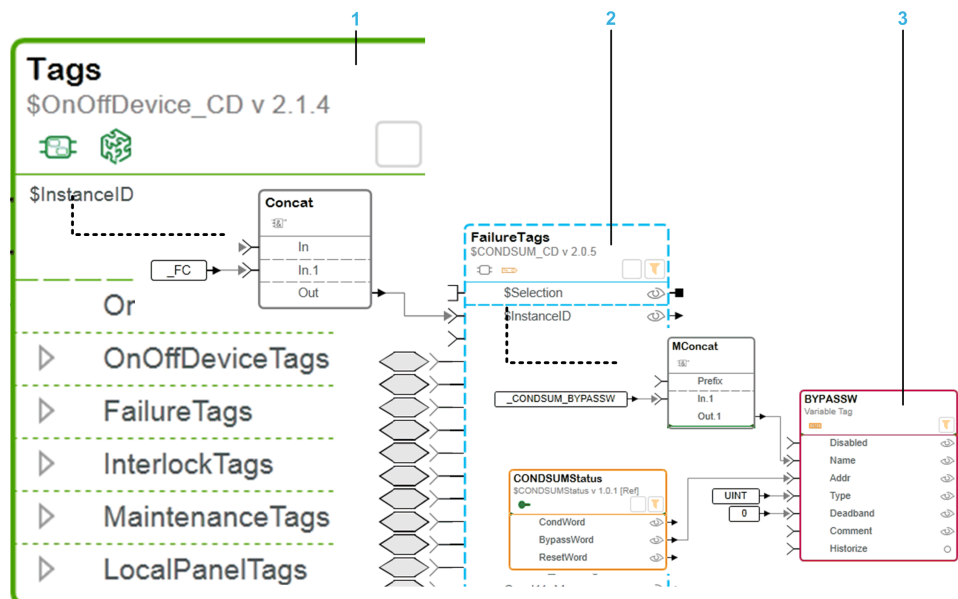
During instantiation, bindings link the resulting identifier to the *\$InstanceID* parameter of the facet templates to create the **Facet** identifier.

During generation, the same identifier is linked to the *Name* parameter of the facet element to create the constituent name (tags and genies).

NOTE: For variable tag addresses, the data is propagated to the Supervision element by an interface, which is connected to the Control element representing the variable.

Variable Tags

Using the *\$Motor* control module as example, the following figure shows how the naming convention is applied to Supervision templates. It starts from the *Supervision* composite template down to the element generating the variable tag, which allows bypassing abnormal conditions. The *Tag Name* of this variable tag that is managed by the corresponding Supervision component is *CM name_FC_CONDSUM_BYPASSW* where *CM name* is the identifier of the control module instance.

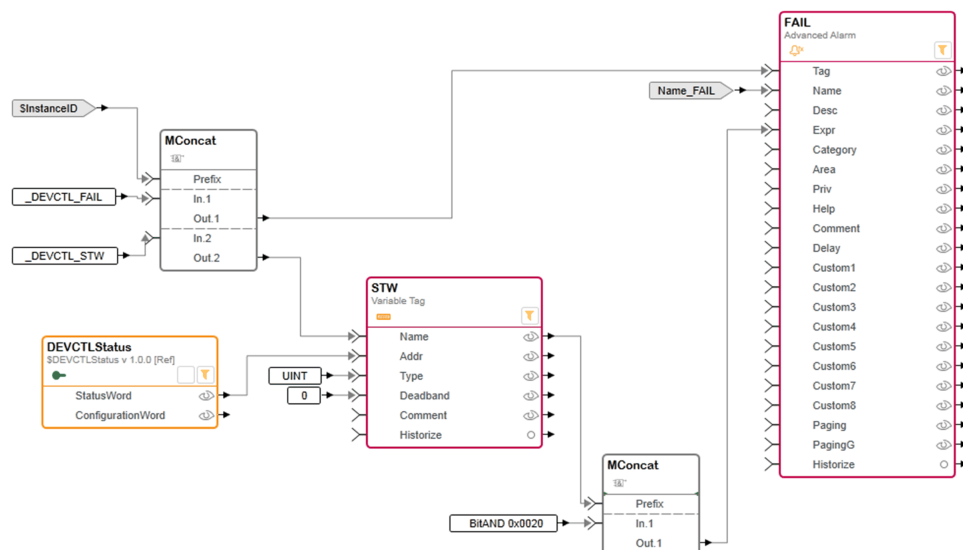


| Item | Description |
|------|--|
| 1 | Inside the <i>\$Motor_CS</i> composite template (<i>Supervision</i>), the instance identifier is passed on to the <i>\$OnOffDevice_CD</i> composite template by connecting the <i>\$InstanceID</i> parameter directly. |
| 2 | Inside <i>\$OnOffDevice_CD</i> , before passing on the instance identifier to the <i>\$CONDSUM_CD</i> facet template, it is concatenated with platform input <i>_FC</i> so that, from this level on, the parameter <i>InstanceID</i> becomes, for example, <i>Motor_1_FC</i> and is applied to child elements. It is displayed in the Facet column of the View Assignments window and the Assignments pane. To represent the service provided by <i>\$CONDSUM_CD</i> in the Application Explorer , the instance identifier is not used but <i>FailureTags</i> . |
| 3 | Finally, inside <i>\$CONDSUM_CD</i> , the instance identifier is concatenated with platform input <i>_CONDSUM_BYPASSW</i> , which leads to the generation of a variable tag name, which is, for example, <i>Motor_1_FC_CONDSUM_BYPASSW</i> where <i>Motor_1</i> is the control module instance identifier. It corresponds to the property of the genie. |

NOTE: The variable tag *Address* is provided by a Control facet template through an interface, page 160.

Advanced Alarm Tags

Using the *\$Motor* control module as example, the following figure shows how the naming convention is applied to elements encapsulated in the *\$DEVCTL_CD* facet template, which generates variable and advanced alarm tags. The name of the advanced alarm tag (*Alarm Tag*) that is managed by the corresponding Supervision component is *CM name_DEVCTL_FAIL*. It is used to indicate an inoperable device.

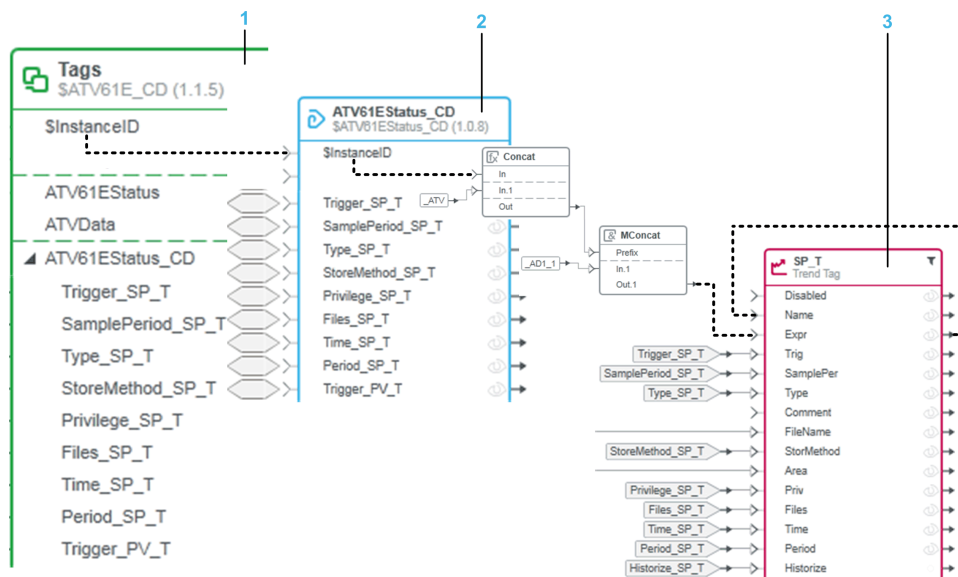


NOTE: Platform inputs of the type *BitAND* that are used to configure advanced alarm tag expressions need to start with a space to respect the syntax of the expression that is configured in the Supervision component. for example, *CM name_DEVCTL_STW BitAND 0x0020*.

NOTE: The name of the advanced alarm that appears in the **Instance Editor** (**Name_FAIL**) is created by using a parameter from the **Toolbox**. At the same time, it allows entering a name (*Alarm Name*) for the alarm during instantiation. This name appears on the Supervision faceplate during operation.

Trend Tags

Using the *\$ATV61E* variable speed drive control module as example, the following figure shows how the naming convention is applied to Supervision templates. It starts from the *Supervision* composite template down to the element generating the trend tag for the setpoint (SP). The *Tag Name* and the *Expression* of this trend tag that are managed by the corresponding Supervision component is *CM name_ATV_AD1_1* where *CM name* is the identifier of the control module instance.



| Item | Description |
|------|--|
| 1 | Inside the <i>\$ATV61E_CS</i> composite template (<i>Supervision</i>), the instance identifier is passed on to the <i>\$ATV61E_CD</i> composite template by connecting the <i>\$InstanceID</i> parameter directly. |
| 2 | Inside <i>\$ATV61E_CD</i> , the instance identifier is again passed on to the <i>\$ATV61EStatus_CD</i> facet template by connecting the <i>\$InstanceID</i> parameter directly. |
| 3 | <p>Finally, inside <i>\$ATV61EStatus_CD</i>, before passing on the instance identifier to the <i>SP_T</i> trend tag element, it is concatenated first with platform input <i>_ATV</i> then with platform input <i>_AD1_1</i>. As a result, the expression that is generated is, for example, <i>ATV61E_1_ATV_AD1_1</i> where <i>ATV61E</i> is the control module instance identifier.</p> <p>Because the name of the trend tag is the same as the expression, the out pin of Expr is linked with the in pin of Name.</p> |

NOTE: The names of the various properties of the trend tag that appear in the **Instance Editor** are created by using parameters from the **Toolbox**. At the same time, these allow entering a value for each property during instantiation.

Instance Identifiers

The naming convention for Supervision resources is based on the uniqueness of the instance identifier. This uniqueness is verified by the software during instantiation among instances of a given template.

The software also verifies the uniqueness of facet identifiers (identifier and path) inside the application.

Other Naming Aspects

Naming conventions are also applied to:

- Template identifiers, page 22
- Services, page 25

- Interfaces, page 29

NOTE: The software validates certain naming rules. For more information, refer to the topic describing naming rules (see *EcoStruxure Process Expert, User Guide*).

Storage Considerations

What's in This Chapter

| | |
|--|----|
| Global Templates Library Structure | 58 |
| Content Repository | 59 |
| Process Expert Database | 61 |

Overview

This chapter describes storage aspects that you need to consider when creating templates.

Global Templates Library Structure

Overview

This topic explains the aspect of storing your templates in a folder structure of your own, describes the typical folder structure of the General Purpose library (Classic), and provides guidelines.

Template libraries use a folder structure that allows storing control modules and other templates by type and category.

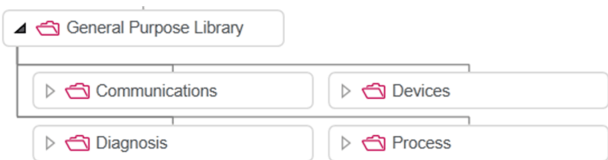
For more information on templates, refer to the help of Schneider Electric template libraries.

Guidelines

- Create your own folder structure within the Global Templates library that reproduces the organization of the General Purpose library.
- Store, [page 284](#) your templates in the corresponding folders to locate the various types of templates that you have created.

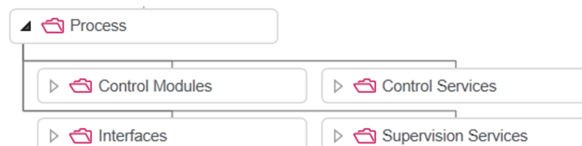
Folders of the General Purpose Library

Templates of the General Purpose are grouped by category of application templates.



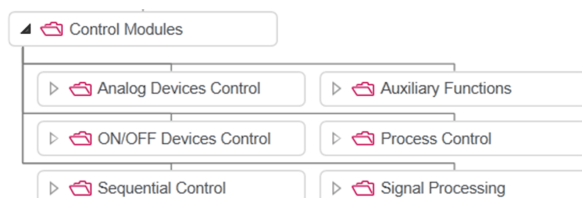
Folders for Template Types

Each category contains folders for control module templates. It also contains at the same level, folders for composite and facet templates grouped by Participant, and folders for interface models.



Folders for Control Module Templates

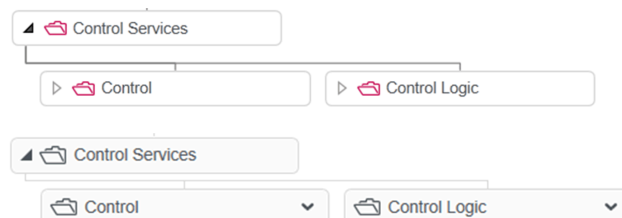
Control module templates are grouped by family.



Folders for Control Composite and Facet Templates

Inside the **Control Services** folder:

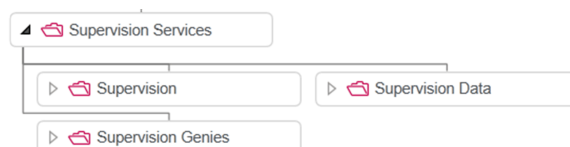
- Composite templates are stored in the **Control** folder.
- Facet templates are stored in the **Control Logic** folder.



Folders for SupervisionComposite and Facet Templates

Inside the **Supervision Services** folder:

- Composite templates are stored in the **Supervision** folder.
- Data facet templates (for variable, advanced alarm, trend, and report tags) are stored in the **Supervision Data** folder.
- Genie facet templates are stored in the **Supervision Genies** folder.



Content Repository

Overview

This topic explains aspects related to the storage of resources in the content repository (see *EcoStruxure Process Expert, User Guide*), how they impact template creation, and provides guidelines.

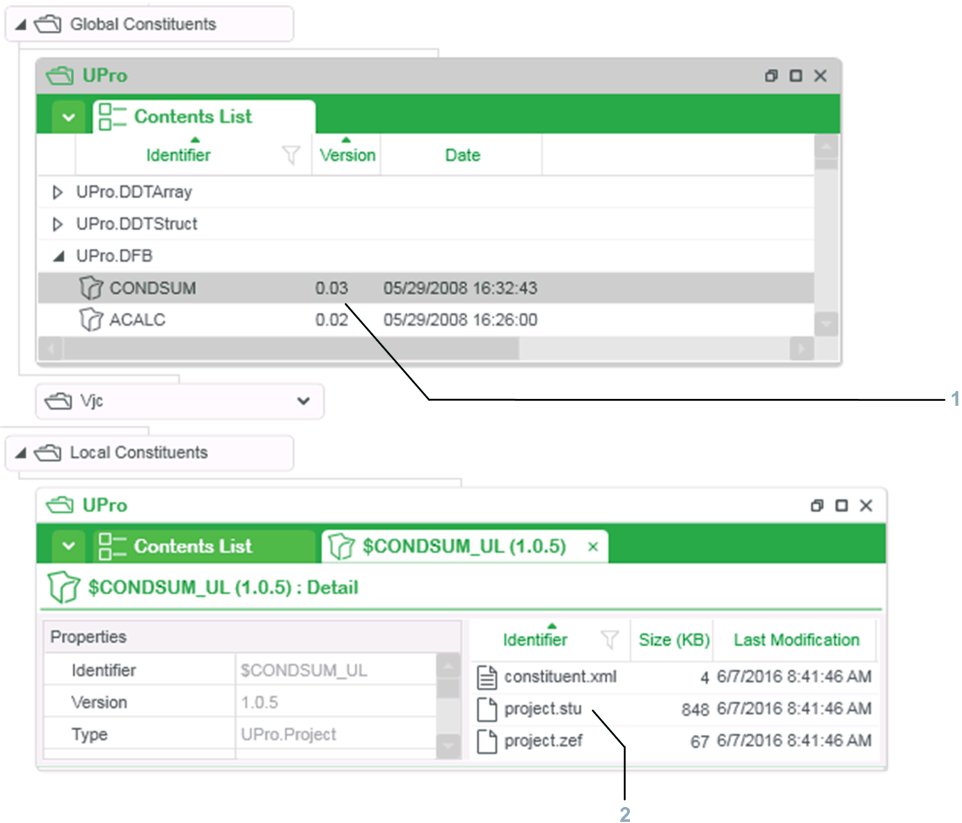
The content repository contains data that is used by templates and that pertains to existing systems.

When you create facet templates and encapsulate constituents, page 137, the corresponding project files are stored in the content repository.

Guidelines

Use the content repository to view global and local constituents. You can save constituents to disk and use them as a starting point to create your own. You can also verify the version of constituents.

The following example shows the entries in the content repository under the **Global Templates** section for Control facet template `$CONDSUM_UL [1.0.5]`:



| Item | Description |
|------|--|
| 1 | The Global Constituents folder allows you to view the version of encapsulated Control resources such as variable structure and DFB types. |
| 2 | The Local Constituents folder gives you access to a copy of the project file (.stu) that was used to encapsulate the CONDSUM DFB. It contains changes that were made during constituent encapsulation and saved. The other project file (.zef) is a full application exchange file that is not Control Participant version dependent. |

Control Resources

FFBs that are encapsulated in facets of the Global Templates library (and therefore stored in the content repository) and those that are stored in the library of the Control Participant need to be identical.

You need to update the Control Participant if the following statements are true:

- You have encapsulated a new version of an FFB in a Control facet.
- You have updated templates that reference this facet.
- Your system contains instances that use the template that you have updated.

- You have updated the template that is used by these instances.
- The corresponding Control facets of these instances are assigned to projects and are generated.

Supervision Resources

If you encapsulate genies in Supervision facets by adding to the content repository, page 272 an Include project that has the same name as an existing one but of a different version, the existing Include project is replaced by this version.

NOTICE

LOSS OF DATA

Resolve a project file conflict that arises when adding a project file to the content repository by selecting the correct file version.

Failure to follow these instructions can result in unusable Supervision projects.

NOTE: Use or create your own Include project rather than creating a new version of the Include project of a Schneider Electric library.

Process Expert Database

Overview

The database of the software contains the templates of the Global Templates library. It is hosted on the computer on which the system server is installed.

Templates that you create and constituents that are stored in the content repository are added to the database.

Guidelines

Back up the database before you start creating templates.

For more information, refer to *Starting the Software* (see *EcoStruxure Process Expert, Installation and Configuration Guide*).

Global Templates Definition

What's in This Chapter

| | |
|-------------------------------------|-----|
| Interface Model Definition | 62 |
| Facet Template Definition..... | 73 |
| Composite Template Definition | 75 |
| Common Definitions | 80 |
| Elements | 92 |
| Binding Functions..... | 108 |

Overview

The definition of a template consists of the editable and non-editable attributes that define how a template is built, behaves, and can be used.

This chapter describes the definition that is specific to the following types of Global Templates:

- Interface models
- Facet templates
- Composite templates

Definitions that these templates have in common are described in a separate section, page 80.

This chapter also describes the following binding functions:

- Transformation functions that you can use with interface models.
- Functions that you can use with fact and composite templates.

Interface Model Definition

Overview

This section describes the definition of interface models, which are a key mechanism of Global Templates by defining:

- Data exchange between instances and between references (for example, define which information is exchanged between an instance of *\$Motor* and *\$DISignal_UL* that are linked by using the **Asset Workspace Editor** or define how variable names are transferred from Control facets to Supervision facets to create tag names).
- Dependencies between instances and between references (for example, define which type of data is required when you instantiate the *\$TesyTMB* motor controller template).

Components of the Interface Model Definition

Overview

The definition of interface models is divided into several categories. The data of each category can be viewed and/or edited in a pane of the same name. You can access these panes by using the **Interface Editor**, page 220.

The contents of each pane is described in the following topics.

Interface Model Definition Categories

The definition of interface models is divided into the following categories. The pane of the same name allows you to view and/or configure data.

| Category/Pane | Description |
|-----------------|--|
| Header | Contains general information about the interface model including the definition of its roles. |
| Elements | Defines the elements of each role of the interface model. These elements represent the data that is shared by the interface model. There is an Elements pane for each role. |
| Rules | Defines the rules for the elements of the interface model. NOTE: The Rules pane becomes available once you have defined an element in the Elements pane. There is an Rules pane for elements of each role. |

NOTE: The **Toolbox** pane does not contain interface model definition information but transformation functions that you can use with bindings. For more information, refer to the topic describing [transformation functions](#), page 68.

Interface Model Header

Header Pane Description

The following figure shows an example of the **Header** pane of an interface model.

| Header | |
|-----------------------------------|-------------------------------------|
| Interface_1 (1.0.0) : Application | |
| General | |
| Identifier | Interface_1 |
| Version | 1.0.0 |
| Type | Application |
| Description | |
| Valid | <input checked="" type="checkbox"/> |
| Protection | Unprotected |
| State | Approved |
| Standard | <input type="checkbox"/> |
| Signature | 361bfb8d-a3b7-43a5-85df- |
| Execution Order | None |
| Role A | |
| Identifier | A |
| Cardinality | 2 |
| Role B | |
| Identifier | B |
| Cardinality | 1 |

| Property | Description |
|------------------------|---|
| – | Identifier, version, and type of the interface model. |
| General section | |
| Identifier | Refer to the description of header common definitions, page 80. |

| Property | Description |
|--|--|
| Version | Version of the interface model, allowing you to use different versions of the interface model that share the same Identifier . For more information, refer to Header Common Definition , page 80. |
| Type | Defines the type of the interface model and as a result, enables or restricts connections between types of templates and instances, page 21. For example, an interface that you want to use in the Hardware Mapping Editor , needs to be of type <i>Mapping</i> . Possible values: <ul style="list-style-type: none"> • <i>Application</i> (default) • <i>Physical</i> • <i>Communication</i> • <i>Mapping</i> |
| Description | You can enter a description by using free form text. Default value: Blank. |
| Valid | Refer to Header Common Definition , page 80. |
| Protection | |
| State | |
| Standard | |
| Signature | |
| Execution order | Defines the execution order of the instances connected through the interface: <ul style="list-style-type: none"> • AtoB: The instance with role <i>A</i> is executed before the instance with role <i>B</i> (default). • BtoA: The instance with role <i>B</i> is executed before the instance with role <i>A</i>. • None: The execution order is not applicable. |
| Role A (left) and Role B (right) sections, page 64 | |

Types of Interface Models

The following types of interface models are predefined:

- **Physical**: Allow data exchange at the physical level (modules) between:
 - Topological references
 - Topological instances
- **Communication**: Allow data exchange at the logical level (I/O scanner, OPC Factory Server software, Supervision I/O devices) between:
 - Topological references
 - Topological instances
- **Application**: Allow data exchange between:
 - Application references/instances
 - Topological references/instances
- **Mapping**: Allow data exchange during the mapping stage between application objects (representing the logical projection of the hardware) and topological objects (representing the actual hardware defined in the topology).

Interface Roles

Each interface model has two **roles**, page 27, which represent both ends of the interface link. Elements and rules are associated to a role. You can configure roles by using the **Role A** and **Role B** sections of the **Header Pane**.

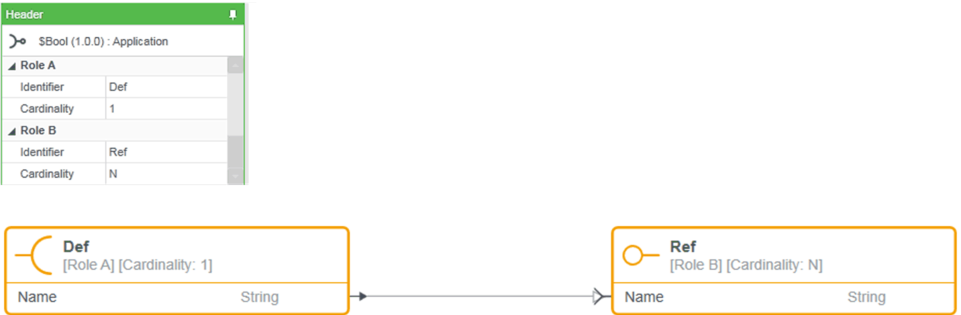
The following figure shows an example of the **Role A** and **Role B** sections.

| | |
|-------------|---|
| ▲ Role A | |
| Identifier | A |
| Cardinality | 2 |
| ▲ Role B | |
| Identifier | B |
| Cardinality | 1 |

| Property | Description |
|-------------|---|
| Identifier | <p>The role identifier is used to identify each role. It needs to be unique for each role of the same interface model.</p> <p>Default value: <i>A</i> and <i>B</i></p> <p>The convention is that role <i>A</i> is associated to the end providing data and role <i>B</i> to the one receiving data.</p> |
| Cardinality | <p>The cardinality indicates to how many objects the role can connect. It can have the following values:</p> <ul style="list-style-type: none"><number>: Number of connections (minimum and default value is 1).N: Infinite number of connections. <p>NOTE: Cardinality needs to be different from 1 to be able to use a transformation function with a <i>Producer</i> element. Changing the cardinality to 1 while an element is connected to a transformation function removes the link.</p> |

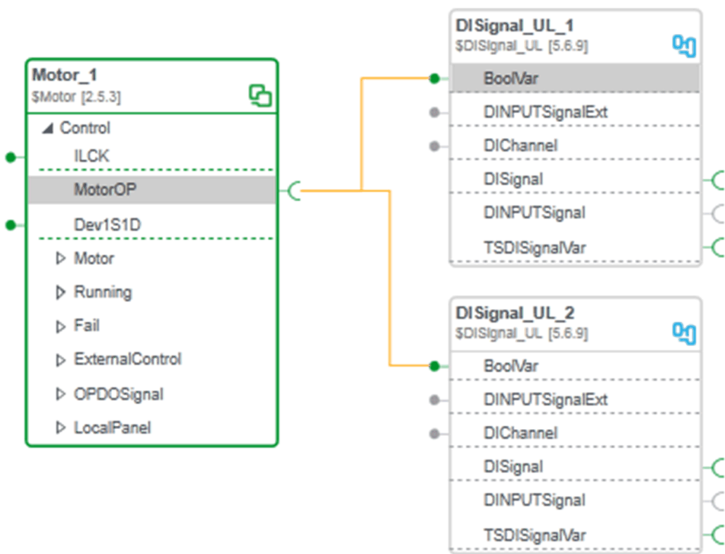
Example of Cardinality

The following figure shows an example of use of the *\$Bool* interface, which has a 1 to *N* cardinality.



Role *Def* (cardinality 1) is used by *\$Motor* and the interface can connect *N* instances by using role *Ref* (cardinality *N*), for example, *\$DISignal_UL*.

In this example, it is not possible to connect the *BoolVar* interface of *\$DISignal_UL_1* or *\$DISignal_UL_2* to another instance because role *Def* (cardinality 1) of the interface can connect only to 1 object.



Interface Model Elements

Elements Pane Description

Each interface model can contain one or more elements that are defined by using the **Elements** pane.

There is an **Elements** pane for each role of the interface model.

The title bar of the **Elements** pane indicates:

- Element identifier: element type* when you click an existing element of a role. You can edit the parameters of an existing element.
- <New Element>**: when no element is selected for a role. Click the white background of the workspace to clear the element selection.

The following figure shows an example of the **Elements** pane of an interface model.

Elements

➔

<New Element> :

General

| | |
|-------------|----------|
| Identifier | |
| Type | String |
| Direction | Producer |
| Mode | Direct |
| Description | |

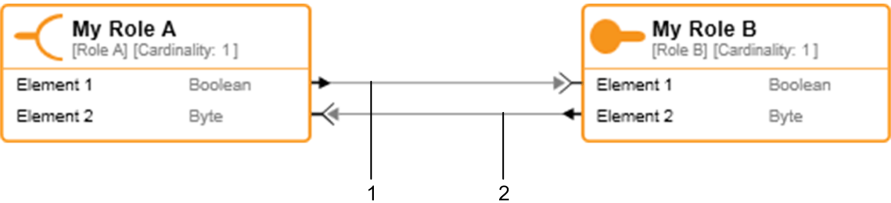
+

| Property | Description |
|-----------------|---|
| General section | |
| Identifier | Element identifier, which is: <ul style="list-style-type: none">Unique for the interface model if the mode for the element is Transform. |

| Property | Description |
|--------------------|--|
| | <ul style="list-style-type: none"> The same in both roles if the mode for the element is Direct. <p>Default value: Blank</p> |
| Type | <p>Defines the type of the data that is shared by the element. For example, to share the name of a variable or an IP address use <i>String</i>, to share a value without decimals use <i>Integer</i>.</p> <p>Default value: <i>String</i></p> <p>For more information, refer to the topic describing supported data types (see <i>EcoStruxure Process Expert, User Guide</i>).</p> |
| Direction | <p>Defines the source and the destination of the data shared by the element.</p> <p>Possible values:</p> <ul style="list-style-type: none"> Consumer: The element receives the value from the other role. Producer: The element produces the value to be transmitted to the other role (default). <p>NOTE: Changing a direction automatically changes the direction of the counterpart element that is connected by a direct link. For elements connected to a transformation function, changing the direction removes the link.</p> |
| Mode | <p>Defines whether data is exchanged directly or by using a transformation function.</p> <p>Possible values:</p> <ul style="list-style-type: none"> Direct: When you create the element, it is connected to an element with the same identifier and opposite direction in the other role of the interface (default). Transform: When you create the element, it is not connected to an element of the other role of the interface nor is a counterpart element created automatically. You need to use a transformation function, page 68 from the Toolbox pane and connect the element, the transformation function, and another element in the other role manually. Using a transformation function requires setting cardinality, page 64 of the <i>Producer</i> element to a value greater than 1. <p>NOTE: Changing the mode:</p> <ul style="list-style-type: none"> From Direct to Transform: Removes the existing link. From Transform to Direct: Removes the existing link with the transformation function and automatically creates an element with the same identifier and opposite direction in the other role of the interface and connects it. The other element that is connected to the transformation function and the function itself remain. |
| Description | <p>Enter a description by using free form text.</p> <p>Default value: Blank</p> |
| + | <p>Click to create a new element by using the information of the fields of the Elements pane.</p> <p>NOTE: The button is available only when <New Element>: appears at the top of the pane.</p> |
| * | <p>Click to reset the fields of the Elements pane to their default value and show <New Element>: at the top of the pane so that you can configure and create a new element.</p> <p>NOTE: The button is available only when an element is selected.</p> |

Element Direction Example

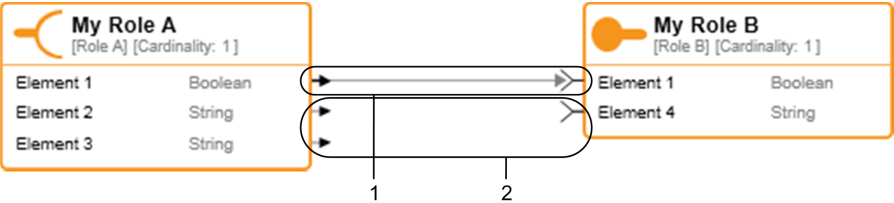
The following figure illustrates how you can define the direction for elements of an interface.



| Item | Description |
|------|---|
| 1 | Element 1 is set as producer for My Role A and as a consumer for My Role B |
| 2 | Element 2 is set as producer for My Role B and as a consumer for My Role A |

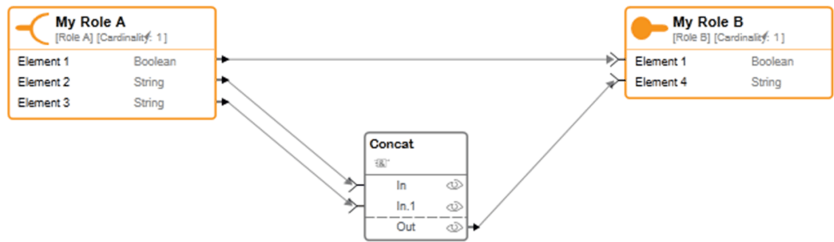
Element Mode Example

The following example illustrates how the element connection behaves depending on the element mode that you select.



| Item | Description |
|------|---|
| 1 | The mode of Element 1 is set to <i>Direct</i> : An element with the same identifier and opposite direction is automatically created in the other role of the interface and both elements are connected. |
| 2 | The mode of Element 2 , Element 3 , and Element 4 is set to <i>Transform</i> : The connection is not established by the Interface Editor nor is Element 4 created automatically. |

For **Element 2**, **Element 3** and **Element 4**, you need to establish the connection manually after adding a transformation function. In this example the *Concat* function is used to connect these elements.



Transformation Functions

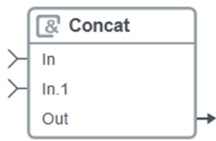
Overview

This topic describes the transformation functions that are available in the **Toolbox** pane of the **Interface Editor**.

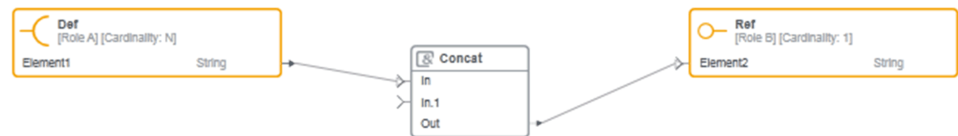
With these functions you can modify data produced by elements of an interface model with 1-to-N cardinality. To use a transformation function, drag it to the workspace.

The inputs and outputs of transformation functions can only be linked to elements of the interface model, not to other transformation functions.

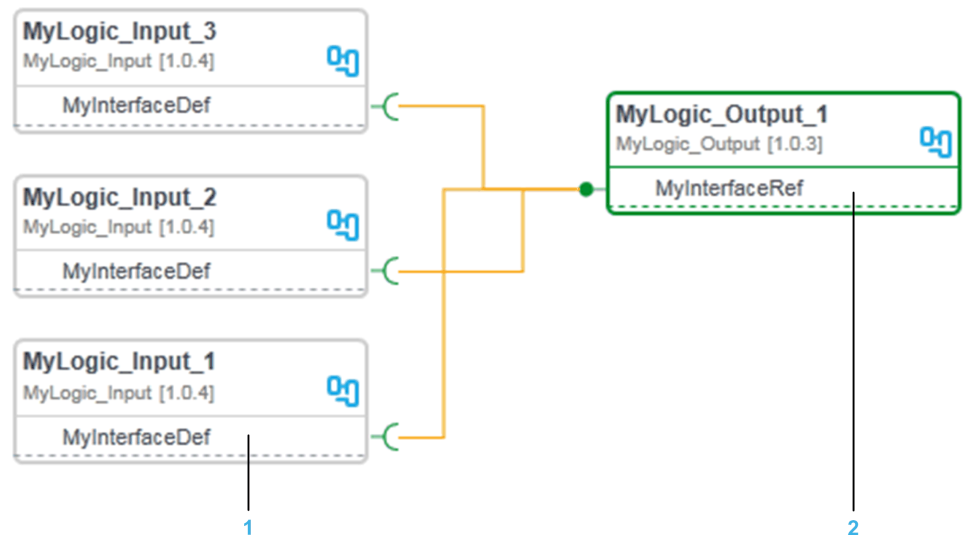
Concatenation of Strings: *Concat*

| Representation | Description | |
|---|---|-----------|
|  | <p>This function concatenates strings received through the <i>In</i> input.</p> <p>When the role of the interface containing the element that is linked to the input of the function is connected several times, each connection providing a value, the various values received by the input are concatenated in the order of connection and assigned to the <i>Out</i> output.</p> <p>Null values are ignored. The output is null if inputs are null.</p> <p>NOTE: The <i>In.1</i> input is not used.</p> | |
| | Input/Output | Data type |
| | <i>In</i> | String |
| | <i>Out</i> | |

The following example shows the *MyInterface* interface with *Element1* in role *Def* (provider) and *Element2* in role *Ref* (consumer). It contains the *Concat* transformation function linking both elements.




Consider the scenario in which three instances are linked to one instance. The four instances reference *MyInterface*.




| Item | Description |
|------|---|
| 1 | <p>Instances referencing the <i>Def</i> role of <i>MyInterface</i>, which provide the following strings (<i>Element1</i>):</p> <ul style="list-style-type: none"> <i>MyLogic_Input_1</i>: String1 <i>MyLogic_Input_2</i>: blank (empty string) <i>MyLogic_Input_3</i>: String3 |
| 2 | <p>Instance referencing the <i>Ref</i> role of <i>MyInterface</i>, which receives the strings provided by the three instances. The instances have been connected in the following order:</p> <ul style="list-style-type: none"> <i>MyLogic_Input_3</i> <i>MyLogic_Input_2</i> <i>MyLogic_Input_1</i> |

The string that is output by the concatenation function to *Element2* is *String3String1*.

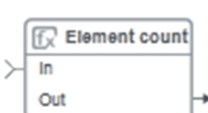
List

| Representation | Description | |
|---|---|-----------|
|  | <p>This function outputs a comma separated list composed of the values received by the List input, for example, when the role of the interface containing the element linked to List is connected several times, each connection providing a value.</p> <p>Null values (empty strings) are ignored.</p> | |
| | Input/Output | Data type |
| | <i>List</i> | String |
| | <i>Output</i> | |

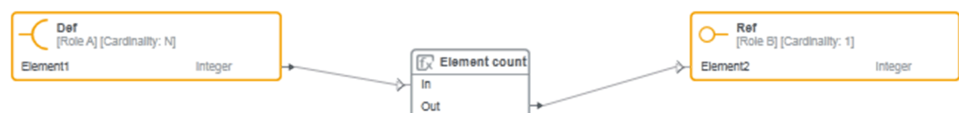
Sum

| Representation | Description | |
|---|--|-----------|
|  | <p>This function adds the numerical values it receives, for example, when the role of the interface containing the element linked to Items is connected several times, each connection providing a value.</p> | |
| | Input/Output | Data type |
| | <i>Items</i> | Integer |
| | <i>Result</i> | |

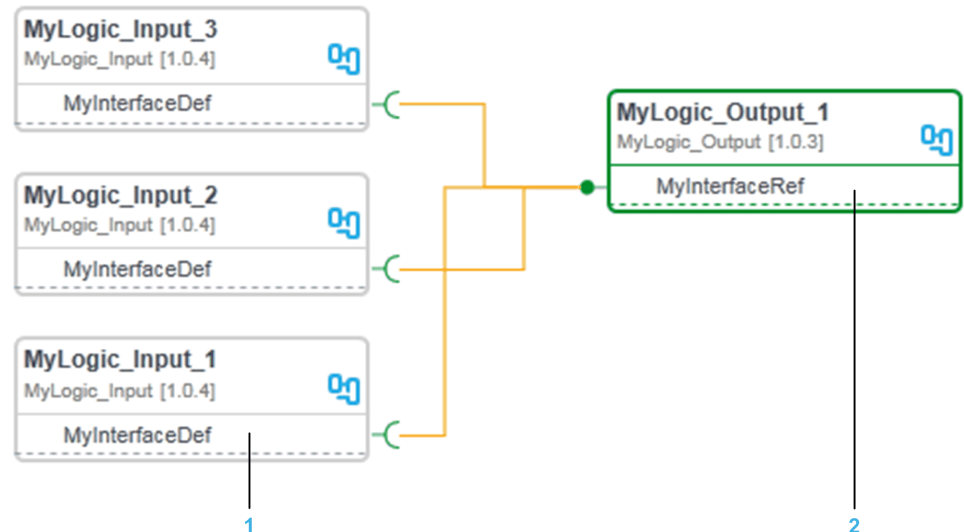
Counting of Element Connections: *Element count*

| Representation | Description | |
|---|--|-----------|
|  | <p>This function returns on the <i>Out</i> output the number of instances and references that are connected and provide a value through an element.</p> <p>Connect the <i>In</i> input to an element in the producer role of the interface with cardinality <i>n</i>.</p> <p>Connect the <i>Out</i> output to an element in the consumer role of the interface with cardinality 1.</p> | |
| | Input/Output | Data type |
| | <i>In</i> | Integer |
| | <i>Out</i> | |

The following example shows the *MyInterface* interface with *Element1* in role *Def* (provider) and *Element2* in role *Ref* (consumer). It contains the *Element count* transformation function linking both elements.



Consider the scenario in which three instances are linked to one instance. The four instances reference *MyInterface*.



| Item | Description |
|------|--|
| 1 | Instances referencing the <i>Def</i> role of <i>MyInterface</i> , which provide the <i>Element1</i> value. |
| 2 | Instance referencing the <i>Ref</i> role of <i>MyInterface</i> , which receives the value returned by the Element count transformation function (<i>Element2</i>). |

If the three connected instances provide a value (*Element1* is not null) then the result of the count (*Element2*) is 3.

If, for example, *MyLogic_Input_3* provides no value for *Element1* while the others do, the result of the count (*Element2*) is 2.

Maximum Value: *Max*

| Representation | Description | |
|----------------|---|-----------|
| | This function outputs the maximum value out of the values received, for example, when the role of the interface containing the element linked to In is connected several times, each connection providing a value. | |
| | Input/Output | Data type |
| | <i>In</i> | Integer |
| | <i>Max</i> | |

Minimum Value: *Min*

| Representation | Description | |
|----------------|---|-----------|
| | This function outputs the minimum value out of the values received, for example, when the role of the interface containing the element linked to In is connected several times, each connection providing a value. | |
| | Input/Output | Data type |
| | <i>In</i> | Integer |
| | <i>Min</i> | |

Interface Model Element Rules

Purpose of Interface Element Rules

Rules defined for elements of an interface model allow you to manage the validity of an interface connection at the interface element level. When the `interface` rule, page 88 that is defined for the interface requires that it be connected, if one of the defined interface element rules is not satisfied when you connect the interface, the **Link** status of the instance referencing the facet becomes invalid.

You can define rules only for *Producer* elements, page 66.

The **Rules** pane contains the rules table, which allows you to create and configure rules when an interface contains several elements. An element appears in the pane of the role in which it is defined as *Producer*.

User-defined rules are applied by using the OR operator (for example, **Rule 1 OR Rule 2 OR Rule 3**).

Rules Pane Description

The following figure shows an example of the element **Rules** pane in which three rules have been defined in the rules table.

| Rules | | | | | |
|--------|------|-----------|----------|---------|----------|
| | Name | HighValue | LowValue | EUValue | Element5 |
| Always | ✓ | | | | ✓ |
| Rule_1 | ✓ | ✓ | ✓ | ✗ | ✓ |
| Rule_2 | ✓ | ✗ | ✗ | ✓ | ✓ |
| Rule_3 | ✓ | | | | ✓ |
| + ✗ | | | | | |

The information shown in the rules table is organized as follows:

- Columns: Each row corresponds to an element that is defined as *Producer* in the corresponding role of the interface.
- Rows: Each row corresponds to a rule that you can define for each of the listed elements.
 - **Always** (required): Rule that is created and selected by default for each *Producer* element. As a result, by default, existing rules and those you add are selected by default and not-editable.
 - **Rule n**: User-defined rule, where *n* is an incremental number starting at 1.
- Cells: Indicate the connection requirement for each rule of an element.
- **+/✗**: Allows adding/deleting a user-defined rule.

Description of Element Connection Requirements

The following table describes the various connection requirements that you can define for an element by using rules.

| Icon | Description |
|--|---|
| <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> | Required: If the interface rule (which applies to the interface as a whole) requires that the interface be connected, the element has to be connected to satisfy this rule. NOTE: <ul style="list-style-type: none"> A black check mark can be edited. A gray check mark is non-editable. It appears automatically for user-defined rules when the Always rule is selected. |
| <input checked="" type="checkbox"/> | Not allowed: If the interface rule (which applies to the interface as a whole) requires that the interface be connected, the element must not be connected. |
| <input type="checkbox"/> | Optional: The connection state of the element is not relevant to satisfy the interface rule that is defined. |

The following figure shows a rules table where two user-defined interface element rules have been configured in addition to the **Always** rule. It is used as an example to explain how rules apply to elements.

| Rules | | | | |
|--------|----------|----------|----------|--|
| | Element1 | Element2 | Element3 | |
| Always | ✓ | | | |
| Rule_1 | ✓ | ✗ | ✓ | |
| Rule_2 | ✓ | ✓ | | |

- The **Always** rule states that only **Element1** must always be connected (default configuration). **Rule 1** and **Rule 2** have been added and configured to manage the connection status of **Element2** and **Element3**.
- Rule 1** requires that **Element1** and **Element3** be connected and **Element2** not connected.
- Or, if **Rule 1** is not satisfied because **Element2** is connected, **Rule 2** requires that **Element1** must also be connected and the connection status of **Element3** is irrelevant.

NOTE: To configure user-defined rules, clear the **Always** rule. To do so, you can also right-click an element and select the **Switch to Optional** command. (Selecting the **Switch to Required** command, sets the **Always** rule.)

Facet Template Definition

Overview

This section describes the definition of facet templates, which defines how a template is built, behaves, and can be used.

Components of the Facet Template Definition

Overview

The definition of facet templates is divided into several categories. The data of each category can be viewed and/or edited in a pane of the same name. You can access these panes by using the **Facet Editor**, page 220.

The contents of each pane is described in the following topics.

Facet Template Definition Categories

The definition of facet templates is divided into the following categories. The pane of the same name allows you to view and/or configure data.

| Category/Pane | Description |
|-------------------------|---|
| Properties | Contains general information about the facet template. |
| Document Outline | Lists the various elements that are referenced by the facet template, including deferred ones, and allows opening a context menu for each one. For more information, refer to <i>Document Outline</i> , page 82. |
| Interface Rules | Defines the rules associated to interfaces referenced by the facet template. For more information, refer to <i>Interface Rules</i> , page 88. |
| Parameters | Defines the editable and non-editable data that is used when you instantiate the facet template. For more information, refer to <i>Template Parameter Common Definition</i> , page 85. |

NOTE: The **Toolbox** pane does not contain facet template definition information but binding functions that you can use with bindings. For more information, refer to the topic describing binding functions, page 108.

Facet Template Properties

Properties Pane Description

The following figure shows an example of the **Properties** pane of the **Facet Editor** in which facet template `$DEVCTL_UL` is open and no element is selected.

| Properties | |
|-----------------------------|-------------------------------------|
| \$DEVCTL_UL (1.0.10) | |
| General | |
| Identifier | \$DEVCTL_UL |
| Version | 1.0.10 |
| Type | Application |
| Subtype | Logic |
| Description | ON/OFF Device Control Unity Logic |
| Valid | <input checked="" type="checkbox"/> |
| Protection | Protected |
| State | Approved |
| Standard | <input checked="" type="checkbox"/> |
| Signature | 73e9b2b8-0bc8-40f2-a432-7984634294 |
| Attributes | |
| Control Platform | All |

| Property | Description |
|-------------------|---|
| Identifier | Refer to the description of properties common definitions, page 80. |
| Version | Version of the facet template, allowing you to use different versions of the facet template that have the same Identifier . For more information, refer to <i>Header and Properties Common Definition</i> , page 80. |
| Type | Type of the facet template, which can be either of: |

| Property | Description |
|-------------------------|--|
| | <ul style="list-style-type: none"> • Topological • Application |
| Subtype | <p>Sub-type of the facet template. Depending on the Type of the facet template, it can be one of the following:</p> <ul style="list-style-type: none"> • Topological <ul style="list-style-type: none"> ◦ Network ◦ Network device ◦ Station Node ◦ Controller ◦ Generic ◦ Device I/O • Application <ul style="list-style-type: none"> ◦ Logic ◦ HMI ◦ Data ◦ Genie ◦ Server Event ◦ Client Event |
| Description | You can enter a description by using free-form text. |
| Valid | Refer to <i>Header and Properties Common Definition</i> , page 80. |
| Protection | |
| State | |
| Standard | |
| Signature | |
| Control Platform | <p>Defines the platform with which a Control facet template is compatible. The property allows assigning the facet only to Control Participant projects, which have the same <i>Controller family</i> (see <i>EcoStruxure Process Expert, User Guide</i>).</p> <p>Possible values:</p> <ul style="list-style-type: none"> • All: Does not restrict assignment. • M340 • M580 • Quantum <p>You can use a comma-separated list to define more than one platform.</p> |

Composite Template Definition

Overview

This section describes the definition of composite templates, which defines how a template is built, behaves, and can be used.

Components of the Composite Template Definition

Overview

The definition of composite templates is divided into several categories. The data of each category can be viewed and/or edited in a pane of the same name. You can access these panes by using the **Composite Editor**, page 220.

The contents of each pane is described in the following topics.

Composite Template Definition Categories

The definition of composite templates is divided into the following categories. The pane of the same name allows you to view and/or configure data.

| Category/Pane | Description |
|-------------------------|---|
| Properties | Contains general information about the composite template. |
| Document Outline | Lists the various elements that are referenced by the composite template, including deferred ones, and allows opening a context menu for each one. For more information, refer to <i>Document Outline</i> , page 82. |
| Interface Rules | Defines the rules associated to interfaces referenced by the composite template and those deferred from child elements. For more information, refer to <i>Interface Rules</i> , page 88. |
| Element Rules | Defines the valid combination of required and optional template elements that can be selected during instantiation (see <i>EcoStruxure Process Expert, User Guide</i>). |
| Parameters | Defines the editable and non-editable data that is used when you instantiate the composite template. For more information, refer to <i>Template Parameter Common Definition</i> , page 85. |

NOTE: The **Toolbox** pane does not contain composite template definition information but binding functions that you can use with bindings. For more information, refer to the topic describing binding functions, page 108.

Composite Template Properties (Header)

Properties Pane Description

The following figure provides a typical view of the composite template **Properties** pane.

| Properties | |
|---------------------|-------------------------------------|
| \$Motor_UC (2.2.13) | |
| ▲ General | |
| Identifier | \$Motor_UC |
| Version | 2.2.13 |
| Type | Application ▼ |
| Subtype | Not Applicable |
| Description | ON/OFF Motor - Unity Control |
| Valid | <input checked="" type="checkbox"/> |
| Protection | Protected ▼ |
| State | Approved |
| Standard | <input checked="" type="checkbox"/> |
| Signature | c8ddf410-a29e-4887-8539-8e853e2 |
| ▲ Attributes | |
| Network type | None ▼ |

| Property | Description |
|---------------------------|---|
| Identifier | Refer to the description of <i>properties common definitions</i> , page 80. |
| Version | Version of the composite template, allowing you to use different versions of the composite template that have the same Identifier . For more information, refer to <i>Header and Properties Common Definition</i> , page 80. |
| Type | Type of the composite template, which can be either of: <ul style="list-style-type: none"> • Topological • Application |
| Sub-type | Sub-type of the composite template. A sub-type is defined only for composite templates of type Topological and can be either of: <ul style="list-style-type: none"> • Generic • Network Device • Station Node • Controller • Device IO |
| Description | You can enter a description by using free-form text. |
| Valid | Refer to <i>Header and Properties Common Definition</i> , page 80. |
| Protection | |
| State | |
| Standard | |
| Signature | |
| Hardware Reference | For more information, refer to <i>Attribute Common Definition</i> , page 84. |
| Hardware Family | |
| Network Type | |

Composite Template Element Rules

Purpose of Element Rules

Rules defined for elements of composite templates allow you to define the valid combination of required and optional elements to be selected during instantiation.

If one of the defined element rules is not satisfied when you instantiate the template, the **Data** status of the instance becomes invalid.

You can define rules for facet and composite elements, which are referenced by the composite template.

The **Element Rules** pane contains the rules table, which allows you to create and configure rules when a composite template references one or more elements. An element appears in the rules table as soon as it is referenced.

User-defined rules are applied by using the OR operator (for example, **Rule 1 OR Rule 2 OR Rule 3**).

NOTE: Verify that element rules do not conflict, page 79 with interface rules that are defined for an element of the template.

Element Rules Pane Description

The following figure shows an example of the **Element Rules** table in which the two default rules have been configured for the **Control** and **Supervision** elements (templates).

| Element Rules | | | |
|---------------|---------|-------------|--|
| | Control | Supervision | |
| Always | ✓ | | |
| Default | ✓ | ✓ | |
| | | | |
| + X | | | |

The information shown in the rules table is organized as follows:

- Columns: Each row corresponds to an element that is referenced.
- Rows: Each row corresponds to a rule that you can define for each of the listed elements.
 - **Always** (required): Rule that is created and selected by default for each element. As a result, existing rules and those you add are selected by default and non-editable.
 - **Default**: Defines if the element is selected by default if the **Always** is disabled.
 - **Rule n**: User-defined rule, where *n* is an incremental number starting at 1.
- Cells: Indicate the selection requirement for each rule of an element.
- **+/x**: Allows adding/deleting a user-defined rule.

NOTE: To configure user-defined rules, clear the **Always** rule. To do so, you can also right-click an element and select **Switch to Optional** (selecting **Switch to Required**, sets the **Always** rule).

Description of Element Selection Requirements

The following table describes the various selection requirements that you can define for an element by using rules.

| Icon | Description |
|--|---|
| <div><input checked="" type="checkbox"/> <input checked="" type="checkbox"/></div> | Required: The element is either always selected or if optional, selected by default. NOTE: <ul style="list-style-type: none">• A black check mark can be edited.• A gray check mark is non-editable. It appears automatically for the Default rule and for user-defined rules when the Always rule is selected. |
| <div><input checked="" type="checkbox"/></div> | Not allowed: Applies only to user-defined rules. The element must not be selected. |
| <div><input type="checkbox"/></div> | Optional: Selecting the element is possible and optional. |

The following figure shows a rules table where two user-defined element rules have been configured in addition to the **Always** and **Default** rules. It is used as an example to explain how rules apply to elements.

| Element Rules | | | | |
|---------------|---------|---------------|---------------|--|
| | Control | Supervision A | Supervision B | |
| Always | ✓ | | | |
| Default | ✓ | ✓ | | |
| Rule_1 | ✓ | ✗ | ✓ | |
| Rule_2 | ✓ | ✓ | ✗ | |

- The **Always** rule states that only **Control** is always selected and **Supervision A** and **Supervision B** are optional elements.
- The **Default** rule:
 - Is implicitly applied to **Control**.
 - States that **Supervision A** is selected by default (editable during instantiation).
 - States that **Supervision B** is not selected by default (editable during instantiation).

Rule 1 and **Rule 2** have been added and configured to provide selection restrictions for **Supervision A** and **Supervision B** elements so that only one or the other can be selected.

- **Rule 1** requires that **Supervision A** be not selected when **Supervision B** is selected.
- Or, if **Rule 1** is not satisfied because **Supervision B** is not selected, **Rule 2** requires that if **Supervision A** is selected, **Supervision B** must not be.

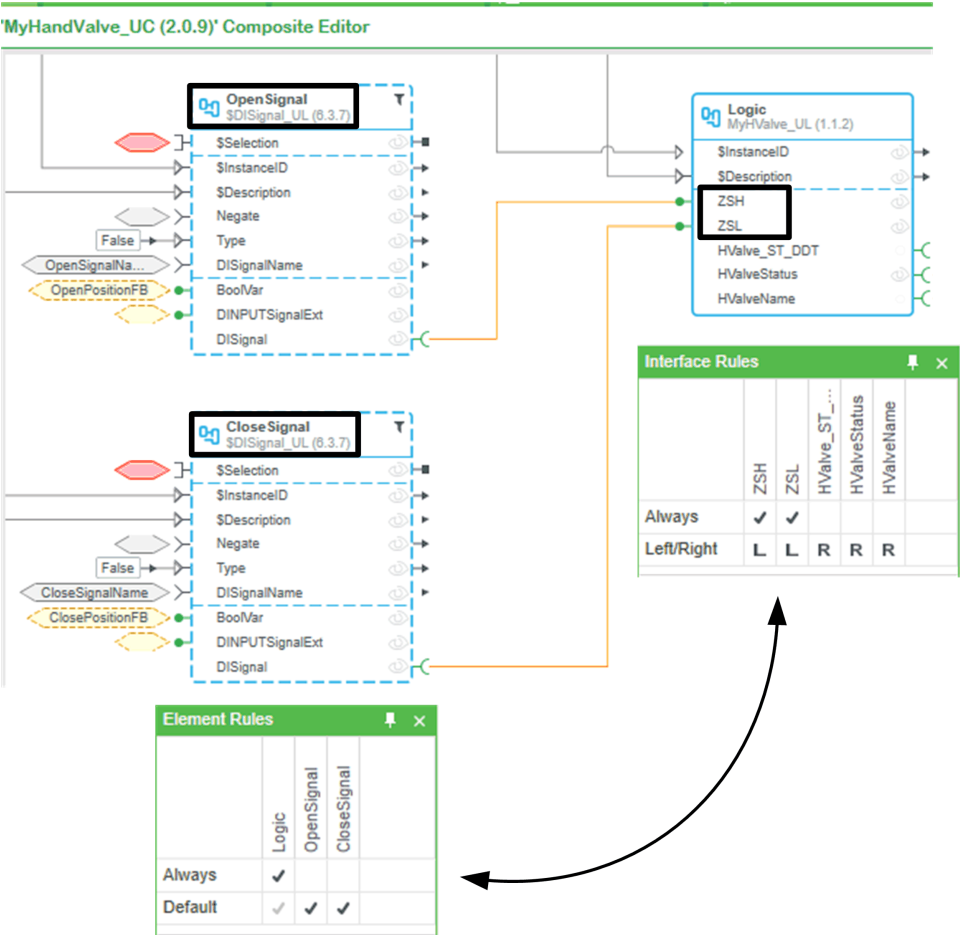
Element and Interface Rule Conflicts

Verify that element rules do not conflict with interface rules that are defined for elements of the template. This may lead to unwanted behavior when selecting services of the instance, such as setting the instance status to *invalid* if, for example, data is defined as mandatory at a lower level of the template composition while at a higher level, the same data is defined as optional.

In the following example, interface rules for *ZSH* and *ZSL* have been defined by editing *MyHValve_UL*. The rules require that data is always propagated to both of them through their connection to elements *OpenSignal* and *CloseSignal* respectively.

At the composite level (*MyHandValve_UC*), elements *OpenSignal* and *CloseSignal* are configured to be selectable at the instance level (*\$Selection* property) since no element rule applies to them (except that they are selected by default).

As a result, if either element is cleared at the instance level, the status of the instance is set to *invalid* because data is not propagated to *ZSH* or *ZSL*, which does not satisfy the interface rules.



Common Definitions

Overview

This section describes the common definitions that are shared by interface models, facet and composite templates during the Global Templates Definition (see *EcoStruxure Process Expert, User Guide*) stage.

Header and Properties Common Definition

Overview

This topic describes the definition that the following Global Templates have in common for the **Header** and **Properties** categories:

- Interfaces models (**Header**)
- Facet templates (**Properties**)
- Composite templates (**Properties**)

Identifier

The **Identifier** property indicates the name of a template or interface model.

Combined with **Version**, it is used as to identify templates uniquely within the Global Templates library.

NOTE: Starting with EcoStruxure Hybrid DCS 2019, the use of the \$ prefix is reserved to identify templates that are created by Schneider Electric. When you update, duplicate, or save such a template, you must change its identifier by, at least, removing the \$ prefix. This has an impact on the strategy to adopt when you modify, page 292 templates.

Templates provided by Schneider Electric that do not have the \$ prefix can be modified to meet your specific requirements.

Version

The **Version** property indicates the version of a template or interface model.

The table describes the three components of the version.

| Version component | Description | Editable | Valid range |
|-------------------|--|----------|-------------|
| Major | Used for major changes in the template behavior. | Yes | 0 to 99 |
| Minor | Used for minor changes in the template behavior. | Yes | 0 to 99 |
| Build number | Automatically managed by the platform, the build number is bound to the implementation of the templates. | Yes | 0 to 9999 |

NOTE: Version *0.0.0* is not allowed. However, when you open a newly created template in read-only or editing mode, you may see in the same location a template with the same identifier and version *0.0.0*, which is a temporary working copy. It is removed when you close the template editor.

Validity

The platform automatically verifies if the template or interface model satisfies the minimum requirements to produce instances. The result is indicated by using the **Valid** property.

| Value (check box) | Description |
|-------------------|--|
| Selected | The template satisfies the minimum requirements to produce instances. |
| Cleared | <p>One of the following rules is not satisfied:</p> <ul style="list-style-type: none"> Any templates: Rules cannot be verified as they depend on data provided during instantiation. The mandatory interfaces must be linked. Composite templates: <ul style="list-style-type: none"> Element rules of references to composite templates do not satisfy the minimum requirements. Element interface rules do not satisfy the minimum requirements. Validation parameter rules do not satisfy the minimum requirements. |

Protection Mode

The **Protection** property defines the following protection modes of a template.

| Mode | Description |
|--------------------|---|
| UnProtected | The template can be used, its contents viewed, and changed. |
| Read only | The template can be used and its contents viewed by using the corresponding editor but it cannot be modified. NOTE: Accessing the contents of the template requires a password, which you define when enabling this mode. |
| Protected | The template can be used but its contents cannot be viewed nor changed by using the corresponding editor. NOTE: Accessing the contents of the template requires a password, which you define when enabling this mode. |

NOTE: The **Protection** property is not functional in this version of Process Expert.

Usability State

The **State** property defines the following usability states of a template.

| Value | Description |
|---------------------|---|
| Not Approved | You cannot use the template to create instances or new references from another template. |
| Approved | You can use the template to create instances or new references from another template. |
| Deprecated | You can use the template to create instances or new references from another template. The state is used to indicate that although the template is outdated, it is kept for legacy purposes. |
| Obsolete | You cannot use the template to create instances or new references from another template. |

Standard

When selected, the **Standard** property indicates that the template is designed by Schneider Electric.

Signature

The **Signature** property is a unique identifier for the content of templates, which triggers a version change when it is modified. The property is used by the software to detect if two or more templates with the same identifier and version exist but which have different content.

The software automatically generates the signature when a new version of the template is created.

Document Outline

Overview

The **Document Outline** pane appears in the **Facet Editor** and **Composite Editor**. It lists the various elements that are referenced by the template including deferred ones. For each one, it allows opening an element-specific context menu, which lets you locate the element in the workspace and gives access to more detailed information about it.

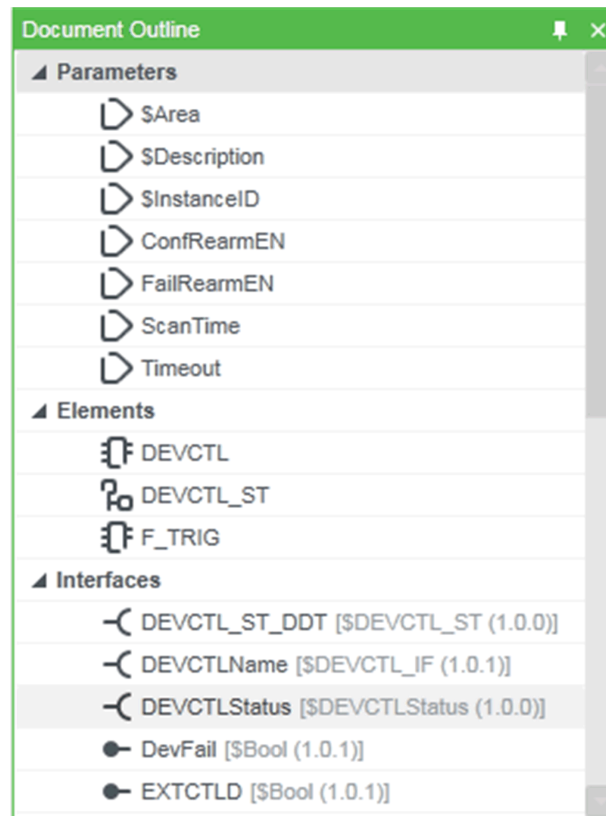
The various elements that appear in the **Document Outline** pane are grouped by type.



Clicking an element in the pane highlights it in the workspace, and the other way around, clicking an element in the workspace highlights it in the pane. In both cases, details are displayed in the **Properties** pane.

NOTE: For a description of the commands of the context menu, refer to the topic describing the corresponding editor.

Document Outline Pane Description

The following figure shows an example of the **Document Outline** pane in the **Facet Editor**.



| Element type | Description |
|-------------------|--|
| Parameters | System parameters, parameters of the template, and those deferred from child elements. For deferred parameters, the path is indicated. The parameters are defined in the Parameters pane, page 85. |
| Elements | Control and Supervision constituents as well as facet and composite references. The identifier and version of templates used by references are indicated. |
| Interfaces | Interfaces of the template and those deferred from child elements. The identifier and version of interface models used by interfaces are indicated. For deferred interfaces, the path is indicated. An icon indicates, which role, page 64 the template references. Role A:  Role B:  |
| Functions | Binding functions, page 108 used by the template. |
| Values | Values of input values, page 228. |

NOTE: A diamond-shaped icon indicates a deferred element.

Attribute Common Definition

Overview

This topic describes the **Attributes** section of the **Properties** pane that the following templates have in common:

- Facet templates
- Composite templates

Attributes are characteristics associated to facet and composite templates of the **Topological** type.

The following figure shows an example of the **Attributes** section.

| ▲ Attributes | |
|--------------------|----------------------|
| Hardware reference | BMENOS0300.2 |
| Hardware family | Communication;Rack ▼ |
| Network type | Ethernet ▼ |

Hardware Reference

The **Hardware reference** attribute defines the reference of the hardware module that the topological facet or composite template models.

For example, *BMENOS0300* for the composite template that models an Mx80 network option switch module.

If the attribute contains several references, they are separated by a semi-colon (without spaces).

Default value: Blank.

Network Type

The **Network type** attribute defines the type of communication that a template is compatible with. If the attribute is not applicable, its value is *None*.

Default value: Blank.

The value of the **Network type** attribute can be one of the following:

- *VJC Runtime*
- *Ethernet*
- *Modbus*
- *Local Bus*
- *Advantys CANopen*
- *CANopen*
- *Profibus DP*
- *RIO Ethernet*

Hardware Family

The **Hardware family** attribute defines the type of hardware that a topological template models. If the attribute is not applicable, its value is *None*.

For each subtype of template, a set of possible values is defined.

For certain templates, the attribute can contain more than one value. In such case, check boxes allow you to select more than one.

Default value: Blank.

The table indicates the possible values of the **Hardware family** attribute depending on the subtype of the template.

| Template subtype | Attribute value |
|---|---|
| <i>Network Device or Composite Network Device</i> | <i>Router</i> |
| | <i>Gateway</i> |
| | <i>PRM</i> |
| <i>Station node or Composite Station node</i> | <i>NIC</i> |
| | <i>Serial Port</i> |
| | <i>Platform Client</i> |
| | <i>Control</i> |
| | <i>Platform Server</i> |
| | <i>Platform Backup Server</i> |
| <i>Controller or Composite Controller</i> | <i>Processor</i> |
| | <i>Rack</i> |
| | <i>Power Supply</i> |
| | <i>Digital IO</i> |
| | <i>Analog IO</i> |
| | <i>Special</i> |
| | <i>Communication</i> |
| | <i>Other</i> |
| <i>Device I/O or Composite Device I/O</i> | <i>Advantys Communication</i> |
| | <i>Advantys Device</i> |
| | <i>Advantys Power</i> |
| | <i>Advantys Digital IO</i> |
| | <i>Advantys Analog IO</i> |
| | <i>Advantys Special</i> |
| | <i>Advantys CANopen Device</i> |
| | <i>Advantys Other</i> |
| | <i>Modbus Device</i> |
| | <i>Modbus TCP Device</i> |
| | <i>Generic Device</i> |
| <i>Generic or Composite Generic</i> | None. You can enter a value by using free form text. |

Parameter Common Definition

Overview

This topic describes the parameters that the following templates have in common:

- Facet templates
- Composite templates

Parameters consist of data that is used by the template and its elements during the instantiation stage (see *EcoStruxure Process Expert, User Guide*). This data defines the default configuration of the instance. Parameter values can be editable or non-editable at the instance level.

You can view and/or edit parameters of templates by using the **Parameters** pane of the **Composite Editor** and the **Facet Editor**.

NOTE: Editable parameters are configured by using the **Instance Editor**.

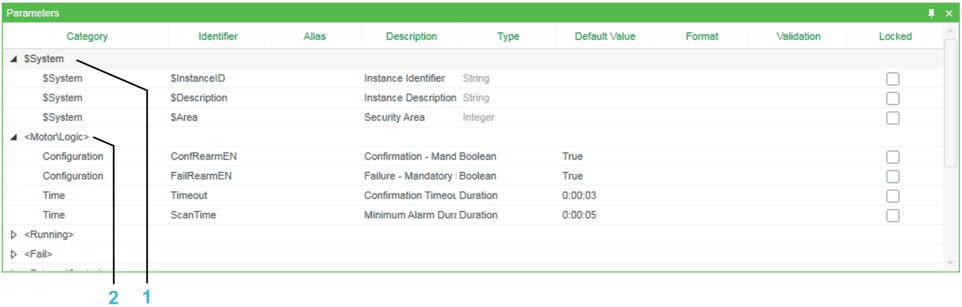
Implicit System Parameters

Each template contains the following implicit system parameters, which belong to the default **\$System** category and are identified by the \$ prefix:

- **\$InstanceID**: Unique identifier for each instance of a template, which is calculated by the software (see *EcoStruxure Process Expert, User Guide*).
- **\$Description**: User-defined description of the instance.
- **\$Area**: Defines the area that the instance belongs to. You can edit the value.

Parameters Pane

The following figure shows an example of the **Parameters** pane displaying the implicit and custom parameters of a template. You can open the pane from the editor toolbar menu (**View** entry).



| Item | Description |
|------|--|
| 1 | Implicit system parameters \$System . |
| 2 | Custom parameters. |

Properties of Parameters

The table describes the properties of parameters that are displayed in the **Parameters** pane.

| Property | Description |
|---------------|--|
| Category | Provides a way to group parameters by category. Categories can be used for sorting, filtering, and grouping parameters during instantiation. Expand a category to view the parameters it contains. |
| Identifier | Unique identifier within the template, which is used for: <ul style="list-style-type: none">• Bindings during definition stage.• Identification of the parameter during instantiation. |
| Alias | Not used in this version of the software. |
| Description | Description of the parameter that is displayed during instantiation. |
| Type | Data type (see <i>EcoStruxure Process Expert, User Guide</i>) of the parameter. |
| Default Value | Value of the parameter when no value is entered during instantiation or if the parameter is non-editable (not deferred). |
| Format | Defines the input format for the parameter, for example '0###' or '###.#'. |

| Property | Description |
|-------------------|--|
| | <p>You can also use the field as a description for the usage of Default Value. For example, if you use it to enter the path to a file, you can enter <code>File path</code>. Format is only visible when you edit the template; not during instantiation.</p> <p>NOTE: For parameters that are used with the Path attribute of document functions, page 115, setting the value to:</p> <ul style="list-style-type: none"> • Content Repository, enables the content repository browse (see <i>EcoStruxure Process Expert, Runtime Navigation Services, User Guide</i>) button: <ul style="list-style-type: none"> ◦ During template edition, in the Default Value property. It lets you select a document that is associated to the template by default. ◦ During instance edition (Instance Editor), in the corresponding parameter if it is deferred. • URL, enables a file browser button during template edition in the Default Value property. The button opens the Select file location dialog box, which lets you browse the computer and select a document that is associated to the template by default. <p>Once you have defined a default value, the Default Value property is replaced by Value.</p> |
| Validation | <p>Allows you to limit the number of characters that a parameter of type String can contain.</p> <p>For example, for Schneider Electric control module application templates, for example, <code>\$Motor</code>, the value for the <code>\$InstanceID</code> parameter is set to 18. This is the maximum number of characters that the instance identifier can contain.</p> <p>A blank field indicates that no validation applies.</p> <p>NOTE: For enumeration sets, the software verifies if the calculated value is contained in the enumeration set.</p> |
| Locked | <p>When the check box is selected, does not allow you to edit the parameter during instantiation.</p> |

Implicit Parameter Properties

The table provides the default value of the properties of implicit parameters.

| Parameter | Property | Default value |
|---------------|---------------|---|
| \$InstanceID | Category | \$System |
| | Identifier | \$InstanceID |
| | Alias | Blank |
| | Description | Instance Identifier |
| | Type | String |
| | Default Value | <p>The name of the instance (value of the \$Name parameter (see <i>EcoStruxure Process Expert, User Guide</i>)), which is by default <i>Template_Identifier_n</i>, where <i>n</i> is an incremental number starting at 1. For Schneider Electric templates, the \$ prefix is omitted.</p> <p>NOTE: When the hierarchical naming function (see <i>EcoStruxure Process Expert, User Guide</i>) is enabled for the instance, the alias of folders in the path of the instance is added as prefix.</p> <p>NOTE: The instance identifier needs to satisfy applicable naming rules (see <i>EcoStruxure Process Expert, User Guide</i>).</p> |
| | Format | Blank |
| | Validation | <p>Blank</p> <p>NOTE: For Schneider Electric control module application templates, for example, \$Motor, the value for the \$InstanceID parameter is set to 18. This is the maximum number of characters that the instance identifier can contain.</p> |
| \$Description | Locked | Not locked (check box cleared) |
| | Category | \$System |
| | Identifier | \$Description |
| | Alias | Blank |
| | Description | Instance Description |
| | Type | String |
| | Default Value | Blank |
| | Format | Blank |
| \$Area | Validation | Blank |
| | Locked | Not locked (check box cleared) |
| | Category | \$System |
| | Identifier | \$Area |
| | Alias | Blank |
| | Description | Security Area |
| | Type | Integer |
| | Default Value | Blank |
| | Format | Blank |
| | Validation | Blank |
| | Locked | Not locked (check box cleared) |

Interface Rules

Purpose of Interface Rules

Rules defined for interfaces of composite and facet templates allow you to define the valid combination of required and optional interfaces to be connected during instantiation.

If one of the defined interface rules is not satisfied when you instantiate the template, the **Link** status of the instance becomes invalid.

You can define rules for interfaces of the child elements that are exposed at the template level (deferred).

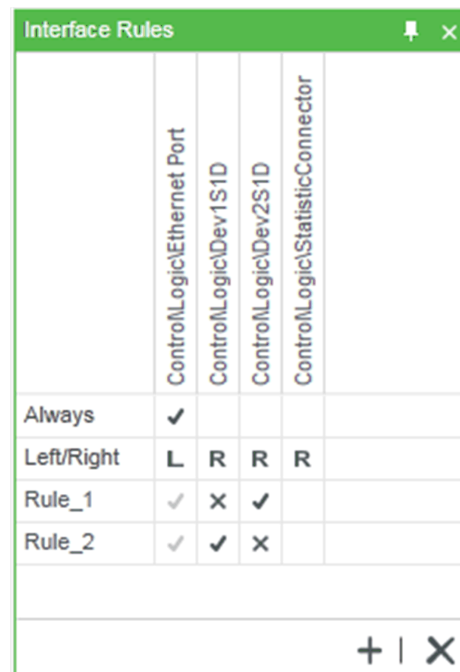
The **Interface Rules** pane contains the rules table, which allows you to create and configure rules when a composite or facet template references one or more deferred interfaces. An interface appears in the rules table as soon as it is deferred.

User-defined rules are applied by using the OR operator (for example, **Rule 1 OR Rule 2 OR Rule 3**).

NOTE: In a composite template, ensure that no element rule, page 79 conflicts with the interface rule of an element.

Interface Rules Pane Description

The following figure shows an example of an **Interface Rules** table in which two user-defined rules have been created in addition to the default **Always** and **Left/Right** rules.



| | Control\Logic\Ethernet Port | Control\Logic\Dev1S1D | Control\Logic\Dev2S1D | Control\Logic\StatisticConnector |
|------------|-----------------------------|-----------------------|-----------------------|----------------------------------|
| Always | ✓ | | | |
| Left/Right | L | R | R | R |
| Rule_1 | ✓ | X | ✓ | |
| Rule_2 | ✓ | ✓ | X | |

+ | X







The information shown in the rules table is organized as follows:

- Columns: Each row corresponds to an interface that is deferred at the template level.
- Rows: Each row corresponds to a rule that you can define for each of the listed interfaces.
 - **Always** (required): The interface must always be connected at the instance level. The rule is created and selected by default for each deferred interface. As a result, existing rules and those you add are selected by default and non-editable.
 - **Left/Right**: Defines the position of the interface connector. This can be at the element level in the **Composite Editor** of the parent template or at the instance level in the **Asset Workspace Editor/Links Editor**.
 - **Rule n**: User-defined rule, where *n* is an incremental number starting at 1.
- Cells: Indicate the connection and position requirement for each element.
- **+/X**: Allows adding/deleting a user-defined rule.

NOTE: To configure user-defined rules, clear the **Always** rule. To do so, you can also right-click an interface and select the **Switch to Optional** command. (Selecting the **Switch to Required** command, sets the **Always** rule.)

Description of Interface Connection and Position Requirements

The following table describes the various connection and position requirements that you can define for an interface by using rules.

| Icon | Description |
|--|---|
|   | Required: The interface must always be connected. NOTE: <ul style="list-style-type: none"> A black check mark can be edited. A gray check mark is non-editable. It appears automatically for the for user-defined rules when the Always rule is selected. |
|  | Not allowed: Applies only to user-defined rules. The interface must not be connected. |
|  | Optional: Connecting the interface is possible and optional. |
|  | The interface connector appears on the left side of the element or instance. |
|  | The interface connector appears on the right side of the element or instance. |

The following figure shows a rules table where two user-defined interface rules have been configured in addition to the **Always** and **Left/Right** rules. It is used as an example to explain how rules apply to interfaces.

| Interface Rules | | | | | |
|-----------------|------------|------------|------------|------------|--|
| | Interface1 | Interface2 | Interface3 | Interface4 | |
| Always | ✓ | | | | |
| Left/Right | L | R | R | R | |
| Rule_1 | ✓ | ✓ | X | | |
| Rule_2 | ✓ | X | ✓ | | |

- The **Always** rule states that only **Interface1** must always be connected.
- The **Left/Right** rule states that:
 - Interface1** appears on the left side of the element/instance.
 - The other interfaces appear on the right side.

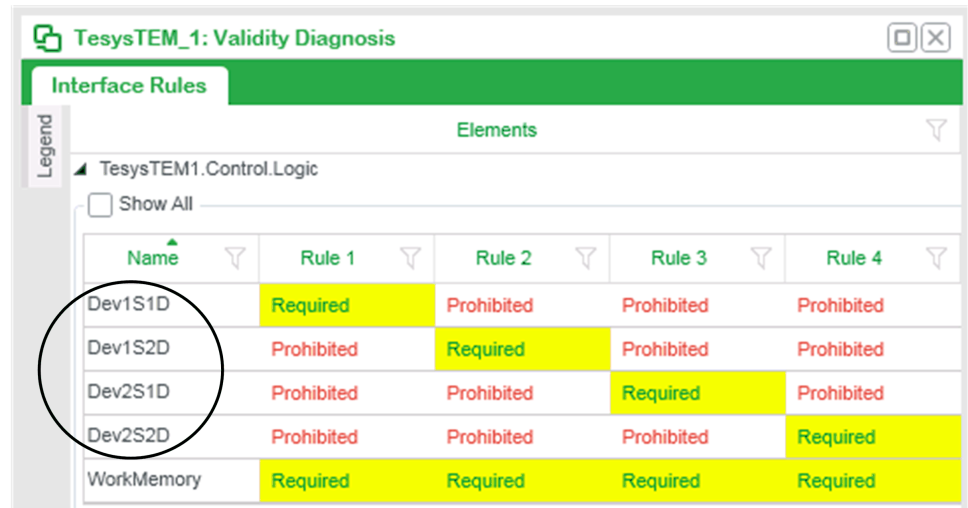
Rule 1 and **Rule 2** have been added and configured to provide connection restrictions for **Interface2** and **Interface3** so that one or the other must be connected.

