Siemens
Industry
Online
Support

APPLICATION EXAMPLE

Engineering guideline for WinCC Unified.

SIMATIC WinCC Unified V18 / V19

SIEMENS

Legal information

Use of application examples
Application examples illustrate the solution of automation tasks through an interaction of several components in the form of text, graphics and/or software modules. The application examples are a free service by Siemens AG and/or a subsidiary of Siemens AG ("Siemens"). They are non-binding and make no claim to completeness or functionality regarding configuration and equipment. The application examples merely offer help with typical tasks; they do not constitute customer-specific solutions. You yourself are responsible for the proper and safe operation of the products in accordance with applicable regulations and must also check the function of the respective application example and customize it for your system.

Siemens grants you the non-exclusive, non-sublicensable and non-transferable right to have the application examples used by technically trained personnel. Any change to the application examples is your responsibility. Sharing the application examples with third parties or copying the application examples or excerpts thereof is permitted only in combination with your own products. The application examples are not required to undergo the customary tests and quality inspections of a chargeable product; they may have functional and performance defects as well as errors. It is your responsibility to use them in such a manner that any malfunctions that may occur do not result in property damage or injury to persons.

Disclaimer of liability
Siemens shall not assume any liability, for any legal reason whatsoever, including, without limitation, liability for the usability, availability, completeness and freedom from defects of the application examples as well as for related information, configuration and performance data and any damage caused thereby. This shall not apply in cases of mandatory liability, for example under the German Product Liability Act, or in cases of intent, gross negligence, or culpable loss of life, bodily injury or damage to health, non-compliance with a guarantee, fraudulent non-disclosure of a defect, or culpable breach of material contractual obligations. Claims for damages arising from a breach of material contractual obligations shall however be limited to the foreseeable damage typical of the type of agreement, unless liability arises from intent or gross negligence or is based on loss of life, bodily injury or damage to health. The foregoing provisions do not imply any change in the burden of proof to your detriment. You shall indemnify Siemens against existing or future claims of third parties in this connection except where Siemens is mandatorily liable.

By using the application examples you acknowledge that Siemens cannot be held liable for any damage beyond the liability provisions described.

Other information
Siemens reserves the right to make changes to the application examples at any time without notice. In case of discrepancies between the suggestions in the application examples and other Siemens publications such as catalogs, the content of the other documentation shall have precedence.

The Siemens terms of use (https://support.industry.siemens.com) shall also apply.

Security information
Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit https://www.siemens.com/industrialsecurity.

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customer's exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed under https://www.siemens.com/cert.

Table of contents

1. Introduction 5

1.1. Overview 5

1.2. Components used 6

1.3. Explanation of the Symbols used 6

2. Preliminary Information 7

2.1. Reasons for using the Engineering Guideline 7

2.2. Workflow for new Configurations 8

2.2.1. Use Case: State dependent scripting: Show / Hide screen content 11

2.3. Relevant sections for Screen change 13

3. Best Practices of Screen Engineering 17

3.1. Use of screen objects 17

3.1.1. Screen object selection 19

3.1.1.1. Use Case: Colored square box 20

3.1.1.2. Use Case: Button without implementation of the press and release event 21

3.1.1.3. Use Case: Simple Text label V19 23

3.1.1.4. Use Case: Colored Screen Background 24

3.1.1.5. Use Case: Custom Styles V19 25

3.1.2. Tidy up the screen / observe system limits 28

3.1.3. Usage of screen windows 29

3.1.4. Screen object visibility 30

3.1.4.1. Use Case: Visibility dynamization of multiple screen objects by the same condition 30

3.1.5. Unified Controls 32

3.1.5.1. Use Case: Different settings in Runtime for Unified Controls 32

3.1.5.2. Use Case: Alarm control 33

3.1.6. Graphics and SVGs 35

3.1.6.1. Use Case: Visualize Patterns and composed objects 35

3.1.6.2. Use Case: Composed Objects with dynamizaton → dynamic SVG 36

3.2. Use of dynamizations 38

3.3. Use of Faceplates 41

3.3.1. Use Case: Strings in the property interface 41

3.3.2. Use Case: Resource Lists with few entries 43

3.3.3. Use Case: Faceplate that is not always visible in Runtime V19 44

© Siemens 2024

3.4. Use of scripts 45
3.4.1. Script Triggers 45
3.4.1.1. Use Case: Scripts using the same trigger tag 46
3.4.2. Efficient Code 48
3.4.2.1. Use Case: Write and read multiple tags 50
3.5. Others 52
3.5.1. PLC UDT Arrays with Multiplexing 53

4. Analysis of an existing project 55

4.1. ShowScripts Add-in 58
4.1.1. Download and installation 58
4.1.2. Usage of the ShowScripts Add-in 59
4.1.3. Preparation of the screen overview file 61
4.1.4. Analysis of the screen overview file 63
4.1.4.1. Analysis of screen context 63
4.1.4.2. Usage of Controls 65
4.1.4.3. Usage of Dynamizations 65
4.1.5. Analysis of the exported scripts 67
4.2. WinCC Unified JS Connector V19 70
4.3. Runtime analysis 72
4.3.1. RTIL Trace Viewer 72
4.3.2. Script Debugger 73

5. Appendix 79

5.1. Service and support 79
5.2. Links and literature 80
5.3. Change documentation 80

1. Introduction

1.1. Overview

SIMATIC WinCC Unified is the completely newly developed visualization system from Siemens in the automation environment. The SIMATIC WinCC Unified system consists of the SIMATIC WinCC Unified visualization software and the new SIMATIC HMI Unified Comfort Panels as well as the Unified Basic Panels.

The Unified Comfort Panels extend the product range of the SIMATIC Advanced Panel-based HMIs and are the successor devices of the SIMATIC HMI Comfort Panels. In addition to the new hardware, there are numerous innovative new features that have a lot of impact in the way you use things. To get a better overview of the steps that help you get the best out of the new features and behaviors in Unified Comfort panels and PC Runtime, this document will address several topics. We will discuss the advantages and guide you through the steps of the changed process.

The content of this document refers to the entire Unified HMI portfolio, unless there are further references to device-specific information.

© Siemens 2024

1.2. Components used

This application example has been created with the following hard- and software components:

Component	Number	Article Number	Note
WinCC Unified Engineering V18	1	6AV215.-...01-8A.5	Engineering in TIA Portal
WinCC Unified Engineering V19	1	6AV215.-...02-3.A5	Engineering in TIA Portal
SIMATIC HMI Unified PC Runtime V18	1	6AV2154-..B01-8.A0	Runtime System
SIMATIC HMI Unified PC Runtime V19	1	6AV2154-..B02-3.A0	Runtime System
SIMATIC HMI Unified Comfort Panel MTP1200	1	6AV2128-3MB06-0AX1	Alternatively, any other SIMATIC HMI Unified Comfort Panel can be used.

You can purchase these components from the Siemens Industry Mall.

NOTE Please note that all functions shown or explained in the document only apply to WinCC Unified V18 including updates and V19. The additional functionalities of the any V19 updates are not covered in this document!

This application example consists of the following components:

Component	File name	Note
Manual	109827603_WinCC_Unified_engineering_guideline_DOC_V1_en.pdf	This document

1.3. Explanation of the Symbols used

The following table shows the used symbols that indicate for which version the described recommendation can be applied. If no symbol is used, the content has no restrictions on the version.

Symbol	Unified Version
V18	Only related to V18
V19	Only related to V19

© Siemens 2024

2. Preliminary Information

2.1. Reasons for using the Engineering Guideline

Unified Panel and PC-RT Devices offer the latest technologies, such as HTML5, JavaScript and scalable vector graphics (SVGs). This makes them more powerful and able to offer a wider range of functions and capabilities than the old Comfort Panels and WinCC RT Advanced systems, allowing you to get more out of your system. In order to use them efficient this document provides some recommendations for the engineering. The presented implementations can be helpful if you start a new project but also if you have an existing project already there.

When creating a new project, it is recommended to first work through the engineering guideline. It will provide the knowledge to use the resources of the devices the most efficient way.

But also, if there is already an existing Unified HMI project, this guideline contributes to further improve the implementation. There is also an explicit section on how to analyze a project to find the areas that can be improved easily.

If optimizations regarding Java Script code in general, the style guide for scripting and the HMI layout can be found in additional documentation: StyleChecker, HMI Template Suite, Tips and Tricks for Scripting

All tools can also be found in section 5.2 Links and literature.

© Siemens 2024

2.2. Workflow for new Configurations

The moment you start creating completely new configurations or even just new screens is the perfect opportunity for you to utilize the new WinCC Unified function in the most efficient way. In this context the term view is used and stands for the entire display of an HMI and thus for a combination of several screens in a screen window arrangement. When creating new views, the first step is to collect all the content that should be displayed in advance. Depending on its reusability, you decide to categorize into individual content or content that will be used in multiple screens. The procedure of the most efficient handling is explained in the following scheme.

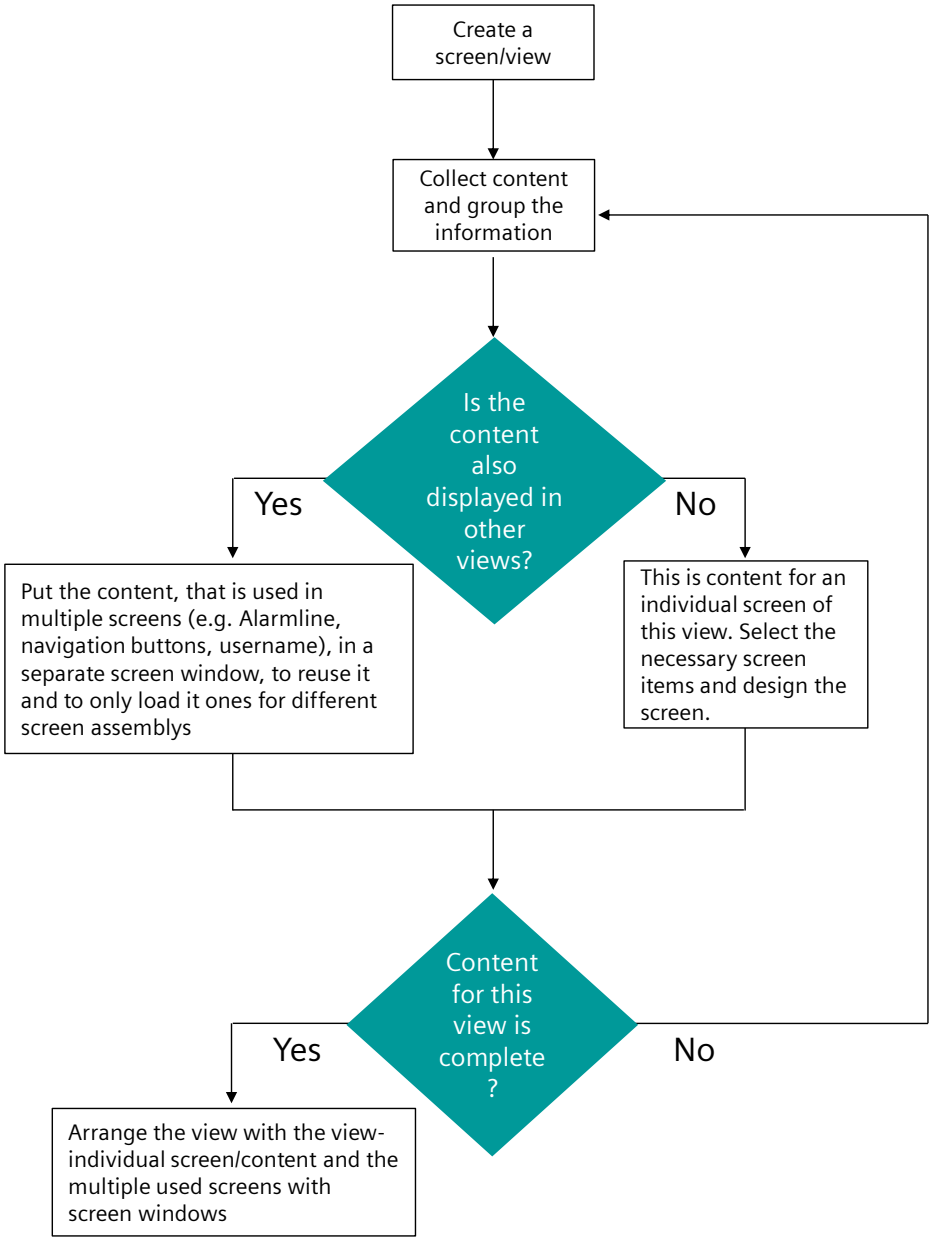


Figure 2-1 Flowchart: Creating a view with screens

The result of this configuration could be a screen layout like it is displayed in the following figure. All individual screen windows are highlighted with an orange frame.

© Siemens 2024 |

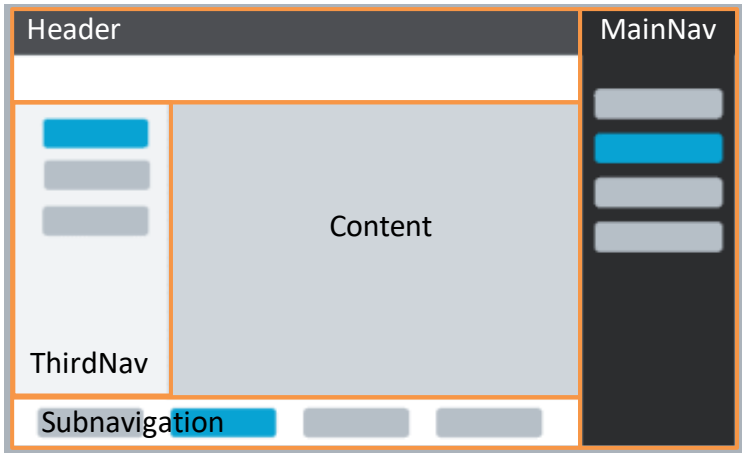


Figure 2-2 Example of a view using several screen windows

Once you have determined which elements are to be placed on which screen, you can continue working on the individual screens. Most of the screen items need dynamic properties or will use events to trigger system functions or scripts. To implement these dynamizations, the following scheme shows you which implementation might be best for your application. Depending on the desired functionality, select a suitable element from the toolbox and drag it onto the screen if it is not already there.

© Siemens 2024 |

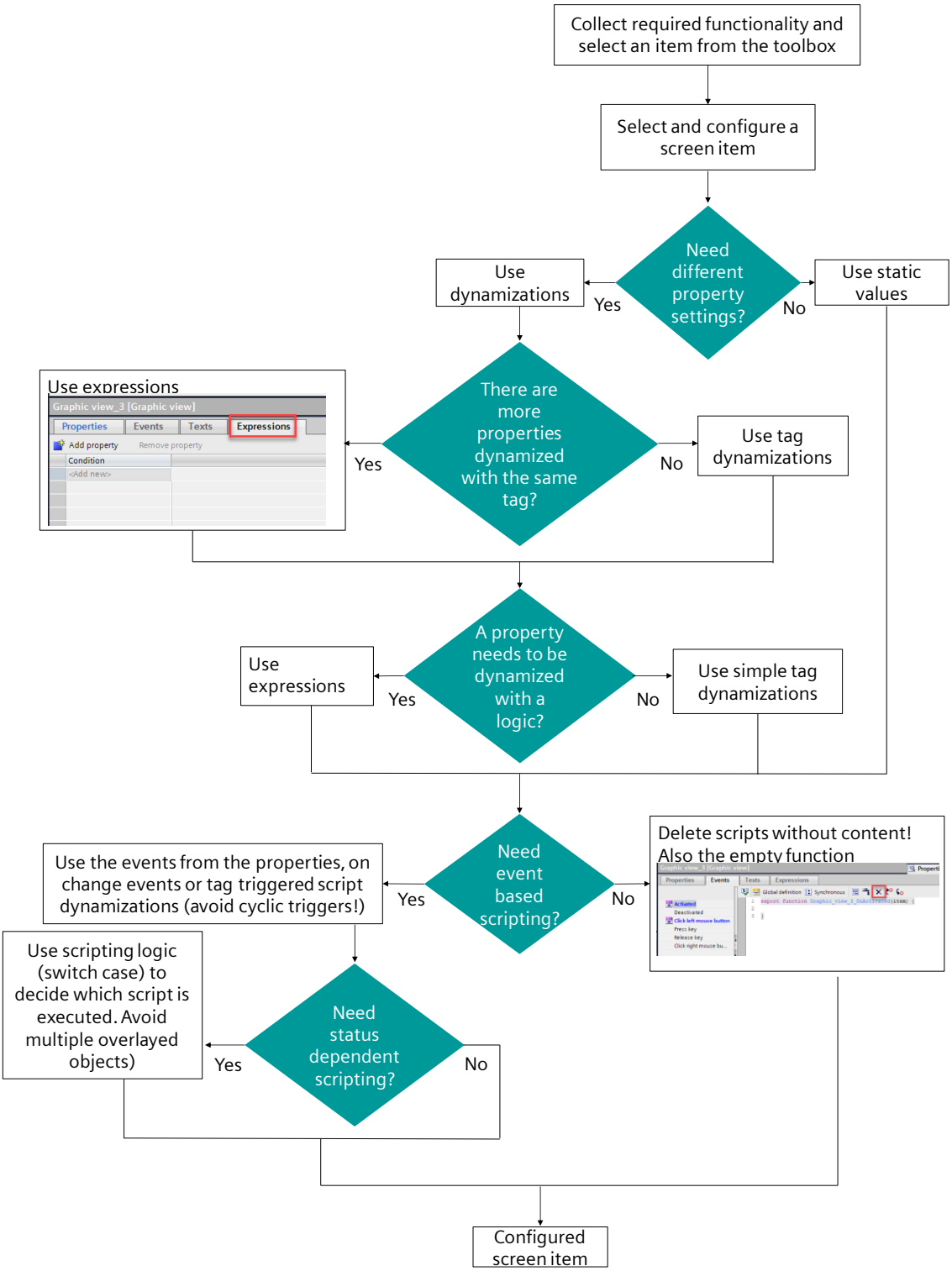


Figure 2-3 Configuring a new screen item flowchart

NOTE In general, tag dynamization is the most recommended dynamization type. If a more complex logic is necessary, use the expressions. Try to avoid script dynamizations. Nevertheless, if a scripting logic can replace multiple overlayed screen items, then use scripts. e.g., a button to show and/or hide other elements: Use one button and decide through scripting which method (show or hide) is executed (see 2.2.1).

© Siemens 2024 |

The following section explains the practice of a scripting logic instead of multiple objects with a simple use case.

2.2.1. Use Case: State dependent scripting: Show / Hide screen content

Scripting in WinCC Unified now offers many more options for configuring screen elements and makes them much more flexible. As described in the following use case, a single button can trigger different actions depending on the current state.

Description

The concrete action or scripting behind an event can often depend on the current state. In this exemplary use case there should be a button to show and hide a specific screen content (here: a rectangle), dependent on the current visibility of that screen content. One solution could be to use two different buttons, that implement once the show-action and once the hide-action and to change the visibility of these two buttons also with the current state. The second solution could be, having only one button, and decide by scripting (switch ... case), which action should be done.

Solution

Use a single button for this task and implement a scripting logic for the decision if the visibility of the content needs to be set to be visible or not. Directly use the same tag (here: "ContentVisible") for

- Indicator of the current state
- the visibility dynamization of the content
- the button text dynamization

As this use case is very simple, the inverting of the state tag "ContentVisible" in the scripting logic could also be just done by a system function through the button click event. But to demonstrate how the implementation could look like for a more complex use case the screenshot shows the implementation with switch case.

To change the button text for the different states, it is dynamized with a text list and again the state tag.