- Rule 1** requires that **Interface2** be connected but not **Interface3**.
- Or, if **Rule 1** is not satisfied because **Interface3** is connected, **Rule 2** requires that **Interface2** must not be connected.
- The connection state of **Interface4** is irrelevant.

Interface Rules and Template Hierarchy

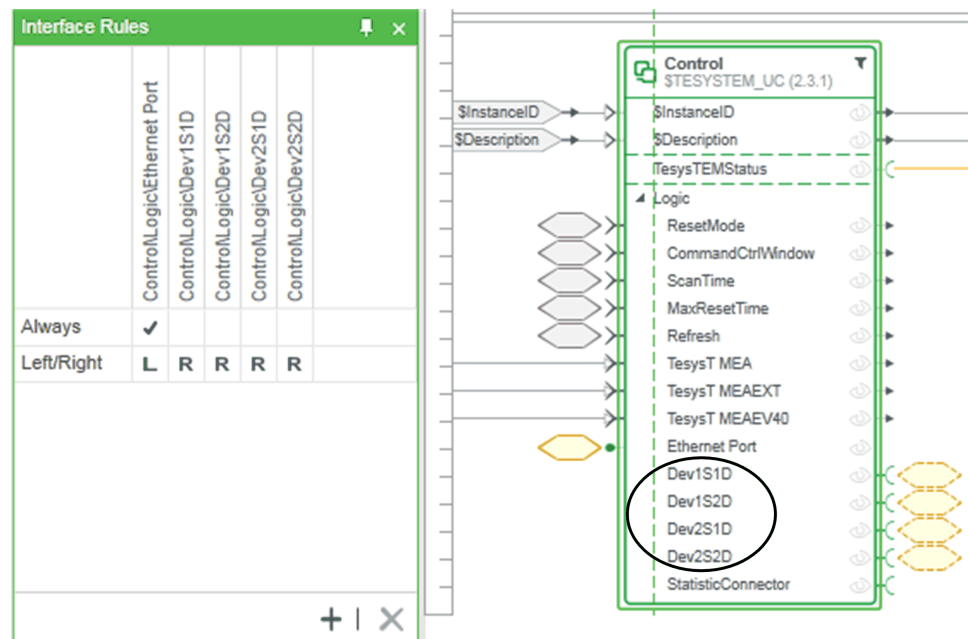
It is possible that rules already exist for an interface at various lower levels of the template hierarchy. These can affect the behavior of rules that you establish at the template level. Such rules are typically defined in the template referencing the interface itself but can exist at each level where the interface is deferred.

This example shows the interface rules for an instance of *\$TesySTEM*, which require that one of the four device speed/direction interfaces be connected.

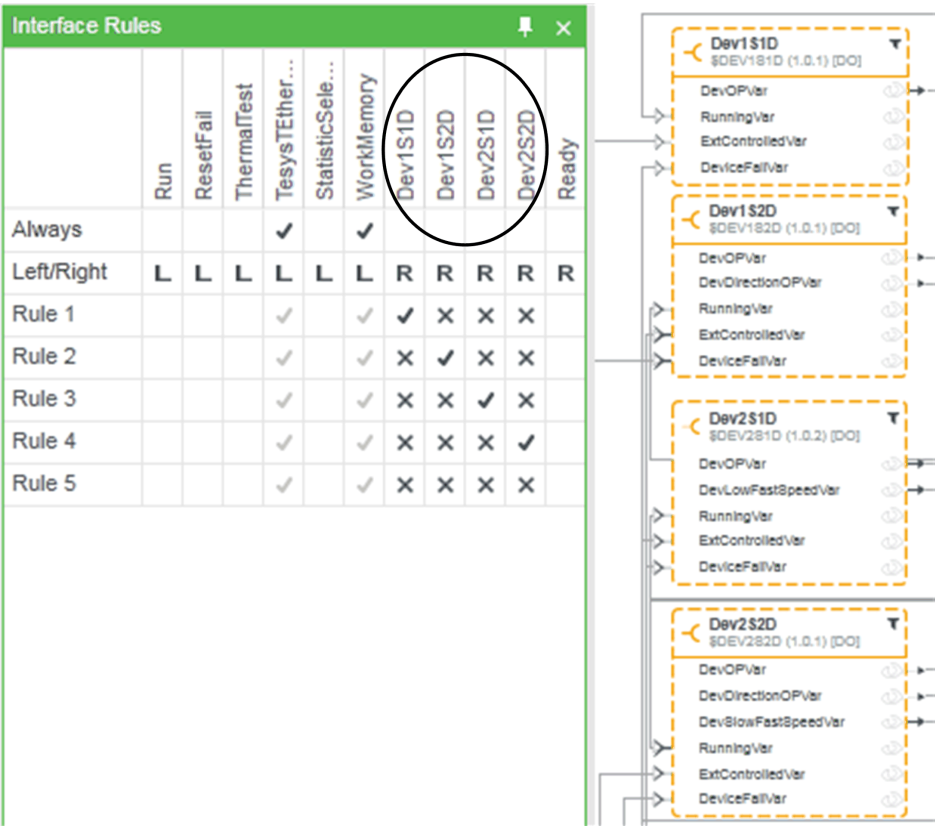


| Name | Rule 1 | Rule 2 | Rule 3 | Rule 4 |
|------------|------------|------------|------------|------------|
| Dev1S1D | Required | Prohibited | Prohibited | Prohibited |
| Dev1S2D | Prohibited | Required | Prohibited | Prohibited |
| Dev2S1D | Prohibited | Prohibited | Required | Prohibited |
| Dev2S2D | Prohibited | Prohibited | Prohibited | Required |
| WorkMemory | Required | Required | Required | Required |

The following figure shows the interface rules that exist for these four device speed/direction interfaces in the *\$TesySTEM* template. It shows that the connection restriction is not defined at the control module template level.



The following figure shows the interface rules that exist when you drill down two levels in the template hierarchy. It shows that the interface rules are defined here in the facet template referencing the four device speed/direction interfaces.



Elements

Overview

Elements are referenced by facet templates and are Participant-specific.

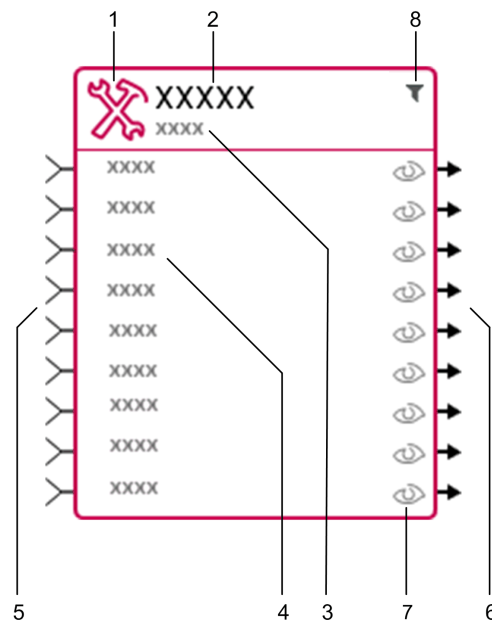
You can create both types of elements by encapsulating Control or Supervision constituents, page 260. In addition, you can select Supervision elements from the **Toolbox** pane, page 238 of the **Facet Editor**.

This contains general information about elements and describes some elements more in detail.

General Description of Elements

Element Representation

The following figure shows a generic element to describe its main components.



| Item | Description |
|------|---|
| 1 | Standard icon representing the type of element. |
| 2 | Element identifier. Depending on the type of element, it may be editable. |
| 3 | Element type such as DFB or variable type, variable tag type, genie name. |
| 4 | Properties that have been defined during Control constituent encapsulation or that correspond to Participant properties for Supervision elements coming from the Toolbox (for example, variable tags) or Include projects (for example, animated graphics, page 280). Right-click a property to open a context menu, page 228. |
| 5 | Input pins acting as destination for bindings. Each pin accepts only one connection. |
| 6 | Output pins acting as origin for bindings. Each pin accepts several connections. |
| 7 | Control to check or uncheck the property so that it can be hidden by using the Hide Unchecked element filter, page 240. |
| 8 | Filter icon indicating that a filter, page 240 is applied. |

Calculated Variable Tag Supervision Element

Overview

The **Calculated Variable Tag** element lets you create in the Supervision Participant project a variable tag whose **Address** property is populated with either:

- A Cicode expression.
- A Cicode function.

The other properties of the element are identical to those of the **Variable Tag** element and properties of the variable tag, page 53 in the Supervision project are populated in the same way.

You can select the element from the **Toolbox** pane, page 238 of the **Facet Editor** and reference it in **Data** Supervision facet templates.

You can reference several **Calculated Variable Tag** elements per facet template and use them in combination with other available tag elements.

Working Principle

Based on predefined rules, the variable tag address takes over the content of either the **CicodeToCall** or **Expression** property of the **Calculated Variable Tag** element. If both values are null, a variable tag is created but the **Address** property value is left blank.

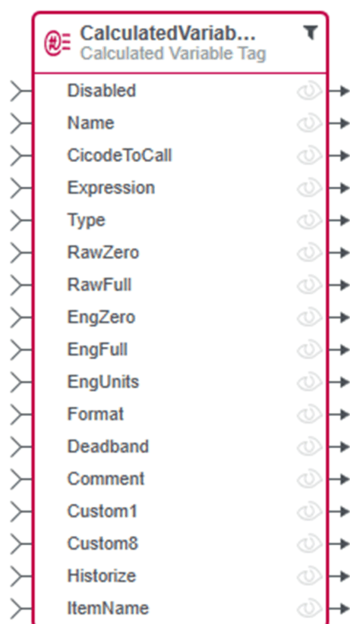
When you use the element, a specific I/O device (see *EcoStruxure Process Expert, User Guide*) is created in the Supervision project but not associated to the tag container to which the facet is assigned. The **I/O Device** property of the calculated variable tag is configured automatically with the appropriate value.

In the Supervision runtime, the result of the expression or Cicode function is used as the tag value.

The values of the other properties of each variable tag are generated based on the value that is configured for the corresponding property in the **Calculated Variable Tag** element.

Element Representation

The following figure shows a **Calculated Variable Tag** element as it can appear in the Facet Editor, page 227 with properties **Custom 2** to **Custom 7** hidden.



The properties of the element are of type String except for **Disabled**, which is boolean.

| Property | Description |
|-----------------|---|
| Disabled | When the boolean value provided to the input pin is true (enter <code>true</code> or <code>1</code>), the element is disabled and does not generate and/or configure a calculated variable tag in the Supervision project. Blank or not connected is equal to false (0). |
| Name | Same as for the Variable Tag property, page 53. Typically, corresponds to a property of an animated graphic (genie). |

| Property | Description | | | | | | | | |
|-----------------------------------|--|------------------------|--------|---------------------|---|------------------------|---|-----------------------------------|--|
| CicodeToCall | <p>Provide a Cicode function and its arguments that will be called in runtime:</p> <ul style="list-style-type: none"> The string cannot exceed 254 characters and cannot contain spaces. You must use a valid Cicode function. <p>During generation, the above requirements are verified and a notification is displayed to inform you if either one is not satisfied.</p> <p>When both CicodeToCall and Expression contain a value, the value of CicodeToCall is used.</p> <p>The property can be left blank.</p> <p>You can use the following syntax for the Cicode function. For details on Cicode functions and calculated variables, refer to the help of the Supervision Participant.</p> <table> <tr> <th>Cicode function syntax</th><th>Result</th></tr> <tr> <td><i>FunctionName</i></td><td>Automatically uses the <i>Cluster Name</i> and <i>Equipment</i> property values as arguments with the function. <i>FunctionName</i>("ClusterName", "Equipment")</td></tr> <tr> <td><i>FunctionName</i>()</td><td>Does not use arguments with the function.</td></tr> <tr> <td><i>FunctionName</i>(<Arguments>)</td><td>Uses the arguments that you declare with the function.</td></tr> </table> | Cicode function syntax | Result | <i>FunctionName</i> | Automatically uses the <i>Cluster Name</i> and <i>Equipment</i> property values as arguments with the function. <i>FunctionName</i> ("ClusterName", "Equipment") | <i>FunctionName</i> () | Does not use arguments with the function. | <i>FunctionName</i> (<Arguments>) | Uses the arguments that you declare with the function. |
| Cicode function syntax | Result | | | | | | | | |
| <i>FunctionName</i> | Automatically uses the <i>Cluster Name</i> and <i>Equipment</i> property values as arguments with the function. <i>FunctionName</i> ("ClusterName", "Equipment") | | | | | | | | |
| <i>FunctionName</i> () | Does not use arguments with the function. | | | | | | | | |
| <i>FunctionName</i> (<Arguments>) | Uses the arguments that you declare with the function. | | | | | | | | |
| Expression | <p>Enter free-form text.</p> <p>The string cannot exceed 254 characters. During generation, a notification is displayed to inform you if the requirement is not satisfied.</p> <p>Expression is not used if CicodeToCall contains a value.</p> <p>The property can be left blank.</p> | | | | | | | | |
| Other properties | <p>Correspond to the properties of variable tags in the Supervision Participant project and are used to populate the variable tag table.</p> <p>NOTE: Certain properties are not managed by using the Calculated Variable Tag element.</p> | | | | | | | | |

Disk Variable Tag Supervision Element

Overview

The **Disk Variable Tag** element lets you create in the Supervision Participant project a reference to a variable whose value is stored in a disk I/O device, which is created at the same time.

The properties of the element are identical to those of the **Variable Tag** element and properties of the *variable tag*, page 53 in the Supervision project are populated in the same way.

You can select the element from the **Toolbox** pane, page 238 of the **Facet Editor** and reference it in **Data** Supervision facet templates.

You can reference several **Disk Variable Tag** elements per facet template and use them in combination with other available tag elements.

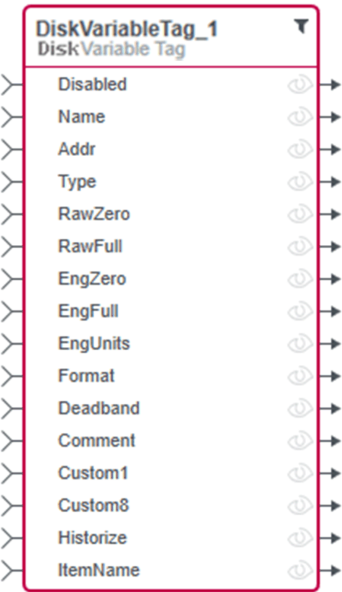
Working Principle

When you use the **Disk Variable Tag** element, an additional specific I/O device (see *EcoStruxure Process Expert, User Guide*) is created in the Supervision project and associated to the tag container to which the facet is assigned. The **I/O Device** property of the disk variable tag is configured automatically with the appropriate value.

The values of the other properties are generated based on the value that is configured for the corresponding property in the **Disk Variable Tag** element.

Element Representation

The following figure shows a **Disk Variable Tag** element as it can appear in the Facet Editor, page 227 with properties **Custom 2** to **Custom 7** hidden.



The properties of the element are of type String except for **Disabled**, which is boolean.

| Property | Description |
|------------------|---|
| Disabled | When the boolean value provided to the input pin is true (enter <code>true</code> or 1), the element is disabled and does not generate and/or configure a calculated variable tag in the Supervision project. Blank or not connected is equal to false (0). |
| Name | Same as for the Variable Tag property, page 53. Typically, corresponds to a property of an animated graphic (genie). |
| Addr | Reference to the control variable, page 53 that is monitored. |
| Other properties | Correspond to the properties of variable tags in the Supervision Participant project and are used to populate the variable tag table. |

Equipment Supervision Element

Overview

The **Equipment** element lets you create a user-configured equipment instance and configure its properties in the Supervision Participant project.

You can select it from the **Toolbox** pane, page 238 of the **Facet Editor** and reference it in **Data** and **Genie** Supervision facet templates.

Although only one element per facet template is allowed, you can reference several elements per instance. This lets you aggregate, for example, one type of alarm for several motors that are represented by various instances independently of the application hierarchy.

You can use it in place of the default method to generate the equipment name based on the hierarchy of the application (see *EcoStruxure Process Expert, User Guide*) or as a combination.

Working Principle

Each instance of a **Data** or **Genie** Supervision facet template that references a properly configured **Equipment** element creates an equipment instance.

The value of the **Name** property of equipment in the Supervision project is generated by concatenating the values of the **NamePath** and **NameID** properties configured for the **Equipment** element. If the resulting value is null, the equipment name is generated based on the hierarchy of the application. Other rules apply.

The software may truncate the resulting concatenated value based on format and length requirements that apply to the aforementioned properties.

For a same cluster, the software accepts only unique equipment names. As a result, when more than one Supervision facet associated to the same cluster generate the same equipment name, these equipment instances are merged and the property values of the facet that is generated last are used to update the single equipment instance. It does not matter whether the facets are referenced by the same instance in the application or different instances.

The values of the other properties of each equipment instance are generated based on the value that is configured for the corresponding **Equipment** element property. Some properties are not managed by the element.

Element Representation

The following figure shows an **Equipment** element as it can appear in the Facet Editor, page 227 with properties **Custom 2** to **Custom 7** hidden.



The properties of the element are of type String except for **Disabled**, which is boolean.

| Property | Description |
|--|---|
| Disabled | <p>When the boolean value provided to the input pin is true (enter <code>true</code> or 1), the element does not generate an equipment instance and cannot be used to configure the properties of an equipment instance in the Supervision project. Blank or not connected is equal to false (0).</p> <p>For an instance of the application, if one Equipment element that it references is disabled, an equipment instance is created but the equipment name is generated based on the location of the instance in the application (see <i>EcoStruxure Process Expert, User Guide</i>).</p> |
| NamePath | <p>Cannot exceed 105 characters and must satisfy the following requirements:</p> <ul style="list-style-type: none"> Maximum number of levels⁽¹⁾ when concatenated with NameID: 10 Maximum number of characters per level⁽¹⁾: 12 <p>The remaining characters are reserved for the following:</p> <ul style="list-style-type: none"> A 2-digit index per level that the software may add to keep the value unique in case of equipment name uniqueness conflict (represents 14 characters maximum). A dot separator for levels (represents 7 characters maximum). No level name must consist of only digits (for example, Cell1.1050.Unit5). <p>During generation, the software removes levels and truncates strings that exceed these numbers. It also removes spaces. A notification is displayed to inform you.</p> <p>Use dots only to define levels within the string.</p> <p>The property can be left blank if at least NameID is configured, page 99.</p> <p>NOTE: You can connect the input to binding functions, page 123 that create a string based on the location of an instance in the application of the system. In such case, select the dot as separator.</p> |
| NameID | <p>Cannot exceed 62 characters and must satisfy the following requirements:</p> <ul style="list-style-type: none"> Maximum number of levels⁽¹⁾: 3 Maximum number of characters per level⁽¹⁾: 18 <p>The remaining characters are reserved for the following:</p> <ul style="list-style-type: none"> A 2-digit index per level that the software may add to keep the value unique in case of equipment name uniqueness conflict (represents 6 characters maximum). A dot separator for levels (represents 2 characters maximum). No level name must consist of only digits (for example, Motor1.1050.Sensor2). <p>The software removes levels and truncates strings that exceed these numbers. It also removes spaces.</p> <p>Use dots only to define levels within the string. Leading dots are removed, page 99 when NamePath is configured.</p> <p>The property can be left blank if at least NamePath is configured.</p> <p>When NamePath is left blank, specific rules apply, page 99.</p> |
| Other properties | <p>Correspond to the properties of equipment in the Supervision Participant project and are used to populate the equipment table during generation.</p> <p>NOTE: Certain properties are not managed, page 98 by using the Equipment element.</p> |
| (1) Refers to the levels in the logical equipment hierarchy of the Supervision project that you want to define and use at runtime. | |

Properties not Generated by the Equipment Element

The table describes which properties are not managed by using the **Equipment** element and the source of the corresponding value that is generated in equipment instances.

| Property | Generated value |
|---------------------|---|
| Cluster Name | Corresponds to the cluster of the tag container to which the facet referencing the Equipment element is assigned to. |
| I/O Device | Corresponds to the I/O device that is associated to the tag container to which the facet referencing the Equipment element is assigned to. |

| Property | Generated value |
|----------------|---|
| Area | Corresponds to the \$Area parameter that is configured for the instance that references the facet containing the Equipment element. |
| Project | The property is not managed by EcoStruxure Process Expert but by the Supervision Participant. |

Equipment Name Creation Rules When Using the Equipment Element

The following tables describe how the software creates the equipment name depending on the values that you configure for the **NamePath** and **NameID** properties of the **Equipment** element.

Scenario 1:

| NamePath | NameID | Result |
|------------------------|------------------------|--|
| A value is configured. | A value is configured. | <p>The equipment name is the result of the concatenation of NamePath and NameID.</p> <p>The other properties of the Equipment element are used to populate the equipment table of the equipment instance.</p> <p>NOTE: If the value of NameID starts with a dot, it is removed.</p> |

Examples:

| NamePath | NameID | Equipment name | Comment |
|--------------|----------|---------------------|--|
| Cell1.Unit1 | Motor_1 | Cell1.Unit1.Motor_1 | – |
| Cell1.Unit1 | .Motor_1 | Cell1.Unit1.Motor_1 | The leading dot of NameID is removed. |
| Cell1.Unit 1 | Motor 1 | Cell1.Unit1.Motor 1 | Spaces in NamePath and NameID are removed. |

Scenario 2: Extending the equipment name that is based on the application hierarchy with up to two levels.

| NamePath | NameID | Result |
|----------|--|--|
| Blank | A value is configured, which starts <i>with</i> a dot. | <p>The software uses a combination of the two methods to generate equipment.</p> <p>The equipment name is the result of the concatenation of the location of the instance in the application (see <i>EcoStruxure Process Expert, User Guide</i>) and NameID.</p> <p>The other properties of the Equipment element are used to populate the equipment table of the equipment instance.</p> <p>NOTE: The following rules apply:</p> <ul style="list-style-type: none"> Maximum number of folders: 7 plus, instance identifier. For details, refer to the equipment naming rules (see <i>EcoStruxure Process Expert, User Guide</i>) when the equipment is generated based on the application. Maximum number of levels for NameID: 2 |

Examples:

| Instance path in the application | Instance identifier | NamePath | NameID | Equipment name |
|----------------------------------|---------------------|----------|----------------|--------------------------------------|
| System_1\Folder_1\Folder_2 | Motor_1 | | .Run | Folder_1.Folder_2.Motor_1.Run |
| System_1\Folder 1\Folder 2 | Motor 1 | | .Run | Folder1.Folder2.Motor1.Run |
| System_1\Folder_1\Folder_2 | Motor_1 | | .Signal.Run.On | Folder_1.Folder_2.Motor_1.Signal.Run |

Scenario 3:

| NamePath | NameID | Result |
|----------|---|--|
| Blank | A value is configured, which starts <i>without</i> a dot. | <p>The equipment name is created by using NameID.</p> <p>The other properties of the Equipment element are used to populate the equipment table of the equipment instance.</p> <p>NOTE: Refer to the description of the NameID property, page 97 for applicable rules.</p> |

Examples:

| Instance path in the application | Instance identifier | NamePath | NameID | Equipment name |
|----------------------------------|---------------------|----------|-------------------|----------------|
| System_1\Folder_1\Folder_2 | Motor_1 | | Run | Run |
| System_1\Folder 1\Folder 2 | Motor 1 | | Run.S1 | Run.S1 |
| System_1\Folder_1\Folder_2 | Motor_1 | | Run.S1.On.Forward | Run.S1.On |

Scenario 4:

| NamePath | NameID | Result |
|----------|--------|--|
| Blank | Blank | <p>The equipment name is created by using the location of the instance in the application (see <i>EcoStruxure Process Expert, User Guide</i>).</p> <p>The other properties of the Equipment element are used to populate the equipment table of the equipment instance.</p> |

Examples:

| Instance path in the application | Instance identifier | NamePath | NameID | Equipment name |
|----------------------------------|---------------------|----------|--------|---------------------------|
| System_1\Folder_1\Folder_2 | Motor_1 | | | Folder_1.Folder_2.Motor_1 |
| System_1\Folder 1\Folder 2 | Motor 1 | | | Folder1.Folder2.Motor1 |

Generating Equipment from Supervision Genie Facets

When a Supervision genie facet references a properly configured **Equipment** element, it generates an equipment instance when you add the animated graphic to a Supervision page and save changes.

Depending on the configuration of the element, it may generate a different equipment instance than the data facet pertaining to the same instance.

The **Cluster Name** property value of the equipment instance that is generated by the genie facet is related to the data facets pertaining to the same instance.

The table describes how the **Cluster Name** property of the equipment instance that is generated by the Supervision genie facet is populated depending on the scenario.

| Scenario | Result |
|---|--|
| The genie facet is generated but no data facet pertaining to the same instance has been generated (the data facet may be assigned to the Supervision project but not generated yet). | An equipment instance is created in the Supervision project for the genie facet but the Cluster Name property remains empty. |
| One or more data facets pertaining to the same instance have been generated after the genie facet. The data facets do not need to reference an Equipment element. | The Cluster Name property of the equipment instance generated by the genie facet is populated. The value corresponds to the cluster to which the data facet that was generated last is associated to. If the data facets reference an Equipment element, which generates a different equipment instance than the genie facet, the Cluster Name property of the equipment instance generated by the genie facet is populated in the same way. |
| A Supervision genie facet has been generated after one or more data facets pertaining to the same instance had been generated. The data facets do not need to reference an Equipment element. | The Cluster Name property of the equipment instance generated by the genie facet is populated. The value corresponds to the cluster to which either data facet is associated to. If the data facets are associated to several clusters, the value corresponds to the first cluster of the Supervision project in the Projects Explorer to which either data facet is associated to. |

Updating Equipment Instances

When you generate an already generated Supervision facet that references the **Equipment** element after you have made changes to components of the Supervision project, the instance referencing the facet, the facet itself, or the **Equipment** element, the changes may have an impact on the corresponding equipment instance, for example, by changing values of properties. In such case, a notification is displayed to inform you of the change.

The following tables describe the impact on existing equipment instances when you generate changes to already generated Supervision facets depending on the type of change.

| Updated Equipment element property | Impact on the corresponding existing equipment instance | Comments |
|---|---|---|
| NamePath and/or NameID | The Name property is updated. | <ul style="list-style-type: none"> If another facet generates an updated equipment name that is identical to an existing name: <ul style="list-style-type: none"> If the facet belongs to a different cluster, a new equipment instance is created. If the facet belongs to the same cluster, no new equipment instance is created to comply with the equipment name uniqueness requirement. The properties of the existing equipment instance are updated (if applicable) by using the property values of the facet that is generated last. <p>The behavior may vary if the software needs to truncate either parameter value during generation. Refer to the examples, page 103.</p> <p>NOTE: It is irrelevant whether the existing equipment name has been created by using the default method or the Equipment element.</p> If the updated equipment name is null (blank) and if no other Equipment element generates the same name, the corresponding equipment instance is deleted. Instead, an equipment instance is created by using the default method (see <i>EcoStruxure Process Expert, User Guide</i>) unless one already exists for the instance of the application. Its properties are configured or updated according to the configuration of the Equipment element. |
| Other properties of the Equipment element | The corresponding properties are updated. | <p>Properties whose updated value is null⁽¹⁾ do not change the current value of the equipment instance.</p> <p>To clear the value of an equipment property instance, page 104, make the updated value empty⁽²⁾. This changes the existing value to empty. This method requires that you update the property value in the same facet (setting its status to Out Of Date).</p> <p>Refer to the examples, page 103.</p> |
| <p>(1) A value is considered null if, at the template level, the input pin of the element property is not connected.</p> <p>(2) A value is considered empty if, at the template level, the input pin of the element property is connected (for example, to a parameter) but no value is provided through the binding.</p> | | |

| Change to the instance or its Supervision data facet | Impact on the corresponding existing equipment instance |
|--|---|
| Area of the instance. | The Area property of the equipment instances that are generated by facets of the instance are updated. |
| Assigning the Supervision data facet to the tag container of another cluster. | The Cluster Name property is updated. |
| Unassigning the Supervision data facet from the tag container or deleting the instance from the application. | The Cluster Name property is changed to null (blank). |

The following scenarios apply only if the equipment instance is created by, at least, a data facet.

| Change to a Supervision project component | Impact on the corresponding existing equipment instance |
|---|---|
| Identifier of the cluster of the Supervision project. | The Cluster Name property is updated. |
| Deleting the tag container or the cluster of the Supervision project. | The Cluster Name property is changed to null (blank). |
| Renaming the I/O device or associating the tag container to a different I/O device. | The I/O Device property is updated when you build the executable of the Supervision project. |

Deleting Equipment Instances from the Supervision Project

The table describes the various ways to delete equipment instances from the Supervision project. It is implied that the necessary steps to apply the described actions are performed (for example, a generation or the update of templates). In addition, it is implied that the actions are performed on the objects that generate the equipment instances that you want to delete.

| Action | Result |
|--|---|
| At the instance/facet level: | |
| The instances that reference the Supervision facets from which the equipment instances have been generated are deleted. | The corresponding equipment instances are deleted from the equipment table of the Supervision project. |
| The Supervision services that create the facets from which the equipment instances have been generated are unselected at the instance level. | |
| The facets from which the equipment instances have been generated are unassigned from the Supervision project. | |
| At the element level: | |
| The Equipment element is removed from the facets from which the equipment instances have been generated. | The corresponding equipment instances are deleted from the equipment table of the Supervision project. However, one equipment is created for the instance of the application by using the default method (see <i>EcoStruxure Process Expert, User Guide</i>). |
| The configuration of the NamePath and NameID properties of the Equipment elements from which the equipment instances have been generated is modified so that the resulting value is null (blank). | |
| The Equipment elements are disabled in facets from which the equipment instances have been generated. | |

Equipment Generation Examples

Overview

This topic illustrates the rules that apply when you generate several facets referencing **Equipment** elements in different situations.

Unless mentioned otherwise, the examples also apply to the properties of other elements that are referenced by Supervision data and/or genie facets, such as **Equipment Parameter** or **Equipment Group of Messages** elements.

Example 1: Merging of Equipment Instances or Creation of New Equipment

Example for the **Equipment** element only.

The following scenarios show the result when you generate, within the same cluster, another facet that references a Supervision element, depending on the values of its **NamePath** and **NameID** parameters.

NOTE: The scenarios apply also if **NamePath** is left blank and **NameID** starts with a dot, page 99. In this case, the path to the instance plus the instance identifier are used in place of **NamePath**.

Scenario 1: The concatenation of **NamePath** (Cell1.Unit1) and **NameID** (Motor) of the not yet generated facet is the same as the equipment name of the already generated facet. No truncation will be performed during generation.

| Item | Already generated value | Not yet generated value | Result after generating Facet_2 |
|--------------------------------------|--|-------------------------|---|
| Facet identifier | Facet_1 | Facet_2 | |
| Equipment name of equipment instance | Cell1.Unit1.Motor NOTE: This equipment name may be the result of a truncation. | Cell1.Unit1.Motor | Both equipment are merged into one equipment: Cell1.Unit1.Motor |

Scenario 2: The concatenation of **NamePath** (Cell12345678.Unit1) and **NameID** (Motor) of the not yet generated facet is different from the equipment name of the already generated facet but the truncation that the software performs, page 97 during generation will result in the same equipment name.

| Item | Already generated value | Not yet generated value | Result after generating Facet_2 | |
|--------------------------------------|---|--|---------------------------------|--|
| Facet identifier | Facet_1 | Facet_2 | Facet_1 | Facet_2 |
| Equipment name of equipment instance | Cell1234567.Unit1.Motor NOTE: Cell1234567 may be the result of a truncation. The value of NamePath before being generated may be, for example, Cell123456789.Unit1 | Cell12345678.Unit1.Motor NOTE: The value of the first level of NamePath (cell12345678) exceeds 12 characters and is truncated into cell1234567. | No change | A new equipment is created: Cell123456700.Unit1.Motor |

Example 2: Updating Properties of Elements When Generating Facets

Scenario 1: The table shows the result when you generate again the same facet after you have modified some of the property values of a Supervision element that is referenced by the facet.

| Item | Already generated value | Value updated in the facet | Result after generating the facet |
|---|-------------------------|----------------------------|-----------------------------------|
| Facet identifier | Facet_1 | No change | Facet_1 |
| Equipment name of the equipment instance generated by the element or for the instance | Level1.Facet_1 | No change | Level1.Facet_1 |
| Element property 1 | A | Empty ⁽¹⁾ | Empty ⁽¹⁾ |
| Element property 2 | B | Null ⁽²⁾ | B |
| Element property 3 | C | C1 | C1 |
| Element property 4 | Empty ⁽¹⁾ | D | D |
| Element property 5 | Null ⁽²⁾ | E | E |

Scenario 2: The table shows the result when you generate, within the same cluster, another facet that references a Supervision element, which creates the same equipment name as an already generated facet, thus merging their properties.

| Item | Already generated value | Not yet generated value | Result after generating Facet_2 |
|--------------------------------------|-------------------------|-------------------------|--|
| Facet identifier | Facet_1 | Facet_2 | |
| Equipment name of equipment instance | Level1.Motor | Level1.Motor | Both equipment are merged into one equipment: Level1.Motor |
| Element property 1 | A | Empty ⁽¹⁾ | A |

| Item | Already generated value | Not yet generated value | Result after generating Facet_2 |
|--------------------|-------------------------|-------------------------|---------------------------------|
| Element property 2 | B | Null ⁽²⁾ | B |
| Element property 3 | C | C2 | C2 |
| Element property 4 | Empty ⁽¹⁾ | D2 | D2 |
| Element property 5 | Null ⁽²⁾ | E2 | E2 |

Scenario 3: The table shows the result when you generate, within the same cluster, two facets that reference a Supervision element, which create the same equipment name as an already generated facet, thus merging their properties. It is implied that Facet_3 is generated last.

| Item | Already generated value | Not yet generated value | | Result after generating Facet_3 |
|--------------------------------------|-------------------------|-------------------------|----------------------|--|
| Facet identifier | Facet_1 | Facet_2 | Facet_3 | – |
| Equipment name of equipment instance | Level1.Motor | Level1.Motor | Level1.Motor | The three equipment instances are merged into one (Level1.Motor) in the following order: <ol style="list-style-type: none"> 1. Equipment generated by Facet_2 is merged with the existing equipment. 2. Equipment generated by Facet_3 is merged with the previous equipment |
| Element property 1 | A | A2 | Empty ⁽¹⁾ | A2 |
| Element property 2 | B | B2 | Null ⁽²⁾ | B2 |
| Element property 3 | C | C2 | C3 | C3 |
| Element property 4 | Empty ⁽¹⁾ | D2 | D3 | D3 |
| Element property 5 | Null ⁽²⁾ | E2 | E3 | E3 |

(1) A value is considered null if, at the template level, the input pin of the element property is not connected.

(2) A value is considered empty if, at the template level, the input pin of the element property is connected (for example, to a parameter) but no value is provided through the binding.

Equipment Parameter Supervision Element

Overview

The **Equipment Parameter** element creates a runtime parameter entry in the equipment parameter table of the Supervision Participant project. The entry is associated to an equipment instance (see *EcoStruxure Process Expert, User Guide*).

You can select the element from the **Toolbox** pane, page 238 of the **Facet Editor** and reference it in **Data** or **Genie** Supervision facet templates.

You can reference several elements per facet template and use them in combination with **Equipment Group of Messages** elements, page 107.

Working Principle

The parameter values that are generated by the **Equipment Parameter** element are associated either to:

- The equipment instance that is generated for the instance by using the default method.
- The equipment instance that is generated by the **Equipment** element, page 96 that is located in the same facet template.

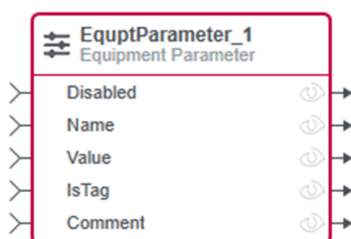
If several runtime parameter entries are created from different facets, which generate the same equipment instance (same equipment name) and these runtime parameters have the same name, then the parameter values coming from the facet that is generated last are used to populate the equipment parameter table. This applies also to runtime parameter entries that are generated by an **Equipment Group of Messages** element, page 107.

The rules that apply when the property of an **Equipment Parameter** element is empty or null, page 104 are the same as those that apply to the generation of **Equipment** elements.

To delete a runtime parameter entry from the equipment parameter table, delete or disable the element.

Element Representation

The following figure shows an **Equipment Parameter** element as it can appear in the Facet Editor, page 227.



The properties of the element are of type String except for **Disabled** and **Is Tag**, which are boolean.

| Property | Description |
|------------------|--|
| Disabled | When the boolean value provided to the input pin is true (enter <code>true</code> or 1), the element is disabled and does not generate a runtime parameter entry for the equipment instance. Blank or not connected is equal to false (0). |
| Name | The name for the runtime parameter. The software accepts only unique parameter names for a same equipment instance. The property must not be left blank and not exceed 254 characters; otherwise, generation does not complete successfully. |
| Value | The value for the generated parameter. Free-form text. The property can be left blank. For more information on the property, refer to <i>equipment runtime parameters</i> in the help of the Supervision Participant. |
| Other properties | Correspond to the properties of equipment runtime parameters in the Supervision Participant project and are used to populate the equipment parameter table. NOTE: The Project property is not managed by EcoStruxure Process Expert but by the Supervision Participant. |

Equipment Group of Messages Supervision Element

Overview

The **Equipment Group of Messages** element lets you create a set of runtime parameter entries in the equipment parameter table of the Supervision Participant project. The entries are associated to one equipment instance.

You can select the element from the **Toolbox** pane, page 238 of the **Facet Editor** and reference it in **Data** or **Genie** Supervision facet templates.

You can reference several elements per facet template and use them in combination with **Equipment Parameter** elements, page 105.

Working Principle

Each message property for which you configure a value creates one runtime parameter entry, page 105.

The values of the runtime parameter entries that are generated by the **Equipment Group of Messages** element are associated either to:

- The equipment instance that is generated for the instance by using the default method.
- The equipment instance that is generated by the **Equipment** element, page 96 that is located in the same facet template.

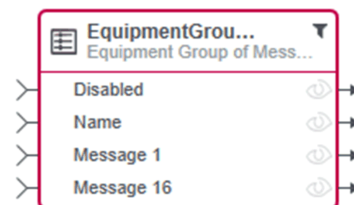
If several runtime parameter entries are created from different facets, page 103, which generate the same equipment instance (same equipment name) and these runtime parameters have the same name, then the parameter values coming from the facet that is generated last are used to populate the equipment parameter table. This applies also to runtime parameter entries that are generated by **Equipment Parameter** elements, page 105.

To delete a runtime parameter entry from the equipment parameter table, either delete or disable the element, or make the corresponding message property empty.

The rules that apply when the property of an **Equipment Group of Messages** element is empty or null, page 104 are the same as those that apply to the generation of **Equipment** elements.

Element Representation

The following figure shows an **Equipment Group of Messages** element as it can appear in the Facet Editor, page 227 with properties **Message 2** to **Message 15** hidden.



The properties of the element are of type String except for **Disabled**, which is boolean.

| Property | Description |
|-----------------|--|
| Disabled | When the boolean value provided to the input pin is true (enter <code>true</code> or 1), the element is disabled and does not generate runtime parameter entries for the equipment instance. Blank or not connected is equal to false (0). |
| Name | The fixed part of the name of each runtime parameter entry that is generated by the element. |

| Property | Description | |
|---|--|--|
| | Refer to Name below for a description of how the value is used. | |
| Message x where x is an integer index from 1 to 16. | For each property for which you enter a value, a runtime parameter entry is created in the equipment parameter table with the following values configured. | |
| | Parameter property | Value |
| | Name | <p>The value of Name concatenated with the _x suffix, where the value of x is the one of the Message x property.</p> <p>The software accepts only unique parameter names for a same equipment instance and the name must not exceed 254 characters.</p> <p>Example:</p> <p>If Name of the Equipment Group of Messages element is <i>MyEqptGrMsg1</i> and only the following two Message x properties are configured:</p> <p>Message 1= <i>Auto</i></p> <p>Message 4= <i>100</i></p> <p>Then, two parameters are created and named <i>MyEqptGrMsg1_1</i> and <i>MyEqptGrMsg1_4</i> respectively.</p> |
| | Value | <p>The value of the Message x property.</p> <p>Example:</p> <p>Continuing with the above example, the values of the two parameters are:</p> <ul style="list-style-type: none"> <i>MyEqptGrMsg1_1 = Auto</i> <i>MyEqptGrMsg1_4 = 100</i> <p>Refer to the description of the Value property for an equipment runtime parameter, page 106.</p> |
| | Is Tag | <i>False</i> |
| | Comment | Empty |

Binding Functions

Overview

This section describes the binding functions that you can use in bindings of facet and composite templates.

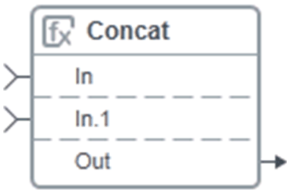
The functions are accessible in the **Toolbox** pane of the **Facet Editor** and **Composite Editor**.

To use a function, drag it to the workspace and make the necessary connections.

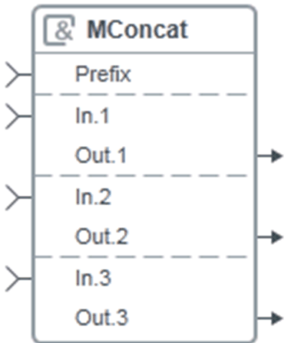
Concatenation Functions

Concatenation of Character Strings: Concat

NOTE: In some cases, you can replace a concatenation function by the *\$FullName* property. For example, to generate parameter names, page 45 of DFBs and DDTs.

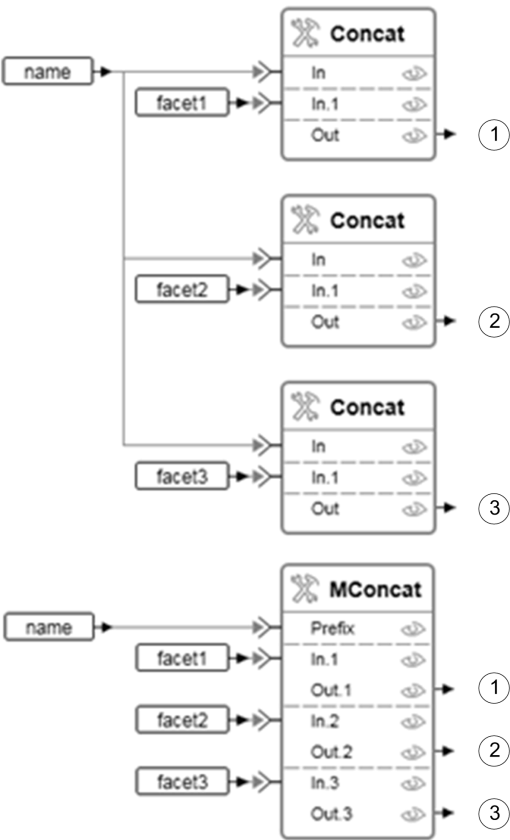
| Representation | Description | |
|---|---|-----------|
|  | <ul style="list-style-type: none">This function concatenates:<ul style="list-style-type: none">A minimum of two input strings.A maximum of eleven input strings when the function is extended.The result on the output of the function is the concatenation of inputs.The input with null value is ignored. The result on the output of the function is null if the inputs are null. | |
| | I/O name | Data type |
| | <i>In</i> | String |
| | <i>In.1 to In.X⁽¹⁾</i> | |
| | <i>Out</i> | |
| <p>(1) The inputs of this function can be extended by right-clicking the function, selecting the Extend Operation command, and entering <i>X</i>, which represents the number of inputs you want to use in addition to <i>In</i>. The valid range for <i>X</i> is 1 to 10.</p> | | |

Multiple Concatenations of Character Strings: MConcat

| Representation | Description | |
|---|--|-----------|
|  | <ul style="list-style-type: none">• This is an extended version of the Concat function allowing you to do multiple concatenations at once.• This function concatenates:<ul style="list-style-type: none">◦ A minimum of two input character strings.◦ A maximum of eleven-character input strings.• The result on the output of the function is the concatenation of <i>Prefix</i> with the related inputs.• The input with null value is ignored. The result on the output of the function is null if <i>Prefix</i> and related inputs are null. | |
| | I/O name | Data type |
| | <i>Prefix</i> | String |
| | <i>In.1 to In.X⁽¹⁾</i> | |
| | <i>Out.1 to Out.X⁽¹⁾</i> | |
| <p>(1) The inputs of this function can be extended by right-clicking the function, selecting the Extend Operation command, and entering <i>X</i>, which represents the number of inputs you want to use in addition to <i>In</i>. The valid range for <i>X</i> is 1 to 10.</p> | | |

Concat and MConcat Example

The following examples illustrate the use of both the `Concat` and the `MConcat` functions. Both functions perform the same operation, where the input *name* is concatenated with inputs *facet1*, *facet2* and *facet3* respectively.



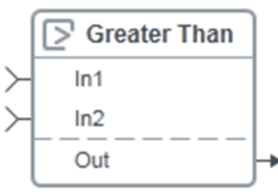
| Item | Description |
|------|---|
| 1 | The value of <i>Out</i> and <i>Out.1</i> is <i>namefacet1</i> . |
| 2 | The value of <i>Out</i> and <i>Out.2</i> is <i>namefacet2</i> . |
| 3 | The value of <i>Out</i> and <i>Out.3</i> is <i>namefacet3</i> . |

Comparison Functions

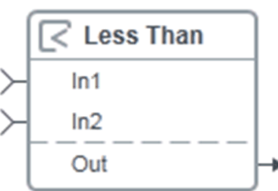
Equal to: Equal

| Representation | Description | |
|----------------|--|-----------|
| | <ul style="list-style-type: none">This function checks if the values of the In1 and In2 inputs are equal. If not, the output is set to false.The data type of the origin for the first input In1 determines the comparison type.The result on the output of the function is a true or false boolean result.The output of the function is false if at least one input is null. The output is true if the two inputs are null. | |
| | I/O name | Data type |
| | <i>In1</i> | Any |
| | <i>In2</i> | |
| | <i>Out</i> | |

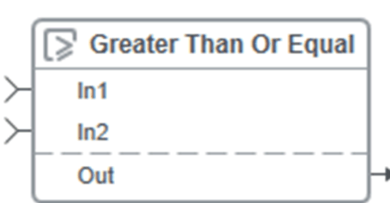
Greater Than

| Representation | Description | |
|---|--|-----------|
|  | <ul style="list-style-type: none"> This function checks if the value of the In1 input is greater than the value of the In2 input. In the affirmative, the output is set to true. The data type of the origin for the first input In1 determines the comparison type. The result on the output of the function is a true or false boolean result. The output of the function is false if at least one input is null. | |
| | I/O name | Data type |
| | <i>In1</i> | Any |
| | <i>In2</i> | |
| | <i>Out</i> | |

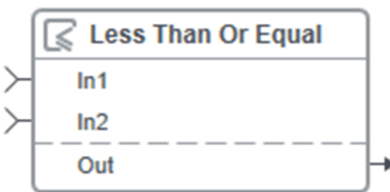
Less Than

| Representation | Description | |
|---|---|-----------|
|  | <ul style="list-style-type: none"> This function checks if the value of the In1 input is less than the value of the In2 input. In the affirmative, the output is set to true. The data type of the origin for the first input In1 determines the comparison type. The result on the output of the function is a true or false boolean result. The output of the function is false if at least one input is null. | |
| | I/O name | Data type |
| | <i>In1</i> | Any |
| | <i>In2</i> | |
| | <i>Out</i> | |

Greater Than Or Equal

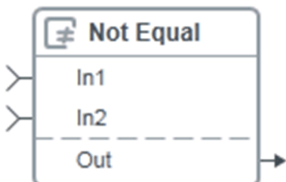
| Representation | Description | |
|---|--|-----------|
|  | <ul style="list-style-type: none"> This function checks if the value of the In1 input is greater than or equal to the value of the In2 input. In the affirmative, the output is set to true. The data type of the origin for the first input In1 determines the comparison type. The result on the output of the function is a true or false boolean result. The output of the function is false if at least one input is null. | |
| | I/O name | Data type |
| | <i>In1</i> | Any |
| | <i>In2</i> | |
| | <i>Out</i> | |

Less Than Or Equal

| Representation | Description | |
|---|---|-----------|
|  | <ul style="list-style-type: none"> This function checks if the value of the In1 input is less than or equal to the value of the In2 input. In the affirmative, the output is set to true. The data type of the origin for the first input In1 determines the comparison type. The result on the output of the function is a true or false boolean result. The output of the function is false if at least one input is null. | |
| | I/O name | Data type |
| | <i>In1</i> | Any |
| | <i>In2</i> | |

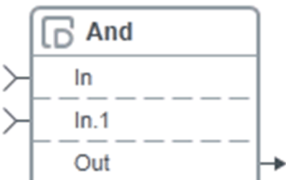
| Representation | Description | |
|----------------|-------------|--|
| | <i>Out</i> | |

Not Equal

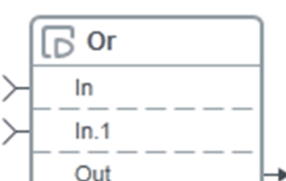
| Representation | Description | |
|---|---|-----------|
|  | <ul style="list-style-type: none"> This function checks whether the values of the In1 and In2 inputs are equal. If not, the output is set to true. The data type of the origin for the first input In1 determines the comparison type. The result on the output of the function is a true or false boolean result. The output of the function is true if at least one input is null. The output is false if the two inputs are null. | |
| | I/O name | Data type |
| | <i>In1</i> | Any |
| | <i>In2</i> | |
| | <i>Out</i> | |

Logical Functions

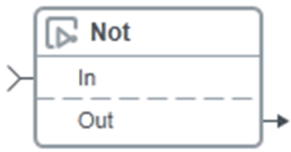
And

| Representation | Description | |
|---|--|-----------|
|  | <ul style="list-style-type: none">This function performs a logical <i>AND</i> operation on the <i>In</i> and <i>In.1</i> to <i>In.X⁽¹⁾</i> inputs and assigns the result to the output.The output of this function is null if the inputs are null. The inputs with null value are ignored. | |
| | I/O name | Data type |
| | <i>In</i> | Bool |
| | <i>In.1</i> to <i>In.X⁽¹⁾</i> | |
| | <i>Out</i> | |
| <p>(1) The inputs of this function can be extended by right-clicking the function, selecting the Extend Operation command, and entering <i>X</i>, which represents the number of inputs you want to use in addition to <i>In</i>. The valid range for <i>X</i> is 1 to 10.</p> | | |

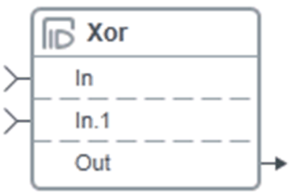
Or

| Representation | Description | |
|---|---|-----------|
|  | <ul style="list-style-type: none">This function performs a logical OR operation on the <i>In</i> and <i>In.1</i> to <i>In.X⁽¹⁾</i> inputs and assigns the result to the output.The output of this function is null if the inputs are null. The inputs with null value are ignored. | |
| | I/O name | Data type |
| | <i>In</i> | Bool |
| | <i>In.1</i> to <i>In.X⁽¹⁾</i> | |
| | <i>Out</i> | |
| <p>(1) The inputs of this function can be extended by right-clicking the function, selecting the Extend Operation command, and entering <i>X</i>, which represents the number of inputs you want to use in addition to <i>In</i>. The valid range for <i>X</i> is 1 to 10.</p> | | |

Not

| Representation | Description | |
|---|--|-----------|
|  | <ul style="list-style-type: none"> This function negates the In input and assigns the result to the output. The output of this function is true if the input is null. | |
| | I/O name | Data type |
| | <i>In</i> | Bool |
| | <i>Out</i> | |

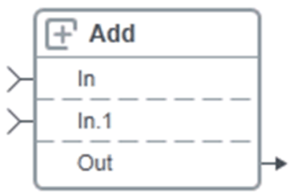
Exclusive OR: xor

| Representation | Description | |
|---|---|-----------|
|  | <ul style="list-style-type: none"> This function perform a logical <i>XOR</i> operation on the <i>In</i> and <i>In.1</i> to <i>In.X⁽¹⁾</i> inputs and assigns the result to the output. The output of this function is true if an input is null. The output is false if the inputs are null. | |
| | I/O name | Data type |
| | <i>In</i> | Bool |
| | <i>In.1</i> to <i>In.X⁽¹⁾</i> | |
| | <i>Out</i> | |

(1) The inputs of this function can be extended by right-clicking the function, selecting the **Extend Operation** command, and entering *X*, which represents the number of inputs you want to use in addition to *In*. The valid range for *X* is 1 to 10.

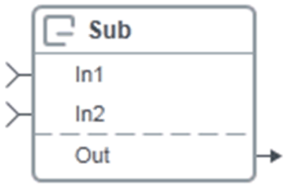
Mathematical Functions

Addition: Add

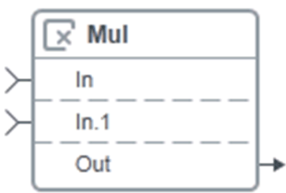
| Representation | Description | |
|---|--|-----------|
|  | <ul style="list-style-type: none"> This function adds up the values of the <i>In.1</i> to <i>In.X</i> inputs and assigns the result to the output. The output of this function is null if the inputs are null. Inputs with null value are ignored. | |
| | I/O name | Data type |
| | <i>In</i> | Real |
| | <i>In.1</i> to <i>In.X⁽¹⁾</i> | |
| | <i>Out</i> | |

(1) The inputs of this function can be extended by right-clicking the function, selecting the **Extend Operation** command, and entering *X*, which represents the number of inputs you want to use in addition to *In*. The valid range for *X* is 1 to 10.

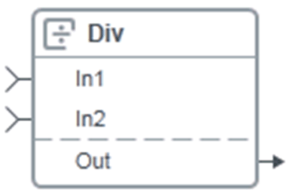
Subtraction: Sub

| Representation | Description | |
|---|--|-----------|
|  | <ul style="list-style-type: none"> This function subtracts the In2 input value from the value In1 input value and assigns the result to the output. The output of this function is null if either input is null. | |
| | I/O name | Data type |
| | <i>In1</i> | Real |
| | <i>In2</i> | |
| | <i>Out</i> | |

Multiplication: **Mu1**

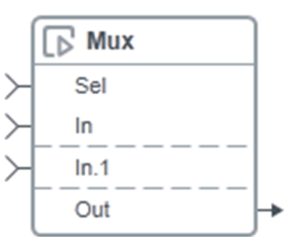
| Representation | Description | |
|---|--|-----------|
|  | <ul style="list-style-type: none">• This function multiplies the input values together and assigns the result to the output.• The output of this function is null if the inputs are null. Inputs with null value are ignored. | |
| | I/O name | Data type |
| | <i>In</i> | Real |
| | <i>In.1 to In.X⁽¹⁾</i> | |
| | <i>Out</i> | |
| <p>(1) The inputs of this function can be extended by right-clicking the function, selecting the Extend Operation command, and entering <i>X</i>, which represents the number of inputs you want to use in addition to <i>In</i>. The valid range for <i>X</i> is 1 to 10.</p> | | |

Division: **Div**

| Representation | Description | |
|---|--|-----------|
|  | <ul style="list-style-type: none">This function divides the value of the In1 input (the dividend) by the value of the In2 input (the divisor) and assigns the result to the output.The output of this function is:<ul style="list-style-type: none">Null if either input is null.Infinity if the divisor is equal to 0. | |
| | I/O name | Data type |
| | <i>In1</i> | Real |
| | <i>In2</i> | |
| | <i>Out</i> | |

Multiplexer Function

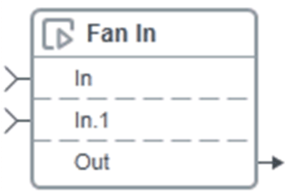
Multiplexer: **Mux**

| Representation | Description | |
|---|---|-----------|
|  | <ul style="list-style-type: none">This function transfers to the output the value of the input that is selected by using Sel (selection).To select In, set the value of Sel to 0, to select In.1 to 1, to select In.2 to 2, and so on.The output is equal to In if the value of Sel is null.The output of this function is null if:<ul style="list-style-type: none">The selected input is null.The value of Sel exceeds the number of defined inputs (for example, you are using inputs In, In.1, and In.2 and the value of Sel is 4). | |
| | I/O name | Data type |
| | <i>Sel</i> | Integer |
| | <i>In</i> | Any |
| | <i>In.1 to In.X⁽¹⁾</i> | |
| <i>Out</i> | | |

(1) The inputs of this function can be extended by right-clicking the function, selecting the **Extend Operation** command, and entering *X*, which represents the number of inputs you want to use in addition to *In*. The valid range for *X* is 1 to 10.

Fan-In Function

Fan In

| Representation | Description | |
|---|---|---------------------------------|
|  | <ul style="list-style-type: none"> This function returns the value that it has received first on either of its inputs. The value received needs to be not null. It prioritizes the value received from the first input that has been connected and resolved (for example, if the inputs are optional elements, the element providing the value first is the one used as output). It allows making several connections to a same input. The output of the function is null if the inputs are null. | |
| | I/O name | Data type |
| | <i>In</i> | Any, including interface links. |
| | <i>In.1 to In.X⁽¹⁾</i> | |
| | <i>Out</i> | Any |
| (1) The inputs of this function can be extended by right-clicking the function, selecting the Extend Operation command, and entering X, which represents the number of inputs you want to use in addition to <i>In</i> . The valid range for X is 1 to 10. | | |

CrDocument Function for Runtime Navigation Services

Overview

The **CrDocument** binding function lets you link one document to an application template or instance thereof. Once linked, you can open the document from an instance when using runtime navigation services on the computer running an operation client.

The document must be located in a content container and the application that is required to open the document needs to be installed on the computer running the operation client.

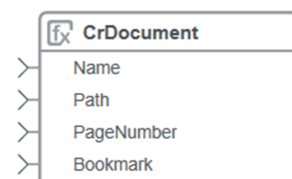
The function lets you configure where and how the document appears in the operation client.

You can also use the **CrDocument** function with topological templates.

NOTE: Control module templates of libraries that support the hyperlink functionality already reference one or more user-configurable **CrDocument** functions. Their parameters appear under the **HyperLink\Documents** element.

Description

The following figure shows the graphical representation of the binding function when used in a template editor.



| Attributes | Data type | Description |
|-------------|-----------|--|
| Name | String | <p>The attribute lets you:</p> <ul style="list-style-type: none"> Select in which section of the operation client a document is displayed depending on the path type, page 117 that you enter. The document is the one that is defined by using the Path attribute. Create a folder structure in the section in which to display the document by entering the corresponding path in the format: <level_1>\<level_2>\level_n where <level_n> represents the name that is shown in the section. For example, <i>Tank1\Levels</i>. Select a display name for the document to be shown instead of the file name. The last element of the path represents the display name. <p>For example, <i>Tank1\Levels\Level Sensor Maintenance</i>.</p> <p>It can be linked to either:</p> <ul style="list-style-type: none"> A platform input: The section, folder structure, and document name displayed in the operation client are defined at the template level and cannot be edited at the instance level. A parameter: Lets you select a default value during template edition for the location, folder structure, and document name. When the parameter is deferred, lets you customize the value while you edit the properties of an instance of the template. <p>NOTE: If the value of Name is null, the document does not appear in the operation client.</p> |
| Path | String | <p>Specifies the path to one document that is linked to instances of the template. The document must be located in a content container.</p> <p>It can be linked to either:</p> <ul style="list-style-type: none"> A platform input: The path is defined at the template level and cannot be edited at the instance level. A parameter: When the parameter format, page 86 is properly configured, lets you define one default document during template edition by browsing existing content containers (see <i>EcoStruxure Process Expert, Runtime Navigation Services, User Guide</i>). When the parameter is deferred, lets you browse other documents from existing content containers while you edit an instance of the template. <p>NOTE:</p> <ul style="list-style-type: none"> The path indicates from where a document is accessed: <ul style="list-style-type: none"> Path starting with crg:// for documents from content containers located in the Global Root\User Contents \... folder structure. When you browse documents during template edition, you can only access documents at this location. Path starting with crs:// for documents from content containers located in the Systems\User Contents \... folder structure. This location becomes accessible when you browse documents during instance edition. The type of the content container (.doc) appears in the path and should not be confused with the document file extension. If the document is located in two content containers with the same identifier at the same path but of different version, the content container with the latest version is automatically selected. |

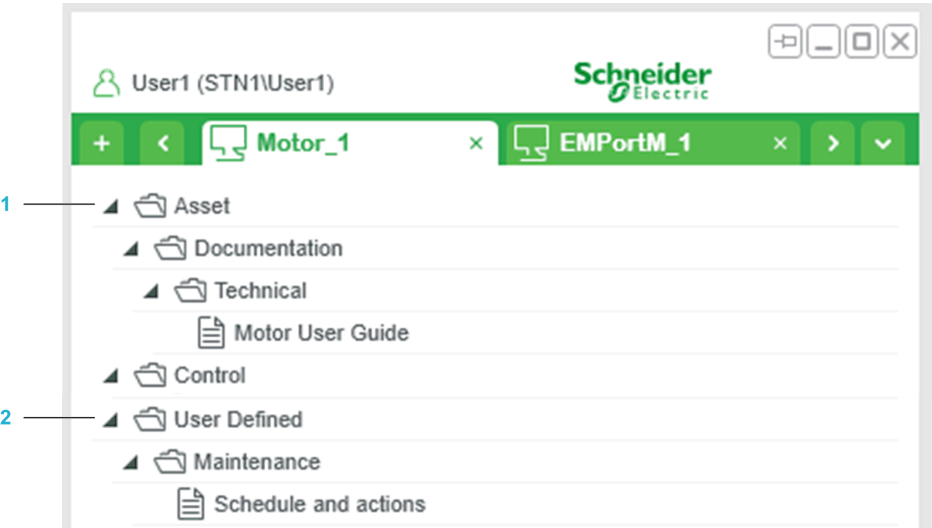
| Attributes | Data type | Description |
|-------------------|-----------|--|
| PageNumber | String | <p>The page where the document opens when accessed from the operation client.</p> <p>For example, 10.</p> <p>The value is used as parameter in the command line that needs to be specified for the corresponding file type in the Document Viewers section of the Settings window (see <i>EcoStruxure Process Expert, User Guide</i>).</p> |
| Bookmark | String | <p>The bookmark in a text document or topic ID in an HTML document where the document opens when accessed from the operation client.</p> <p>For example, <i>overview.htm</i>.</p> <p>The value is used as parameter in the command line that needs to be specified for the corresponding file type in the Document Viewers section of the Settings window (see <i>EcoStruxure Process Expert, User Guide</i>).</p> |

Configuration of the Name Attribute

The following table indicates in which section of the operation client a document, file, or link and its folder structure appear depending on the location of the **CrDocument** function or **Url** function, page 118 and the path type that you enter in the **Name** attribute.

| Function referenced by | Path type | Operation client section |
|---|--|--------------------------|
| Application composite template ⁽¹⁾ | Relative (starts without \) | Asset |
| Topological composite or facet template | For example, Folder_1\Folder_2 \MyDocumentDisplayName | Asset |
| Application Control facet template | | Control |
| Application Supervision facet template | | Supervision |
| Application composite or facet template | Absolute (starts with \) | User Defined |
| Topological composite or facet template | For example, \Folder_1\Folder_2 \MyDocumentDisplayName | |
| (1) This is from where the user-configurable CrDocument and URL functions are referenced for templates of the General Purpose library. It corresponds to the HyperLink\Documents and HyperLink\URL elements that you can see when editing instances by using the Instance Editor . | | |

The following figure shows a partial view of the operation client window when runtime navigation services are used. It illustrates the use of the **Name** parameter to display links to documents in sections of the window.



| Item | Description |
|------|---|
| 1 | Folder structure and link to document that is created when you enter in the Name parameter the path <i>Documentation\Technical\Motor User Guide</i> (relative path) for one of the default CrDocument functions referenced by a Schneider Electric template (HyperLink\Documents element in the Instance Editor). |
| 2 | Folder structure and link to document that is created when you enter in the Name attribute the path <i>Maintenance\Schedule and actions</i> (absolute path) for a CrDocument function referenced anywhere in a template. |

Url Function for Runtime Navigation Services

Overview

The **Url** binding function lets you associate one web link or one path to a web document or file to an application template or instance thereof. You can open the link, web document, or file from an instance when using runtime navigation services on the computer running an operation client. The link opens in the default browser of the computer.

The function lets configure where and how the web link or file appears in the operation client.

You can also use the **Url** function with topological templates.

NOTE: Control module templates of libraries that support the hyperlink functionality already reference one or more user-configurable **Url** functions. Their parameters appear under the **HyperLink\URL** element.

Description

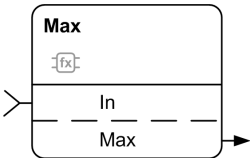
The following figure shows the graphical representation of the binding function when used in a template editor.



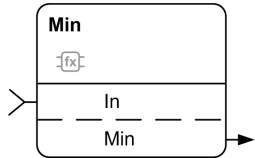
| Attributes | Data type | Description |
|-------------|-----------|---|
| Name | String | <p>The attribute lets you:</p> <ul style="list-style-type: none"> Select in which section of the operation client a web link/file is displayed depending on the path type, page 117 that you enter. The link/file is the one that is defined by using the Path attribute. Create a folder structure in the section in which to display the web link/file by entering the corresponding path in the format: <level_1>\<level_2>\<level_n> where <level_n> represents the name that is shown in the section. For example, Tank1\Levels. Select a display name for the web link/file to be shown instead of the URL. The last element of the path represents the display name. <p>For example, Tank1\Levels\Level Sensor Web Diagnostics.</p> <p>It can be linked to either:</p> <ul style="list-style-type: none"> A platform input: The section, folder structure, and web link/file name displayed in the operation client are defined at the template level and cannot be edited at the instance level. A parameter: Lets you select a default value during template edition for the location, folder structure, and web link/file name. When the parameter is deferred, lets you customize the value while you edit the properties of an instance of the template. <p>NOTE: If the value of Name is null, the web link/file does not appear in the operation client.</p> |
| Path | | <p>Specifies the path to one web link/file that is associated with instances of the template.</p> <p>If you associate it to a file, it must be available at the specified path on the computer running the operation client. You need to include the file extension. If several computers are used, the file location must be the same on each one.</p> <p>It can be linked to either:</p> <ul style="list-style-type: none"> A platform input: The path is defined at the template level and cannot be edited at the instance level. A parameter: When the parameter format, page 86 is properly configured, lets you browse and select a file during template edition. When the parameter is deferred, lets you enter a URL or path while you edit the properties of an instance of the template. |

Miscellaneous Functions

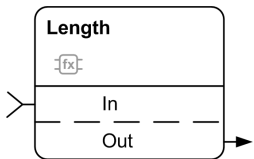
Maximum Value: **Max**

| Representation | Description | |
|---|---|-----------|
|  | <ul style="list-style-type: none">This function assigns the greatest input value to the output.The output of this function is null if the inputs are null. The inputs with null value are ignored. | |
| | I/O name | Data type |
| | <i>In</i> | Real |
| | <i>In. 1 to In.X⁽¹⁾</i> | |
| | <i>Out</i> | |
| <p>(1) The inputs of this function can be extended by right-clicking the function, selecting the Extend Operation command, and entering <i>X</i>, which represents the number of inputs you want to use in addition to <i>In</i>. The valid range for <i>X</i> is 1 to 10.</p> | | |

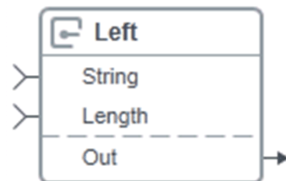
Minimum Value: **Min**

| Representation | Description | |
|---|---|-----------|
|  | <ul style="list-style-type: none">This function assigns the smallest input value to the output.The output of this function is null if the inputs are null. The inputs with null value are ignored. | |
| | I/O name | Data type |
| | <i>In</i> | Real |
| | <i>In.1 to In.X⁽¹⁾</i> | |
| | <i>Out</i> | |
| <p>(1) The inputs of this function can be extended by right-clicking the function, selecting the Extend Operation command, and entering <i>X</i>, which represents the number of inputs you want to use in addition to <i>In</i>. The valid range for <i>X</i> is 1 to 10.</p> | | |

Length of Character String: **Length**

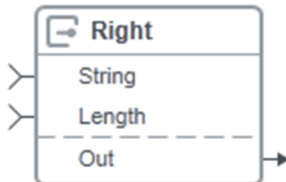
| Representation | Description | |
|---|--|-----------|
|  | <ul style="list-style-type: none"> This function assigns to the output the number of characters of the input. The output of this function is 0 if the input is null. | |
| | I/O name | Data type |
| | <i>In</i> | String |
| | <i>Out</i> | Integer |

Extraction of Characters to the Left: **Left**

| Representation | Description | |
|---|--|-----------|
|  | <ul style="list-style-type: none"> This function extracts the number of characters defined by Length starting from the leftmost character of the String input. The String input is treated as false if the input is null. If the value of Length exceeds the number of characters of the String input, the output returns only the characters of the String input. Length is treated as 0 if the input is null or negative. The output is an empty string if Length is 0. | |
| | I/O name | Data type |
| | <i>String</i> | String |
| | <i>Length</i> | Integer |
| | <i>Out</i> | String |

For example, the **Left** function used with *abcde* and 2 as inputs *String* and *Length* respectively, returns *ab* on the output.

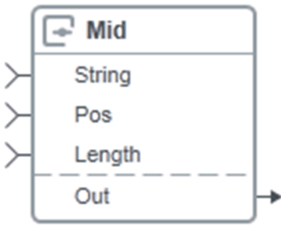
Extraction of Characters to the Right: **Right**

| Representation | Description | |
|---|---|-----------|
|  | <ul style="list-style-type: none"> This function extracts the number of characters defined by Length starting from the rightmost character of the String input. The String input is treated as false if the input is null. If the value of Length exceeds the number of characters of the String input, the output returns only the characters of the String input. Length is treated as 0 if the input is null or negative. The output is an empty string if Length is 0. | |
| | I/O name | Data type |
| | | |

| Representation | Description | |
|----------------|---------------|---------|
| | <i>String</i> | String |
| | <i>Length</i> | Integer |
| | <i>Out</i> | String |

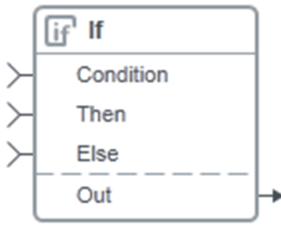
For example, the `Right` function used with `abcde` and `2` as inputs *String* and *Length* respectively, returns `de` on the output.

Extraction of a Substring of Characters: `Mid`

| Representation | Description | |
|---|---|-----------|
|  | <ul style="list-style-type: none"> This function extracts a substring of characters from the String input of length defined by Length and starting from a position defined by Pos. Pos starts counting from 0, which is the leftmost character of the String input. The String input is treated as "" if the input is null. If Pos exceeds the number of characters of the String input, the output is an empty string. Length and Pos are treated as 0 if their respective value is null or negative. The output is an empty string if Length or Pos is 0. | |
| | I/O name | Data type |
| | <i>String</i> | String |
| | <i>Pos</i> | Integer |
| | <i>Length</i> | |
| | <i>Out</i> | String |

For example, the `Mid` function used with `abcde`, `1`, and `2` as inputs *String*, *Pos*, and *Length* respectively, returns `bc` on the output.

Condition: `If`

| Representation | Description | |
|---|---|-----------|
|  | <p>This function returns a value depending on the evaluation of a condition:</p> <ul style="list-style-type: none"> Then: Is the value assigned to the output when Condition is true. Else: Is the value assigned to the output when Condition is false. <p>The Condition input is treated as false if the input is null. The output of this function is null if the selected input is null.</p> | |
| | I/O name | Data type |
| | <i>Condition</i> | Bool |
| | <i>Then</i> | Any |
| | <i>Else</i> | |
| | <i>Out</i> | |

LocationGet Instance Path Calculation Function

Overview

The **LocationGet** binding function lets you view information about the location of an instance in the application of the system.

You can view the information in the **Inspect Instance** window (see *EcoStruxure Process Expert, User Guide*) without linking an output of the function. You need to

reference the function at the level of your choice in a template, except in interfaces.

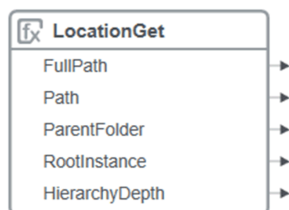
You can use the function to provide instance path data to the **LocationExtract** function, page 123.

Guidelines

Avoid using more than one **LocationGet** binding function per template because they produce the same results independently of their location in the composition of the template.

Description

The following figure shows the graphical representation of the binding function when used in a template editor.



The function has no user-configurable inputs.

| Attributes | Data type | Description |
|------------------------|---------------|--|
| FullPath | <i>String</i> | <p>Indicates the full path of the instance in the application of the system.</p> <p>The path ends with the InstanceID instance identifier, page 86.</p> <p>For example, <i>Folder_1\Folder_2\MyInstance</i>.</p> <p>NOTE:</p> <ul style="list-style-type: none"> The path ends with the InstanceID of the instance (for example, <i>Motor_1</i>) even if the function is referenced by an element of the instance, such as, a composite, a Control, or a Supervision facet template (for example, <i>Logic</i> or <i>InterlockTags</i> of <i>Motor_1</i>). The system root folder is not included in the path (for example, <i>System_1</i>). Path information is updated when the instance identifier (this includes using the folder alias) or the location of the instance changes. You can link the output to the Path input of the LocationExtract binding function, page 123. |
| Path | | <p>Indicates the path of the instance in the application of the system.</p> <p>For example, <i>Folder_1\Folder_2</i>.</p> <p>It does not include:</p> <ul style="list-style-type: none"> The instance identifier. The system root folder. |
| ParentFolder | | <p>Folder that contains the instance specified by FullPath.</p> <p>For example, <i>Folder_2</i>.</p> |
| RootInstance | | <p>InstanceID of the instance specified by FullPath.</p> <p>For example, <i>MyInstance</i>.</p> |
| Hierarchy-Depth | <i>UInt</i> | <p>Indicates the number of folder levels in the path of the instance without the system root folder.</p> <p>For example, 2 for an instance located at the path <i>Folder_1\Folder_2</i>.</p> |

LocationExtract Instance Path Extraction Function

Overview

The **LocationExtract** binding function lets you view information about the location of an instance in the application of the system.

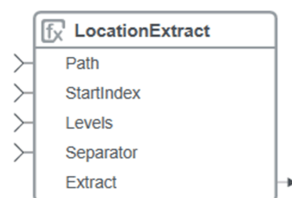
You can view the information directly in the **Inspect Instance** window (see *EcoStruxure Process Expert, User Guide*) without linking the output.

Using LocationExtract

- You can reference as many functions as required at various levels of a template composition (except in interfaces).
- To be able to adjust the extracted data at the instance level (**Instance Editor**), link parameters to the inputs and defer them at the highest level template.
- You can link the output to an interface, for example, to propagate complete path information of an instance (this is, its location in the application) or parts of it to a Supervision facet template. In such case, renaming or moving the instance sets generated facets that are interfaced to **Out Of Date**.
- Use this function with the **LocationGet** function, page 121 to provide path input data.

Description

The following figure shows the graphical representation of the binding function when used in a template editor.

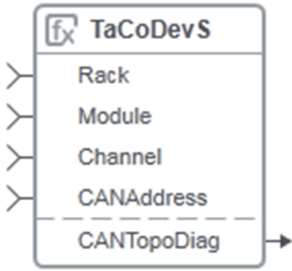


| Attributes | Data type | Description |
|-------------------|-----------|--|
| Path | String | <p>A string containing elements separated by \ (default separator for folder paths) or one of the supported separators (refer to the description of Separator). Each separator creates a level.</p> <p>Link the Fullpath output of the LocationGet function to this input.</p> <p>For example, <i>Folder_1\Folder_2\MyInstance</i> where MyInstance is the InstanceID instance identifier, page 86. The system root folder is not included in the path.</p> <p>NOTE: If the input is empty, the output of the function is null.</p> |
| StartIndex | UInt | <p>Specifies at which level of the path (Path) the extraction begins.</p> <p>0 corresponds to the first level (for example, it corresponds to Folder_1 for Path equal to <i>Folder_1\Folder_2</i>). 1 corresponds to the next level (for example, Folder_2), and so on.</p> <p>If you link Path to FullPath of the LocationGet function, 0 corresponds to the first folder level after the system root folder.</p> <p>NOTE:</p> <ul style="list-style-type: none"> The default value is 0. Negative values are treated as 0. If the value is higher than the number of levels in Path, the output of the function is null. |

| Attributes | Data type | Description |
|------------------|-----------|--|
| | | <ul style="list-style-type: none"> You can specify an index value that corresponds to the instance identifier contained in Path (if linked to FullPath). For example, 2 corresponds to MyInstance when Path is equal to <i>Folder_1\Folder_2\MyInstance</i>. |
| Levels | UInt | <p>Specifies the number of levels contained in the extracted path starting at the level specified by StartIndex.</p> <p>For example, if Path is equal to <i>Folder_1\Folder_2\MyInstance</i> and StartIndex is set to 0, a value of 2 for Levels corresponds to levels Folder_1 and Folder_2.</p> <p>NOTE:</p> <ul style="list-style-type: none"> If the value is negative or equal to 0, the output of the function is null. If the value is greater than the number of levels contained in Path, all the levels of Path are used (starting from StartIndex). |
| Separator | String | <p>Specifies the character to be used as separator for the output instead of the one used in Path.</p> <p>Supported characters are (.), (\), (%), (#), (-), (\$), (,), (;), (:), (^), (!), (&), and (@).</p> <p>(Enter only the character to be used as separator.)</p> <p>For example, entering % outputs <i>Folder_1%Folder_2</i> instead of <i>Folder_1\Folder_2</i>.</p> <p>NOTE: The use of this field is optional. Leaving it empty or entering a non-supported character uses the same separator as it appears in Path.</p> |
| Extract | String | <p>Outputs the extracted path based on the configuration of the input attributes of the function.</p> <p>Example:</p> <p>If path is equal to <i>Folder_1\Folder_2\MyInstance</i>,</p> <p>StartIndex is set to 1 and Levels to 2,</p> <p>Separator is equal to %,</p> <p>Then, Extract is equal to <i>Folder_2%MyInstance</i>.</p> |

Topological Address Calculation Functions

CANopen Device Status Address: TaCoDevS

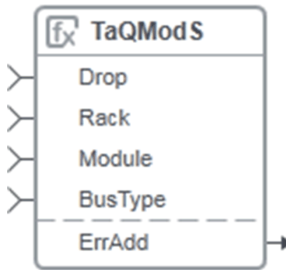
| Representation | Function Description | | |
|---|---|----------------------|---|
|  | This function provides the topological address of the device diagnostic word. | | |
| | I/O name | I/O data type | Description |
| | <i>Rack</i> | Integer | Address of the rack of the CANopen master (for example, 0) |
| | <i>Module</i> | | Address of the module of the CANopen master. |
| | <i>Channel</i> | | Channel of the CANopen master |
| | <i>CANAddress</i> | | Address of the device |
| | <i>CANTopoDiag</i> | String | Topological address of the CANopen device diagnostic word (%IW, refer to the description of the function algorithm) |

TaCoDevS function algorithm:

- $CANTopoDiag = \%IW.Rack.Module.Channel.Slave_Active$
Where $Slave_Active = Value1.Value2$

- Value for *Value1*:
 - If $1 \leq \text{CANAddress} \leq 16$ then *Value1* = 35
 - If $17 \leq \text{CANAddress} \leq 32$ then *Value1* = 36
 - If $33 \leq \text{CANAddress} \leq 48$ then *Value1* = 37
 - If $49 \leq \text{CANAddress} \leq 63$ then *Value1* = 38
- Value for *Value2*: $(\text{CANAddress} - 1) \text{ Mod } 16$

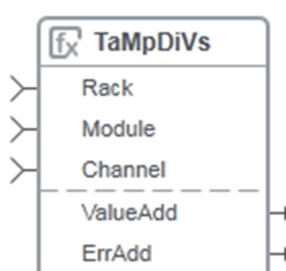
Quantum Module Status Address: TaQModS

| Representation | Function Description | | |
|---|--|---------------|--|
|  | This function provides the topological address of the module status system word. | | |
| | I/O name | I/O data type | Description |
| | <i>Drop</i> | Integer | Address of the drop (for example, 1) |
| | <i>Rack</i> | | Address of the rack in the drop (for example, 1) |
| | <i>Module</i> | | Address of the module on the rack (for example, 4) |
| | <i>BusType</i> | String | Defines the bus type that is used. |
| | <i>ErrAdd</i> | | Topological address of the module status system word (%SW, refer to the description of the function algorithm) |
| Function algorithm: | | | |

TaQModS function algorithm:

- $\text{ErrAdd} = \text{Concat}(\text{'\%SW'; Station; '.'; Position})$
- Where *Station* depends on *BusType* as follows:
 - If *BusType* = *LocalQuantumBus* then $\text{Station} = (\text{Drop} - 1) \times 5 + 180 + (\text{Rack} - 1)$
 - If *BusType* = *RioBus* then $\text{Station} = (\text{Drop} - 1) \times 5 + 185 + (\text{Rack} - 1)$
 - If *BusType* = *ERioBus* then $\text{Station} = 641 + (\text{Drop} - 1) \times 2 + (\text{Rack} - 1)$
- Where *Position* = $16 - \text{Module}$

M340/M580 Digital Input Channel Value and Status Addresses: TaMpDiVs

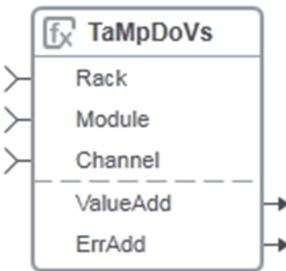
| Representation | Function Description | | |
|---|---|---------------|--|
|  | This function provides the topological address of the signal value and channel status bits for the channel of a digital input module. | | |
| | I/O name | I/O data type | Description |
| | <i>Rack</i> | Integer | Address of the rack in the drop (for example, 1) |
| | <i>Module</i> | | Address of the module on the rack (for example, 4) |
| | <i>Channel</i> | | Number of the channel of the module (for example, 0) |
| | <i>ValueAdd</i> | String | Topological address of the digital input signal value bit (%I, refer to the description of the function algorithm) |
| | <i>ErrAdd</i> | | Topological address of the channel status bit (%I, refer to the description of the function algorithm) |

TaMpDiVs function algorithm:

- $\text{ValueAdd} = \text{Concat}(\text{'\%I'; Rack; '.'; Module; '.'; Channel})$

- $ErrAdd = Concat(' \%I'; Rack; ' .'; Module; ' .'; Channel; ' .ERR')$

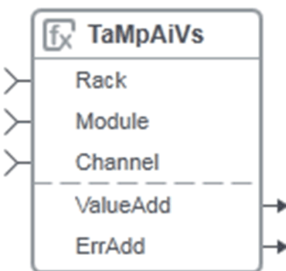
M340/M580 Digital Output Channel Value and Status Addresses: TaMpDoVs

| Representation | Function Description | | |
|---|--|---------------|---|
|  | This function provides the topological address of the signal value and channel status bits for the channel of a digital output module. | | |
| | I/O name | I/O data type | Description |
| | <i>Rack</i> | Integer | Address of the rack in the drop (for example, 1) |
| | <i>Module</i> | | Address of the module on the rack (for example, 4) |
| | <i>Channel</i> | | Number of the channel of the module (for example, 0) |
| | <i>ValueAdd</i> | String | Topological address of the digital output signal value bit (%Q, refer to the description of the function algorithm) |
| | <i>ErrAdd</i> | | Topological address of the channel status bit (%I, refer to the description of the function algorithm) |

TaMpDoVs function algorithm:

- $ValueAdd = Concat(' \%Q'; Rack; ' .'; Module; ' .'; Channel)$
- $ErrAdd = Concat(' \%I'; Rack; ' .'; Module; ' .'; Channel; ' .ERR')$

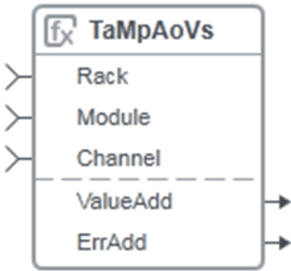
M340/M580 Analog Input Channel Value and Status Addresses: TaMpAiVs

| Representation | Function Description | | |
|---|---|---------------|--|
|  | This function provides the topological address of the signal value word and channel status bit for the channel of an analog input module. | | |
| | I/O name | I/O data type | Description |
| | <i>Rack</i> | Integer | Address of the rack in the drop (for example, 1) |
| | <i>Module</i> | | Address of the module on the rack (for example, 4) |
| | <i>Channel</i> | | Number of the channel of the module (for example, 0) |
| | <i>ValueAdd</i> | String | Topological address of the analog input signal value word (%IW, refer to the function algorithm) |
| | <i>ErrAdd</i> | | Topological address of the channel status bit (%I, refer to the function algorithm) |

TaMpAiVs function algorithm:

- $ValueAdd = Concat(' \%IW'; Rack; ' .'; Module; ' .'; Channel)$
- $ErrAdd = Concat(' \%I'; Rack; ' .'; Module; ' .'; Channel; ' .ERR')$

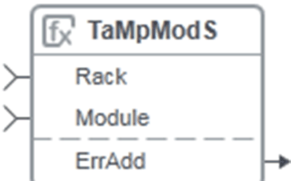
M340/M580 Analog Output Channel Value and Status Addresses: TaMpAoVs

| Representation | Function Description | | |
|---|--|---------------|---|
|  | This function provides the topological address of the signal value word and channel status bit for the channel of an analog output module. | | |
| | I/O name | I/O data type | Description |
| | <i>Rack</i> | Integer | Address of the rack in the drop (for example, 1) |
| | <i>Module</i> | | Address of the module on the rack (for example, 4) |
| | <i>Channel</i> | | Number of the channel of the module (for example, 0) |
| | <i>ValueAdd</i> | String | Topological address of the analog output signal value word (%QW, refer to the description of the function algorithm)) |
| | <i>ErrAdd</i> | | Topological address of the channel status bit (%I, refer to the description of the function algorithm) |

TaMpAoVs function algorithm:

- *ValueAdd* = Concat('%QW';*Rack*;'.'; *Module*;'.'; *Channel*)
- *ErrAdd* = Concat('%I';*Rack*;'.'; *Module*;'.'; *Channel*;'.'ERR')

M340/M580 Analog Module Status Address: TaMpModS

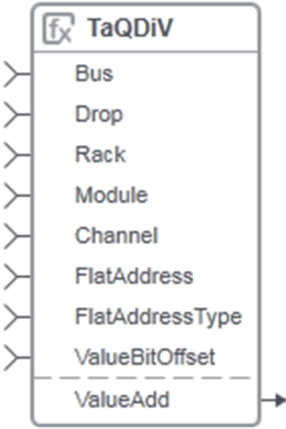
| Representation | Function Description | | |
|---|--|---------------|--|
|  | This function provides the topological address of the status bit for an analog module. | | |
| | I/O name | I/O data type | Description |
| | <i>Rack</i> | Integer | Address of the rack in the drop (for example, 1) |
| | <i>Module</i> | | Address of the module on the rack (for example, 4) |
| | <i>ErrAdd</i> | String | Topological address of the analog module status bit (%I, refer to the description of the function algorithm) |

TaMpModS function algorithm:

- *ErrAdd* = Concat('%I';*Rack*;'.'; *Module*;'.'; *MOD*;'.'ERR')

Other Address Calculation Functions

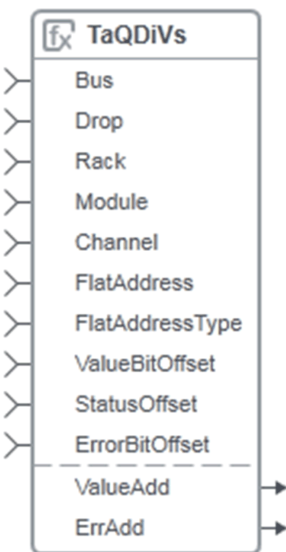
Quantum Digital Input Channel Value Address: **TaQDiV**

| Representation | Function Description | | |
|---|---|---------------|---|
|  | This function provides the topological address of the signal value bit for the channel of a digital input module. | | |
| | I/O name | I/O data type | Description |
| | <i>Bus</i> | Integer | Bus of the rack that contains the signal |
| | <i>Drop</i> | | Address of the drop (for example, 1) |
| | <i>Rack</i> | | Address of the rack in the drop (for example, 1) |
| | <i>Module</i> | | Address of the module on the rack (for example, 4) |
| | <i>Channel</i> | | Number of the channel of the module (for example, 0) |
| | <i>FlatAddress</i> | | Base state RAM address of the value (for example, 100) |
| | <i>FlatAddressType</i> | String | Prefix of the address to define the addressing type to be used (%I or %IW) |
| | <i>ValueBitOffset</i> | | Specific bit in the word for the signal value if using word addressing and value different from null |
| | <i>ValueAdd</i> | | Topological address of the signal value bit (%I or %IW, refer to the description of the corresponding function algorithm) |

TaQDiV function algorithm:

- For state RAM bit addressing:
 $ValueAdd = Concat(FlatAddressType, x)$
 Where $x = FlatAddress + Channel - 1$
- For state RAM word extracted bit addressing:
 $ValueAdd = Concat(FlatAddressType, x.y)$
 Where $x = FlatAddress + ((Channel - 1)/16)$ and $y = (ValueBitOffset == null)? 15 - MOD(Channel - 1, 16) ValueBitOffset$

Quantum Digital Input Channel Value and Status Addresses: TaQDiVs

| Representation | Function Description | | |
|---|---|---------------|---|
|  | This function provides the address of the signal value and channel status bits for the channel of a digital input module. | | |
| | I/O name | I/O data type | Description |
| | <i>Bus</i> | Integer | Bus of the rack that contains the signal |
| | <i>Drop</i> | | Address of the drop (for example, 1) |
| | <i>Rack</i> | | Address of the rack in the drop (for example, 1) |
| | <i>Module</i> | | Address of the module on the rack (for example, 4) |
| | <i>Channel</i> | | Number of the channel of the module (for example, 0) |
| | <i>FlatAddress</i> | | Base state RAM address of the value (for example, 100) |
| | <i>FlatAddressType</i> | String | Prefix of the address to define the direct addressing type to be used (%I or %IW) |
| | <i>StatusOffset</i> | Integer | Offset (number of words) to reach the status word |
| | <i>ValueBitOffset</i> | String | Specific bit in the word for the signal value if using word addressing and value different from null |
| | <i>ErrorBitOffset</i> | | Specific bit in the word for the status signal if using word addressing and value different from null |
| | <i>ValueAdd</i> | | Topological address of the signal value bit (%I or %IW, refer to the description of the corresponding function algorithm) |
| | <i>ErrAdd</i> | | Topological address of the channel status bit (%I or %IW, refer to the description of the corresponding function algorithm) |

TaQDiVs function algorithm:

- For state RAM bit addressing:

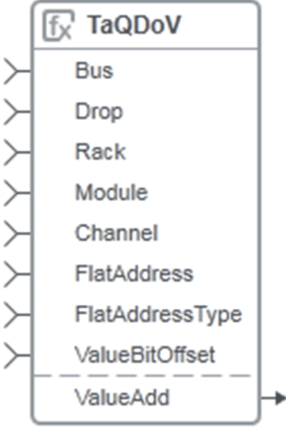
$$ValueAdd = \text{Concat}(FlatAddressType, x)$$
 Where $x = FlatAddress + Channel - 1$

$$ErrAdd = \text{Concat}(FlatAddressType, x)$$
 Where $x = FlatAddress + Channel - 1$
- For state RAM word extracted bit addressing:

$$ValueAdd = \text{Concat}(FlatAddressType, x.y)$$
 Where $x = FlatAddress + ((Channel - 1)/16)$ and $y = (ValueBitOffset == null)? 15 - \text{MOD}(Channel - 1, 16) : ValueBitOffset$

$$ErrAdd = \text{Concat}(FlatAddressType, x.y)$$
 Where $x = FlatAddress + ((StatusOffset + Channel - 1)/16)$ and $y = (ErrorBitOffset == null)? 15 - \text{MOD}(Channel - 1, 16) : ErrorBitOffset$

Quantum Digital Output Channel Value Address: TaQDoV

| Representation | Function Description | | |
|---|--|---------------|---|
|  | This function provides the address of the signal value bit for the channel of a digital output module. | | |
| | I/O name | I/O data type | Description |
| | <i>Bus</i> | Integer | Number of the bus (for example, 1) |
| | <i>Drop</i> | | Address of the drop (for example, 1) |
| | <i>Rack</i> | | Address of the rack in the drop (for example, 1) |
| | <i>Module</i> | | Address of the module on the rack (for example, 4) |
| | <i>Channel</i> | | Number of the channel of the module (for example, 0) |
| | <i>FlatAddress</i> | | Base state RAM address of the value (for example, 100) |
| | <i>FlatAddressType</i> | String | Prefix of the address to define the direct addressing type to be used (%M or %MW) |
| | <i>ValueBitOffset</i> | | Specific bit in the word for the value if using word addressing and value different from null |
| | <i>ValueAdd</i> | | Topological address of the signal value bit (%M or %MW, refer to the description of the corresponding function algorithm) |
| | | | |

TaQDoV function algorithm:

- For state RAM bit addressing:

$ValueAdd = Concat(FlatAddressType, x)$

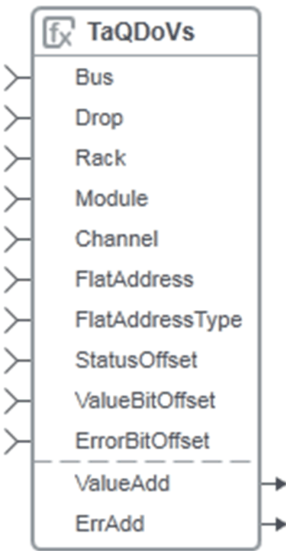
Where $x = FlatAddress + Channel - 1$

- For state RAM word extracted bit addressing:

$ValueAdd = Concat(FlatAddressType, x.y)$

Where $x = FlatAddress + ((Channel - 1)/16)$ and $y = (ValueBitOffset == null)? 15-MOD(Channel - 1, 16) ValueBitOffset$

Quantum Digital Output Channel Value and Status Addresses: TaQDoVs

| Representation | Function Description | | |
|---|--|---------------|---|
|  | This function provides the address of the signal value and channel status bits for the channel of a digital output module. | | |
| | I/O name | I/O data type | Description |
| | <i>Bus</i> | Integer | Number of the bus (for example, 1) |
| | <i>Drop</i> | | Address of the drop (for example, 1) |
| | <i>Rack</i> | | Address of the rack in the drop (for example, 1) |
| | <i>Module</i> | | Address of the module on the rack (for example, 4) |
| | <i>Channel</i> | | Number of the channel of the module (for example, 0) |
| | <i>FlatAddress</i> | | Base state RAM address of the value (for example, 100) |
| | <i>FlatAddressType</i> | String | Prefix of the address to define the direct addressing type to be used (%M or %MW) |
| | <i>StatusOffset</i> | Integer | Offset (number of words) to reach the status word |
| | <i>ValueBitOffset</i> | String | Specific bit in the word for the value if using word addressing and value different from null |
| | | | |

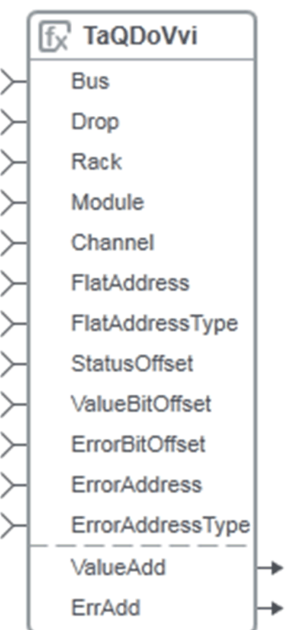
| Representation | Function Description | | |
|----------------|-----------------------|--|---|
| | <i>ErrorBitOffset</i> | | Specific bit in the word for the status if using word addressing and value different from null |
| | <i>ValueAdd</i> | | Topological address of the signal value bit (%M or %MW, refer to the description of the corresponding function algorithm) |
| | <i>ErrAdd</i> | | Topological address of the channel status bit (%M or %MW, refer to the description of the corresponding function algorithm) |

TaQDoVs function algorithm:

- For state RAM bit addressing:
 $ValueAdd = Concat(FlatAddressType, x)$
Where $x = FlatAddress + Channel - 1$
 $ErrAdd = Concat(FlatAddressType, x)$
Where $x = FlatAddress + Channel - 1$
- For state RAM word extracted bit addressing:
 $ValueAdd = Concat(FlatAddressType, x.y)$
Where $x = FlatAddress + ((Channel - 1)/16)$ and $y = (ValueBitOffset == null)? 15-MOD(Channel - 1, 16) : ValueBitOffset$
 $ErrAdd = Concat(FlatAddressType, x.y)$
Where $x = FlatAddress + ((StatusOffset + Channel - 1)/16)$ and $y = (ErrorBitOffset == null)? 15-MOD(Channel - 1, 16) : ErrorBitOffset$

Quantum Digital Output Channel Value and Status Addresses: *TaQDoVvi*

Quantum digital output signal value address (%M or %MW) for modules having value and status data in separated memory areas (applies to module references 140DVO85300 and 140DDM69000):

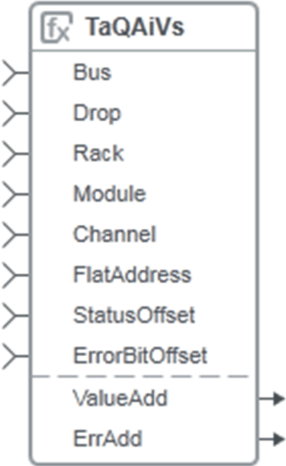
| Representation | Function Description | | |
|---|---|-----------|--|
|  | This function provides the address of the signal value and channel status for the channel of a digital output module. | | |
| | I/O name | Data type | Description |
| | <i>Bus</i> | Integer | Number of the bus (for example, 1) |
| | <i>Drop</i> | | Address of the drop (for example, 1) |
| | <i>Rack</i> | | Address of the rack in the drop (for example, 1) |
| | <i>Module</i> | | Address of the module on the rack (for example, 4) |
| | <i>Channel</i> | | Number of the channel of the module (for example, 0) |
| | <i>FlatAddress</i> | | Base state RAM address of the value (for example, 100) |
| | <i>FlatAddressType</i> | String | Prefix of the address to define the direct addressing type to be used. Either (%M or %MW) |
| | <i>StatusOffset</i> | Integer | Offset (number of words) to reach the status word |
| | <i>ValueBitOffset</i> | String | Specific bit in the word for the value if using word addressing and value different from null |
| | <i>ErrorBitOffset</i> | | Specific bit in the word for the value if using word addressing and value different from null |
| | <i>ErrorAddress</i> | Integer | Base state RAM address for status if using word addressing and value different from null (for example, 10) |
| | <i>ErrorAddressType</i> | String | Prefix of the address to define the direct addressing type to be used for the channel status. Either (%M or %MW) |

| Representation | Function Description | | |
|----------------|----------------------|--|--|
| | <i>ValueAdd</i> | | Topological address of the digital output signal value bit (%M or %MW, refer to the description of the corresponding function algorithm) |
| | <i>ErrAdd</i> | | Topological address of the channel status bit (%M or %MW, refer to the description of the corresponding function algorithm) |

TaQDoVvi function algorithm:

- For state RAM bit addressing:
 $ValueAdd = Concat(FlatAddressType, x)$
Where $x = FlatAddress + Channel - 1$
 $ErrAdd = Concat(FlatAddressType, x)$
Where $x = FlatAddress + Channel - 1$
- For state RAM word extracted bit addressing:
 $ValueAdd = Concat(FlatAddressType, x.y)$
Where $x = FlatAddress + ((Channel - 1)/16)$ and $y = (ValueBitOffset == null)? 15-MOD(Channel - 1, 16) : ValueBitOffset$
 $ErrAdd = Concat(FlatAddressType, x.y)$
Where $x = FlatAddress + ((StatusOffset + Channel - 1)/16)$ and $y = (ErrorBitOffset == null)? 15-MOD(Channel - 1, 16) : ErrorBitOffset$

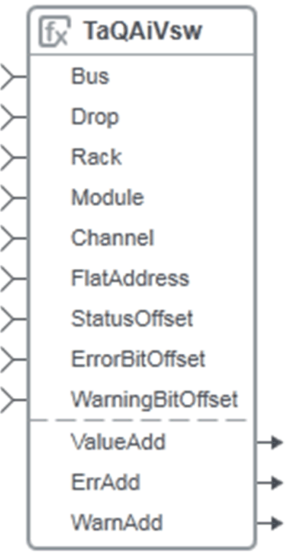
Quantum Analog Input Channel Value and Status Addresses: *TaQAiVs*

| Representation | Function Description | | |
|---|---|---------------|---|
|  | This function provides the address of the signal value word and channel status bit for the channel of an analog input module. | | |
| | I/O name | I/O data type | Description |
| | <i>Bus</i> | Integer | Number of the bus (for example, 1) |
| | <i>Drop</i> | | Address of the drop (for example, 1) |
| | <i>Rack</i> | | Address of the rack in the drop (for example, 1) |
| | <i>Module</i> | | Address of the module on the rack (for example, 4) |
| | <i>Channel</i> | | Number of the channel of the module (for example, 0) |
| | <i>FlatAddress</i> | | Base state RAM address of the value (for example, 100) |
| | <i>StatusOffset</i> | | Offset (number of words) to reach the status word |
| | <i>ErrorBitOffset</i> | String | Specific bit in the word for the status if using word addressing and value different from null |
| | <i>ValueAdd</i> | | Topological address of the analog input signal word (%IW, refer to the description of the function algorithm) |
| | <i>ErrAdd</i> | | Topological address of the channel status bit (%IW, refer to the description of the function algorithm) |

TaQAiVs function algorithm:

- $ValueAdd = \%IWx$
Where $x = FlatAddress + Channel - 1$
- $ErrAdd = \%IWx.y$
Where $x = FlatAddress + StatusOffset$ and $y = ErrorBitOffset$

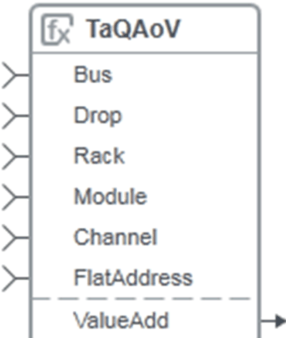
Quantum Analog Input Channel Value, Status, and Alert Addresses: TaQAiVsw

| Representation | Function Description | | |
|---|---|---------------|---|
|  | This function provides the topological address of the signal value word as well as the channel status and alert bits for the channel of an analog input module. | | |
| | I/O name | I/O data type | Description |
| | <i>Bus</i> | Integer | Number of the bus (for example, 1) |
| | <i>Drop</i> | | Address of the drop (for example, 1) |
| | <i>Rack</i> | | Address of the rack in the drop (for example, 1) |
| | <i>Module</i> | | Address of the module on the rack (for example, 4) |
| | <i>Channel</i> | | Number of the channel of the module (for example, 0) |
| | <i>FlatAddress</i> | | Base state RAM address of the value (for example, 10) |
| | <i>StatusOffset</i> | | Offset (number of words) to reach the status word |
| | <i>ErrorBitOffset</i> | String | Specific bit in the word for the status |
| | <i>WarningBitOffset</i> | | Specific bit in the word for the alert |
| | <i>ValueAdd</i> | | Topological address of the analog input signal value word (%IW, refer to the description of the function algorithm) |
| | <i>ErrAdd</i> | | Topological address of the channel status bit (%IW, refer to the description of the function algorithm) |
| | <i>WarnAdd</i> | | Topological address of the channel alert bit (%IW, refer to the description of the function algorithm) |

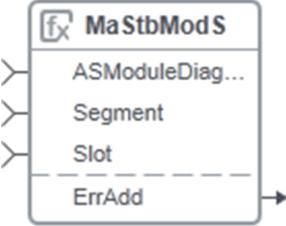
TaQAiVsw function algorithm:

- $ValueAdd = \%IWx$
Where $x = FlatAddress + Channel - 1$
- $ErrAdd = \%IWx.y$
Where $x = FlatAddress + StatusOffset$ and $y = ErrorBitOffset$
- $WarnAdd = \%IWx.y$
Where $x = FlatAddress + StatusOffset$ and $y = WarningBitOffset$

Quantum Analog Output Channel Value Address: TaQAoV

| Representation | Function Description | | |
|---|---|---------------|--|
|  | This function provides the topological address of the signal value word for the channel of an analog output module. | | |
| | I/O name | I/O data type | Description |
| | <i>Bus</i> | Integer | Number of the bus (for example, 1) |
| | <i>Drop</i> | | Address of the drop (for example, 1) |
| | <i>Rack</i> | | Address of the rack in the drop (for example, 1) |
| | <i>Module</i> | | Address of the module on the rack (for example, 4) |
| | <i>Channel</i> | | Number of the channel of the module (for example, 0) |

M340/M580 Analog Module Status Address: MaStbModS

| Representation | Description | | |
|---|--|---------------|--|
|  | This function provides the topological address of the module status word for an analog module. | | |
| | I/O name | I/O data type | Description |
| | <i>ASmoduleDiagnostic</i> | Integer | STB communicator address received from the I/O scanner |
| | <i>Segment</i> | | STB I/O segment or device |
| | <i>Slot</i> | | STB I/O module or device |
| | <i>ErrAdd</i> | String | Topological address of the analog module status word (%MW, refer to the description of the function algorithm) |

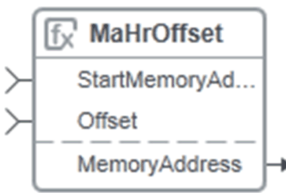
MaStbModS function algorithm:

- $ErrAdd = IF(Segment = 1; IF(Slot < 17; CONCAT(ASModuleDiagnostic; '.'; Slot - 1); CONCAT(ASModuleDiagnostic + 1; '.'; Slot - 17); CONCAT(ASModuleDiagnostic + Segment; '.'; Slot - 1))$

Example:

- $ASmoduleDiagnostic = \%MW100$, $Segment = 3$, $Slot = 4$
- $ErrAdd = \%MW103.3$

Holding Register Address: MaHrOffset

| Representation | Description | | |
|---|--|---------------|---|
|  | This function provides the memory address with the offset. | | |
| | I/O name | I/O data type | Description |
| | <i>StartMemoryAddress</i> | String | Start memory address (for example, %MW100) |
| | <i>Offset</i> | Integer | Memory offset |
| | <i>MemoryAddress</i> | String | Memory address with offset (refer to the description of the function algorithm) |

MaStbModS function algorithm:

- $MemoryAddress = StartMemoryAddress + Offset$

Example:

- $StartMemoryAddress = \%MW10$, $Offset = 5$
- $MemoryAddress = \%MW15$

Control And Supervision Resources

What's in This Part

Prerequisites..... 137

Prerequisites

What's in This Chapter

| | |
|--------------------------------|-----|
| Control Constituents | 137 |
| Supervision Constituents | 140 |

Overview

This chapter describes prerequisites for the Control and Supervision resources that you want to use as part of your template. It also explains how to proceed if you want to create or modify such resources.

Control Constituents

Preparing Control Constituents

Overview

Control resources (constituents) that you want to encapsulate, page 23 in Control facet templates need to be either:

- Contained in a project file (.stu), which is a version-dependent file format and must be compatible with the Control Participant version embedded in the software.
- Created by using the Control Participant. You can open it from the **Facet Editor** and retrieve the compatible project file (.stu) from the **content repository**, page 263.

If you are using the standalone version of the Control Participant to create a project file, ensure that the version is the same as that installed by EcoStruxure Process Expert. If the project uses device type managers (DTMs), the EcoStruxure Control Expert DTM library version also needs to be identical.

You may need to install hotfixes that are installed in the Control Participant. For details, refer to the platform *Release Notes*.

NOTE: You cannot use project files for which an application password is set. You may, however, protect your application once you have encapsulated the control constituents.

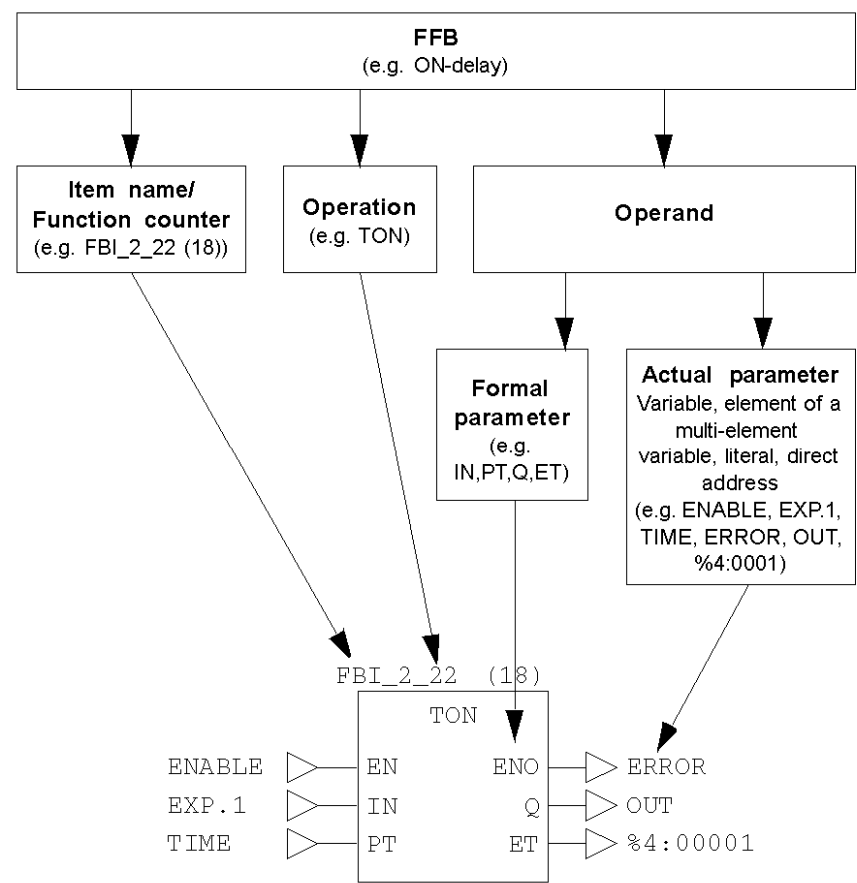
Guidelines

- When creating project files, save them as XEF or ZEF file instead of STU because these file types are forward compatible. This allows you to open them with the Control Participant and save them as STU files with the same version.
- To reduce memory consumption, keep only variables that you need and remove the others.
- Set initial values for variables that are not configured at the instance level.
- Position FFBs in the top, left-hand corner of the section to minimize space usage. You can configure the **free space to be reserved**, page 266 around the resource.
- If possible, transform DFBs into EFBs because it helps improve performance and reduces memory consumption. Use the Control Expert toolkit to do so.
- Select the *HMI* attribute for variables that you encapsulate and that are exchanged between the controller and Supervision to help improve performance (see *EcoStruxure Process Expert, User Guide*).

NOTE: As logical references to EFs cannot be used across sections, you can create a DFB out of an EF to avoid using variables.

Control Resources

Using the example of an EFB, the following figure describes the Control resources that you can encapsulate, page 261 in a facet template.



| Data element | Description |
|-------------------|--|
| Type | Only the type of the FFB that you encapsulate is relevant. The name of the instance that you use for encapsulation is not used. |
| Formal parameters | Selecting input or output pins during encapsulation makes them available in the template editors so that they can be connected to platform inputs, editable parameters, or interfaces by using bindings, page 244. |
| Actual parameters | Selecting variables allows you to create a name for them that is unique, page 48 for each instance. For example, variable <i>PV</i> encapsulated in a template can create variables <i>Motor1_AINPUT_PV</i> where <i>Motor1</i> is the instance identifier and <i>AINPUT</i> is the type. |

Project File Format

You can only encapsulate constituents contained in an STU file.

To use constituent files of the following formats, open them in the Control Participant and save them as STU:

- ZEF
- XEF

NOTE: STU files are version dependant.

Using Existing Control Resources

You may already have Control resources that you want to use.

Before encapsulating them, verify that they have been created with the appropriate software and that they are compatible with the purpose, page 20 and the functionality, page 142 of the template.

Also, verify that no application password is set.

NOTE: Do not use DFB types or DDTs whose names starts with a digit. Otherwise, you cannot create instances of templates in which they are encapsulated.

Modifying Control Resources

To modify constituents encapsulated in a Control facet template (or that appear in the **Document Outline** pane), edit the template and click **Templatizer**. The **Select Variables** window opens, which lets you modify the constituent selection, page 270 and/or open the Control Participant to modify the Control resources, page 270.

NOTE: Each time you save a change to a DFB, increment its version so that changes can be propagated to new versions of templates.

Using Expressions in Constituent Files

Do not use expressions in the Control logic contained in the constituent file (.STU file) that you encapsulate in a Control facet template (_UL). Otherwise, generation will not succeed when you generate the facet of an instance of the template.

An expression is an actual parameter containing a string with operators and operands that are supported by the Control Participant.

To use an expression in a template, enter it in a platform input, page 34 that you create after encapsulating the constituent (at the facet template level) or in an element (at the composite template level). You can also enter it while editing an instance of the template (see *EcoStruxure Process Expert, User Guide*) in a deferred parameter, page 151 of a Control service.

NOTE: Do not use expressions in a platform input or parameter that is linked to an elementary function block (EFB).



Creating Control Resources

Creating Control resources associates them with the facet template that you edit and stores the resulting project file (.stu) in the content repository.

Before creating Control resources, complete the preparation work, page 141 to have a thorough understanding of the template that you want to create. This helps you create a constituent file that is adapted to the purpose of the template.

To create constituents in a Control facet template by using the Control Participant, proceed as follows.

| Step | Action |
|------|---|
| 1 | Create an empty Logic facet template, page 252 and edit it. |
| 2 | Click Templatizer . Result: The Content Not Found dialog box opens. |
| 3 | Select Create New project . |

| Step | Action |
|------|---|
| 4 | <p>From the menu, select the controller platform that you want to associate to the Control facet template.</p> <p>NOTE: This parameter affects the assignment and hardware mapping capabilities of the facet template.</p> |
| 5 | <p>Click OK.</p> <p>Result: The software opens an empty Control Participant project in the Control Participant window (see <i>EcoStruxure Process Expert, User Guide</i>).</p> |
| 6 | <p>Create your own Control resources.</p> <p>NOTE: Do not use types whose name starts with a digit.</p> |
| 7 | <p>Save your changes by clicking the save button.</p>  <p>NOTE: Saving changes does not add constituents to the content repository.</p> |
| 8 | <p>Close the Control Participant window.</p> <p>Result: The Select Variables window, page 261 opens.</p> <p>NOTE: To open the Control Participant window again, click Open Participant.</p> |
| 9 | <p>Select the constituents that you want to encapsulate and save the selection by clicking the save button.</p>  <p>NOTE: Closing the Select Variables window without selecting constituents and saving the selection discards the Control resources that you created in the Control Participant in steps 6 and 7.</p> |
| 10 | <p>Close the Select Variables window.</p> <p>Result: The view reverts to the Control facet template and your selection of constituents appears in the Document Outline pane. The constituents can be used in the template, page 262.</p> <p>NOTE: To open the Select Variables window again, click Templatizer.</p> |
| 11 | <p>Save the Control facet template.</p> <p>Result: The constituents are added to the content repository, page 263 from where you can retrieve them and encapsulate them in another Control facet template, if needed.</p> |

Supervision Constituents

Creating and Modifying Supervision Genies

About Creating and Modifying Genies

Supervision animated graphics (Genies) to be encapsulated in Supervision *Genie* facet templates need to be contained in a project file in .ctz format (Include project).

Starting with EcoStruxure Process Expert 2021, you can readily access included projects that are stored in the content repository or add new ones and use the Graphics Builder of the Supervision Participant to create and modify Genies, page 273.

Preparation

What’s in This Part

| | |
|---|-----|
| Defining Services Provided by Templates | 142 |
| Defining Parameters of Elements | 151 |
| Defining Links and Linking Capabilities | 157 |
| Materializing the Template | 167 |

Overview

This part outlines the preparation work that you need to accomplish to facilitate template creation.

Preparation is key to making powerful templates with less engineering effort.

Defining Services Provided by Templates

What's in This Chapter

| | |
|-------------------------------------|-----|
| Function Analysis | 142 |
| Making a Sketch..... | 143 |
| Defining Control Services..... | 145 |
| Defining Supervision Services | 147 |

Overview

This chapter describes the steps to define the elements of the template that are going to provide services at the instance level.

Function Analysis

Objective

The objective of this step is to analyze the functions to be fulfilled by the control module template.

This analysis is required if you do not have a [function block](#), [page 137](#) yet and you are going to create one to be encapsulated. Nevertheless, if you already have a function block, the analysis helps you determine if there is an advantage in breaking it up. The objective is not to encapsulate one large function block that provides the complete functionality of the control module.

Using or creating a state diagram may be useful to complete the function analysis.

Scope of the Function Analysis

The function analysis of the control module should also encompass the following aspects:

- Application: Interaction of instances of the template with other instances to receive and/or provide data through links.
- Communication: Possibility to share data through the network.
- Hardware: Possibility to connect to I/O modules alternatively to being controlled logically.

NOTE: Consider functions that you may not require immediately but that could be part of the functionality.

Function Analysis Outcome

The analysis of a control module should allow defining the following category of functions:

- Basic, mandatory functions that are directly related to the control of the device being automated, such as timers, conditions, resetting, operating modes, position feedback, out-of-service status, and so on.
- Additional functions such as managing multiple interlock conditions, controlling the device remotely, simulation of parameters, counting hours of operation and cycles, and so on.

Making a Sketch

Objective

From this point on, it is helpful to start making sketches of the composition of the control module template or parts thereof. You can use software or draw them on paper.

Representing graphically the Control and Supervision services that you are about to define makes it easier to organize and group them.

It also helps visualizing the relationship between Control and Supervision.

Update the sketch as you define new parts of the template. Once you have a drawing that represents the entire control module template composition with interfaces and naming convention, use it as your guideline for the implementation in the software.

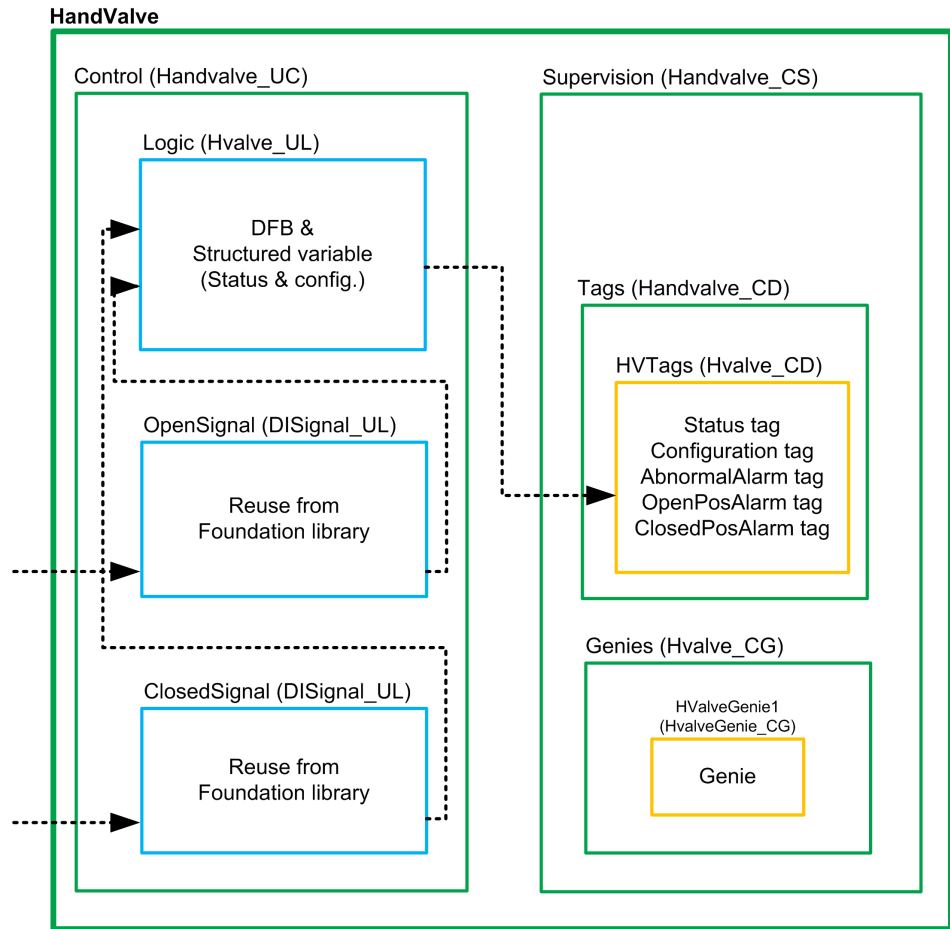
Color Codes

You can use the following color codes to identify the various template components:

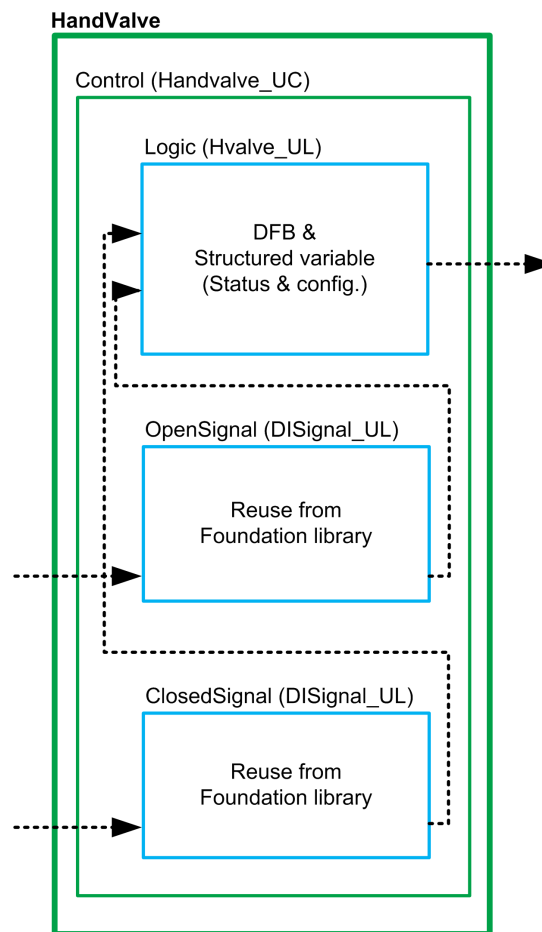
- Composite templates: Green.
- Control facet templates: Blue.
- Supervision facet templates: Orange.
- Encapsulated constituents (FFBs, variables, tags, and genies): Red.

Example

The following figure shows an example of a sketch representing the composition, naming convention of templates, and interfacing (dotted lines) for a simple control module template that models a hand valve.



The following figure shows an example of a sketch representing the composition, naming convention of templates, and interfacing (dotted lines) for a simple control module template that models a hand valve.



To help you formalize the naming convention and relationships between references, you can also create a Microsoft® Excel® file in which you indicate the identifiers of references, encapsulated elements, and names that need to be propagated. The following figure gives an example of such a file for the Control templates of a simple control module modeling a hand valve.

| | A | B | C | D | E | F | G |
|----|---|--------------|-----------------------------------|--|--|------------------------------------|--------------------------------|
| 1 | HandValve Control Module Template | | | | | | |
| 2 | | | | | | | |
| 3 | Control Naming Strategy and Interfacing | | | | | | |
| 4 | | | | | Output | | |
| 5 | Template ID | Reference ID | Template Elements | Element ID | What | To | Interface Model/ID /Role used |
| 6 | HandValve_UC | Control | Logic | | Status & Config variable names | Supervision | see G12 |
| 7 | | | OpenSignal | | | | |
| 8 | | | ClosedSignal | | | | |
| 9 | | | | | | | |
| 10 | HValve_UL | Logic | DFB (Type: HVALVE) | SInstanceID_HValve | Logical references: | Supervision template (Hvalve_CD) | HValveStatus/HValveStatus /Def |
| 11 | | | Structured variable | SInstanceID_HValve_ST | | Status variable tag address | |
| 12 | | | (for Status & Configuration data) | | | Configuration variable tag address | |
| 13 | | | | | | | |
| 14 | | | | | | | |
| 15 | SDISignal_UL | OpenSignal | DFB & Variables | SInstanceID_HValve_ZSHx (x is letter suffix) | Open position variable name: | ZSH pin of HVALVE DFB | SBool/DISignal /Def |
| 16 | | | | | InstanceID_HValve_ZSHV | | |
| 17 | | | | | or name comes from SBool interface if linked | | |
| 18 | | | | | | | |
| 19 | SDISignal_UL | ClosedSignal | DFB & Variables | SInstanceID_HValve_ZSLx (x is letter suffix) | Closed position variable name: | ZSL pin of HVALVE DFB | SBool/DISignal /Def |
| 20 | | | | | InstanceID_HValve_ZSLV | | |
| 21 | | | | | or name comes from SBool interface if linked | | |
| 22 | | | | | | | |

Defining Control Services

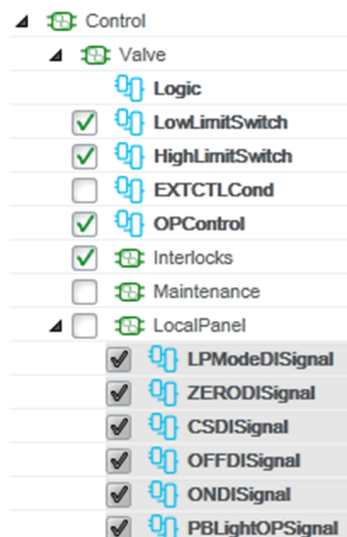
Objective

The objective of this step is to group the functions that you have defined through the function analysis. These represent the Control services that the control module template is to deliver as part of the **Control** composite template.

A group is a set of facet templates that are contained in a Control composite template. You can have groups at different levels of the template composition, page 40.

These services become the elements of the instance, page 32.

The following figure shows one core service (*Logic*), four optional services (for example, *LowLimitSwitch*, and three optional groups of services (*Interlocks*, *Maintenance*, and *LocalPanel*).



Guidelines

Applying the following guidelines may help define Control services depending on your specific applicative requirements:

- Look at control modules of the General Purpose Library (Classic) that provide services that are similar to the one you are creating. This allows you to see how Control services are grouped. Refer to the various templates user guides in the help of the software, which describe the composition of templates.
- You can use the same names as Schneider Electric templates for services and groups providing similar services.
- If you can define identical groups, you may be able to reuse the corresponding facet and/or composite template, which reduces the engineering effort.
- No service (facet template) should be directly located inside the **Control** composite template.

Applying the following guidelines can help optimize element selection at the instance level:

- Make as many services as possible optional. This reduces memory usage by generating only code and/or data for the services that are selected. The instance should function properly with only the core elements.
- Services that are likely to be used in most instances should be optional and selected by default.
- If two services are exclusive, meaning that only one or the other can generate constituents, selecting one should automatically disable the other one. This helps avoid incorrect configurations by reducing complexity.
- Enter a meaningful description for elements because it appears in a tooltip in the **Instance Editor** of the instance.

Defining Services

In Schneider Electric templates the following main services have been defined. You may reuse these names for your templates and/or create new ones:

- **Logic**: The basic functions that are required to control the valve, such as timers, conditions, resetting, operating modes, open/closed feedback, out-of-service status, and so on.
- **Running**: **Signal**.
- **ExternalControl**: **Signal**.
- **xSignal**: Services that use a hardware abstraction layer facet template, where *x* is a description of the signal, its type, and whether it is received or transmitted. For example, **OFFDISignal** for the digital off signal that is received from a local panel.

Defining Groups

Typically, under the **Control** node, you can find groups of services that have names representing their function, for example:

- **DeviceName**: Main group that encompasses services to operate the modeled device. Usually, the name is similar to the name of the control module template.

For example, **Motor** and **PIDController** are groups of the *\$Motor* and *\$PID* templates respectively.



- **Interlocks**: Detailed management of multiple interlock conditions.
- **Failures**: Detailed management of multiple conditions that make the valve inoperable.
- **Maintenance**: Services to count hours of operation and cycles.
- **LocalPanel**: Services to operate the valve from a local panel in the field.
- **AnalogAlarms**: Management of threshold values. Can include simulation.
- **Position**: Management of position feedback. Can include simulation.

The list is not exhaustive and you may use your own names as needed.

Core and Optional Services

For both services and group of services, define if they are **core** or **optional**, page 33.

For optional services, define if they are **exclusive**, page 33 and if they need to be selected by default.

Defining Supervision Services

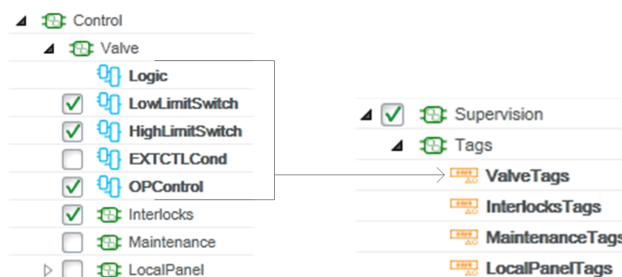
Objective

The objective of this step is to associate Supervision services to the Control services that you have defined for the control module template.

Guidelines

- Look at control modules of the General Purpose Library (Classic) that provide services that are similar to the one you are creating. This allows you to see how Supervision services are organized. Refer to the various Supervision services user guides in the help of the software, which describe the Supervision components that are used by Schneider Electric templates.
- Supervision services should be optional as a whole.
- Selecting a Control service should automatically enable the corresponding Supervision service.
- Use names that make it easy to identify the related Control service.

The following figure shows the correspondence between a set of Control services and the related Supervision services for a control module.



Defining Services

Variable, advanced alarm, trend, and report tags are usually provided by one Supervision data facet with the name `xTags` where `x` represents the name of the corresponding Control service.

For example, `ValveTags` or `InterlockTags`.

Defining Groups

Usually under the **Supervision** node, you find the **Tags** group, which contains the Supervision services related to the main Control services group.

In certain cases, at the next child level you can find groups with the same name as other Control service.

In this example, the **LowLimitSwitch** and **HighLimitSwitch** groups of Supervision services correspond to Control services of the same name, implementing two alarm tags each.

| Name | Template |
|--------------------|--------------------|
| MValveWithPos_1 | \$MValveWithPos |
| ▲ Control | \$MValveWithPos_UC |
| ▷ MValve | \$MVALVE_UC |
| ▷ Position | \$AINPUT_UC |
| ▲ HighLimitSwitch | \$DINPUT_UC |
| Logic | \$DINPUT_UL |
| DISignal | \$DISignal_UL |
| ▲ LowLimitSwitch | \$DINPUT_UC |
| Logic | \$DINPUT_UL |
| DISignal | \$DISignal_UL |
| ▲ Supervision | \$MValveWithPos_CS |
| ▲ Tags | \$MVALVE_CD |
| Valve | \$MOTVALVE_CD |
| Motor2 | \$MOTOR2_CD |
| Position | \$AINPUT_CD |
| Failures | \$CONDSUM_CD |
| Interlocks | \$CONDSUM1_CD |
| Maintenance | \$DEVMNT_CD |
| LocalPanel | \$MVALVELP_CD |
| ▲ HighLimitSwitch | \$DigitalInput_CD |
| Tags | \$DINPUT_CD |
| ▲ LowLimitSwitch | \$DigitalInput_CD |
| Tags | \$DINPUT_CD |

Defining OPC Items

Make a list of the variables that are exchanged with the Supervision system with their complete name. It allows you to configure the *address* of the corresponding variable tag.

Defining Animations

Based on the Supervision services and OPC items that you have defined, select genies to visualize control module data and faceplates to interact with control modules during operation.

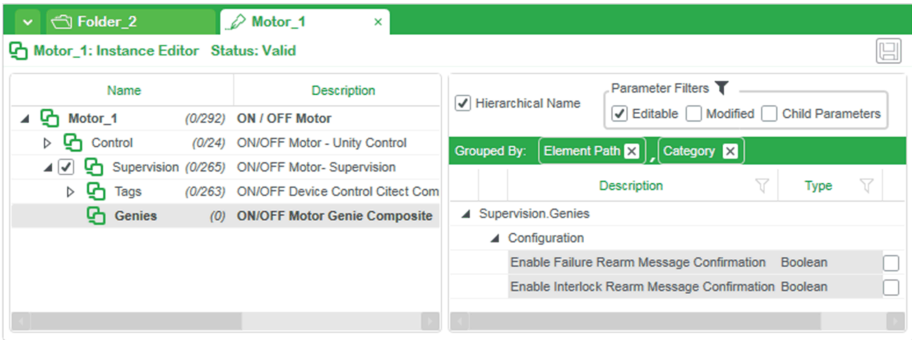
Templates of the General Purpose library use genies that are contained in an Include project.

For example, the following figure shows the *sgc_hlmotor_10* genie of the *sgc_devctl_motors* library that is used for the management of on/off motors.

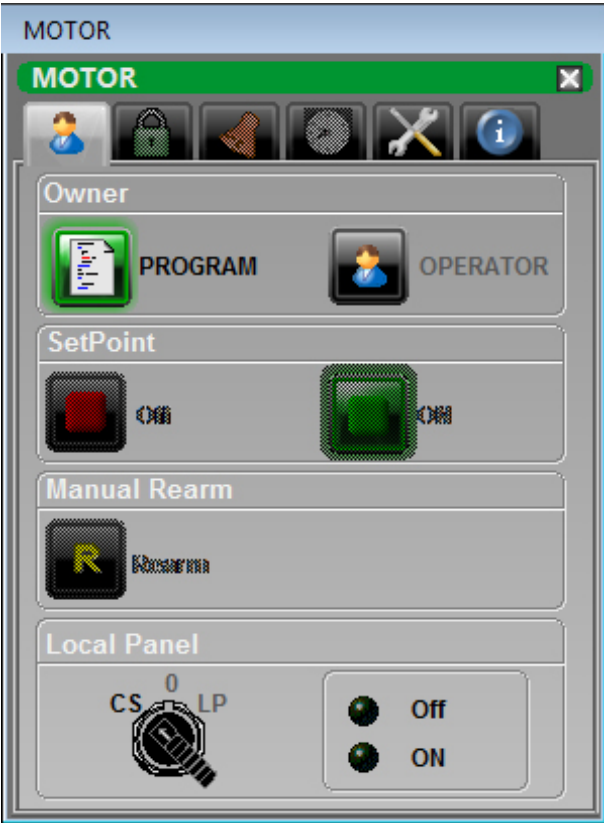


NOTE: Supervision facet templates encapsulating genies do not appear in the **Instance Editor** of the instance as elements unless they contain at least one configurable parameter.

For example, the configurable parameters **Failure Rearm Confirmation** and **Interlock Rearm Confirmation** of the genie composite *Genies* make it visible in the **Instance Editor** of the instance.



The following figure shows the faceplate that is associated to genies of the *sgc_devctl_motors* library. The **Local Panel** section is optional and is displayed when the Control service of the same name is selected.



NOTE: Faceplates are not encapsulated in templates but are in the Include project.

Defining Parameters of Elements

What's in This Chapter

| | |
|---|-----|
| Parameter Description | 151 |
| Creating and Customizing Parameters | 153 |

Overview

This chapter describes the steps to define the parameters of elements of the template that users can view and configure at the instance level.

Parameter Description

Objective

The objective of this step is to define how values for formal and actual parameters, page 138 will be provided during instantiation in the **Instance Editor**.

This applies to each element of a facet template that requires configurable parameters.

Implicit Parameters

For each template, the following implicit parameters are created by default and are accessible during instantiation at the instance level:

- **\$InstanceID**
- **\$Description**
- **\$Area**

By default, these parameters are assigned to the **\$System** category and are identified by the \$ sign prefix.

NOTE: Do not use the \$ sign prefix for other parameters because it is reserved for the three implicit parameters of the **\$System** category.

Guidelines

- To provide data to elements:
 - Create platform inputs for values that are constant.
 - Create parameters for values that need to be configured once.
 - Values on pins that you want to be able to change during online refinement should not be generated from the instance.
- Use the software data types (see *EcoStruxure Process Expert, User Guide*) as opposed to the Participant-specific data types. These are typically more user-friendly. The software converts them during generation as needed.
- Reuse parameter categories inside a control module template so that you can view the entirety of parameters of its instance in the **Instance Editor** by using the **Category** filter.
- Organize parameters inside each template because you cannot reorder deferred parameters.

The following figure shows how the entirety of parameters of an instance is grouped in eight categories. Most categories contain parameters of more than one element.

| Grouped By : | Category | ✓ | El |
|--------------|------------------------------|---|----|
| Description | | | |
| ▷ | \$System | | |
| ▷ | Configuration | | |
| ▷ | Time | | |
| ▷ | Basic Alarm Configuration | | |
| ▷ | Advanced Alarm Configuration | | |
| ▷ | Historize | | |
| ▷ | Message | | |
| ▷ | Variable Tag Disable | | |

Parameter Properties

Parameters can be created directly on the element or be used together with transformation functions.

Each parameter is defined by the following set of properties, such as:

- *Category*
- *Identifier*
- *Description*
- *Data type*
- *Default value*
- *Format*

You need to configure the values of the various properties when you create parameters.

For more information on the properties, refer to the topic describing the *common definition of templates*, page 85.

Viewing Parameters

At each level of the composition of a template, you can view the parameters, page 238 of its elements and child elements.

As such, when you edit a facet template, you can view the parameters of the encapsulated elements.

When you edit a composite containing a facet template, you can view the parameters of the facet template and those of the elements that it encapsulates.

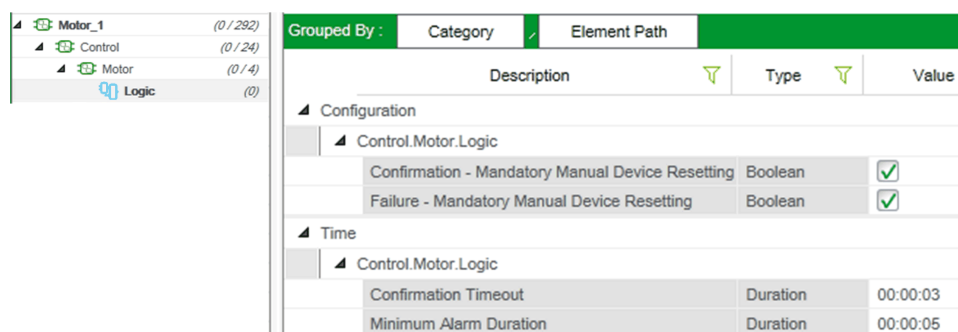
Defining Categories

For each element of the instance, you can create categories and assign parameters to them. This facilitates the management and configuration of parameters during template creation and in the **Instance Editor**.

You can only create categories for parameters that already exist.

By default, parameters are assigned to the **Others** category when you create them.

In this example, the four parameters of element **Logic** are grouped in two categories, **Configuration** and **Time**.



The table indicates the categories that are defined for parameters of Schneider Electric templates and the order in which they appear in the **Instance Editor**.

| Category | Order | Type of template using the category |
|------------------------------|-------|-------------------------------------|
| Configuration | 1 | Control facet templates |
| Time | 2 | |
| Range | 3 | |
| Basic Alarm Configuration | 1 | Supervision data facet templates |
| Advanced Alarm Configuration | 2 | |
| Trend | 3 | |
| Alarm | 4 | |
| Variable Tag Disable | 5 | |
| Alarm Tag Disable | 6 | |
| Trend Tag Disable | 7 | |
| Configuration | 8 | |
| Message | 9 | |
| Historize | 10 | |

Creating and Customizing Parameters

Creating Parameters

For more information on how to create parameters, refer to the topic describing element actions in the description of the **Facet Editor**, page 228.

Creating Enumerations

To create an enumeration for a parameter, proceed as follows.

| Step | Action |
|------|---|
| 1 | If the parameter does not exist yet, create it. NOTE: You cannot edit parameters of child elements. |
| 2 | Click the button to display the Parameters pane. |
| 3 | Click the Type field of the parameter and select Enum . |
| 4 | In the Default Value column of the parameter, click the menu button to add enumerations. |

| Step | Action |
|------|---|
| | Result: The Add/Edit Enumerations dialog box opens. |
| 5 | Click the + button. Result: A default enumerator is created. NOTE: To remove an enumerator, select it, and click the X button. |
| 6 | Click the Name field and edit the string. This is the name that appears in the Value menu of the parameter when you edit an instance by using the Instance Editor . NOTE: Names need to be unique within an enumeration. |
| 7 | If required, click the Value field and edit the associated numerical value. NOTE: Values must be of type Int and unique within an enumeration. |
| 8 | To create additional enumerators, repeat steps 5 to 7. |
| 9 | Click OK . Result: The enumeration is created. |
| 10 | In the Parameters pane, click the Default Value field of the parameter and select the enumerator that is to be used by default. |

NOTE: To edit an enumeration, click the menu button in the **Default Value** field.

Assigning Parameters to a Category

To assign a parameter to a category, proceed as follows.

| Step | Action |
|------|--|
| 1 | Open the Parameters pane, page 238. |
| 2 | Expand the Others category. |
| 3 | Click the name of the category (Others) of the parameter and enter the name of a category. |
| 4 | Press Enter . Result: If you have entered the name of a category that already exists for this template, the software assigns the parameter to it; otherwise the software creates the new category and assigns the parameter to it. The parameter also appears in the new category in the Instance Editor . |

NOTE: To assign the parameter to another category, repeat the procedure by entering the name of the category to which you want to assign the parameter.

Deferring Parameters

For parameters to become accessible during instantiation, you need to defer them at each level of the control module composition.

By default, new parameters and interfaces are deferred (see *EcoStruxure Process Expert, User Guide*) automatically.

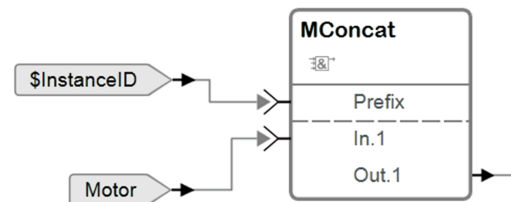
In addition, you can use the **Create Deferred** context menu command, page 231.

When you defer a parameter, you can change its name, page 170 that is displayed in the **Instance Editor** by using an alias. This allows you to customize the name of the parameter of an element that is referenced by several control module templates. In each instance of the respective control module template, the parameter can have a specific name.

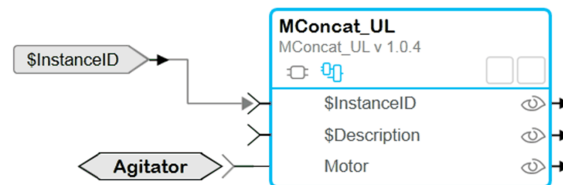
Deferred parameters can be hidden in the representation of the element by using a filter, page 240.

NOTE: In a composite template, you also need to defer parameters of elements referenced by facet templates so that their input and output pins become available for data propagation by using bindings, page 43 at the parent level.

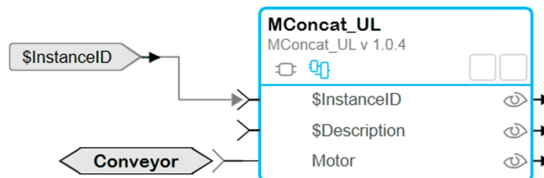
In this example, facet template `MConcat_UL` contains user-defined parameter *Motor*. The facet template is referenced by two control module templates, *Agitator* and *Conveyor*.



In the scope of *Agitator*, deferred parameter *Motor* of reference `MConcat_UL` uses alias *Agitator*.



In the scope of *Conveyor*, deferred parameter *Motor* of reference `MConcat_UL` uses alias *Conveyor*.



Changing the Position of Parameters and Categories

Parameters and categories appear in the **Instance Editor** in the position that you have defined in the **Parameters** pane of the template editor.

To change the position of a parameter within its category, proceed as follows.

| Step | Action |
|------|--|
| 1 | Open the Parameters pane of the template. |
| 2 | Expand the category that contains the parameter whose position you want to modify. |
| 3 | Select the parameter and drag it to the new position within the category. Result: A tooltip indicates the position that the parameter will occupy when you release the mouse button. |
| 4 | Release the mouse button. Result: The parameter is moved to the new position in the Parameters pane. |

NOTE: By selecting a category instead of a parameter, you can change the position of an entire category.

Defining Links and Linking Capabilities

What's in This Chapter

| | |
|------------------------------------|-----|
| Working Principle | 157 |
| Types of Interface Links | 158 |
| Other Interface Link Aspects | 163 |

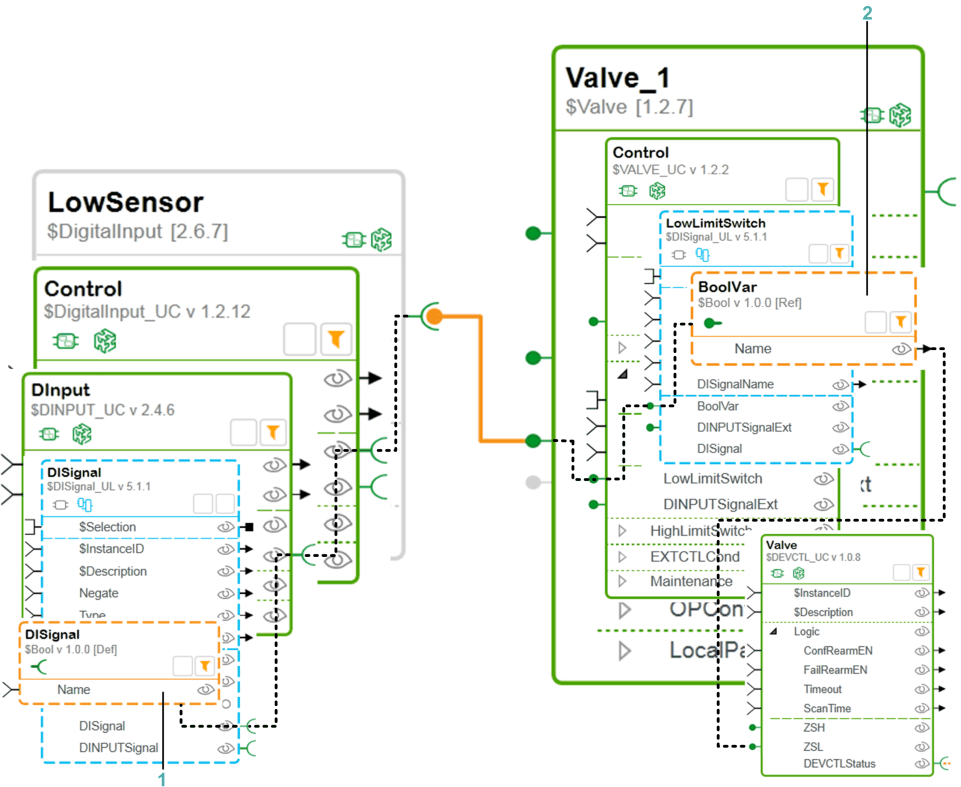
Objective

The objective of this step is to define the interfaces that are required to exchange data within the control module template, with other instances of the application, and with facets modeling hardware I/O modules.

Working Principle

Illustration

The following example illustrates the working principle of an interface, which is used to propagate the name of the variable holding the boolean value of a low limit sensor. Data is propagated between the instance modeling the sensor and the instance modeling valve functions.



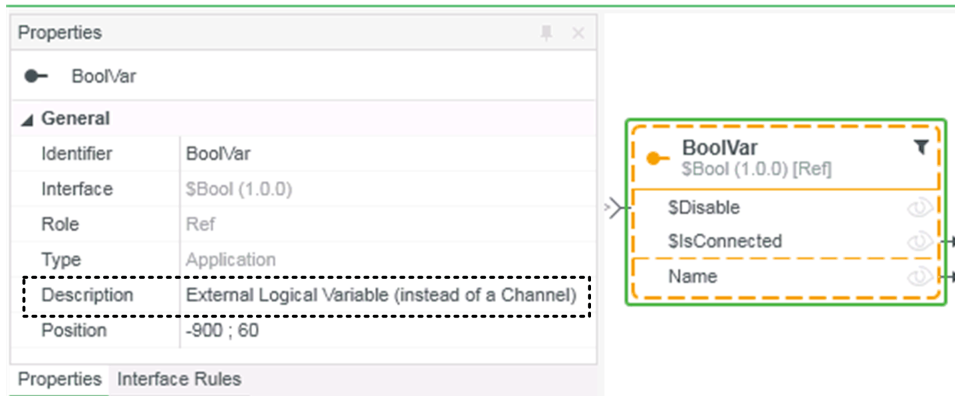
| Item | Description |
|------|---|
| 1 | The name of a boolean variable is provided by role <i>Def</i> of interface <i>DISignal</i> , which uses interface model <i>\$Bool</i> . The interface originates at the constituent level (not represented). This unbound interface is deferred up to the <i>\$DigitalInput</i> control module level to be exposed and available in the Asset Workspace Editor/Links Editor . |
| 2 | An interface link is established with role <i>Ref</i> of interface <i>Bool/Var</i> , which also uses interface model <i>\$Bool</i> . Inside control module <i>\$Valve</i> , the interface originates at the constituent level (not represented) and is deferred up to the highest level to be exposed. Internally, this interface passes data on to the <i>ZSL</i> element of the template in charge of creating the logic of the valve function. NOTE: The identifiers of the interfaces are not relevant for the link to be established. Only roles and interface model matter. |

Guidelines

- Prefer links or logical references over variables to propagate data.
- Reuse existing interfaces of the Foundation library as much as possible.
- When several elements need to be exchanged between templates, combine them into one interface.
- Provide a meaningful description for interfaces. The description that you enter in the *Description* property of the interface appears in a tooltip in the **Asset Workspace Editor/Links Editor**.
- To propagate addresses of *HMI* variables that are exchanged with the Supervision system, create a common interface with identifier *xStatus* (where *x* represents the name of the corresponding Control element.).

For example, interface *Bool/Var* is an element of facet template *\$DISignal_UL*. When editing *\$DISignal_UL*, click *Bool/Var* to view its properties in the **Properties** tab. The description that you enter in the *Description* field appears in a tooltip in the **Asset Workspace Editor/Links Editor** when you hover with the pointer over the interface.

 \$DISignal_UL (5.3.4) : Facet Editor



Types of Interface Links

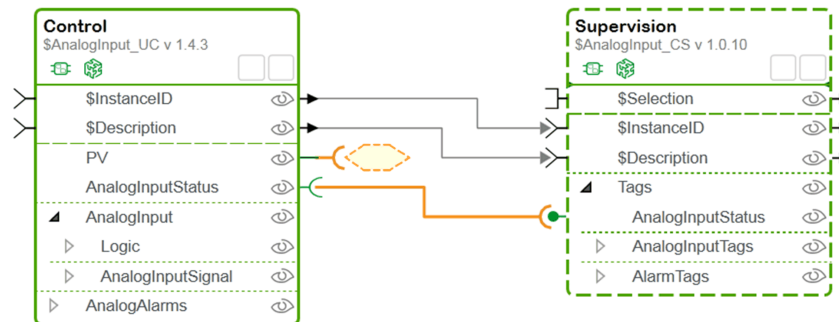
Overview

For application templates, you need to distinguish between two types of links:

- Internal links where both roles of an interface are used inside the control module template. They are used to propagate data within the control module.

- External links where only one role is used inside the template; the other role is not used and may be exposed. They are used to receive and/or provide data from/to other instances or topological entities.
 - For control modules, external links represent the linking capabilities of their instances through exposed application interfaces and mapping interfaces.
 - For facet or composite templates, they allow these templates to exchange data when they are used as references inside a control module template. In this case, when the unused end of the interface is connected to another element of the same control module, such links become internal links.

Based on the *\$AnalogInput* control module template, the figure shows both internal and exposed interfaces.

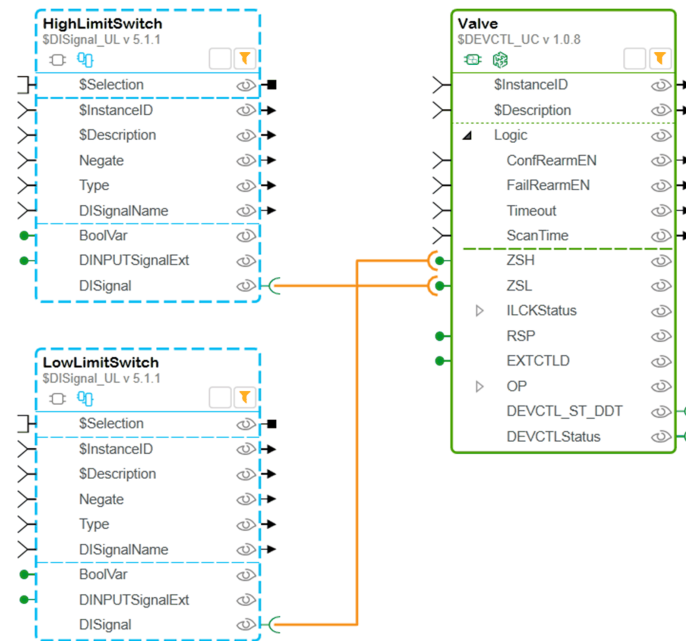


| | |
|-------------------|--|
| PV | <p>Linking capability of instances of <i>\$AnalogInput</i>. The interface that propagates the logical reference holding the process value (PV) originates at the constituent, three levels down in the composition of composite Control. It uses the <i>\$Real</i> interface model.</p> <p>The interface is deferred at each level and thus is exposed for use by other instances during instantiation.</p> <p>Because role <i>Def</i> is used by the template, only interfaces using <i>\$Real</i> and role <i>Ref</i> can be connected to it.</p> |
| AnalogInputStatus | <p>This internal interface links elements of <i>\$AnalogInput_UC</i> and <i>\$AnalogInput_CS</i> inside control module <i>\$AnalogInput</i>. It is used to propagate names of status variables from Control resources to Supervision resources.</p> <p>At the instance level, this interface is not visible.</p> |

Internal Interface Links

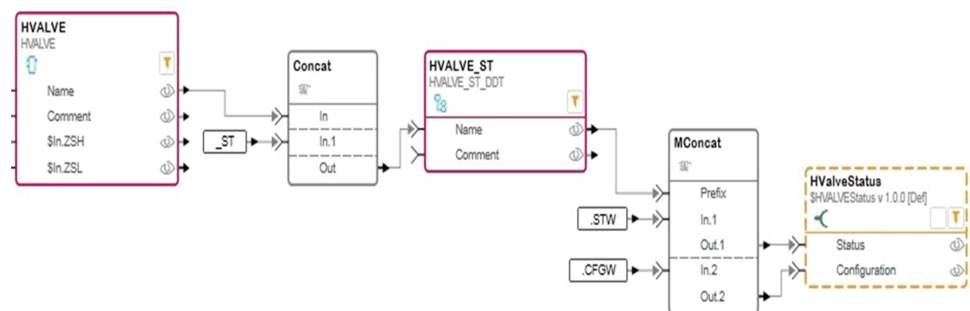
Interface links do not only link templates of different Participants. They also allow linking Control templates to each other. This mechanism is helpful to make standard templates, such as *\$DISignal_UL*, reusable by making the necessary connectors for data exchange readily available.

The following figure gives an example of internal interface links among Control templates of the same control module. It shows how the names of signals for low and high limit switches are propagated to the template that is in charge of creating the logic. Reading signals from channels of digital input modules is a typical use case for the `$DISignal_UL` HAL template.



Propagating Addresses by Internal Interface Link

The following example shows multi-element interface *HValveStatus*, which is created to propagate from Control two tag addresses to be used in the Supervision data facets. The addresses correspond to the logical references of the two child parameters *STW* (for status word) and *CFGW* (for configuration word) of the *HVALVE_ST* data structure. The tag addresses that are created through concatenation and propagated are respectively `$InstanceId_ST.STW` and `InstanceId_ST.CFGW`. The producer role *Def* of the interface, shown here, is connected to Control and the consumer role *Ref* is connected to Supervision.



NOTE: Alternatively, you can use the `$FullName` property, page 46 with the *HVALVE_ST* data structure instead of using the *MConcat* binding function.

NOTE: The value of the `$InstanceId` parameter is provided by the parent levels of the composition of the control module.

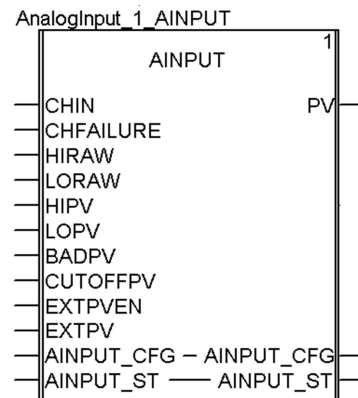
External Links Using Exposed Interfaces

This topic explains the mechanisms of interface links between two instances:

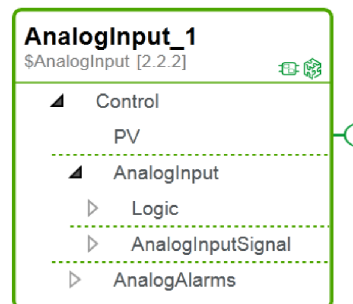
- *AnalogInput_1*: Models a sensor providing an analog position feedback to a motorized valve instance through the PV output pin of data type REAL of its DFB *AnalogInput_1_AINPUT*. The pin is exposed through the *PV* interface.

- *MValveWithPos*: Models the analog valve functionality to receive the position feedback data on the *EXTPV* input pin of its DFB *MValveWithPos_AI*. The pin is exposed through the *RealVar* interface.