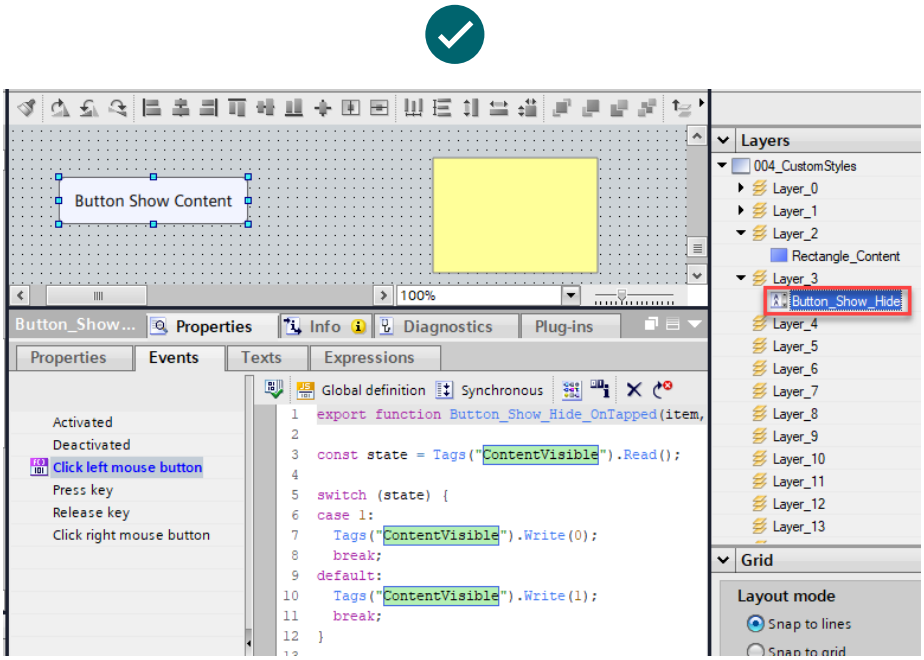


Figure 2-4 Show-Hide button use case scripting logic

© Siemens 2024

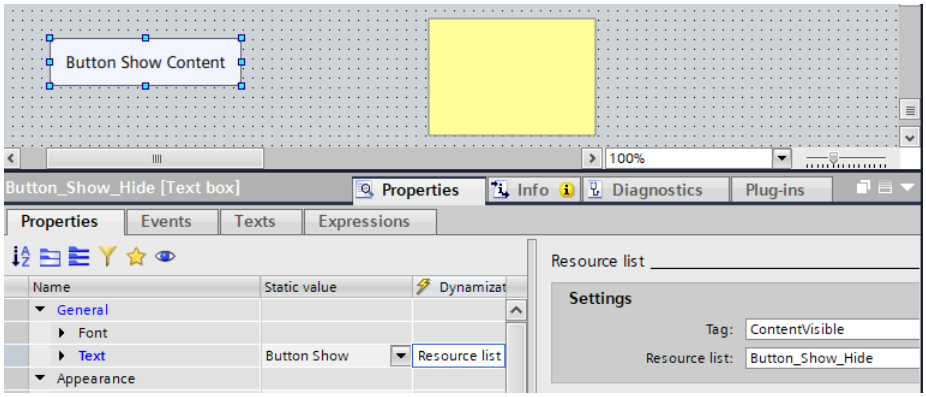


Figure 2-5 Button Show-Hide use case button text dynamization

In the following solution, that is NOT recommended, two overlaying buttons are used, one to set the visibility of the rectangle to true and a second one to set the visibility to false. To make the buttons accessible at the right time, their visibility is also dynamized with the visibility state of the rectangle. For such use cases with a state depended action, the previous shown scripting solution is better than having multiple overlayed screen item, in order to minimize the number of objects per screen.

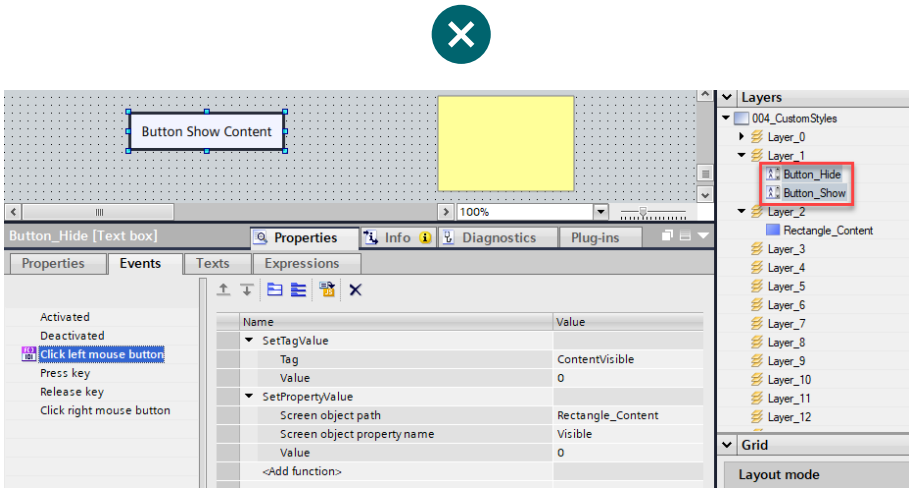


Figure 2-6 Two overlaying buttons for the show-hide-task

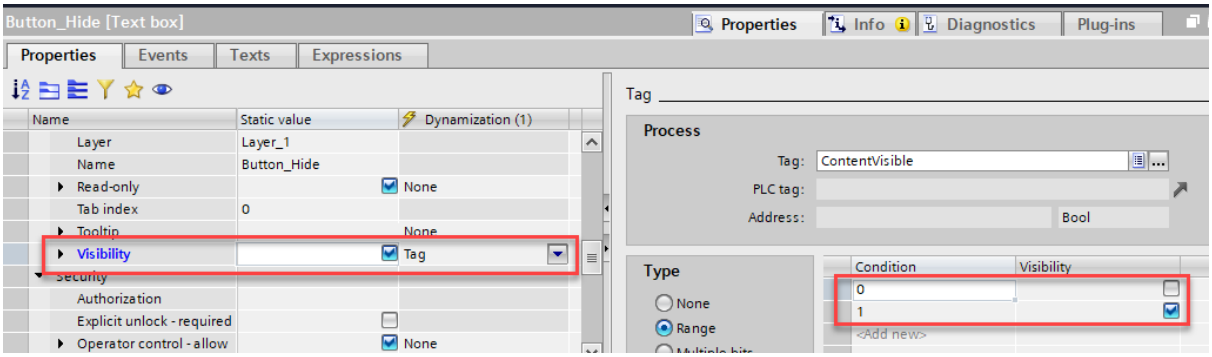


Figure 2-7 Visibility dynamization of the two buttons

The reason why the use case example applies text boxes as buttons, is explained in 3.1.1.2 Use Case: Button without implementation of the press and release event.

© Siemens 2024

2.3. Relevant sections for Screen change

In Unified as well as in other visualization systems, screen changes play a central role in the representation of your various plant screens and in the user experience. A quick screen change gives the user a good feeling of efficiency in the projected elements, but a slow or long-lasting change makes them feel uncomfortable. With a few small tricks, you can prepare the screen changes even better.

To get the most out of your engineering steps, it is important that you understand how to realize the steps in the best way. Before going into too many details about the "right" implementation, it is also important to get familiar with the procedure of how a screen loads in runtime.

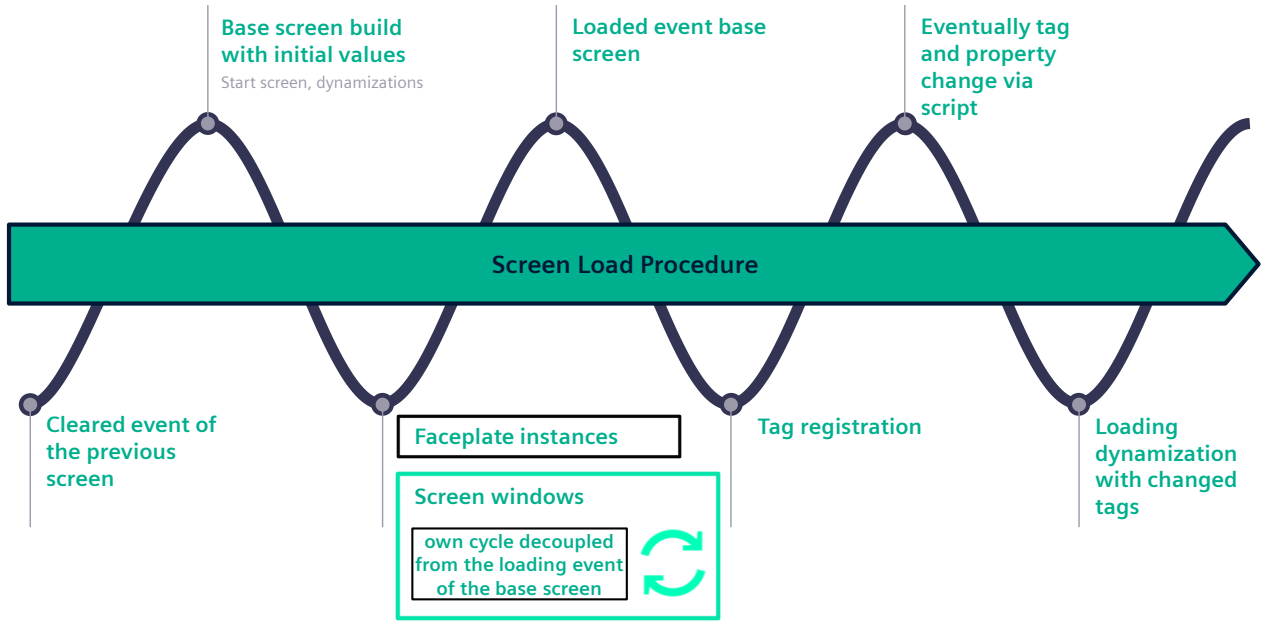


Figure 2-8 Screen load procedure

The screen load procedure of an individual screen is mostly separated in synchronous sections, meaning that one thing happens sequential after another. As the layout is often made by the composition of nested screens and faceplates, it is important to know, that for each screen window and faceplate instance there is an own decoupled cycle.

At the very beginning of a screen change, even before the new screen is loaded, the previous one gets cleared.
After this step is finished, the base screen is built and all properties, even if they are dynamized, e.g., linked to tags, are loaded with the static value for an initial load of a screen. All nested faceplates and screens in screen windows are loaded in an own decoupled cycle also first with static values.

Next the screen "loaded" event, that also can be customized in the engineering, is executed.
The tag registration happens after the initial load of the screen. When the tag registration is done, all dynamizations are considered and the screen item properties that have a linked tag in the dynamization area can change again from the static value to the value of the dynamized tag.
If tags, linked to any dynamization or any property are changed in this context again, also the screen item properties can change again. Through this order, the on-change event of a property can be executed twice during the screen load procedure. This second call of the change event is highlighted orange in the following figure (Figure 2-9).

© Siemens 2024 |

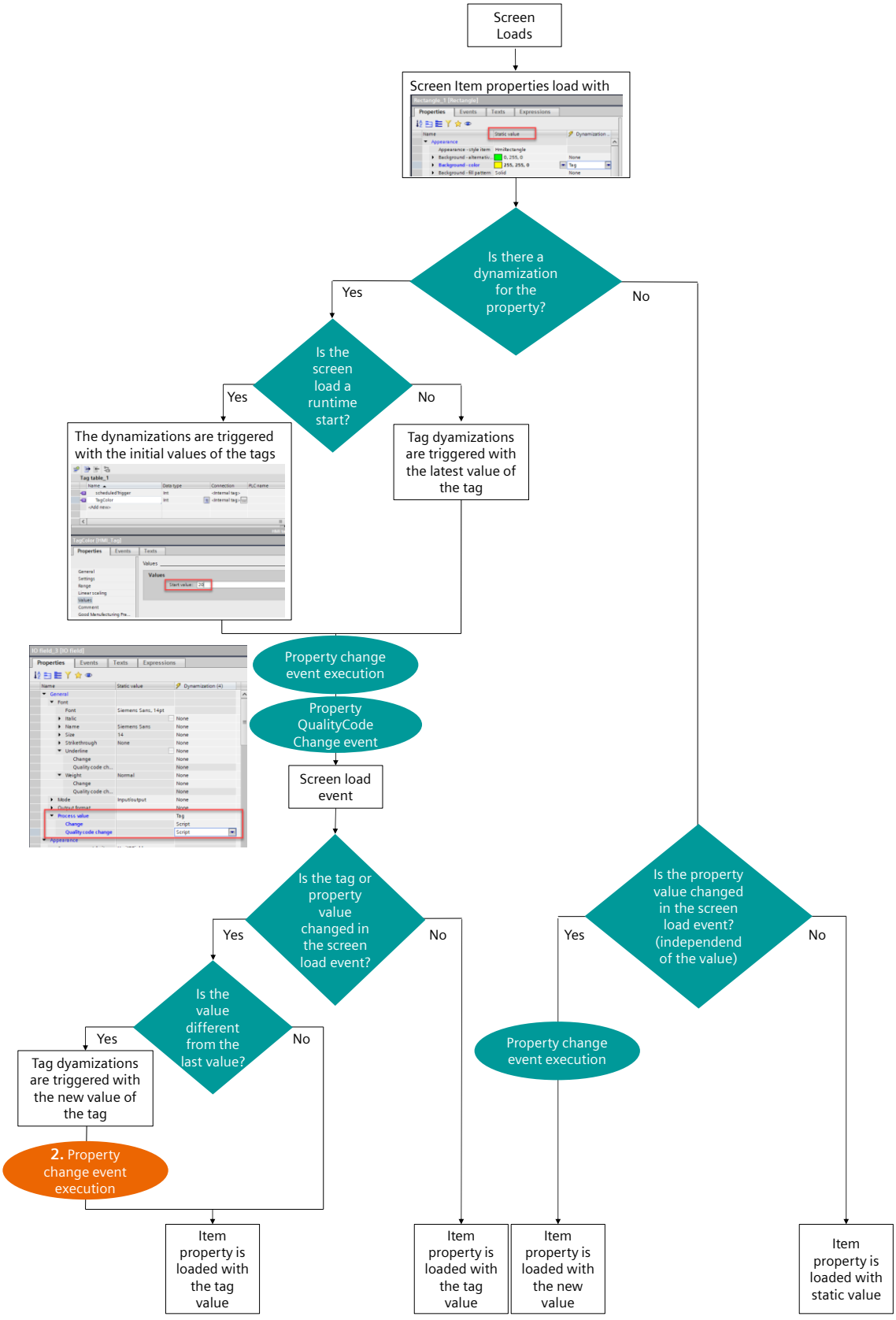


Figure 2-9 Call of the change event of screen item properties during screen load

© Siemens 2024 |

Another important aspect are the execution contexts. A context is an independent runtime environment where a specific script or task is performed. The different contexts work parallel in runtime; therefore, you can handle several scripts and tasks at the same time.

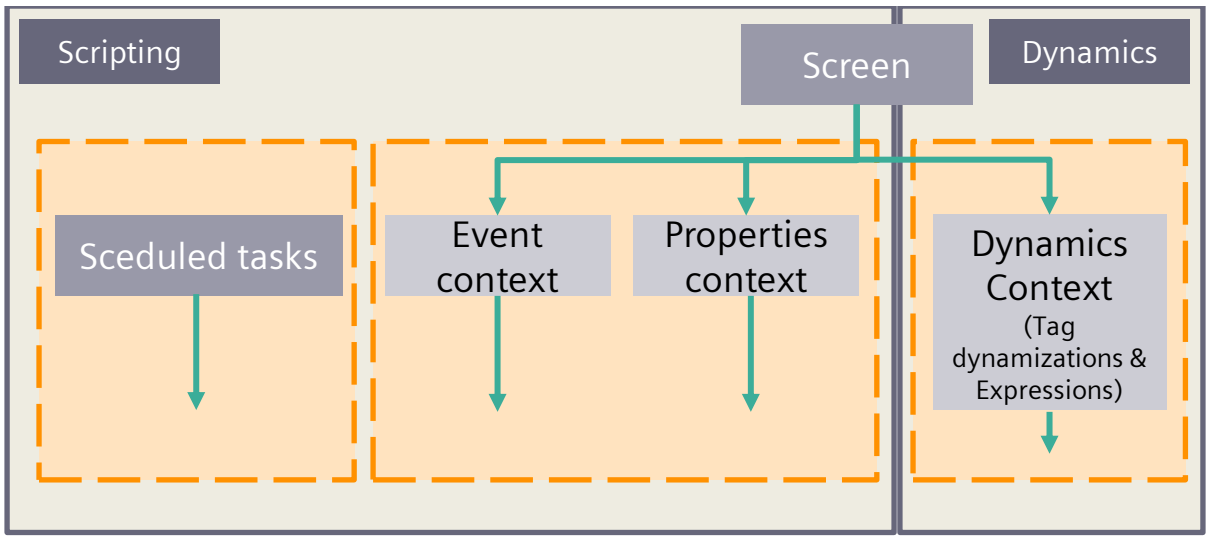


Figure 2-10 Script and dynamics execution contexts

As you can see in the figure above, the main separation is made into scripting and dynamics. This already indicates why the tag dynamization and expressions are preferred to the script dynamization as they work in parallel to all processed scripts.

For the scripting the separation is between the scheduled tasks context and screen context. Each screen has its own scripting context for scripts and system functions. The namespace (global definition area) is unique for each scripting context. The scheduled tasks and also scripts in the screen context can be triggered cyclically, tag triggered, or alarm triggered. Both is important for the load procedure, the decision if the script is implemented in the scheduled task or screen context but also which trigger type is used (see 3.4.1).

With this information it can already be defined what configurations are relevant for screen load.

Script	Is relevant	More information
Screen loaded event	✓	Figure 2-8 Screen load procedure
Script in the global definition area of screens	✓	
Scheduled tasks	-	Figure 2-10 Script and dynamics execution contexts
Script modules (referenced)	✓	
Scripts in „Click left mouse button" events	-	
Property on change event scripts	✓	Figure 2-9 Call of the change event of screen item properties during screen load
Dynamization scripts	✓	

Table 2-1 Overview of screen load relevant scripts

© Siemens 2024 |

Object / Implementation	Is relevant	More information
Faceplate interface	✓	3.3 Use of Faceplates
Expressions	✓	
Dynamizations	✓	
Number of used HMI tags	✓	See system limits in Unified manual \9\
Number of screen elements (items, faceplate-instances, screen windows,...)	✓	See system limits in Unified manual \9\
Type of screen element	✓	3.1.1 Screen object selection

Table 2-2 Overview of screen load relevant objects

As you can see, there are a lot of things that affect the procedure of screen change at the runtime device. To give you all the detailed information on the single aspects, we're focusing even more on the following areas. You will also receive a description and best practice how to implement the features in the most efficient way. If a project needs to be analyzed there is a guideline provided in chapter 4. But first this document describes the best practices for screen engineering in more detail.

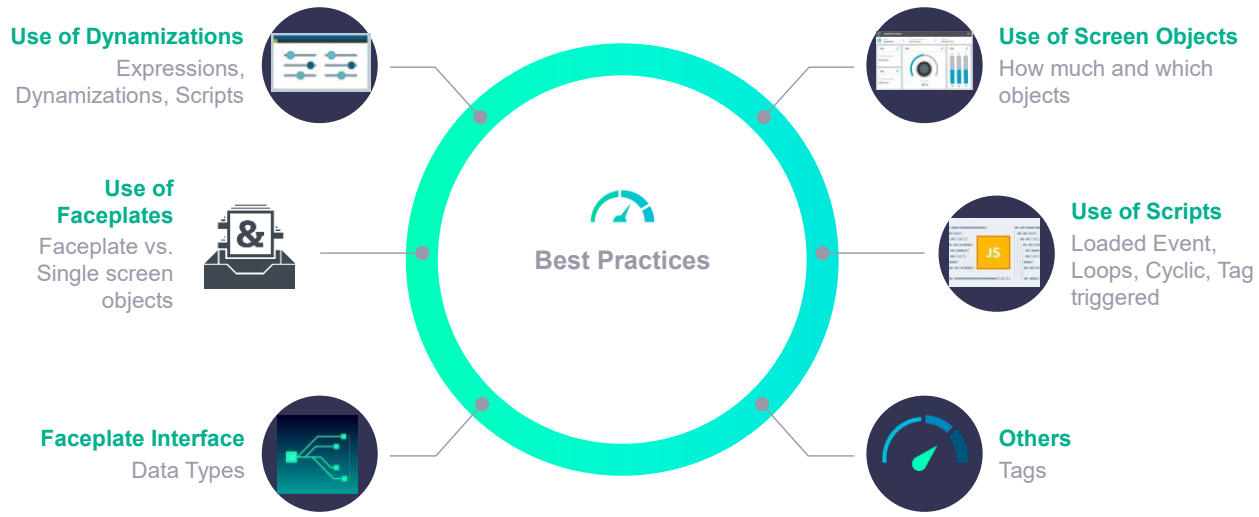


Figure 2-11 Best Practices Topics Overview

© Siemens 2024

3. Best Practices of Screen Engineering

In this section lots of recommendations for the implementation of an HMI are described. Therefore, we will first show you on which aspects you can have an influence and which possibilities you have. Then we will give you illustrated details with the help of an exemplary use case. We split the sections in the use of screen objects, dynamizations, faceplates, scripts and others.

NOTE To support the comprehension of these best practices, the general screen load procedure is described in the chapter 2.3.

3.1. Use of screen objects

In the following figure, the available items of the Unified Toolbox for a PC-RT can be seen (The content for a Unified panel can differ). In general, there are five relevant sections that separate them: The Basic objects, Elements, Controls, My controls and the Dynamic widgets.

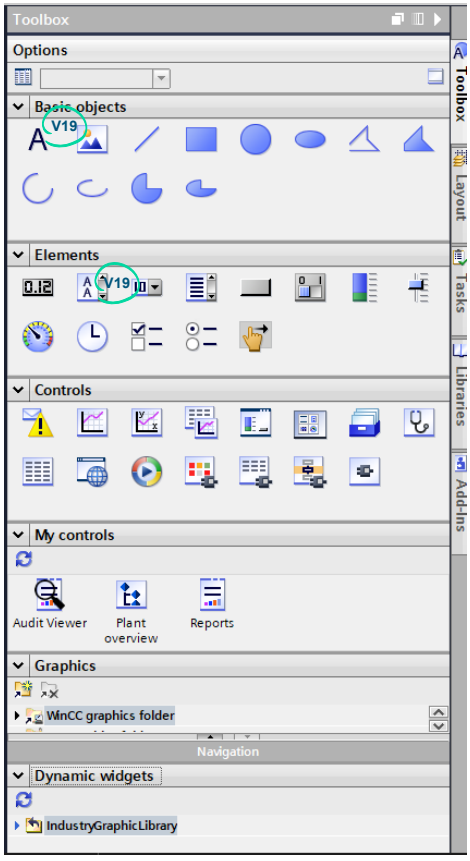


Figure 3-1 Unified Toolbox V19 for PC-RT

In general, all recommendations of this section about screen objects follow on rule.

© Siemens 2024

Screen object usage

Minimize the number of objects on the screen and select the ones with the smallest necessary footprint

Before we go on, it is important to mention that each object has a footprint. With footprint the rendering effort is described. A small footprint is associated with a small number of properties and the low complexity of the screen item, e.g., a rectangle. Objects with a higher footprint often contain a lot of properties as the controls do have. Besides the fixed properties, also the customization through dynamizations can increase the rendering effort of a screen item and therefore influence the loading procedure of the whole screen.

© Siemens 2024 |

3.1.1. Screen object selection

First, when placing a new object on the screen, select the one with the smallest footprint and just the necessary functionality. In general, you can say that the complexity raises from Basic objects to Controls, as the order in the toolbox. The following illustration shows all screen objects in the order of the required rendering effort. Please note that the rendering effort shown in this illustration has no time equivalent and only displays the relation. The actual rendering effort depends also on the configuration (dynamizations and connected data).

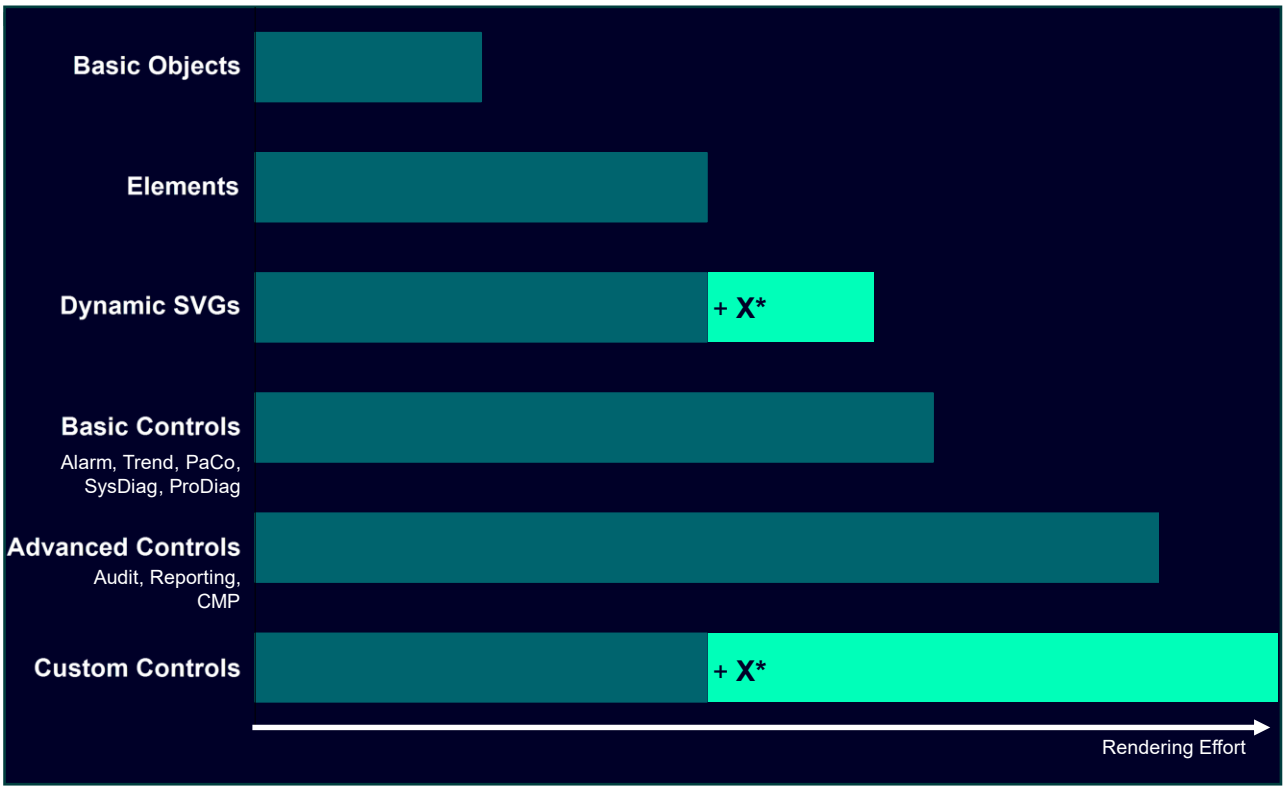


Figure 3-2 Relative comparison of the screen items' rendering effort

In the groups themselves there are differences as well and the most important thing to understand is that each screen object has an individual rendering effort (footprint). Before we go on, make sure that you know the toolbox elements and even the adaptions in V19. The text and text box adaption is shown in the figure below.

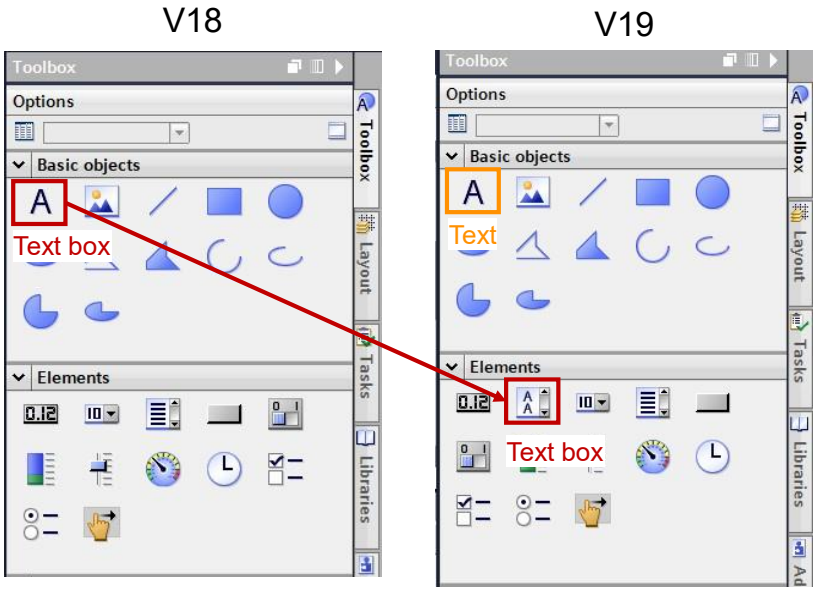


Figure 3-3 Toolbox V18 vs Toolbox V19

© Siemens 2024 |

NOTE V18 The Text box from the basic objects has moved to the elements section for V19. The same looking item in the basic objects of the V19 toolbox is now "only" a text, comparable to a simple label. So, if the engineering system is a V18 version, the rendering effort of the Text box is also higher than for the basic objects and comparable with the items from the elements section

Sometimes, there is more than one possibility to display a functionality with different objects. Even if two objects look the same when configured on screen, they can have different influence on loading time of the whole screen. In the following some examples are shown.

Even if the following examples do not cover a complete real use case, keep these points in mind when configuring a new screen.

3.1.1.1. Use Case: Colored square box

Description

This use case is about the configuration of a colored square box as you can see in the figure below. Possible solutions can be a rectangle or a Text box without text assignments because they can look the same on a screen.

Solution

Instead of using a Text box, the rectangle element is the better choice due to its smaller rendering effort.

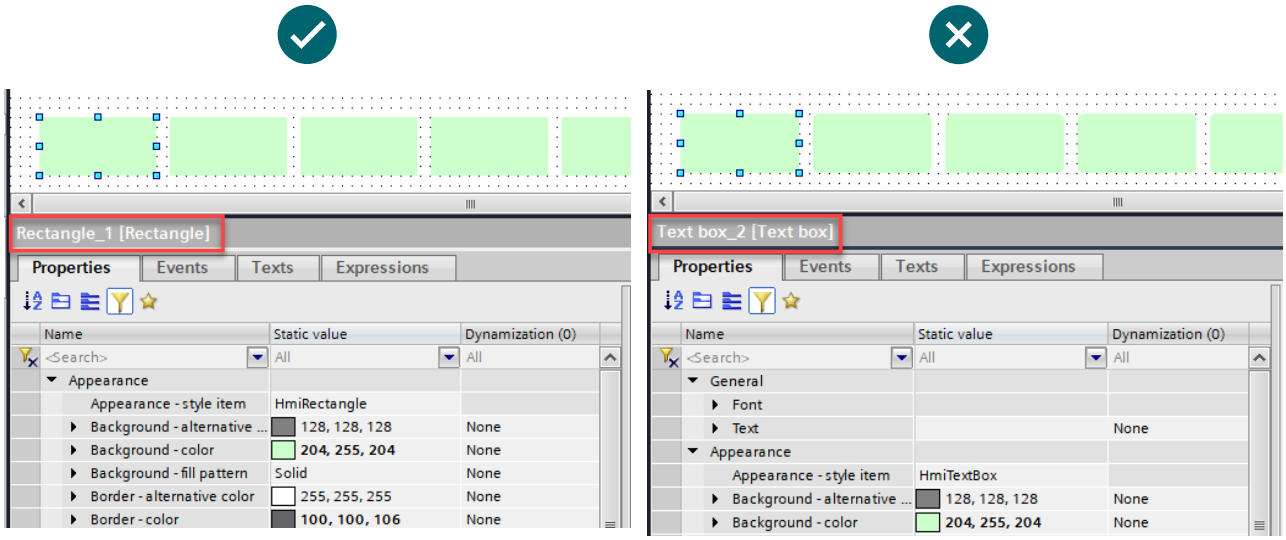


Figure 3-4 Rectangle vs. Text box

Screen object selection

Always use the screen object with the smallest footprint and the necessary functionality

© Siemens 2024

3.1.1.2. Use Case: Button without implementation of the press and release event

Description

This use case is about the configuration of a Button that only uses the “Click left mouse button” event and without implementation of the press or release event. Possible solutions can be a Text box, a Text (V19) or a Button.

Solution

In terms of rendering effort, the text has the smallest footprint, followed by the text box and the button. Therefore, use a Text box if you need a background color, otherwise a Text if they fulfil your requirements. Also, these items have on mouse click events. Only use a button when the press and release events are necessary.

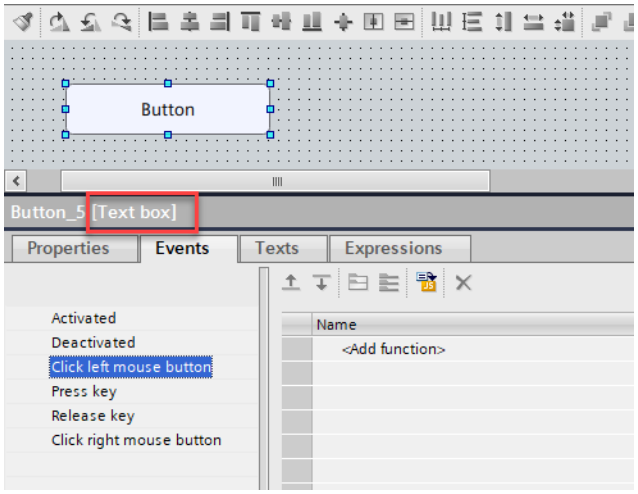


Figure 3-5: Text box as a button

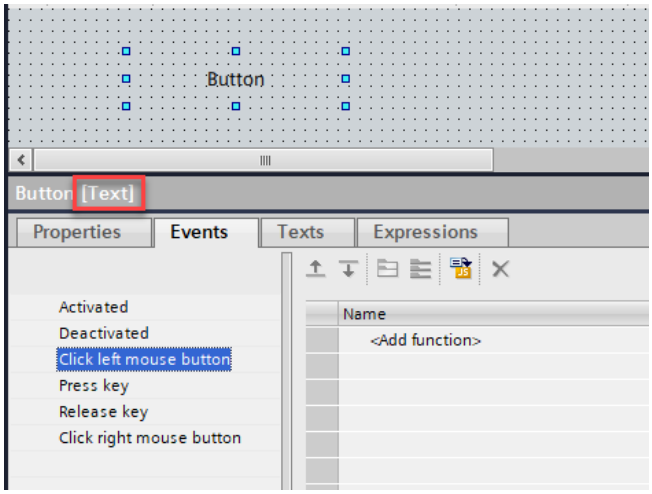


Figure 3-6: Text as a button

© Siemens 2024 |

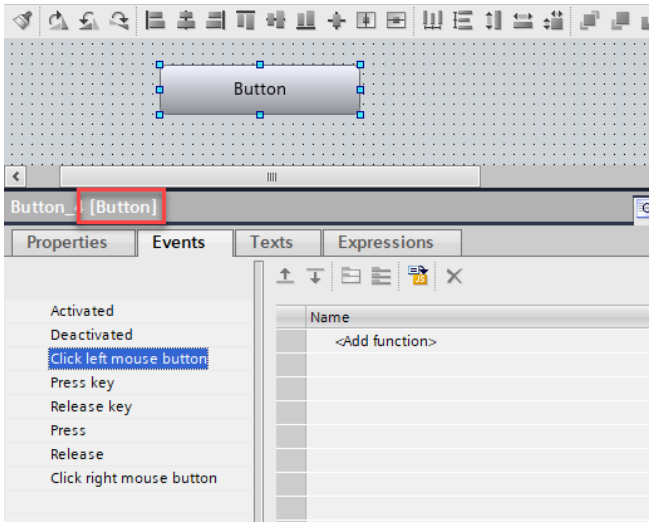


Figure 3-7: Button as a button

© Siemens 2024 |

3.1.1.3. Use Case: Simple Text label V19

Description

For text element of the screen a Text box or a Text can be used.

Solution

Whenever possible, especially for simple labels, use a Text instead of a Text box. The following table shows the different feature sets of a Text compared to a Text box. Compared to the text box, the reduced functional scope of the text causes the lower food print and rendering effort at the runtime device.

Property	Text	Text box
Colors	Foreground	Foreground, background, border, alternative colors
Border	X	Color, Width
Format – Spacing	X	✓
Format – Text trimming	X	✓
Format – Text break	X	✓
Miscellaneous – Connection quality	X	✓
Miscellaneous – Read only	X	✓
Parameter Field	✓	✓
Editable / Input Text (in RT)	X	✓
Copy and paste (in RT)	X	✓

Table 3-1 Comparison of a Text and Text box screen item

© Siemens 2024 |

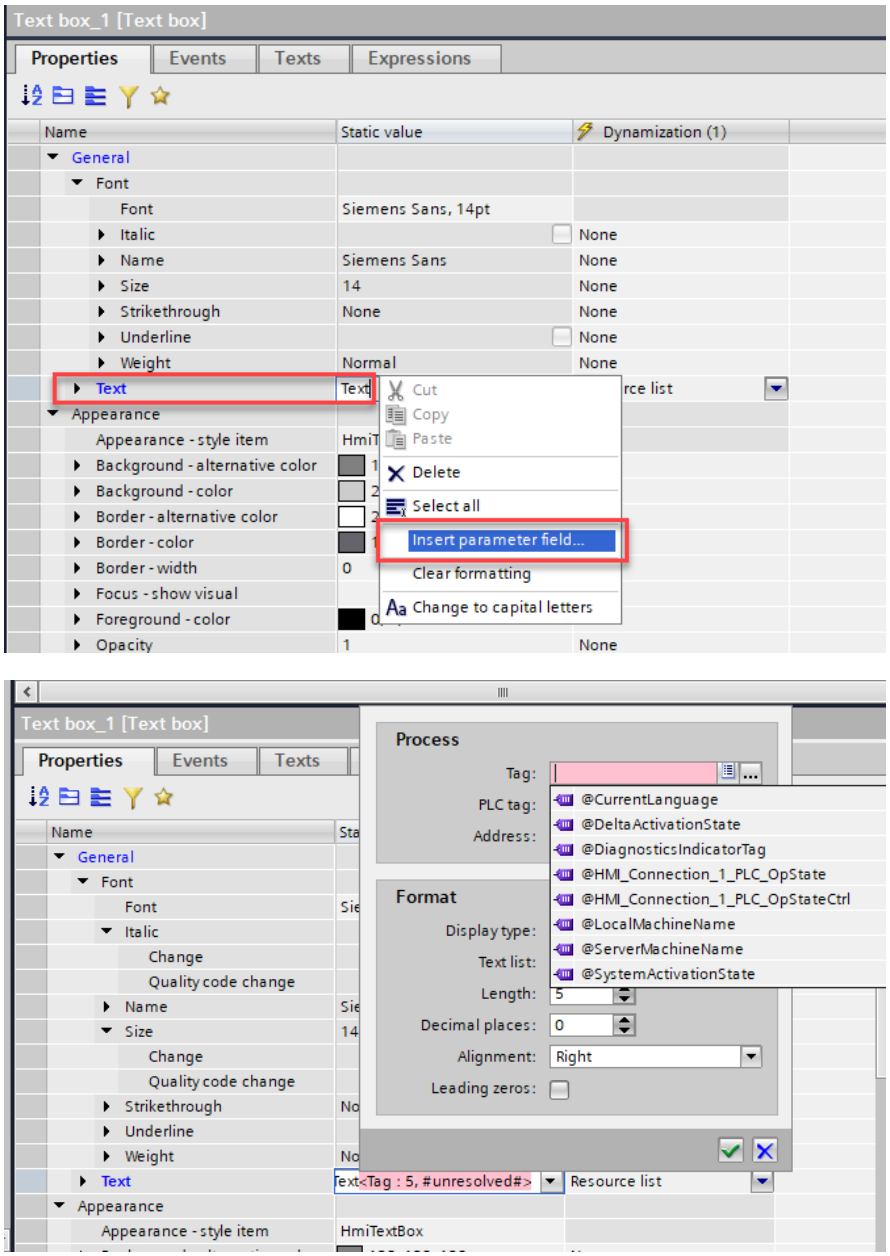


Figure 3-8 Text property parameter field

3.1.1.4. Use Case: Colored Screen Background

Description

In some cases, the default background color of a screen needs to be adapted.

Solution

Instead of adding an additional rectangle to the screen and using its background color as the screen background, change the color of the screen element directly. This is also relevant for faceplates.

© Siemens 2024

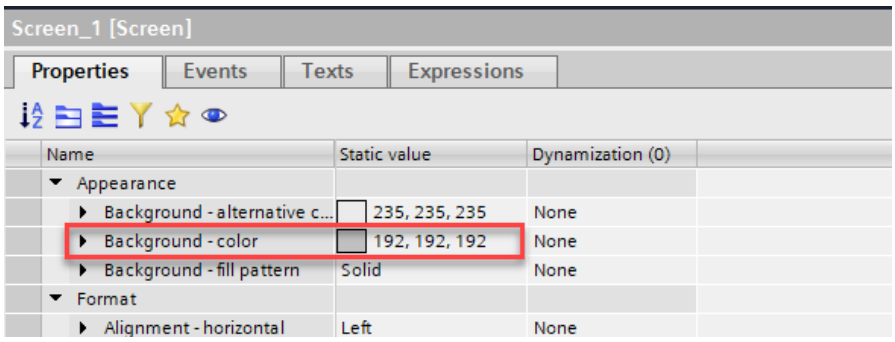


Figure 3-9 Background color configuration of a screen element

3.1.1.5. Use Case: Custom Styles V19

Description

It is common that a corporate layout also comes with a corporate color palette and designs. Through the object properties the screen items can be adapted to this layout and design. But also, the overall appearance of objects like buttons can be part of the corporate design or even the visualization of more complex objects like sliders.

Solution

It is always preferable to use the Unified screen objects with their intended function. The reproduction of user-defined designs of an item with the help of several screen items should be avoided, as this leads to a higher number of screen objects and usually to additional unused functions of these auxiliary items. Also, the use of faceplates as a standardization method for corporate designed objects create an unnecessary overload. Use the SIMATIC WinCC Unified Corporate Designer to create these designed objects and also colors as part of your own style and use this style in your project.

Inside the corporate designer a new style can be created already based on existing styles. Then you can either change the style of existing objects or also create completely new objects. These styles can also be linked to color palettes and fonts.

When the style design is finished, the style needs to be exported and the generated file must be copied to the project directory (Userfiles > Styles). After these steps the style can be selected in the runtime settings.