The following figure shows the *AnalogInput_1_AINPUT* DFB generated by the *AnalogInput_1* instance and its *PV* output pin.



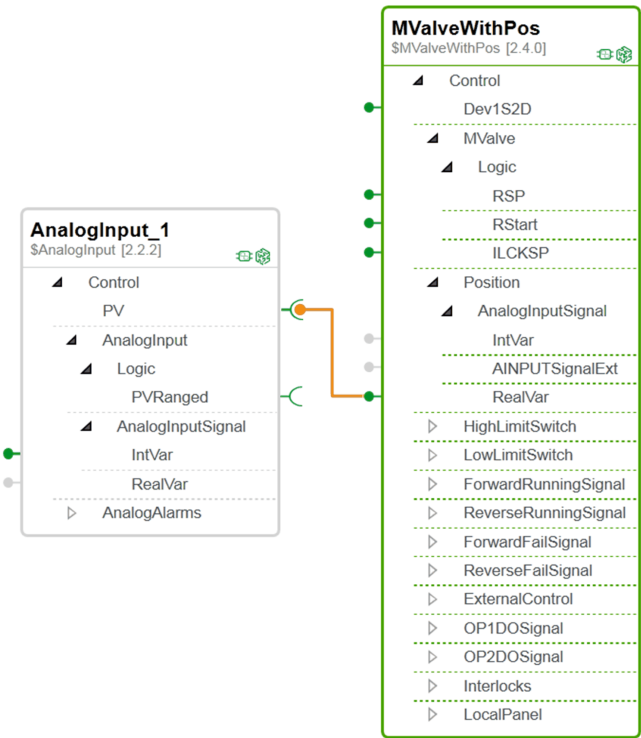
This representation of an instance of *\$AnalogInput* in the **Links Editor** shows the exposed interface for *PV*, which is positioned on the same side as the corresponding input pin.



The **Interfaces** node of the **Inspect Instance** window for *AnalogInput_1* shows the interface model of *PV (\$Real)*, its role (*Def* as provider), and the name of the logical reference that it propagates (*AnalogInput_1_AINPUT.PV*). Click an element to view details in the right-hand section of the window.

| Name | Description | Template | Name | Description | Type | Value |
|--|-------------|----------|---|-------------|------|-------|
| <div> <div> <div></div> <div>AnalogInput1_1</div> </div> <div> <div></div> <div> <div> <div>Elements</div> <div> <div>Control</div> <div> <div>Elements</div> <div> <div>Interfaces</div> <div> <div>PV</div> <div>Def</div> </div> </div> </div> </div> </div> </div> </div> <div> <div>Analog Indicator with Configuration Ran: \$AnalogInput1</div> <div>Analog Indicator with Configurable Ranç: \$AnalogInput1_UC</div> <div></div> <div>\$Real</div> <div>\$Real</div> </div> </div> | | | <div> <div> <div></div> <div><Not Categorized></div> </div> <div> <div>\$Disable</div> <div>Boolean</div> <div>False</div> </div> <div> <div>\$IsConnected</div> <div>Boolean</div> <div>False</div> </div> <div> <div>\$InstanceID</div> <div>String</div> <div></div> </div> <div> <div>\$TemplateID</div> <div>String</div> <div></div> </div> <div> <div>\$InstanceDesc</div> <div>String</div> <div></div> </div> </div> | | | |
| | | | | | | |

The following figure shows both instances linked through their exposed interfaces *PV* and *RealVar*.



The **Inspect Instance** window of *MValveWithPos_1* shows the interface model of *RealVar* (*\$Real*), its role (*Ref* as receiver), and the name of the logical reference that it receives (*AnalogInput_1_AINPUT.PV*, which is the one propagated by interface *PV* of *AnalogInput_1*).

MValveWithPos_1: Inspect Instance

Name

MValveWithPos_1

Elements

Control

Elements

MValve

Position

Elements

Logic

AINPUTSignal

Elements

Interfaces

IntVar

AINPUTSignalExt

RealVar

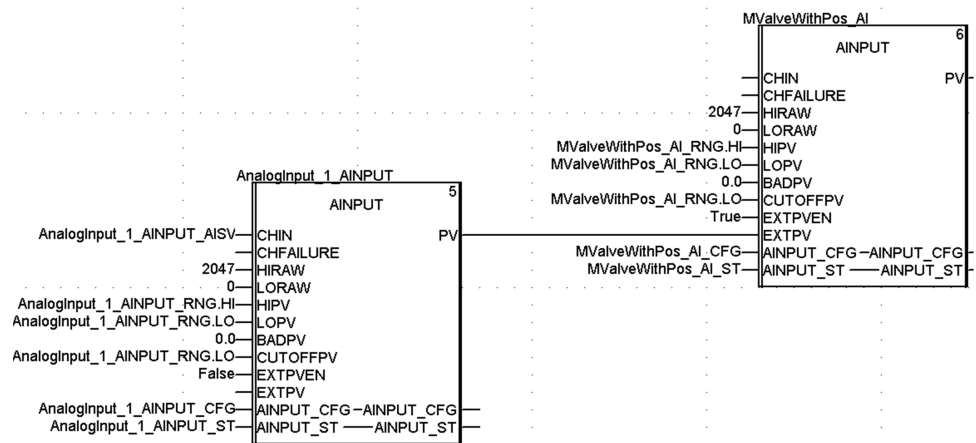
Ref

| Name | Description | Type | Value |
|--------------------|-------------|---------|---------------|
| <Not Categorized> | | | |
| \$Disable | | Boolean | False |
| \$IsConnected | | Boolean | True |
| \$InstanceID | | String | AnalogInput_1 |
| \$TemplateID | | String | \$AnalogInput |
| \$InstanceDesc | | String | |
| \$IsConnected.Name | | Boolean | True |

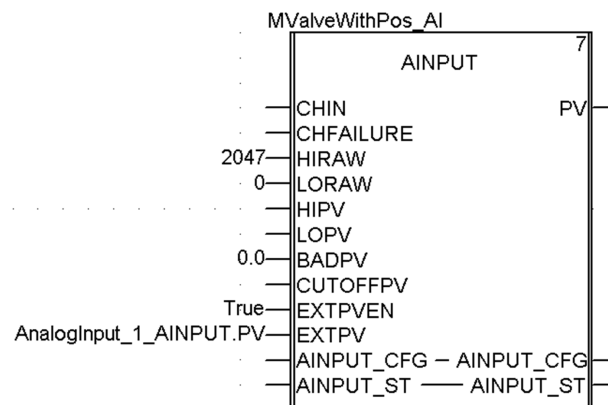
| Name | Description | Type | Value |
|-------------------|-------------|--------|-------------------------|
| <Not Categorized> | | | |
| Name | | String | AnalogInput_1_AINPUT.PV |

NOTE: You can see that the interface *RealVar* is connected and the source of the connection through the *\$IsConnected* and *\$InstanceID* parameters. They are visible in the right-hand section of the **Inspect Instance** window.

The following figure shows the resulting code in the Control Participant when facets of the two instances encapsulating the DFBs are assigned to the same section. The interface creates a link between the pins:



When facets encapsulating the DFBs are assigned to different sections, the interface creates a logical reference on the **EXTPV** pin of the **MValveWithPos_AI** DFB of the **MValveWithPos** instance:



External Links Using Mapping Interfaces

Use HAL templates of the Foundation library to include mapping interfaces, page 35 with your templates. They allow connecting instances of the application with facets representing I/O modules of the topology, independently of the hardware platform that is used.

Other Interface Link Aspects

Overview

This topic describes some of the properties of the interface definition, page 62 that have an impact on interface links.

Deferring Interfaces

After you have added an element to a template, defer its interfaces so that they become available at the parent level.

The identifiers of deferred items, page 170 appear in the graphic representation of the parent template where you can view their properties.

You do not need to defer mapping interfaces. These become automatically available in the corresponding mapping editor.

NOTE: By default, new interfaces are deferred (see *EcoStruxure Process Expert, User Guide*) automatically.

NOTE: Deferred parameters can be hidden in the representation of the element by using a filter, page 240.

Positioning Interfaces

Position an interface on the same side as the corresponding pin on the DFB to which it connects by using the **Left/Right** property in the **Interface Rules** pane, page 88.

Selecting the Order of Appearance

You can change the position in which an interface appears at the parent level by dragging the interface label in the **Interface Rules** pane to the desired position.

NOTE: The left-to-right order in the **Interface Rules** pane corresponds to a top-to-bottom order in the graphical representation at the parent level.

The following example shows how inverting interfaces *BoolVar* and *DINPUTSignalExt* in facet template *\$DISignal_UL* inverts the way they are displayed when you edit the composite template *\$DINPUT_UC*, which references *\$DISignal_UL*. (The identifiers of the two interfaces have been highlighted in yellow for the purpose of this example.)



NOTE: To implement such a change, you need to create a new version of the facet template in which you have inverted the interfaces. Then, update the composite template to reference the new version of the facet template.

Selecting Roles

When selecting the role, page 64 for an interface, take into consideration the following properties of the interface:

- Execution order.
- Direction of elements.
- Cardinality.
- Icon representing the role.

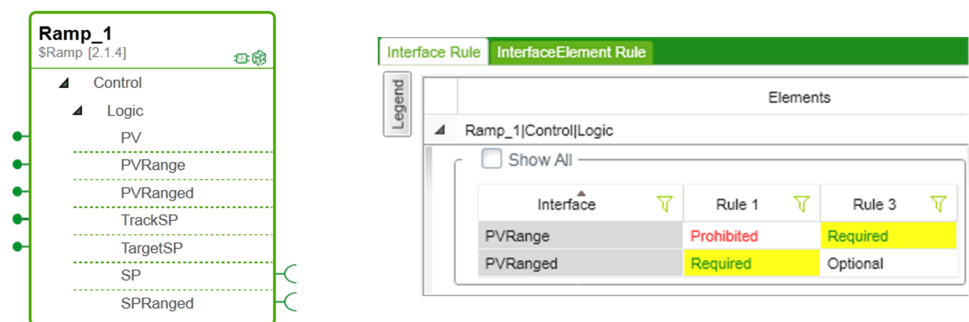
Defining Interface Rules

Define interface rules, page 88 for interface links that are mandatory so that instances do receive data they require to function properly. Also, define rules to help avoid incompatible links.

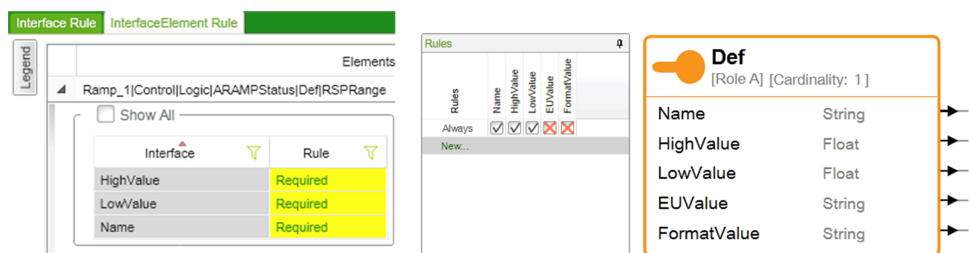
In addition to interface rules, you also need to define rules for each element of a multi-element interface. With this additional level of validation, you make it mandatory or optional to provide specific data to an instance.

As long as interface and interface element rules are not satisfied, you cannot use the instance further in the system engineering life cycle.

This example shows an instance of control module template *\$Ramp* for which an interface rule applies to the *PVRange* and *PVRanged* interfaces. At least one of them needs to be linked. If *PVRanged* is linked, *PVRange* cannot.



For the *PVRange* interface, which uses multi-element interface model *\$Range* interface element rules apply in addition. The following figure shows the **Validity Diagnosis** dialog box and how the rules are defined in the interface model for elements *Name*, *HighValue*, and *LowValue*, which are mandatory.

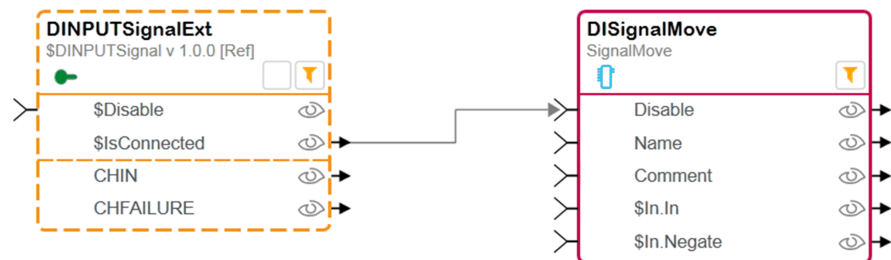


Enabling and Disabling Interfaces

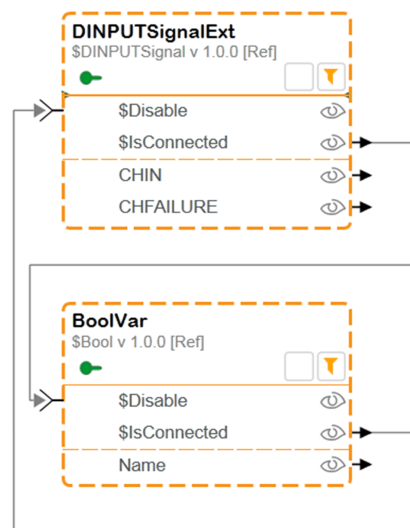
You can disable interfaces and get their connection status by using the following interface properties:

| | |
|---------------|--|
| \$Disable | When the boolean value provided to the property through a binding, page 244 is <i>TRUE</i> , the interface is disabled. In such case, an exposed interface is visible but unavailable, a mapping interface is not visible, and no other interface propagates data. |
| \$IsConnected | This property outputs a boolean value <i>TRUE</i> when the interface link is successfully established. This implies that applicable interface and interface element rules are satisfied; otherwise the value is <i>FALSE</i> . This connection status is usually used to condition the status of other elements. |

The following figure illustrates the use of the *\$IsConnected* parameter of an exposed interface to disable an element. In this example, when the value of the *\$IsConnected* parameter is *TRUE*, the value of the *Disable* parameter is *TRUE*, and the function block is not generated in the logical Participant project.



The following figure illustrates the use of the *\$IsConnected* and *\$Disable* parameters of two exposed interfaces to make them exclusive one another. In this example, when the *DINPUTSignalExt* interface is connected, the *BoolVar* interface is disabled, and the other way around.



Editing/Extending Interfaces In-Place

When you edit a template that references an interface, you can edit and/or extend it in-place, page 308 by using the **Edit/Extend Interface** context menu command.

You also need to update the other role of this interface in the templates that reference it.

Materializing the Template

What's in This Chapter

| | |
|------------------------------------|-----|
| Reusing Global Templates | 167 |
| Defining a Naming Convention | 168 |
| Defining Instance Appearance..... | 171 |

Overview

This chapter describes the final steps of the preparation work that impact the composition and appearance of the control module and its instances.

Reusing Global Templates

Objective

The objective of this step is to define which Global Templates of the General Purpose Library (Classic) and Foundation Library you can reuse with no or little adaptation to reduce the engineering effort. You can duplicate, page 305 these templates.

Commonly Used Control Services

Schneider Electric control module templates use composite and facet templates that provide many standard services, such as:

- Signals: Facet templates of the Foundation library with identifiers ending with **Signal_UL*, for example *\$DISignal_UL* modeling digital input signals.
- Local panel: Composite templates with identifiers ending with **LP_UC*. For example, *\$DEVLP_UC* modeling local panel functions of on/off devices such as motors.
- Interlock or initial condition management: Composite template *\$CONDSUM1_UC*.
- Diagnostic information management: Composite template *\$CONDSUM_UC*.
- Maintenance information management: Composite template *\$DEVMNT_UC*.
- Genie facet templates. Refer to the help of the General Purpose Library (Classic) for a description of the templated genies.

Commonly Used Supervision Services

Schneider Electric control module templates use templates that provide many standard services, such as:

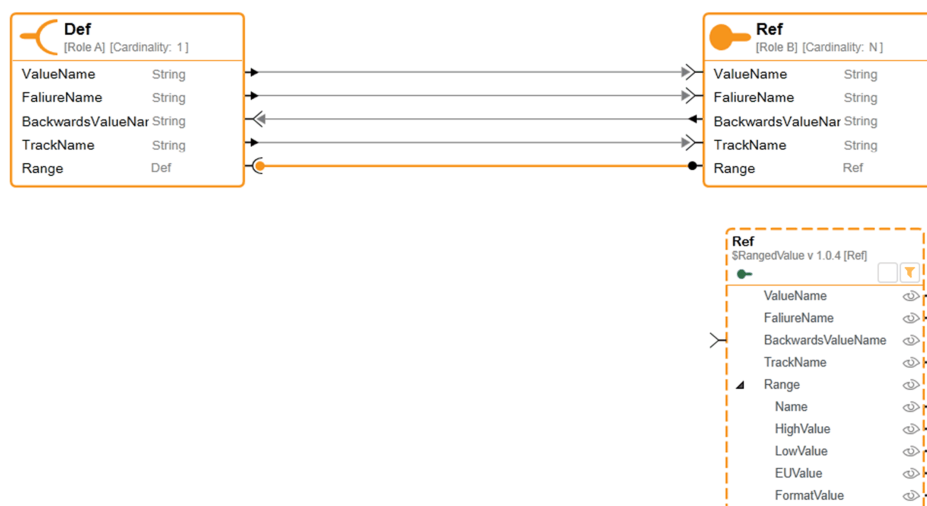
- Services associated to existing Control services: Templates that have the same identifier as the Control service but with a different suffix. For example, facet template *\$DEVMNT_CD* provides Supervision services for the *\$DEVMNT_UC* maintenance Control services.
- Animated graphics: Refer to the help of the General Purpose Library (Classic) for a description of the genies that are templated and the tags that they manage.

Commonly Used Interface Models

Schneider Electric control module templates use generic interface models to propagate data within templates and/or between instances, such as:

- Elementary interfaces, for example, *\$Bool*, *\$Real*
- Multi-element interfaces, for example *\$DEV1S1D* to connect an on-off device type instance to a drive instance, *\$RangedValue*.

The following figure shows the *\$RangedValue* interface and the representation of its *REF* role in a template editor. This interface, which contains the *\$Range* nested interface, page 27 is commonly used in templates where low and high range values are required along with their unit and display format.



Defining a Naming Convention

Objective

The objective of this step is to take into consideration the name aspect in the template creation.

As such, names can be:

- For templates, interface models, and their references: Identifiers.
- For parameters: Identifiers, aliases and categories.

The concept of name can also be extended to include descriptions.

Guidelines

- You cannot use the \$ prefix for templates that you create or modify, page 39.
- At the first level of a control module template, name composite templates like the Participants they are associated to. For example, Control and Supervision.
- Use the same names as Schneider Electric templates for services and groups providing similar services. For example, Maintenance.
- Use the same names for Control and Supervision templates and distinguish them by using the appropriate suffix. For example, *DEVLP_UC* and *DEVLP_CD*.
- Be aware of the length and character limitations of names of elements that are generated in Participant projects.

- Apply the naming convention, page 48 and use appropriate identifiers, page 22 for templates and interfaces.

Using Descriptions

Name length limitations may not allow you to use descriptive identifiers.

Use descriptions to help users understand the purpose of items.

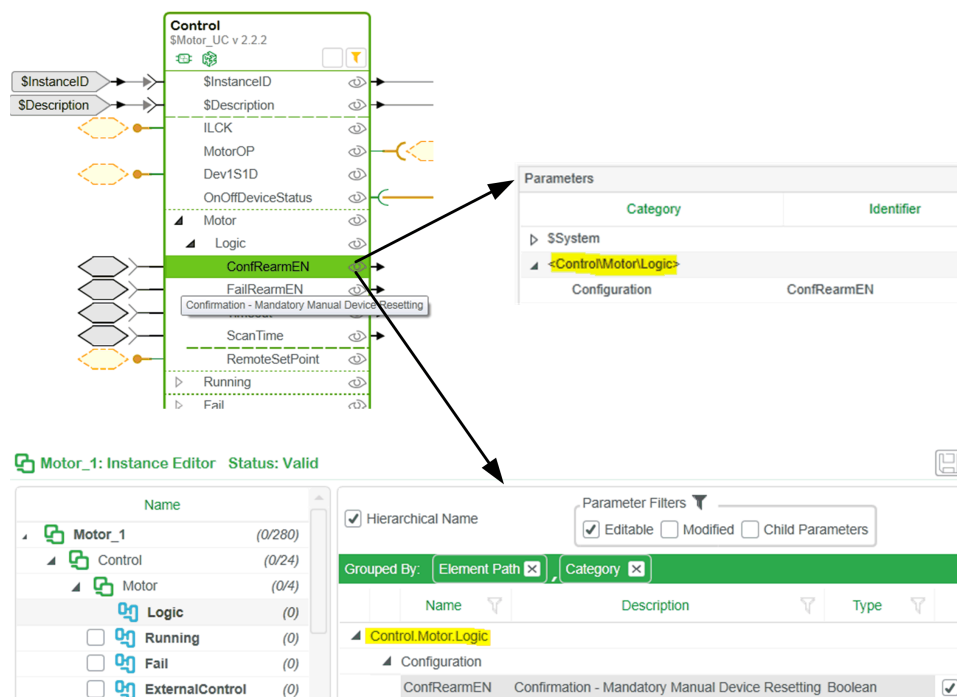
Descriptions may appear in tooltips at the instance level.

Parameters

The path of parameters is displayed in various explorers and editors of the software. The path includes the identifiers of the references of the control module hierarchy.

Using appropriate identifiers helps locating parameters and indicating their purpose.

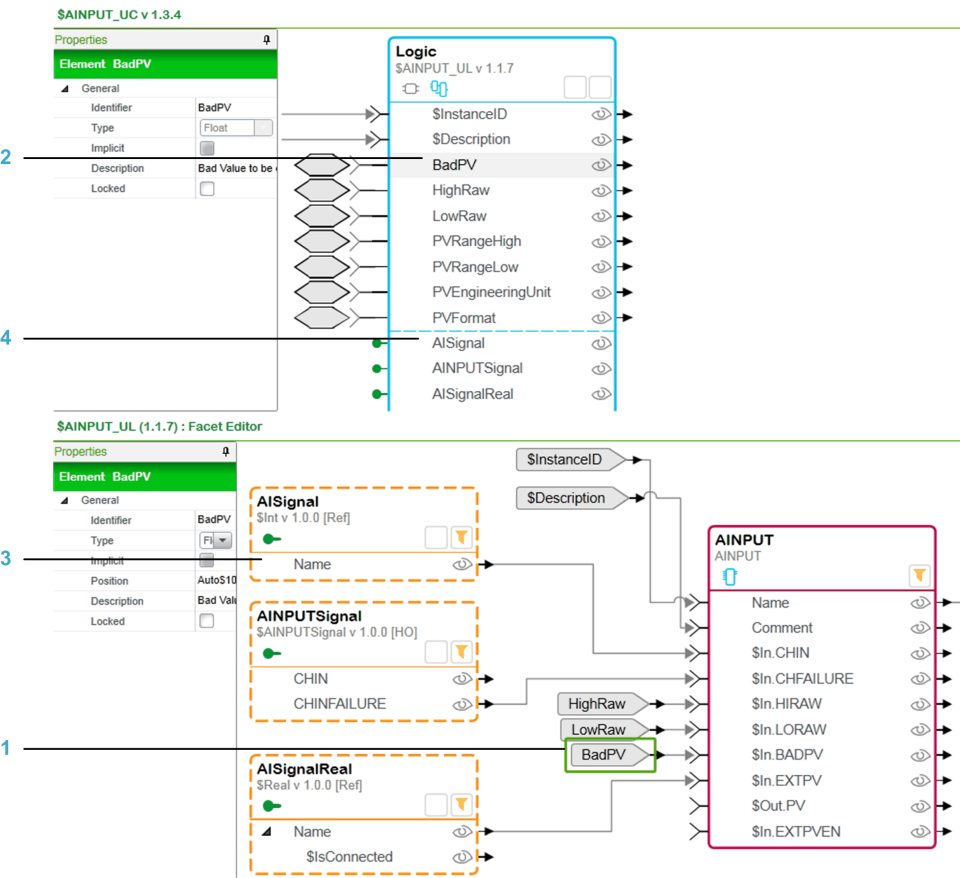
In the following example, the path *Control\Motor\Logic* for the **ConfRearmEN** parameter appears in the **Parameters** pane when you edit a *\$Motor* sample template and in the **Instance Editor** of an instance of *\$Motor* (the paths have been highlighted in yellow for the purpose of this example).



NOTE: *\$Motor_UC (Control)* is an element of *\$Motor*.

Deferred Parameters and Interfaces

In this example, sample facet template \$AINPUT_UL is an element of composite template \$AINPUT_UC. The following figure shows both templates in their respective editor.

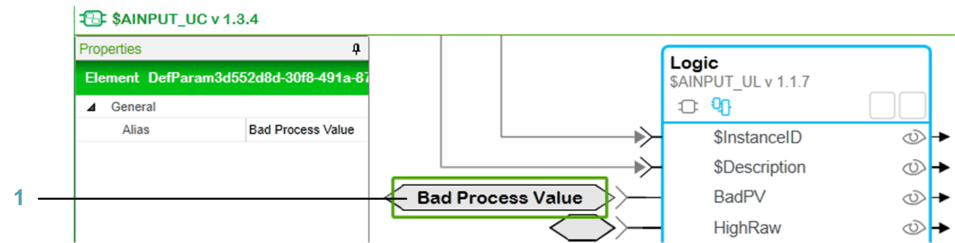


| Item | Description |
|------|---|
| 1 | Parameter <i>BadPV</i> is connected to the input pin <i>BADPV</i> of DFB <i>AINPUT</i> . When you select the parameter, <i>BadPV</i> appears under identifier in the Properties pane. |
| 2 | The same identifier appears in the graphical representation of the facet template and in the Properties pane when you select the parameter (instead of <i>BADPV</i>). You cannot edit the identifier at the composite level. You need to edit it at the facet level. The same applies to the parameter description. |
| 3 | Interface <i>AISignal</i> is connected to the input pin <i>CHIN</i> of DFB <i>AINPUT</i> . When you select the interface, <i>AISignal</i> appears under identifier in the Properties pane. You can edit this identifier here. |
| 4 | The same identifier appears in the graphical representation of the facet template and in the Properties pane when you select the interface (instead of <i>CHIN</i>). You cannot edit the identifier at the composite level. The same applies to the interface description. |

Using Aliases

For deferred parameters and interfaces, you can enter aliases, which are displayed at the instance level instead of the identifier. This is possible at the various levels of the composition.

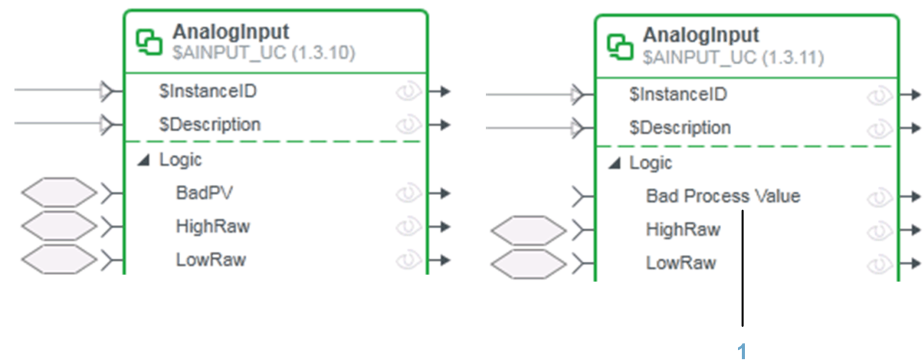
It allows you to display customized names when the same element is used in various control module templates.



1 For deferred parameter *BadPV*, alias *Bad Process Value* was entered, which appears under **Alias** in the **Properties** pane. You can edit this alias here.

The alias appears in the **Name** column in the **Instance Editor** of the instance instead of the parameter identifier.

The alias of the deferred parameter *BadPV* appears instead of the identifier at the parent level. The following figure on the left shows the original identifier (*BadPV*) of the deferred parameter. The figure on the right shows the result after entering the alias and updating the various levels of templates.



1 Alias *Bad Process Value*

NOTE: The same applies to deferred interfaces.

Defining Instance Appearance

Objective

The objective of this step is to define how you want instances of a control module template to appear in the **Instance Editor** and to organize the template composition accordingly; yet the composition strategy, page 40 should be followed.

Organizing Services

In this example, using an *\$AnalogInput* sample control module template, the appearance of the instance has been defined based on the following criteria:

- Control services related to analog input signals need to be optional.
- Control services related to analog alarms need to be optional.

For the rest, the composition strategy that is described in this manual has been applied.

NOTE: You can change the order in which nodes appear by using the sorting function but hierarchical dependencies do not change.

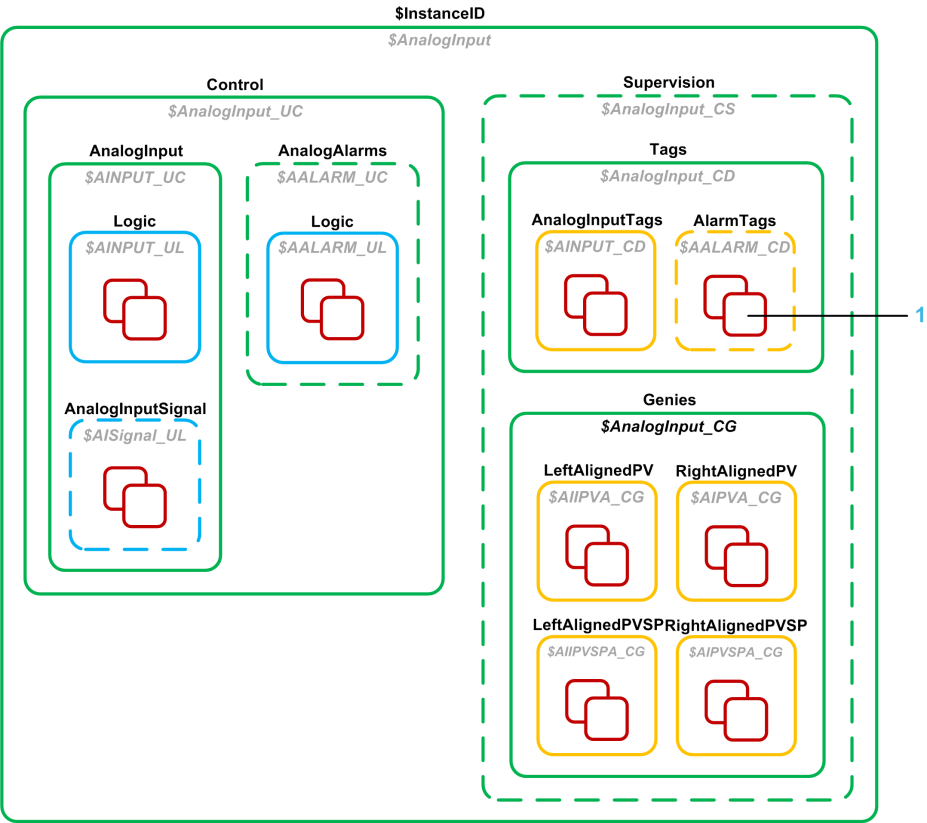
The following graphic shows the appearance of an instance of *\$AnalogInput* in the **Instance Editor**.

▼ + AnalogInput_1 x

AnalogInput_1

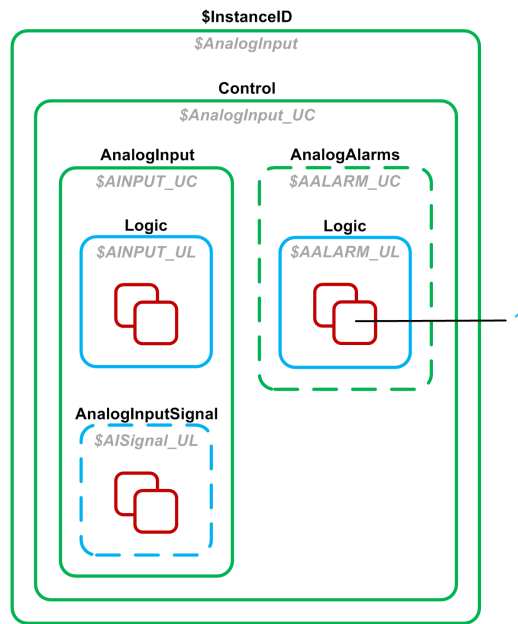
| Name | | Template |
|------------------------|-----------|------------------|
| ▲ AnalogInput_1 | (0 / 145) | \$AnalogInput |
| ▲ Control | (0 / 18) | \$AnalogInput_UC |
| ▲ AnalogInput | (0 / 10) | \$AINPUT_UC |
| Logic | (0) | \$AINPUT_UL |
| ✓ AnalogInputSignal(0) | | \$AISignal_UL |
| ▲ AnalogAlarms | (0 / 8) | \$AALARM_UC |
| Logic | (0) | \$AALARM_UL |
| ▲ ✓ Supervision | (0 / 124) | \$AnalogInput_CS |
| ▲ Tags | (0 / 124) | \$AnalogInput_CD |
| AlarmTags | (0) | \$AINPUT_CD |
| AlarmTags | (0) | \$AALARM_CD |

To obtain the desired appearance, Control and Supervision services need to be organized as shown in the following figure.



1 Constituents

To obtain the desired appearance, Control services need to be organized as shown in the following figure.



1 Constituents

Managing, Modifying, and Creating Templates


What's in This Part

| | |
|---|-----|
| Template Creation Wizard | 175 |
| Global Templates Editors | 220 |
| Editing, Creating, and Saving Global Templates..... | 244 |
| Configuring Interface Models..... | 257 |
| Configuring Facet Templates..... | 260 |
| Configuring Composite Templates | 286 |
| Configuring Control Module Templates..... | 289 |

Overview

Creating, modifying, updating, replacing, or duplicating templates may affect the function of these templates and/or systems, and must be performed by qualified personnel.

Refer also to the topic containing information about template design guidelines and key aspects, page 16.

 **WARNING**

LOSS OF CONTROL

- Verify that templates generate correct addresses when their mapping interfaces are mapped by using the **Hardware Mapping Editor**.
- 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.

Template Creation Wizard

What's in This Chapter

| | |
|--|-----|
| Overview | 176 |
| Template Creation Wizard – Composite Window | 180 |
| Template Creation Wizard Control Facet Windows 1/2 | 182 |
| Template Creation Wizard Control Facet Window 2/2 | 184 |
| Template Creation Wizard Supervision Data Facet Window 1/2 | 190 |
| Template Creation Wizard Supervision Data Facet Window 2/2 | 192 |
| Template Creation Wizard Genie Facet Window 1/2 | 197 |
| Template Creation Wizard Genie Facet Window 2/2 | 199 |
| Template Creation Wizard Composition Summary Window | 201 |
| Identifier Management | 203 |
| Creating Templates by Using the Wizard | 204 |
| Template Creation Example | 207 |
| Template Creation Example – Control Logic Facet 1/2 | 208 |
| Template Creation Example – Control Logic Facet 2/2 | 209 |
| Template Creation Example – Supervision Data Facet 1/2 | 212 |
| Template Creation Example – Supervision Data Facet 2/2 | 212 |
| Template Creation Example – Supervision Genie Facet 1/2 | 216 |
| Template Creation Example – Supervision Genie Facet 2/2 | 217 |
| Template Creation Example – Summary | 218 |

Overview

Purpose of the Wizard

With the necessary preparation work, page 141 completed, the wizard lets you create a functional application control module template with Control and/or Supervision functionality in a few steps.

For Control, it only requires a project file containing the necessary Control resources, page 137 and the wizard lets you select and encapsulate the constituents that you want to use.

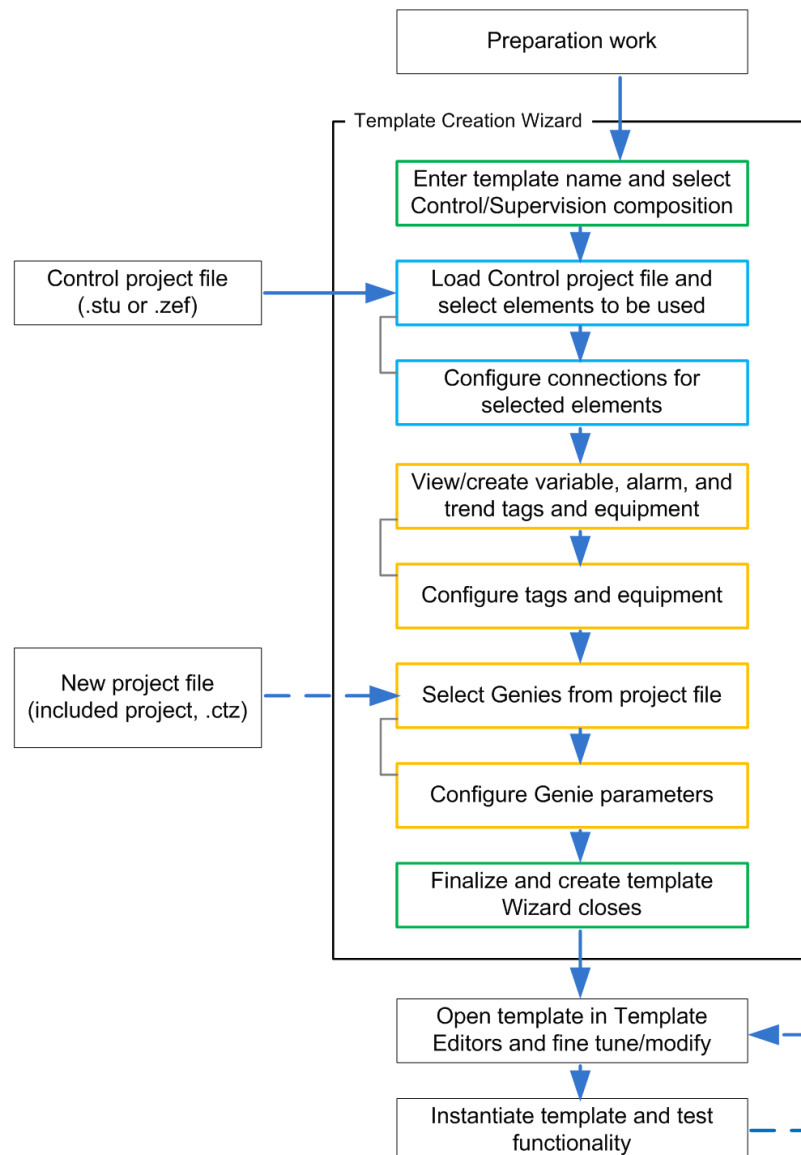
For Supervision, you can use the project files (Include projects, .ctz) that are already present in the content repository or add additional ones to encapsulate your own animated graphics (Genies).

The following pages describe the various windows of the wizard and the procedure to create a template. An example, page 207 illustrates the descriptions.

NOTE: The wizard does not allow creating or editing Genies, page 272. You need to do so before opening the wizard.

Workflow

The following figure shows the workflow to create a template containing Control and Supervision functionality by using the template creation wizard. If either functionality is not required, the corresponding steps can be skipped.



| | |
|-----|-------------------------------|
| | Global Template functionality |
| | Control functionality |
| | Supervision functionality |
| — | Required |
| --- | Optional |

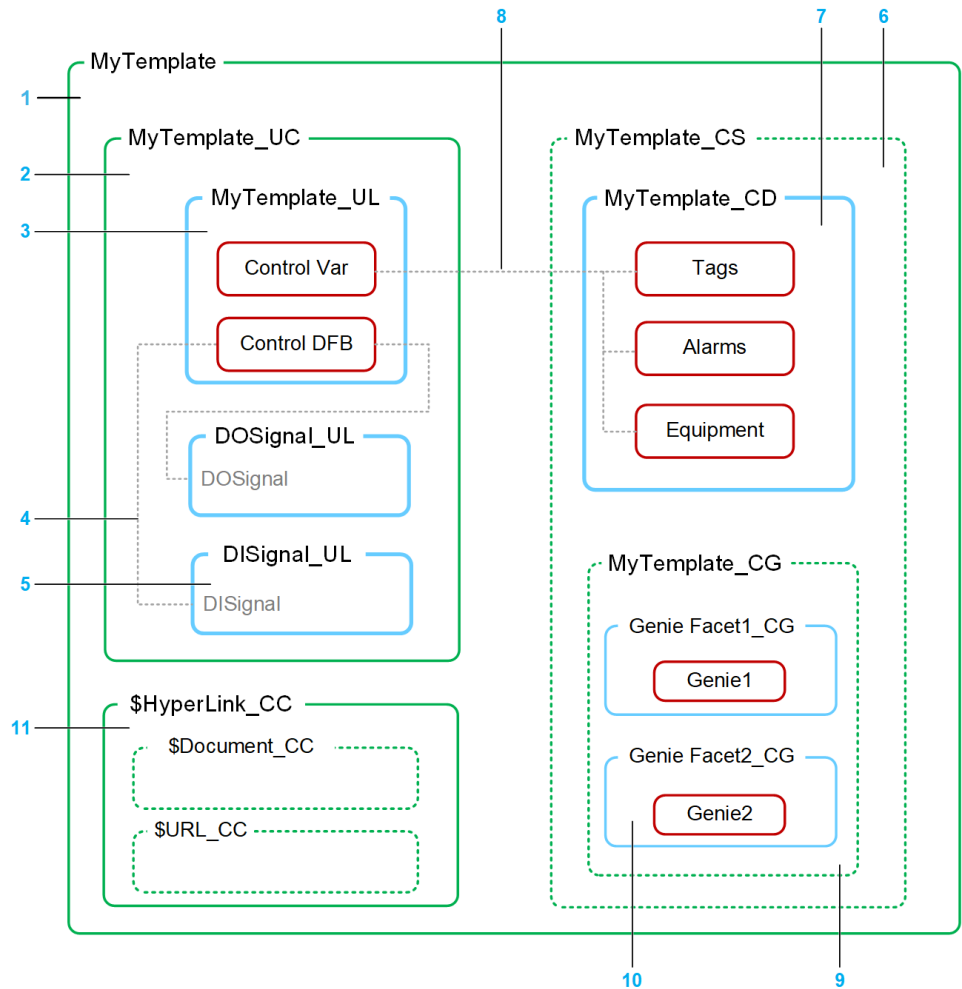
Typically, once the preparation work is completed, the creation of the template by using the wizard takes approximately one to two hours depending on its complexity.

In most cases, some fine tuning is required to finalize the template. This step has to be performed by using the various template editors because once you have created the template, you cannot modify it by using the wizard.

NOTE: The wizard opens in a tab, which allows you to work in other tabs in parallel. However, you cannot close the wizard during the template creation process and resume later.

Control Module Template Composition

The following figure shows an example of the composition of a control module template (the *composite* template) that you can create by using the wizard. Not all bindings and interfaces are shown. You can create fewer child templates, but at least one Control logic (_UL) or Supervision data (_CD) or Genie (_CG) facet template needs to be created.



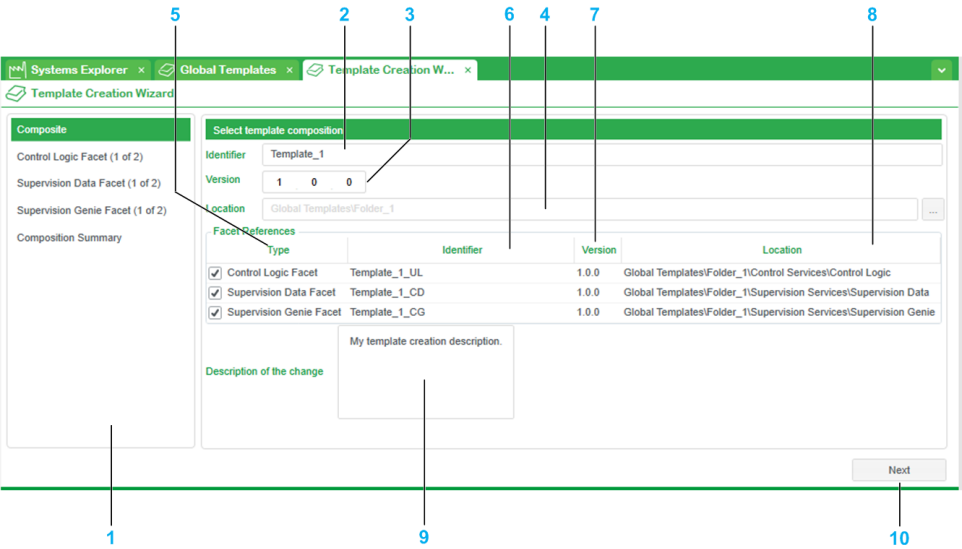
| Item | Description |
|------|--|
| 1 | Highest level composite template (control module). |
| 2 | Control composite template that the software automatically creates to regroup Control elements. No configuration is required. |
| 3 | Control logic facet template that contains the Control constituents that you select from an external project file (.stu or .zef). |
| 4 | Interface-HAL interface link (Hardware Abstraction Layer). |
| 5 | HAL facet template that the software automatically creates based on your configuration. It lets you perform hardware mapping. |
| 6 | Supervision composite template that the software automatically creates to regroup Supervision elements. No configuration is required. By default, the template is configured as <i>optional</i> . |
| 7 | Supervision data facet template that contains various tags, alarms, and equipment depending on your configuration. |
| 8 | Interface-HMI interface link. |

| Item | Description |
|------|--|
| 9 | Supervision Genie composite template that regroups the Genie facet templates. By default, the Supervision Genie facet template is configured as <i>optional</i> . |
| 10 | Genie facet templates that contain the Genies that you select from existing or compatible external project files (Include projects, .ctz). One template per Genie. |
| 11 | Optional template that you can add to use the hyperlink functionality. It lets you open your documents (see <i>EcoStruxure Process Expert, Runtime Navigation Services, User Guide</i>), files, and web links on the computer running an operation client when you use runtime navigation services. |

Template Creation Wizard – Composite Window

Composite Window Description

The following figure shows an example of the composite template window that opens when you select the **Template Creation Wizard** command from the context menu of the **Global Templates** root folder. It lets you select the composition of the template that you want to create. A template name has been entered in the **Identifier** text box.



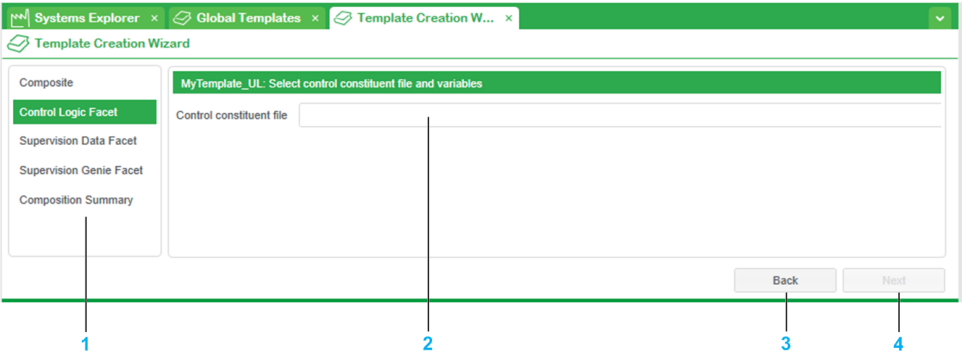
| Item | Description |
|------|---|
| 1 | <p>Pane showing the steps of the template creation process and your progression.</p> <p>The content of the pane reflects the selection of child templates in the Facet References column.</p> <p>To go from one step to other, use the Next and Back buttons in each window.</p> |
| 2 | <p>Identifier of the composite template that you are creating.</p> <p>The software accepts only unique combinations of Identifier and Version.</p> <p>When you create the templates in the last step, the software verifies the uniqueness of the Identifier and Version combination and lets you change either one in case of conflict with an existing template.</p> |
| 3 | <p>Version of the composite template that you are creating.</p> |
| 4 | <p>Location in the Global Templates Library where the composite template is stored once created.</p> <p>By default, folder from where you have selected the Template Creation Wizard command.</p> <p>Browse to select another existing location in the Global Templates Library.</p> <p>NOTE: You can view the location of other templates that the software uses in the Composition Summary window, page 201.</p> |
| 5 | <p>Lets you select which facet templates the composite will contain.</p> <p>At least one facet template is required.</p> <p>You can come back to this window and modify your selection at any time while the wizard is open. Your facet configuration is retained when you clear a facet and select it again.</p> <p>Selecting/clearing a check box enables/disables the corresponding windows (steps) as you progress through the wizard.</p> <p>NOTE:</p> <p>For the Control facets (_UL) and Supervision facets (_CG and _CD), intermediate composite templates are automatically created (_UC and _CS, page 22 respectively). In addition, Hardware Abstraction Layer (HAL) templates can be created depending on your configuration, page 184.</p> |

| Item | Description |
|------|--|
| | The complete list of templates is shown in the Composition Summary window, page 201 (except interfaces). |
| 6 | <p>Identifier of the facet template.</p> <p>By default, the identifier is composed of the identifier of the composite and a facet template specific suffix, page 22.</p> <p>When you create the templates in the last step, the software verifies the uniqueness of the Identifier and Version combination and lets you change either one in case of conflict with an existing template.</p> |
| 7 | Version of the facet template. |
| 8 | <p>Location where the facet template is stored once created.</p> <p>Starting from the folder that will contain the composite, the software recreates a folder structure, page 58 similar to that of the Global Templates Library.</p> |
| 9 | Lets you enter a description with free-form text that will appear in the changes log of the composite template and its child templates. |
| 10 | Opens the next window, which lets you start the configuration process of the facet templates. |

Template Creation Wizard Control Facet Windows 1/2

Project File Selection

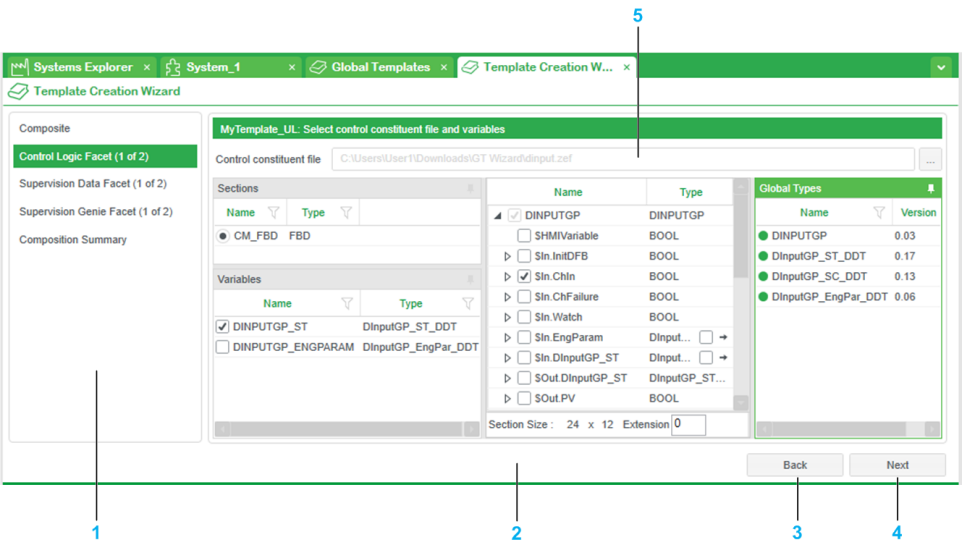
The following figure shows an example of the first Control logic facet window that is shown if you have selected the *Control Logic Facet* in the composite window, page 180. No Control constituent file has been selected yet.



| Item | Description |
|------|---|
| 1 | Pane showing the steps of the template creation process and your progression. |
| 2 | You must select a project file, page 204 on your computer or the network, which contains the Control constituents that you want to encapsulate. |
| 3 | Lets you go back to the Composite window. The selections you have made in this window are retained. |
| 4 | The button to proceed to step 2/2 is enabled once the data of the Control constituent file is loaded. |

Control Constituent Selection

The following figure shows an example of the Control logic facet window once the Control constituent file that you have selected is loaded. It shows the content of the file and lets you select the Control constituents that you want to use in the template.



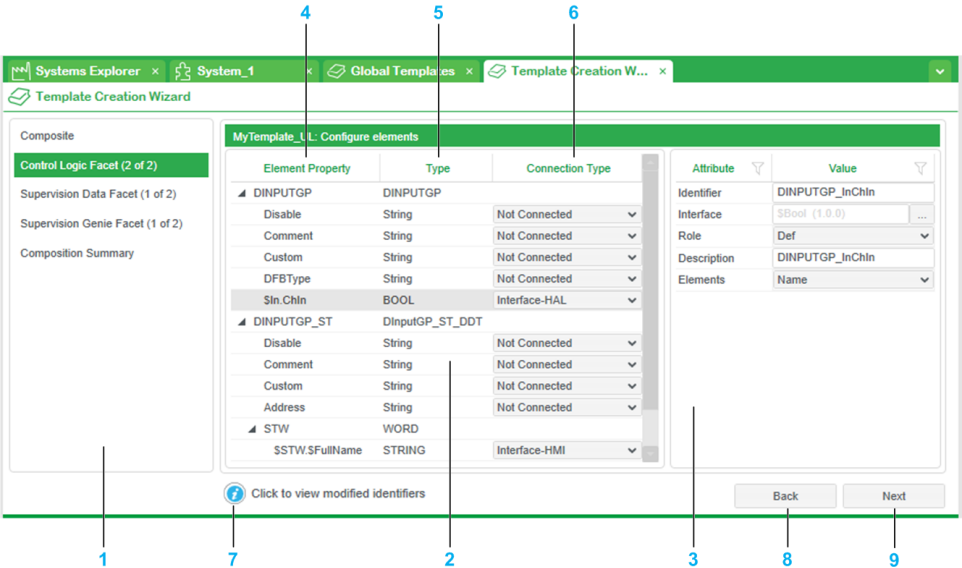
| Item | Description |
|------|---|
| 1 | Pane showing the steps of the template creation process and your progression. |
| 2 | For a description of the main section of this window, refer to the topic describing the Select Variables window, page 261. |

| Item | Description |
|------|---|
| 3 | Lets you go back to the previous window. The selections you have made in this window are retained. |
| 4 | Opens the next window, which lets you configure the elements of the facet template that will be created based on the constituents that you have selected. |
| 5 | Lets you select another constituent file. This action discards the constituent selection. |

Template Creation Wizard Control Facet Window 2/2

Control Element Property Configuration

The following figure shows an example of the second Control logic facet window, which lets you configure the elements, page 93 representing the Control constituents that you selected in the previous step. It gives you access to the properties of each element.



| Item | Description |
|------|---|
| 1 | Pane showing the steps of the template creation process and your progression. |
| 2 | <p>Properties of elements that you have selected in step 1/2 and for which you can configure a type of connection for data transmission within the template and to/from other templates.</p> <p>The connections are represented graphically when you edit the templates once they are created.</p> |
| 3 | <p>Lets you configure the connection that you selected for each property. For details, refer to the sections Connection Attribute Descriptions, page 186 after this table.</p> <p>NOTE: Value text boxes support copy/paste.</p> |
| 4 | <p>Each element is shown as a node.</p> <p>For each element, in addition to the system properties (such as <i>Disable</i>, <i>comment</i>), the properties that you selected in the previous window are shown.</p> <p>NOTE: The <i>Identifier</i> property is not shown because it is automatically connected.</p> |
| 5 | Data type of the property. |

| Item | Description |
|------|--|
| 6 | <p>Select a type of connection for each property.</p> <p>Possible values:</p> <ul style="list-style-type: none"> • Parameter: Default value unless otherwise mentioned. Creates and connects an editable parameter , page 186 to the input pin of the property. • Input Value: Creates and connects a fixed-value input, page 187 to the input pin of the property. • Interface: Default value for properties that only have an output pin. Connects the pin of the property to the element of an interface, page 187 that you select from a list of existing interfaces of the Global Templates Library. You can connect several properties, page 206 to different elements of the same interface. • Interface-HMI: Connects the pin of the property to the element of an interface, page 187 that you select from a list of existing interfaces of the Global Templates Library. <p>In addition, creates in the Supervision Data Facet window, page 190 (if selected) the corresponding variable tag element.</p> <ul style="list-style-type: none"> • Interface-HAL: Connects the pin of the property to an elementary variable interface, page 188 that you select from a list and creates a supporting Hardware Abstraction Layer (HAL) signal template, page 189 that is connected to the other end of this interface. • Binding: Connects the input of the property to the output of a property of another element within the same template by using a binding, page 189. <p>NOTE: Connecting the output of a property to the input of another element is performed by using a template editor once the template is created.</p> <ul style="list-style-type: none"> • Not Connected: The property is not bound and hidden, page 240. Select this connection type when no other connection type is required. <p>NOTE: If you change the connection type, the attribute configuration that you made in the left-hand pane for this property is discarded.</p> |
| 7 | <p>The information icon appears if a system-generated variable name exceeds 50 characters.</p> <p>Click it to open the Modified Identifiers dialog box, page 203.</p> |
| 8 | <p>Button to go back to the previous window.</p> <p>The selections you made in this window are retained. However, if you clear an item in the previous window, the corresponding property and connection configuration is discarded.</p> |
| 9 | <p>Button to open the next window of the template creation process.</p> |

Parameter Connection Attribute Description

The following table describes the attributes when **Connection Type** for a property is set to *Parameter*. For details about parameters, refer to the topic describing the definition of parameters, page 85.

| Attribute | Description |
|----------------------|---|
| Category | <p>Lets you group properties by category.</p> <p>This grouping is used in the Parameters pane, page 85 of template editors and in the Instance Editor.</p> <p>Editable free-form text box.</p> <p>Default value: <Element name></p> |
| Identifier | <p>Name of the parameter.</p> <p>It appears in the Instance Editor under Name.</p> <p>Editable free-form text box.</p> <p>Default value: <Element name>_<Property name></p> <p>NOTE: You can connect to a property one of the following existing implicit system parameters, page 85 by entering its name in the text box:</p> <ul style="list-style-type: none"> • \$InstanceID • \$Description • \$Area <p>In this case, enter \$System in Category, which is the existing category for system parameters.</p> |
| Description | <p>Description of the parameter.</p> <p>The string appears for the parameter in the Instance Editor. Optional.</p> <p>Editable free-form text box.</p> <p>Default value: <Element name>_<Property name></p> |
| Type | <p>Data type of the parameter.</p> <p>The type restricts values that can be entered for the parameter while editing the instance.</p> <p>Read-only.</p> <p>Default value: The value is type and property specific.</p> |
| Default value | <p>Value that is set by default for the parameter and that appears in the Instance Editor under Value. Optional.</p> <p>Editable free-form text box.</p> <p>Default value: Blank</p> |

Input Value Connection Attribute Description

The following table describes the attributes when **Connection Type** for a property is set to *Input Value*.

| Attribute | Description |
|--------------------|---|
| Value | Fixed value that is used as input for the property and that cannot be edited during instantiation. Editable free-form text box. Default value: Blank |
| Description | Description of the value. String that appears for the input value in the Instance Editor . Optional. Editable free-form text box. Default value: <Element name>_<Property name> |

Interface and Interface-HMI Connection Attribute Description

The following table describes the attributes when **Connection Type** for a property is set to *Interface* or *Interface-HMI*. For details about interfaces, refer to the topic describing the definition of interfaces, page 157.

| Attribute | Description |
|--------------------|---|
| Identifier | Identifier, page 62 of the interface that is created. Editable free-form text box. Default value: <Element name>_<Property name> |
| Interface | Interface type. Opens an interface browser that lets you select an existing interface, page 25 of the Global Templates Library. Select a type that is compatible with the type of the property. Default value: Blank |
| Role | Role, page 27 of the interface that is connected to the property. The role can be selected only if the property has both an input and output. Otherwise, the available role automatically corresponds to the input or output. Typically, inputs use role <i>B</i> (or <i>Ref</i>), outputs use role <i>A</i> (or <i>Def</i>). The names of roles vary depending on the interface. |
| Description | Description of the interface. Optional. The string appears as a tooltip for the interface in the Asset Workspace Editor . Editable free-form text box. Default value: <Property name> |
| Element | Element, page 66 that is connected to the property. Select one element from a list of elements that the interface contains. NOTE: You can connect several properties, page 206 to different elements of the same interface. |

Interface-HAL Connection Attribute Description

The following table describes the attributes when **Connection Type** for a property is set to *Interface-HAL*. For details, about interfaces refer to the topic describing the definition of interfaces, page 157.

| Attribute | Description |
|--------------------|--|
| Identifier | <p>Identifier, page 62 of the interface and HAL facet template that are created.</p> <p>Editable free-form text box.</p> <p>Default value: <i><Element name>_<Property name></i></p> |
| Interface | <p>Interface type.</p> <p>Opens an interface browser that lets you select one of the following elementary variable interface, page 25 from the Global Templates Library:</p> <ul style="list-style-type: none"> • <i>\$Bool</i> • <i>\$Int</i> • <i>\$Real</i> <p>The software automatically creates a corresponding Hardware Abstraction Layer (HAL) signal facet, page 189.</p> <p>Default value: Interface of the same type as the property; otherwise, blank.</p> |
| Role | <p>Role, page 27 of the interface that is connected to the property.</p> <p>The role can be selected only if the property has both an input and output. Otherwise, inputs use role <i>Ref</i>, outputs role <i>Def</i>.</p> <p>NOTE: The other role is connected to the HAL signal facet, page 189.</p> |
| Description | <p>Description of the interface. Optional.</p> <p>Editable free-form text box.</p> <p>Default value: <i><Element name>_<Property name></i></p> |
| Element | <p>Element, page 66 that is connected to the property.</p> <p>Select one element from a list of elements that the interface contains.</p> <p>Typically, elementary variable interfaces contain only the <i>Name</i> element.</p> |

Binding Connection Attribute Description

The following table describes the attributes when **Connection Type** for a property is set to *Binding*. For details about bindings, refer to the topic describing the definition of bindings, page 244.

| Attribute | Description |
|---------------|--|
| Source | <p>Opens the Create Bindings dialog box, page 246 that lets you select the output of another property as the source of the binding. It will be connected to the input of the property by using a binding.</p> <p>After you select a source, the path of the source is indicated in Value.</p> <p>If you leave the value blank, the property is not connected and hidden, page 240.</p> <p>NOTE: If you have set the connection of another property to <i>Interface</i> or <i>Interface-HMI</i>, the interface that you have selected may appear in the dialog box as a possible source for the binding.</p> |

NOTE: Connecting the output of a property to the input of another element is performed by using a template editor once the template is created.

Interface-HAL Connection Type Description

When you set **Connection Type** to **Interface-HAL**, the software creates an interface and connects it to the property. It also connects the other role of this interface to an instance of a HAL template (see *EcoStruxure Process Expert, Foundation Library, Application Templates User Guide*) that is located at the parent level. The HAL template lets you make a logical connection and/or map the signal to a hardware module (hardware mapping).

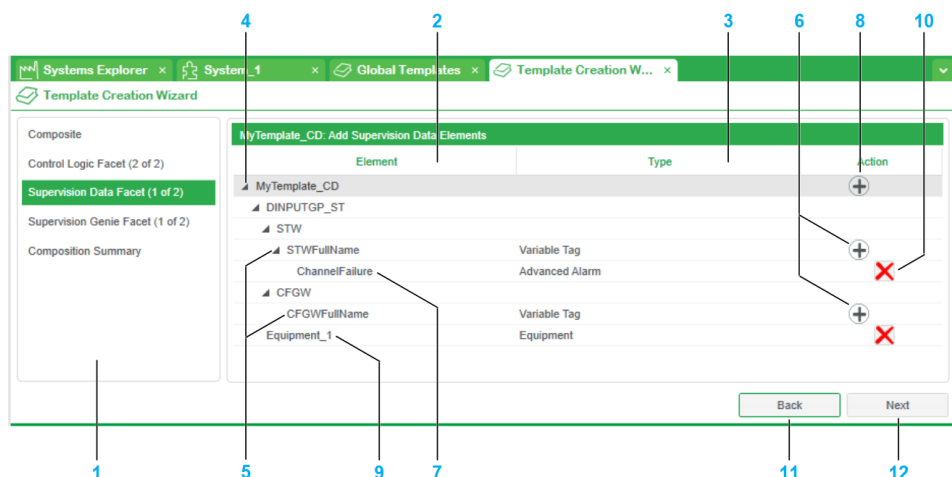
The following table describes which type of interface/role you can select and which HAL template is used to create the instance in the parent template.

| Type of the interface | Role | Inter-face element name | HAL template that is used | Template identifier | HAL template interface that is connected |
|-----------------------|------|-------------------------|---------------------------|---|--|
| \$Bool | Def | Name | \$DOSignal_UL | Value that was entered for the Identifier attribute of the connection. | DOSignal |
| | Ref | | \$DISignal_UL | | DISignal |
| \$Int | Def | | \$AOSignal_UL | | AOSignal |
| | Ref | | \$AISignal_UL | | AISignal |
| \$Real | Def | | \$AOSignal_UL | | AOSignal |
| | Ref | | \$AISignal_UL | | AISignal |

Template Creation Wizard Supervision Data Facet Window 1/2

Tag, Alarm, and Equipment Creation

The following figure shows an example of the Supervision data facet element creation window, which opens if you have selected *Control Logic Facet* and *Supervision Data Facet* in the composite window, page 180. It shows the variable tag elements that will be created based on the configuration of Control properties and lets you create additional elements.



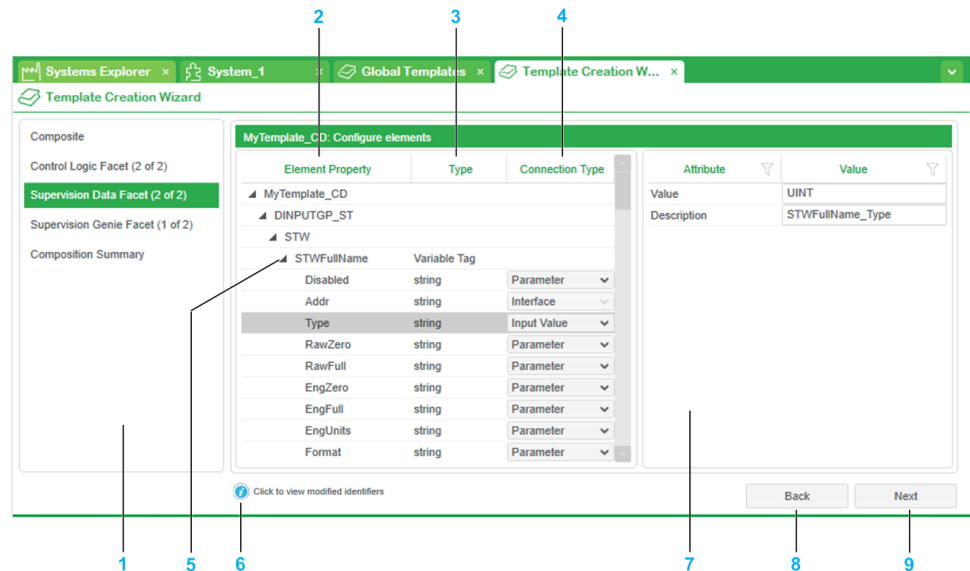
| Item | Description |
|------|---|
| 1 | Pane showing the steps of the template creation process and your progression. |
| 2 | By default, shows properties for which you have selected Interface-HMI as connection type in the Control facet element configuration, page 184 window (step 2/2). The column also shows additional tags and equipment that you create. |
| 3 | Type of element that will be created at the variable tag or Supervision data facet template level. Possible values: <ul style="list-style-type: none"> Variable Tag (default for Control element properties with Interface-HMI connection type) Advanced Alarm Digital Tag Trend Tag Equipment NOTE: To add other types of elements, page 92, use the template editors once the template has been created. |
| 4 | Identifier of the Supervision data facet template. |
| 5 | Properties for which you have selected Interface-HMI as connection type. A variable tag element is automatically created for each one with the main properties configured, page 195. |
| 6 | Lets you add one or more advanced alarm, digital alarm, and trend tag elements at the variable tag level. These tag element names are editable. NOTE: If you rename an element after having configured its properties in the next window, this configuration is discarded. |
| 7 | Advanced alarm element that has been added manually by using the <i>Add</i> button (6) of the property. |
| 8 | Lets you add one or more variable tag and one equipment, page 96 elements at the Supervision data facet template level. These tag and equipment element names are editable. |

| Item | Description |
|------|--|
| | NOTE: If you rename an element after having configured its properties in the next window, this configuration is discarded. |
| 9 | Equipment element that has been added manually by using the <i>Add</i> button (8) of the Supervision data facet template. |
| 10 | Lets you remove elements that have been added manually. |
| 11 | Lets you go back to the previous window. Tag elements that you have created in this window are retained unless you change the connection type of the corresponding properties in the previous window. |
| 12 | Opens the second Supervision Data facet window. |

Template Creation Wizard Supervision Data Facet Window

Tag, Alarm, and Equipment Configuration

The following figure shows an example of the second Supervision data facet window, which lets you configure the elements that were shown in the previous step. It gives you access to the properties of each element.



| Item | Description |
|------|--|
| 1 | Pane showing the steps of the template creation process and your progression. |
| 2 | <p>Shows the tags and equipment that were created in the previous step (1/2) and the properties of these elements, page 93.</p> <p>For each one, you can configure a type of connection for data transmission within the template and to/from other templates.</p> <p>The connections is represented graphically when you edit the templates once they are created.</p> |
| 3 | Data type of the property. |
| 4 | <p>Select a type of connection for each property.</p> <p>Possible values:</p> <ul style="list-style-type: none"> • Parameter: Default value . Creates and connects an editable parameter , page 193 to the input pin of the property. • Input Value: Creates and connects a fixed-value input, page 194 to the input pin of the property. • Interface: Default value for properties that have only an output pin. Connects the pin of the property to the element of an interface, page 194 that you select from a list of existing interfaces of the Global Templates library. You can connect several properties, page 206 to different elements of the same interface. • Binding: Connects the input of the property to the output of a property of another element within the same template by using a binding, page 195. <ul style="list-style-type: none"> NOTE: Connecting the output of a property to the input of another element is not supported and needs to be performed by using a template editor once the template is created. • Not Connected: The property is not bound and hidden, page 240. Select this connection type when no other connection type is required. <ul style="list-style-type: none"> NOTE: When you change the connection type, the attribute configuration that you made in the left-hand pane for this property is discarded. |
| 5 | Control property for which the variable tag element is created with the configured values. |
| 6 | <p>The icon appears in case a system-generated variable name exceeds 50 characters.</p> <p>Click it to open the Modified Identifiers dialog box, page 203.</p> |

| Item | Description |
|------|---|
| 7 | Lets you configure the connection that you selected for each property. NOTE: Value text boxes support copy and paste by using the mouse buttons or keyboard shortcuts (you cannot paste a value on attributes with a browse button). |
| 8 | Lets you go back to the previous window. The configuration that you made in this window is retained. However, if you remove a tag or equipment in the previous window, the corresponding property and connection configuration is discarded. |
| 9 | Opens the next window of the template creation process. |

Parameter Connection Attribute Description

The following table describes the attributes when **Connection Type** for a property is set to *Parameter*. For details about parameters, refer to the topic describing the definition of parameters, page 85.

| Attribute | Description |
|----------------------|---|
| Category | Lets you group properties by category. This grouping is used in the Parameters pane, page 85 of template editors and in the Instance Editor . Editable free-form text box. Default value: <Element name> |
| Identifier | Name of the parameter. It appears in the Instance Editor under Name . Editable free-form text box. Default value: <Element name>_<Property name> NOTE: You can connect to a property one of the following existing implicit system parameters, page 85 by entering its name in the text box: <ul style="list-style-type: none"> • \$InstanceID • \$Description • \$Area In this case, enter \$System in Category , which is the existing category for system parameters. |
| Description | Description of the parameter. The string appears for the parameter in the Instance Editor . Optional. Editable free-form text box. Default value: <Element name>_<Property name> |
| Type | Data type of the parameter. The type restricts values that can be entered for the parameter while editing the instance. Read-only. Default value: The value is type and property specific. |
| Default value | Value that is set by default for the parameter and that appears in the Instance Editor under Value . Optional. Editable free-form text box. Default value: Blank |

Input Value Connection Attribute Description

The following table describes the attributes when **Connection Type** for a property is set to *Input Value*.

| Attribute | Description |
|--------------------|---|
| Value | Fixed value that is used as input for the property and that cannot be edited during instantiation. Editable free-form text box. Default value: Blank |
| Description | Description of the value. String that appears for the input value in the Instance Editor . Optional. Editable free-form text box. Default value: <Element name>_<Property name> |

Interface Connection Attribute Description

The following table describes the attributes when **Connection Type** for a property is set to *Interface*. For details about interfaces, refer to the topic describing the definition of interfaces, page 157.

| Attribute | Description |
|--------------------|---|
| Identifier | Identifier, page 62 of the interface that is created. Editable free-form text box. Default value: <Element name>_<Property name> |
| Interface | Interface type. Opens an interface browser that lets you select an existing interface, page 25 of the Global Templates library. Select a type that is compatible with the type of the property. Default value: Blank |
| Role | Role, page 27 of the interface that is connected to the property. The role can be selected only if the property has both an input and output. Otherwise, the available role automatically corresponds to the input or output. Typically, inputs use role <i>B</i> (or <i>Ref</i>), outputs use role <i>A</i> (or <i>Def</i>). The names of roles vary depending on the interface. |
| Description | Description of the interface. Optional. The string appears as a tooltip for the interface in the Asset Workspace Editor . Editable free-form text box. Default value: <Property_name> |
| Element | Element, page 66 that is connected to the property. Select one element from a list of elements that the interface contains. NOTE: You can connect several properties, page 206 to various elements of the same interface. |

Binding Connection Attribute Description

The following table describes the attributes when **Connection Type** for a property is set to *Binding*. For details about bindings, refer to the topic describing the definition of bindings, page 244.

| Attribute | Description |
|---------------|--|
| Source | <p>Opens the Create Bindings dialog box, page 246, which lets you select the output of another property as the source of the binding. It will be connected to the input of the property by a using a binding.</p> <p>After you select a source, the path of the source is indicated in Value.</p> <p>If you leave the value blank, the property is not connected and hidden, page 240.</p> <p>NOTE: If you set the connection of another property to <i>Interface</i> or <i>Interface-HMI</i>, the interface that you selected may appear in the dialog box as a possible source for the binding.</p> |

NOTE: Connecting the output of a property to the input of another element is not supported and needs to be performed by using a template editor once the template is created.

Supervision Data Facet Element Pre-configuration

The following tables only show the attributes that are configured with a value that is different from the default value.

Variable tags

| Element property | Connection type | Attribute | Pre-configured value |
|--|-----------------------|--------------------------|-------------------------------------|
| Addr | Interface (read-only) | Interface ⁽¹⁾ | <i>\$String</i> |
| | | Role ⁽¹⁾ | <i>Ref</i> |
| | | Element ⁽¹⁾ | <i>Name</i> |
| Type | Input Value | Value ⁽¹⁾ | <i>STRING</i> |
| Comment | | | <i><Element name>_Comment</i> |
| Historize | | | <i>false</i> |
| ItemName | | | <i><Element name></i> |
| (1) For variable tags that are added at the Supervision facet level, no value is pre-configured. | | | |

Trend tags

| Element property | Connection type | Attribute | Pre-configured value |
|-------------------|--------------------|-------------------|---|
| Expr | Input Value | Value | <i><Variable tag name></i> |
| SamplePer | | | <i>00:00:05</i> |
| Type | | | <i>TRN_EVENT</i> |
| Comment | | | <i><Trend tag name>_Comment</i> |
| FileName | | | <i>[Data]:<Variable tag name></i> |
| StorMethod | | | <i>Floating Point (8-byte samples)</i> |
| Area | Parameter | Identifier | <i>\$Area</i> |
| Files | Input Value | Value | <i>5</i> |

| Element property | Connection type | Attribute | Pre-configured value |
|------------------|-----------------|-----------|----------------------|
| Time | | | 12:00:00 |
| Period | | | <i>Tuesday</i> |
| Historize | | | <i>false</i> |
| ItemName | | | <Trend tag name> |

Advanced Alarms

| Element property | Connection type | Attribute | Pre-configured value |
|------------------|-----------------|------------|----------------------------------|
| Area | Parameter | Identifier | \$Area |
| Comment | Input Value | Value | <Alarm element name>_ Comment |
| Historize | | | <i>false</i> |
| ItemName | | | <Alarm element name> |

Digital Alarms

| Element property | Connection type | Attribute | Pre-configured value |
|------------------|-----------------|------------|----------------------------------|
| Area | Parameter | Identifier | \$Area |
| Comment | Input Value | Value | <Alarm element name>_ Comment |
| Historize | | | <i>false</i> |
| ItemName | | | <Alarm element name> |

Equipment

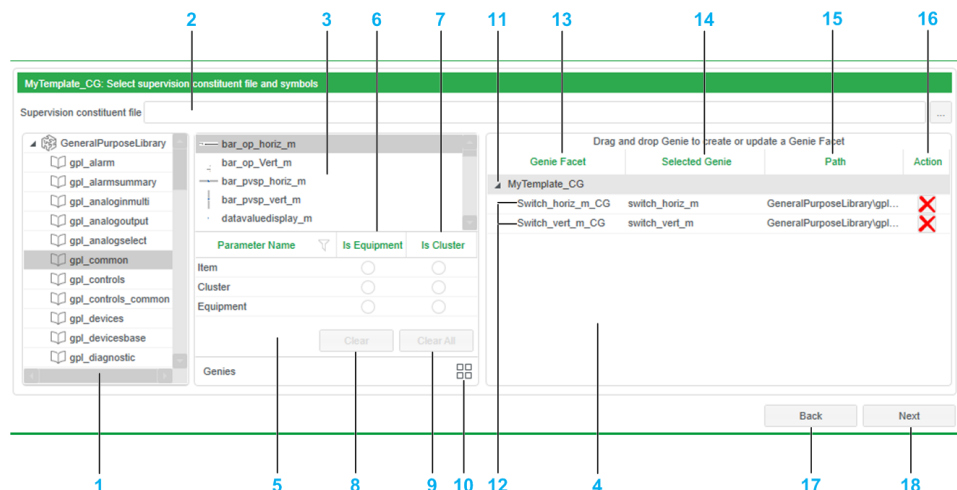
| Element property | Connection type | Attribute | Pre-configured value |
|------------------|-----------------|------------|------------------------|
| DisplayName | Parameter | Identifier | \$InstanceID |
| Comment | Input Value | Value | <Element name>_Comment |

Template Creation Wizard Genie Facet Window 1/2

Genie Selection

The following figure shows an example of the main section of the Supervision Genie facet window that is shown if you have selected the *Supervision Genie Facet* in the composite window, page 180. It lets you select Genies from project files (.ctz) that exist in the content repository or that you are adding.

NOTE: For Genies of an Include project of the GPL, it lets you also configure the *Is Equipment* and *Is Cluster* parameters, page 280.