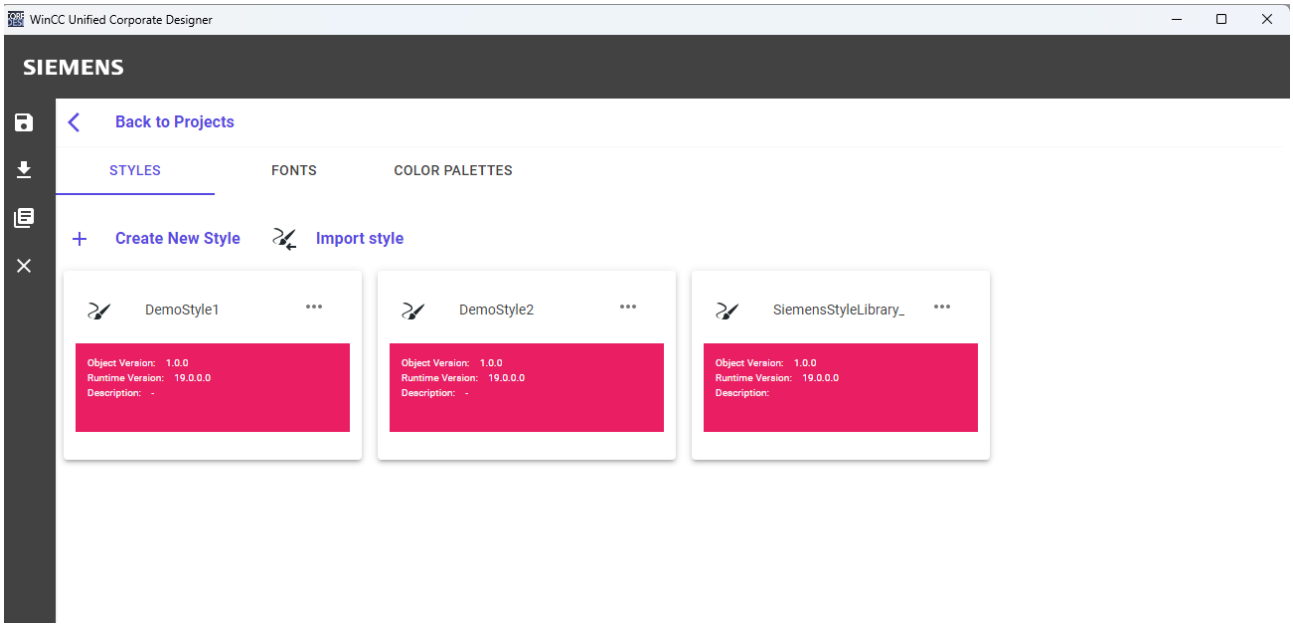


Figure 3-10 Unified Corporate Designer Project View

© Siemens 2024 |

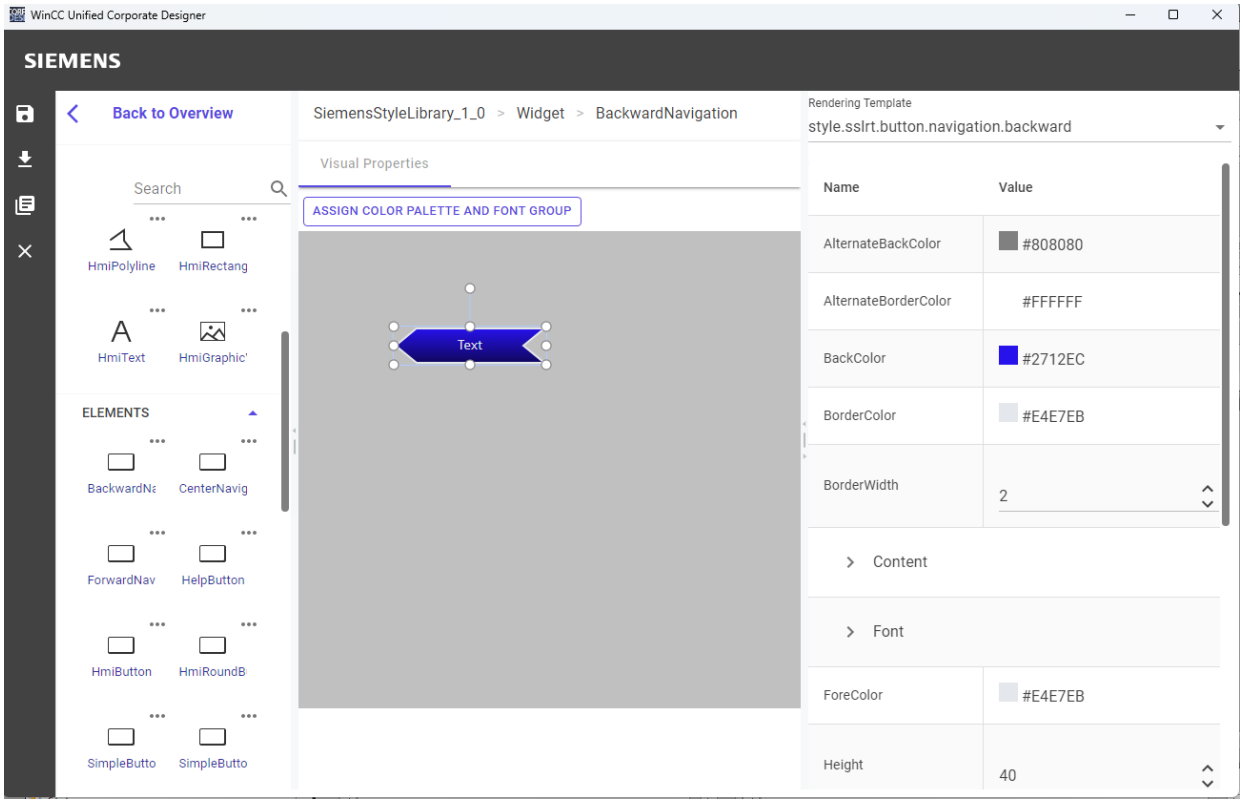
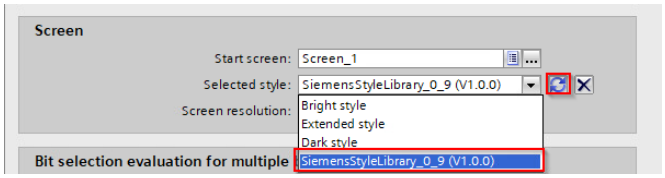


Figure 3-11 Unified Corporate Designer Edit View

The style that is selected in the runtime settings of the device, can still be changed on demand in runtime. Also, with the new custom styles.



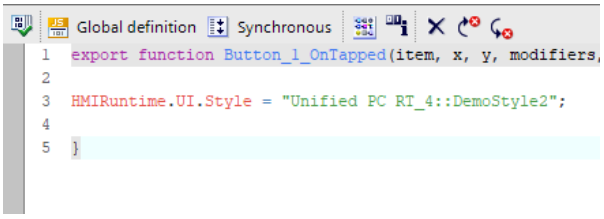


Figure 3-12 Style selection for the Unified Runtime

© Siemens 2024 |

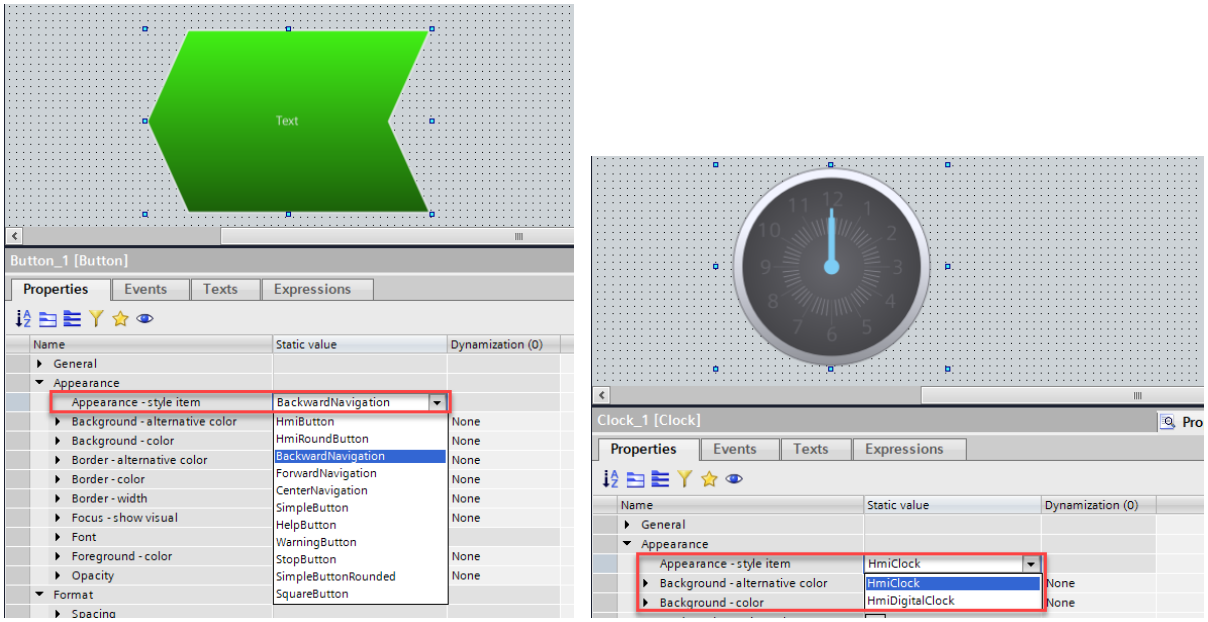


Figure 3-13 Change the style item of system and customized screen objects in engineering

Custom Style

Define a custom style instead of using faceplates for single object configurations

© Siemens 2024 |

3.1.2. Tidy up the screen / observe system limits

As you've already seen, there are elements that have more rendering effort and elements that require less. In any case, every element on a screen affects the loading time and there are also system limits defined that must be observed. To achieve the shortest screen loading time, especially shortly before a project is used as productive, all objects that were generated only for the implementing phase for debugging or other supportive reasons, should be deleted again.

- Delete unused objects that are in the background or not visible
- Delete unused objects that are out of the range of the screen. There is also a task card in the layout section available, that indicates the objects out of range.

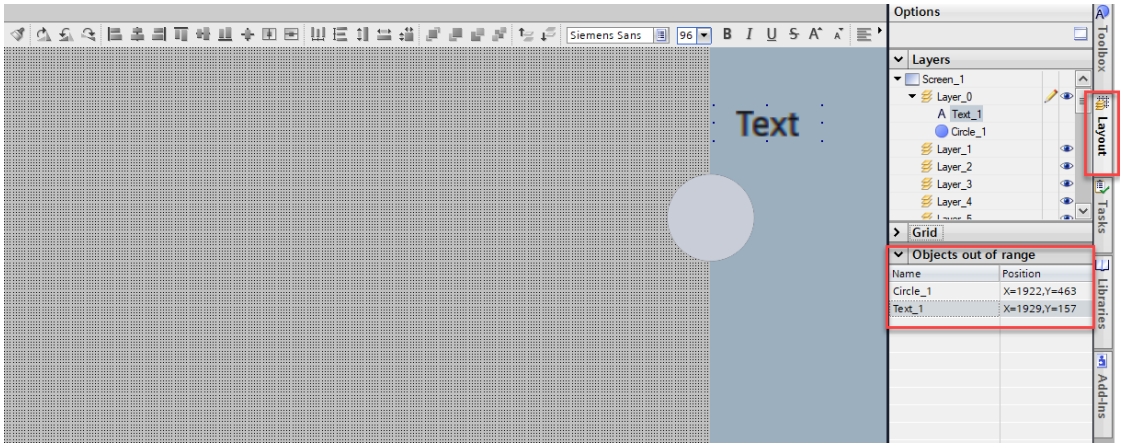


Figure 3-14 Task card objects out of range

System limits

Delete unused and invisible objects to achieve a lower loading effort

© Siemens 2024 |

3.1.3. Usage of screen windows

Screen windows are often used to create a screen layout and to separate individual content from content that is apparent for more views and does not need to be loaded for each screen change (e.g., Header). Also, for pop up screens they are applied. To clarify when to use a screen window as a pop und when to configure it already in runtime, follow the flow chart below. The recommendation is only in terms of a good engineering. Keep in mind that screen windows in general have a different behavior as by system function OpenScreenInPopUp generated popup screen windows regarding zoom, scroll, position, screen layer and live time. So, the specific use case can require explicit one of the implementations.

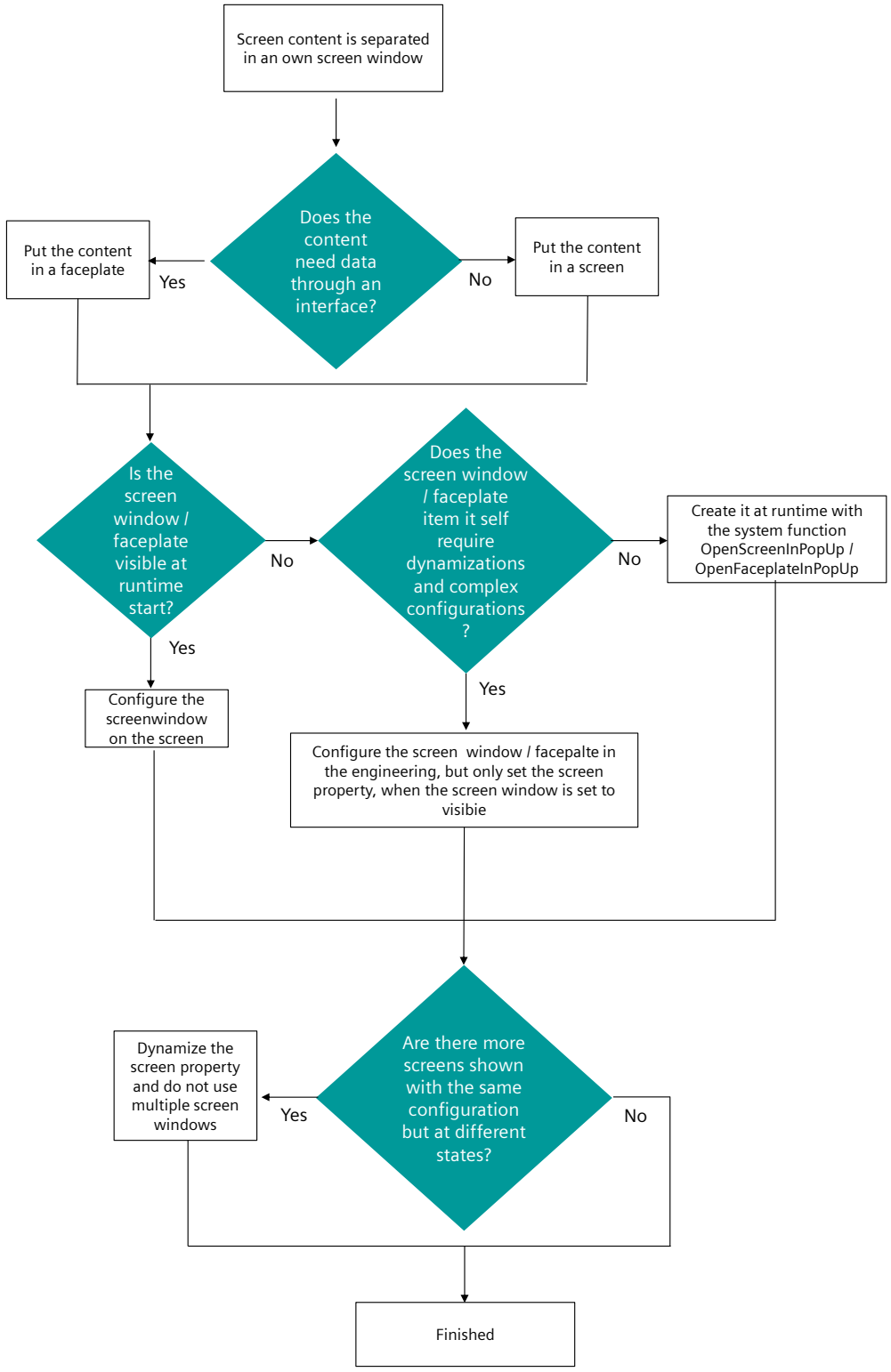


Figure 3-15 Configuration of screen windows

© Siemens 2024 |

3.1.4. Screen object visibility

If the visibility of a screen item is dynamized, set the static value to false, especially when it can be assumed that the used dynamization, e.g., tag dynamization causes an invisible object during screen load.

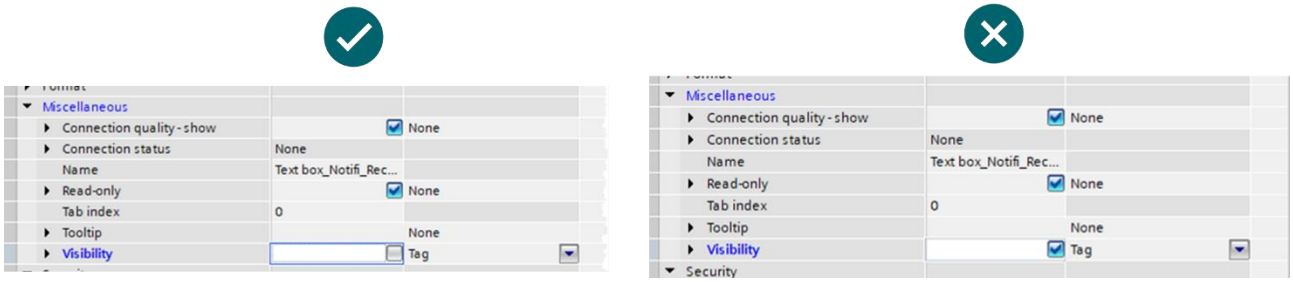


Figure 3-16 Static value for Visibility if the item is assumed to be invisible at screen load

3.1.4.1. Use Case: Visibility dynamization of multiple screen objects by the same condition

Description

The visibility of more than one screen items needs to be changed in runtime by the same condition.

Solution

If only the visibility is changed depending on the same condition, move all objects in one layer and dynamize the runtime visibility of the layer. The visibility of the layer can be changed also in runtime by scripting.

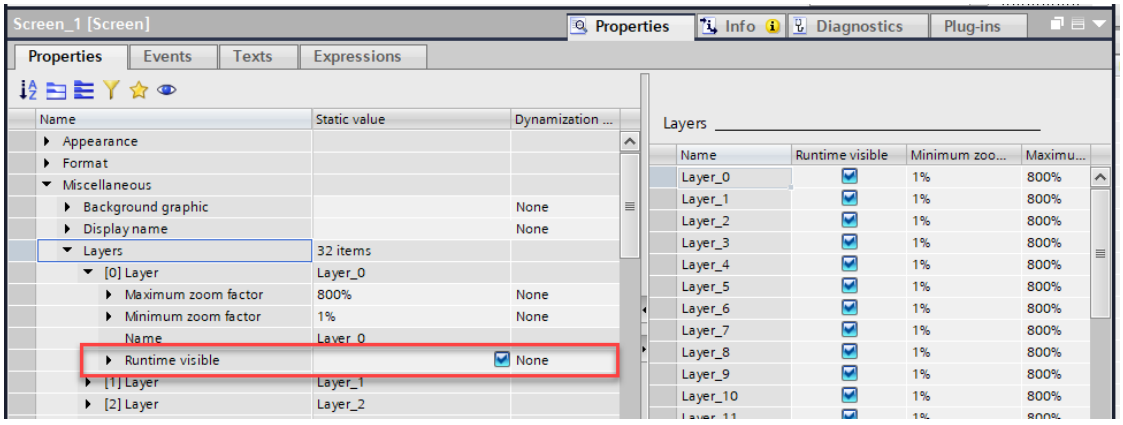


Figure 3-17 Runtime visibility of a screen layer

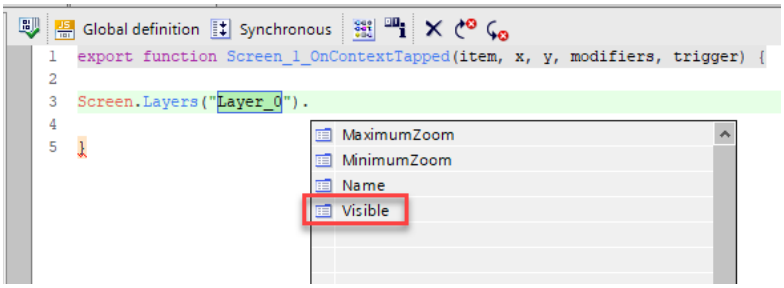


Figure 3-18 Access layer visibility in runtime

If there are more properties that have the same behavior or condition (e.g., background color, authorization…), group the objects and configure the property change for the whole group at once.

© Siemens 2024

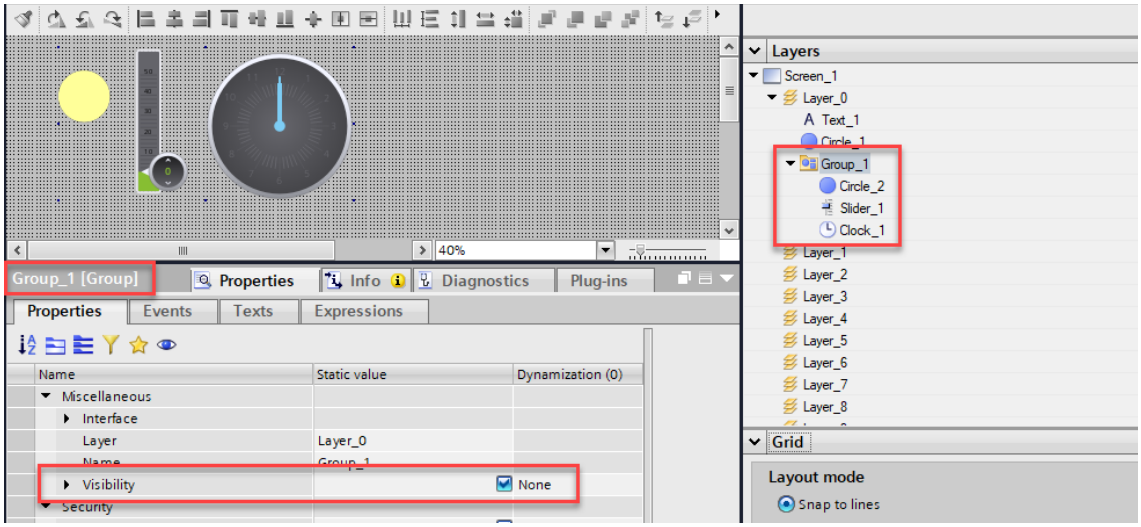


Figure 3-19 Visibility dynamization of a group of screen items

Visibility of screen items

If the visibility of more than one item is dynamized by the same condition, put them together in a group or layer and dynamize its visibility.