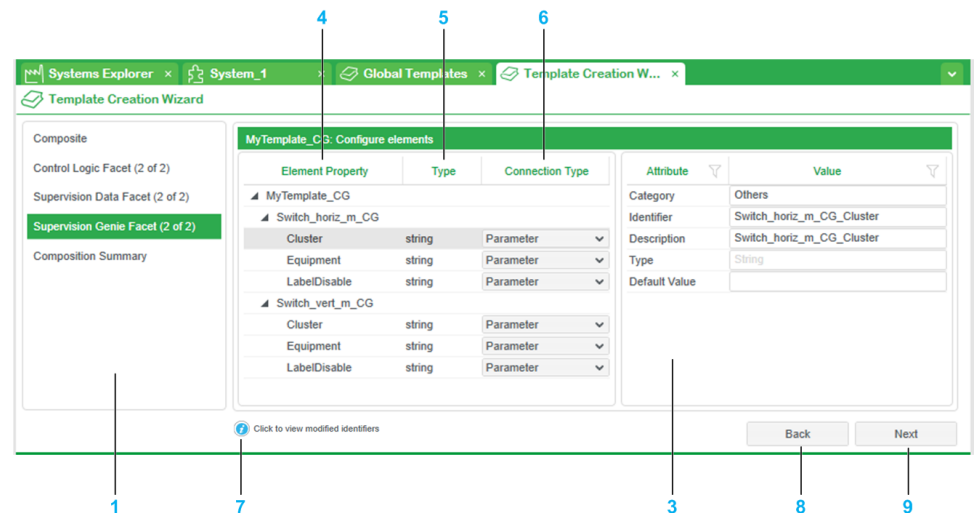
| Item | Description |
|------|--|
| 1 | Pane showing the project files (Include projects, .ctz) that are present in the content repository and the libraries they contain. |
| 2 | Lets you add a project file (.ctz), page 281 to the content repository. Once added, you can view its libraries and select a Genie. |
| 3 | Center pane showing the Genies that the selected library contains and that you can encapsulate. NOTE: The window does not allow creating or editing Genies, page 272. You need to do so before opening the wizard. |
| 4 | Genie facet pane showing the Genies that will be encapsulated as elements in the Supervision Genie facet template. |
| 5 | Pane showing the parameters that are available for a Genie that is selected. Parameters that you do not configure here can be configured in the next window (2/2). NOTE: The parameter selection applies to the Genie that is selected in the center pane and not to the Genie that is selected in the Genie facet pane. |
| 6 | Lets you select which parameter contains the equipment name. It corresponds to the value of the Name property, page 280 of the equipment instance. You can select only one parameter and it cannot be selected if it is already used for another property (for example, Is Cluster). Selecting a parameter is optional. |
| 7 | Lets you select which parameter contains the cluster name. It corresponds to the value of the Cluster Name property, page 280 of the equipment instance. You can select only one parameter and it cannot be selected if it is already used for another property (for example, Is Equipment). Selecting a parameter is optional. |
| 8 | Clears the selection for the parameter that is selected. |
| 9 | Clears the selection for the parameters of the selected Genie. |
| 10 | Toggles the display mode of the Genies pane between grid view and icon view. |
| 11 | Identifier of the Supervision Genie facet template that will contain the Genie facets. |

| Item | Description |
|------|---|
| 12 | <p>One row is created for each Genie that you have selected in the center pane and dragged onto the header that displays the <i>name of the Supervision Genie facet template</i> (11).</p> <p>NOTE: If you drag a Genie onto an existing row, it replaces this Genie with your new selection after your confirm the change. The name of the Genie facet, however, does not change.</p> |
| 13 | <p>Identifier of the facet template that will contain the Genie that you have selected to be encapsulated.</p> <p>By default, it is the identifier of the Genie.</p> <p>You can edit the identifier, which must satisfy applicable naming rules.</p> |
| 14 | Identifier of the Genie that will be encapsulated. |
| 15 | Path of the selected Genie in the project file (.ctz). |
| 16 | <p>Lets you remove the Genie from the facet template.</p> <p>This does not remove the Genie from the project file.</p> |
| 17 | <p>Lets you go back to the previous window.</p> <p>The selections you have made in this window are retained.</p> |
| 18 | Opens the next window, which lets you configure the properties (parameters) of the Genies. |

Template Creation Wizard Genie Facet Window 2/2

Genie Parameter Configuration

The following figure shows an example of the Supervision Genie facet window that is shown if you have selected the *Supervision Genie Facet* in the composite window, page 180. It lets you configure properties of Genies that you have not configured in the previous in step 1/2.



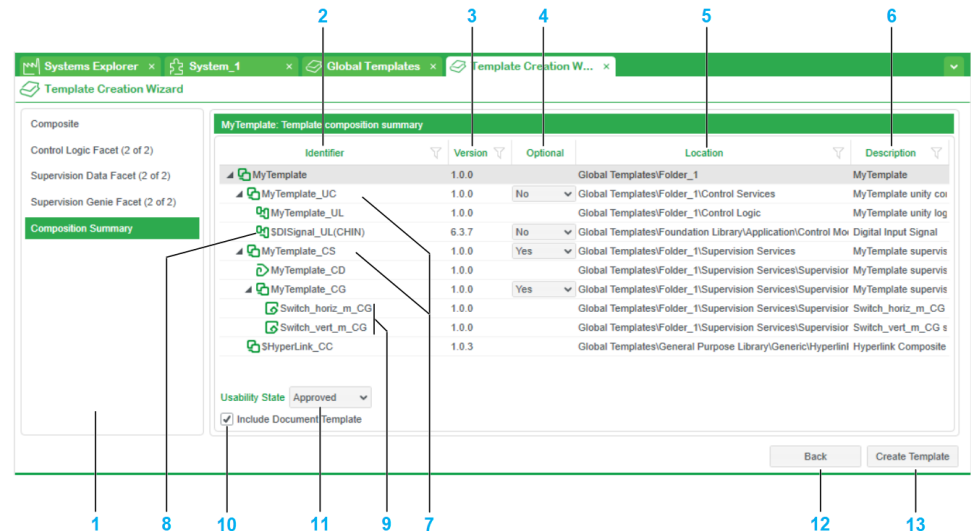
| Item | Description |
|------|--|
| 1 | Pane showing the steps of the template creation process and your progression. |
| 2 | Pane showing the Genies that you have selected in the previous step (1/2) and their properties. |
| 3 | Lets you configure the connection that has been selected for each Genie property. NOTE: Value text boxes support copy/paste by using the mouse buttons or keyboard shortcuts (you cannot paste a value on attributes with a browse button). |
| 4 | Each Genie is shown as a node. For each Genie, only the properties that you have not configured in step 1/2 are shown. |
| 5 | Data type of the property. |
| 6 | Select a type of connection for each property. Possible values: <ul style="list-style-type: none"> Parameter: Default value . Creates and connects an editable parameter, page 193 to the input pin of the property. Input Value: Creates and connects a fixed-value input, page 194 to the input pin of the property. Interface: Connects the pin of the property to the element of an interface, page 194 that you select from a list of existing interfaces of the Global Templates library. You can connect several properties, page 206 to different elements of the same interface. Binding: Connects the input of the property to the output of a property of another element within the same template by using a binding, page 195. NOTE: Connecting the output of a property to the input of another element is not supported and needs to be performed by using a template editor once the template is created. Not Connected: The property is not bound and hidden, page 240. Select this connection type when no other connection type is required. NOTE: When you change the connection type, the attribute configuration that you have made in the left-hand pane for this property is discarded. |
| 7 | The icon appears in case a system-generated variable name exceeds 50 characters. Click it to open the Modified Identifiers dialog box, page 203. |

| Item | Description |
|------|---|
| 8 | Lets you go back to the previous window. The selections you have made in this window are retained unless you remove the Genie or clear its parameter selection in the previous window. |
| 9 | Opens the last window of the template creation process. |

Template Creation Wizard Composition Summary Window

Template Composition Window Description

The following figure shows an example of the template composition summary window that is shown in the last step. It lets you view the structure of the template, page 178, finalize and create it.



| Item | Description |
|------|---|
| 1 | Pane showing the steps of the template creation process and your progression. |
| 2 | <p>Identifiers of the composite and facet templates that the software will create or use based on your selections in the previous windows.</p> <p>The column reflects the hierarchy of the template. That is to say, templates that appear as children of other templates are referenced by these templates.</p> <p>NOTE: Identifiers of instances of these templates have the <code>_X</code> suffix (where X is an integer) except for the HAL facet templates.</p> |
| 3 | <p>Version of the templates.</p> <p>By default, templates that the software creates have version 1.0.0.</p> <p>For existing templates, the highest available version with usability state <i>Approved</i> is used.</p> |
| 4 | <p>Lets you make the template optional, page 32 or always generate its constituents. When optional, a check box lets you select the functionality provided by the template when you edit the instance in the Instance Editor.</p> <p>The option is available only for the following templates:</p> <ul style="list-style-type: none"> The <code>_UC</code> Control composite template that regroups the Control functionality The HAL facet templates The <code>_CS</code> Supervision composite template that regroups the Supervision functionality The <code>_CG</code> Genie facet template that regroups the Genie facets |
| 5 | Locations in the Global Templates Library where the templates are stored. |
| 6 | <p>Editable description of the template.</p> <p>Free-form text.</p> |
| 7 | Control (<code>_UC</code>) and Supervision (<code>_CS</code>) composite templates, page 178 that are automatically created to regroup the various Control and Supervision facet templates in the same way as it is done for Schneider Electric templates. |
| 8 | Example of HAL facet template that is automatically created for properties of Control elements, page 184 with connection type set to Interface-HAL . The string in brackets is the value that was entered for the Identifier attribute of the connection and it is the identifier of the HAL template instance. |
| 9 | Genie facets each containing one Genie. |

| Item | Description |
|------|--|
| 10 | <p>Creates a <i>\$HyperLink_CC</i> composite template, page 178, which allows opening documents (see <i>EcoStruxure Process Expert, Runtime Navigation Services, User Guide</i>) stored in the content repository and web pages when using Runtime Navigation Services.</p> <p>The <i>Documents</i> and <i>URL</i> child templates are optional and each one contains five <i>CrDocument</i> and <i>Url</i> functions respectively, which you can configure at the instance level.</p> |
| 11 | <p>Usability state, page 82 of the composite template.</p> <p>Possible values:</p> <ul style="list-style-type: none">• Approved• Not Approved <p>The usability state of child templates is always <i>Approved</i>.</p> |
| 12 | <p>Lets you go back to the previous window.</p> <p>The selections you have made in this window are retained. However, modifications that you make to the configuration in the previous windows may impact your selections.</p> |
| 13 | <p>Closes the wizard, creates the templates, and opens the folder containing the composite template in the Global Templates Explorer.</p> <p>NOTE: You cannot modify the composite template by using the wizard. To modify it, use the various template editors.</p> |

Identifier Management

Modified Identifiers Dialog Box

The following figure shows an example of the dialog box that opens when you click the information icon in one of the control, data, or Genie facet configuration windows. Data shown in this dialog box is read-only.

| Variable Name | Modified Identifier | Default Generated Identifier | Variable Path |
|---------------|---|--|--|
| ItemName | DINPUTGP_ST_STW_STWFullName_ChannelFailure_Item | DINPUTGP_ST_STW_STWFullName_ChannelFailure_ItemName | DINPUTGP_ST_STW_STWFullName_ChannelFailure_ItemName |
| Historize | DINPUTGP_ST_STW_STWFullName_ChannelFailure_Hist | DINPUTGP_ST_STW_STWFullName_ChannelFailure_Historize | DINPUTGP_ST_STW_STWFullName_ChannelFailure_Historize |
| Category | DINPUTGP_ST_STW_STWFullName_ChannelFailure_Cate | DINPUTGP_ST_STW_STWFullName_ChannelFailure_Category | DINPUTGP_ST_STW_STWFullName_ChannelFailure_Category |
| Disabled | DINPUTGP_ST_STW_STWFullName_ChannelFailure_Disa | DINPUTGP_ST_STW_STWFullName_ChannelFailure_Disabled | DINPUTGP_ST_STW_STWFullName_ChannelFailure_Disabled |

| Item | Description |
|------|--|
| 1 | Name of the property for which you have configured a connection and this connection generates a variable name exceeding 50 characters. |
| 2 | The new variable name that has been truncated to satisfy applicable naming rules. |
| 3 | Original variable name exceeding 50 characters. |
| 4 | Lets you locate the property that generates the variable. |

NOTE:

If you edit the name of a property that is included in the identifier, for example, because you prefer creating your own shorter identifier instead of the truncated one, when you reopen the **Modified Identifiers** dialog box, this identifier does not appear anymore.

For the truncated identifiers shown in the above dialog box, you can edit to shorten it, for example, the name of the advanced alarm tag in the Supervision data facet window 1/2, page 190 from *ChannelFailure* to *ChFailure*.

This will generate a variable name of less than 50 characters.

If you do so, when you reopen the dialog box, no more notifications appear for these identifiers.

(If no other identifiers have been truncated, the information icon itself does not appear.)

Creating Templates by Using the Wizard

Prerequisites

The following are prerequisites to creating templates by using the Wizard:

- A user associated to a profile with the necessary rights.
- To create a Control logic facet, you have a project file that fulfills the following requirements:
 - File format: .stu or .zef
If you select an STU file, it was exported from a version of Control Expert that is compatible with the version of the Control Participant.
 - It contains the Control resources that you want to encapsulate in the facet template.
You can also add other constituents, page 260 later by editing the template.
- To create a Supervision Genie facet, the Genie is contained in an Include project file that is either present in the content repository (see *EcoStruxure Process Expert, User Guide*) or that you add to the content repository while using the wizard. You can only add a project file that was created with a version of AVEVA Plant SCADA that is the same as the Supervision Participant.

Template Creation Procedure

To create a composite template with Control and Supervision functionality by using the wizard, proceed as follows.

| Step | Action |
|------|--|
| 1 | In the Global Templates Explorer , right-click a folder and select Template Creation Wizard . Result: The wizard opens. |
| 2 | Enter an identifier for the composite template, select child templates, and click Next . Result: The first Control facet window opens. NOTE: If Control or Supervision facets are not required, skip the corresponding steps in this procedure. |
| 3 | For a template with Control logic, select the project file that contains the Control constituents. |
| 4 | Select the constituents (function blocks and/or variables) that you want to encapsulate as elements in the template and click Next . Result: The second Control facet window opens. |
| 5 | Select a connection type for the properties of each element and configure them. Then, click Next . Result: The first Supervision data facet window opens. |
| 6 | Verify that the necessary variable tags are shown and create alarm tags, equipment, and additional variable tags as needed. Then, click Next . Result: The second Supervision data facet window opens. |
| 7 | Select a connection type for the properties of tags and equipment that you want to use and configure them. Then, click Next . Result: The first Supervision Genie facet window opens. |
| 8 | Select an Include project and a library. If required, add a compatible Include project. Result: The Genies that it contains are shown in the center pane. |
| 9 | Select a Genie, select its parameters, and drag it to the gray header showing the name of the Supervision Genie facet in the right-hand pane. Repeat steps 8 and 9 to encapsulate additional Genies. |
| 10 | Click Next . Result: The second Supervision Genie facet window opens. NOTE: You can perform the parameter selection before or after dragging the Genie to the right-hand pane. The parameter selection applies to the Genie that is selected in the center pane and not to the Genie that is selected in the Genie facet pane (left). NOTE: If you drag a Genie to an existing row, it replaces the Genie with your new selection after you confirm the change. The name of the Genie facet, however, is not updated. |
| 11 | Select a connection type for the properties of Genies that you want to use and configure them. Then, click Next . Result: The composition summary window opens (last window). NOTE: When using Genies of the EcoStruxure Process Expert General Purpose Library (GPL), this step is not required because the software automatically configures the necessary values for the <i>Equipment</i> and <i>Cluster</i> parameters given you have configured them when you selected the Genie in the first Supervision Genie facet window. In this case, set the connection type of the other parameters to Not Connected if you do not need them. |
| 12 | Verify the composition of the composite template, finalize the configuration (document template, optional and usability states, descriptions), and click Create Template . Result: A confirmation message opens. NOTE: If the software detects a template duplication, a dialog box opens indicating the impacted template and version. The wizard opens again at the Composite window where you can make the necessary modifications to the template identifier and/or version. |

| Step | Action |
|------|---|
| | NOTE: Once the template is created, you cannot open the wizard for this template anymore. To modify it, use the template editors. |
| 13 | Click OK . Result: The wizard closes. The composite and its child templates are created. The folder containing the composite template opens in the Global Templates Explorer and the template is highlighted. |
| 14 | To view and/or modify a template, right-click it and select Edit . |
| 15 | Validate the template, page 318 and modify it if required. |

Connecting Several Properties to Elements of a Same Interface

When an interface contains several elements, you can connect each one to another property of the Control resource. Proceed as follows.

| Step | Action |
|------|--|
| 1 | In the Control element configuration window (step 2/2), select a first property and set Connection Type to <i>Interface</i> , page 184. |
| 2 | In the left pane, select an Interface and configure the attributes of the interface (Identifier , Role , and Element). |
| 3 | In the right pane, select a second property and set Connection Type to <i>Interface</i> as well. |
| 4 | In the left pane, select the same Interface as for the first property. |
| 5 | Enter the same Identifier and select the same Role as for the first property. |
| 6 | Select an Element that is not yet connected. |

Template Creation Example

Overview

The following topics describe how to create a simple template that is similar to the template, page 178 shown at the beginning of this chapter except that it does not contain a *DOSignal_UL* HAL template.

The template has a digital input functionality.

The step-by step creation process that is illustrated is only an example.

Template Identification and Functionality

The default settings are used in the **Composite** window of the wizard for the template identifier, version, and functionality (Control as well as Supervision Data and Genies).

Select template composition

Identifier: Template_1

Version: 1.0.0

Location: Global Templates\Folder_1

| Type | Identifier | Version | Location |
|---|---------------|---------|--|
| <input checked="" type="checkbox"/> Control Logic Facet | Template_1_UL | 1.0.0 | Global Templates\Folder_1\Control Services\Control Logic |
| <input checked="" type="checkbox"/> Supervision Data Facet | Template_1_CD | 1.0.0 | Global Templates\Folder_1\Supervision Services\Supervision Data |
| <input checked="" type="checkbox"/> Supervision Genie Facet | Template_1_CG | 1.0.0 | Global Templates\Folder_1\Supervision Services\Supervision Genie |

Description of the change: My template creation description.

Instantiated Template

The following figure shows an instance of the template open in the **Instance Editor**. The additional *_UC* and *_CS* templates are automatically created by the software while the *DI_Signal* template is created based on the user configuration.

Template_1_2

System_1 : Application 'Template_1_2' : Instance Editor

| Name | Description | Template |
|-----------------------|--|----------------|
| Template_1_2 (1/42) | Template_1 | Template_1 |
| Template_1_UC_1 (0/7) | Template_1 unity composite | Template_1_UC |
| Template_1_UL_1 (0/1) | Template_1 unity logic | Template_1_UL |
| DI_Signal (0/6) | Digital Input Signal | \$DI_Signal_UL |
| Template_1_CS_1 (0/2) | Template_1 supervision composite | Template_1_CS |
| Template_1_CD_1 (0/2) | Template_1 supervision data | Template_1_CD |
| Template_1_CG_1 (0/0) | Template_1 supervision genie composite | Template_1_CG |
| HyperLink_1 (0/30) | Hyperlink Composite of Documents and URL | \$HyperLink_CC |
| Documents (0/20) | User defined Documents Composite | \$Document_CC |
| URL (0/10) | User defined URL Composite | \$URL_CC |

Template Creation Example – Control Logic Facet 1/2

Control Constituent Selection

In this step, a control project file (.zef) is loaded and the constituents that you want to include in the template are selected. The window shows the Control constituents contained in the project file. Those that have been selected or cleared are highlighted in yellow for the purpose of the example.

Template_1_UL: Select control constituent file and variables

Control constituent file C:\Users\User1\Downloads\GT Wizard\input.zef

Sections

| Name | Type |
|--------|------|
| CM_FBD | FBD |

Variables

| Name | Type |
|---------------|---------------|
| DINPUTGP_ST | DInputGP_ST_I |
| DINPUTGP_ENGP | DInputGP_EngF |

| Name | Type |
|--|--------------------|
| <input checked="" type="checkbox"/> DINPUTGP | DINPUTGP |
| <input type="checkbox"/> SHMIVariable | BOOL |
| <input type="checkbox"/> \$In.InitDFB | BOOL |
| <input checked="" type="checkbox"/> \$In.ChIn | BOOL |
| <input type="checkbox"/> \$In.ChFailure | BOOL |
| <input type="checkbox"/> \$In.Watch | BOOL |
| <input type="checkbox"/> \$In.EngParam | DInputGP_EngPar... |
| <input type="checkbox"/> \$In.DinputGP_ST | DInputGP_ST_DDT |
| <input type="checkbox"/> \$Out.DinputGP_ST | DInputGP_ST_DDT |
| <input type="checkbox"/> \$Out.PV | BOOL |
| <input type="checkbox"/> \$Out.Ch | BOOL |
| <input type="checkbox"/> \$Out.Alarm | BOOL |
| <input type="checkbox"/> \$Out.Normal | BOOL |
| <input type="checkbox"/> \$Public.SC | DInputGP_SC_DDT |
| <input checked="" type="checkbox"/> DINPUTGP_ST | DInputGP_ST_DDT |
| <input type="checkbox"/> SHMIVariable | BOOL |
| <input checked="" type="checkbox"/> STW | WORD |
| <input type="checkbox"/> \$STW.\$InitValue | WORD |
| <input type="checkbox"/> \$STW.\$Comment | STRING |
| <input checked="" type="checkbox"/> \$STW.\$FullName | STRING |
| <input checked="" type="checkbox"/> CFGW | WORD |
| <input type="checkbox"/> \$CFGW.\$InitValue | WORD |
| <input type="checkbox"/> \$CFGW.\$Comme | STRING |
| <input checked="" type="checkbox"/> \$CFGW.\$FullNam | STRING |

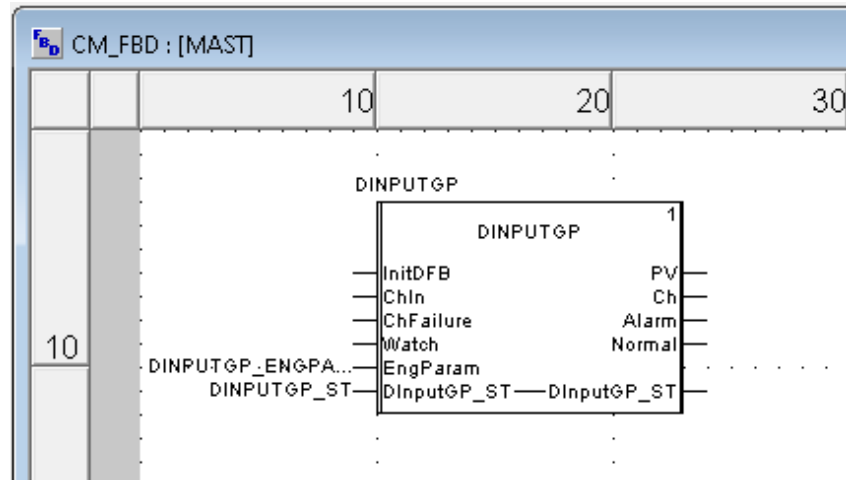
Global Types

| Name | Version |
|---------------------|---------|
| DINPUTGP | 0.03 |
| DInputGP_ST_DDT | 0.17 |
| DInputGP_SC_DDT | 0.13 |
| DInputGP_EngPar_DDT | 0.06 |

Section Size : 24 x 12 Extension 0

Control Project File Content

The following figures show the content of the project file that was selected in this step.



The Data Editor window displays a list of variables and their properties. The table below represents the data shown in the screenshot.

| Name | Type | Value | Comment | HMI variable |
|--------------------|---------------------|-------|---|--------------|
| DINPUTGP_ENGPARAM | DinputGP_EngPar_DDT | | | |
| BadPV | BOOL | | Value to be considered (EU) when channel fails | |
| AlarmSt | BOOL | | Setpoint determines the alarm condition (1 = High PV is an alarm condition) | |
| TOOnSP | TIME | | Timer to activate digital input connection (0s = Disabled) | |
| TOOffSP | TIME | | Timer to deactivate digital input connection (0s = Disabled) | |
| DINPUTGP_ST | DinputGP_ST_DDT | | | |
| STW | WORD | | Status word (Read-only) | |
| State | BOOL | | 1 = Input state is high | |
| AlarmActive | BOOL | | 1 = Alarm is detected | |
| ChFailure | BOOL | | 1 = Input channel is not operational | |
| AlarmWatch | BOOL | | 1 = Enables alarm monitoring | |
| AlarmSetpoint | BOOL | | 1 = High PV is an alarm condition | |
| InputValue | BOOL | | Input channel value (1 = Channel input is active) | |
| CFGW | WORD | | Configuration word (Read/Write) | |
| AlarmEnable | BOOL | | 1 = Enables alarm detection | |
| Override | BOOL | | 1 = Enables override input functionality | |
| OverrideInputValue | BOOL | | 1 = Input value is active | |
| Maint | BOOL | | 1 = Maintenance mode active | |

Template Creation Example – Control Logic Facet 2/2

Control Property Configuration

In this step, for each Control element, connections are configured for the relevant properties so that variables are generated with the required names and can be propagated to Supervision elements of the template. Properties that are not used are set to *Not Connected*.

In this example, the following configurations are made:

| Property | Selected connection type | Configured connection attributes | Result |
|-----------|--------------------------|--|--|
| \$In.ChIn | Interface-HAL | <ul style="list-style-type: none"> Identifier: Optional Role: Ref is required for an input <p>For the other attributes, default values are used.</p> | Automatically creates and connects a digital input HAL facet template. |

Template_1_UL: Configure elements

| Element Property | Type | Connection Type |
|-------------------|-----------------|-----------------|
| ▲ DINPUTGP | DINPUTGP | |
| Disable | String | Not Connected ▼ |
| Comment | String | Not Connected ▼ |
| Custom | String | Not Connected ▼ |
| DFBType | String | Parameter ▼ |
| \$In.ChIn | BOOL | Interface-HAL ▼ |
| ▲ DINPUTGP_ST | DInputGP_ST_DDT | |
| Disable | String | Not Connected ▼ |
| Comment | String | Not Connected ▼ |
| Custom | String | Not Connected ▼ |
| Address | String | Not Connected ▼ |
| ▲ STW | WORD | |
| \$STW.\$FullName | STRING | Interface-HMI ▼ |
| ▲ CFGW | WORD | |
| \$CFGW.\$FullName | STRING | Interface-HMI ▼ |

| Attribute | Value |
|-------------|---------------------|
| Identifier | DI_Signal |
| Interface | \$BBool (1.0.0) ... |
| Role | Ref ▼ |
| Description | DINPUTGP_InChIn |
| Element | Name ▼ |

| Property | Selected connection type | Configured connection attributes | Result |
|----------------------|--------------------------|---|---|
| \$STW. \$FullName | Interface-HMI | <ul style="list-style-type: none"> Interface: \$String is selected <p>For the other attributes, default values are used.</p> | <ul style="list-style-type: none"> Connects an interface that propagates the variable name to the Supervision elements. Creates a variable tag element. |

Template_1_UL: Configure elements

| Element Property | Type | Connection Type |
|-------------------|-----------------|-----------------|
| ▲ DINPUTGP | DINPUTGP | |
| Disable | String | Not Connected ▼ |
| Comment | String | Not Connected ▼ |
| Custom | String | Not Connected ▼ |
| DFBType | String | Parameter ▼ |
| \$In.ChIn | BOOL | Interface-HAL ▼ |
| ▲ DINPUTGP_ST | DInputGP_ST_DDT | |
| Disable | String | Not Connected ▼ |
| Comment | String | Not Connected ▼ |
| Custom | String | Not Connected ▼ |
| Address | String | Not Connected ▼ |
| ▲ STW | WORD | |
| \$STW.\$FullName | STRING | Interface-HMI ▼ |
| ▲ CFGW | WORD | |
| \$CFGW.\$FullName | STRING | Interface-HMI ▼ |

| Attribute | Value |
|-------------|-----------------------------|
| Identifier | DINPUTGP_ST_STW_STWFullName |
| Interface | \$String (1.0.0) ... |
| Role | Def ▼ |
| Description | STW_STWFullName |
| Element | Name ▼ |

| Property | Selected connection type | Configured connection attributes | Result |
|-----------------------|--------------------------|---|---|
| \$CFGW. \$FullName | Interface-HMI | <ul style="list-style-type: none"> Interface: \$String is selected <p>For the other attributes, default values are used.</p> | <ul style="list-style-type: none"> Connects an interface that propagates the variable name to the Supervision elements. Creates a variable tag element. |

Template_1_UL: Configure elements

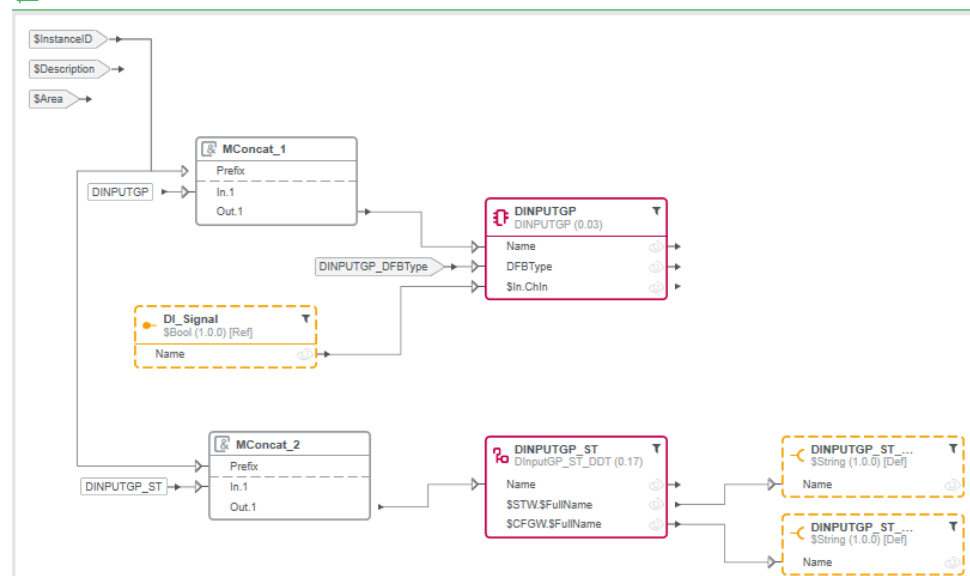
| Element Property | Type | Connection Type |
|-------------------|-----------------|-----------------|
| ▲ DINPUTGP | DINPUTGP | |
| Disable | String | Not Connected ▼ |
| Comment | String | Not Connected ▼ |
| Custom | String | Not Connected ▼ |
| DFBType | String | Parameter ▼ |
| \$In.ChIn | BOOL | Interface-HAL ▼ |
| ▲ DINPUTGP_ST | DinputGP_ST_DDT | |
| Disable | String | Not Connected ▼ |
| Comment | String | Not Connected ▼ |
| Custom | String | Not Connected ▼ |
| Address | String | Not Connected ▼ |
| ▲ STW | WORD | |
| \$STW.\$FullName | STRING | Interface-HMI ▼ |
| ▲ CFGW | WORD | |
| \$CFGW.\$FullName | STRING | Interface-HMI ▼ |

| Attribute | Value |
|-------------|-------------------------------|
| Identifier | DINPUTGP_ST_CFGW_CFGWFullName |
| Interface | \$String (1.0.0) ... |
| Role | Def ▼ |
| Description | CFGW_CFGWFullName |
| Element | Name ▼ |

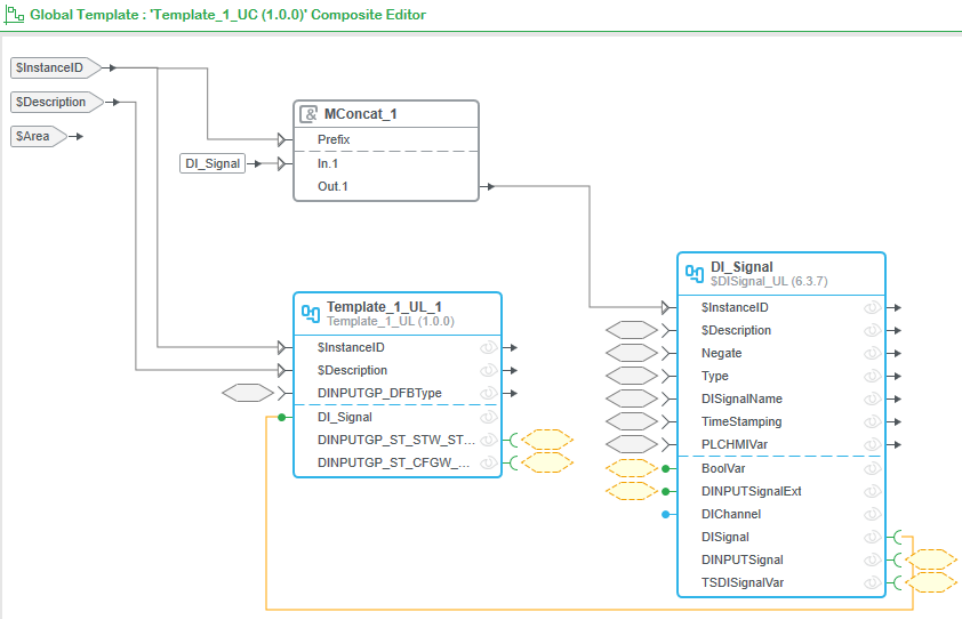
Resulting Templates

The following figure shows the Control elements and connections that are the result of configurations performed in Control logic facet windows 1/2 and 2/2. They are created in the Control facet template (_UL) once the wizard is closed.

Global Template : 'Template_1_UL (1.0.0)' Facet Editor



The following figure shows the *DI_Signal* HAL template that is created at the same level as the *_UL* template. Both are referenced in a composite template (*_UC*) that is automatically created to regroup the Control functionality of the template.



Template Creation Example – Supervision Data Facet 1/2

Variable Tag, Advanced Alarm, and Equipment Creation

In this step, the *ChFailure* advanced alarm tag and the *Equipment_1* equipment have been added manually. *ChFailure* is the edited name of the alarm tag.

The *STWFullName* and *CFGWFullName* variable tags are the result of the configuration of *Interface-HMI* connections in the previous step.

| Template_1_CD: Add Supervision Data Elements | | |
|--|----------------|--------|
| Element | Type | Action |
| Template_1_CD | | + |
| DINPUTGP_ST | | |
| STW | | |
| STWFullName | Variable Tag | + |
| ChFailure | Advanced Alarm | ✗ |
| CFGW | | |
| CFGWFullName | Variable Tag | + |
| Equipment_1 | Equipment | ✗ |

Template Creation Example – Supervision Data Facet 2/2

In this step, the connections of properties of variable tags, advanced alarm tags, and equipment from the previous step are configured to generate the necessary data for the Supervision Participant project.

Variable Tag Configuration

The following figure shows which properties of the *STWFullName* variable tag are configured and the corresponding attribute values. For other properties, default values are used.

NOTE: In the wizard, only one property can be selected and configured at a time.

Template_1_CD: Configure elements

| Element Property | Type | Connection Type |
|------------------|--------------|-----------------|
| Template_1_CD | | |
| DINPUTGP_ST | | |
| STW | | |
| STWFullName | Variable Tag | |
| Disabled | string | Parameter |
| Addr | string | Interface |
| Type | string | Input Value |
| RawZero | string | Parameter |
| RawFull | string | Parameter |
| EngZero | string | Parameter |
| EngFull | string | Parameter |
| EngUnits | string | Parameter |
| Format | string | Parameter |
| Deadband | string | Parameter |
| Comment | string | Input Value |
| Custom1 | string | Parameter |
| Custom2 | string | Parameter |
| Custom3 | string | Parameter |
| Custom4 | string | Parameter |
| Custom5 | string | Parameter |
| Custom6 | string | Parameter |
| Custom7 | string | Parameter |
| Custom8 | string | Parameter |
| Historize | string | Input Value |
| ItemName | string | Input Value |

| Attribute | Value |
|-------------|--------------------------|
| Value | UINT |
| Description | STWFullName_Type |
| Value | @(Status Word Bit Array) |
| Description | STWFullName_Comment |
| Value | StatusWord |
| Description | STWFullName_ItemName |

The following figure shows which properties of the *CFGWFullName* variable tag are configured and the corresponding attribute values. For other properties, default values are used.

NOTE: In the wizard, only one property can be selected and configured at a time.

Template_1_CD: Configure elements

| Element Property | Type | Connection Type |
|------------------|--------|-----------------|
| CFGWFullName | | |
| Disabled | string | Parameter |
| Addr | string | Interface |
| Type | string | Input Value |
| RawZero | string | Parameter |
| RawFull | string | Parameter |
| EngZero | string | Parameter |
| EngFull | string | Parameter |
| EngUnits | string | Parameter |
| Format | string | Parameter |
| Deadband | string | Parameter |
| Comment | string | Input Value |
| Custom1 | string | Parameter |
| Custom2 | string | Parameter |
| Custom3 | string | Parameter |
| Custom4 | string | Parameter |
| Custom5 | string | Parameter |
| Custom6 | string | Parameter |
| Custom7 | string | Parameter |
| Custom8 | string | Parameter |
| Historize | string | Input Value |
| ItemName | string | Input Value |

| Attribute | Value |
|-------------|---------------------------------|
| Value | UINT |
| Description | CFGWFullName_Type |
| Value | @(Configuration Word Bit Array) |
| Description | CFGWFullName_Comment |
| Value | ConfigurationWord |
| Description | CFGWFullName_ItemName |

Advanced Alarm Configuration

The following figure shows which properties of the *ChFailure* advanced alarm tag are configured and the corresponding attribute values. For other properties, default values are used.

NOTE: In the wizard, only one property can be selected and configured at a time.

Template_1_CD: Configure elements

| Element Property | Type | Connection Type | Attribute | Value |
|------------------|----------------|-----------------|-------------|--|
| ChFailure | Advanced Alarm | | Value | ChannelFailure |
| Disabled | string | Parameter | Description | ChFailure_Name |
| Name | string | Input Value | Value | @(Channel Failure Alarm) |
| Desc | string | Input Value | Description | ChFailure_Desc |
| Expr | string | Input Value | Value | GPL_BitValueCheck(<ChannelFailureVariableTagName>,2) |
| Category | string | Input Value | Description | ChFailure_Expr |
| Area | string | Parameter | Value | 4004 |
| Priv | string | Parameter | Description | ChFailure_Category |
| Help | string | Parameter | Value | 00:00:00 |
| Comment | string | Input Value | Description | ChFailure_Delay |
| Delay | string | Input Value | | |
| Custom1 | string | Parameter | | |
| Custom2 | string | Parameter | | |
| Custom3 | string | Parameter | | |
| Custom4 | string | Parameter | | |
| Custom5 | string | Parameter | | |
| Custom6 | string | Parameter | | |
| Custom7 | string | Parameter | | |
| Custom8 | string | Parameter | | |
| Paging | string | Parameter | | |
| PagingG | string | Parameter | | |
| Historize | string | Input Value | | |
| ItemName | string | Input Value | Value | ChFailure |
| | | | Description | ChFailure_ItemName |

Equipment Configuration

The following figure shows which properties of the *Equipment_1* equipment are configured and the corresponding attribute values. For other properties, default values are used.

NOTE: In the wizard, only one property can be selected and configured at a time.

Template_1_CD: Configure elements

| Element Property | Type | Connection Type |
|------------------|-----------|-----------------|
| Equipment_1 | Equipment | |
| Disabled | string | Parameter |
| NamePath | string | Parameter |
| NameID | string | Parameter |
| DisplayName | string | Parameter |
| Type | string | Input Value |
| Location | string | Parameter |
| Page | string | Parameter |
| Content | string | Input Value |
| Help | string | Parameter |
| Comment | string | Input Value |
| Parameters | string | Parameter |
| Tagprefix | string | Parameter |
| Hidden | string | Parameter |
| Custom1 | string | Parameter |
| Custom2 | string | Parameter |
| Custom3 | string | Parameter |
| Custom4 | string | Parameter |
| Custom5 | string | Parameter |
| Custom6 | string | Parameter |
| Custom7 | string | Input Value |
| Custom8 | string | Parameter |
| Scheduled | string | Parameter |
| DefaultState | string | Parameter |
| ScheduledID | string | Parameter |
| DeviceSchedule | string | Parameter |

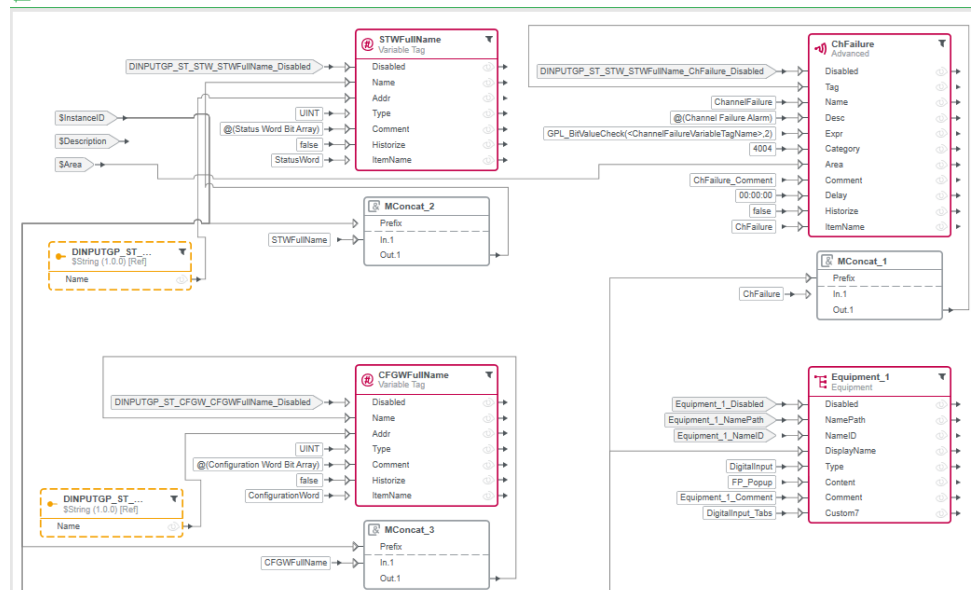
| Attribute | Value |
|---------------|-------------------------|
| Category | \$System |
| Identifier | \$InstanceID |
| Description | Equipment_1_DisplayName |
| Type | String |
| Default Value | |
| Value | DigitalInput |
| Description | Equipment_1_Type |
| Value | FP_Popup |
| Description | Equipment_1_Content |
| Value | DigitalInput_Tabs |
| Description | Equipment_1_Custom7 |

Resulting Templates

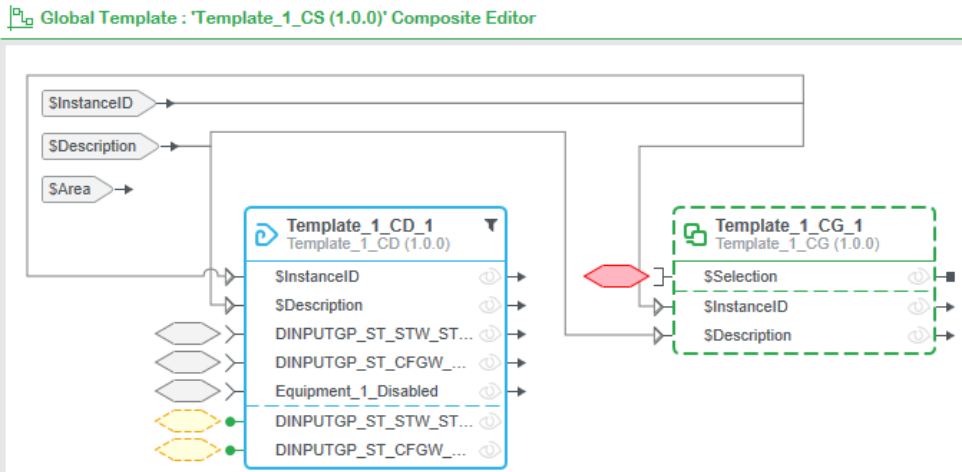
The following figure shows the Supervision elements that are the result of configurations performed in Supervision data facet windows 1/2 and 2/2. They are created in the Supervision data facet template (_CD) once the wizard is closed.

Properties that are configured with the default connection type *Parameter* and that have a blank identifier have been hidden manually for better clarity.

Global Template : 'Template_1_CD (1.0.0)' Facet Editor



The following figure shows the Supervision composite template (_CS) that is automatically created to regroup the Supervision functionality of the template.

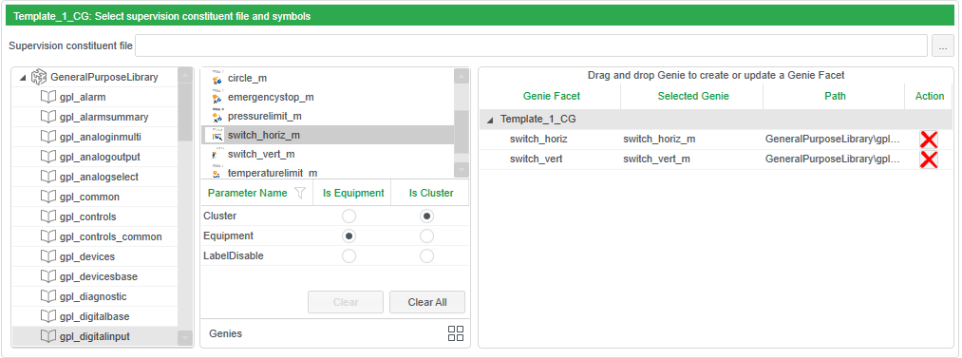


Template Creation Example – Supervision Genie Facet 1/2

Genie Selection

In this step, the Genies that need to be included in the template are selected. In this example, the *switch_horiz_m* and *switch_vert_m* from the existing *GeneralPurposeLibrary Include* project of the GPL have been selected. The corresponding Genie facets have been renamed *switch_horiz* and *switch_vert* respectively in the **Genie Facet** column.

Because the Genies are from the EcoStruxure Process Expert GPL, the *Is Equipment* and *Is Cluster* parameters have been configured for each one.



Template Creation Example – Supervision Genie Facet 2/2

Genie Property Configuration

In this step, you can configure Genie properties that you have not configured in the previous step.

In this example, *LabelDisable* does not need to be configured and is therefore set to *Not Connected*.

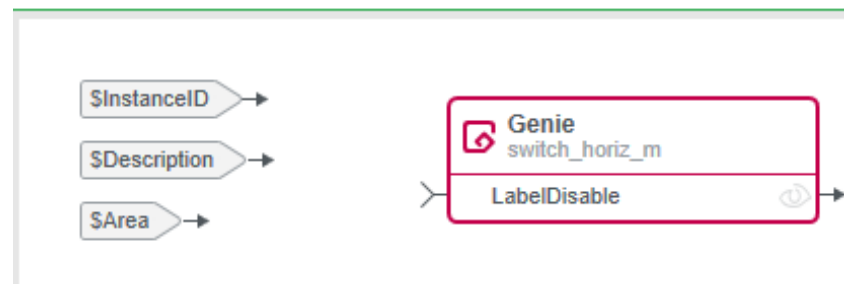
| Element Property | Type | Connection Type |
|------------------|--------|-----------------|
| Template_1_CG | | |
| switch_horiz | | |
| LabelDisable | string | Not Connected |
| switch_vert | | |
| LabelDisable | string | Not Connected |

| Attribute | Value |
|-----------|-------|
|-----------|-------|

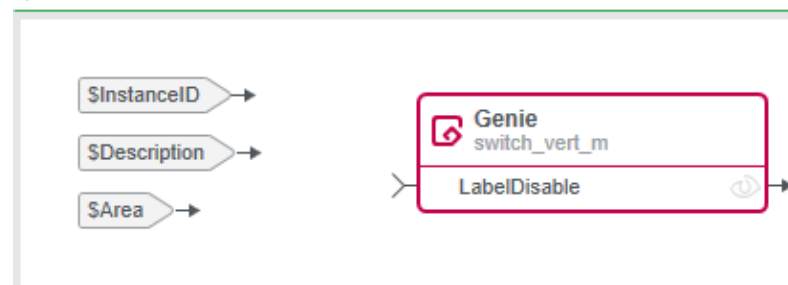
Resulting Templates

The following figure shows the two Genie elements that are the result of configurations performed in Supervision Genie facet windows 1/2 and 2/2. Each one is encapsulated in a Genie facet template (*switch_horiz* and *switch_vert*) once the wizard is closed. (Genie facet templates have *_CG* as default suffix unless you rename the facet template like it was done in this example in step 1/2.)

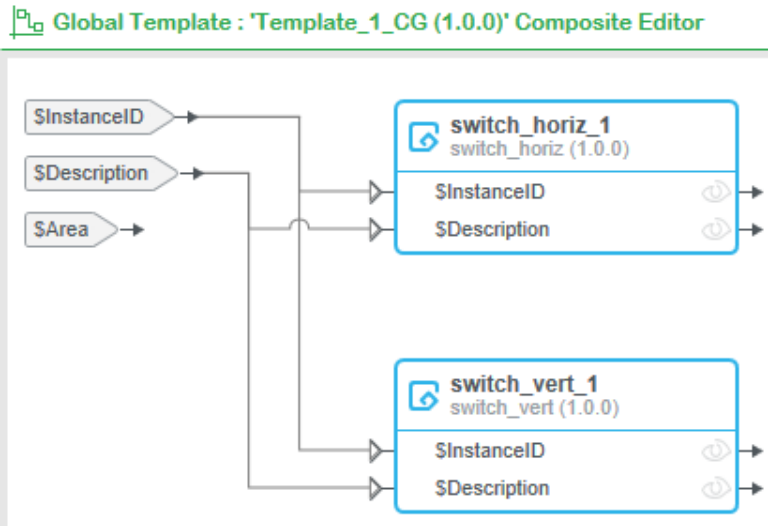
Global Template : 'switch_horiz (1.0.0)' Facet Editor



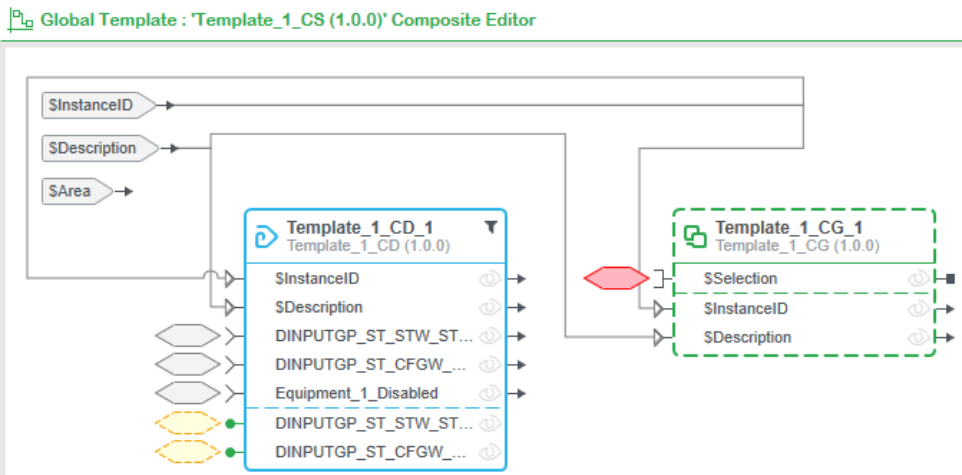
Global Template : 'switch_vert (1.0.0)' Facet Editor



The following figure shows the two Genie facet templates that are regrouped in one Supervision composite template (_CG).



The Supervision composite template (_CG) itself is referenced in another Supervision composite template (_CS) that is automatically created to regroup the Supervision functionality of the template. The _CG template is shown with hashed outline because it is optional, page 218.



Template Creation Example – Summary

Finalizing the Template Configuration

In this step, the default values for **Optional**, **Usability State**, **Include Document Template** and template **Description** have been used.

| Template_1: Template composition summary | | | | | |
|--|-------|----------|---|---|--|
| Identifier | V... | Optional | Location | Description | |
| Template_1 | 1.0.0 | | Global Templates\Folder_1 | Template_1 | |
| Template_1_UC | 1.0.0 | No | Global Templates\Folder_1\Control Services | Template_1 unity composite | |
| Template_1_UL | 1.0.0 | | Global Templates\Folder_1\Control Logic | Template_1 unity logic | |
| SDISignal_UL(DI_Signal) | 6.3.7 | No | Global Templates\Foundation Library\Application\Control Modules\Un... | Digital Input Signal | |
| Template_1_CS | 1.0.0 | Yes | Global Templates\Folder_1\Supervision Services | Template_1 supervision composite | |
| Template_1_CD | 1.0.0 | | Global Templates\Folder_1\Supervision Services\Supervision Data | Template_1 supervision data | |
| Template_1_CG | 1.0.0 | Yes | Global Templates\Folder_1\Supervision Services\Supervision Genie | Template_1 supervision genie composite | |
| switch_horiz | 1.0.0 | | Global Templates\Folder_1\Supervision Services\Supervision Genie | switch_horiz supervision genie facet | |
| switch_vert | 1.0.0 | | Global Templates\Folder_1\Supervision Services\Supervision Genie | switch_vert supervision genie facet | |
| \$HyperLink_CC | 1.0.3 | | Global Templates\General Purpose Library\Generic\Hyperlink Services | Hyperlink Composite of Documents and... | |

Usability State

Approved

☒ Include Document Template

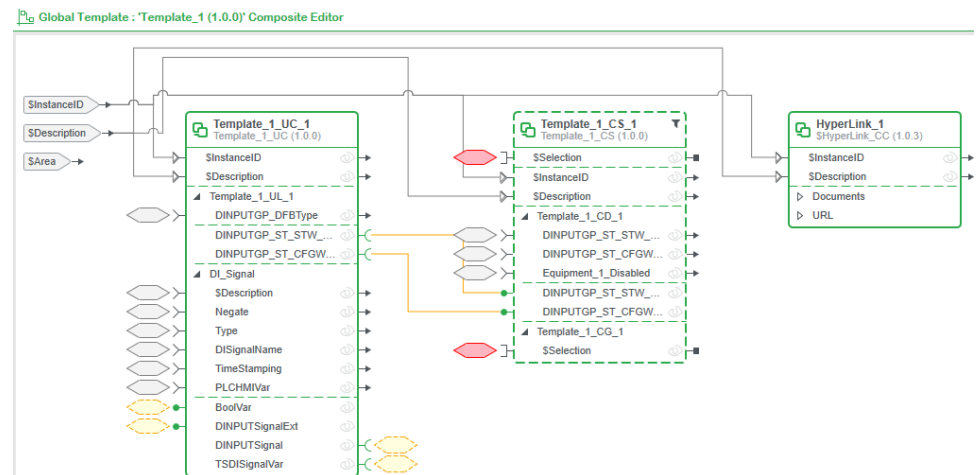
Resulting Templates

The following figure shows the composite template *Template_1* open in the Composite Editor. It references the *_UC* and *_CS* composite templates that regroup the Control and Supervision child templates respectively.

The *_CS* template is shown with a hashed outline because it is optional.

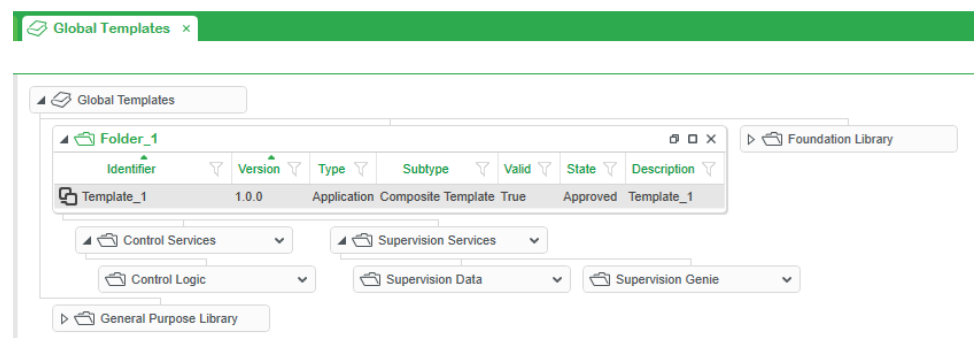
The *Hyperlink_CC* template has been created because the **Include Document Template** check box is selected.

NOTE: Most of the properties of elements that are configured with the default *Parameter* connection type and that have a blank identifier have been hidden manually for better clarity.



Folder Structure in the Global Templates Library

The following figure shows the folder structure that has been created for child templates in the **Global Templates Explorer** starting from Folder_1 (except for HAL templates, which are located in the folder structure of the Foundation library).



Global Templates Editors

What's in This Chapter

| | |
|--|-----|
| Opening Global Templates Editors | 220 |
| Interface Editor..... | 221 |
| Facet and Composite Template Editors | 227 |
| Common Template Editor Components | 233 |

Overview

This chapter describes the Global Templates editors, which allow you to view and edit the entire definition of Global Templates. They also allow you to create new templates based on existing ones or by starting from empty base templates.

You can view and edit:

- Composite templates by using the **Composite Editor**.
- Facet templates by using the **Facet Editor**.
- Interface models and interfaces by using the **Interface Editor**.

Starting from the control module template level, you can drill down through the entire composition of the template. At each level, you can view dependencies and which templates reference the one you are viewing or editing.

NOTE: Some of the commands that are described in this chapter may not be available when you open a template editor in read-only mode.

Opening Global Templates Editors

Opening Global Templates Editors

Overview

Modifying Schneider Electric Global Templates or templates created by users may affect the function of these templates and must be performed by qualified personnel. Before proceeding, refer to the [Overview of this part, page 174](#).

Opening Global Templates Editors

To open a Global Templates editor, you need to open a template. You cannot open a Global Templates editor without opening a template either in read-only or editing mode.

| Step | Action |
|------|--|
| 1 | In the tree view of the Global Templates Explorer , open the folder that contains the template that you want to view or edit. |
| 2 | Right-click the template and select: <ul style="list-style-type: none">• Read-only to open the template in the corresponding editor in read-only mode. This mode does not allow you to change the definition of the template. The background of the workspace has a light shade of gray to distinguish it from the editing mode.• Edit to open the template in the corresponding editor in editing mode. This mode allows you to modify the entire definition of the template. NOTE: Double-clicking the template opens the corresponding editor in read-only mode by default. |

NOTE: From the **Global Templates explorer**, you can also look up templates by entering a key word in the search field (see *EcoStruxure Process Expert, User Guide*). In the list of results, right-click the template and select **Read-only** or **Edit**. The software opens the selected template in the corresponding editor in either mode.

Working Copy of Templates

When you open a template in read-only or editing mode, the software creates a temporary working copy of the template with the same identifier and a specific version, page 81. It removes this working copy when you close the template editor.

Interface Editor

Interface Editor

Overview

The graphical **Interface Editor** allows you to manage the entire definition of interface models and interfaces.

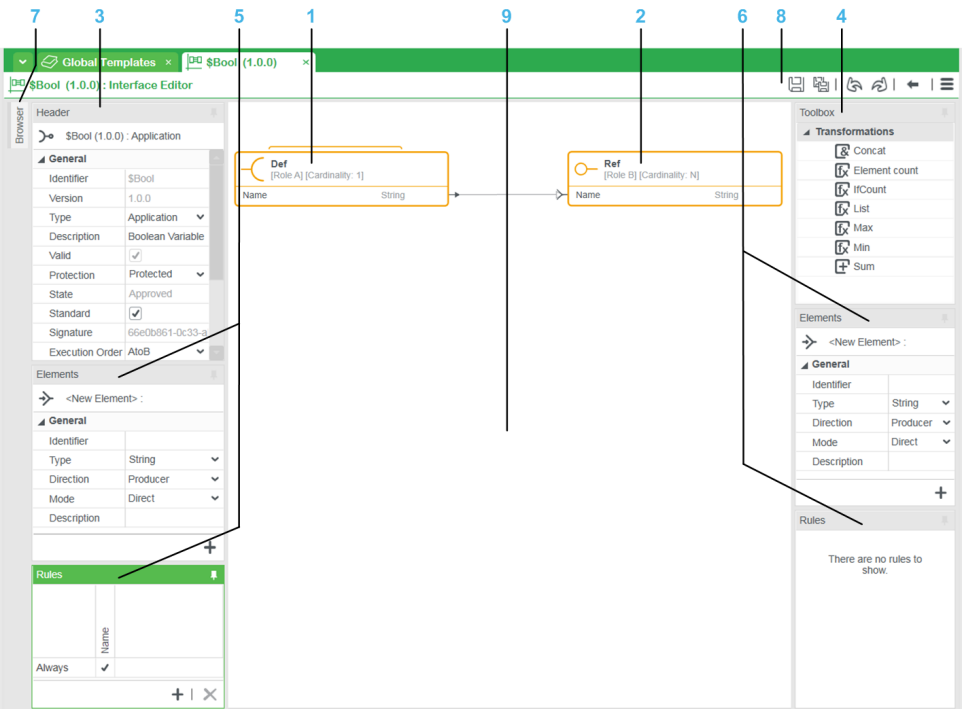
This topic describes the panes, toolbars, and menus of the **Interface Editor**.

Certain menus and commands may be available only in editing mode.

NOTE: This topic does not contain information on the definition of interface models or interfaces, page 62.

Interface Editor

The following figure shows an example of the **Interface Editor** in edit mode.



| Item | Description |
|------|--|
| 1 | Graphical representation of role A of the interface. |
| 2 | Graphical representation of role B of the interface. |
| 3 | Interface Header pane |
| 4 | Toolbox pane. |
| 5 | Elements and Rules panes for role A of the interface. |
| 6 | Elements and Rules panes for role B of the interface. |
| 7 | Button to show the interface Browser pane. |
| 8 | Editor toolbar containing commands that apply to the interface being viewed or edited. |
| 9 | Workspace. You can use the mouse to move the view. |

Editor Panes

The **Interface Editor** uses various panes to group properties and data that are related. The table describes the purpose of the panes.

| Pane | Description |
|-----------------|---|
| Header | Allows you to view and/or edit general properties of the interface. |
| Toolbox | Allows you to browse and select transformation functions, page 68 to be used in the template. |
| Elements | Allows you to view, create, and modify elements of a role of the interface. |
| Rules | Allows you to view, create, and modify rules for the elements of a role of the interface. |
| Browser | Allows you to search interface models and interfaces that are available in the Global Templates library. Move the pointer over the Browser tab to display the pane. Click the pin button to toggle between minimized and visible pane view. NOTE: The pane is available only in editing mode. |

| Pane | Description |
|---|--|
| Used By ⁽¹⁾ | Allows you to view the templates and interfaces that reference the interface, which is open in the editor. You can right-click a template or interface in the Used By pane to open it in a new editor window. You can also view its dependencies, page 238 or which other templates reference it. |
| Locations ⁽¹⁾ | Allows you to view the path to the interface and its copies, page 256. Right-click an entry and select Open Reference Container to open the corresponding folder in the Global Templates explorer. |
| Changes Log ⁽¹⁾ | Keeps track of changes that are made to the interface and saved. The pane indicates: <ul style="list-style-type: none"> • The action that was performed. • The state of the interface after the change. • The mandatory description of changes that is entered when the interface is saved. • The version of the interface after the change. • The user name of the user logged in to the engineering client. |
| (1) Use the corresponding command in the editor toolbar menu to open the pane. | |

NOTE: For information on working with panes, refer to the topic describing the engineering client workspace (see *EcoStruxure Process Expert, User Guide*).

Additional Information

The list indicates where you can find a detailed description of the contents of the **Interface Editor** panes:

- Header definition, page 63 and header common definition, page 80 (shared among interface models, facet, and composite templates).
- Elements definition, page 66.
- Rules definition editors, page 72.
- Interface toolbox, page 68.

Editor Toolbar

The figure shows the toolbar that is located in the top right corner of the **Interface Editor**.



| Item | Description |
|------|---|
| 1 | Edit button. Switches the editor from read-only to editing mode. |
| 2 | Save button. Opens the Save dialog box. For more information, refer to the topic describing how to save changes in templates, page 254. |
| 3 | Save as button. Opens the Save as dialog box. For more information, refer to the topic describing how to save changes in templates, page 254. |
| 4 | Undo and redo buttons. Lets you undo and redo most actions. |
| 5 | Shows the parent from which you have opened the interface and highlights it. If you had opened an interface that is an element of a template, only the template is highlighted. The parent template must be open for the command to be enabled. The command is not enabled when the interface is opened from the Global Templates Explorer . |
| 6 | Opens the editor toolbar menu that contains editor-specific commands. |

Editor Menu

The table describes the commands that are available in the editor menu of the **Interface Editor**.

| Command | | Description |
|---------|------------------|--|
| New... | | Opens the Global Templates dialog box, which contains the same base templates as the Toolbox of the Global Templates explorer. It allows you to create a new template in one or more existing folders of the Global Templates library and open it in edit mode in the corresponding editor. For more information, refer to the topic describing how to create templates, page 252. |
| View | Show Changes Log | The commands allow you to bring the corresponding pane to the front. |
| | Used By | If the pane is closed, the command opens it. |
| | Locations | For a description of the panes, refer to the topic describing editor panes, page 222. |
| Save | | Save button. Opens the Save dialog box. For more information, refer to the topic describing how to save changes in templates, page 254. |

| Command | | Description |
|----------------|----------------------------|---|
| Save As | | <p>Save as button.</p> <p>Opens the Save As dialog box.</p> <p>For more information, refer to the topic describing how to save changes in templates, page 254.</p> |
| Export | Standard Backup | <p>Opens the Export window, which allows you to export the interface definition to file (.sbk).</p> <p>You can only export an interface if you have saved changes.</p> <p>For more information on export feature, refer to the topic describing template export (see <i>EcoStruxure Process Expert, User Guide</i>).</p> |
| | Various image file formats | <p>Each command opens a Save dialog box, which allows you to create an image file with the corresponding file extension. The image captures the contents of the workspace as if it were shown with the Fit to content display ratio.</p> <p>Panes and toolbars are not captured.</p> |
| Close | | <p>Closes the template editor.</p> <p>If you have made changes, opens the Save dialog box.</p> <p>For more information, refer to the topic describing how to save changes in templates, page 254.</p> |

Workspace Actions

Right-click an empty area of the workspace to open a context menu with the following commands.

| Command | Description |
|-------------------------|------------------------------------|
| Show changes log | Opens the Changes log pane. |

Interface and Interface Element Actions

Right-click the header of an interface role in the workspace to open a context menu with the following command.

| Command | Description |
|---------------|--|
| Rename | <p>Allows you to modify the identifier of the role.</p> <p>For more information, refer to the topic describing interface roles, page 27.</p> |

Right-click an element of an interface role to open a context menu with the following commands. Commands may vary depending on the role of the element.

| Command | Description |
|---------------------------|---|
| Switch to Optional | <p>Sets the rule for the element, which is the producer to <i>optional</i>. In such case, the element is not required to provide data for the interface link to be valid.</p> <p>The default value is <i>required</i>.</p> <p>NOTE: If you set a role to optional and later you change the role to consumer, the role is reset to <i>required</i>.</p> |
| Switch to Required | <p>Sets the rule for the element, which is the producer to <i>required</i> (default value). In such case, the element is required to provide data for the interface link to be valid.</p> |
| Use as Consumer | <p>Switches the direction from <i>producer</i> to <i>consumer</i>.</p> <p>NOTE: Using this command resets the element rule to <i>required</i> if it was set to <i>optional</i>.</p> |
| Use as Producer | <p>Switches the direction from <i>consumer</i> to <i>producer</i>.</p> |

| Command | Description |
|--------------------|---|
| Switch Mode | <p>Switches the mode of the link between role A and role B of an element from <i>Direct</i> to <i>Transform</i> and the other way around.</p> <p>Right-clicking an element and switching the mode to:</p> <ul style="list-style-type: none"> • <i>Transform</i> <ul style="list-style-type: none"> ◦ Removes the direct link between role A and role B of the element. ◦ Removes the element from the other role. ◦ Requires that you create a new element in the other role with mode <i>Transform</i> and with the opposite direction. • <i>Direct</i> <ul style="list-style-type: none"> ◦ Removes the link between the element and the transformation function. ◦ Creates the element in the other role of the interface. ◦ Creates a direct link between both roles of the element. ◦ Requires that you reconfigure the transformation function, and possibly, the elements that are linked to it. <p>NOTE: When you switch the mode to <i>Transform</i>, ensure that the cardinality that is defined in the Header pane for either role A or role B makes it possible to use transformation functions.</p> <p>NOTE: The command is not available for nested interfaces.</p> |
| Delete | Deletes the element from both roles of the interface. |

Right-click a nested interface of an interface role to open a context menu with the following commands. Commands may vary depending on the role of the element.

| Command | Description |
|--------------------------------|---|
| Switch to Optional | <p>Sets the rule for the element to <i>optional</i>. In such case, the element is not required to provide data for the interface link to be valid.</p> <p>The default value is <i>required</i>.</p> <p>NOTE: The command is available by default for the <i>Def</i> role of the nested interface. To make the command available for the <i>Ref</i> role, add the element to the Rules table first by using the Add to Rules table command first.</p> |
| Switch to Required | Sets the rule for the element, which is the producer to <i>required</i> (default value). In such case, the element is required to provide data for the interface link to be valid. |
| Add to Rules table | Adds the element that has the <i>Ref</i> role to the Rules table so that you can define a rule for it (<i>optional</i> or <i>required</i>). |
| Remove from Rules table | <p>Removes the element that has the <i>Ref</i> role from the Rules table.</p> <p>NOTE: Removing the element from the Rules table makes the Switch to Optional and Switch to Required commands unavailable.</p> |
| Inspect | Opens a submenu that lets you open the Used By and Dependencies Tree panes, page 222. |
| Read-only | Opens the nested interface in another Interface Editor in read-only mode. |
| Edit | Opens the nested interface in another Interface Editor in editing mode. |
| Update | Allows you to update the version of the nested interface to the latest one, which exists in the Global Templates library. |
| Replace | Opens the Replace dialog box. It allows you to replace the nested interface by a different one, which exists in the Global Templates library. |
| Delete | Deletes the nested interface from both roles of the interface. |

Saving Changes

Refer to Saving Changes in Global Templates, page 254.

Facet and Composite Template Editors

Overview

This topic describes the user interface of the **Facet Editor** and **Composite Editor**.

Facet Editor

Overview

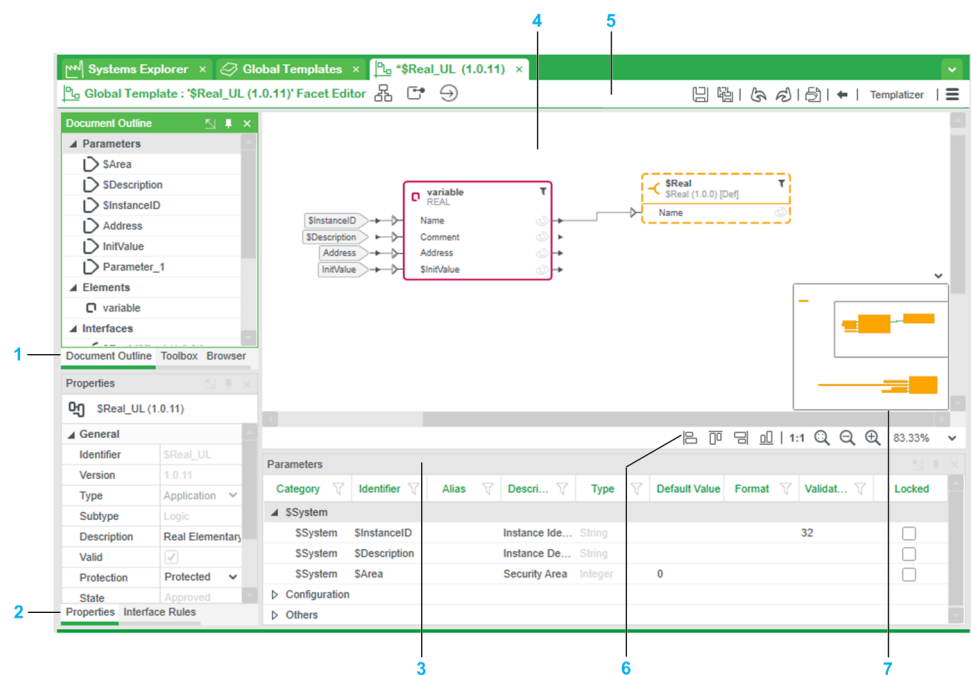
The graphical **Facet Editor** allows you to manage the entire definition of Control and Supervision facet templates.

For a description of panes, toolbars, and menus, refer to the topic describing common template editor components, page 233.

NOTE: This topic does not contain information on the definition of facet templates, page 73.

Facet Editor

The following figure shows an example of the **Facet Editor** in editing mode.



| Item | Description |
|------|--|
| 1 | Document Outline, Toolbox, and Browser panes. |
| 2 | Properties and Interface Rules panes. |
| 3 | Parameters pane. |
| 4 | Workspace area displaying a graphical representation of the elements of the template that you are editing. |
| 5 | Editor toolbar, page 233 containing commands that apply to the template being viewed or edited. |
| 6 | Workspace toolbar, page 234. |
| 7 | Radar view, page 241. |

Workspace Actions

For a description of the context menu that opens when you right-click an empty area of the workspace, refer to the description of the [workspace context menu](#), page 237.

Element Actions

Right-click the header of an element in the workspace of the **Facet Editor** to open a context menu with the following commands.

| Command | Description |
|--------------------------|---|
| Optimize Bindings | Reroutes bindings of the element automatically. |
| Show/Hide Unchecked | The commands have been replaced by a filter menu , page 240 at the element level. |
| Show/Hide Unbound | |
| Go To | Opens a submenu that contains the name and path of other objects and/or their properties that are linked to the element through bindings. It helps you locate them , page 250. When you select an entry, the object and the binding are highlighted in the workspace. |
| Copy | Copies the element for pasting in this template or another template. |
| Exclude | After you confirm the command, removes the element from the workspace and makes it available again in the Elements section of the Document Outline pane. Existing bindings with the element are discarded. NOTE: The command is available only for Control elements. |
| Delete | After you confirm the command, removes the element from the facet template. Existing bindings with the element are discarded. NOTE: The command is available only for certain types of elements. |
| Rename | Allows you to modify the identifier of the element. |
| Properties | Displays the properties of the element in the Properties pane. |

Right-click the property of an element in the workspace of the **Facet Editor** to open a context menu with the following commands.

| Command | Description |
|--------------------------------|--|
| Create Parameter | Allows you to create a parameter that you can customize , page 153. |
| Create Input Value | Allows you to enter an absolute value. |
| Show/Hide \$IsConnected | Displays or hides the <i>\$IsConnected</i> property, which outputs the boolean value <i>TRUE</i> when the interface link is successfully established. This implies that applicable interface and interface element rules are satisfied; otherwise the value is <i>FALSE</i> . The command is available for interfaces only. |
| Go To | Opens a submenu that contains the name and path of other objects and/or their properties that are linked to this property through bindings. It helps you locate them , page 250. When you select an entry, the object and the binding are highlighted in the workspace. |
| Binding To Here From... | Use this command, page 245 on the property that is the destination of the binding to be created. It enables the Bind to command. |
| Binding From Here To... | Use this command, page 245 on the object that is the source of the binding to be created. It enables the Bind from command. |
| Properties | Displays the properties of the element in the Properties pane. |

Right-click the header of an interface element in the workspace of the **Facet Editor** to open a context menu with the following commands.

| Command | Description |
|------------------------------------|--|
| Switch to Optional/Required | <p>Optional: Makes the element optional in the context of the template.</p> <p>The element is represented with a dotted outline.</p> <p>Required: Makes the element mandatory in the context of the template.</p> <p>The element is represented with a solid outline.</p> |
| Optimize Bindings | Reroutes bindings of the element automatically. |
| Show/Hide Unchecked | The commands have been replaced by a filter menu, page 240 at the element level. |
| Show/Hide Unbound | |
| Inspect | The menu entry has been removed. Use the left-hand toolbar buttons instead to open the Used By and Dependencies Tree panes, page 238. |
| View | Opens the element in the corresponding editor in read-only mode. |
| Edit | Opens the element in the corresponding editor in edit mode. |
| Edit/Extend Interface | <p>Opens the Edit/Extend Interface window, page 308, which lets you edit, add, and remove elementary elements (in direct mode only) in the interface without the need to edit the interface by using the Interface Editor.</p> <p>The command is not available for deferred interfaces and when the interface element contains a nested interface.</p> <p>Updating the templates that reference the other role of the interface is required.</p> |
| Go To | Opens a submenu that contains the name and path of other objects and/or their properties that are linked to the element through bindings. It helps you locate them, page 250. When you select an entry, the object and the binding are highlighted in the workspace. |
| Copy | Copies the element for pasting in this template or another template. |
| Update , page 294 | Allows you to update the version of the interface model that is used by the interface to the latest one, which exists in the Global Templates library. |
| Replace , page 300 | Opens the Replace dialog box. It allows you to replace the interface model that is used by the interface by a different one, which exists in the Global Templates library. |
| Delete | Removes the element from the template. |
| Rename | Allows you to modify the identifier of the element. |
| Properties | Displays the properties of the element in the Properties pane. |

Additional Information

The list indicates where you can find a detailed description of the contents of the various panes of the **Facet Editor**:

- Properties, page 74 and properties common definition, page 80 (shared among interface models, facet, and composite templates).
- Facet Elements, page 92
- Interfaces Rules, page 88
- Toolbox: binding functions, page 108 and/or Supervision Participant elements, page 238.
- Parameter pane, page 85

Saving Changes

Refer to Saving Changes in Global Templates, page 254.

Composite Editor

Overview

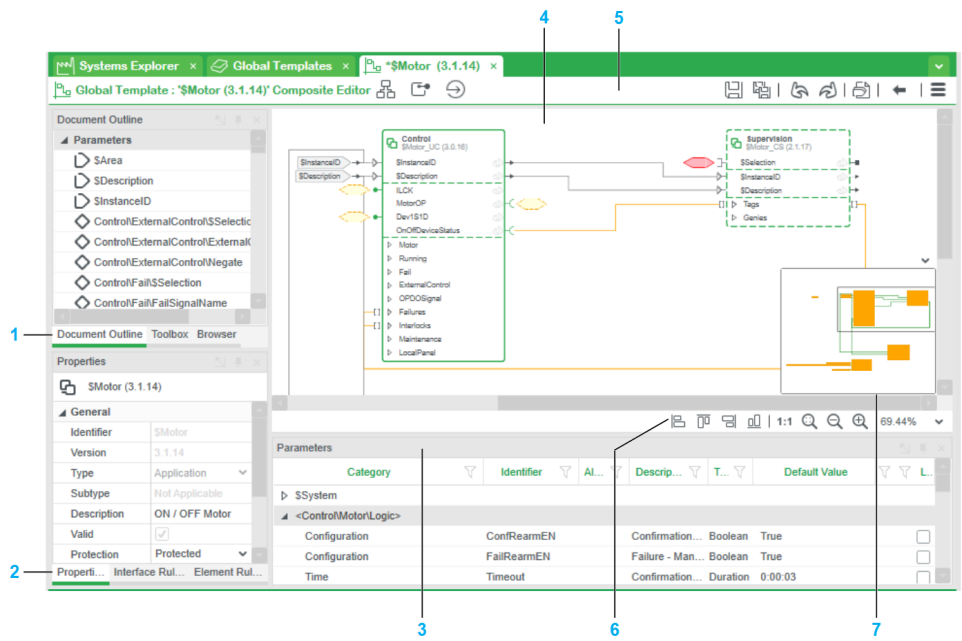
The graphical **Composite Editor** allows you to manage the entire composition and definition of composite templates.