© Siemens 2024 |

3.1.5. Unified Controls

Unified controls make it much easier to implement common use cases such as alarming, reporting and trend control. Nevertheless, there are also aspects in this area that should be considered during use.

3.1.5.1. Use Case: Different settings in Runtime for Unified Controls

Description

There are lot of configurations that can be done in the engineering for the Unified Controls. Sometimes a Control is needed in Runtime with different settings. Due to the very limited possibility of dynamizing the controls in the former WinCC Basic, Comfort and Advanced projects, it was common practice to place several statically parameterized controls in the screens of those projects and to make them visible alternately at runtime.
WinCC Unified provides very extensively configurable and dynamizable controls, so that the requirement to make various settings available at runtime can be met by implementing and controlling a single control.

Solution

Change the properties in runtime and do not configure multiple controls or even change the screen to realize different settings. Therefore, use the system function “SetPropertyValue” or the object model “Screen.Items(« Itemname »)”. The name of the properties can be easily copied in the engineering from the properties tab.

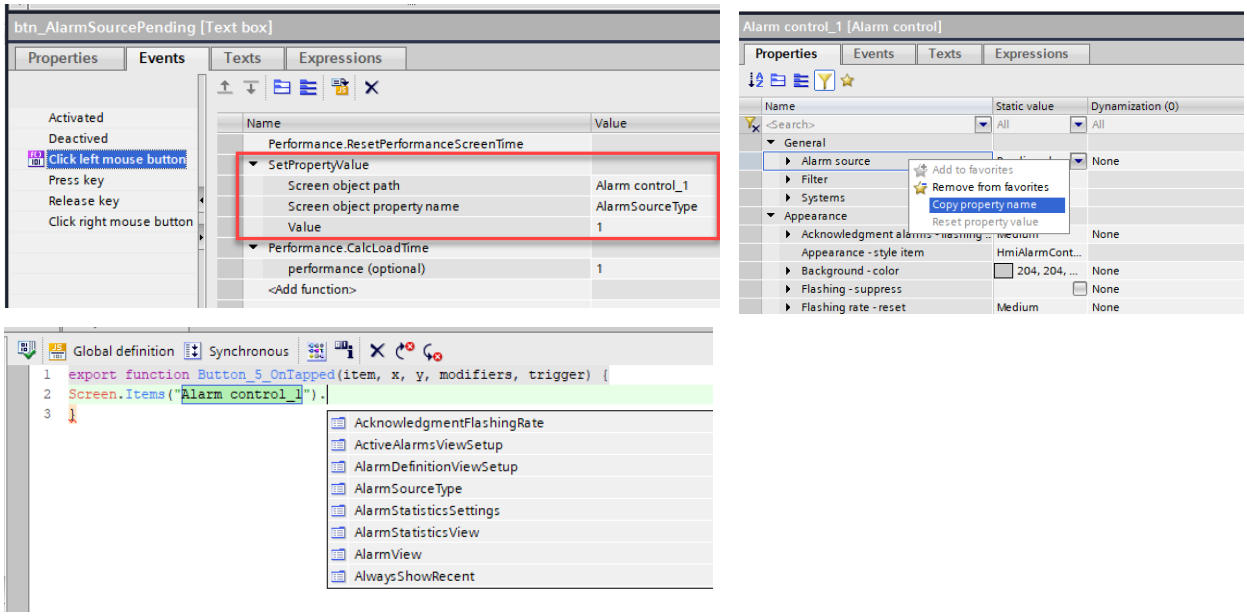


Figure 3-20 Accessing Unified control properties

All accessible properties can be found in the WinCC Unified object model directly in the TIA help or on the SIMATIC HMI WinCC Unified Engineering V18 System Manual on SIOS.

Unified Controls

Reconfigure the control in runtime, when different settings are needed, instead of having multiple controls.

© Siemens 2024

3.1.5.2. Use Case: Alarm control

Description

For accessing alarms in general an alarm view is provided. This control comes with a lot of functionality as well in the engineering to configure but also in runtime. Additionally, there are also a lot of system function, to access the same functionality by scripting, without the Unified control.

Solution

As we in general recommend designing a screen layout with screen windows (see 3.1.3 and Figure 2-2), there are screens that change the content and screens that remain. The alarm control belongs to the screen items with a huge footprint (see 3.1.1 and Figure 3-2) therefore it is recommended to put it in screen windows, that are not reloaded that often like e.g., the header. Since these screen windows like the header are not that big, you can configure the alarm view as an alarm line with the following steps:

1. Use the simplified appearance style

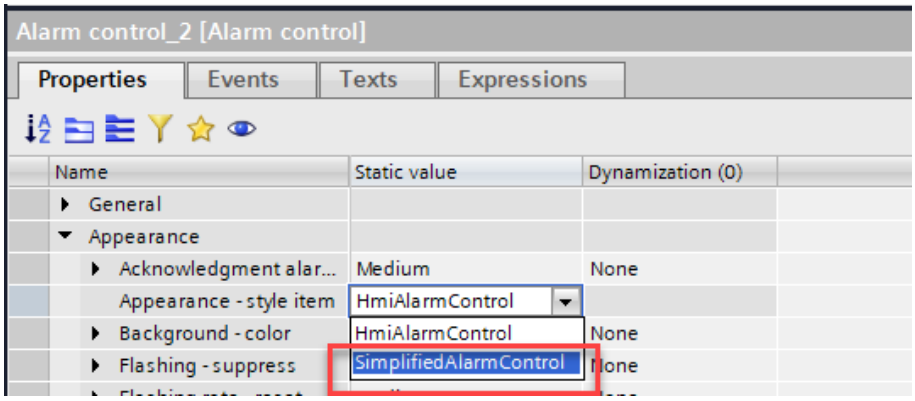


Figure 3-21 Simplified alarm control style item

2. Set the window settings to always on top

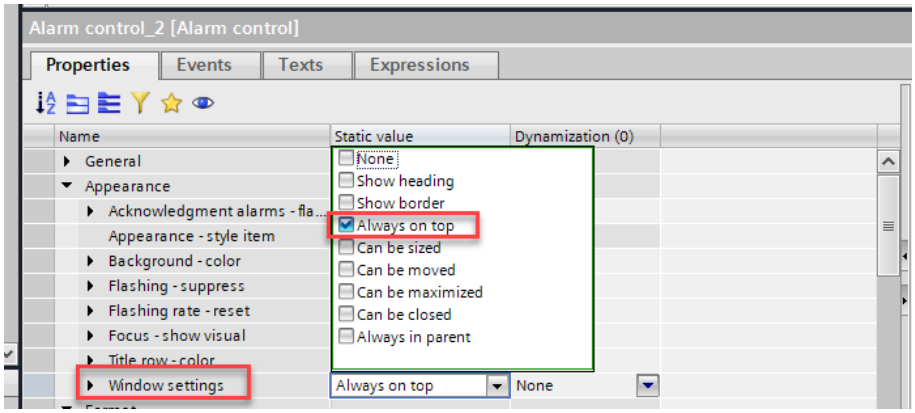


Figure 3-22 Alarm control window settings

3. Set sorting direction to ascending

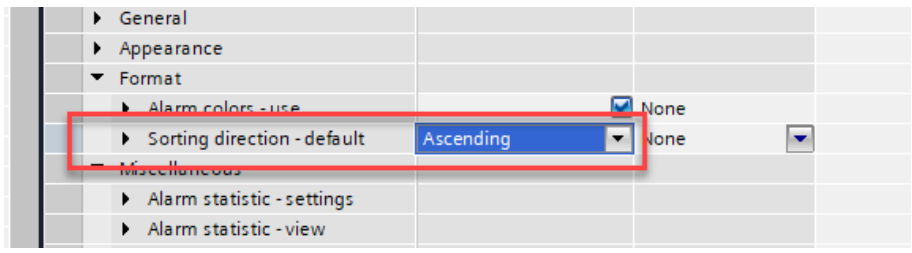


Figure 3-23 Alarm control sorting direction

© Siemens 2024

4. Go to Miscellaneous > Alarm view > Header-settings and change column header to none and row header to none

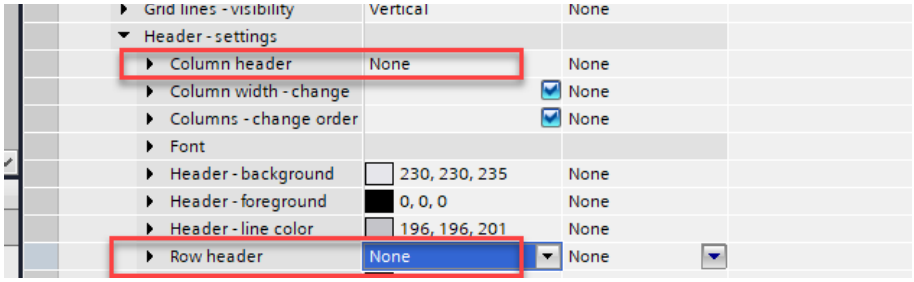


Figure 3-24 Alarm control column and row header

5. Go to Miscellaneous > Alarm view and collapse the scroll bars

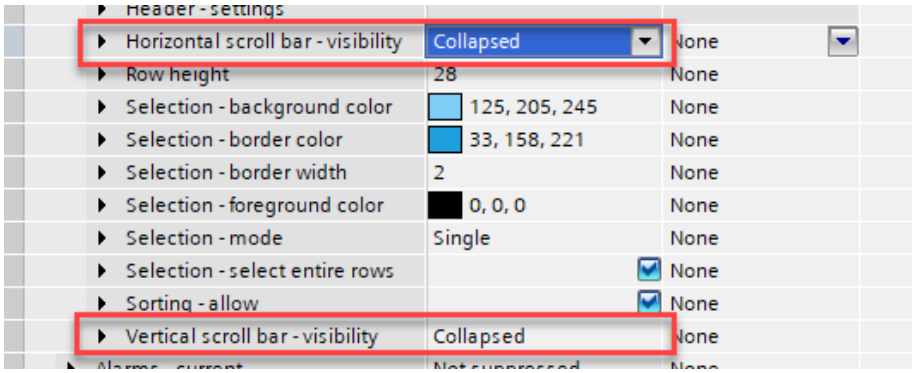


Figure 3-25 Alarm control horizontal and vertical scroll bars

6. Adapt the size to one or only a view lines

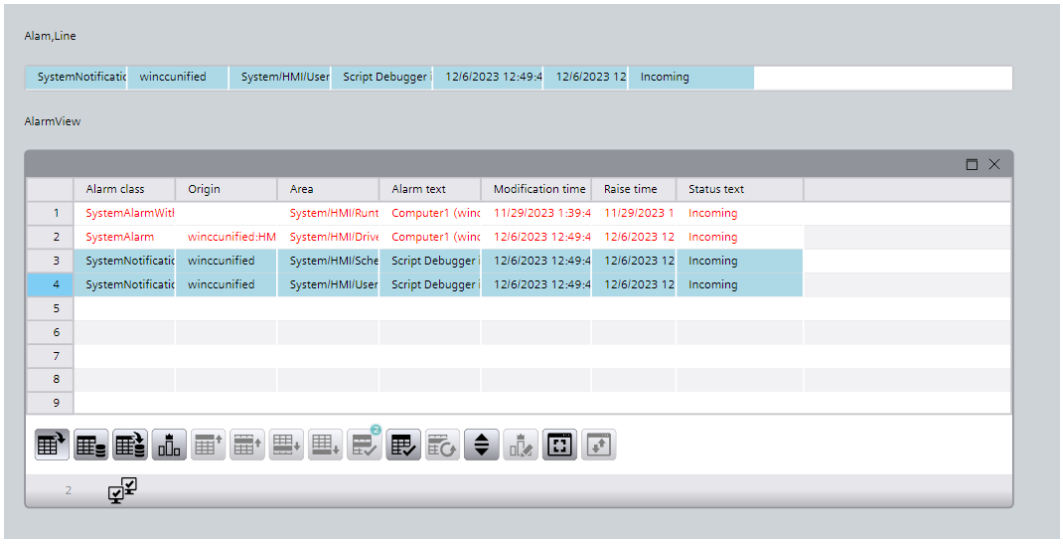


Figure 3-26 Alarm line vs Alarm View

If the alarm view with the whole functionality is necessary, show it as a pop up on demand.

© Siemens 2024 |

3.1.6. Graphics and SVGs

Additionally, to all the textual information, an HMI usually also contains images. These can be small icons, bigger images, or animated graphics / dynamic SVGs. Depending on the use case there are also recommendations on how to use images.

Image format:

- **JPG** (joint photographic experts group) are recommended for UCPs as the format reduces the size of large files. So there is also a loss of quality through the compression, but the files can also be decoded faster. When a graphic is needed in a high resolution, a JPG can also be used in a high quality. JPG does not automatically mean less quality.
- **PNG** (portable network graphic) is a raster graphics format with lossless data compression. Therefore it is recommended when transparency is required or when exact pixel accuracy is important.
- **SVG** (scalable vector graphics) is used for displaying two-dimensional graphics, diagrams and illustrations on websites. As a vector format, SVG images can be scaled to different sizes without affecting the resolution. Therefore, this format is recommended when high quality zooming is required or dynamic SVGs are used. In other cases, it is better to convert the file in the size used to a PNG or, if no transparency is required, to a JPG, whereby the largest size used for multiple use should be selected.

Graphics

Regardless of the graphic format, the file size should not be larger than the maximum size /resolution that is needed for this HMI

Graphics Format

The recommended order for selecting the graphic format is

1. JPG
2. PNG
3. SVG

If there are further requirements for the graphic, still the SVG can be the most suitable. But select it carefully!

3.1.6.1. Use Case: Visualize Patterns and composed objects

Description

Some objects or patterns can be created by multiple objects. Furthermore, these combined objects might be reused multiple times on the screen.

Solution

Do not create pattern by the use of single items but combine them to one image. This reduces the number of objects on the screen. Even if the objects are basic like rectangles, the combination to one object reduces the rendering effort. When dynamizations are necessary create a custom dynamic SVG (see next use case). This also reduces the number of object containers on the screen and helps to comply with the system limits.

© Siemens 2024 |

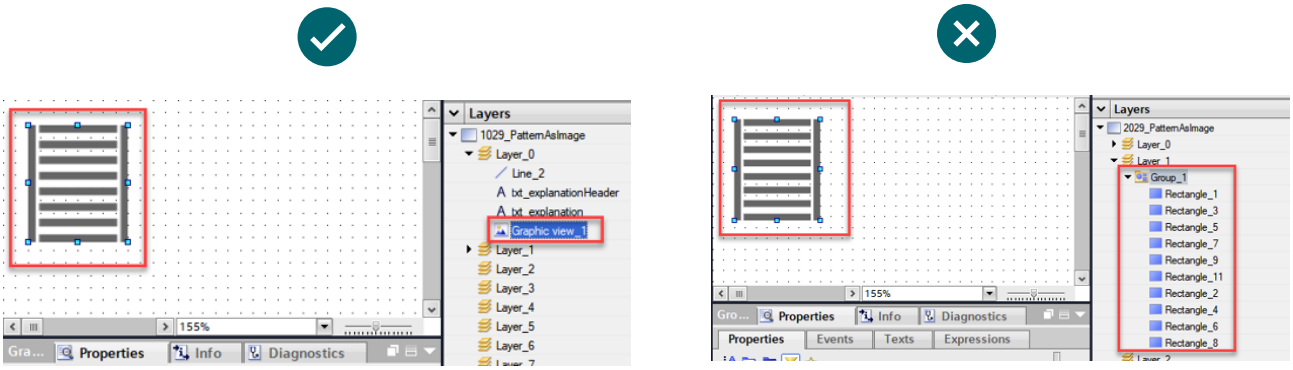


Figure 3-27 Pattern as SVG example

Combined objects

Reduce the number of elements on the screen, by creating SVGs and dynamic SVGs of combined objects

Also use graphics in their original resolution and only as large as it is necessary.

3.1.6.2. Use Case: Composed Objects with dynamizaton → dynamic SVG

Description

Some illustrations with a dynamic visualization can be either created by the composition of multiple dynamized screen objects or directly by a dynamic SVG.

Solution

As already described in the previous use case, to minimize the number of screen items per screen, composed objects should be combined to one object container. There are dynamic SVGs available for a wide range of standard components in the IndustryGraphicLibrary. Before building your own component with several screen items, check if there is already a solution in the graphic library. Through the interface some properties can be configured and changed in runtime dynamically. More information about the usage of the dynamic widgets can be found in the Unified Manual.

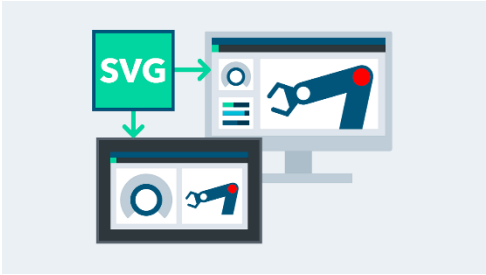


Figure 3-28 Industry Graphic Library

© Siemens 2024

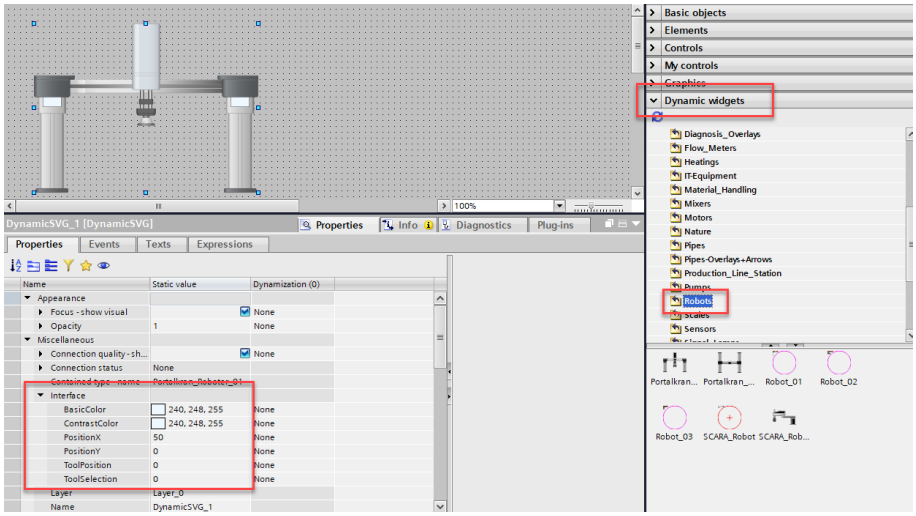


Figure 3-29 Robot dynamic SVG from Siemens Graphic Library

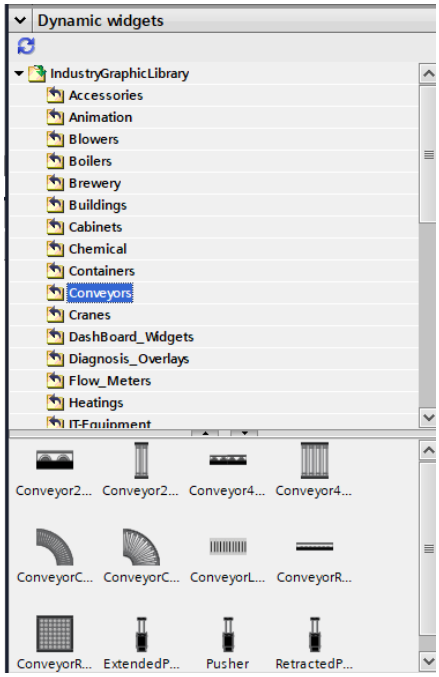


Figure 3-30 Dynamic Widgets Conveyor examples

© Siemens 2024 |

3.2. Use of dynamizations

In this document different types of dynamizations are mentioned.

A tag dynamization is shown in the following figure. It can have types of configurations. You can take “None” to directly transfer the tag value to the property value. With the “Range” type, a condition can be specified for several tag value ranges and linked to a property value. The “Multiple bits” type makes it possible to change the property depending on various single bit positions of the dynamization tag. And finally, the “Single bit” type is limited to one bit of the dynamization tag, to specify conditions for the property.

Every time the tag value changes, the property value is adapted.

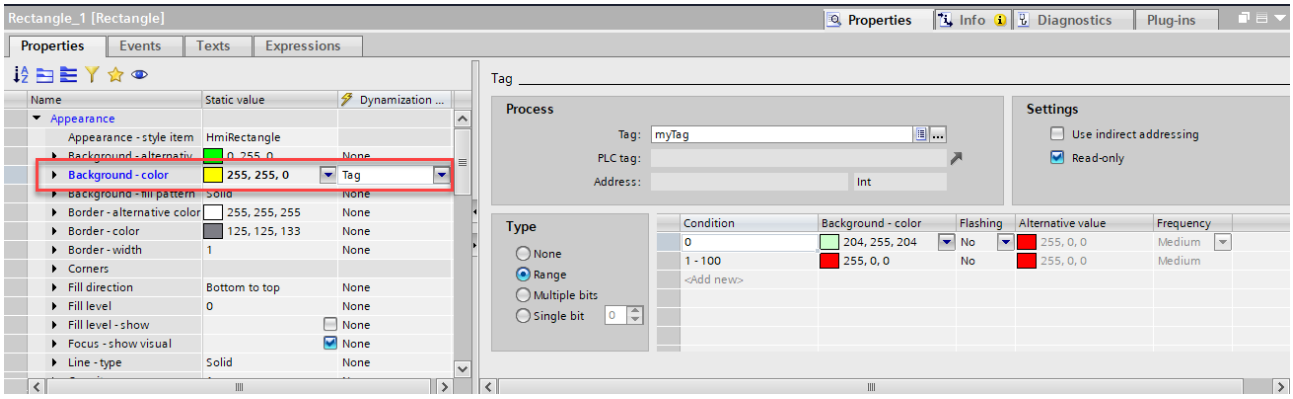


Figure 3-31 Tag dynamization

In this context it is important to mention, that there is also a “Change” event. Through this event, an additional script can be executed If the property is changed at runtime, e.g., if the tag value of the dynamization tag is changed. That happens when this event is triggered during the screen load process can be seen in detail in Figure 2-9.

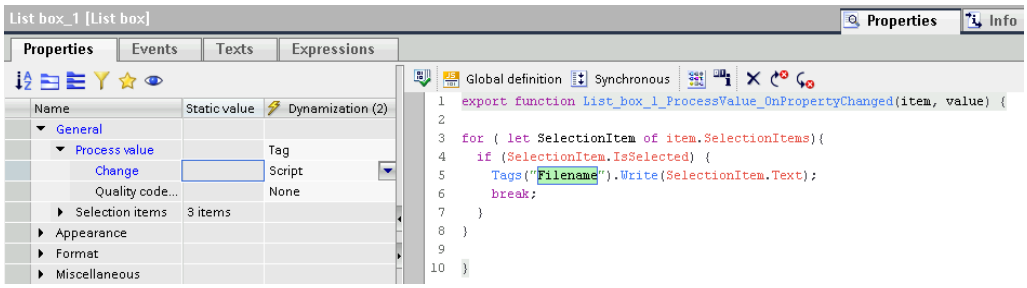


Figure 3-32 Tag dynamization – change event

The second dynamization is a script dynamization. The return value of this script defines the property value after the script is executed. To execute the script a trigger needs to be specified. Information about triggers is given in section 3.4.1.

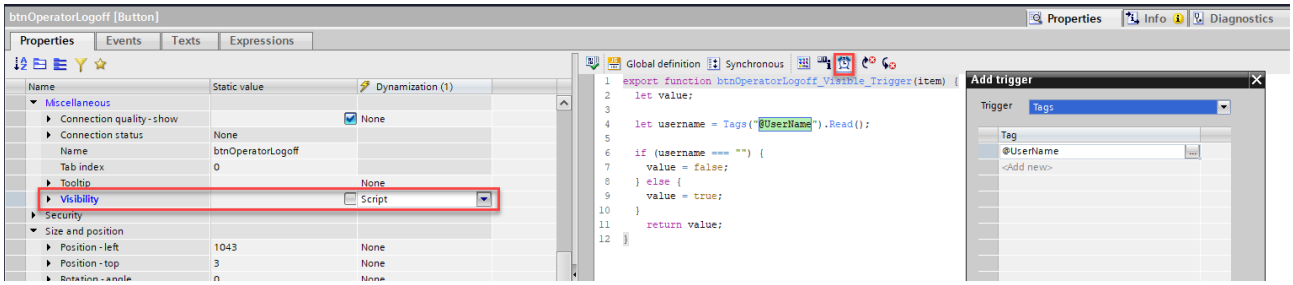


Figure 3-33 Script dynamization

© Siemens 2024

The expressions are another way to dynamize a screen item property. To configure them, there is an additional tab available. With an expression a condition based on multiple tags can be defined and multiple properties can be changed by the conditions.

- The expression is evaluated when one of the tag values changes.
- Properties change as soon as the result of the expression changes.
- If none of the conditions returns TRUE, the default value is assumed.
- If multiple conditions are defined, the first condition in the list that returns TRUE is applied.
- Expressions that cannot be evaluated are skipped, e.g. because of a syntax error or a tag that cannot be accessed.

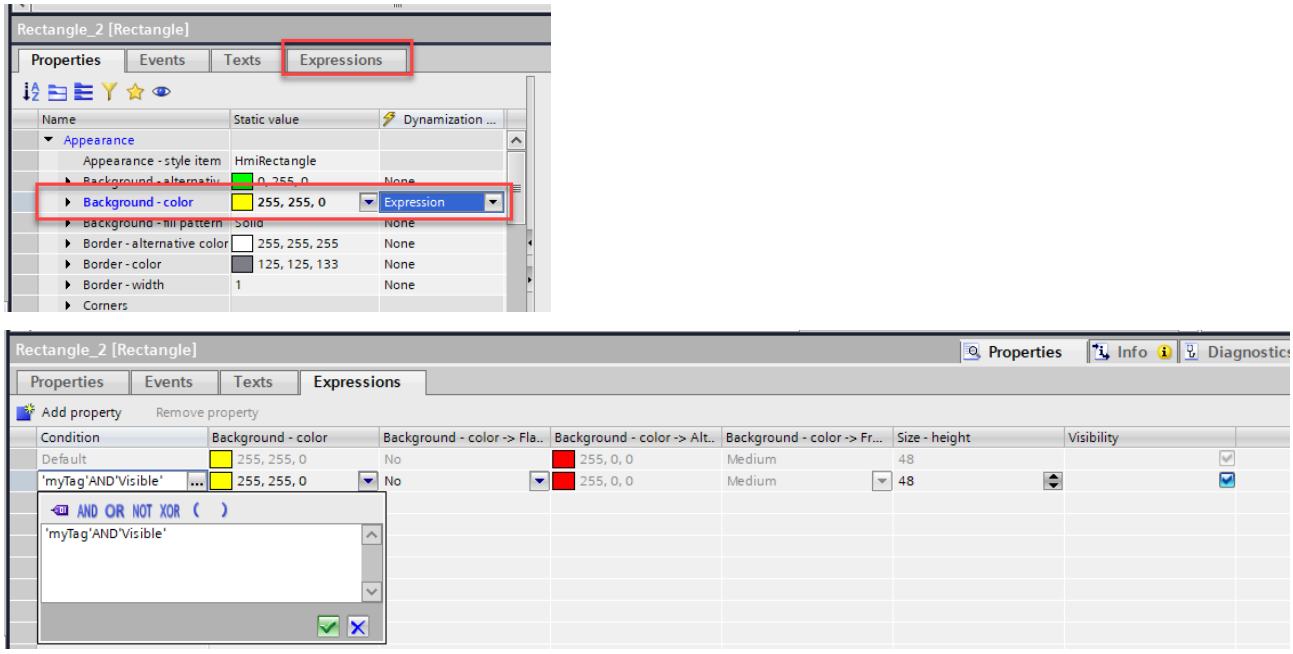


Figure 3-34 Expression as a dynamization

Be aware of the order in which the expressions are defined. They are executed from top to button and once a condition is true, the others are not executed anymore. In the following example, the order must be as follows. If it is defined in reverse order, the AND condition is never evaluated, as the OR condition is always fulfilled first.

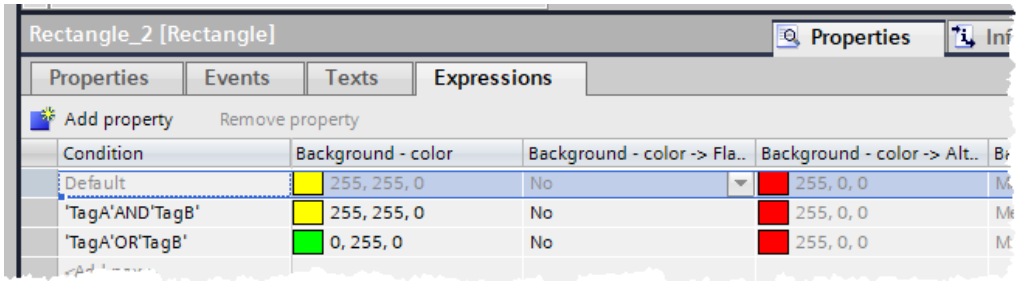


Figure 3-35 Expression execution order

Depending on the property it is also possible to use a resource list or flashing as a dynamization. The flashing option is available for color properties. The resource list for text properties.

© Siemens 2024

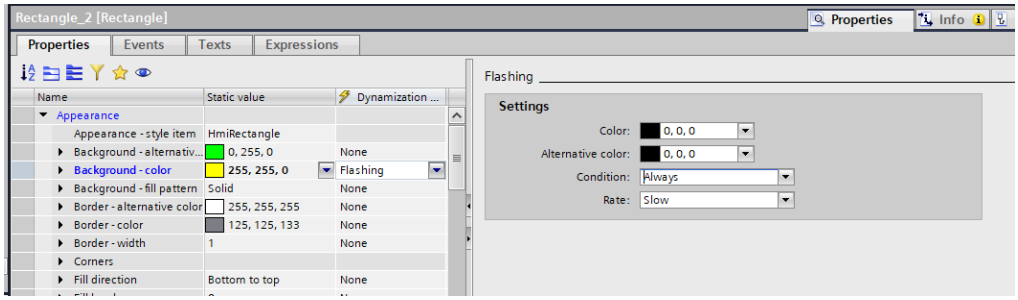


Figure 3-36 Flashing dynamization for color properties

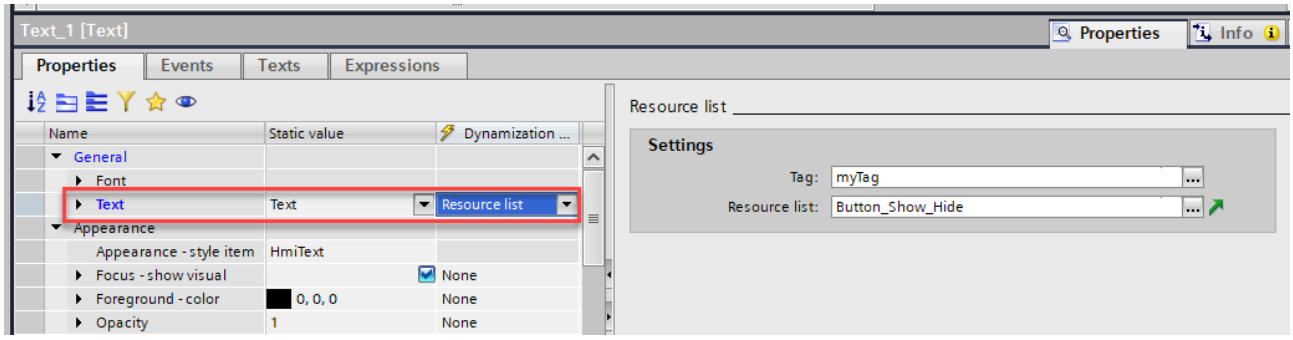


Figure 3-37 Resource list dynamization for text properties

In general, it can be said that expressions and tag dynamizations are the preferred solution for property changes. But if a script can be used to avoid multiple screen objects, scripts are recommended (see 2.2.1).

Dynamizations

Use, if possible, tag dynamization. When more complex logic is required use expressions.

Try to avoid script dynamizations

Information about the different trigger types (cyclic, tag, alarm, event-driven,...) are documented in detail in SIMATIC WinCC Unified – Tips and Tricks for Scripting (Java Script).

© Siemens 2024 |

3.3. Use of Faceplates

Faceplates are used to bring standardization to your projects. If you want to use them, make sure you do configurations with multiple screen objects. The use of a faceplate for individual objects, such as a single rectangle, is not recommended for performance reasons, since the additional overhead due to the interface. Use the WinCC Unified Corporate Designer for this use case (see Use Case: Custom Styles V19). Another thing is the use of faceplates with versions <V19. Check if their functionality can be reimplemented with the new features. Also pay attention to the following.

- Use faceplates screen objects with sophisticated and extensive logical configuration
- Chose appropriate datatypes in the faceplate interface
- Keep in mind the system limits \9\ of a screen when creating a faceplate. Sample calculation for a Unified Comfort Panel 7-12 inch:

  Number of objects per screen for a UCP 7-12“: 800

  20 faceplates on the Screen

  20 Additional Objects on the Screen

  → 800 - 20 Screen Objects - 20 faceplates = 760 Objects that are remaining for the faceplates

→ 760 / 20 = 38 Objects that can be used per faceplate in this use case

3.3.1. Use Case: Strings in the property interface

Description

Within the faceplate property interface different data types can be selected. For a string there are two options, the Multilingual text and the WString (configuration string).

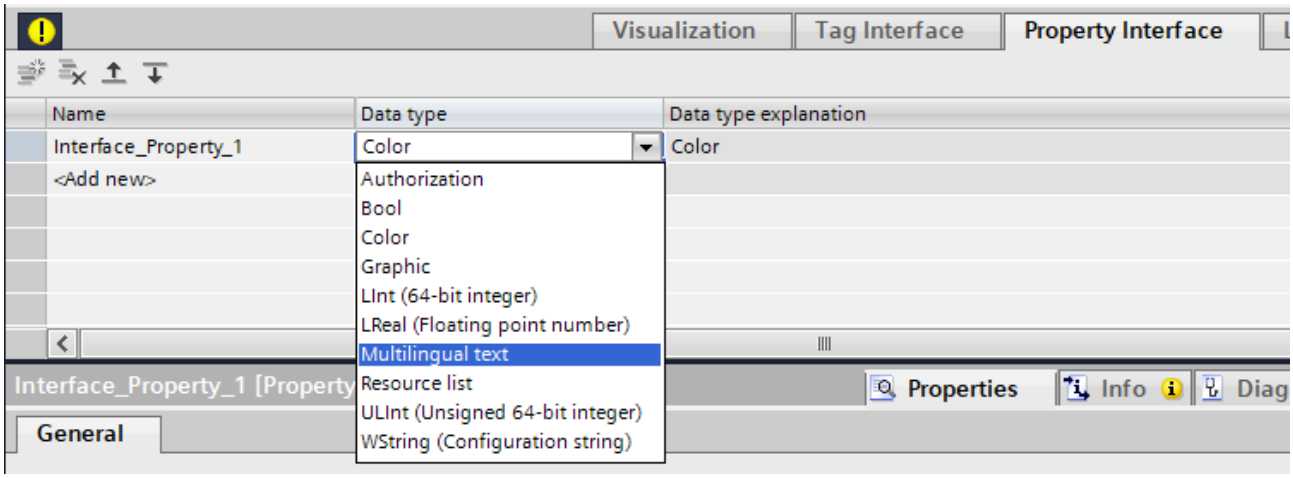


Figure 3-38 Data Types for a Property of a faceplate interface

Solution

If you need a string that is used for a tag dynamization inside a faceplate type, select the multilingual text. A configuration string cannot be linked to a property as a dynamization, but only through scripting.

Configuration strings are only recommended for the transfer of configuration settings e.g., trends in a trend control or alarm filters.

Multilingual strings also have the benefit of saving the text information for the different runtime languages and can be switched during runtime.

© Siemens 2024 |

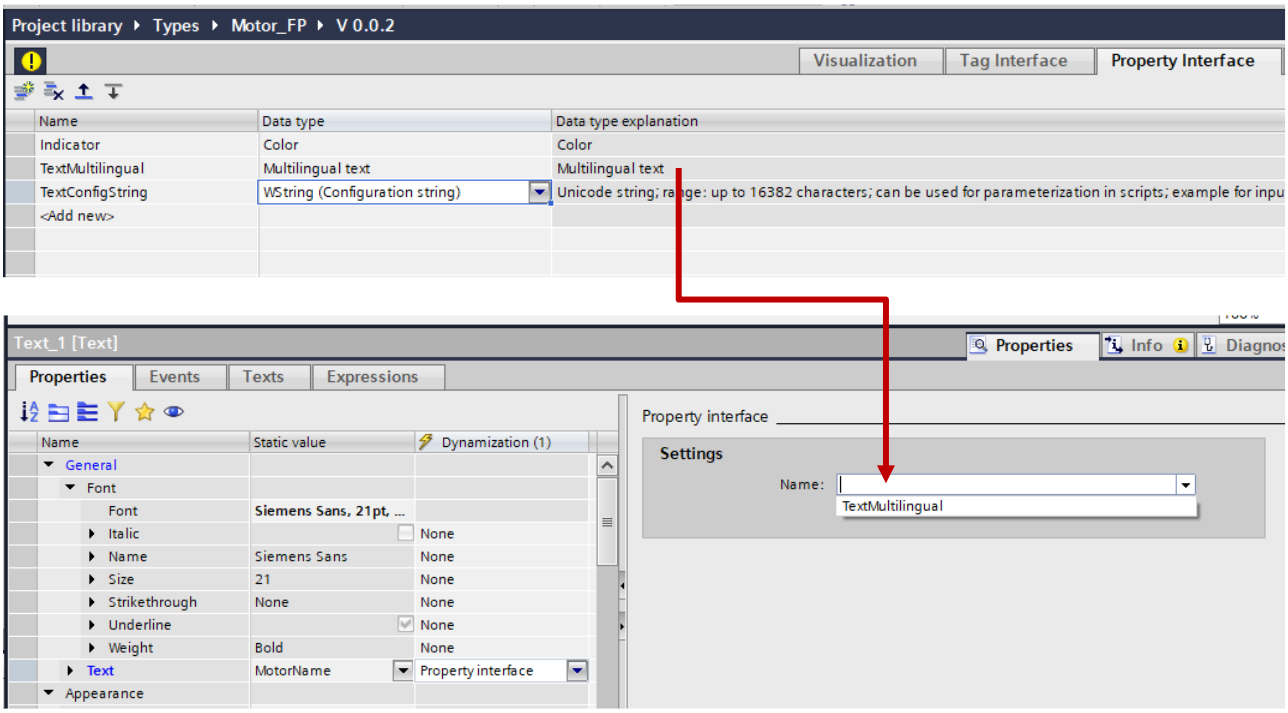


Figure 3-39 Multilingual text and configuration string in faceplate property interface

© Siemens 2024 |

3.3.2. Use Case: Resource Lists with few entries

Description

For a property in a faceplate interface also text- and graphic lists can be selected with the type of resource list. Some text lists might be only 2 or 3 entries long, or only some entries are used inside of the faceplate.

Solution

If only some entries from the text list are used inside of the faceplate or the entries are further individually evaluated for example are written to different screen item properties, transfer them as single texts and as a multilingual text. In the following use case only the NoError,SoftwareError and HardwareError from the ErrorStates text list are needed inside of the faceplate. Therefore, it is enough to transfer these entries instead of the whole list.