For a description of panes, toolbars, and menus, refer to the topic describing common template editor components, page 233.

NOTE: This topic does not contain information on the definition of composite templates, page 75.

Composite Editor

The following figure shows an example of the **Composite Editor** in editing mode.



| Item | Description |
|------|--|
| 1 | Document Outline, Toolbox, and Browser panes. |
| 2 | Properties, Interface Rules, and Element Rules panes. |
| 3 | Parameters pane. |
| 4 | Workspace area displaying a graphical representation of the elements of the template that you are editing. |
| 5 | Editor toolbar, page 233 containing commands that apply to the template being viewed or edited. |
| 6 | Workspace toolbar, page 234. |
| 7 | Radar view, page 241. |

Workspace Actions

For a description of the context menu that opens when you right-click an empty area of the workspace, refer to the description of the workspace context menu, page 237.

Element Actions

Right-click the header of an element in the workspace of the **Composite Editor** to open a context menu with the following commands.

| Command | Description |
|------------------------------------|--|
| Switch to Optional/Required | Optional: makes the element optional in the context of the template. The element is represented with a dotted outline. Required: makes the element mandatory in the context of the template. The element is represented with a solid outline. |
| Defer | Defers items to make them available in the top-level template referencing the composite that you are editing (for example, in the control module). Selecting the command displays a submenu with the following commands: <ul style="list-style-type: none"> • Unbound selection/parameters: Makes available element selections (only optional elements) and parameters for which no binding exists. • Unbound interfaces: Makes available interfaces for which no binding exists. • All unbound: Makes available element selections (only optional elements), parameters, and interfaces for which no binding exists. |
| Optimize Bindings | Reroutes bindings of the element automatically. |
| Show/Hide Unchecked | The commands have been replaced by a filter menu, page 240 at the element level. |
| Show/Hide Unbound | |
| Inspect | The menu entry has been removed. Use the left-hand toolbar buttons instead to open the Used By , Dependencies Tree , and External References panes, page 238. |
| View | Opens the element in the corresponding editor in read-only mode. |
| Edit | Opens the element in the corresponding editor in edit mode. |
| Edit/Extend Interface | Opens the Edit/Extend Interface window, page 308, which lets you edit, add, and remove elementary elements (in direct mode only) in the interface without the need to edit the interface by using the Interface Editor . The command is not available for deferred interfaces and when the interface element contains a nested interface. Updating the templates that reference the other role of the interface is required. |
| Go To | Opens a submenu that contains the name and path of other objects and/or their properties that are linked to the element through bindings. It helps you locate them, page 250. When you select an entry, the object and the binding are highlighted in the workspace. |
| Copy | Copies the element for pasting in this template or another template. |
| Update , page 294 | Allows you to update the version of the template that is used by the element to the latest one, which exists in the Global Templates library. In case of an interface element, lets you replace the interface model. |
| Replace , page 300 | Opens the Replace dialog box. It allows you to replace the template that is used by the element by a different one, which exists in the Global Templates library. In case of an interface element, lets you replace the interface model. |
| Delete | Removes the element from the template. |
| Rename | Allows you to modify the identifier of the element. |
| Properties | Displays the properties of the element in the Properties pane. |

Right-click the parameter of an element in the workspace of the **Composite Editor** to open a context menu with the following commands.

| Command | Description |
|--|---|
| Create Deferred | Defers the parameter to make it available in the top-level template referencing the composite that you are editing. |
| The other commands are described in the table describing the context menu commands of properties of elements of facet templates, page 228. | |

Right-click the interface of an element in the workspace of the **Composite Editor** to open a context menu with the following commands.

| Command | Description |
|----------------------------------|---|
| Create Deferred | Defers the interface to make it available in the top-level template referencing the composite that you are editing. |
| Create Extended | Shows the interface as an element of the template. |
| Explode Implode | <ul style="list-style-type: none"> Collapsed state: Makes a connector available that represents the complete interface. The connection is made by using an interface link (shown in orange). Expanded state: Makes each element of a multi-element interface available individually for connection to other elements by using bindings (shown in gray). |
| Inspect | <p>The menu entry has been removed. Use the left-hand toolbar buttons instead to open the Used By and Dependencies Tree panes, page 238.</p> <p>They allow you to view respectively:</p> <ul style="list-style-type: none"> The templates that reference it. Nested interfaces that it references. |
| View | Opens the interface in the corresponding editor in read-only mode. |
| Edit | Opens the interface in the corresponding editor in edit mode. |
| Go To | Opens a submenu that contains the name and path of other objects and/or their properties that are linked to the interface through bindings. It helps you locate them, page 250. When you select an entry, the object and the binding are highlighted in the workspace. |
| Properties | Displays the properties of the interface in the Properties pane. |

NOTE: Additional commands are described in the topic describing actions for properties of elements, page 228 in the **Facet Editor**.

Additional Information

The list indicates where you can find a detailed description of the contents of the various panes of the **Composite Editor**:

- Properties (Header), page 76 and properties common definition, page 80 (shared among interface models, facet, and composite templates).
- Composite Elements, page 77
- Interfaces Rules, page 88
- Toolbox: binding functions, page 108
- Parameter pane, page 85

Saving Changes

Refer to Saving Changes in Global Templates, page 254.

Common Template Editor Components

Overview

The **Facet Editor** and **Composite Editor** have several components in common, which are described in this topic. Items that are specific to either editor are pointed out.

Common Template Editor Toolbars and Menus

Overview

The **Facet Editor** and **Composite Editor** feature toolbars and menus, which allow you to access the various tools and functions that you require to create and edit Global Templates.

This topic describes toolbars and menus that the **Facet Editor** and **Composite Editor** have in common. Those that are specific to either editor are pointed out.

Certain menus and commands may be available only in editing mode.

NOTE: Unless otherwise mentioned, toolbars contain commands that apply to the template that is being viewed or edited. To interact with an element of a template, open the context menu of the element by right-clicking it.

Editor Toolbars

The figure shows the toolbar that is located in the top left corner of the editors.



The buttons apply to the template that is being edited. However, if an element inside this template is selected, they apply to this element.

| Item | Description |
|------|---|
| 1 | Displays the Dependencies Tree pane, page 238. |
| 2 | Displays the Used By pane, page 238. |
| 3 | Displays the External References pane, page 238. |

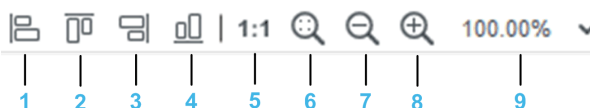
The figure shows the toolbar that is located in the top right corner of the editors.



| Item | Description |
|------|--|
| 1 | Edit button. Switches the editor from read-only to editing mode. |
| 2 | Save button. Opens the Save dialog box. For more information, refer to the topic describing how to save changes in templates, page 254. |
| 3 | Save as button. Opens the Save As dialog box. For more information, refer to the topic describing how to save changes in templates, page 254. |
| 4 | Undo and redo buttons. Lets you undo and redo most actions. |
| 5 | Opens the Parameters pane, page 85. |
| 6 | Shows the parent from which you have opened the template and highlights the template. If you had opened an element inside a template, only the template is highlighted. The parent template must be open for the command to be enabled. The command is disabled when the child template is opened from the Global Templates Explorer . |
| 7 | Opens the Select Variables or Select Genie window, which allows you to encapsulate constituents in Control and compatible Supervision facet templates respectively. It also allows you to modify encapsulated elements with the help of the corresponding Participant. If a Control facet contains no constituents, the Content Not Found dialog box opens, which allows you to start the encapsulation process. For more information, refer to the topic describing the configuration of facet templates, page 260. NOTE: The command is available only in the Facet Editor . |
| 8 | Opens the editor menu, page 235 that contains editor-specific commands. |

Workspace Toolbar

The figure shows the workspace toolbar that is located in the bottom right-hand corner of the **Facet Editor** and **Composite Editor**.



| Item | Description |
|------|---|
| 1 | Aligns the left edge of a selected item in the workspace with the left edge of the item that you have selected first. NOTE: The graphical alignment rules do not apply to parameters and input values, except for the three <code>\$\$System</code> parameters <code>\$\$InstanceID</code> , <code>\$\$Description</code> , and <code>\$\$Area</code> . |
| 2 | Aligns the top edge of a selected item in the workspace with the top edge of the item that you have selected first. |
| 3 | Aligns the right edge of a selected item in the workspace with the right edge of the item that you have selected first. |

| Item | Description |
|------|---|
| 4 | Aligns the bottom edge of a selected item in the workspace with the bottom edge of the item that you have selected first. |
| 5 | Reverts to the default zoom level (100%). |
| 6 | Adjusts the zoom level to fit the elements of the template inside the visible workspace. If the layout of the template is very large, some objects may not fit inside the visible workspace. You can use the <i>radar view</i> , page 241 to locate these objects. |
| 7 | Zooms out (see <i>EcoStruxure Process Expert, User Guide</i>). Alternatively, use the minus (-) keyboard key. When you zoom out to 50%, the display switches to the <i>simplified view</i> , page 242. To return to the normal view, zoom in. |
| 8 | Zooms in. Alternatively, use the plus (+) keyboard key. |
| 9 | Current zoom level. Use a predefined level by clicking the arrow and selecting a value from the list. You can set a custom level by double-clicking the value, entering a new one, and pressing Enter . Range: 50 to 200. |

NOTE: For information on commands that allow you to select multiple items in the workspace and move the workspace around, refer to the topic describing the engineering client workspace (see *EcoStruxure Process Expert, User Guide*).

Editor Menu

The table describes the submenus and commands of the menu located in the toolbar of template editors.

| Command | | Description |
|----------------|------------------------------------|--|
| New... | | <p>Opens the Global Templates dialog box, which contains the same base templates as the Toolbox of the Global Templates explorer. It allows you to create a new template in one or more existing folders of the Global Templates library and open it in editing mode in the corresponding editor.</p> <p>For more information, refer to the topic describing how to create templates, page 252.</p> |
| Edit | Copy | <p>Copies the selected element for pasting in this template or another template.</p> <p>You can copy multiple elements by selecting them first.</p> |
| | Paste | <p>Pastes copied elements.</p> <p>When a naming rule applies to elements, the software uses the <i>_n</i> suffix (where <i>n</i> is an incremental number starting at 1) to create unique names.</p> |
| | Select All | Selects elements in the workspace. |
| | Select None | Clears selected elements in the workspace. |
| | Invert Selection | Selects elements that are not selected and clears selected elements in the workspace. |
| | Find | <p>Displays a search field, which allows you to find strings in elements of the workspace by entering entire or partial key words.</p> <p>Strings that contain the word you entered appear in a list with an indication of their path.</p> <p>Right-click a result and select Navigate to highlight it in the workspace.</p> <p>For example, entering <i>in</i> returns various results, such as:</p> <ul style="list-style-type: none"> • The <i>\$InstanceID</i> system parameter. • The <i>\$InstanceID</i> parameter of an element. • The <i>In</i> input of a binding function. • <i>Interfaces</i> to indicate that an element references interfaces. |
| View | Show Changes Log | <p>The commands allow you to bring the corresponding pane to the front.</p> <p>If the pane is closed, the command opens it.</p> <p>For a description of the panes, refer to the topic describing template editor panes, page 238.</p> |
| | Document Outline | |
| | Dependencies Tree | |
| | Used By | |
| | External References | |
| | Locations | |
| | Parameters | |
| | Interface Rules | |
| | Element Rules⁽¹⁾ | |
| | Toolbox | |
| | Browser | |
| | Properties | |
| Save | | <p>Save button.</p> <p>Opens the Save dialog box.</p> <p>For more information, refer to the topic describing how to save changes in templates, page 254.</p> |
| Save As | | <p>Save as button.</p> <p>Opens the Save As dialog box.</p> <p>For more information, refer to the topic describing how to save changes in templates, page 254.</p> |

| Command | | Description |
|---|--------------------------------------|--|
| Export | Standard Backup | Opens the Export window, which allows you to export the template definition to file (.sbk). You can only export a template if you have saved changes. For more information on export feature, refer to the topic describing template export (see <i>EcoStruxure Process Expert, User Guide</i>). |
| | Various image file formats | Each command opens a Save dialog box, which allows you to create an image file with the corresponding file extension. The image captures the entire layout of the template independently of the zoom level. Panels and toolbars are not captured. |
| Print | Only Visible Content | Opens the Print dialog box, which lets you print on one page, the portion of the template layout that is visible in the current view. |
| | Editor Content On Single Page | Opens the Print dialog box, which lets you print the entire layout of the template on one page. |
| Close | | Closes the template editor. If you have made changes, opens the Save dialog box. For more information, refer to the topic describing how to save changes in templates, page 254. |
| (1) The command appears only in the menu of the Composite Editor | | |

Workspace Context Menu

Right-click an empty area of the workspace of a template editor to open a context menu with the following commands.

| Command | Description |
|---------------------------|---|
| Inspect | The menu entry has been removed. Use the left-hand toolbar buttons instead to open the Used By and Dependencies Tree panes, page 238. |
| Align Left Edges | Refer to the description of the corresponding commands in the workspace toolbar, page 234. The commands are available only if two or more elements are selected in the workspace. |
| Align Top Edges | |
| Align Right Edges | |
| Align Bottom Edges | |
| Copy | Copies the selected element for pasting in this template or another template. You can copy multiple elements by selecting them first. |
| Paste | Pastes copied elements. When a naming rule applies to elements, the software uses the <i>_n</i> suffix (where <i>n</i> is an incremental number starting at 1) to create unique names. |
| Show Changes Log | Opens the Changes Log pane, page 238, which lets you view the history of changes that were made to the template and saved. |
| Properties | Brings the Properties pane to the front to display properties of the template that is open in the editor. If the pane is closed, the command opens it. |

Binding Context Menu

Right-click a binding or its connector, page 244 in the workspace of a template editor to open a context menu with the following commands.

| Command | Description |
|-----------------------------|---|
| Convert To Line | Lets you toggle between binding styles , page 248. |
| Convert to Connector | |
| Go To | For bindings in line style, opens a submenu that contains the name and path of objects and/or their properties that are linked to this object through bindings. It helps you locate them, page 250. When you select an entry, the object and the binding are highlighted in the workspace. For bindings in connector style, highlights the object that is connected to the other end of the binding. |
| Update Binding | Opens the Update Binding dialog box, which lets you move the destination, page 251 of the binding to another property or element. The command is not available for source connectors of bindings. |
| Delete | Deletes the binding after you confirm the command. |

Common Template Editor Panes

Introduction

The **Facet Editor** and **Composite Editor** use various panes to group properties and data that are related.

This topic describes panes that are common to both. Panes that are specific to either editor are pointed out.

Certain panes may be available only in editing mode.

For information on working with panes, refer to the topic describing the engineering client workspace (see *EcoStruxure Process Expert, User Guide*).

NOTE: This topic does not describe how to define or modify the contents of the panes.

Editor Panes

The table describes the purpose of the panes that you can access in the **Facet Editor** and **Composite Editor**.

| Pane | Description |
|-------------------------------------|--|
| Document Outline | Allows you to view and/or edit: <ul style="list-style-type: none"> • Parameters • Elements • Interfaces Click a parameter, element, or interface to display its properties in the Properties pane. The corresponding item is selected in the workspace. NOTE: To view and/or edit the properties of the template, click Parameters . |
| Interface Rules | Allows you to view and define the rules for the interfaces, page 88 referenced by the template. |
| Element Rules ⁽¹⁾ | Allows you to view and define the rules for the elements, page 77 referenced by the template. |
| Properties | Allows you to view and/or edit the properties of the selected element of the template. |
| Toolbox | Allows you to browse and select binding functions, page 108 to be used in the template. For facet templates of the Supervision category, the Toolbox pane also contains the following elements, page 92: |

| Pane | Description |
|-----------------------------|--|
| | <ul style="list-style-type: none"> For facet templates of the Data subtype: <ul style="list-style-type: none"> Advanced Alarm Calculated Variable Tag Disk Variable Tag Digital Alarm Equipment Equipment Parameter Equipment Group of Messages Local Variable Tag Message Time Stamped Digital Alarm Trend Tag Variable Tag For facet templates of the Genie subtype: <ul style="list-style-type: none"> Equipment Equipment Parameter Equipment Group of Messages For facet templates of the Client Event subtype: Event. For facet templates of the Server Event subtype: Report. <p>These elements contain the same properties as the corresponding elements of the Supervision Participant and allow populating the respective databases.</p> |
| Browser | <p>In the Facet Editor, allows you to browse and select interfaces to be referenced by the template.</p> <p>In the Composite Editor, allows you to browse and select facet and composite templates, and interfaces to be referenced in the template.</p> |
| Dependencies Tree | <p>In the Facet Editor, allows you to view the interfaces that the template references.</p> <p>In the Composite Editor, allows you to view the interfaces, facet and composite templates that the template references.</p> <p>You can right-click an item to view and/or edit it in a new editor. You can also view the dependencies of the item or which other templates reference it.</p> |
| Used By | <p>Allows you to view the templates that reference the template, which is open in the editor. You can right-click a template in the Used By pane to open it in a new editor. You can also view its dependencies or which other templates reference it.</p> |
| External References | <p>Allows you to view:</p> <ul style="list-style-type: none"> In which systems an instance of the template exists. For each system, the identifier of the instances using the template. <p>You can right-click an application instance to:</p> <ul style="list-style-type: none"> Open the folder containing the instance in the Application Explorer. Open the Inspect Instance window (see <i>EcoStruxure Process Expert, User Guide</i>). |
| Parameters , page 85 | <p>The Parameters pane allows you to:</p> <ul style="list-style-type: none"> View and manage parameter data that is used during instantiation to customize elements of an instance. Customize the layout of the parameters that are visible when you edit an instance by using the Instance Editor: <ul style="list-style-type: none"> Change the position of parameters within their category. Change the position of categories. Assign parameters to another category. |
| Locations | <p>Allows you to view the path to the template and its copies, page 256.</p> <p>Right-click an entry and select Open reference container to open the corresponding folder in the Global Templates explorer.</p> |

| Pane | Description |
|---|--|
| Changes Log | Keeps track of changes that are made to the template and saved. The pane indicates: <ul style="list-style-type: none">• The action that was performed.• The state of the template after the change.• The mandatory description of changes that is entered when the template is saved.• The version of the template after the change.• The user name of the user logged in to the engineering client. |
| (1) The pane appears only in the Composite Editor . | |

Common Template Editor Filters

Overview

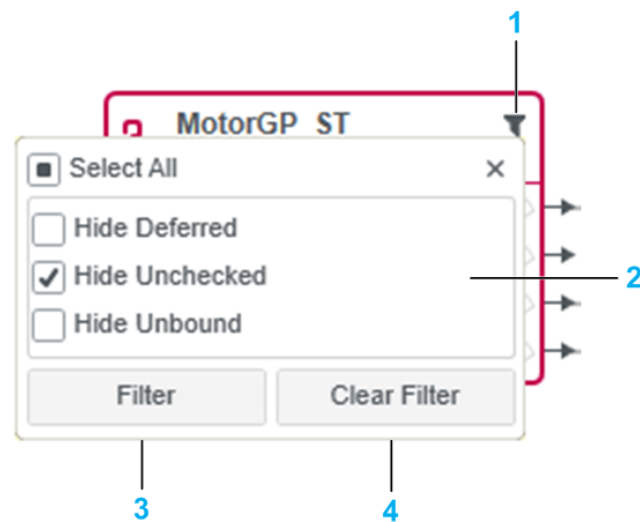
Starting with version 2021, the following context menu commands of elements, which appear in the **Facet Editor** and **Composite Editor** have been replaced by a filter menu.

- **Show/Hide Unchecked**
- **Show/Hide Unbound**

Also, the **Hide Deferred** filter criteria has been added.

Filter Menu Description

The following figure shows an example of the filter menu that is located in the header an element with its default setting.



| Item | Description | |
|------|---|--|
| 1 | Click the Filter button to open the element filter menu. When at least one filter is applied, the button is shown with a black fill. | |
| 2 | Hide Deferred | When selected, hides properties and interfaces that are deferred. |
| | Hide Unchecked | When selected, hides properties and interfaces with an eye icon that is located on the right-hand side and that is not visible (unchecked). To uncheck a property/interface, click its icon. The icon disappears. NOTE: A property/interface that has a binding connected to it cannot be hidden even if it is unchecked. |
| | Hide Unbound | When selected, hides properties and interfaces that have no binding connected. |
| 3 | Click the Filter button to apply the selected filter criteria to the representation of the element. | |
| 4 | Click the Clear Filter button to clear the filter criteria selection and update the representation of the element. | |

Using the Radar View

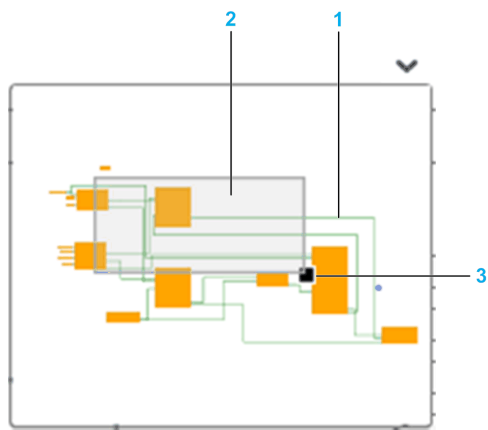
Overview

The radar view appears in the bottom right corner of certain graphical editors. It shows you which area of the layout of a template or workspace you are seeing in the editor.

You can also use it to navigate the inside the workspace and adjust the zoom level.

Description

The following figure shows an example of the **Radar View** in a template editor.



| Item | Description |
|------|--|
| 1 | Shows the objects that appear in the workspace of the editor. The view is refreshed when you modify the layout of objects. |
| 2 | Area that is visible in the editor. You can move the translucent rectangle by dragging it with the pointer. The view inside the editor moves accordingly. Alternatively, click anywhere inside the Radar View to move the center of the translucent rectangle to where you have clicked. When you change the zoom level, the size of the translucent rectangle adjusts accordingly. |
| 3 | Zoom level adjustment. Dragging the corner inward or outward lets you zoom in or out respectively. NOTE: While the pointer is over the Radar View , you can also use the mouse wheel to zoom. |

When the **Radar View** is collapsed, it appears as a button 

Using the Simplified View in Template Editors

Overview

Zooming out to the lowest level in the **Composite Editor** or **Facet Editor** switches the display to the simplified view.

This view mode lets you see, at a glance, the elements of a template and the relation between them without showing the details of bindings and element properties.

Some template engineering functionalities are not available.

To exit the simplified view, zoom in.

Restrictions of the Simplified View

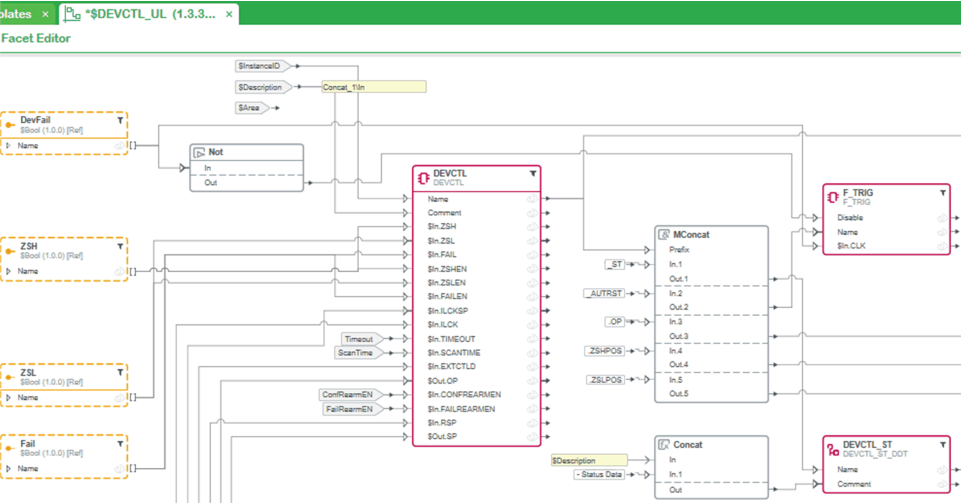
The table describes the restrictions in the template editor while the simplified view is active.

| Action or object | Restrictions |
|---|-------------------------|
| Actions related to bindings (such as creating, moving, deleting, switching styles). | Actions are not allowed |
| Creating parameters or platform inputs. | |

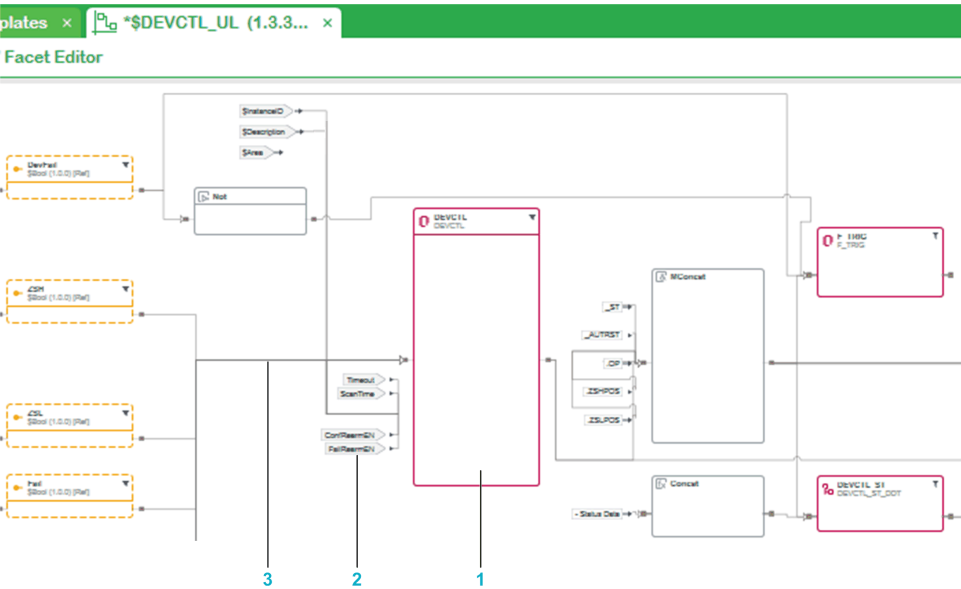
| Action or object | Restrictions |
|---|--|
| Element properties and deferred inputs and outputs. | Are not shown. |
| Multiple bindings to and from an element. | Are shown as a single line. |
| Binding connectors. | Are not shown. Bindings are shown in line style instead. |

Description

The following figure shows an example of template as it normally appears in the Facet Editor.



The following figure shows the same template when simplified view is active (the view is enlarged for better visibility).



| Item | Description |
|------|--|
| 1 | Elements keep their relative size and position to retain the aspect of the layout. |
| 2 | Parameters and platform inputs are shown in their actual position. |
| 3 | Bindings to and from each element are grouped into a single line. |

Editing, Creating, and Saving Global Templates

What's in This Chapter

| | |
|--|-----|
| Creating and Managing Bindings | 244 |
| Creating and Managing Global Templates | 252 |

Overview

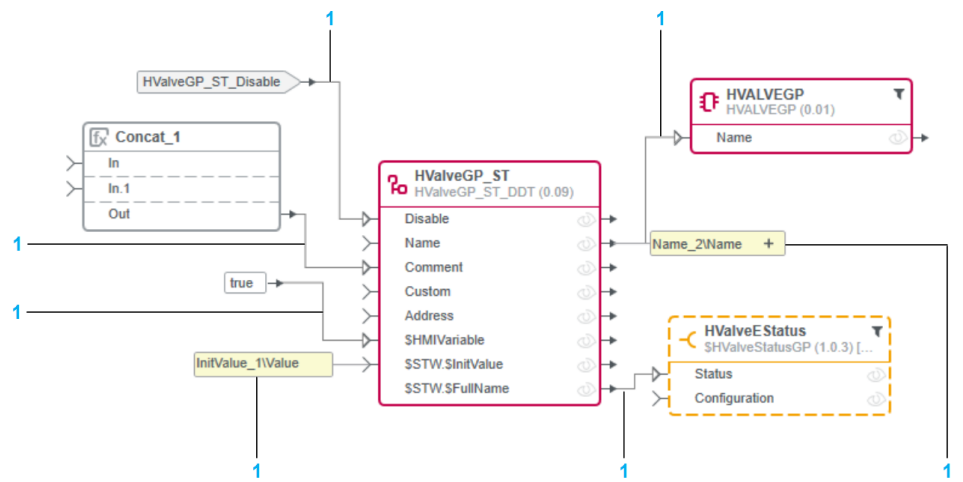
This chapter describes how to use the Global Templates explorer and its editors to manage templates of the Global Templates library.

Creating and Managing Bindings

Overview

Bindings are links connecting the output and input pins of various elements in order to propagate data, page 43. Bindings can only be created inside the template that you are editing.

The following figures shows several examples of bindings and their representation.



1 Binding

Pins of elements accept the following connections:

- Input pin: One connection
- Output pin: Several connections

You can create bindings in different styles, page 248 by using several methods, which are described in this topic.

Once a binding is created, you can highlight, locate its source and destination, convert, move its destination, and delete it.

Creating Bindings by Using the Mouse

Overview

Creating bindings by using the mouse is a graphical method, which is practical when you are creating few bindings and the pins of the elements to be connected are visible in the current view of the editor. The binding is represented by a line.

NOTE: Use the mouse also to move an existing binding, page 251.

Creating Bindings

To create a binding by using the mouse, proceed as follows.

| Step | Action |
|------|---|
| 1 | In a template editor, click the arrow-shaped output pin of the source element and drag the pointer away from it. Result: An arrow is drawn. NOTE: Press Esc to cancel the arrow drawing. |
| 2 | Release the left mouse button and move the pointer over the destination element. Result: Input pins that you can connect to are highlighted in green. |
| 3 | Move the pointer so that the arrow snaps to the input pin and click it. Result: A line is drawn representing the binding. |

NOTE: You can also create bindings by starting from an input pin.

NOTE: To move the workspace while the arrow is drawn and not yet connected, move the pointer either:

- Outside the workspace and use the scrollbars or the zoom buttons.
- Onto the **Radar View** and use the [view](#), page 241 to move the workspace.

When you move the pointer back inside the workspace, the arrow follows it automatically.

Managing Bindings

Refer to the topic describing how to [manage bindings](#), page 248.

Creating Bindings by Using Context Menu Commands

Overview

Creating bindings by using context menu commands of the **Facet Editor** and **Composite Editor** combines a graphical method with the ability to create several bindings faster.

It lets you create one or more bindings from a source to multiple destinations or to one destination from one source.

This method creates [connector-style bindings](#), page 248.

Bindings From a Source to One or More Destinations

The table describes the context menu commands that you can use to create one or more bindings by starting from the source (the output) and then selecting the destinations.

| Command | Description |
|---|---|
| Binding from here to... | Use this command on the object ⁽¹⁾ that is the source of the binding to be created. Tags the object so that you can create a binding to a compatible input pin by using the context menu command Bind from . |
| Bind from x | Appears in the context menu command of properties that have a compatible and available input pin once you have selected the Binding from here to... command on object ⁽¹⁾ x. The command remains available with the same source x until you select the Binding from here to... command again. |
| (1) A property of an element, a parameter, or a platform input. | |

Binding to a Destination From a Source

The table describes the context menu commands that you can use to create one binding by starting from the destination (the input) and then selecting the source.

| Command | Description |
|---|---|
| Binding to here from... | Use this command on the property that is the destination of the binding to be created. Tags the property so that you can create a binding from a compatible output pin by using the context menu command Bind to . |
| Bind to x | Appears in the context menu command of objects ⁽¹⁾ that have a compatible and available output pin once you have selected the Binding to here from... command on property x. The command is available until you create the link. |
| (1) A property of an element, a parameter, or a platform input. | |

Managing Bindings

Refer to the topic describing how to manage bindings, page 248.

Using the Create Bindings Dialog Box

Overview

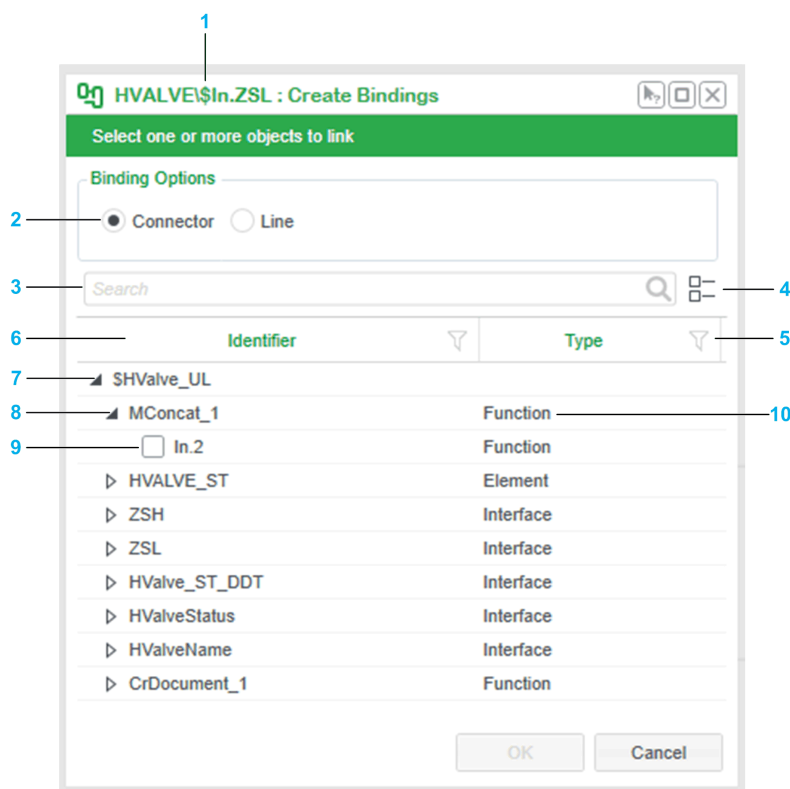
Creating bindings by using the **Create Bindings** dialog box is convenient when the elements that you want to connect are far apart or when you are creating several bindings from the same output.

To open the dialog box, right-click an input or output pin and select the command from the context menu.

This method lets you also select the style of binding, page 248.

Create Bindings Dialog Box

The following figure shows an example of the dialog box that opens when you right-click an output and select **Create Binding**.



| Item | Description |
|-------------|--|
| 1 | Indicates the identifier of the element and pin from which the bindings are being created. |
| 2 | Lets you select the style of the binding. |
| 3 | Search field. Lets you find elements and objects by typing the entire or part of their identifier. You do not need to use wildcards. The search is not case-sensitive. Results are displayed as you type. |
| 4 | Toggles between tree and grid view. |
| 5 | You can sort and filter (see <i>EcoStruxure Process Expert, User Guide</i>) the contents of columns. |
| 6 | <p>Lists the identifiers of compatible objects that you can create bindings with. These objects are either inputs/outputs of properties, parameters, or platform inputs and belong to elements of the template that you are editing.</p> <p>An object appears in the list even if it is already connected given it supports multiple connections.</p> <p>By default, items are listed in ascending alphabetical order.</p> <p>In tree view, the objects are grouped by element.</p> <p>NOTE: The column also shows properties that are hidden in the template editor given they are compatible.</p> |
| 7 | Identifier of the template that you are editing. |
| 8 | Identifier of an element of the template. |
| 9 | Identifier of the property that is available for connection. |
| 10 | Type of the object. |
| Path | <p>Indicates the full path of the object within its parent element.</p> <p>The column appears only in grid view.</p> |

Creating Bindings

To create one or more bindings by using the **Create Bindings** dialog box, proceed as follows.

| Step | Action |
|------|--|
| 1 | <p>Right-click either the source (output) or destination (input) pin of the link that you want to create and select Create Binding. You can also double-click the pin.</p> <p>If you want to create several bindings from the same source at once, right-click the corresponding output pin.</p> <p>Result: The Create Bindings dialog box opens.</p> |
| 2 | <p>Select the binding style, page 248.</p> <p>NOTE: You can toggle styles after the binding is created.</p> |
| 3 | <p>Select one or more objects to connect and click OK.</p> <p>Result: The Create Bindings dialog box closes and the bindings are created.</p> |

Managing Bindings

Refer to the topic describing how to manage bindings, page 248.

Managing Bindings

Overview

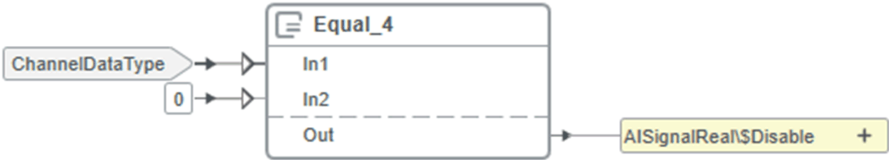
This topic describes the various actions that you can perform on existing bindings.

Binding Styles

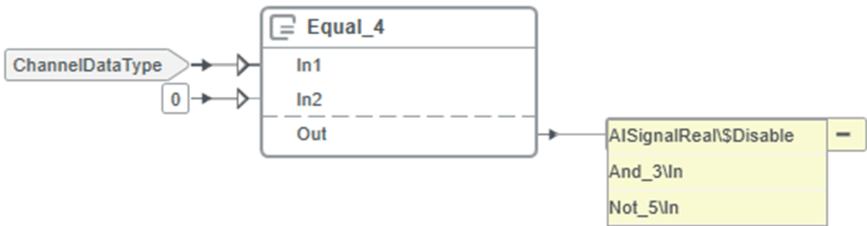
Bindings can appear as a line or as source and destination connector. Prefer the connector style when the connected elements are far apart or to reduce the use of lines, which can clutter the layout.

You can toggle styles by using the context menu, page 237 of a binding line or connector. You cannot toggle styles when bindings of mixed styles are selected.

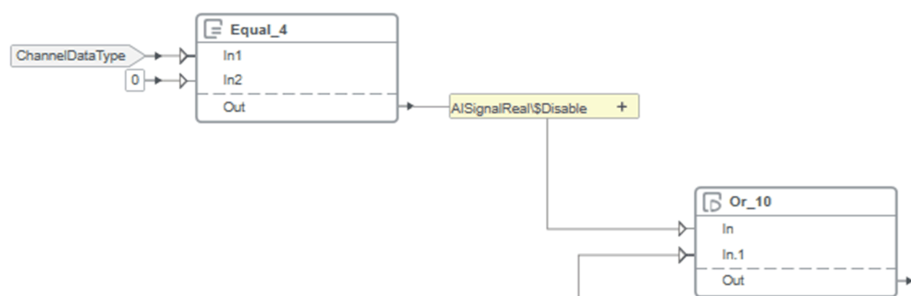
The following figure shows an output pin connected to three bindings in connector style. The connector group is shown collapsed, which lets you see the destination element and property of only one of them.



The following figure shows the same connector group expanded, which lets you see the destination of each binding.

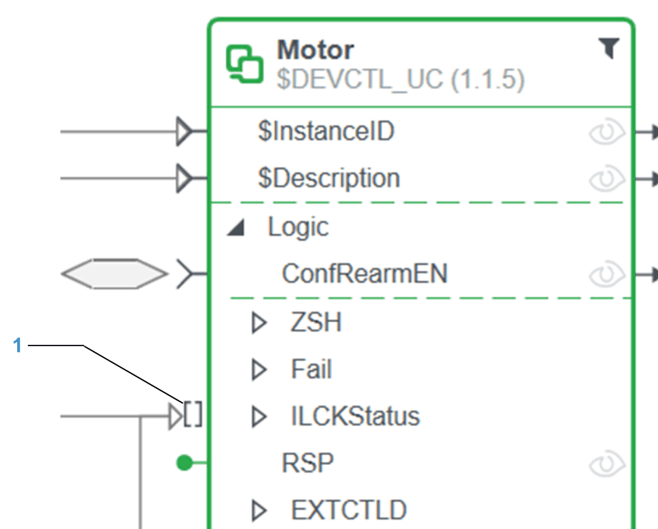


NOTE: A pin can be connected to bindings of both styles at the same time.



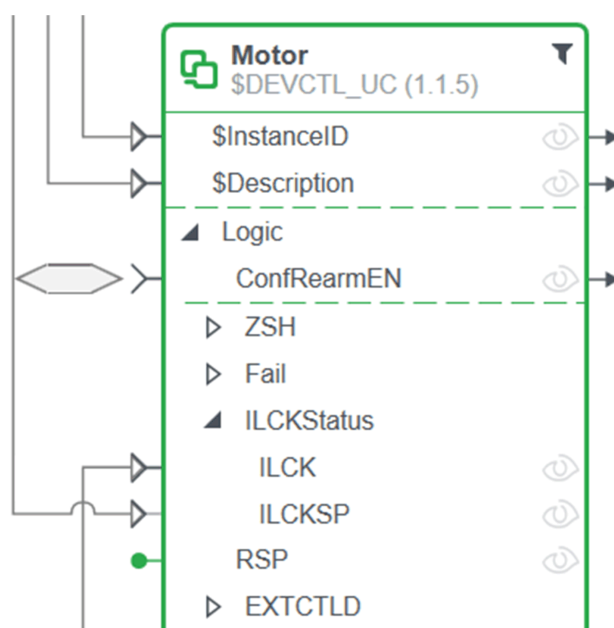
Bindings to Hidden Pins

Brackets next to an element that is collapsed indicate that one or more of its properties are connected by bindings. In such case, a single binding is shown at the element level.

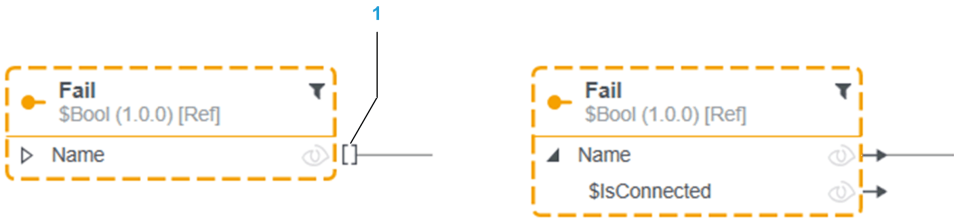


1 Brackets

Expand the element to view the connection to each property.



The same principle applies to elements of interfaces and their properties. The following example shows an interface with its *Name* element collapsed and expanded.



1 Brackets

Selecting and Highlighting Bindings

The table describes the results on bindings and/or connected objects depending on the action that you perform.

| Action | Result |
|---|--|
| Clicking a binding in line style. | Selects the binding, which appears in orange. |
| Clicking a collapsed connector group. | Selects the bindings of the group. The connector group appears with a green outline. <div></div> |
| Clicking one or more connectors of an expanded connector group. | Selects the corresponding bindings. <div></div> |
| Clicking an element header, parameter, or platform input. | Highlights the bindings connected to the object by using blue color. |

NOTE: When you press **Ctrl+A** while editing a template, bindings are highlighted but not selected.

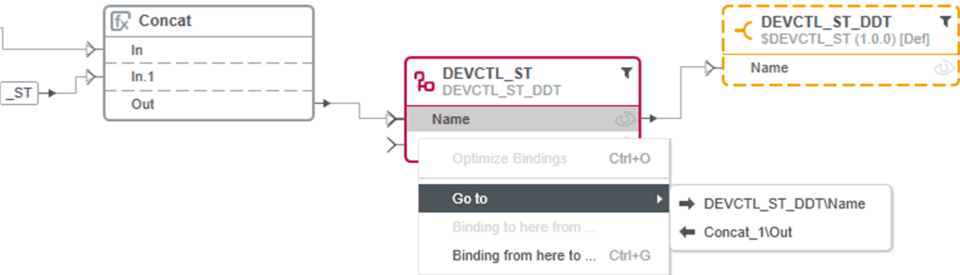
Locating Source and Destination of Bindings

You can locate the source or the destination of a binding by using the **Go to** context menu command, page 237 of the binding line, single connector, or connected object (for example, the property of a Control element).

For binding lines and connected objects, a submenu shows the name and path of connected properties, parameters, or platform inputs. An arrow indicates the direction along the binding or starting from the object.

The software selects the connected object.

The following example illustrates the use of the **Go to** command to locate the objects that are connected to the *Name* property of the *DEVCTL_ST* element.



NOTE: Select the command at the property/parameter/platform input level rather than at the binding level to see existing connections.

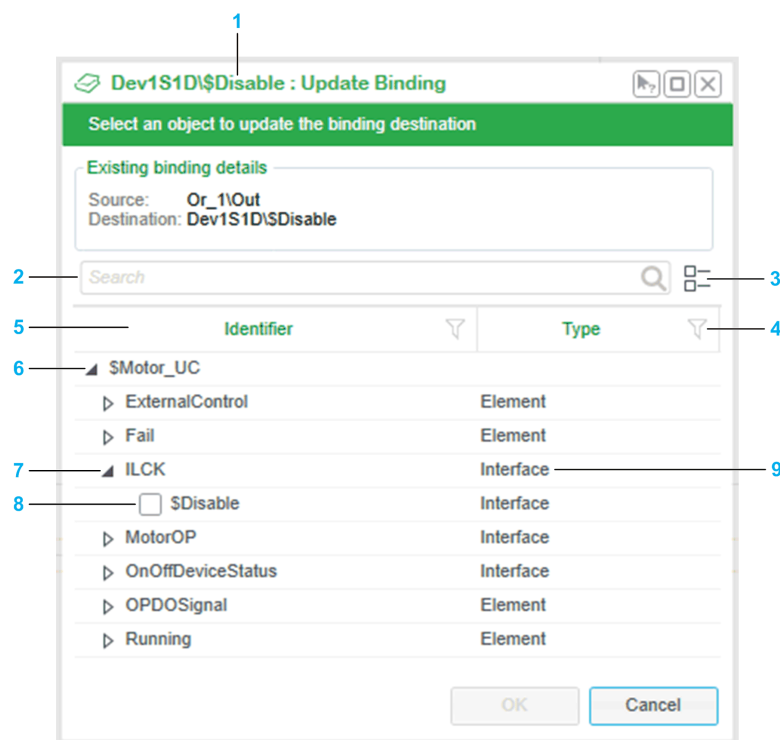
Moving Bindings

You can move the destination of a binding to another available and compatible input pin either way:

- Manually, by selecting the binding and dragging the black dot that appears on its destination pin to another pin. This method works only for bindings in line style.
- By using the **Update Binding** command, which appears in the context menu of line bindings and destination connectors. It lets you select a new destination and connects the binding to it.

NOTE: You cannot move the destination of a binding if its source is hidden, page 249 because the element is collapsed.

The following figure shows an example of the **Update Binding** dialog box.



| Item | Description |
|------|---|
| 1 | Indicates the identifier of the element and pin to which the binding is connected. |
| 2 | Search field. Lets you find elements and objects that you can create bindings with by typing the entire or part of their identifier. You do not need to use wildcards. The search is not case-sensitive. Results are displayed as you type. |
| 3 | Toggles between tree and grid view. |
| 4 | You can sort and filter the contents of columns. |
| 5 | Lists the identifiers of compatible objects that you can create bindings with. These objects are inputs of properties, which belong to elements of the template that you are editing. An object appears in the list even if it is already connected given it supports multiple connections. By default, items are listed in ascending alphabetical order. In tree view, the objects are grouped by element. NOTE: The column also shows properties that are hidden in the template editor given they are compatible. |
| 6 | Identifier of the template that you are editing. |
| 7 | Identifier of an element of the template. |

| Item | Description |
|-------------|---|
| 8 | Identifier of the property that is available for connection. |
| 9 | Type of the object. |
| Path | Indicates the full path of the object within its parent element. The column appears only in grid view. |

Deleting Bindings

Select one or more bindings of the same or different styles and press **Del** or use the context menu command.

When you delete a source or destination connector, the counterpart connector is also removed, deleting the entire binding.

NOTE: A binding is deleted also if either the source or destination object is deleted or removed.

Creating and Managing Global Templates

Creating Global Templates

Overview

In the **Global Templates Explorer**, you can create facet and composite templates, as well as interface models by using the appropriate item of the Global Templates **Toolbox** (see *EcoStruxure Process Expert, User Guide*).

You can also create a template while you view or edit one.

The Global Templates are empty when you create them and you need to configure their definition.

To create a functional template, use the [template creation wizard](#), page 175.

Creating Global Templates

To create a template or an interface model, proceed as follows.

| Step | Action |
|------|---|
| 1 | Open the Global Templates Explorer (see <i>EcoStruxure Process Expert, User Guide</i>). |
| 2 | If needed, create folders (see <i>EcoStruxure Process Expert, User Guide</i>). |
| 3 | in the Toolbox pane, drag the item that corresponds to the template or interface model that you want to create to a folder. Result: The software creates the corresponding template or interface model and displays it in the tree view. |

NOTE: Alternatively, you can right-click a folder and use the **Create** command.

Creating Global Templates from the Template Editor Menu

This procedure describes an alternative method to create a template or an interface model by using the [editor menu](#), page 233. This method allows you to create [linked copies](#), page 256 of templates and interface models in several

folders at once. It opens the newly created template or interface model in the corresponding editor in editing mode.

| Step | Action |
|------|---|
| 1 | Create folders in the Global Templates library where you want to store the template or interface model. NOTE: You can only create a template or interface model in an existing folder. |
| 2 | In a template or interface editor, click the editor menu icon. NOTE: The editor can be in either read-only or editing mode. |
| 3 | Click New.... Result: The Global Templates dialog box opens. it contains the same base templates as the Toolbox pane (see <i>EcoStruxure Process Expert, User Guide</i>) of the Global Templates explorer. |
| 4 | Select the item that corresponds to the template or interface model that you want to create. |
| 5 | In the Identifier text box, enter a name for the template or interface model that is unique in the Global Templates library. |
| 6 | To create the template or interface model in a folder other than the default one or to create linked copies in additional folders, click the folder icon next to the Locations text box. Otherwise, proceed to step 9. Result: The Select Template Location dialog box opens. NOTE: The default folder is indicated in the Locations text box before clicking the folder icon. |
| 7 | In the left-hand section of the Select Template Location dialog box, select one or more existing folders where you want create the template or interface model. Result: The path of each selected folder is shown in the right-hand section of the dialog box. NOTE: You can unselect a folder by clicking X next to its path, including the default folder. |
| 8 | Click OK . Result: The view reverts to the Global Templates dialog box and shows the selected folders in the Locations text box. The paths are separated by a semi-colon. |
| 9 | Click OK . Result: The software: <ul style="list-style-type: none">Leaves the template editor from which you have selected the New... command open.Creates the template or interface model in the selected folders as linked copies (if applicable).Opens the template or interface model in a new editor in editing mode. |

Global Templates Actions

For a description of actions that you can perform on the templates and interface models that you have created, refer to the topic describing Global Templates (see *EcoStruxure Process Expert, User Guide*).

Global Templates Default Parameters

The table indicates the default value of the common parameters of templates and interface models that you create.

| Parameter | Description |
|-------------------|---|
| Identifier | The identifier ends with the suffix _x (where x is an incremental number) unless you have entered the identifier manually. |
| Version | 1.0.0 |

| Parameter | Description |
|-----------|--------------|
| Valid | True |
| State | Not Approved |

NOTE: The value of the other parameters depends on the template or interface model that you have created. For a more information, refer to the topic describing the [Global Templates definition](#), page 62.

Saving Changes In Global Templates

Overview

When Global Templates are open in their respective editor, [page 220](#), you can save changes by using the following commands.

| | |
|----------------|---|
| Save | Allows you to save changes to templates that you have created by using various save options. Use the command to rename a template or change its usability state. The command is available only if the template is not referenced by another template and not used by an instance of the application; otherwise the software opens the Save as dialog box. |
| Save as | Allows you to save changes to Schneider Electric Global Templates, create a new template, or a new version of a template. |

When you select the **Save as** command, you can select one or more locations where to store the template.

If you have edited the template from within its parent, you have the possibility to update the parent template after closing.

NOTE: To save a template with a different identifier and/or version without editing it, use the **Duplicate** command, [page 305](#).

Restrictions When Saving Schneider Electric Templates

[Restrictions](#), [page 80](#) related to the template identifier apply when you save a Schneider Electric template.

Tracking Saved Changes

When you save a template, the software keeps a record of the save operation. You can view details in the **Changes log** pane, which is available in each template editor, [page 238](#).

Save Options

The table describes the items of the **Save** and **Save as** dialog boxes that the software displays when you select the respective command.

| Item | Description |
|-----------------------------------|---|
| Keep Version⁽¹⁾ | Select this versioning scheme to save changes in a template without changing its version number. Selected by default. |
| New Build | For information on the usage of the different version components, refer to Version , page 81 . By default, New Build is selected when you open the Save As dialog box. |
| New Minor | |
| New Major | |

| Item | Description |
|--|--|
| Other | <p>Selecting this versioning scheme allows you to edit the following parameters in the dialog box before saving the template:</p> <ul style="list-style-type: none"> The identifier. The three version components (major, minor, build number). <p>Select it to rename the template or create a new template based on the one you are editing.</p> <p>NOTE: By default, the version that is selected is the same as for New Build.</p> |
| Version | For a description of the version format, refer to Version , page 81. |
| Identifier | Within the Global Templates library, the software requires that the combination of identifier , page 80 and version be unique for each template. |
| Locations⁽³⁾ | <p>Allows you to select one or more existing folders of the Global Templates library to store the template.</p> <p>Default value: The current location of the template. If linked copies, page 256 exist, their location is indicated also.</p> <p>Click the browse button to open the Select Template Location dialog box, which lets you change locations and create linked copies by selecting additional locations.</p> |
| Usability State⁽¹⁾ | <p>For information on the parameter, refer to Usability State, page 82.</p> <p>Default value: Current state of the template.</p> |
| New Version Usability State⁽²⁾ | <p>Allows you to define the usability state of the new template that you are creating.</p> <p>Default value: Not Approved</p> <p>For information on the parameter, refer to Usability State, page 82.</p> |
| Old Version Usability State⁽²⁾ | <p>Allows you to define what will be the usability state of the template that you are editing after you create the new template.</p> <p>Default value: Approved</p> <p>For information on the parameter, refer to Usability State, page 82.</p> |
| Changes Description | <p>You must enter a description by using free form text to be able to save changes.</p> <p>The description that you enter is visible in the Changes log of the template editor. Entering a detailed description of changes allows you to keep track of the version history of the template.</p> |
| <p>(1) The item is displayed only in the Save dialog box.</p> <p>(2) The item is displayed only in the Save As dialog box for templates that are already referenced or instantiated.</p> <p>(3) The item is displayed only in the Save As dialog box.</p> | |

Saving Changes In Global Templates

Modifying Schneider Electric Global Templates or creating templates may affect the function of these templates and/or systems, and must be performed by qualified personnel. Before proceeding, refer to the [Overview of this part](#), page 174.

To save changes in a Global Template, proceed as follows.

| Step | Action |
|------|---|
| 1 | <p>In the template editor, click the Save or Save As button.</p> <p>Result: The software opens the appropriate dialog box.</p> |
| 2 | Select the versioning scheme that you want to use. |
| 3 | Edit the Identifier , Change Description , and/or Usability State as needed. |
| 4 | <p>Click the Save button.</p> <p>Result:</p> |

| Step | Action |
|------|--|
| | <ul style="list-style-type: none"> For the Save dialog box: <ul style="list-style-type: none"> The dialog box closes. Changes are saved in the template. The template remains open inside the editor. For the Save as dialog box: <ul style="list-style-type: none"> A new template, which includes your changes is created. It is opened in the editor. The source template is closed without saving. <p>NOTE: Click the Cancel button to close the Save or Save as dialog box without saving changes.</p> |
| 5 | <p>Close the template.</p> <p>Result: If you have edited the template from its parent and the parent is open in edit mode, the Update/Replace References dialog box opens. Otherwise, the template is closed.</p> |
| 6 | <p>Click either button:</p> <ul style="list-style-type: none"> Yes: Closes the template and updates/replaces, page 292 the references of it in the parent template. The parent template is shown. For each reference for which at least one binding cannot be recreated, a dialog box opens. It contains detailed information about the conflicts and lets you skip the update/replacement. <p>NOTE: Other templates that also reference the template that you have edited are not updated.</p> <ul style="list-style-type: none"> No: Closes the template. |

Copying and Pasting Global Templates and Folders

Creating Linked Copies of Templates

The **Copy** and **Paste** commands let you place copies of a template in other folders of the Global Templates library. The copies are linked to the source template. Changes that you make to the source template are also made to each copy that you have created.

NOTE: To create an independent copy of a template, in the same or a different folder, use the **Save as** or **Duplicate** command from the corresponding template editor or **Global Templates Explorer**.

Locating Copies of Templates

To locate the linked copies of a template, use the **Locations** context menu command (see *EcoStruxure Process Expert, User Guide*).

Deleting Copies of Templates

To delete a linked copy of a template, use the **Remove** context menu command (see *EcoStruxure Process Expert, User Guide*).

Copying Folders

You cannot paste an empty library folder.

When you copy a folder containing templates and you paste it on another folder, the software creates a subfolder with the same identifier, which contains a linked copy of each template of the source folders. Subfolders and their contents are created in the same way.

Configuring Interface Models

What's in This Chapter

Configuring Interface Models..... 257

Configuring Interface Models

Overview

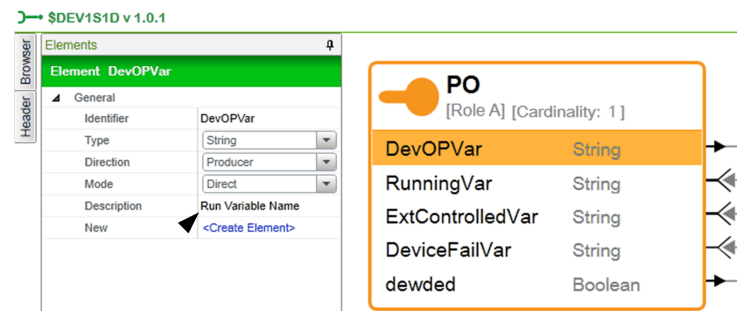
This topic describes the various configuration aspects of an interface model and its elements, as well as guidelines to adopt.

Guidelines

- Duplicate, page 305 a Schneider Electric interface model that you want to modify. Save the new template in your own folder structure, page 58.
- Save your work regularly, page 254.
- Enter a detailed description of the changes that you make when saving changes. This allows you to keep track of the version history through the **Changes log pane**, page 222.

The following example illustrates how the description of the element (*DevOPVar*) of an interface (*\$Dev1S1D*) appears in a tooltip in two different situations. This helps understand the purpose of the interface element.

When the interface is used as an element.



When the interface is shown in the expanded state at the parent level.



Configuring the Header

Configure properties of the interface model in the **Header** pane.

Configuring Elements

To create and configure interface elements, proceed as follows:

| Step | Description | Comment |
|------|---|---|
| 1 | In the Elements pane, enter the element identifier, page 29. | If you have selected an existing element and want to create a new element, click the * icon to create a new element first. The + icon appears. |
| 2 | Configure: <ul style="list-style-type: none"> • Type • Direction • Mode • Description (optional) Click the + icon to create the element in both roles, and a binding. | If you select Transform as mode, the element is created only for one role without binding. Select the transformation function from the Toolbox and create the counterpart element in the other role. Make the necessary bindings. |
| 3 | Configure interface element rules, page 165 as needed. | By default the rule Always is created for the role from which you create the element. |

Modifying Configured Elements

To modify properties of an already configured interface element, proceed as follows:

| Step | Description | Comment |
|------|---|---|
| 1 | In the graphical representation of one of the roles of the interface, select the element that you want to modify. | Changes to properties are applied to the selected element. You can move elements up and down within roles by dragging them to a new position. |
| 2 | Edit as required: <ul style="list-style-type: none"> • Type • Direction • Mode • Description (optional) | If you modify the mode of an element from Direct to Transform , the counterpart element in the other role and the binding are deleted. You cannot modify a mode from Transform to Direct when a binding exists between the element and a transformation function. NOTE: When you change the Type or Direction of an element with a direct binding, the corresponding property of the counterpart element is changed as well. |

Creating Nested Interfaces

To create a nested interface, proceed as follows:

| Step | Action |
|------|---|
| 1 | Edit the interface in which you want to include the nested interface. |
| 2 | From the Browser pane, select the interface that you want to include. |
| 3 | Drag the interface onto the workspace and select which role you want to include in role A of the interface. Result: The nested interface appears as an element in both roles with an interface link in between. NOTE: Nested interfaces are positioned last in the list of elements. You can reorder nested interfaces among themselves by moving them to a new position in the roles of the interface. |
| 4 | Edit the identifier of the nested interface if required. |
| 5 | Edit and/or create rules for the nested interface. |

NOTE: You cannot use transformation functions with nested interfaces.

Saving Changes

As long as the interface model is not referenced, you can save, [page 254](#) it with the same identifier and version.

Editing and Extending Interfaces In-Place

When you edit a template that references an interface, you can [edit and/or extend it in-place, page 308](#) by using the **Edit/Extend Interface** context menu command.

Remember to also update the other role of this interface in the templates that reference it.

Configuring Facet Templates

What's in This Chapter

| | |
|--|-----|
| Encapsulating Control Constituents | 260 |
| Modifying Encapsulated Control Constituents | 270 |
| Genie Creation Workflows | 272 |
| Using, Creating, and Modifying Supervision Animated Graphics | 273 |
| Encapsulating and Configuring Genies | 278 |
| Modifying and Replacing Encapsulated Genies | 281 |
| Configuring Facet Templates | 283 |

Overview

This chapter describes how to use the **Facet Editor** and software Participants to encapsulate constituents in Control logic and Supervision genie facet templates. It also describes how to configure facet templates, which can be referenced by a control module template.

Encapsulating Control Constituents

Overview

By using the **Facet Editor**, the **Select Variables** window, and a project file (.stu), you can encapsulate constituents of the Control Participant inside a Control facet template, page 252.

During the encapsulation process, you can modify the constituents, page 270 contained in the project file by using the Control Participant.

The encapsulated local constituents become elements of the facet template and are part of its definition.

New types and a copy of the project file are added to the content repository, page 263. An Application password is set on the project file and file encryption is enabled if the **Control constituent application password protection** setting is enabled (see *EcoStruxure Process Expert, User Guide*) in the **Global Templates Explorer**.

NOTE: If you do not have a project file (.stu), you can create it, page 139.

NOTE: You cannot use project files for which an application password is set.

Project File Version

You can only use a project file (.stu) that was created with a version of Control Expert that is the same as the Control Participant; otherwise, you need to make it compatible first.

For more information about the Control Participant version, refer to the platform release notes.

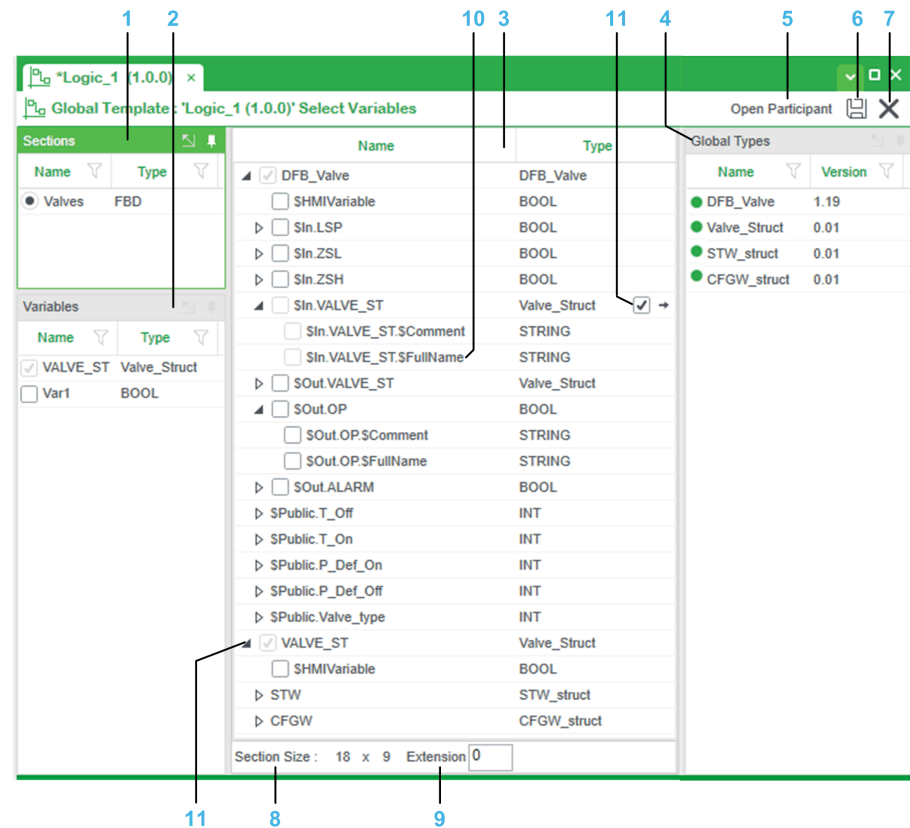
NOTE: To overcome version restrictions, you can also use archived application files (.sta).

Opening the Facet Editor

Refer to the topic describing how to open Global Templates editors, page 220.

Select Variables Window

The following figure shows an example of the **Select Variables** window, which shows the constituents that are contained in a Control project file and which you can select to encapsulate them in a facet template *Logic_1*. In this example, the *Valves* FBD section has been selected. It contains DFB *DFB_Valve*, input/output parameter *VALVE_ST*, and elementary variable *Var1*.




| Item | Description |
|------|---|
| 1 | Sections pane. Displays the FBD sections that are contained in the project file. You can select only one section at a time. |
| 2 | Variables pane. Displays the variables that the project file contains and their type. Selecting a variable adds it to the center pane where you can refine your selection. |
| 3 | Center pane. Displays the name and type of: <ul style="list-style-type: none"> FFB instances and variables of the section that is selected in the Sections pane. Variables that are selected in the Variables pane and that are not part of an FBD section. Each FFB that is contained in a selected section and each variable that you select in the Variables pane becomes an element that is encapsulated in the template. Expand each entry to view subitems such as, properties, pin names (formal parameters) of DFBs, comments, and initialization values of variables. Selected subitems appear in the element after encapsulation, which lets you connect them by using bindings. NOTE: For FFBs, the selection that you make in this pane does not impact the Control constituents that are generated in the logical Control Participant project. For variables, only those that you select are generated. NOTE: You cannot clear the check box of FFBs that are contained in a selected section. |

| Item | Description |
|------|--|
| 4 | <p>Global Types pane.</p> <p>Displays the types whose instances are used in the selected section or by selected variables.</p> <p>A colored dot next to each type indicates the following:</p> <ul style="list-style-type: none"> Green dot: The type or type version is new and will be added to the content repository. Gray dot: The type with the same name and version already exists in the content repository; it will not be added. You can encapsulate the constituents. <p>NOTE: Position the pointer over a dot to view a description in a tooltip.</p> |
| 5 | Button to open the Control Participant window, which lets you view and/or modify the Control resources, page 270. |
| 6 | Button to save changes that you have made in the Select Variables window. |
| 7 | Button to close the Select Variables window. |
| 8 | <p>Indicates the space (columns x rows) occupied by Control resources (for example, the DFB instance) in the section that is selected in the Sections tab. The values include the value specified by Extension.</p> <p>To modify the values, open the Control Participant. Then, move the resource inside the section and save changes.</p> <p>For details, refer to the example, page 266.</p> |
| 9 | <p>Indicates the number of columns (to the right of the resource) and rows (below the resource) that are added as free space during generation.</p> <p>You can modify the parameter value by editing the value in the Select Variables window.</p> <p>For details, refer to the example, page 266.</p> |
| 10 | <p>For each parameter of an FFB and child variable of a DDT, you can select the <i>\$FullName</i> property, page 45, which makes the corresponding parameter name available at an output pin for further processing.</p> <p>You can select the property even if you do not selected the corresponding parameter or variable itself.</p> |
| 11 | <p>The check box is available for pins to which a variable is assigned.</p> <p>When the <i>check box</i>, page 264 is selected, selects the corresponding variable in the Variables pane and adds it to the center pane to be encapsulated as a separate variable element. The parameter of the DFB however is cleared, becomes read-only, and will not be part of the Control element.</p> <p>Clearing the check box lets you select the parameter of the DFB so that it will be available in the Control element and, optionally, lets you select the variable to create it as a separate variable element.</p> <p>NOTE: When the check box is selected, click the arrow to highlight the corresponding variable in the center pane.</p> |

Encapsulating Constituents

To encapsulate constituents of the Control Participant into a Control facet template that does not contain constituents yet, proceed as follows.

| Step | Action |
|------|--|
| 1 | <p>In the Global Templates Explorer, right-click the Control facet template and select Edit.</p> <p>Result: The Facet Editor opens.</p> |
| 2 | <p>Click Templatizer.</p> <p>Result: The Content Not Found dialog box opens.</p> |
| 3 | <p>Select Select Existing Project and click OK.</p> <p>Result: The Add Project dialog box opens.</p> <p>NOTE: To create constituents instead, refer to the topic describing how to create Control constituents, page 139.</p> |

| Step | Action |
|------|--|
| 4 | <p>Browse to a compatible project file (.stu) that contains the constituents that you want to encapsulate, select it and click Open.</p> <p>Result: The Control Participant window (see <i>EcoStruxure Process Expert, User Guide</i>) opens.</p> |
| 5 | <p>If required, by using the Control Participant, you can:</p> <ul style="list-style-type: none"> • View the constituents that are contained in the project file. • Modify the constituents. • Create new constituents, page 137. <p>You can, for example, reposition a DFB in a section, configure attributes, or create variables. Refer to the topic describing Control constituent guidelines, page 137.</p> <p>When done, save your changes by clicking the Save button .</p> <p>For more information about interactions with the Control Participant, refer to <i>EcoStruxure Process Expert, Control Participant Services, User Guide</i>.</p> |
| 6 | <p>Close the Control Participant window.</p> <p>Result: The Select Variables window opens.</p> <p>NOTE: To open the Control Participant window again, click Open Participant.</p> |
| 7 | <p>In the Sections and/or Variables panes of the Select Variables window, select the FBD section and/or variables that you want to encapsulate in the Control facet template.</p> |
| 8 | <p>Refine your selection by selecting one or more items in the Name column of the center pane.</p> |
| 9 | <p>When you are done with the constituent selection, click the Save button in the Select Variables window.</p> <p>Result: The software stores a copy of the project file and files of selected constituents in the content repository as global constituents.</p> <p>NOTE: Closing the Select Variables window without saving discards your selection and changes that you have made in the Control Participant.</p> |
| 10 | <p>Close the Select Variables window.</p> <p>Result: The view reverts to the Facet Editor and the selected constituents appear in the Elements section of the Document Outline pane. You can drag them to the workspace to use them.</p> |
| 11 | <p>Save the changes, page 254 in the facet template.</p> <p>Result: The software stores a copy of the project file and files of selected constituents in the content repository as local constituents under the name of the facet template.</p> <p>NOTE: If you have not dragged the constituents to the workspace, they remain in the Elements section of the Document Outline pane when you save changes.</p> |

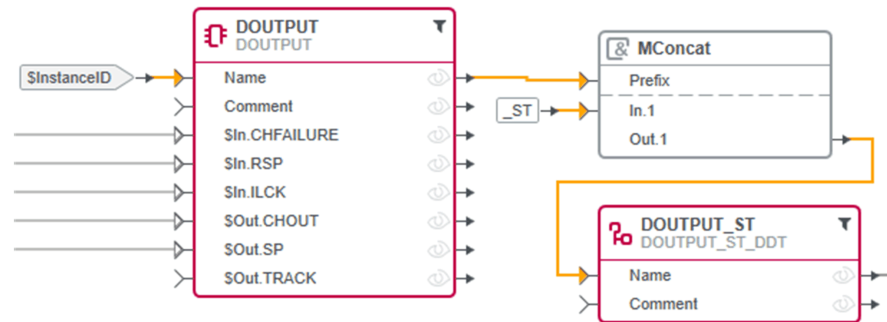
Viewing and Retrieving Constituent Files

To view and/or retrieve the set of files that the software adds to the content repository once you have completed the encapsulation process, proceed as follows.

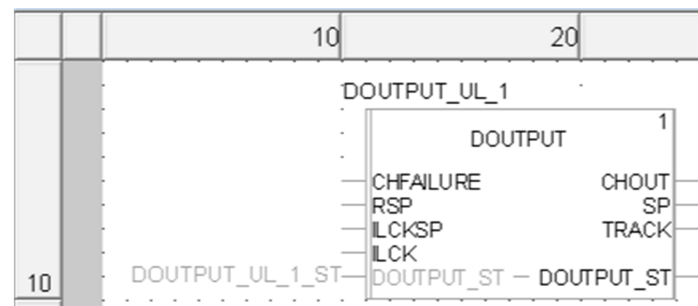
| Step | Action |
|------|--|
| 1 | Open the content repository. |
| 2 | Expand Global Root > Global Templates > Local Constituents > Upro . |
| 3 | <p>Browse to the identifier of the template in which you have encapsulated Control constituents, right-click the template, and select View Content Detail.</p> <p>Result: The software opens a tab in which you can view and retrieve the constituent files, for example, the project file (.stu).</p> |

NOTE: For more information, refer to the chapter describing the content repository (see *EcoStruxure Process Expert, User Guide*).

The following figure shows the necessary bindings that have been made between the two Control elements at the facet template level (shown in orange).



The following figure shows an example of the corresponding Control constituents that are generated in the logical Control project based on these bindings: An instance of the *DOUTPUT* DFB and the *DOUTPUT_UL_1_ST* structured variable (where *DOUTPUT_UL_1* is the instance identifier).

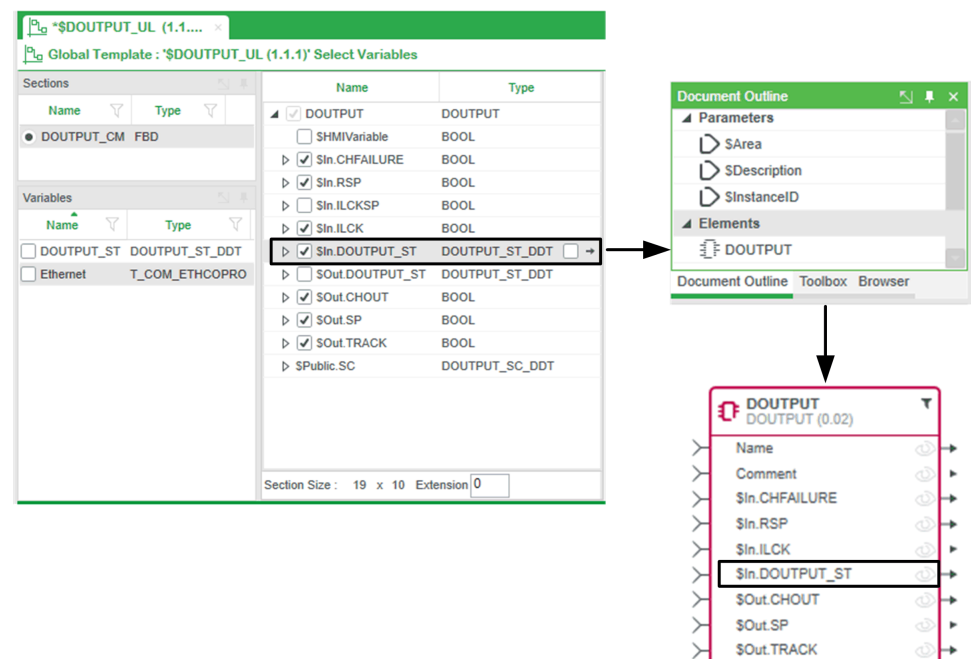


In the second example, the check box is cleared.

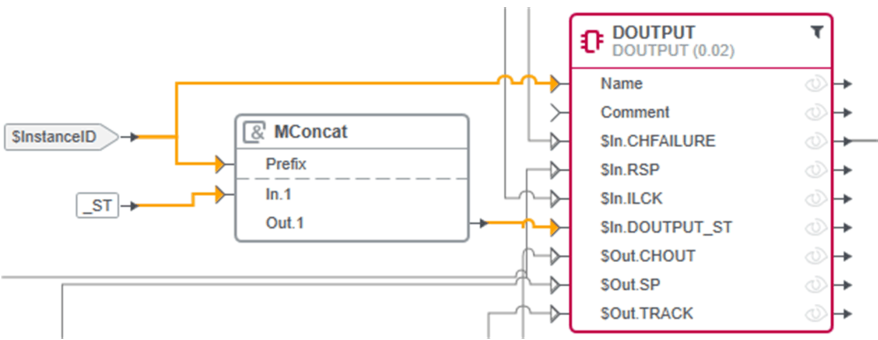
As a result, in the center pane:

- The check box of `$In.DOUTPUT_ST` itself can be selected. When selected, the parameter will become a property of the *DOUTPUT* Control element (DFB).
- The *DOUTPUT_ST* structure variable can be selected. If it is not, it will not be created as a separate variable element.

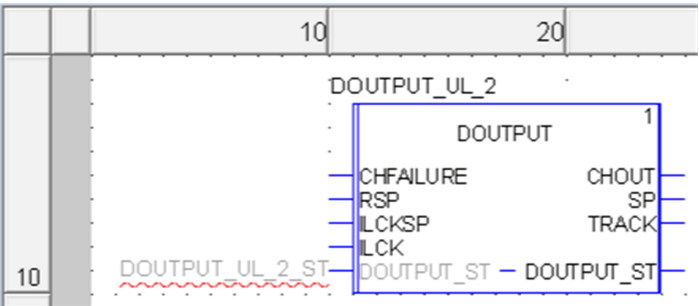
By saving the configuration shown below, the corresponding Control element appears in the **Document Outline** pane and, once dragged to the workspace, its properties reflect the configuration.



The following figure shows the necessary bindings that have been with the Control element at the facet template level (shown in orange).



The following figure shows an example of the corresponding Control constituent that is generated in the logical Control project based on these bindings: An instance of the *DOUTPUT* DFB, which is the same as the one in the first example. The *DOUTPUT_UL_2_ST* structured variable (where *DOUTPUT_UL_2* is the instance identifier) is assigned to the pin of the DFB but not declared.

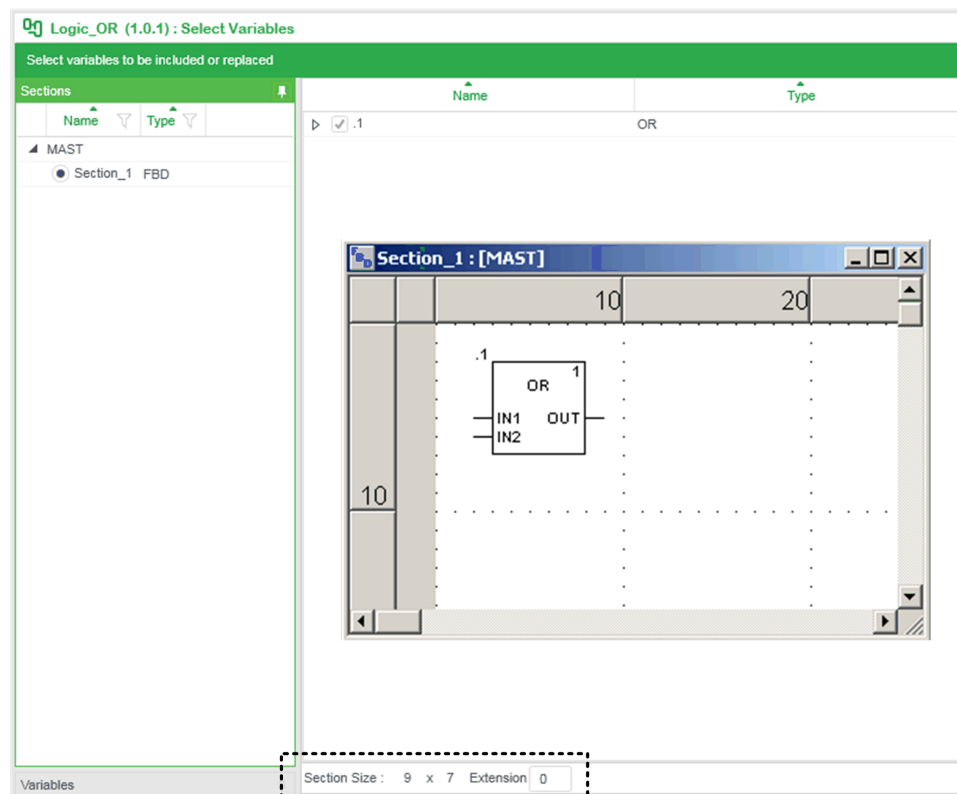


NOTE: In both examples, it can be seen that although the *\$In.ILCKSP* parameter of the *DOUTPUT* DFB is not selected in the **Select Variables** window and, as a result, is not a property of the *DOUTPUT* Control element, it is present as a parameter of the DFB instance.

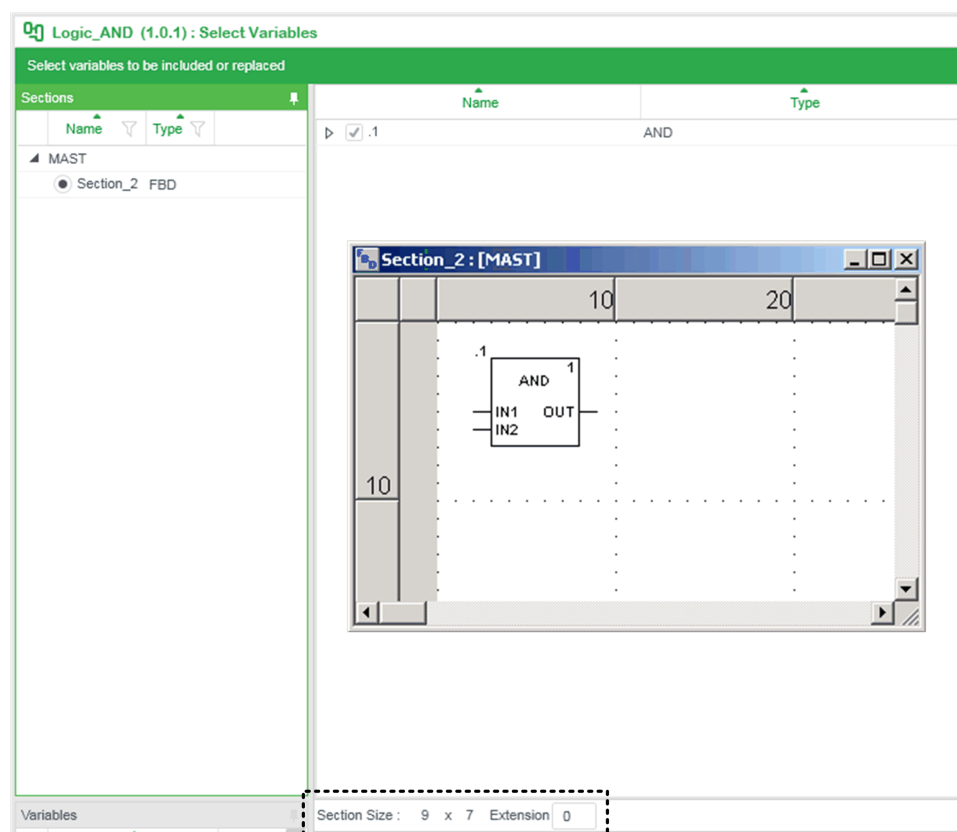
Constituent Positioning Example

This example uses two elementary functions (EFs) to illustrate which parameters affect the position of resources during encapsulation and generation.

The OR function is encapsulated in the *Logic_OR* Control facet template. The EF is positioned to leave 2 columns of free space to the left and 1 row above. The total space occupied by the EF is indicated at the bottom of the **Select Variables** window after saving changes in the Control Participant. No extension space is configured.



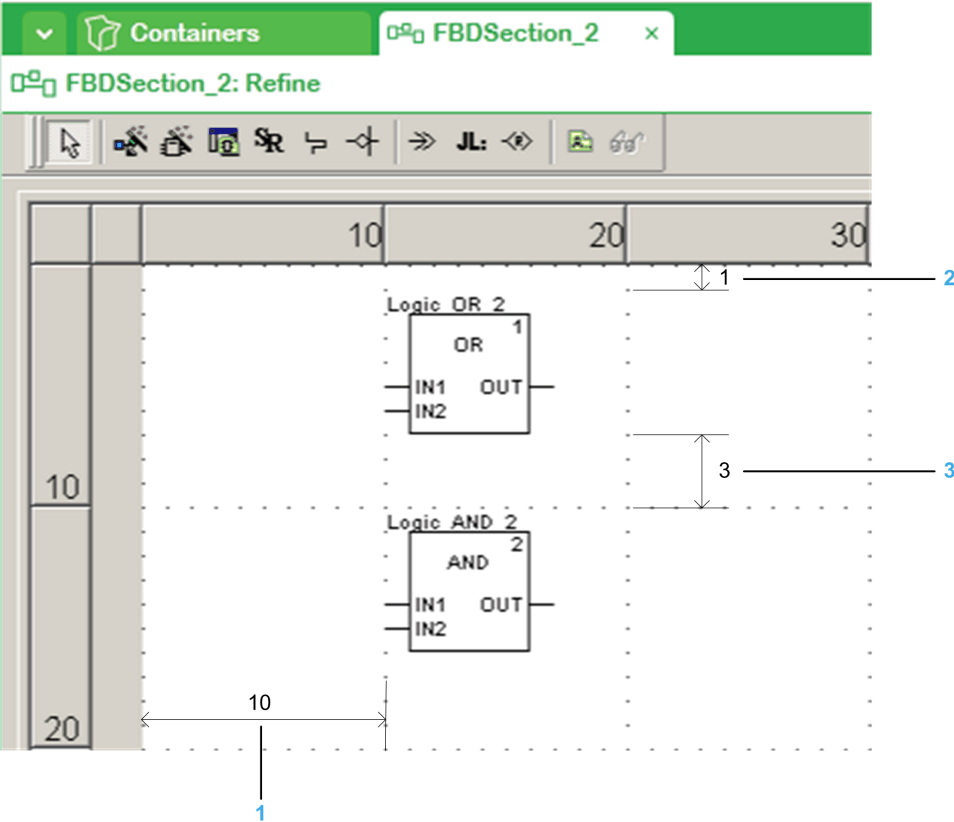
In the same way, the AND function is encapsulated in the *Logic_AND* Control facet template. The EF is positioned to leave 2 columns of free space to the left and 1 row above. The total space occupied by the EF is indicated at the bottom of the **Select Variables** window after saving changes in the Control Participant. No extension space is configured.



Both templates are instantiated and their facet assigned to one section of a Control Participant project so that the facet containing the OR EF is generated first.

| MAST | | | | | | | | | | |
|----------------------------|-------|--------------|-------------|----------------|-------|-------------|---------------|------|-------|------------|
| Containers | | | | | | | | | | |
| FBDSection_2 - Assignments | | | | | | | | | | |
| Identifier | Order | Container | Instance | Instance Te... | State | Facet | Facet Temp... | Path | Order | Assignment |
| FBDSection_2_0 | | FBDSection_2 | Logic_OR_3 | Logic_OR | Valid | Logic_OR_3 | Logic_OR | | 0 | Assigned |
| | | FBDSection_2 | Logic_AND_3 | Logic_AND | Valid | Logic_AND_3 | Logic_AND | | 1 | Assigned |

The following figure shows the result when you refine the section.



| Item | Description |
|------|--|
| 1 | The 10 columns free space to the left of the EF are created because of the value of the <i>UnityMinSpaceLeft</i> parameter in the <i>GenerationSettings.xml</i> file: <ul style="list-style-type: none">When the position of the resource on the x-axis inside the template is less than the value of <i>UnityMinSpaceLeft</i>, during generation the resource is moved to <i>UnityMinSpaceLeft</i> (x-axis), like in this example.When the position of the resource on the x-axis inside the template is equal or greater than the value of <i>UnityMinSpaceLeft</i>, during generation the resource is not moved. |
| 2 | The 1 row of space above the EF comes from the position of the resource on the y-axis inside the template. |
| 3 | The 3 rows of space between the two EFs come from the sum of the following: <ul style="list-style-type: none">1 row of space above the AND EF (y-axis) due to the position of the resource inside the template.2 rows of space (y-axis) created because of the value of the <i>UnityMinDistanceY</i> parameter in the <i>GenerationSettings.xml</i> file. |

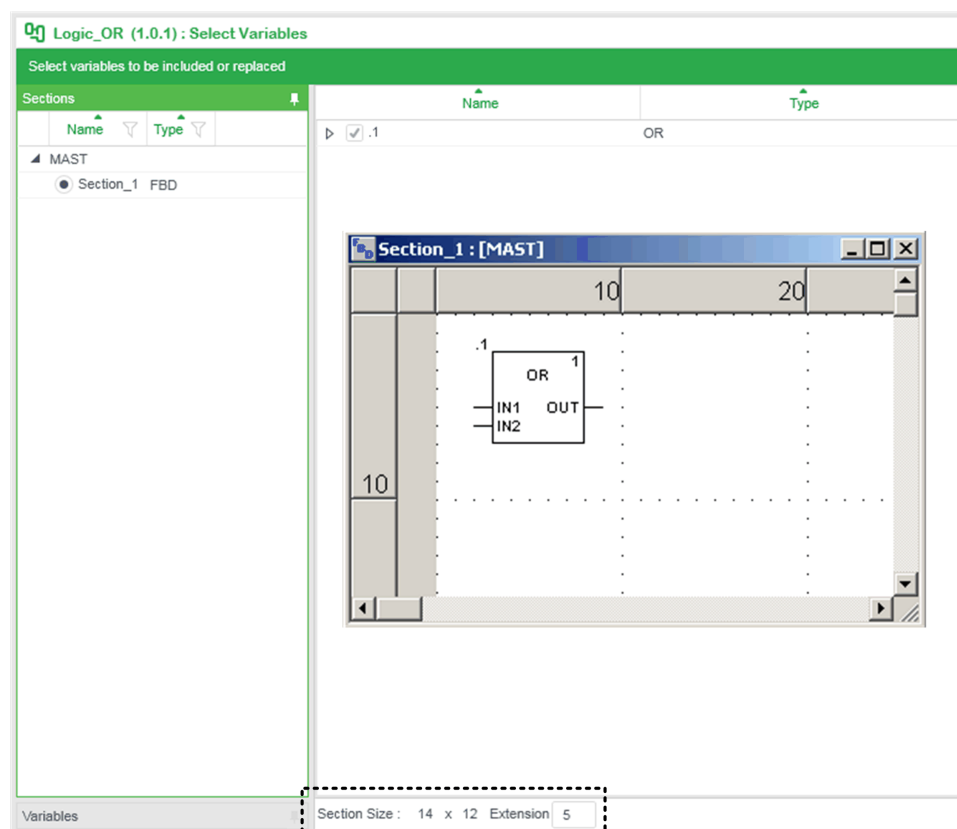
The following figure shows the default values of the previously mentioned parameters in the *GenerationSettings.xml* file.

```

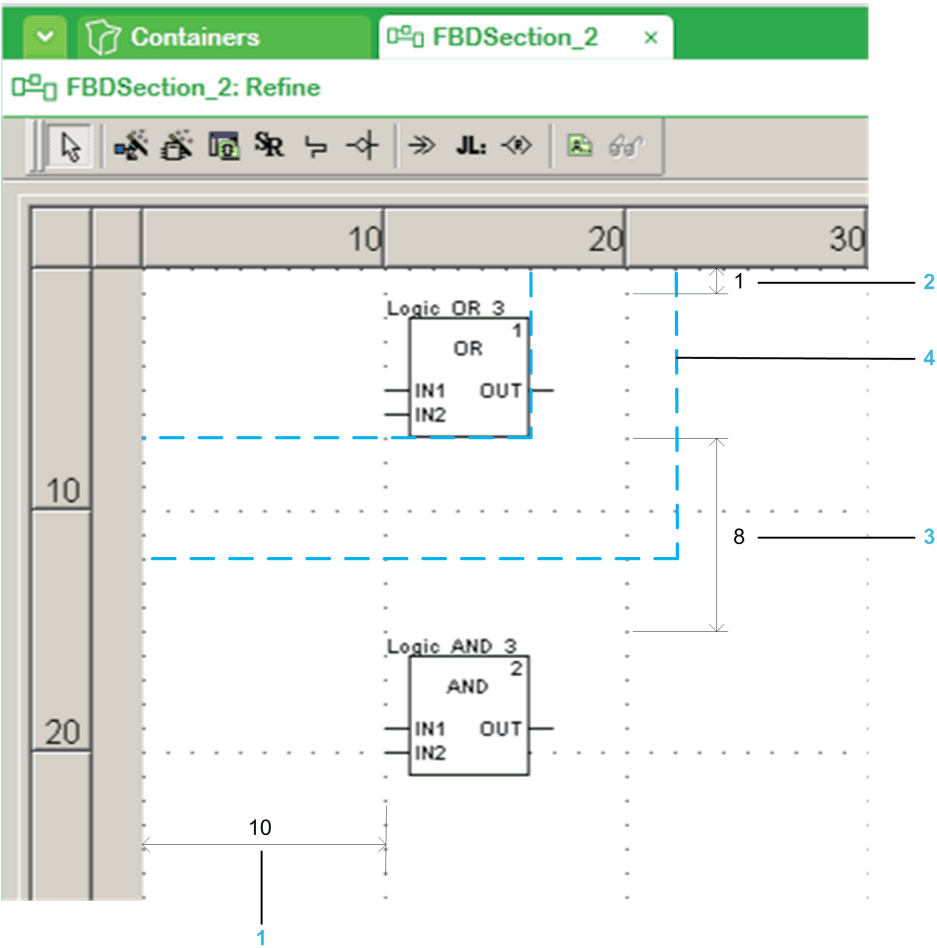
1  <?xml version="1.0" encoding="utf-8" ?>
2  <GenerationSettings>
3    <parameters>
4      <parameter name="ReGenerateNonFbdSections" value="true"/>
5      <parameter name="UnityProIgnoreFbdComments" value="true"/>
6      <parameter name="UnityProMinSpaceTop" value="0"/>
7      <parameter name="UnityProMinSpaceLeft" value="10"/>
8      <parameter name="UnityProMinDistanceX" value="5"/>
9      <parameter name="UnityProMinDistanceY" value="2"/>
10     <parameter name="UnityProUseEffectiveSize" value="false"/>
11     <parameter name="UnityProInsertToEnd" value="true"/>
12     <parameter name="CheckConsistency" value="true"/>
13     <parameter name="SaveAfterNumberSectionGenerated" value="30"/>
14     <parameter name="NumberOfUnitySectionRows" value="36"/>
15     <parameter name="NumberOfUnitySectionColumns" value="24"/>
16   </parameters>
17 </GenerationSettings>

```

To illustrate the purpose of the *Extension* parameter, the template encapsulating the OR EF is edited and *Extension* is set to 5 without moving the resource. Each value of **Section Size** is automatically increased by 5 (from 9 x 7).



The updated *Logic_OR* template and the *Logic_AND* template are assigned to a new section and generated in the same order as before (OR first). The following figure shows the result when you refine the section.



| Item | Description |
|------|--|
| 1 | The 10 columns free space to the left of the EF are unchanged. |
| 2 | The 1 row of space above the EF is unchanged. |
| 3 | The 3 rows of space between the two EFs are increased to 8 because and consist of: <ul style="list-style-type: none">• 1 row of space above the AND EF (y-axis) inside the template.• 2 rows of space (y-axis) created because of the value of the <i>UnityMinDistanceY</i> in the <i>GenerationSettings.xml</i> file.• 5 rows of space added by using the <i>Extension</i> parameter. |
| 4 | The dotted lines show the additional free space of 5 columns and 5 rows that are created to the right and below the OR EF respectively because of the <i>Extension</i> parameter setting. |

NOTE: For more information on the function block layout and the *GenerationSettings.xml* file, refer to the topic describing the generation stage, Control project specifics (see *EcoStruxure Process Expert, User Guide*).

Modifying Encapsulated Control Constituents

Overview

Once you have encapsulated Control constituents in a Control facet template by using the **Facet Editor**, you can modify:

- The constituent selection.
- The Control resources contained in the project file (.stu).

Modifying Constituents in the Project File

For more information about interactions with the Control Participant, refer to the *Control Participant Services User Guide* in the software help.

Select Variables Window

Modifying the constituent selection is done by using the **Select Variables** window, page 261.

Modifying the Constituent Selection

To modify the constituent selection in a Control facet template, proceed as follows.

| Step | Action |
|------|--|
| 1 | In the Global Templates Explorer , right-click the Control facet template whose element selection you want to modify and select Edit . Result: The facet opens in the Facet Editor . |
| 2 | Click Templatizer . Result: The Select Variables window opens. The constituents that are shown as selected correspond to the elements that are encapsulated in the facet template. |
| 3 | In the Sections and/or Variables panes, proceed with the modification of the constituent selection: <ul style="list-style-type: none"> Select additional constituents that you want to add to the facet template. Clear constituents that you want to remove from the facet template. |
| 4 | In the center pane, refine your selection as needed. |
| 5 | Optionally, you can modify the Control resources, page 271. |
| 6 | Save your changes in the Select Variables window. Result: The elements of the facet template are updated according to your selection. Newly selected elements are displayed in the Elements section of the Document Outline pane. NOTE: Elements that you clear in the Select Variables window are removed from the workspace if you had already dragged them there. |
| 7 | Close the Select Variables window. |
| 8 | Save the changes, page 254 in the facet template. Result: The software updates the constituent files that are stored in the content repository to reflect your changes. |

Modifying the Control Resources

To modify the Control resources contained in the project file, proceed as follows.

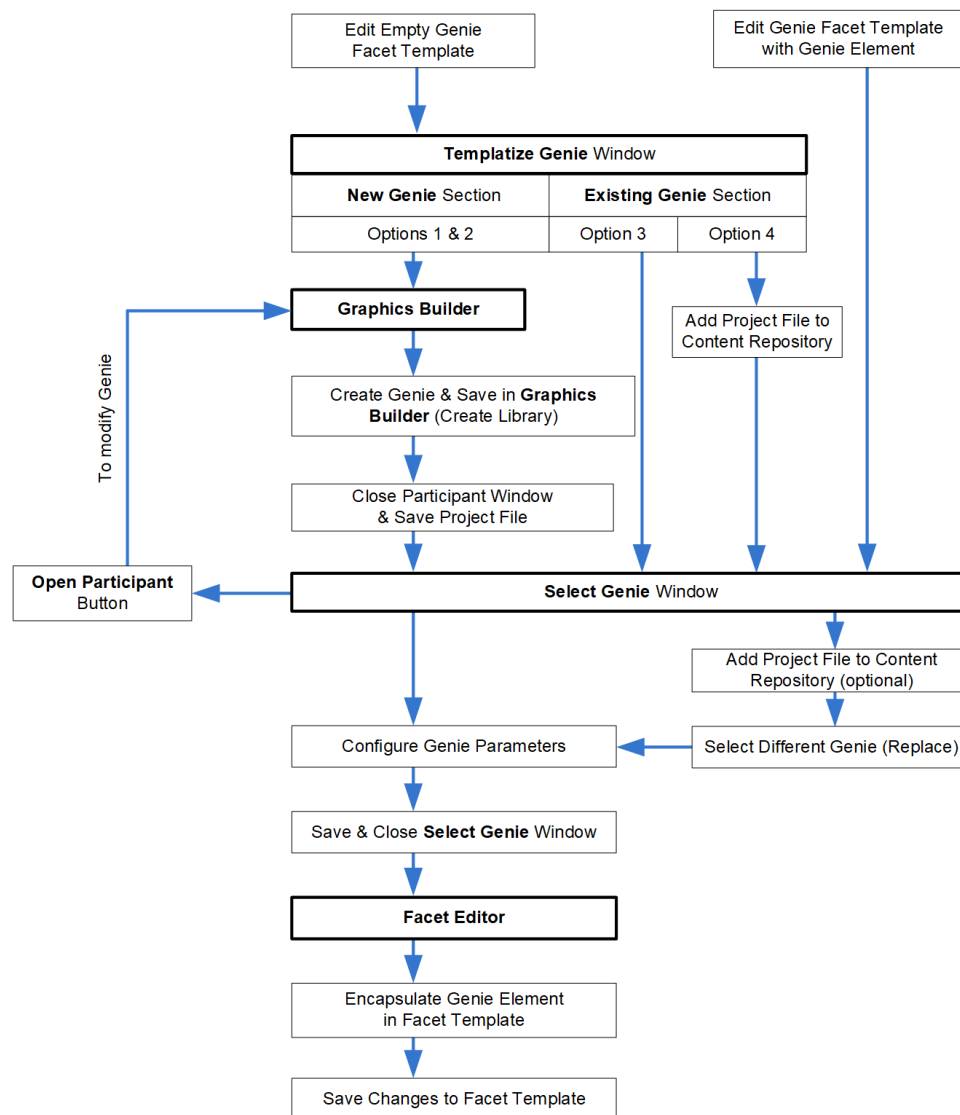
| Step | Action |
|------|---|
| 1 | In the Global Templates Explorer , right-click the Control facet template whose Control resources you want to modify and select Edit . Result: The facet opens in the Facet Editor . |
| 2 | Click Templatizer . Result: The Select Variables window opens. The constituents that are shown as selected correspond to the elements that are encapsulated in the facet template. |

| Step | Action |
|------|---|
| 3 | Click Show Templatizer Tool . Result: The Control Participant window opens. |
| 4 | Proceed with the modification of the Control resources. NOTE: Do not use types whose name starts with a digit. |
| 5 | Save your changes and close the Control Participant window. Result: The Select Variables window opens. |
| 6 | Modify the constituent selection, page 271 as needed. |
| 7 | Save your changes in the Select Variables window. Result: The elements of the facet template are updated according to your selection. Newly selected elements are displayed in the Elements section of the Document Outline pane. NOTE: Closing the Select Variables window without saving discards your selection and changes that you have made in the Control Participant. |
| 8 | Close the Select Variables window. |
| 9 | Save the changes, page 254 in the facet template. Result: The software updates the constituent files that are stored in the content repository to reflect your changes. |

Genie Creation Workflows

The following figure shows the various workflows to create and modify Supervision animated graphics (Genies). By using the Graphics Builder of the Supervision Participant, you can also create and modify other Supervision resources such as pages, faceplates, and labels.

Each workflow is described more in detail in the next topics of this chapter.



Using, Creating, and Modifying Supervision Animated Graphics

Overview

This topic describes how to perform the following actions with Supervision animated graphics (Genies) and new *Genie* facet templates:

- Create a new Genie by using the Graphics Builder of the Supervision Participant.
- Use an existing Genie.
- Create a Genie based on an existing one and modify it.

You can use a project file (Include project (.ctz)) that is already present in the content repository (see *EcoStruxure Process Expert, User Guide*), add an external one, or create a new project and add it to the content repository.

You can create or modify several Genies at a time per Include project. In addition to Genies, you can create or modify Supervision pages, faceplates, and/or labels.

In a second step, you can encapsulate the Genie, page 278 in a Supervision *Genie* facet template as an element.

For an overview of the process, refer to the topic describing the Genie creation workflows, page 272.

NOTE: The software deploys project files that you add to the content repository together with the Supervision project containing instances of the Supervision *Genie* facet template.

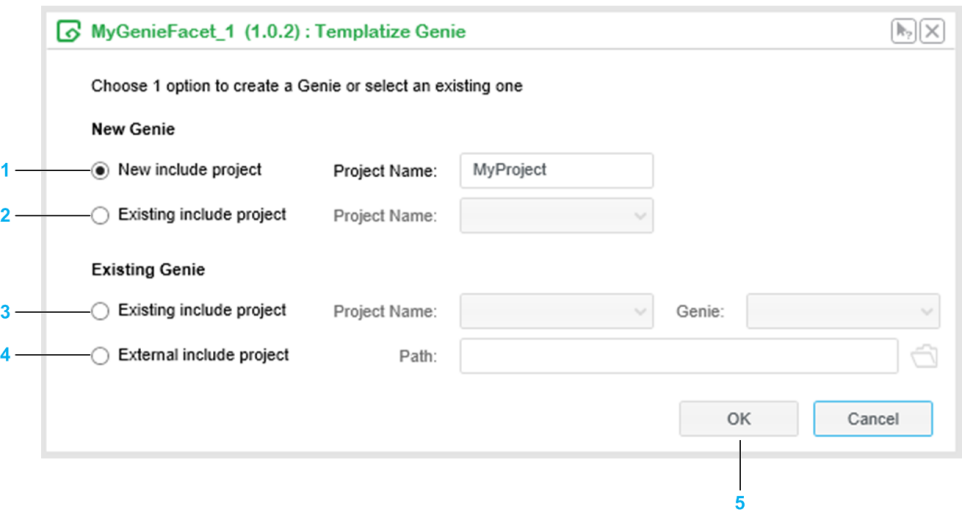
NOTE: The modification of a Genie that is already encapsulated in a *Genie* facet template is described in a separate topic, page 281.

Before Creating Genies

Before creating Genies, complete the preparation work, page 141 to have an in-depth understanding of the template that you want to create. This helps you create facet templates and interfaces that communicate with each other.

Templatize Genie Window Description

The following figure shows an example of the **Templatize Genie** window that opens when you click **Templatizer** while editing an empty *Genie* facet template. Information was entered in the text boxes of the first option.



| Item | Description | |
|------|---|---|
| 1 | Select this option to create a new Genie in a new Include project that is added to the content repository. | |
| | Project Name | Enter the name of the Include project that you want to create. Underscore is the only allowed special character. NOTE: You can create the library when you save your Genie. |
| 2 | Select this option to create a new Genie in an Include project from the content repository. | |
| | Project Name | Select from a list of Include projects that exist in the content repository. NOTE: You can select an existing library or create a new one when you save your Genie. |
| 3 | Select this option to use an existing Genie from an Include project from the content repository. NOTE: This option lets you also edit the existing Genie, page 282 and/or create a new Genie and save it in an existing or new library of the selected project. | |
| | Project Name | Select from a list of Include projects from the content repository. The Include projects that are available depend on the Schneider Electric template libraries that are installed. |
| | Genie | Select from a list of Genies that exist in the libraries of the selected Include project. NOTE: Genies that are used in templates of a Schneider Electric library are described in the help of the library. Refer to the <i>Genies</i> topic in the <i>Supervision</i> section of the help. |

| Item | Description | |
|------|--|---|
| 4 | Select this option to use a Genie from a library of an external Include project, which will be added to the content repository. NOTE: This option lets you also edit the existing Genie, page 282 and/or create a new Genie and save it in an existing or new library of the selected project. | |
| | Path | Enter the path to an Include project (.ctz) on the network or click the browse button to open the Select Project dialog box and select a project file. |
| 5 | <ul style="list-style-type: none"> For a New Genie, opens the Graphics Builder of the Supervision Participant to create the Genie and encapsulate it. For an Existing Genie, opens the Select Genie window, page 278, which shows the libraries of the selected Include project and the Genies they contain. <ul style="list-style-type: none"> For an Existing Include project, automatically highlights the Include project and library that contains the Genie that you select. For an External Include project, automatically adds the Include project to the content repository even if you do not encapsulate a Genie. | |

NOTE: Switching options discards the data that you entered or selected.

Project File Version

When you add an external Include project to the content repository, you can only use a project file that was created with a version of Plant SCADA that is the same as the Supervision Participant.

For more information about the Supervision Participant version, refer to the platform release notes.

Creating a New Genie

To create a new Genie in a new Include project or in one that exists in the content repository, proceed as follows.

| Step | Action |
|------|--|
| 1 | Create and edit a facet template of type <i>Genie</i> from the Global Templates Explorer . |
| 2 | In the Facet Editor , click Templatizer . Result: The Templatize Genie window opens. |
| 3 | Select one of the two options of the New Genie section and enter or select the appropriate information. Click OK . Result: The Graphics Builder opens in a Supervision Participant window. |
| 4 | Create the Genie, configure its parameters, page 280, and save it by using the Save button of the Graphics Builder. Result: The Save As dialog box opens. |
| 5 | In the Project list, select the project that you entered or selected in step 3. If required click New to create a library, enter a name for the Genie, and click OK . NOTE: If you save the Genie in a different project than the one you entered or selected in the Templatize Genie window, you cannot access the Genie once you close the Supervision Participant window. |
| 6 | Ensure to save other changes that you make thereafter in the Graphics Builder. Unsaved changes in the Graphics Builder are discarded when you close the Supervision Participant window and save changes to the project. |
| 7 | Click the Close button of the Participant window. Result: The Save dialog box opens, which lets you save changes to the project in the content repository. |
| 8 | Select a versioning scheme, enter a change description, and click Yes . Result: The Supervision Participant window closes and the Select Genie window opens in the Facet Editor . The project and the library are selected in the left-hand pane. The Genie that you created is represented as an icon in the center pane. If the library contains other Genies, they appear as well. NOTE: Saving changes to a large Include project may take time. NOTE: To modify the Genie, click Open Participant to open the Graphics Builder again. |
| 9 | Follow the steps of the procedure describing how to encapsulate a Genie, page 278. |

Using an Existing Genie

The following procedure describes how to use a Genie from an existing Include project or from one that you are adding to the content repository.

You cannot store several versions of the same Include project in the content repository.

Replacing an Include project file in the content repository can affect the functionality of templates and their instances.

NOTICE

LOSS OF DATA

Resolve a project file conflict that arises when adding a project file to the content repository by selecting the correct file version.

Failure to follow these instructions can result in equipment damage.

| Step | Action |
|------|--|
| 1 | Create and edit a facet template of type <i>Genie</i> from the Global Templates Explorer . |
| 2 | In the Facet Editor , click Templatizer . Result: The Templatize Genie window opens. |
| 3 | By using one of the two options in the Existing Genie section of the Templatize Genie window, select an Include project from the list or browse to an external Include project and click OK . Result: If you selected an external project and a project with the same identifier already exists in the content repository, the Replace Project dialog box opens. Otherwise, the Select Genie window opens in a Supervision Participant window. The project that you selected and the first library in alphabetical order are selected in the left-hand pane. The Genies of this library appear as icons in the center pane. Proceed to step 6. |
| 4 | In the Replace Project dialog box, verify the information that is displayed: <ul style="list-style-type: none"> The project file name. Current version: Version of the file that exists in the content repository. New version: Version of the file that you selected. |
| 5 | Click either button: <ul style="list-style-type: none"> Yes: Overwrites the existing project file with the selected project file and opens the Select Genie window. No: Cancels without adding the project file to the content repository. |
| 6 | In the Select Genie window, select a library and Genie. NOTE: To view the Genie, click Open Participant to open the Graphics Builder. |
| 7 | Follow the steps of the procedure describing how to encapsulate a Genie, page 278. |

Creating a Genie Based on an Existing one and Modifying it

The following procedure describes how to create a Genie based on an existing one within the same Include project.

You can add an Include project to the content repository but you cannot store several versions of the same Include project in the content repository.

Replacing an Include project file in the content repository can affect the functionality of templates and their instances.

NOTICE

LOSS OF DATA

Resolve a project file conflict that arises when adding a project file to the content repository by selecting the correct file version.

Failure to follow these instructions can result in equipment damage.

NOTE: If you want to modify a Genie that is already encapsulated in a *Genie* facet template, follow a [specific procedure](#), page 281.

| Step | Action |
|------|---|
| 1 | Create and edit a facet template of type <i>Genie</i> from the Global Templates Explorer . |
| 2 | In the Facet Editor , click Templatizer . Result: The Templatize Genie window opens. |

| Step | Action |
|------|--|
| 3 | <p>By using one of the two options in the Existing Genie section of the Templatize Genie window, select an Include project from the list or browse to an external Include project and click OK.</p> <p>Result: If you selected an external project and a project with the same identifier already exists in the content repository, the Replace Project dialog box opens.</p> <p>Otherwise, the Select Genie window opens in a Supervision Participant window. The project that you selected and the first library in alphabetical order are selected in the left-hand pane. The Genies of this library appear as icons in the center pane. Proceed to step 6.</p> |
| 4 | <p>In the Replace Project dialog box, verify the information that is displayed:</p> <ul style="list-style-type: none"> The project file name. Current version: Version of the file that exists in the content repository. New version: Version of the file that you selected. |
| 5 | <p>Click either button:</p> <ul style="list-style-type: none"> Yes: Overwrites the existing project file with the selected project file and opens the Select Genie window. No: Cancels without adding the project file to the content repository. |
| 6 | <p>In the Select Genie window, select a library and Genie and click Open Participant.</p> <p>Result: The selected Genie opens in the Graphics Builder inside a Supervision Participant window.</p> |
| 7 | Click File > Save As . |
| 8 | <p>In the Project list, select the project that you selected in step 3, select a library or create a new one, enter a name for the Genie that you are creating, and click OK.</p> <p>Result: A copy of the Genie is created.</p> <p>NOTE: If you save the Genie in a different project than the one you selected in the Templatize Genie window, you cannot access the Genie once you close the Supervision Participant window.</p> |
| 9 | Modify the Genie, configure its parameters, page 280, and save it by using the Save button of the Graphics Builder. |
| 10 | Ensure to save other changes that you make thereafter in the Graphics Builder. Unsaved changes in the Graphics Builder are discarded when you close the Supervision Participant window and save changes to the project. |
| 11 | <p>Click the Close button of the Participant window.</p> <p>Result: The Save dialog box opens, which lets you save changes to the project in the content repository.</p> |
| 12 | <p>Select a versioning scheme, enter a change description, and click Yes.</p> <p>Result: The Supervision Participant window closes and the Select Genie window opens in the Facet Editor. The project and the library are selected in the left-hand pane. The Genie that you created is represented as an icon in the center pane. If the library contains other Genies, they appear as well.</p> <p>NOTE: Saving changes to a large Include project may take time.</p> <p>NOTE: To modify the Genie, click Open Participant to open the Graphics Builder again.</p> |
| 13 | Follow the steps of the procedure describing how to encapsulate a Genie, page 278. |

Encapsulating and Configuring Genies

Overview

The **Select Genie** window lets you encapsulate in an empty Supervision *Genie* facet template a Genie contained in a project file (Include project (.ctz)) that is already present in the content repository.

The window opens when you perform the following actions:

- Select an option of the **Existing Genie** section of the **Templatize Genie** window, page 273.
- Save changes to an Include project in the Graphics Builder of the Supervision Participant after creating or modifying a Genie, page 273.

Once encapsulated, the Genie becomes an element of the Supervision *Genie* facet template, page 252 and is part of its definition.

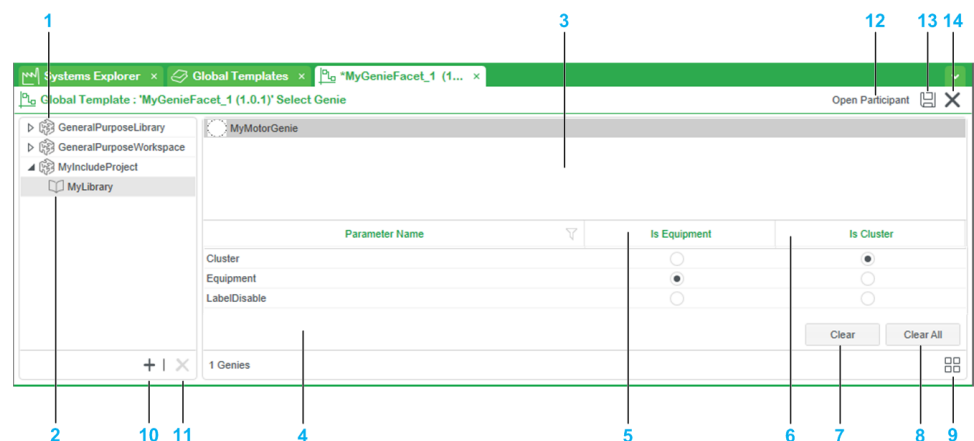
NOTE: The **Select Genie** window also opens when you click **Templatizer** in the **Facet Editor** while editing, page 281 a *Genie* facet template that already contains a Genie element.

OPC Items

Before encapsulating a Genie, verify which OPC items the genie uses because data that is propagated from and to Control resources by interfaces needs to be consistent.

Select Genie Window

The following figure shows an example of the **Select Genie** window for a user-created *Genie* facet template.



| Item | Description |
|------|--|
| 1 | Pane showing the Include project files that are present in the content repository and those that you have added. |
| 2 | List of libraries that each project file contains. |
| 3 | Genies that the selected library contains and that you can encapsulate. NOTE: When you open the Select Genie window for a <i>Genie</i> facet template in which a Genie is already encapsulated, this animated graphic is automatically selected. |
| 4 | Pane listing the parameters that are available for the selected Genie. |
| 5 | Lets you select which parameter contains the equipment name. It corresponds to the value of the Name property, page 280 of the equipment instance. You can select only one parameter and it cannot be selected if it is already used for another property (for example, Is Cluster). Selecting a parameter is optional. |
| 6 | Lets you select which parameter contains the cluster name. It corresponds to the value of the Cluster Name property, page 280 of the equipment instance. You can select only one parameter and it cannot be selected if it is already used for another property (for example, Is Equipment). Selecting a parameter is optional. |
| 7 | Clears the selection for the parameter that is selected. |

| Item | Description |
|------|--|
| 8 | Clears the selection for the parameters of the selected Genie. |
| 9 | Toggles the display mode of the Genies pane between grid view and icon view. |
| 10 | Lets you add a project file (.ctz), page 281 to the content repository. Once added, you can view its content and select a Genie. |
| 11 | Deletes the selected project file from the content repository. You can only delete a project file that you have added and if no resource of the project file is used in a template or Supervision project. |
| 12 | Opens the Graphics Builder in a Supervision Participant window, which lets you modify the selected Genie. |
| 13 | Creates an element containing the selected Genie. NOTE: When you open the Select Genie window for a facet template in which Genie is already encapsulated and have modified the parameter selection for Is Equipment and Is Cluster , applies the changes to the Genie element. Existing bindings may be discarded. NOTE: When you replace a Genie element, the software retains existing bindings if possible. |
| 14 | Closes the Select Genie window and discards unsaved changes. |

Selecting Parameters for Equipment and Cluster Names

To take advantage of the equipment within a Supervision Participant project, you can select two parameters of a Genie that will be used for equipment name and cluster name. The combination of the **Equipment** and **Item Name** properties is used to reference the corresponding controller data.

Once you have encapsulated the Genie in a Supervision *Genie* facet template, the two selected parameters do not appear anymore in the **Facet Editor** and are managed by the software.

To modify the parameter selection of an already encapsulated Genie, use the **Templatizer** button in the **Facet Editor** to show its current parameter selection and edit it. If you select a parameter that was already connected and apply changes, the parameter is hidden and bindings are discarded.

Populating the Selected Genie Parameters

The value of the Genie parameters that you have selected to contain the equipment name and cluster name is set automatically as soon as a value for the corresponding property is generated or updated in the equipment instance of the Supervision project.


The equipment instance is the one that is generated by the Supervision *Genie* facet (or by the instance in case you have selected to generate equipment based on the hierarchy).

For information on how equipment instances are created and the values of their properties are generated and updated, refer to the topic describing the generation of equipment (see *EcoStruxure Process Expert, User Guide*).

Encapsulating a Genie

The following procedure describes how to encapsulate in an empty *Genie* facet template a Genie from an Include project that is already present in the content repository.

It requires that you have already followed the [procedure, page 273](#) to create, modify, or use Genie and the **Select Genie** window is open.

| Step | Action |
|------|---|
| 1 | For the selected Genie, in the Is Equipment and Is Cluster columns, select one parameter, page 280 for each. |
| 2 | Click the save button  and then, the close button of the Select Genie window. Result: The <i>Genie</i> facet template is shown and the selected Genie appears as Genie in the Elements section of the Document Outline pane of the editor. |
| 3 | Drag the Genie from the Elements section to the workspace. Result: The Genie appears as element with its configurable parameters. |
| 4 | Select the Genie and edit its identifier and description in the Properties pane, page 227. |
| 5 | Save the changes, page 254. |

Modifying and Replacing Encapsulated Genies

Overview

By using the **Facet Editor** and **Select Genie** window, page 279, you can perform the following actions on a Genie that is already encapsulated in a facet template:

- Open the Graphics Builder of the Supervision Participant to modify the Genie.
- Replace the Genie element by another one of the same, another existing, or an external project file (Include project (.ctz)).

To create or modify a Genie that is not yet encapsulated, refer to the topic describing how to use, create, and modify animated graphics, page 273.

NOTE: The software deploys project files that you add to the content repository along with the Supervision project containing instances of the Supervision genie facet template.

Modifying Genies Already Added to Supervision Pages

When you modify a Genie that has already been added to a Supervision page, the modifications appear on the page as soon as you save changes to the project file that contains this Genie. The status of the corresponding *Genie* facet however, remains **Generated**. The status of the executable changes to **Out Of Date** if it was **Built**.

The same applies to composite genies if you modify one of their individual genies.

Adding Project Files

If you want to replace the Genie by one contained in an external project file, you need to add it to the content repository first by using the **Select Genie** window, page 279.

You can only use a project file that was created with a version of Plant SCADA that is the same as the Supervision Participant.

Once you have initiated the operation to add a project file, you cannot cancel it.

You cannot store several versions of the same project file in the content repository.

Replacing a project file in the content repository can affect the functionality of templates and their instances.

NOTICE

LOSS OF DATA

Resolve a project file conflict that arises when adding a project file to the content repository by selecting the correct file version.


Failure to follow these instructions can result in unusable Supervision projects.

To add a project file to the content repository, proceed as follows.

| Step | Action |
|------|---|
| 1 | In the Select Genie window, page 278, click the button to add a project file. Result: The Add Project dialog box opens. |
| 2 | Select a project file (.ctz) and click Open . Result: If a project file with the same name does not exist in the content repository, it is added and appears in the Select Genie window. Otherwise, the Replace Project dialog box opens; proceed to step 3. |
| 3 | Verify the information that is displayed: <ul style="list-style-type: none"> The project file name. Current version: Version of the file that exists in the content repository. New version: Version of the file that you have selected. |
| 4 | Click Yes to overwrite the existing project file with the selected project file or No to cancel the operation. NOTE: Closing the Select Genie window without saving does not remove project files that you have added to the content repository. |

Modifying a Genie


To modify a Genie that is encapsulated in a *Genie* facet template, proceed as follows.

| Step | Action |
|------|---|
| 1 | Edit the <i>Genie</i> facet template that contains the Genie. Result: The facet template opens in the Facet Editor . |
| 2 | Click the Templatizer button. Result: The Select Genie window opens. The Genie that is encapsulated and the corresponding Include project and library are selected. If the library contains other Genies, they appear as well. |
| 3 | Click the Open Participant button. Result: The Genie opens inside the Graphics Builder in a Supervision Participant window. |
| 4 | Proceed with the modification of the genie and save it by using the Save button of the Graphics Builder. Result: Changes are saved. |
| 5 | Click the Close button of the Supervision Participant window. Result: The Save dialog box opens, which lets you save changes to the Include project in the content repository. |
| 6 | Select a versioning scheme, enter a change description, and click Yes . Result: The Supervision Participant window closes and the Select Genie window opens in the Facet Editor . NOTE: Saving changes to a large Include project may take time. |
| 7 | In the Select Genie window, select the genie that you have modified and click the Save button  . |

| Step | Action |
|------|---|
| | Result: The Replace Genie dialog box opens. |
| 8 | Click Yes . NOTE: Clicking No closes the dialog box and also updates the encapsulated Genie if you have modified the Genie that was encapsulated in this <i>Genie</i> facet template. However, if in step 2 you had selected another Genie, your modifications are discarded. |
| 9 | Click the Close button of the Select Genie window. Result: The view reverts to the Facet Editor and the Genie inside the Genie element has been replaced with the one that you have modified in the Graphics Builder. |
| 10 | Select the Genie and edit its identifier and description in the Properties pane, page 227 if needed. |
| 11 | Save the changes, page 254 to the facet template. |

Replacing a Genie Element

To replace a Genie element that is encapsulated in a *Genie* facet template, proceed as follows.

| Step | Action |
|------|---|
| 1 | Edit the <i>Genie</i> facet template that contains the Genie. Result: The facet template opens in the Facet Editor . |
| 3 | In the Facet Editor , click Templatizer . Result: The Select Genie window opens. The Genie that is encapsulated and the corresponding Include project and library are selected. If the library contains other Genies, they appear as well. |
| 4 | Select another Genie. If the Genie is in another library and/or project, select them in the left-hand pane first. |
| 5 | In the Is Equipment and Is Cluster columns, select one parameters, page 280 for each. |
| 6 | Click the Save button  of the Select Genie window. Result: The Replace Genie dialog box opens. |
| 7 | Click Yes . NOTE: Clicking No closes the dialog box and does not replace the encapsulated Genie element by the one that you have selected. |
| 8 | Click the Close button of the Select Genie window. Result: The view reverts to the Facet Editor and the Genie inside the element has been replaced with the one that you have selected. |
| 9 | Select the Genie and edit its identifier and description in the Properties pane, page 227 if needed. |
| 10 | Save the changes, page 254 to the facet template. |

Configuring Facet Templates

Overview

This topic describes the various configuration aspects of a facet template and its elements. It also provides guidelines.

Guidelines

- Duplicate, page 305 a Schneider Electric template that you want to modify. Save the new template in your own folder structure, page 58.
- Save your work regularly, page 254.
- Enter a detailed description of the changes that you make when saving changes. This allows you to keep track of the version history through the **Changes log pane**, page 238.

Configuring Template Properties

Configure properties of the template in the **Properties** pane.

Referencing Elements

For Supervision data facet templates, which do not encapsulate constituents, you can choose from a selection of **Supervision** elements in the **Toolbox** pane, page 238.

For each function, group the elements in the same facet template. View *\$HValve_CD* (simple) and *\$PIDCTL_CD* (more complex) as example.

Configuring Elements

After you drag an element to the workspace, follow these steps.

| | Description | Comment |
|---|---|---|
| 1 | Enter the element identifier. | You cannot edit the identifier of Control elements. By default, it is the name of the resource instance in the constituent file. For Supervision elements (animated graphics), you can edit the identifier in the Properties pane or by double-clicking it. |
| 2 | Edit the element description. | - |
| 3 | Connect implicit parameters: <ul style="list-style-type: none"> • \$InstanceID • \$Description • \$Area | \$Area is required for genie elements. \$Area is not used for Control facet templates. |
| 4 | Hide unused element parameters. | - |
| 5 | Create platform inputs. | For platform inputs on constituents, use data types supported by Participants (see <i>EcoStruxure Process Expert, Control Participant Services, User Guide</i>) instead of data types supported by the software. For example, to define a duration for a Control resource, use Time instead of Duration. Use transformation functions as needed (Toolbox pane). |
| 6 | Create parameters. | <ul style="list-style-type: none"> • Order of creation: Parameters appear at the parent level in the order they are created. • Create categories. • Configure parameter properties. |

Configuring Interfaces

When you drag an interface from the **Browser** to the workspace, follow these steps.

| | Description | Comment |
|---|---|---|
| 1 | Enter the interface identifier. | - |
| 2 | Select the role that you want to use in the facet template. | Both available roles are displayed. |
| 3 | Make the interface required. | Default status is optional. Optional interfaces are shown with a dotted outline. |
| 4 | Hide unused interface properties. | - |
| 5 | Connect elements and/or properties by using bindings. | Shown in gray. Use transformation functions, page 28 as needed (Toolbox pane). |
| 6 | Define interface rules. | Includes position of the connector and order in which the interface is to be displayed at the parent level. |

NOTE: Interfaces appear at the parent level in the order you create them.

Saving Changes

As long as the facet template is not referenced nor instantiated, you can save, page 254 it with the same identifier and version.

Configuring Composite Templates

What's in This Chapter

Configuring Composite Templates 286

Overview

This chapter describes the various configuration aspects of a composite template and its elements, and provides guidelines.

Configuring Composite Templates

Overview

This topic describes the various configuration aspects of a composite template and its elements. It also provides guidelines.

Guidelines

- Duplicate, page 305 a Schneider Electric template that you want to modify. Save the new template in your own folder structure, page 58.
- Save your work regularly, page 254.
- Use the position of Control elements referenced by a composite template to determine the execution order of these elements.
- Enter a detailed description of the changes that you make when saving changes. This allows you to keep track of the version history through the **Changes log pane**, page 238.

Configuring Template Properties

Configure properties of the template in the **Properties** pane.

Referencing Elements

In Control composite templates, you can reference, page 41:

- Control facet templates.
- Control composite templates.
- Interfaces

In Supervision composite templates, you can reference, page 41:

- For data composite templates (**_CD**): Supervision data facet templates.
- For genie composite templates (**_CG**): Supervision genie facet templates.
- For highest level composite templates (**_CS**): Supervision data and genie composite templates.
- Interfaces.

Select elements from the **Browser** pane.

Control Element Position

The position of Control elements inside a composite template determine their execution order.

The rules are the same as those used by the Control Participant.

NOTE: Use the graphical tools of the **Composite Editor** located at the bottom right-hand corner of the workspace.

Configuring Composite and Facet Elements

After you drag a composite or facet template to the workspace, follow these steps:

| | Description | Comment |
|----|---|---|
| 1 | Enter the element identifier. | - |
| 2 | Edit the element description. | - |
| 3 | Connect implicit parameters: <ul style="list-style-type: none"> • \$InstanceID • \$Description • \$Area | \$Area is not used for Control and Supervision composite templates. |
| 4 | Hide unused element parameters and interfaces. | - |
| 5 | Create platform inputs. | Use transformation functions as needed (Toolbox pane). |
| 6 | Create parameters. | <ul style="list-style-type: none"> • Order of creation: Parameters appear at the parent level in the order they are created. • Create categories. • Configure parameter properties. |
| 7 | Create bindings between parameters and interfaces of various elements. | Shown in gray. Use transformation functions as needed (Toolbox pane). |
| 8 | Expand or collapse interfaces, page 231. | <ul style="list-style-type: none"> • Collapsed state: Makes a connector available that represents the complete interface. The connection is made by using an interface link. • Expanded state: Makes each element of a multi-element interface available individually for connection to other elements by using bindings. |
| 9 | Create interface links between elements (facet and composite references) or links with interfaces that you create as elements. | Shown in orange. |
| 10 | Defer unbound parameters and interfaces. | Shown as gray and orange diamonds respectively. NOTE: You can enter an alias, page 170. |
| 11 | Configure the \$Selection property for optional elements. | Making an element optional is a two-step process: <ol style="list-style-type: none"> 1. Right-click the element header and select Switch to Optional. 2. Right-click the \$Selection property and select Create Deferred. The possibility to select and optional element is shown by a red diamond. Optional elements are shown with a dotted outline. |
| 12 | Define element and interface rules. | Includes position, page 164 of the interface connector and order, page 164 in which the interface is displayed at the parent level. You can also change the order in which facet and composite elements are displayed at the parent composite level by dragging their label in the Element Rules pane to the desired position. The left-to-right order in the Element Rules pane corresponds to a top-to-bottom order at the parent composite level. |

Configuring Interfaces

When you drag an interface from the **Browser** to the workspace, follow these steps:

| | Description | Comment |
|---|---|--|
| 1 | Enter the interface identifier. | - |
| 2 | Select the role that you want to use in the facet template. | Both available roles are displayed. |
| 3 | Make the interface required. | Default status is optional. Optional interfaces are shown with a dotted outline. |
| 4 | Hide unused interface properties. | - |
| 5 | Connect elements and/or properties by using bindings. | Shown in gray. Use transformation functions, page 28 as needed (Toolbox pane). |
| 6 | Define interface rules. | Includes position, page 164 of the connector and order, page 164 in which the interface is displayed at the parent level. By default, interfaces appear at the parent level in the order you create them. |

Saving Changes

As long as the composite template is not referenced nor instantiated, you can save, page 254 it with the same identifier and version.

Configuring Control Module Templates

What's in This Chapter

Configuring Control Module Templates..... 289

Overview

This chapter describes the various configuration aspects of a control module template and its elements, and provides guidelines.

Configuring Control Module Templates

Overview

This topic describes the various configuration aspects of a control module template and its elements. It also provides guidelines.

Guidelines

- Duplicate, page 305 a Schneider Electric template that you want to modify. Save the new template to your own folder structure, page 58.
- Save your work regularly, page 254.
- Enter a detailed description of the changes that you make when saving changes. This allows you to keep track of the version history through the **Changes log pane**, page 238.

Configuring Template Properties

Configure properties of the template in the **Properties** pane.

Referencing Elements

In control module template, you can reference, page 40:

- One Control composite template.
- One Supervision composite template (_CS).

Select both from the **Browser** pane.

Configuring Composite Elements

After you drag the composite templates to the workspace, follow these steps.

| | Description | Comment |
|---|-------------------------------|---|
| 1 | Enter the element identifier. | Use the Participant name as identifier, for example, Control. |
| 2 | Edit the element description. | - |

| | Description | Comment |
|---|---|--|
| 3 | Connect implicit parameters: <ul style="list-style-type: none"> • \$InstanceID • \$Description • \$Area | \$Area is not used for Control and Supervision composite templates. |
| 4 | Hide unused element parameters and interfaces. | - |
| 5 | Create bindings between parameters of the two elements. | Shown in gray. For example, to link the <i>\$InstanceID</i> and <i>\$Description</i> parameters. |
| 6 | Create interface links between the composite elements. | Shown in orange. For example, to link the exposed interface of the Control element to the exposed interface of the Supervision element to propagate tag data. |
| 7 | Defer unbound parameters and interfaces. | Shown as gray and orange diamonds respectively. NOTE: You can enter an alias, page 170. |
| 8 | Configure the <i>\$Selection</i> property for the Supervision element. | Making an element optional is a two-step process: <ol style="list-style-type: none"> 1. Right-click the element header and select Switch to Optional. 2. Right-click the \$Selection property and select Create Deferred. The possibility to select and optional element is shown by a red diamond. Optional elements are shown with a dotted outline. |

Saving Changes

As long as the control module template is not instantiated, you can [save](#), page 254 it with the same identifier and version.

Updating and Replacing Templates

What’s in This Part

Updating and Replacing Templates..... 292

Updating and Replacing Templates

What's in This Chapter

| | |
|---|-----|
| Template Modification Strategy | 292 |
| Updating Global Templates and Templates of Elements | 294 |
| Updating Global Templates at the Folder Level | 297 |
| Replacing the Template of an Element of a Global Template | 300 |
| Duplicating Global Templates | 305 |
| Replace Dialog Box | 308 |
| Editing and Extending Interfaces In-Place | 308 |
| Updating References in Parent After Editing Child Element – Example | 312 |

Overview

This chapter explains the aspect of updating templates of elements with their latest version or replacing them with a different template.

Template Modification Strategy

Overview

Various commands are available that allow you to modify Schneider Electric and/or user-created templates. Select the appropriate command depending on:

- Whether you want to edit a template or not.
- Whether the identifier of the template or one of its child templates starts with the \$ prefix, page 80.
- The scope of changes to be performed.

Starting with EcoStruxure Process Expert 2021, when you edit a template from within its parent, you have the possibility to [update its references](#), page 293 in the parent template after you save changes and close.

The table shows the commands that are available to modify Global Templates and their specifics.

| Command | Use in | Use with | Purpose | Comment |
|---------|---------------------------|-----------|---|---|
| Update | Global Templates Explorer | Templates | To update a template with the latest version of child templates that exist in the Global Templates library. | <ul style="list-style-type: none"> • Cannot be used to update templates whose identifier starts with the \$ prefix. |
| | | Folders | To update the templates inside a folder, page 297, and optionally inside its subfolders, with the latest version of child templates that exist in the Global Templates library. | <ul style="list-style-type: none"> • The software finds the latest versions of templates with the same identifier automatically. • When used at the folder level, you can select which templates to update. |
| | Template editors | Elements | To update the template that is used by a specific element with the latest version that exists in the Global Templates library. | <ul style="list-style-type: none"> • Lets you updated the template that is used by a specific element (instance). • The software finds the latest version automatically. |
| Replace | Template editors | Elements | To update the template that is used by a specific element with a different version or a different template that exists in the Global Templates library. | <ul style="list-style-type: none"> • Lets you replace the template that is used by a specific element (instance). • Can be used to undo a template update. |

| Command | Use in | Use with | Purpose | Comment |
|------------------------------|--------------------------------------|----------------------|---|---|
| Duplicate | Global Templates Explorer | Templates | To create a new template from an existing one with the possibility to keep, duplicate, or replace each child template that is referenced. | <ul style="list-style-type: none"> The replace action impacts the elements (instances) of the template. Can be used in place of the Update command with templates whose identifier starts with the \$ prefix. |
| Edit/Extend Interface | Facet and composite template editors | Interface references | To edit, add, and remove elementary elements in an interface, page 308 from within its parent template. | <ul style="list-style-type: none"> Provides basic modification capability. Updating templates that use the other role of the interface is required. The command is not available for deferred interfaces and when the interface element contains a nested interface. |

Updating References in Parent After Editing Child Templates or Interfaces

When you edit a template or interface by opening it from within its parent, after saving changes and closing it, page 254, the **Update/Replace References** dialog box opens. It lets you automatically update in this parent template the references of the template/interface that you have edited. This requires that the parent template is open in a template editor in edit mode. If it is open in read-only mode, you can switch it to edit mode.

If you rename the template while saving changes, references are replaced instead of being updated.

Once you save changes and close the parent, you are again given the choice to update the next higher level template given it is open, and so on. Refer to the example, page 312 for details.

The impact of the template update/replacement is the same as when you use the **Update** or **Replace** command.

NOTE: Other templates that also reference the template that you have edited are not updated.

NOTE: The **Update/Replace References** dialog box opens also when you edit/extend an interface in-place, page 308 or edit a nested interface.

Impact of Template Update or Replacement

The table describes the impact on the various template components when you update or replace a template.

| Current template component | In new template | Impact |
|---|--|---|
| Bindings to the element selection of composite references | Are the same as in current template. | The software retains the bindings. |
| | Are not the same as in current template. | The software removes the bindings to the element selection of the composites that are not part of the new template. |
| Bindings and attributes of the element parameters | Are the same as in current template. | The software retains the bindings and attributes. |
| | Are not the same as in current template. | The software removes the bindings and attributes of the element parameters that are not part of the new template. |
| Bindings to element interfaces | Are the same as in current template. | The software retains the bindings. |
| | Are not the same as in current template. | The software removes the bindings to the element interfaces that are not part of the new template. |

Applying Template Modifications to Existing Instances

Modifications to templates need to be propagated to instances of the application to take effect. The method to apply modifications to existing instances depends on the template identifier.

| Result of template modification | Method to apply |
|---|---|
| A new version of the template (same identifier) | Update the template used by instances (see <i>EcoStruxure Process Expert, User Guide</i>) |
| A template with a new identifier | Replace the template used by instances (see <i>EcoStruxure Process Expert, User Guide</i>) |

Updating Global Templates and Templates of Elements

Overview

The **Update** command lets you perform different actions depending on where you use it.

In the **Global Templates Explorer**, it lets you create a new version of your template, in the same location, by using the latest approved version of child templates that are referenced by this template.

In the **Facet Editor** and the **Composite Editor**, it lets you update the template or interface model that is used by a specific element by the latest available version.

In both cases, the update process allows you to integrate in the composition of your template the latest versions of composite and facet templates, and interface models.

NOTE: The update of the template of an element impacts templates that reference this element and requires [updating](#), page 292 the higher-level templates.

Restrictions When Updating Schneider Electric Templates

[Restrictions](#), page 80 related to the identifier of the parent template apply when you update a Schneider Electric template.

You cannot use the **Update** command in the **Global Templates Explorer** on a template whose identifier contains the \$ prefix. You need to use the **Duplicate** command, page 305 instead.

Guidelines

- If you know which highest level templates reference templates of which a later version is available, proceed with the update from these highest level templates. Typically, the highest level template is the control module template.
- If you do not know which highest level templates need to be updated, use the **Inspect > Used By** context-menu command on the template of which you have created a new version. This allows you to determine which are the highest level templates that you need to update with the new version child template.

Working Principle of the Update Functionality

When the **Update** command is used on a template in the **Global Templates Explorer**, the software proceeds as follows:

- Searches the tree of dependencies of the template for elements with the same identifier and for which a later version exists in the Global Templates library.
- Displays in a dialog box the latest version of templates that it has found in the dependencies and lets you select:
 - The usability state, page 82 for the current version of templates after the new versions are created.
 - The versioning scheme, page 81 and usability state for the new version of templates that are created.
- Once you proceed, creates the new templates, proceeds with the update, and processes bindings, page 244.

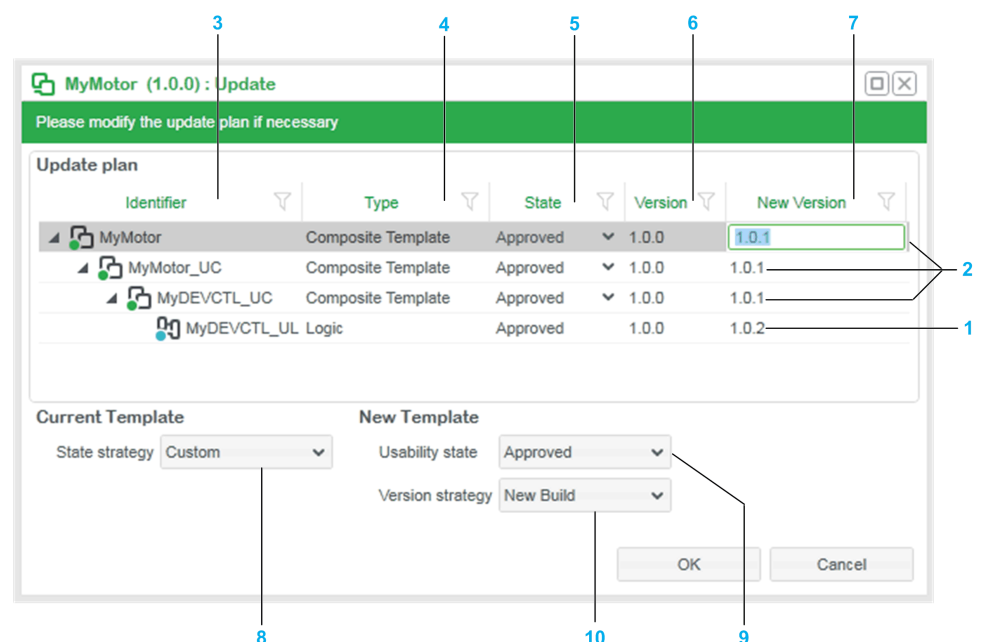
When the **Update** command is used in a template editor, the software proceeds as follows:

- Verifies if a template with a later version exists in the Global Templates library.
- If so, processes bindings and:
 - If a binding cannot be recreated, informs you and gives you the choice to proceed or cancel the update.
 - If bindings can be recreated, proceeds with the update of the template.

NOTE: For the template update process to complete, the latest version of the template or interface model must have the **Usability State Approved** or **Deprecated**.

Update Dialog Box

The following figure shows an example of the **Update** dialog box in a scenario where the **Update** command was used on the *MyMotor (1.0.0)* control module template in the **Global Templates Explorer**. The template references *MyMotor_UC (1.0.0)*. This *UC* composite template references *MyDEVCTL_UC (1.0.0)*, which in turn references the *MyDEVCTL_UL (1.0.0)* facet template. *MyDEVCTL_UL* versions 1.0.1 and 1.0.2 exist in the Global Templates library.



| Item | Description |
|------|---|
| 1 | Template of which the software has found a later version in the Global Templates library compared to the version that is used. |
| 2 | Templates of which the software will automatically create a later version because the version of their child template is being incremented. |
| 3 | Shows the hierarchy of the templates that will be updated and the templates of which a later version exists in the Global Templates library. Other child templates whose version is unchanged are not shown. At the top level, identifier of the template from which you have selected the Update command. |
| 4 | Type of the template. |
| 5 | Usability state, page 82 of the current version of the template after the software has created the new version based on the value of the State strategy parameter. You can select it individually for templates of which the software creates a new version. |
| 6 | Current version of the template from which you have selected the Update command and current version of its child templates. |
| 7 | Version of the template after the update and latest version of child templates with the usability state <i>Approved</i> that the software has found in the Global Templates library. For parent templates, the version is created by the software during the update based on the value of the Version strategy parameter. NOTE: You can modify the version number, page 81 manually for each template version that the software creates during the update. Modifying the version number manually sets the Version strategy parameter to Custom . You cannot enter the same value as in Version . NOTE: If Version for a template that the software will update is already 99.99.9999, New Version is blank and you need to enter a value manually. |
| 8 | Allows you to select the usability state of the current version (Version column) of the templates after the software has created a new version of them. Default value: <i>Custom</i> |
| 9 | Allows you to select the usability state of the new version of the templates after the update. Default value: <i>Approved</i> NOTE: The usability state applies only to new template versions that the software creates. |
| 10 | Allows you to select the versioning scheme, page 81 for new template versions that the software creates. Default value: <i>New Build</i> NOTE: The version strategy applies only to new template versions that the software creates. |

Updating Global Templates in the Global Templates Explorer

Updating Schneider Electric or user-created templates may affect the function of these templates and must be performed by qualified personnel. Before proceeding, refer to the [Overview](#), page 174.

To update Global Templates, proceed as follows.

| Step | Action |
|------|---|
| 1 | In the tree view of the Global Templates Explorer , right-click the template that you want to update and select Update . Result: The Update dialog box opens. |
| 2 | For the current template version and its elements, select the usability state strategy that you want the software to apply from the menu of the State strategy parameter. NOTE: If you select Custom , select the usability state for each template in the State column. |
| 3 | For the new template version and its elements, select the usability state strategy that you want the software to apply from the menu of the Usability State parameter. |

| Step | Action |
|------|--|
| 4 | <p>For the new template version and its elements, select the version strategy that you want the software to apply from the menu of the Version strategy parameter.</p> <p>NOTE: You can modify the version number manually for each template that the software will create by double-clicking the New Version field:</p> <ul style="list-style-type: none"> If the version number that you have entered is invalid, the software applies the default versioning scheme for new templates. To undo your version change, select New Build as the version strategy. This changes the version of new templates to the default value. |
| 5 | <p>Click OK.</p> <p>Result: The software:</p> <ul style="list-style-type: none"> Proceeds with the update. Displays the new templates that it has created in their respective folders within the Global Templates library. Displays Completed in the notification panel when the update process is completed. <p>NOTE: Click Cancel to close the dialog box without proceeding with the update.</p> |

Updating Global Templates in Template Editors

Updating Schneider Electric Global Templates or user-created templates may affect the function of these templates and must be performed by qualified personnel. Before proceeding, refer to the [Overview](#), page 174.

To update the template or interface model that is used by an element of a facet or composite template, proceed as follows.

| Step | Action |
|------|---|
| 1 | <p>In the template editor, right-click the element whose template or interface model you want to update and select Update.</p> <p>Result: If a later version of the same template or interface model with the required usability state exists in the Global Templates library, the software processes binding information; otherwise, the software indicates that the template is already up-to-date.</p> |
| 2 | <p>Result: If a binding cannot be recreated, the Replace Conflicts dialog box opens; otherwise, the update of the template completes and bindings are recreated.</p> |
| 3 | <p>Verify the information that is displayed in the Replace Conflicts dialog box and click:</p> <ul style="list-style-type: none"> Yes: Proceeds with the update of the template, discards conflicting bindings, and recreates the other bindings. No: Closes the Replace Conflicts dialog box without updating the template. |
| 4 | Recreate bindings as needed. |
| 5 | Save the change, page 254. |

NOTE: To undo a template update that you have performed in a template editor, you can use the **Replace** command, page 300.

Updating Global Templates at the Folder Level

Overview

The **Update** command that appears in the context menu of folders of the **Global Template Explorer** lets you create a new version of templates contained in a folder, and optionally, in its subfolders.

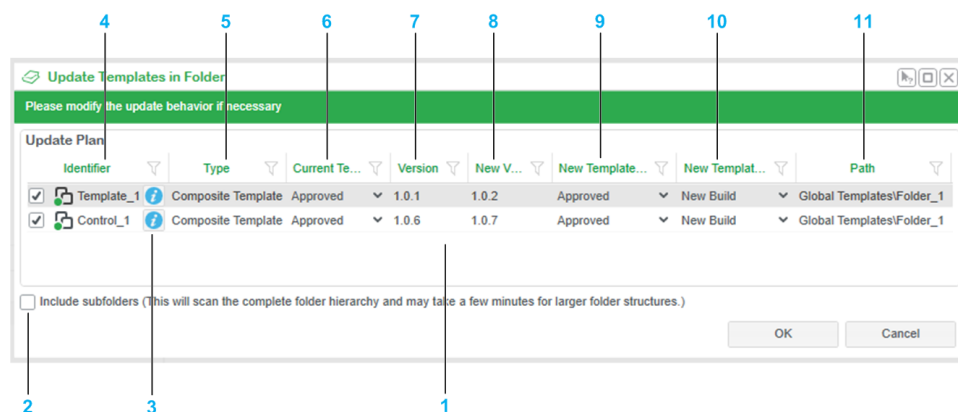
Restrictions When Updating Schneider Electric Templates

Restrictions related to the identifier, page 80 of the parent template and its elements apply when you update a Schneider Electric template.

When you use the **Update** command at the folder level and the folder contains a template whose identifier starts with the \$ prefix, the template is not updated. You need to edit the template and save it with a new identifier or use the **Duplicate** command, page 305.

Update Templates in Folder Window

The following figure shows an example of the **Update Templates in Folder** window.



| Item | Description |
|------|--|
| 1 | <p>List of templates in the folder for which a later version of at least one child template is available in the Global Templates library.</p> <p>NOTE:</p> <p>If a folder contains several versions of the same template and more than one can be updated, only the highest version will be shown and can be updated.</p> <p>This applies also If some of the versions that can be updated are in a subfolder and you have selected to include subfolders; only the highest version among existing folders will be shown.</p> |
| 2 | Lets you include templates located inside subfolders. Their location is indicated in the Path column. |
| 3 | Click the button to view details about which newer version child templates will be used to update the template. |
| 4 | <p>Identifier of the templates that are candidate for the update.</p> <p>The operation is performed only on templates whose checkbox is selected.</p> |
| 5 | Type of the template. |
| 6 | Usability state, page 82 of the current version of the template after the software has created the new version. |
| 7 | Current version of the template. |
| 8 | <p>Version of the template after the update.</p> <p>The version is based on your selection for New Template Version strategy.</p> <p>NOTE: You can modify the version number, page 81 manually. This sets New Template Version strategy to <i>Custom</i>. You cannot enter the same value as it appears in Version.</p> <p>NOTE: If Version of a template to be updated is 99.99.9999, New Version is blank and you need to enter a value manually.</p> |
| 9 | <p>Allows you to select the usability state of the new version of the template after the update.</p> <p>Default value: <i>Approved</i></p> |
| 10 | <p>Allows you to select the versioning scheme for the new template version.</p> <p>Default value: <i>New Build</i></p> |
| 11 | Location of the template in the Global Templates library. |

Updating Templates at the Folder Level

To update the templates contained in a folder, proceed as follows.

| Step | Action |
|------|---|
| 1 | <p>In the Global Template Explorer, right-click a folder and select Update.</p> <p>Result: The Update Templates in Folder window opens and displays the templates for which a child template with a later version exists in the Global Templates library.</p> <p>NOTE: The command is not available from the Global Templates library root folder.</p> |
| 2 | Select to display also templates of subfolders if needed. |
| 3 | Select which templates you want to update. |
| 4 | <p>For each one, select the appropriate usability states and versioning schemes for the current and new template versions and click OK.</p> <p>Result: The selected templates are updated and appear in the tree view of the Global Templates library. A message in the notification panel indicates the status of the operation.</p> |

Replacing the Template of an Element of a Global Template

Overview

In the **Facet Editor** and **Composite Editor**, the **Replace** context-menu command allows you to replace the template that is used by an element with another template of the Global Templates library.

You can replace the template of the following types of elements:

- Composite templates
- Facet templates
- Interfaces

The software informs you if it detects binding-related conflicts before proceeding with the replacement of the template.

The replacement of the template of an element impacts templates that reference this element and requires [updating, page 292](#) the higher-level templates.

Restrictions When Replacing Schneider Electric Templates

[Restrictions, page 80](#) related to the identifier of parent templates apply when you replace a template.

Before Starting

Depending on the situation, consider the following before starting:

- To replace the template of an element by the latest version of the same template, use the **Update** command, [page 294](#) instead.
- You can use the **Replace** command to undo the [update of the template of an element, page 294](#). Undoing a template update does not restore bindings that the software was not able to recreate.

Replace Dialog Box

For a description of the template browser that opens, refer to the topic describing the **Replace** dialog box, page 308.

Replacing The Template That Is Used by an Element

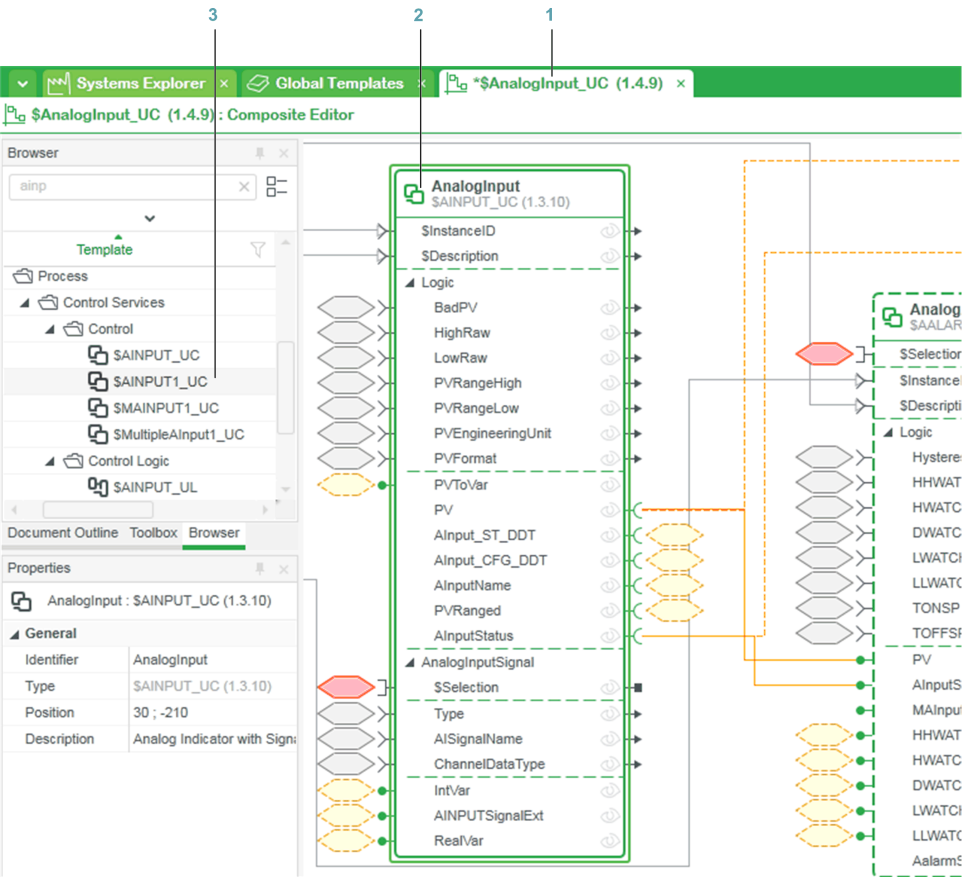
Modifying Schneider Electric Global Templates or user-created templates may affect the function of these templates and must be performed by qualified personnel. Before proceeding, refer to the [Overview](#), page 174.

To replace the template that is used by an element of a Global Template, proceed as follows.

| Step | Action |
|------|---|
| 1 | <p>In a template editor, right-click the element whose template you want to replace and select Replace.</p> <p>Result: The Replace dialog box opens.</p> <p>NOTE: Alternatively, you can select the template in the Browser pane and drag it onto the header of the element until Drop to update the reference with x is displayed in a tooltip (where x represents the name and version of the template that you want the element to use).</p> |
| 2 | <p>Select the template that you want the element to use instead and click OK.</p> <p>Result: The Replace confirmation dialog box opens.</p> |
| 3 | <p>Verify the information that is displayed and click Yes.</p> <p>Result: If a binding cannot be recreated, the Replace Conflicts dialog box opens; otherwise, the replacement of the template completes and bindings are recreated.</p> |
| 4 | <p>Verify the information that is displayed in the Replace Conflicts dialog box and click:</p> <ul style="list-style-type: none">• Yes: Proceeds with the replacement of the template, discards conflicting bindings, and recreates the other bindings.• No: Closes the Replace Conflicts dialog box without replacing the template. |
| 5 | <p>Recreate bindings as needed.</p> |
| 6 | <p>Save changes, page 254.</p> |

Example

The following example illustrates the replacement of the template of a composite element, which is referenced by a parent composite reference. The highest level composite template being *\$AnalogInput* (control module).



| Item | Description |
|------|--|
| 1 | <i>\$AnalogInput_UC</i> [1.4.9], which references element <i>AnalogInput</i> is opened in edit mode in the Composite Editor . |
| 2 | Element <i>AnalogInput</i> uses template <i>\$AINPUT_UC</i> , which is going to be replaced. |
| 3 | Browser pane that allowed locating template <i>\$AINPUT1_UC</i> , which is going to replace <i>\$AINPUT_UC</i> . |

The software displays the list of bindings that cannot be recreated after the template replacement. Write down these bindings so that you can recreate them later on if required.

?