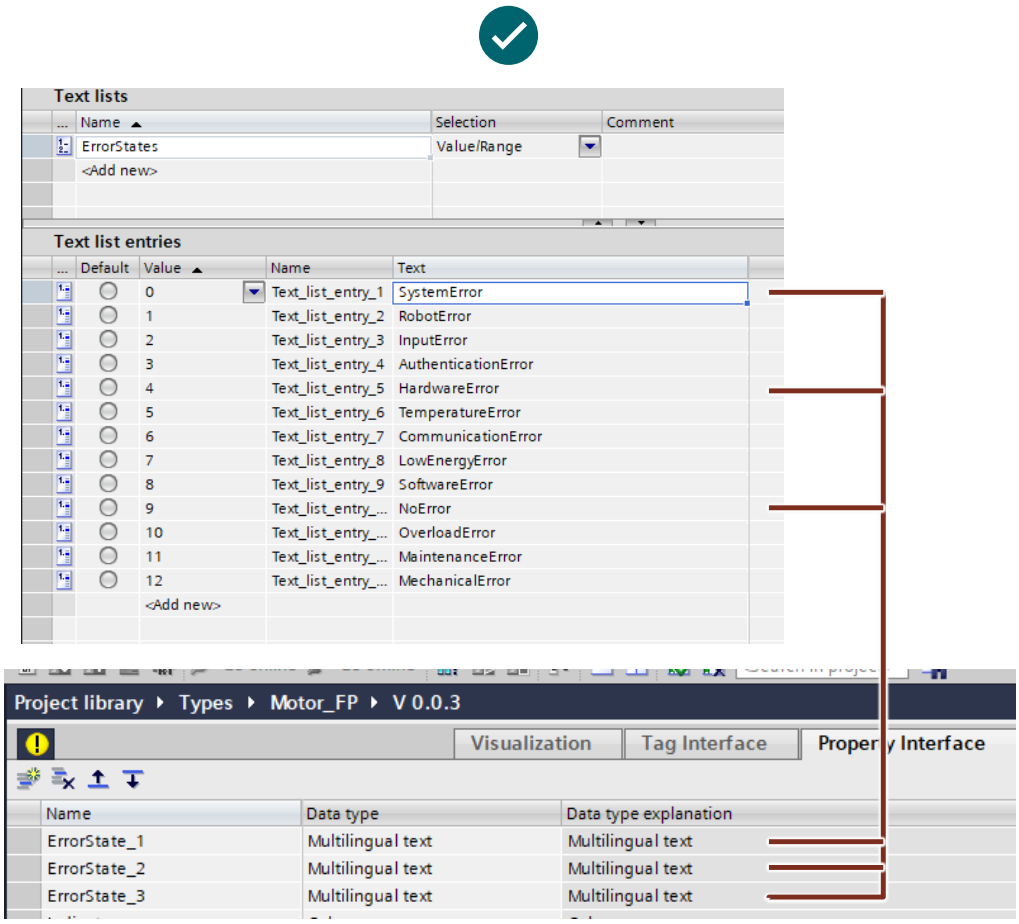


Figure 3-40: Single text list entries as multilingual text in faceplate interface

© Siemens 2024 |

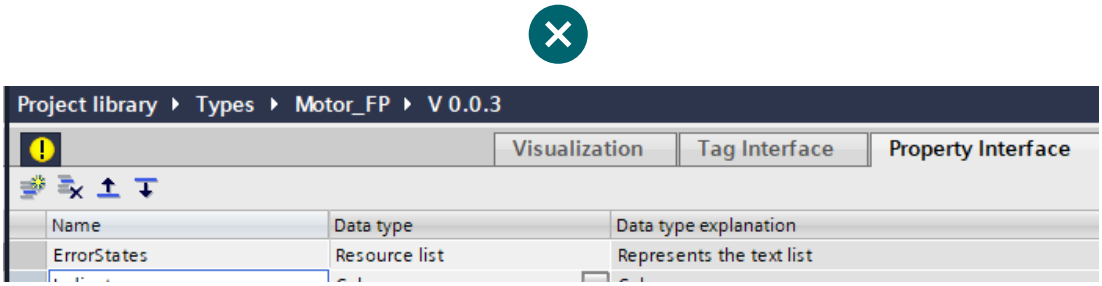


Figure 3-41 Resource list with unneeded entries in faceplate interface

3.3.3. Use Case: Faceplate that is not always visible in Runtime V19

Description

A faceplate, which content is only necessary on demand, can be shown as a popup or can already be configured on screen in the engineering but set to invisible-/.

Solution

Activate the suspendable flag for these faceplate instances. With this setting the cyclic script or scripts triggered by tag changed are not executed when the faceplate is not visible. Not visible refers also to the case that it is out of the current visible screen area.

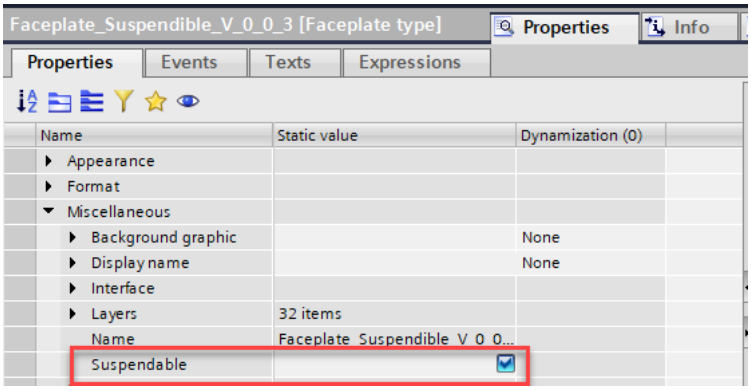


Figure 3-42 Faceplate property suspendable

© Siemens 2024

3.4. Use of scripts

3.4.1. Script Triggers

Avoid cyclic triggers and use tag triggers instead. If you nevertheless need cyclic scripts and the use case permits it (e.g., when synchronizing data or for data exchange with databases), then configure the scripts in the Task Scheduler. The Task Scheduler runs in a separate process in the background and places less load on your project than if you had configured the scripts in the screen.

But also, be aware when using the setting “Tags-automatic”, that all referenced tags of the script are added. This can lead to a lot of triggers and endless loops when writing tags.

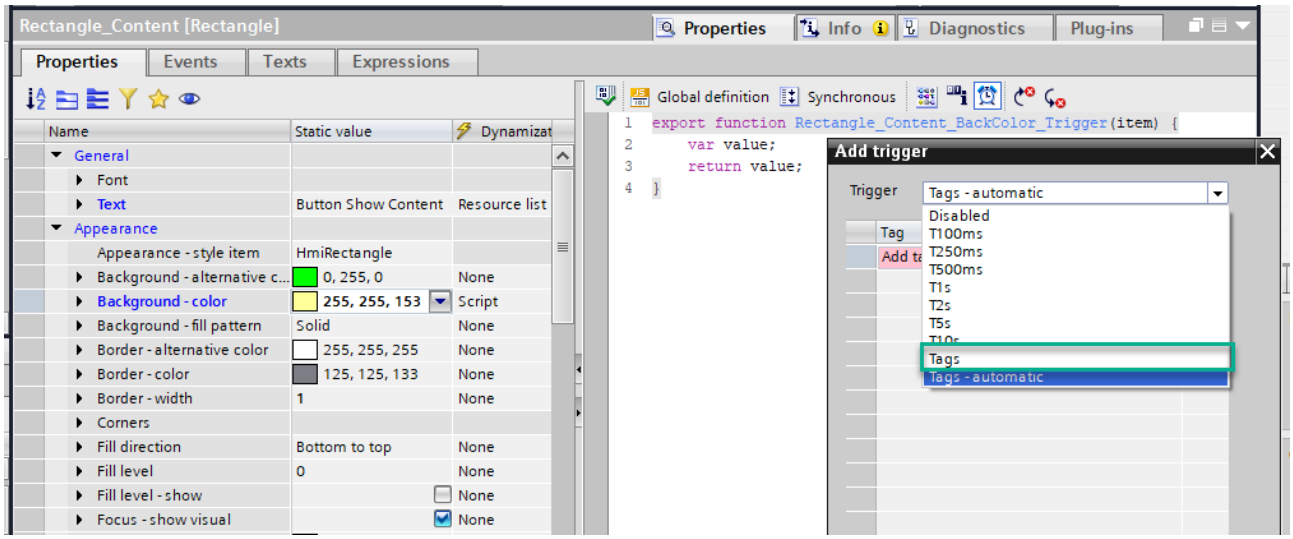


Figure 3-43 Script triggers for script dynamization

If two scripts are triggered by the same tag, combine them into one single script.

Script triggers

Use tag triggers and avoid cyclic triggers

© Siemens 2024 |

3.4.1.1. Use Case: Scripts using the same trigger tag

Definition

When script dynamizations are necessary, some might be triggered with the same tag. So multiple scripts are triggered by the same tag.

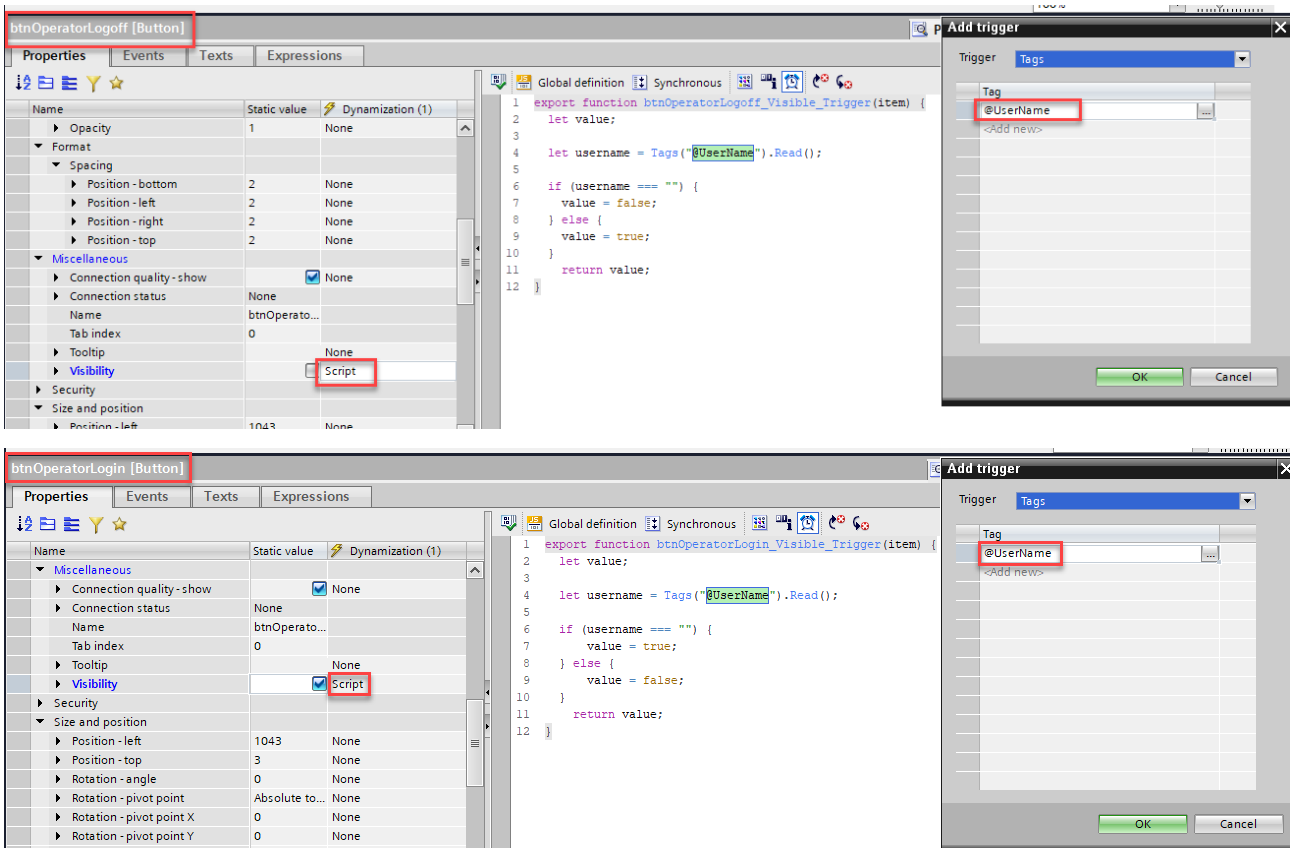


Figure 3-44 Script dynamizations with same trigger tag

Solution

Create a subscription in the loaded event of the screen for this tag and manage centrally the property changes with one script. There is already a snippet for the tag subscription available. Especially when the same scripting logic is implemented in the scripts (e.g., if-else) you save a lot of script execution through the one subscription.

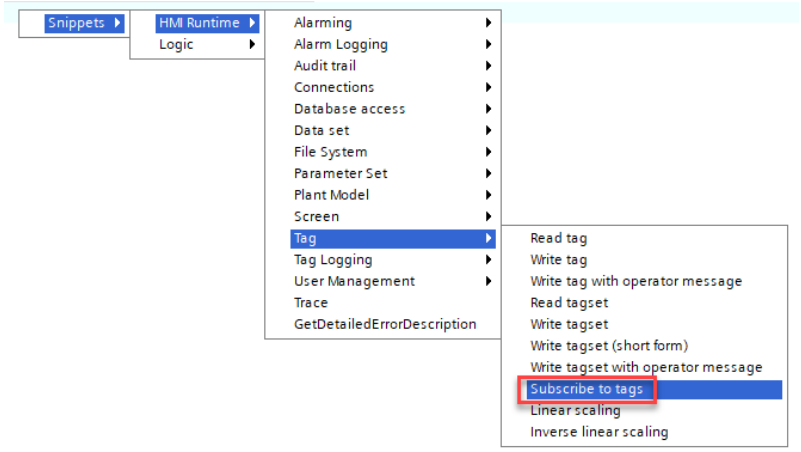


Figure 3-45 Tag subscription snippet

© Siemens 2024

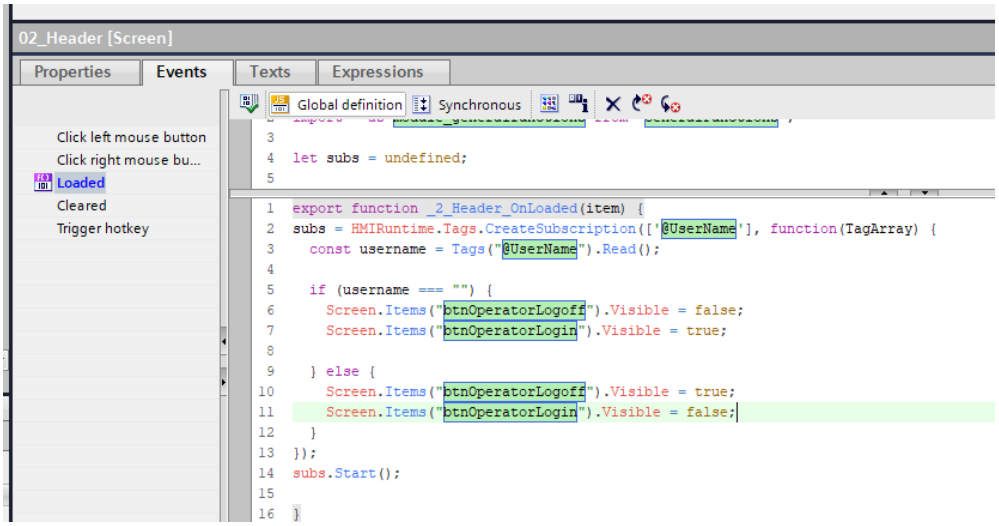


Figure 3-46 Tag subscription in loaded event of the screen

© Siemens 2024 |

3.4.2. Efficient Code

Efficient coding is all about reducing parts in your scripts that may cause to problems and even about part that do not have an influence on the result of the scripts. There are a lot of points that you should pay attention to. Therefore, just check if your code does not contain elements of the following list:

- Delete functions or import statements in the global definition area that are not called

Prefer system functions from the system function list, instead of scripting in the JavaScript editor

- Delete empty events

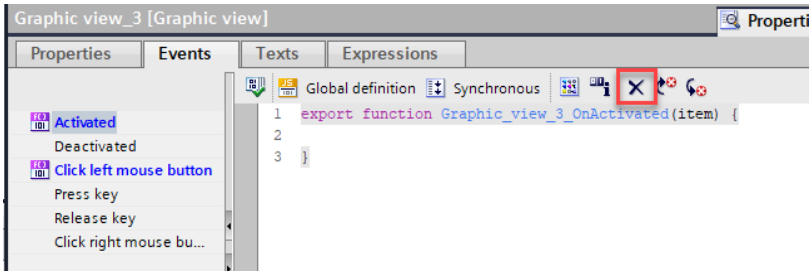


Figure 3-47

- Avoid unnecessary loop iterations by targeted use of the break statement

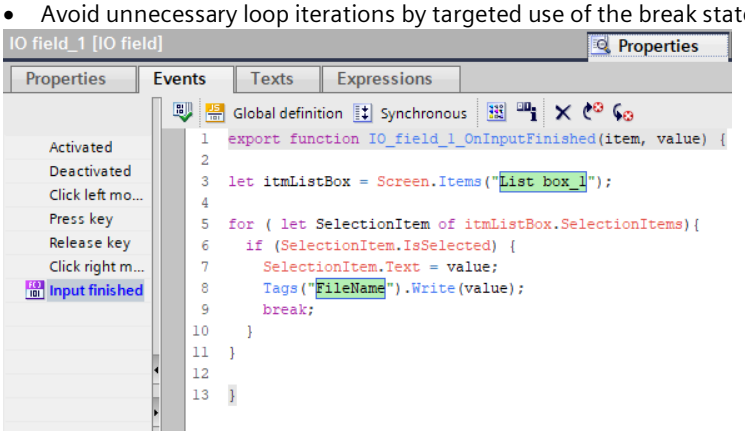


Figure 3-48 Break statement to jump earlier out of for loop

- Delete or comment out constants, variable definitions, debugging traces that are not used anymore
- Only read tags that are used in the script
- If two scripts are triggered by the same tag, combine them into one single script.
- Read tags once, save them in a variable and reuse this one if the tag is used multiple time in the script
- Use a “const” definition when possible (if the value does not need to be changed in the script context) and otherwise define a variable with “let”. Do not use “var” anymore, because it is a deprecated java script standard.
- Use a switch-case instead of if-else

© Siemens 2024

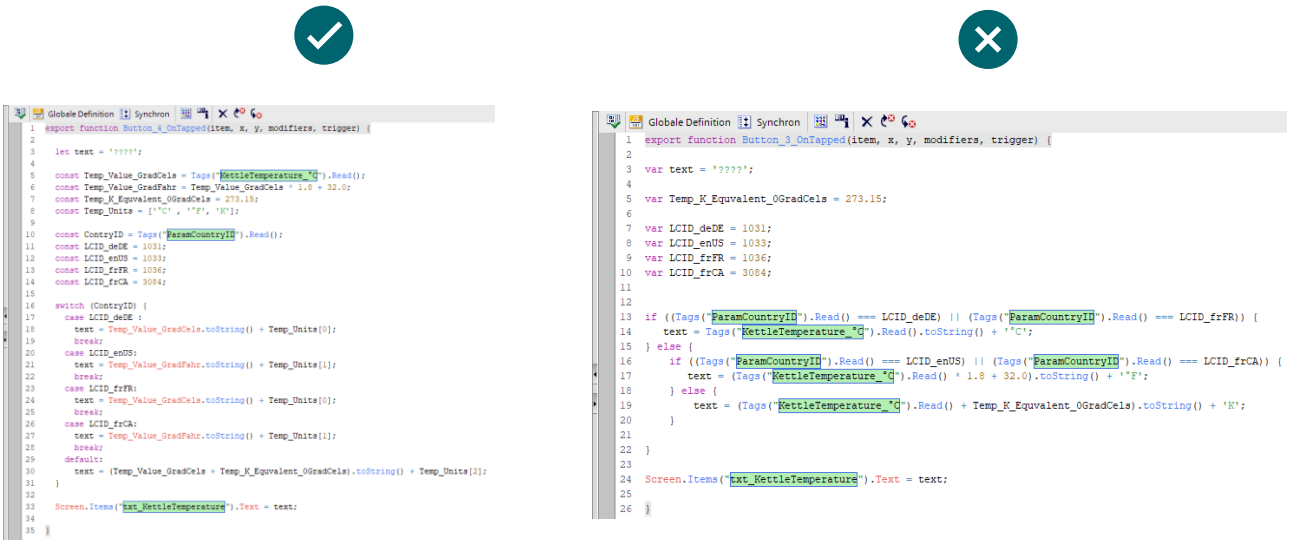


Figure 3-49 Switch case vs. if-else

Use asynchronous scripts if possible. For calls that can take longer e.g. database access, only the async variant is offered anyway. For tag access the sync and the async mode is supported because tag accesses are normally faster. More information regarding the script call can be found in the Java Script Tips and Tricks.

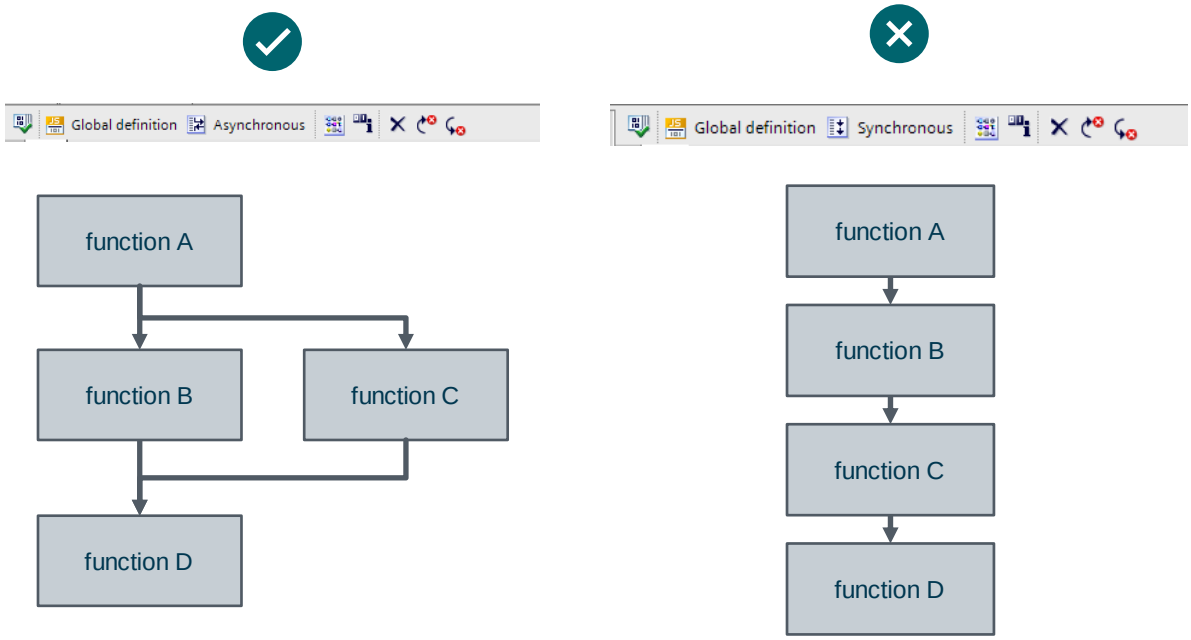


Figure 3-50 Compare of asynchronous and synchronous script call

- Be aware of how often scripts are executed, especially during the screen load process (see Figure 2-9). The following example is for a script in the on change event of the dynamized process value of a list box. The visibility of these property change events can be switched in the engineering through the eye icon V19.