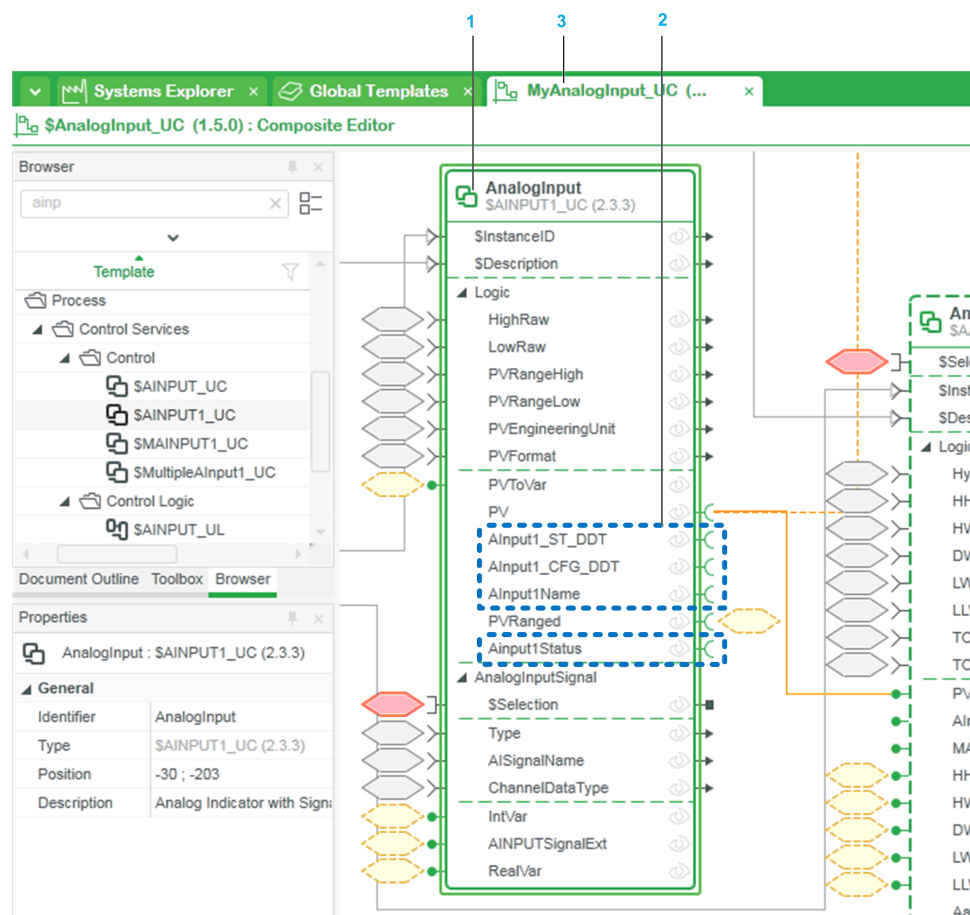
The following conflicted bindings will be deleted.
Are you sure that you want to replace the reference?

| Origin | Destination |
|--|----------------------------------|
| DefParam4bfd2f2f-b202-4dd3-abf8-1d7aad4ea... | AnalogInputLogic\BadPV |
| 96b6c9e4-822b-4135-9599-f47fab3a9a71 | AnalogInputLogic\AIInput_ST_DDT |
| 47f5a6d6-d655-4a8d-b84a-fa54b6c088d4 | AnalogInputLogic\AIInput_CFG_DDT |
| 9453ef46-c22e-4015-89e8-934000bcc308 | AnalogInputLogic\AIInputName |
| AnalogInputLogic\AIInputStatus | AnalogAlarmsLogic\AIInputStatus |
| AnalogInputLogic\AIInputStatus | AnalogInputStatus\AINPUTStatus |

Yes

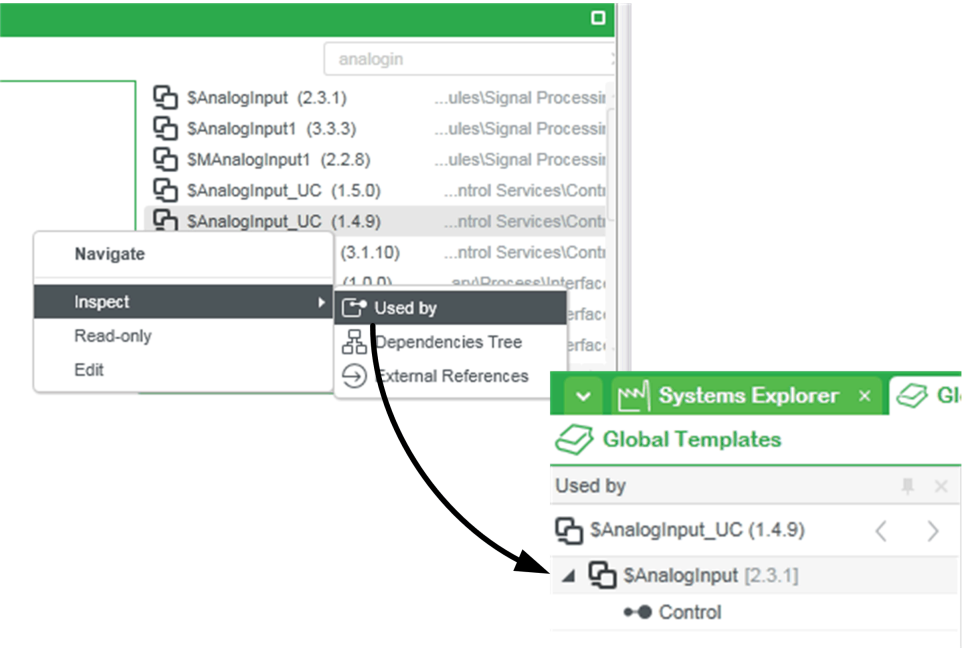
No

The figure shows the *AnalogInput* element after template replacement is complete and the template referencing it has been saved.

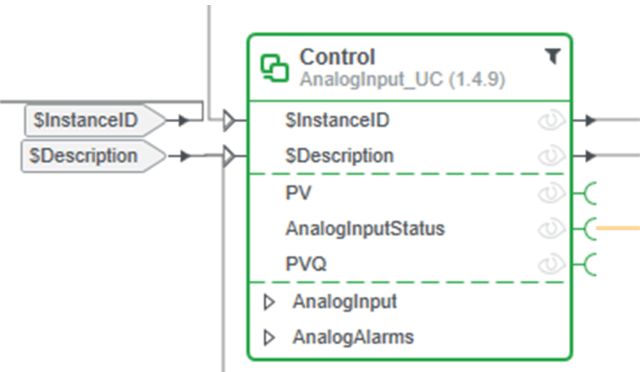


| Item | Description |
|------|--|
| 1 | Element <i>AnalogInput</i> uses template <i>\$SAINPUT1_UC</i> . Its identifier is unchanged as shown in the element header and the Properties pane. |
| 2 | The bindings that were listed in the Replace Conflicts dialog box have not been recreated. |
| 3 | <i>\$AnalogInput_UC</i> [1.4.9] has to be saved with a different identifier, page 80 (for example, <i>MyAnalogInput_UC</i> [1.2.0]) with Usability state approved . |

Once template replacement is complete, you need to update templates that reference *\$AnalogInput_UC* [1.4.9] so that they use *MyAnalogInput_UC* [1.2.0] instead. To identify these templates, use the **Used by** command on *\$AnalogInput_UC* [1.4.9]. In this example, the command returns *\$AnalogInput* [2.3.1], which uses *\$AnalogInput_UC* [1.4.9] for element *Control*.



The following figure shows in a partial view, element *Control* referenced by *\$AnalogInput* [2.3.1].



To update *\$AnalogInput* [2.3.1] with *MyAnalogInput_UC* [1.2.0], navigate to *\$AnalogInput* [2.3.1] and use the **Duplicate** command, page 305. This process creates a copy with a different identifier, which includes *MyAnalogInput_UC* [1.2.0]. In this example, duplicating *\$AnalogInput* [2.3.1] creates *MyAnalogInput* [1.0.0]. (The template that is replaced is highlighted for the example.)

\$AnalogInput (2.3.1) : Duplicate

Please specify the details for the templates to be duplicated

New Identifier
☒ Prefix ☐ Suffix Duplicate All

| Identifier and Version | Type | Action | New Identifier | New Version |
|--------------------------|--------------------|-----------|------------------|-------------|
| \$AnalogInput(2.3.1) | Composite Template | Duplicate | MyAnalogInput | 1.0.0 |
| \$AnalogInput_CS (1.1.7) | Composite Template | None | | |
| \$AnalogInput_CD (1.1.8) | Composite Template | None | | |
| \$AALARM_CD (2.2.1) | Data | None | | |
| \$AINPUT_CD (2.1.2) | Data | None | | |
| \$AnalogInput_CG (1.0.4) | Composite Template | None | | |
| \$AIIPVA_CG (1.0.3) | Genie | None | | |
| \$AIIPVSPA_CG (1.0.2) | Genie | None | | |
| \$AIPVA_CG (1.0.2) | Genie | None | | |
| \$AIPVSPA_CG (1.0.2) | Genie | None | | |
| \$AnalogInput_UC (1.4.9) | Composite Template | Replace | MyAnalogInput_UC | 1.2.0 |
| \$AALARM_UC (2.2.5) | Composite Template | None | | |
| \$AALARM_UL (2.2.1) | Logic | None | | |
| \$AINPUT_UC (2.0.10) | Composite Template | None | | |

☐ Display interfaces

Description of the change
 Template duplicated from the template AnalogInput (2.3.1)

Usability State
 Approved

OK Cancel

NOTE: You may need to update application instances that use *\$AnalogInput* [2.3.1] and/or *\$AnalogInput_UC* [1.4.9] to use the new templates *MyAnalogInput* [1.0.0] and/or *MyAnalogInput_UC* [1.2.0] respectively. Use the **Replace** command (see *EcoStruxure Process Expert, User Guide*) to perform the update.

Duplicating Global Templates

Overview

The **Duplicate** command that is available in the context-menu of templates in the **Global Templates Explorer**, lets you create a copy or a new version of a template with the original and/or modified child templates.

For each child template, you can select to keep the original, duplicate (rename or change the version), or replace it by another existing template.

The software informs you if it detects binding-related conflicts because of a template replacement before proceeding with the duplication of the template.

NOTE: This command can be used instead of the **Update** command for templates whose identifier contains the \$ prefix. However, you need to know for which child templates a new version already exists and select it manually.

Restrictions When Duplicating Schneider Electric Templates

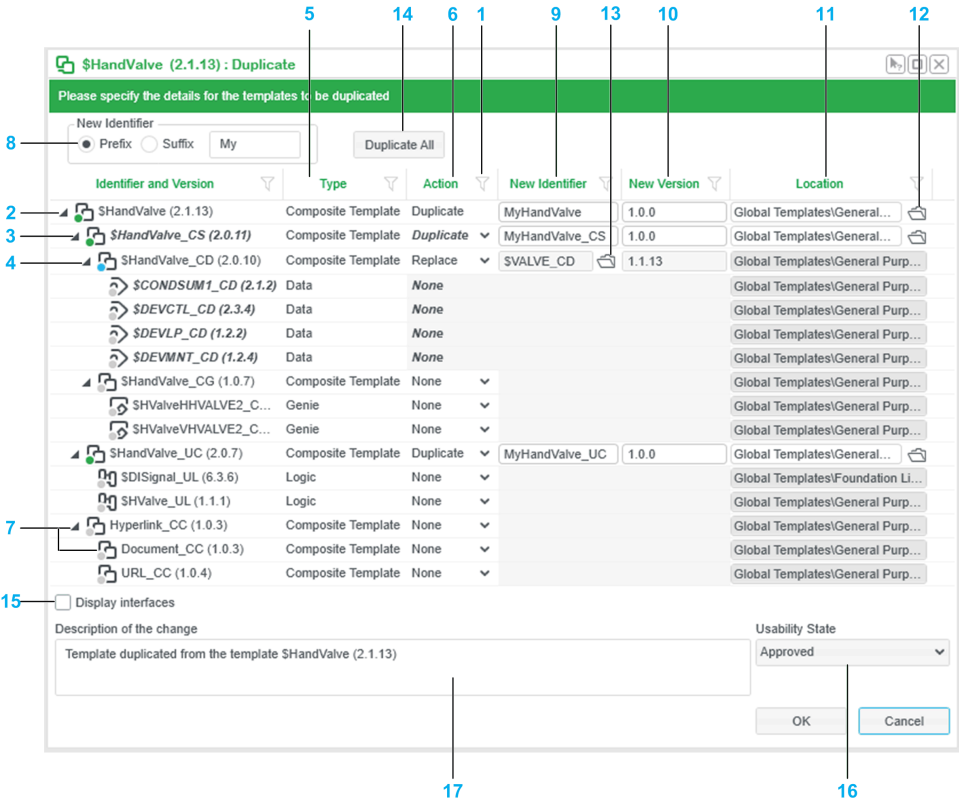
Restrictions, page 80 related to the template identifiers apply when you duplicate a Schneider Electric template .

Guidelines

- Use the **Duplicate** command as a one-step solution to create a copy of a Schneider Electric template whose child templates you can edit and save without using the \$ prefix, page 80.
- Avoid duplicating templates of the Foundation library and interface models because these are low-level resources that are common to Schneider Electric templates to make them compatible. You can identify foundation templates by their location.

Duplicate Window

The following figure shows an example of the **Duplicate** window for a *\$HandValve* sample template in which duplication and replacement actions for child templates have been selected.



| Item | Description |
|------|--|
| 1 | You can sort and filter (see <i>EcoStruxure Process Expert, User Guide</i>) data. |
| 2 | Identifier and version of the template from which you have selected the Duplicate command. |
| 3 | Identifier and version of child templates that the template references. At each level of the template composition, each template is shown only once even if several instances of it exist. By default, interface models are not shown. |
| 4 | When you replace a template, the child templates of the replacement are shown under the template to be replaced in read-only mode. |

| Item | Description |
|------|--|
| 5 | Type (see <i>EcoStruxure Process Expert, User Guide</i>) of the template. |
| 6 | <p>Action to be performed for each referenced child template. Each type of action is identified by a colored dot:</p> <ul style="list-style-type: none"> • Duplicate (green): Creates a new template. If the template references child templates, they are reused without change unless you duplicate them as well. • None (gray): No action is performed. The template is reused in the duplicated parent template. <p>For replacement templates and their children, it is the only setting that is allowed.</p> <ul style="list-style-type: none"> • Replace (blue): Opens the Replace dialog box, page 308, which lets you select an existing template in the Global Templates library to be used in place of the current template and its child templates. The replacement template is used for existing instances of the current template. <p>The template must have the usability state, page 82 Approved.</p> |
| 7 | <p>Duplicating or replacing a template automatically sets the action of the parent templates to Duplicate.</p> <p>Entries in italic indicate a change that is the result of an action that you have selected.</p> |
| 8 | Free form text fields to enter either a prefix or suffix that is added to the identifier of each duplicated template. |
| 9 | Identifier of the template that is proposed by the software for duplicated templates (editable) or identifier of the replacement template that you have selected. |
| 10 | Version that is proposed by the software for duplicated templates (editable) or version of the replacement template. |
| 11 | <p>Location in the Global Templates library of:</p> <ul style="list-style-type: none"> • An original template (None action). The location is read-only. • The duplicated template. You can select a different location and/or add additional ones. <p>The location that you select must exist. When you select more than one location, the software creates <i>linked copies</i>, page 256 of the template.</p> <p>In the location selection dialog box, a check box lets you set the same location for the duplicated templates.</p> <ul style="list-style-type: none"> • The replacement template. The location is read-only. • Child templates of a replacement template (None action). The location is read-only |
| 12 | Button to modify the location of duplicated templates within the Global Templates library. |
| 13 | Button to open the Replace dialog box, page 308 to select a different template. |
| 14 | <p>Sets the action of the child templates to Duplicate. Applies to interfaces only if Display interfaces was selected when the Duplicate All button was clicked.</p> <p>This action overrides changes that you have made for child templates for which you have selected the Replace action.</p> <p>NOTE: You cannot revert this action for the child templates at once unless you cancel and discard changes.</p> |
| 15 | <p>When selected, shows the interfaces that are referenced by child templates. You can select an action for each interface. The changes that you have made to interfaces are retained when you clear the check box.</p> <p>When you select the Duplicate command for an interface, select it to show nested interfaces.</p> |
| 16 | Lets you select the usability state, page 82 that is set for the duplicated templates. |
| 17 | Free-form text field to enter a description of the changes. |

Duplicating Global Templates

Duplicating Schneider Electric Global Templates or user-created templates may affect the function of these templates and must be performed by qualified personnel. Before proceeding, refer to the *Overview*, page 174.

To duplicate a global template, proceed as follows.

| Step | Action |
|------|--|
| 1 | In the tree view of the Global Templates Explorer , right-click the template that you want to duplicate and select Duplicate . Result: The Duplicate window opens. |
| 2 | Select an action for referenced child templates as needed. |
| 3 | Configure the parameters for duplicated templates as needed. |
| 4 | Select a usability state for duplicated templates. |
| 5 | Click OK . Result: If the software detects no binding-related conflicts, it duplicates the template based on your selections; otherwise, the Replace Conflicts dialog box opens. |
| 6 | Verify the information that is displayed in the Replace Conflicts dialog box and click: <ul style="list-style-type: none"> • Yes: Proceeds with the duplication of the template and discards conflicting bindings. • No: Closes the Replace Conflicts dialog box and reverts to the Duplicate window. |
| 7 | Recreate bindings as needed. |

Replace Dialog Box

Overview

The **Replace** dialog box opens when you select the corresponding command in the **Duplicate** window, page 305 or in the context menu of an element in a template editor, page 300.

It lets you browse the Global Templates library to select a replacement template.

Description

The browser of the **Replace** dialog box behaves like the templates browser (see *EcoStruxure Process Expert, User Guide*) of the **Global Templates Explorer** with the following exceptions:

- The scope of the search is context-sensitive.
- Facet templates are shown by default.
- Only templates with the usability state **Approved** are shown.

Editing and Extending Interfaces In-Place

Overview

The **Edit/Extend Interface** command lets you edit, add, and remove elementary elements of an interface that is referenced in a template without using the **Interface Editor**.

You can add elements in direct mode only.

The operation creates a new version of the interface and updates the reference. It also allows you to update other references of this interface (independently of their role) that exist in this template.

The command is not available for deferred interfaces and when the interface element contains a nested interface.

NOTE: The update of an interface reference impacts the other role of this interface and requires updating, page 292 the templates that reference the previous version of the other role. Refer to the example, which describes the workflow to edit/extend an interface, page 310.

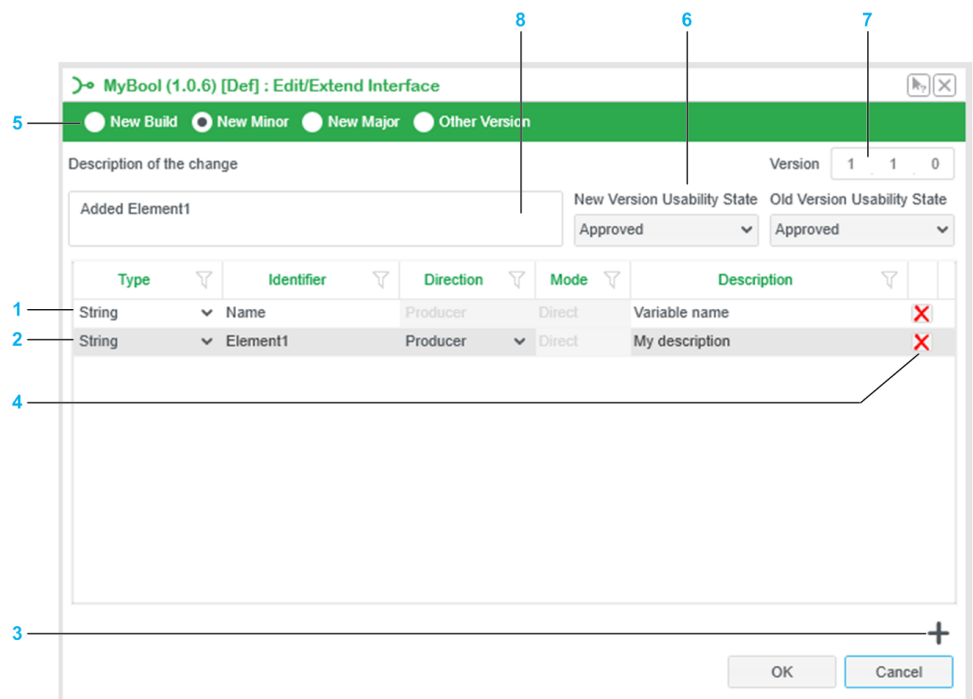
Restrictions When Updating Schneider Electric Templates

Restrictions, page 80 related to the identifier of the parent template and its elements apply when you update a Schneider Electric template.

You cannot use the **Edit/Extend Interface** command on an interface whose identifier starts with the \$ prefix. You need to edit the interface from within the template and save it with a new identifier or use the **Duplicate** command, page 305.

Edit/Extend Interface Window

The following figure shows an example of the **Edit/Extend Interface** window.



| Item | Description |
|------|---|
| 1 | Existing element of the interface. Once an element is added and the interface saved, you cannot change its Direction and Mode properties. NOTE: Modifying Type does not discard existing bindings. However, modifying Identifier does. |
| 2 | Element that is being added to the interface by using the Edit/Extend Interface command. The Mode interface property, page 62 is read-only; only direct data propagation is supported when you use the command. |
| 3 | Button to add a new element. |
| 4 | Button to remove the element. NOTE: If you remove a bound element and recreate it with identical identifier and configuration, the binding with this element is retained when you update the interface. |

| Item | Description |
|------|---|
| 5 | Lets you select the versioning scheme that you want to use for the new version of the interface. Select Other Version as version scheme to edit the value. |
| 6 | Lets you select the following: <ul style="list-style-type: none"> Usability state of the new version of the interface that you are creating. NOTE: Only <i>Approved</i> or <i>Deprecated</i> allow using the new version of the interface in the template. Usability state of the interface that you are editing/extending after you create the new version of interface. NOTE: For more information see <i>Usability State</i>, page 82. |
| 7 | Version of the interface that you are creating. |
| 8 | Mandatory change description. The description is visible in the changes log of the interface. |

Editing/Extending Interfaces

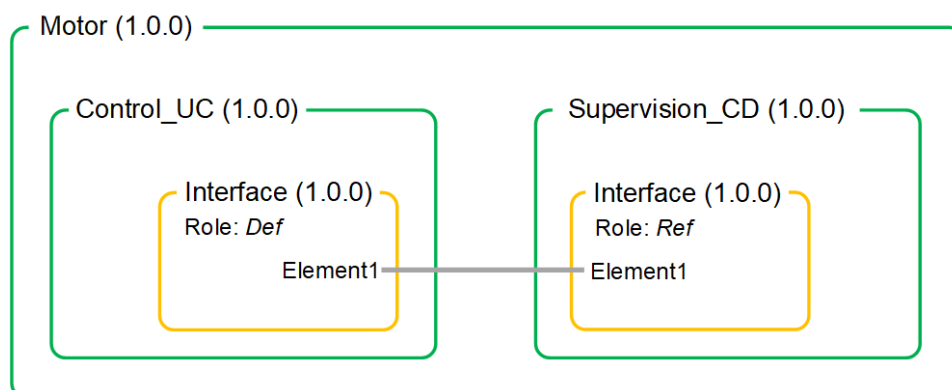
To edit/extend an interface from within its parent template, proceed as follows.

| Step | Action |
|------|--|
| 1 | In the Facet Editor or Composite Editor , right-click the header of an interface element and select Edit/Extend Interface . |
| 2 | In the Edit/Extend Interface window, perform the required changes, enter a change description, and click OK . Result: The window closes and a new version of the interface is created. NOTE: If New Version Usability State is <i>Not Approved</i> or <i>Obsolete</i> , a new version of the interface is created but the interface role element is not replaced in the template. |
| 3 | If other references (independently of their role) of this interface exist in this template, the Update References dialog box, page 293 opens. Click either button: <ul style="list-style-type: none"> Yes: Updates the references of the interface in the template if existing bindings can be recreated. No: Only updates the interface that you have edited if existing bindings can be recreated. |
| 4 | If, in the new interface version, at least one element that was bound is removed, a Replace Conflicts dialog box opens in sequence for each reference that is updated. In each dialog box, verify the information that is displayed about bindings that cannot be recreated and click either button: <ul style="list-style-type: none"> Yes: Replaces the reference without recreating bindings with removed elements. No: Does not replace the reference. |
| 5 | Save changes in the template that you are editing. |
| 6 | Update the other templates that reference other role of each interface reference. You can identify these templates by using the Inspect > Used By context menu command on the previous version of the interface in the Global Templates Explorer or by using the Update command at the folder level. |

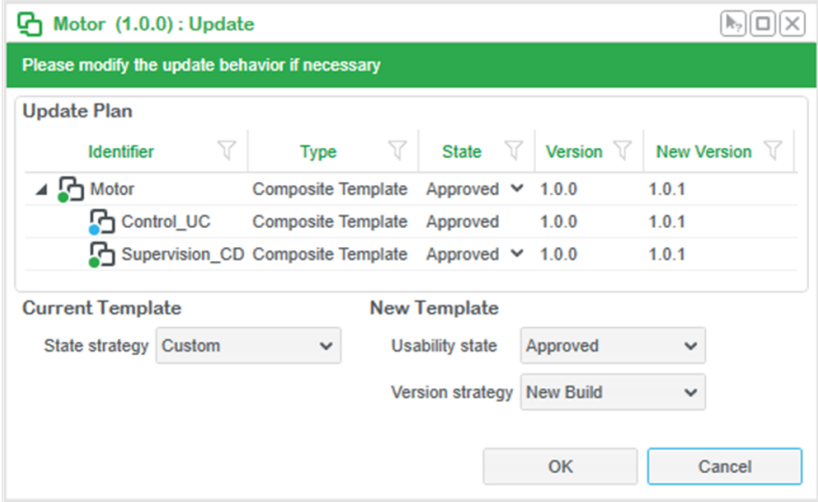
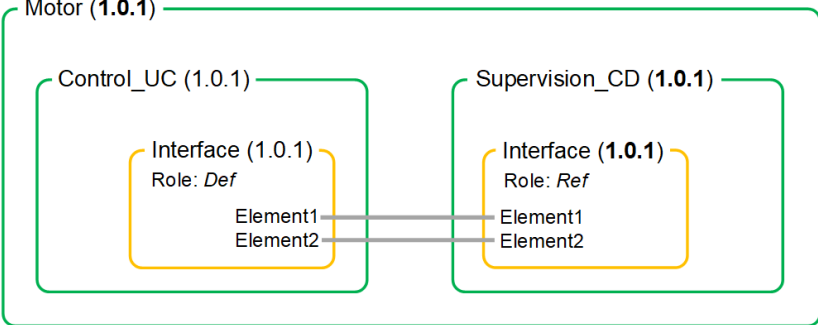
Workflow to Edit/Extend an Interface in a Control Module Template

The following table describes the workflow to propagate changes that were made to an interface by using the **Edit/Extend Interface** command to the entire composition of the template referencing this interface. A simple template is used as example.

The control module template *Motor (1.0.0)* contains two child templates. Each of them references a role of *Interface (1.0.0)*. The interface consists of one element to propagate data.



| Step | Action |
|------|--|
| 1 | <p>From the Global Templates Explorer, edit the template that contains either role of the interface.</p> <p>(<i>Control_UC</i> is used in this example. However, you can also edit <i>Supervision_CD</i> instead. The workflow is similar.)</p> |
| 2 | <p>Select the Edit/Extend Interface command on the interface and modify it.</p> <p>(<i>Interface</i> and adding <i>Element2</i> are used in this example.)</p> |
| 3 | <p>Save changes to this role of the interface with a new version and usability state <i>Approved</i>.</p> <p>(<i>Interface (1.0.1)</i>, and role <i>Def</i> are used in this example.)</p> <p>NOTE: Do not rename the interface because this prevents the propagation of the changes throughout the template. If the identifier of the interface starts with the \$ prefix, first use the Duplicate command, page 305 to remove the prefix.</p> <p>NOTE: At this point, data is not propagated anymore to other templates referencing the original version of other role of the interface (in this example, no more data propagation to <i>Supervision_CD (1.0.0)</i> referencing role <i>Ref</i> of <i>Interface (1.0.0)</i>).</p> |
| 4 | <p>Save the template (Save As) that contains the modified interface role with a new version, usability state <i>Approved</i>, and close it.</p> <p>(<i>Control_UC (1.0.1)</i> is used in this example.)</p> |

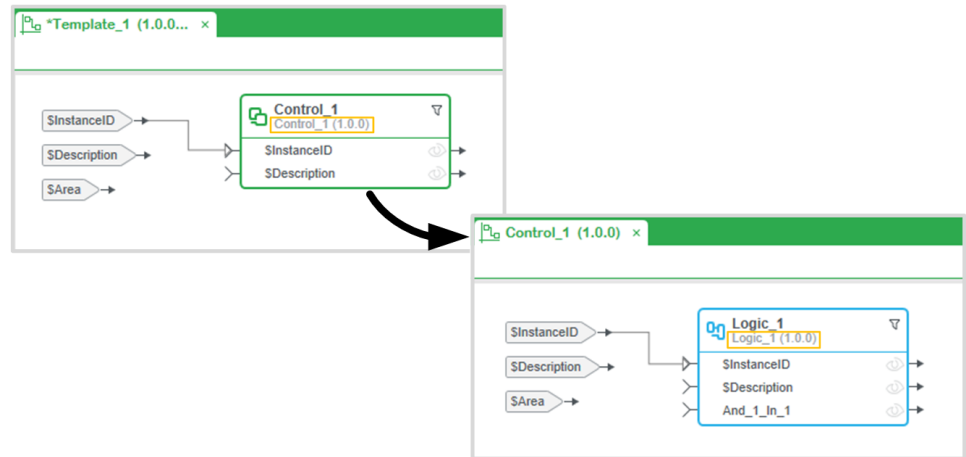
| Step | Action |
|------|--|
| 5 | <p>In the Global Templates Explorer, right-click the highest level template that contains the interface and its parent and select Update.</p> <p>(<i>Motor (1.0.0)</i>, which contains <i>Interface</i> and its parent <i>Control_UC (1.0.0)</i> is used in this example. However, in your template, it could be several levels higher.)</p> <p>Result: The Update window opens and lists, among other templates, the child templates that contain a reference of either role of the original interface version, which will be updated to the new version containing your changes.</p> <div></div> <p>NOTE: The interface that you have modified does not appear.</p> |
| 6 | <p>In this example, clicking OK creates <i>Motor (1.0.1)</i>, which is updated with the following:</p> <ul style="list-style-type: none">Existing <i>Control_UC (1.0.1)</i> (which already contains <i>Interface (1.0.1)</i>, <i>Def</i>)<i>Supervision_CD (1.0.1)</i> created by the software (in which <i>Interface (1.0.0)</i>, <i>Ref</i> is updated to <i>Interface (1.0.1)</i>, <i>Ref</i>) <p>As a result, the binding between <i>Element1</i> is automatically re-established and a new binding between <i>Element2</i> is created.</p> <div></div> |

Updating References in Parent After Editing Child Element – Example

This topic describes the update of references inside the parent template by using a simple three-level template as an example. It shows how to update the parent after the lowest-level template has been modified.

Starting Point – Parent Template

To illustrate the automatic update process, page 293, this example uses parent *Template_1* version 1.0.0, which references composite *Control_1* version 1.0.0, which references facet *Logic_1* version 1.0.0.



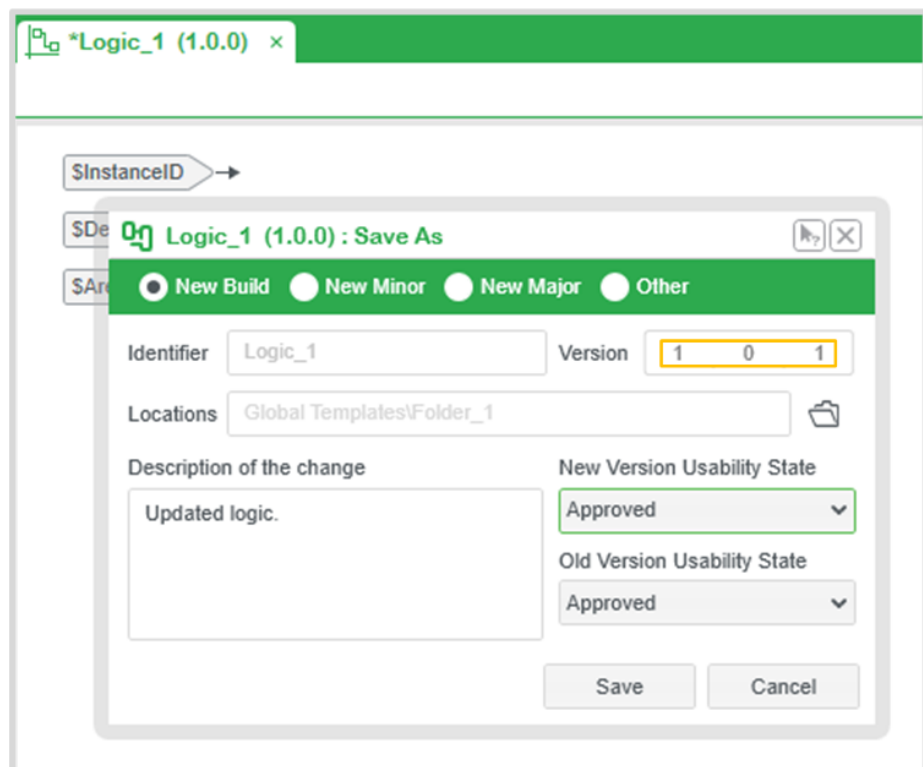
NOTE: Template identifiers and versions are highlighted with an orange outline for illustration purposes only.

Modifying the Lowest-Level Template and Saving Changes

Template_1 is edited.

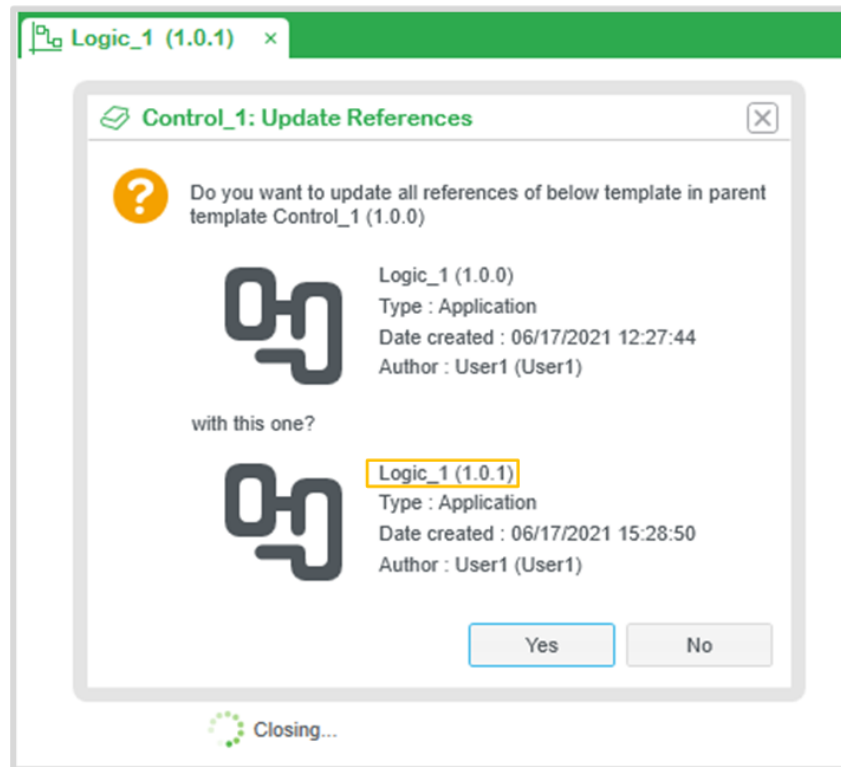
From within *Template_1*, *Control_1* is edited (by using the context menu command). The templates must remain open.

Then, from within *Control_1*, *Logic_1* is edited, modified, and saved with a new version (1.0.1) and usability state *Approved*.



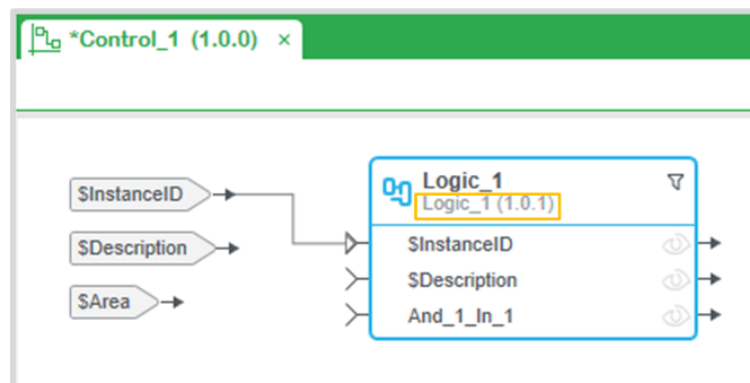
Closing the Lowest-Level Template and Updating the Reference in Parent (Mid-Level Template)

After closing *Logic_1*, the **Update References** dialog box opens, which lets you update reference *Logic_1* v1.0.0 with version 1.0.1 inside *Control_1*.



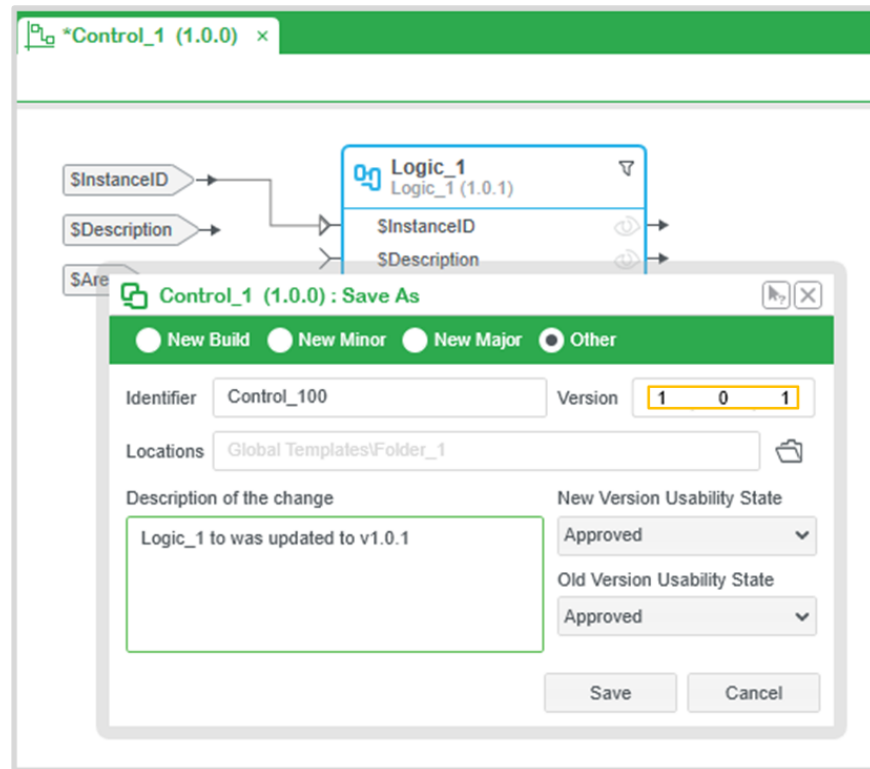
NOTE: If *Control_1* contains several references of *Logic_1* v1.0.0, clicking **Yes** replaces them. Bindings are managed by the software and a **Replace Conflicts** dialog box opens for each reference for which at least one binding cannot be recreated. You can select not to proceed with the update.

The following figure shows the result in *Control_1* after clicking **Yes** in the dialog box.



Saving Changes in Mid-Level Template

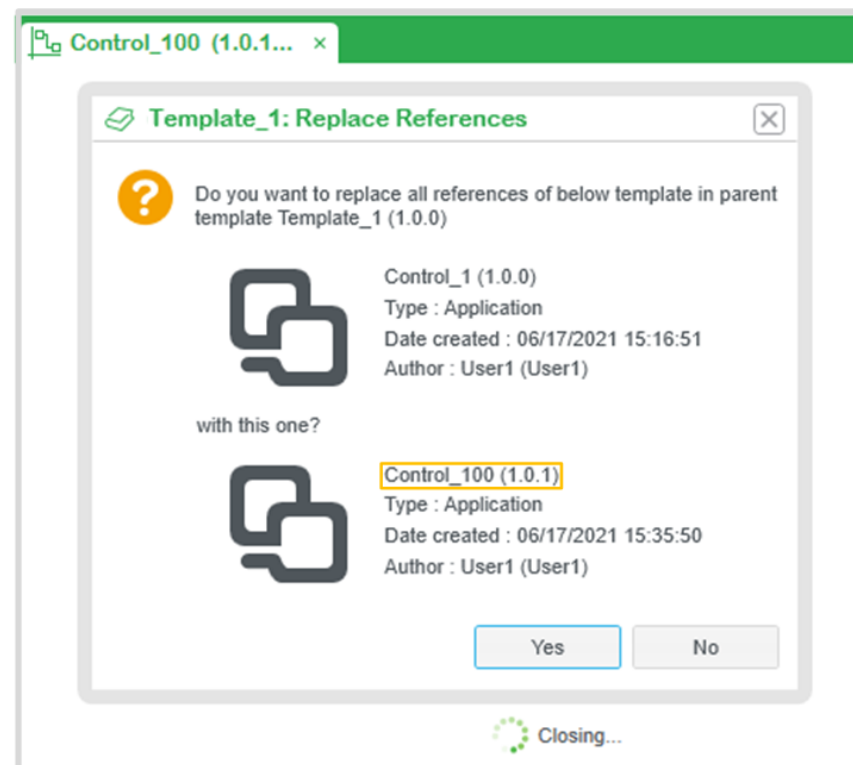
To apply the update of *Logic_1*, *Control_1* needs to be saved with a new version (1.0.1) and usability state *Approved*. In this example, *Control_1* is renamed *Control_100* to show the impact of renaming a template when saving it.



NOTE: You can perform additional modifications in *Control_1* before saving it.

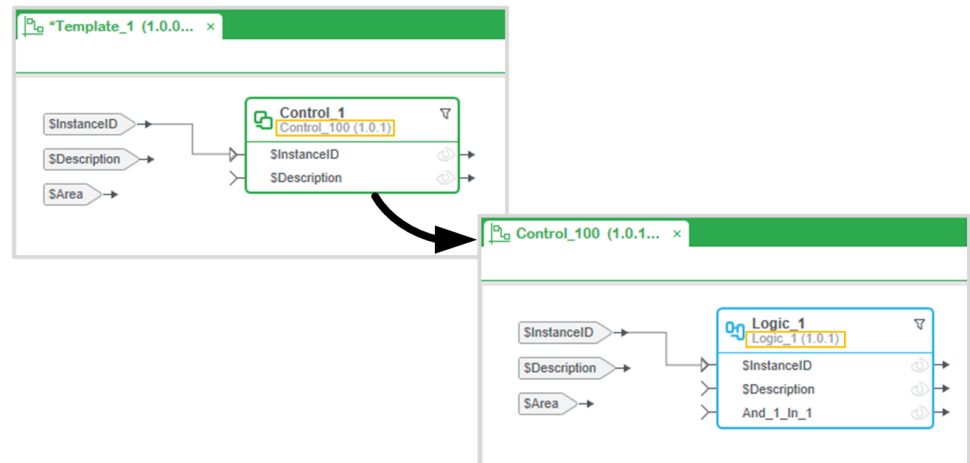
Closing the Mid-Level Template and Updating the Reference in Parent

After closing *Control_100*, the **Replace References** dialog box opens, which lets you *replace* reference *Control_1* v1.0.0 with *Control_100* version 1.0.1 inside *Template_1*.



Result of the Update Process

The following figure shows the result in *Template_1* after clicking **Yes** in the dialog box. The modifications in *Logic_1* and *Control_100* have been propagated to the highest level. To finalize the update process, *Template_1* needs to be saved.



Validating Templates

What’s in This Part

Validating Templates..... 318

Validating Templates

What’s in This Chapter

Validating Templates..... 318

Validating Templates

Overview

When you finish creating templates and/or have updated control module templates, you need to ensure that your design works as expected.

Template validation is a mandatory step in the process of creating templates before using them in the system engineering life cycle.

⚠ WARNING

LOSS OF CONTROL

- Perform a Failure Mode and Effects Analysis (FMEA) 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.
- Review the implications of communication link interruptions and take actions to mitigate.
- Provide independent paths for control functions (for example, emergency stop, over-limit conditions, and fault conditions) according to the safety analysis and applicable codes, and regulations.
- Apply local accident prevention and safety regulations and guidelines. ¹
- Test each implementation of this library 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.

Guidelines

- Instantiate control module templates instead of their individual elements because the generated facets and constituents may be different.
- Use the **Inspect Instance window** (see *EcoStruxure Process Expert, User Guide*) to gather complete information on the control module definition and the instance configuration.

Validation Process

To ensure that your control module works as expected, perform at least the following tasks in a test environment.

| Step | Action |
|------|---|
| 1 | Instantiate the control module. |
| 2 | In the folder of the application, ensure that the Data and Link statuses appear as expected. |
| 3 | Ensure that the elements appear correctly in the Instance Editor of the instance. Ensure that each element uses the correct template and version. |
| 4 | Ensure that for each element, the parameters appear as configured in the template. Ensure in particular if the parameter properties are as expected, for example, default values, data types and validation rules. |
| 5 | Select and clear optional services to test the possible combinations. |
| 6 | For each combination of element selection and parameter configuration, ensure that the appropriate facets are created by using the View Assignments window. Verify the identifiers of each facet. |
| 7 | For each combination, ensure that the exposed interfaces appear as expected in the Asset Workspace Editor or Links Editor and link them to ensure that they are functional. |
| 8 | Assign the facets to Control and Supervision Participant projects, and refine each one to ensure that code and data are generated as expected. For the Supervision Participant project, edit a page and verify genie assignment. |
| 9 | Generate each Participant project to help ensure that generation completes successfully. |
| 10 | Perform mappings. |
| 11 | Build each Participant project to help ensure that the process completes successfully. |
| 12 | Deploy and run both executables to test Control and Supervision functions. |

NOTE: Depending on the type of template that you have created, additional validation steps may be required.

Validating Implementation of the Naming Convention

To ensure that you have correctly implemented the naming convention in the control module work, perform, at least, the following tasks in a test environment:

| Step | Action |
|------|---|
| 1 | Create two instances of the control module. |
| 2 | Configure both instances to be identical. |
| 3 | Assign the facets to a Control Participant project, and refine each one to ensure that code is generated as expected. |
| 4 | Generate the Participant project to help ensure that generation completes successfully. |

NOTE: Depending on the type of template that you have created, additional validation steps may be required.

Glossary

A

application interface:

Mechanism that allows sharing data and manage dependencies between 2 instances/references.

The *application interface* allows you to link:

- applications instances/references to each other
- topological instances/references to each other

application link:

Links describing connections between instances of the application that are made by using application interfaces. The **Asset Workspace Editor** and **Links Editor** allow you to create and edit such links.

application template:

Global Template contained in the Global Templates library that models an object of the application of a system and its associated functions. The template encapsulates the necessary Participant capabilities.

Application templates are instantiated and can be configured to create the application of a system.

application:

The *application* models the complete functionality provided by the system through an application folder structure and instances of application templates.

area:

The *area*, when referring to access control, defines a topological, functional, or another user-based rule to restrict access in the scope of an automation system.

audit trail:

The *audit trail* records the following information for a given period:

- Who accesses a computer system.
- Which operations are performed.

B

binding:

The *binding* is the key mechanism that establishes relations between the following items of the facet and composite templates:

- Parameters
- Interfaces
- Elements

built Participant project:

The *built Participant project* is a generated, optionally refined, and built project, which is created by the corresponding Participant and that exists at the platform level in the form of an executable.

For example, files with .stu and .ctz extensions that are represented by Control and Supervision executables respectively are built Participant projects.

C

cardinality:

The *cardinality*, in the context of the interface model, is the number of connections that are supported by the objects playing the other role of the interface model.

communication channel:

A *communication channel* is the logical representation, at the platform level, of communication between controllers or between a controller and devices.

For example, a controller I/O scanner row is modeled by a *communication channel* for Modbus TCP communication in the executable of the Control project.

communication interface:

Mechanism that allows sharing data and manage dependencies between 2 instances or references.

Communication interfaces allow the platform to link topological instances/ references at the logical level (for example, I/O scanner, OPC Factory Server, Supervision I/O devices).

communication link:

Links describing logical connections between topological instances that are made by using communication interfaces.

For example, the communication link between 2 controllers exchanging data.

communication mapping:

The *communication mapping* process defines the communication aspects of a logical Participant project after being mapped to the topology.

composite template:

The *composite template* combines the capabilities of 1 or more facet templates, each encapsulating functionalities provided by one software Participant, and/or of other *composite templates*.

composite:

The *composite* is an instance of a composite template.

constituent encapsulation:

Process that is performed by using the **Facet Editor** and that allows you to:

- Select constituents that are created with a software Participant.
- Optionally, modify constituents inside the corresponding Participant.
- Include the selected constituents inside a facet template.

constituent:

Set of data provided by a software Participant, which can be global or local.

contents repository:

The *contents repository* is a global storage provided by the platform to manage global constituents and other files, such as Participant projects, used by the libraries and the systems.

D

deferring mechanism:

Mechanism that allows you to make available in the composite template that is at the highest level optional element selections, parameters, and/or interfaces of its references. By default, the mechanism is enabled for new items and can be disabled. For existing items, commands allow you to defer them manually.

deployed Participant project:

A *deployed Participant project* is a built project that has been transferred to a controller or other entity of the topology acting as an engine, and that you can execute.

DFB type:

The *derived function block type* is a programming language element that consists of:

- A data structure definition partitioned into input, output, public, and internal variables.
- A set of operations written in IEC 61131 programming languages to be performed upon the elements of the data structure when an instance of the function block type is started.

E**EcoStruxure Process Expert:**

Third generation name of the software platform. Formerly, StruxureWare Process Expert then, EcoStruxure Hybrid DCS.

element:

Element is the term that is used to describe the contents of templates at the next lowest level as well as the contents of interface models:

- For facet templates, elements are the constituents that the facet encapsulates.
- For composite templates, elements are:
 - Facet references
 - Composite references
- For interface models, elements are the data that is shared. See *interface elements*.

You can define properties and rules for elements during the Global Templates definition stage.

engine:

An *engine* is the projection of the hardware and software defined in the topology that is able to execute the deployed Participant project.

engineering client:

EcoStruxure Process Expert client connecting to the system server that you use to engineer and maintain systems. It can run on the system server and/or an engineering station.

Engineering clients use software Participants.

engineering station:

The *engineering station* is a computer running an EcoStruxure Process Expert engineering client to design, and maintain systems that exist on the server to which the engineering client connects.

executable:

The *executable* is a project component that represents the built Participant project. It contains the mapping information.

execution domain:

Property of the executable of a Participant project, which serves as a filter for selecting the scope of applicable services.

For example, it is used to define the boundaries for runtime navigation services or to restrict the exchange of data through peer to peer communication between projects that have executables with the same execution domain.

F

faceplate:

The *faceplate* is a component of a Supervision genie that provides a user interface to interact with the object that it represents.

facet element:

The *facet element* is the contents that can be accessed at the system level. Depending on the type of facet template, such elements can be either:

- The visible part of the constituents stored in the facet templates.
- A set of data defined by the facet template.

facet template:

The *facet template* is the smallest available template that encapsulates a specific capability provided by 1 software Participant only.

facet:

The *facet* is an instance of a facet template.

G

genie:

Animated graphic that is assigned to Supervision pages and that represents functionalities of instances of the application during operation.

Genies are encapsulated in Supervision facets.

global constituent:

Set of data provided by a software Participant that is a common resource referenced by several Global Templates. Global constituents are stored in the contents repository.

Global Templates:

Global Templates encapsulate one or more functions that can be customized and reused in any system created with EcoStruxure Process Expert. They are stored in the Global Templates library.

H

hardware mapping:

The *hardware mapping* process links the logical projection of the hardware that is defined in the application and assigned to projects to the hardware defined in the topology of the system.

For example, linking Control variables representing I/O signals to I/O channels of an I/O module.

HMI:

Abbreviation for *human machine interface*. It is a graphical operator interface for industrial equipment.

I

IEC:

The *International Electrotechnical Commission* is a non-profit and non-governmental international standard organization that prepares and publishes international standards for electrical, electronic, and related technologies.

instance element selection:

The *instance element selection* is a mechanism of the instantiation stage that allows you to customize an instance by selecting services that are provided by the template that the instance uses.

instance parameter:

Instance parameters are properties of the elements of an instance that you may be able to customize.

instance:

Abbreviation for object instance. It is the result of the instantiation of a template.

instantiation naming convention:

The *instantiation naming convention* defines the naming convention that the platform applies when you create instances.

interface element:

Represents the data that will be shared through an interface. Interface elements are defined during in the interface definition. Interface elements can be transformed by the interface by using expressions.

interface link:

Links describing logical connections between instances that are made by using interfaces.

The following types of links exist:

- Physical links
- Communication links
- Application links

Each type of link is made by using the corresponding interface (physical, communication, or application interface).

interface model:

The *interface Model* is a type of template that is available in the Global Templates library and that you can configure.

interface:

The *interface* is a reference, inside a composite or facet template, of an interface model. Interfaces are exposed by the facets of an instance, allowing you to make different types of links with other instances to share data.

In the context of the template definition, interfaces are a mechanism to define the links between references by declaring compatibility and/or requirement rules.

I/O:

Abbreviation for *Inputs/Outputs*.

L**local constituent:**

Set of data that is provided by a software Participant, encapsulated inside a facet template but not used at the system level. The local constituent is used to generate the contents of the logical Participant project. An example is Control Participant code that is encapsulated in a Control facet template.

logical Participant project:

The *logical Participant project* is a generated and refined project, which is created by the Participant but, which is not associated to the topology.

NOTE: The refinement of the project is optional.

M

mapping interface:

Mechanism that allows sharing data between two facets.

Mapping interfaces allow you to perform the hardware mapping. It is the process whereby you link facets assigned to projects to facets representing the hardware defined in the topology through matching mapping interfaces that these facets expose.

MES:

The *Manufacturing Execution Systems* is a control system for managing and monitoring work-in-process in a factory plant.

N

network variable:

The *network variable* is a peer to peer communication mechanism allowing you to share data between 2 or more Control projects.

O

object template:

An *object template* is a generic term that covers several reusable templates such as facet or composite templates.

OFS:

Abbreviation for OPC Factory Server.

operation client:

An EcoStruxure Process Expert client connecting to the system server that you can use during runtime to monitor and troubleshoot a system.

operator station:

The *operator station* is a computer running a Supervision client software.

P

Participant services:

Participant services are the functions provided by a software Participant when interacting with EcoStruxure Process Expert.

peer to peer communication:

Peer to peer communication is a data exchange mechanism between 2 or more Control projects, which uses the I/O scanner function of the controller acting as client.

physical (interface) link:

Links describing logical connections between topological instances that are made by using physical interfaces.

For example, the connection of a controller to an Ethernet network.

physical connection:

Link between topological entities representing the physical connections between controllers, station nodes, devices, and communication networks.

physical interface:

Physical interfaces allow the platform to link, at a physical level, topological instances to model the topology of the system.

platform:

Abbreviation for system platform. Represents the services that are provided by EcoStruxure Process Expert apart from the software Participants.

privilege:

Defines groups of functions that are provided by an application and granted to users through access control.

project container:

The *project container* is an organizational structure of a project to organize the facets that are assigned to it. Such a structure models the ones managed by the corresponding software Participant, and that are visible at the system level.

project facet:

Facet that is assigned to a project.

project:

Component of a system associated to a software Participant. Its structure contains elements that are managed by the software participant.

R

redundant controller:

Generic term that is used to refer a Quantum Hot Standby controller and/or an M580 redundant controller.

reference:

Defines facet and composite templates, which are contained inside other facet or composite templates in order to distinguish templates, which are used in the composition of other templates from the highest level templates, such as control modules.

role:

The *role* can have 2 different meanings,

- For access control:

The *role* groups functionalities to grant different levels of user rights, which combine areas and privileges to fulfill a set of services.

- For interface models:

The *role* defines the 2 sides of an interface, role A and role B.

runtime navigation services:

Describes the complete set of functionalities that are provided by EcoStruxure Process Expert operation client in runtime, such as process monitoring, viewing of instance information, diagnostics.

S

service mapping:

The *service mapping* links the execution capabilities of a project represented by the executable to an engine of the topology.

For example, it can link:

- The I/O server of a Supervision project to a station node representing the operation server.
- The executable of a Control project to a controller.

service:

In the context of projects, the *project service* is an organizational structure of a project to organize the execution capabilities. It models the structures that are managed by the corresponding software Participant, and that are visible and configured at the system level.

In the context of templates, a *service* represents a functionality provided by a Participant under the form of a facet referenced by the template.

software Participant:

An external tool that is used by the system server and clients on which it is installed. For example, the Control Participant.

software:

Refers to EcoStruxure Process Expert.

station node:

A *station node* represents a computer that can act as an engine to execute a Control project by using software emulating a simulator, or a Supervision project by using Supervision software.

subnet mask:

A 32-bit value that indicates the number of available host addresses on a subnet, which uses TCP/IP knowing that the first and last addresses are reserved (for example, subnet mask 255.255.192.0 allows 317 addresses for classless devices). It also indicates whether addressing is classful or classless. For classful addressing, it indicates the class of the network (for example, 255.255.255.0 is the subnet mask for class C networks).

Supervision client:

Client connecting to the Supervision server that provides runtime services to operate and monitor automation systems. The client runs on an operator station and/or operation server.

Supervision server:

Server running on the operation server or on the system server, and that provides runtime data to Supervision clients.

system engineering life cycle:

The *system engineering life cycle* represents the stages to engineer a system in EcoStruxure Process Expert.

system server:

EcoStruxure Process Expert server that hosts:

- Software Participants
- The database containing template libraries and system data

The system server manages requests from the EcoStruxure Process Expert clients.

system:

Models a physical automation system.

The system consists of the following components:

- Topology
- Application
- Participant projects

T

topological entity:

A *topological entity* is the representation of piece of hardware infrastructure.

For example, a controller.

topology:

Models the hardware and software infrastructure of a system through topological folders and entities.

Index

| | |
|--|-----|
| \$ parameters | |
| description | 151 |
| \$ prefix | |
| template identifier restrictions | 80 |
| \$Disable | |
| interface properties | 165 |
| \$FullName property | |
| description | 45 |
| \$IsConnected | |
| interface properties | 165 |
| \$System | |
| implicit parameters | 151 |

A

| | |
|---|-----|
| add | |
| binding functions | 113 |
| addresses | |
| propagating tag addresses by interfaces | 160 |
| alarm tags | |
| naming convention | 53 |
| aliases | |
| defining a naming convention | 168 |
| and | |
| binding functions | 112 |
| animated graphics | |
| configuring Genie parameters | 278 |
| creating and modifying Supervision Genies | 273 |
| encapsulating | 278 |
| Genie creation workflows | 272 |
| appearance | |
| instance appearance | 171 |
| attributes common definitions | |
| hardware family | 84 |
| hardware reference | 84 |
| network type | 84 |

B

| | |
|---|-----|
| binding functions | |
| description | 108 |
| LocationExtract | 123 |
| LocationGet | 121 |
| miscellaneous binding functions | 119 |
| other address calculation functions | 128 |
| topological address calculation functions | 124 |
| bindings | |
| Create Bindings dialog box | 246 |
| creating by using commands | 245 |
| creating by using the mouse | 245 |
| description | 244 |
| managing bindings | 248 |
| moving bindings | 251 |
| updating a binding destination | 251 |

C

| | |
|--|-----|
| Calculated Variable Tag elements | |
| description | 93 |
| cardinality | |
| examples of interface role cardinality | 65 |
| categories | |
| assigning parameters to categories | 153 |
| parameter categories | 151 |

| | |
|---|-----|
| changes log | |
| accessing | 238 |
| interfaces | 222 |
| saving changes in Global Templates | 254 |
| comparison functions | |
| binding functions | 110 |
| equal | 110 |
| greater than | 111 |
| greater than or equal | 111 |
| less than | 111 |
| less than or equal | 111 |
| not equal | 112 |
| composite definition | |
| properties (header) | 76 |
| Composite Editor | |
| changes log | 238 |
| menus and toolbars | 233 |
| panes | 238 |
| parameter pane | 238 |
| presentation | 230 |
| using the radar view | 241 |
| using the simplified view | 242 |
| composite template definition | |
| composite templates element rules | 77 |
| composite templates | |
| changes log | 238 |
| composite template definition | 75 |
| configuring | 286 |
| creating | 252 |
| modification strategy | 292 |
| composites | |
| composite template highlights | 30 |
| composition | |
| template composition strategy | 40 |
| concat | |
| binding functions | 108 |
| interface model element transformation | |
| functions | 69 |
| concatenation functions | |
| binding functions | 108 |
| concat | 108 |
| example | 110 |
| mconcat | 109 |
| configuration | |
| instance parameter configuration | 34 |
| connections | |
| binding description | 244 |
| consistency check | |
| impact on template creation | 37 |
| constituents | |
| creating with the control Participant | 137 |
| encapsulating (control) | 260 |
| encapsulating control constituents example | 266 |
| generation of data | 36 |
| modifying encapsulated control constituents | 270 |
| prerequisites for control | 137 |
| project file compatibility | 260 |
| project file compatibility (supervision) | 275 |
| using the control Participant | 137 |
| content repository | |
| description | 59 |
| control | |
| creating control constituents | 137 |
| prerequisites for constituents | 137 |
| Control Expert project file | |
| creating control constituents | 137 |
| control module templates | |
| configuring | 289 |
| validating | 318 |

| | |
|--|-----|
| control modules | |
| template modification strategy | 292 |
| control services | |
| defining | 145 |
| convention | |
| control resource naming convention | 48 |
| resource naming convention | 48 |
| supervision resource naming convention | 53 |
| copying | |
| Global Templates and folders | 256 |
| CrDocument | |
| binding functions | 115 |
| creating | |
| Global Template editors | 220 |
| guidelines for template creation | 16 |

D

| | |
|---|-----|
| data | |
| propagating | 43 |
| propagation principles | 43 |
| data dictionary | |
| HMI attribute for variables | 137 |
| database | |
| templates in the database | 61 |
| deferring parameters | 154 |
| definition | |
| composite template definition | 75 |
| facet template definition | 73 |
| global template definition | 21 |
| Global Template definition | 62 |
| interface model definition | 62 |
| Deprecated | |
| template usability state | 82 |
| DFB | |
| creating control constituents | 137 |
| direction | |
| interface model element direction | 68 |
| Disk Variable Tag elements | |
| description | 95 |
| div | |
| binding functions | 114 |
| document functions | |
| CrDocument | 115 |
| Url | 118 |
| document outline | |
| composite template definition | 82 |
| facet template definition | 82 |
| duplicating | |
| Global Templates | 305 |

E

| | |
|--|-----|
| Edit/Extend Interface | |
| command description | 308 |
| editing interfaces | |
| changes log | 222 |
| editors | |
| Composite Editor | 230 |
| Facet Editor | 227 |
| Interface Editor | 221 |
| opening Global Template editors | 220 |
| element count | |
| interface model element transformation | |
| functions | 70 |
| element rules | |
| composite template definition | 77 |
| elements | |

| | |
|---|-----|
| Calculated Variable Tag element description | 93 |
| description of facet elements | 92 |
| Disk Variable Tag element description | 95 |
| Equipment element description | 96 |
| Equipment Group of Messages element | |
| description | 107 |
| Equipment Parameter element description | 105 |
| general description | 93 |
| interface definition elements pane | 66 |
| replacing templates of elements | 300 |
| selection | 32 |
| template element layout | 42 |
| updating templates of elements | 294 |
| encapsulating | |
| control constituent example | 266 |
| control constituents | 260 |
| modifying encapsulated control constituents | 270 |
| supervision Genies | 278 |
| enumerations | |
| creating as parameters | 153 |
| equal | |
| binding functions | 110 |
| Equipment elements | |
| description | 96 |
| Equipment Group of Messages elements | |
| description | 107 |
| equipment names | |
| creation and update examples | 103 |
| Equipment Parameter elements | |
| description | 105 |
| equipment properties | |
| update examples | 103 |
| Exclude command | |
| description | 228 |
| expressions | |
| in control constituents | 139 |
| extension | |
| extension in Select Variables window | 266 |

F

| | |
|---|-----|
| facet definition | |
| properties of facet templates | 74 |
| Facet Editor | |
| changes log | 238 |
| description | 227 |
| menus and toolbars | 233 |
| panes | 238 |
| parameter pane | 238 |
| using the radar view | 241 |
| using the simplified view | 242 |
| facet template | |
| encapsulating control constituents | 260 |
| modifying encapsulated control constituents | 270 |
| facet templates | |
| changes log | 238 |
| configuring | 283 |
| creating | 252 |
| encapsulating supervision Genies | 278 |
| facet template definition | 73 |
| facet template highlights | 23 |
| facet template properties | 74 |
| modification strategy | 292 |
| facets | |
| description of facet elements | 92 |
| fan-in functions | |
| binding functions | 115 |
| filters | |
| template editor element filters | 240 |

| | | | | | |
|---|-----|--|--|---|----------|
| folders | | | | defining control services | 146, 168 |
| copying and pasting library folders | 256 | | | defining parameters | 151 |
| library folder structure | 58 | | | defining supervision services | 148 |
| Format | | | | library folder structure | 58 |
| parameter property of templates | 86 | | | modifying Schneider Electric templates | 39 |
| foundation library | | | | prerequisites for control constituents | 137 |
| reusing Global Templates | 167 | | | Process Expert database | 61 |
| FullName property | | | | template composition strategy | 40 |
| description | 45 | | | template design guidelines | 16 |
| functions | | | | template element layout | 42 |
| binding function description | 108 | | | validating templates | 318 |
| defining control services | 145 | | | | |
| defining supervision services | 147 | | | | |
| function analysis of the template | 142 | | | | |
| if | 121 | | | | |
| left | 120 | | | | |
| length | 120 | | | | |
| making a sketch | 143 | | | | |
| max | 119 | | | | |
| mid | 121 | | | | |
| min | 120 | | | | |
| miscellaneous binding functions | 119 | | | | |
| right | 120 | | | | |
| | | | | | |
| G | | | | H | |
| general-purpose library | | | | hardware family | |
| reusing Global Templates | 167 | | | composite and facet template definition | 84 |
| generation | | | | hardware reference | |
| constituent generation | 36 | | | composite and facet template definition | 84 |
| Genies | | | | header | |
| configuring Genie parameters | 278 | | | composite template definition | 76 |
| creating | 273 | | | interface model common definition | 80 |
| creation workflows | 272 | | | headers | |
| encapsulating | 278 | | | interface definition header pane | 63 |
| modifying | 273 | | | Hide Deferred filter | 240 |
| modifying Genies in facet templates | 281 | | | Hide Unbound filter | 240 |
| replacing Genie elements in facet templates | 281 | | | Hide Unchecked filter | 240 |
| reusing existing genies | 273 | | | HMI attribute | |
| Global Template editors | | | | selecting when encapsulating | 137 |
| opening | 220 | | | | |
| Global Templates | | | | I | |
| common definitions | 80 | | | identifier | |
| composite template definition | 75 | | | template identifier and version | 22 |
| copying and pasting | 256 | | | identifiers | |
| definition | 62 | | | changing the identifier of Global Templates | 254 |
| dialog box | 252 | | | template identifier description | 80 |
| duplicating | 305 | | | if | |
| facet template definition | 73 | | | binding functions | 121 |
| interface model definition | 62 | | | implicit parameters | |
| renaming | 254 | | | default property values | 87 |
| replacing templates of elements | 300 | | | description | 151 |
| saving changes | 254 | | | include project files | |
| updating | 294 | | | adding supervision project files | 278 |
| updating at folder level | 297 | | | include projects | |
| good practices | | | | project file compatibility (supervision) | 275 |
| control naming strategy | 48 | | | input pins | |
| interface links | 158 | | | element input pins | 43 |
| greater than | | | | instances | |
| binding functions | 111 | | | appearance | 171 |
| greater than or equal | | | | instantiation | |
| binding functions | 111 | | | troubleshooting instantiation issues | 139 |
| guidelines | | | | Interface Editor | |
| configuring composite templates | 286 | | | pane description | 222 |
| configuring control module templates | 289 | | | presentation | 221 |
| configuring facet templates | 284 | | | interface links | |
| configuring interface models | 257 | | | defining | 157 |
| content repository | 60 | | | Inspect Instance window | 35 |
| control naming strategy | 43 | | | interface links for instances | 35 |
| | | | | other interface link aspects | 163 |
| | | | | types of links | 158 |
| | | | | interface models | |
| | | | | configuring | 257 |
| | | | | creating | 252 |
| | | | | element direction | 68 |
| | | | | elements mode | 68 |
| | | | | interface definition elements pane | 66 |
| | | | | interface definition header pane | 63 |
| | | | | interface definition Toolbox pane | 68 |

| | |
|---|-----|
| interface element rules | 72 |
| interface model definition | 62 |
| interface model highlights | 25 |
| properties | 29 |
| interface properties | |
| \$Disable and \$IsConnected | 165 |
| interface model properties | 29 |
| interface rules | |
| composite and facet template definition | 88 |
| interfaces | |
| changes log | 222 |
| editing/extending interfaces | 308 |
| modification strategy | 292 |

L

| | |
|--|-----|
| layout | |
| template element layout | 42 |
| left | |
| binding functions | 120 |
| length | |
| binding functions | 120 |
| less than | |
| binding functions | 111 |
| less than or equal | |
| binding functions | 111 |
| libraries | |
| folder structure | 58 |
| modifying Schneider Electric templates | 39 |
| reusing Global Templates | 167 |
| template identifier and version | 22 |
| linking elements | |
| Create Bindings dialog box | 246 |
| creating bindings by using commands | 245 |
| creating bindings by using the mouse | 245 |
| links | |
| binding description | 244 |
| defining | 157 |
| interface links for instances | 35 |
| managing bindings | 248 |
| other interface link aspects | 163 |
| types of links | 158 |
| Links Editor | |
| instance appearance | 171 |
| using the radar view | 241 |
| list | |
| interface model element transformation | |
| functions | 70 |
| LocationExtract | |
| binding functions | 123 |
| LocationGet | |
| binding functions | 121 |
| logical functions | |
| and | 112 |
| binding functions | 112 |
| not | 113 |
| or | 112 |
| xor | 113 |

M

| | |
|-------------------------|-----|
| mathematical functions | |
| add | 113 |
| binding functions | 113 |
| div | 114 |
| mul | 114 |
| sub | 113 |
| max | |

| | |
|--|-----|
| binding functions | 119 |
| interface model element transformation | |
| functions | 71 |
| Mconcat | |
| binding functions | 109 |
| menus | |
| common template editor menus | 233 |
| mid | |
| binding functions | 121 |
| min | |
| binding functions | 120 |
| interface model element transformation | |
| functions | 71 |
| mode | |
| interface model element mode | 68 |
| modifying | |
| Schneider Electric templates | 39 |
| mul | |
| binding functions | 114 |
| multiplexer functions | |
| mux | 114 |
| <i>mux</i> | |
| binding functions | 114 |

N

| | |
|---|-----|
| names | |
| defining | 168 |
| propagating | 43 |
| variable and section name limitations | 48 |
| naming | |
| control naming convention | 48 |
| resource naming convention | 48 |
| supervision naming convention | 53 |
| naming convention | |
| defining | 168 |
| for Schneider Electric templates | 22 |
| nested interfaces | |
| creating and configuring | 257 |
| highlights | 25 |
| network type | |
| composite and facet template definition | 84 |
| not | |
| binding functions | 113 |
| Not Approved | |
| template usability state | 82 |
| not equal | |
| binding functions | 112 |

O

| | |
|-------------------------------------|-----|
| Obsolete | |
| template usability state | 82 |
| operators and operands | |
| in control constituents | 139 |
| optional | |
| element selection | 32 |
| or | |
| binding functions | 112 |
| order of elements | |
| defining display order | 287 |
| other address calculation functions | |
| MaHrOffset | 135 |
| MaStbModS | 135 |
| TaQAiVs | 132 |
| TaQAiVsw | 133 |
| TaQAoV | 133 |
| TaQAoVs | 134 |

| | | | |
|---|-----|---|-----|
| TaQDiV | 128 | replacing templates of elements | 300 |
| TaQDiVs | 129 | replacing Genie elements in facet templates | 281 |
| TaQDoV | 130 | Replacing references in parent templates | 293 |
| TaQDoVs | 130 | right | |
| TaQDoVvi | 131 | binding functions | 120 |
| output pins | | rules | |
| element output pins | 43 | composite template element rules | 77 |
| P | | Rules pane (Interface Editor) | 72 |
| panes | | template interface rules | 88 |
| common template editor panes | 238 | runtime navigation services | |
| parameter common definitions | | CrDocument functions | 115 |
| properties | 86 | Url functions | 118 |
| parameter names | | runtime parameters (supervision) | |
| generating and propagating | 45 | Equipment Group of Messages element | |
| parameter pane | | description | 107 |
| changing the position of parameter categories | 155 | Equipment Parameter element description | 105 |
| changing the position of parameters | 155 | S | |
| in template editors | 238 | saving | |
| parameters | | Global Templates | 254 |
| assigning parameters to categories | 154 | saving templates | |
| configuration | 34 | tracking changes in the changes log | 254 |
| creating | 153 | Schneider Electric | |
| customizing | 153 | modifying templates | 39 |
| deferring parameters | 154 | sections | |
| defining | 151 | name length limitations | 48 |
| implicit parameters | 151 | Select Genie | |
| implicit parameters of templates | 86 | window description | 278 |
| template common definition | 85 | Select Variables | |
| Participant | | window description | 261 |
| creating control constituents | 137 | Select Variables window | |
| pasting | | configuring extension | 266 |
| Global Templates and folders | 256 | services | |
| prefix | | defining control services | 145 |
| for Schneider Electric template names | 22 | defining parameters | 151 |
| prerequisites | | defining supervision services | 147 |
| for control constituents | 137 | function analysis | 142 |
| printing | | making a sketch | 143 |
| template layouts | 235 | signature | |
| project file | | template common definition | 82 |
| creating control constituents | 137 | simplified view | |
| project files | | template editors | 242 |
| adding supervision project files | 278 | standard | |
| project file (.ctz) compatibility (supervision) | 275 | template common definition | 82 |
| propagation | | state | |
| data propagation principles | 43 | template common definition | 82 |
| properties | | sub | |
| composite template common definition | 80 | binding functions | 113 |
| facet template common definition | 80 | sum | |
| facet template definition | 74 | interface model element transformation | |
| instance appearance | 171 | functions | 70 |
| interface model properties | 29 | supervision services | |
| properties (header) | | defining | 147 |
| composite template definition | 76 | support | |
| protection | | guidelines to facilitate support | 16 |
| template common definition | 81 | system parameters | |
| R | | implicit parameters of templates | 86 |
| radar view | | T | |
| description | 241 | tag addresses | |
| renaming | | propagating by interfaces | 160 |
| Global Templates | 254 | tags | |
| Replace dialog box | | naming convention | 53 |
| description | 308 | template | |
| Replace References dialog box | 293 | identifier and version | 22 |
| replacing | | Template Creation Wizard | 175 |

| | | | |
|---|----------|---|-----|
| template definition | | TaMpAiVs..... | 126 |
| attributes common definition | 84 | TaMpAoVs..... | 127 |
| comparison binding functions..... | 110 | TaMpDiVs | 125 |
| concatenation binding functions | 108 | TaMpDoVs | 126 |
| CrDocument functions | 115 | TaMpModS..... | 127 |
| fan in function | 115 | TaQModS | 125 |
| LocationExtract functions..... | 123 | tracking changes | |
| LocationGet functions..... | 121 | template changes log | 254 |
| logical functions | 112 | transformation functions | |
| mathematical functions..... | 113 | concat..... | 69 |
| miscellaneous binding functions..... | 119 | element count..... | 70 |
| multiplexer function | 114 | list..... | 70 |
| other address calculation functions..... | 128 | max | 71 |
| parameter common definitions | 85 | min | 71 |
| template interface rules | 88 | sum | 70 |
| topological address calculation functions | 124 | transformations | |
| Url functions | 118 | interface definition transformations functions..... | 68 |
| template functions | | troubleshooting | |
| function analysis | 142 | troubleshooting instantiation issues | 139 |
| making a sketch | 143 | | |
| templates | | U | |
| actions on new templates | 253 | Update Binding | |
| common default parameters | 253 | dialog box description..... | 251 |
| composite template highlights | 30 | Update References dialog box | 293 |
| composition strategy | 40 | updating | |
| configuring composite templates | 286 | Global Templates | 294 |
| configuring control module templates..... | 289 | Global Templates at folder level..... | 297 |
| configuring facet templates | 283 | template modification strategy..... | 292 |
| control resource naming convention | 48 | templates of elements | 294 |
| copying and pasting Global Templates..... | 256 | updating parent templates | |
| creation process summary..... | 17 | editing templates..... | 255 |
| duplicating Global Templates | 305 | Updating references in parent templates | 293 |
| element layout | 42 | Url | |
| facet template highlights | 23 | binding functions..... | 118 |
| global template definition | 21 | usability state | |
| Global Template definition..... | 62 | template common definition | 82 |
| Global Template editors..... | 220 | | |
| guidelines for template creation..... | 16 | V | |
| instance appearance | 171 | valid | |
| interface model highlights | 25 | template common definition | 81 |
| library folder structure..... | 58 | validating | |
| modification strategy | 292 | templates | 318 |
| modifying Schneider Electric templates..... | 39 | variable names | |
| purpose..... | 20 | generating and propagating | 45 |
| renaming Global Templates | 254 | variable tags | |
| replacing templates of elements..... | 300 | naming convention | 53 |
| resource naming convention | 48 | variables | |
| reusing Global Templates | 167 | name length limitations..... | 48 |
| saving changes in Global Templates..... | 254 | propagating variable names..... | 43 |
| supervision resource naming convention | 53 | version | |
| template highlights | 21 | control project file for encapsulation..... | 260 |
| template identifier description..... | 80 | template identifier and version..... | 22 |
| template version components | 81 | versions | |
| troubleshooting instantiation issues | 139 | DFB versions..... | 137 |
| types of templates..... | 21 | project file versions | 137 |
| updating Global Templates | 294 | template version description | 81 |
| updating templates at folder level | 297 | | |
| validating..... | 318 | W | |
| Templatize Genie | | wizards | |
| window description..... | 273 | Template Creation Wizard..... | 175 |
| Templatizer | | | |
| replacing Genies elements in facet templates..... | 281 | X | |
| toolbars | | xor | |
| common template editor toolbars | 233 | | |
| Toolbox | | | |
| interface definition Toolbox pane | 68 | | |
| topological address calculation functions | | | |
| binding functions..... | 124, 128 | | |
| TaCoDevS..... | 124 | | |

binding functions..... 113

Z

zooming
 using the radar view241
zooming out
 simplified view in template editors242

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.

© 2023 Schneider Electric. All rights reserved.

EIO0000001986.07