© Siemens 2024

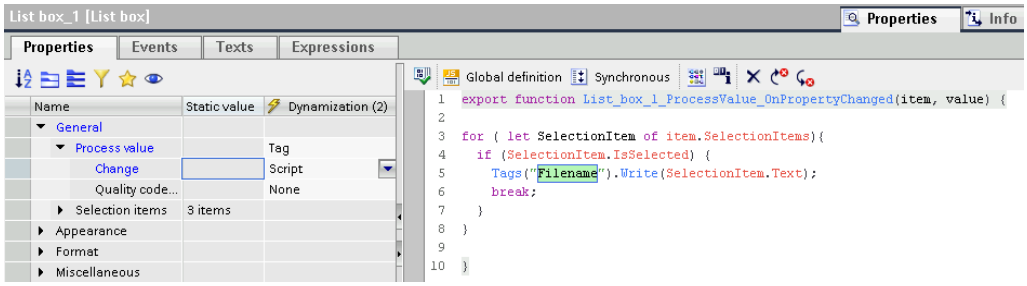


Figure 3-51: On change event from process value of a list box

- Read several text list entries at once via JS with

  HMIRuntime.Recources.TextList(txtList).TextsByValue(HMIRuntime.Language, value); instead of using a for loop. V19

```
let value = {}
for(let i = 1; i >= count; i++)
{
  values.push( i + currentPage * 6);
}
let texts = HMIRuntime.Resources.TextLists(txtList).TextsByValue(HMIRuntime.Language, values);
}
```

Figure 3-52 Read test list entires at once

3.4.2.1. Use Case: Write and read multiple tags

Description

To write and read single tags a lot system functions are provided and also through Tags(“Tagname”).Write(“value”) or Tags(“Tagname”).Read() the tags can be addressed through scripting.

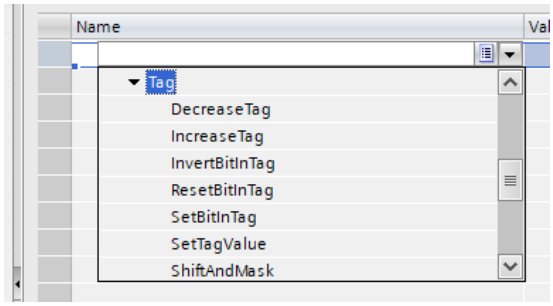


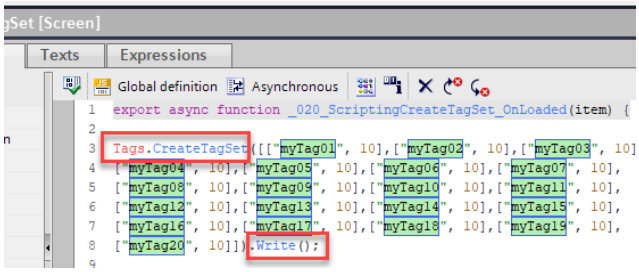
Figure 3-53: Excerpt from the system function in the tags section

But there could be also a tag set created, that covers multiple tags in a container.

Solution

Create a tag set when writing or reading multiple tags and use its Write and Read method to only send one collective order to the PLC.

© Siemens 2024 |



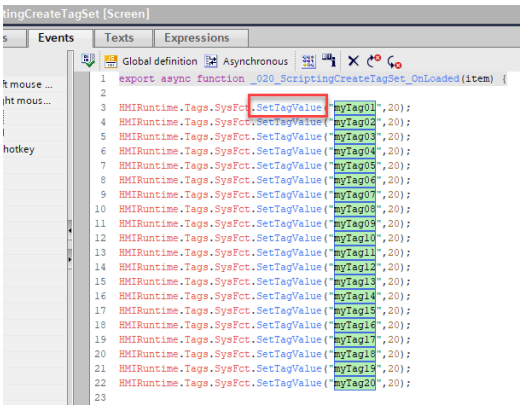


Figure 3-54

Read and Write Tags

Use a TagSet to read and write multiple tags instead of single calls of SetTagValue. Especially for synchronous reading and writing of several control variables.

For more information regarding scripting read the SIMATIC WinCC Unified – Tips and Tricks for Scripting (Java Script).

© Siemens 2024 |

3.5. Others

- Consider the used acquisition cycle for PLC tags

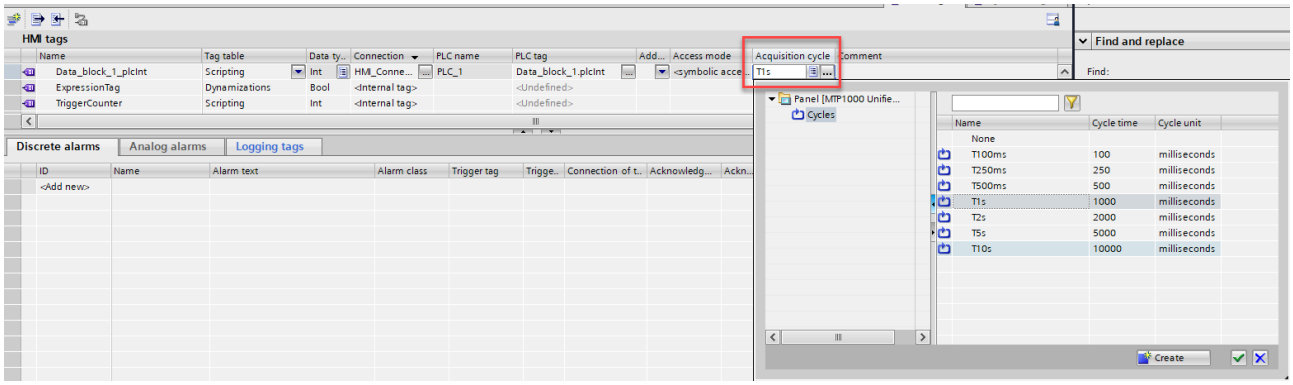


Figure 3-55: Acquisition cycle for HMI tags with PLC connection

- Reduce empty textlist entries
- Be aware of the system limits

Screens

	Unified Comfort 7-12"	Unified Comfort 15-22"
Maximum size in the engineering system	20,000 * 20,000 pixels	
Maximum size in runtime	20,000 * 20,000 pixels	
Number of screens	1200	
Number of lower-level screen windows	10	
Number of objects per screen	800	1200
Number of objects from the "Controls" area per screen	40	80
Number of tags per screen	600	800

Figure 3-56: Excerpt from the system limits defined in \9\

- Only relevant for PC-RT: Script Debugger should be disabled in productive use of the runtime project

© Siemens 2024

3.5.1. PLC UDT Arrays with Multiplexing

Definition

To describe objects (e.g., a motor) with tag values, UDTs are a good way to structure the data that belongs to the object

Data_block_1

	Name	Data type	Start value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint
1	Static							
2	Motor1	"UDTMotor"						
3	Name	WString	WSTRING#''					
4	Speed	Real	0.0					
5	Acc	Real	0.0					
6	Motor2	"UDTMotor"						
7	Name	WString	WSTRING#''					
8	Speed	Real	0.0					
9	Acc	Real	0.0					

Figure 3-57 Small motor PLC UDT

If there are a lot of instances of this object (e.g., 20), but not all data of all instances need to be available at once in runtime, there is the option of multiplexing. Thereby only a subset from this array (e.g., 5 motors) is addressed and when the data of another motor is required only the reference to the instance needs to be changed by selecting another index of the array.

Data_block_1

	Name	Data type	Start value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint	Supe
10	Motors	Array[0..20] ...							
11	Motors[0]	"UDTMotor"							
12	Name	WString	WSTRING#''						
13	Speed	Real	0.0						
14	Acc	Real	0.0						
15	Motors[1]	"UDTMotor"							
16	Motors[2]	"UDTMotor"							
17	Motors[3]	"UDTMotor"							
18	Motors[4]	"UDTMotor"							
19	Motors[5]	"UDTMotor"							
20	Motors[6]	"UDTMotor"							
21	Motors[7]	"UDTMotor"							
22	Motors[8]	"UDTMotor"							
23	Motors[9]	"UDTMotor"							
24	Motors[10]	"UDTMotor"							
25	Motors[11]	"UDTMotor"							

Figure 3-58: Array of 20 motor PLC UDTs

Sometimes the UDTs are much more complex and store a lot of data that might be necessary for the PLC but not for the HMI.

© Siemens 2024 |

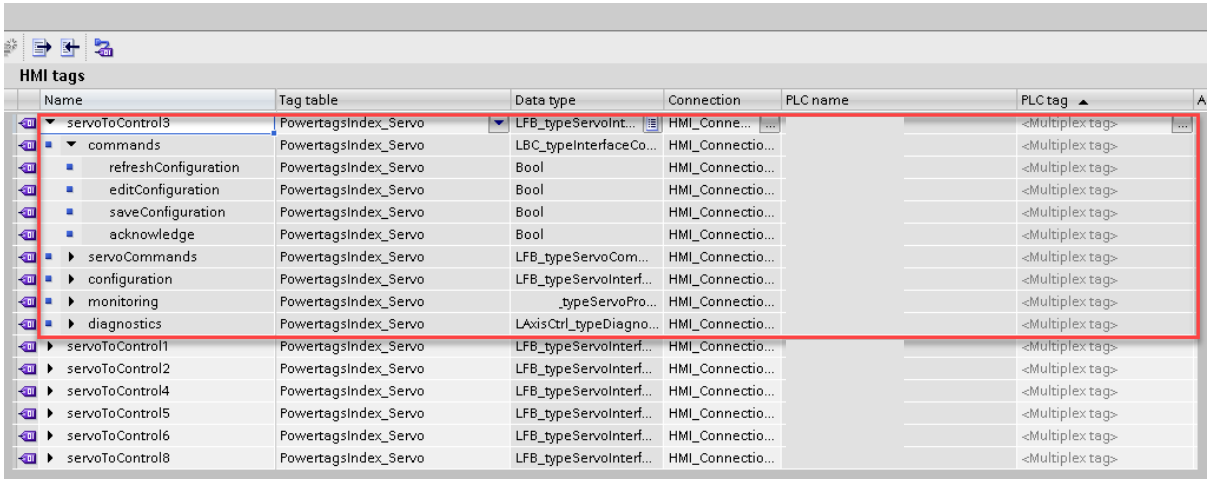


Figure 3-59: Complex PLC UDT with more data than required in HMI

Solution

If only a small amount of the data from the UDT instances is required in the visualization, do not use the multiplexing. Even if only one datapoint from the UDT is linked to an HMI property, when changing the concrete instance of the UDT, the whole structure is loaded from PLC and not only the part that is needed.

For data stored in complex PLC UDTs load each UDT as an own instance and do not use arrays with multiplexing. Every time an index is changed the whole structure at the PLC will be refreshed even if it is not in use.

Multiplexing of PLC UDTs

Only use multiplexing for simple PLC UDTs or when all data from the UDT instances is also required in the visualization

© Siemens 2024 |

4. Analysis of an existing project

When there is already an existing project, regardless of whether problems have already occurred during runtime or not, it is always beneficial to carry out an analysis whether there are opportunities for improvement in the engineering of a project. In this section a guideline is provided that shows you all possibilities to further analyze your WinCC Unified project and therefore also gives you an overview about a possible step-by-step analysis procedure.

If there is a known problem in runtime (screens are loaded slowly, some functionality is not working as intended), you can directly jump in the object or script analysis. Otherwise, you can test the project on the target device in runtime and check if everything works as expected.

For an analysis it is also important to be familiar with the general runtime workflow (e.g., the order of running scripts) and what is running maybe in the background (e.g., cyclic scripts for timers), to be able to go to the responsible location (script, configuration, task) in the engineering when a problem is found. If there is to less knowledge about the running scripts and order of processing during runtime, the Google Chrome script debugger gives a good opportunity to automatically navigate through all the executed scripts during screen load, event execution through a button click or a specific screen change.

Another method is to put traces into scripts and check through the RTIL Trace Viewer when they are triggered.

Once you are familiar with the general script execution and processes in your project, you have a good knowledge base for the further analysis.

The following scheme shows a possible procedure in which order and how to choose the tools for analyzing the project, when to look at the runtime or when to focus more on the scripting part of the project. The analysis also consists of two parts, the object and the script analysis. The order, which one is made first, can be changed depending on the project.

All procedures that are mentioned in the flow chart are described in detail in the following chapters.

© Siemens 2024 |

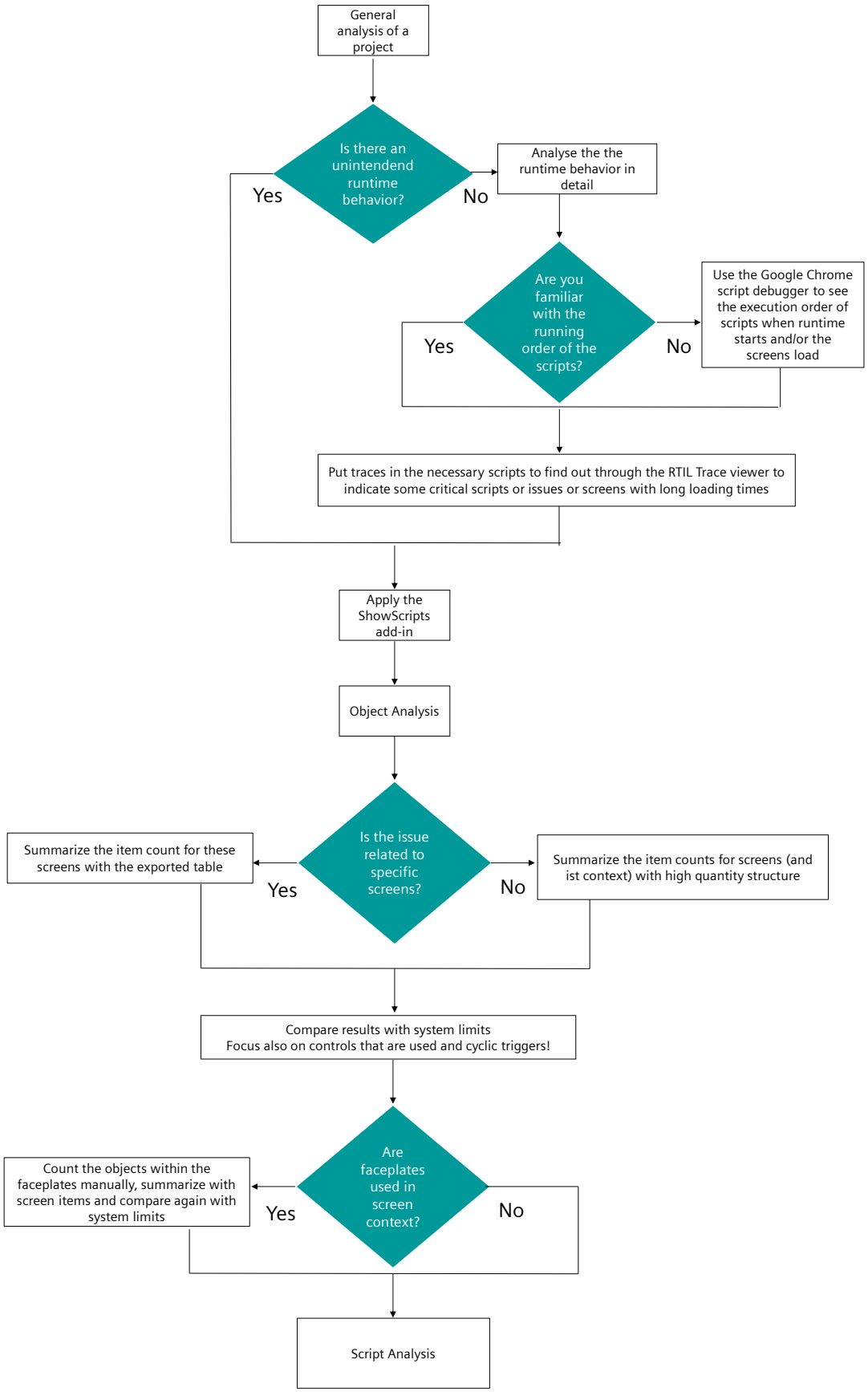


Figure 4-1 Project analysis startup flow chart

© Siemens 2024 |

The following flowchart shows the different possibilities for script analysis.

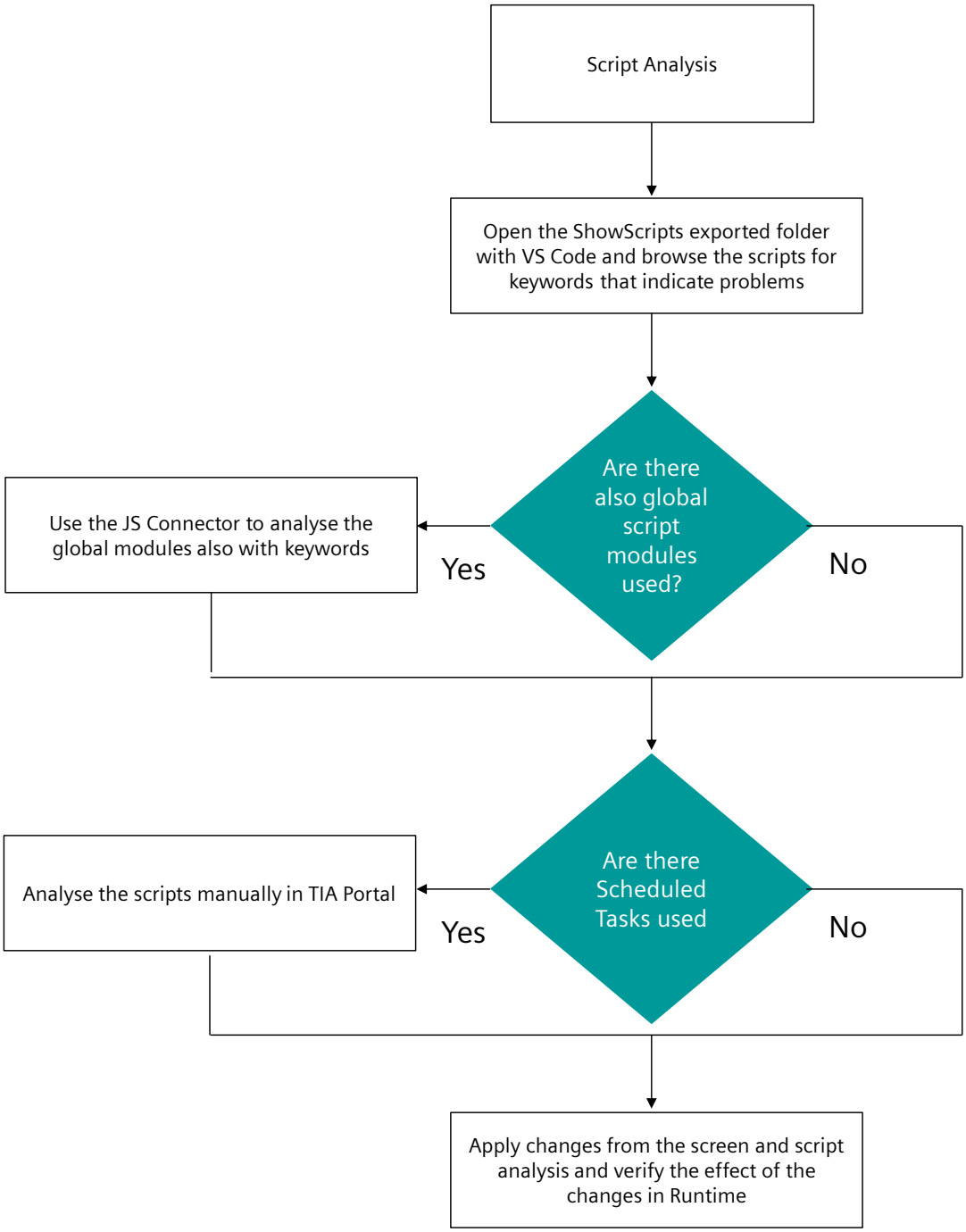


Figure 4-2 Project analysis script analysis flow chart

© Siemens 2024 |

4.1. ShowScripts Add-in

The ShowScripts add-in is a TIA Portal add-in which provides an overview about HMI devices in TIA Portal projects and their quantity structures regarding screens, dynamizations, scripts and events.

The following chapters describe how to retrieve the add-in and analyze the export data of an HMI device regarding efficient engineering.

4.1.1. Download and installation

The add-in is an open-source tool and provided free of charge via Github. Navigate to the releases to download the latest version.

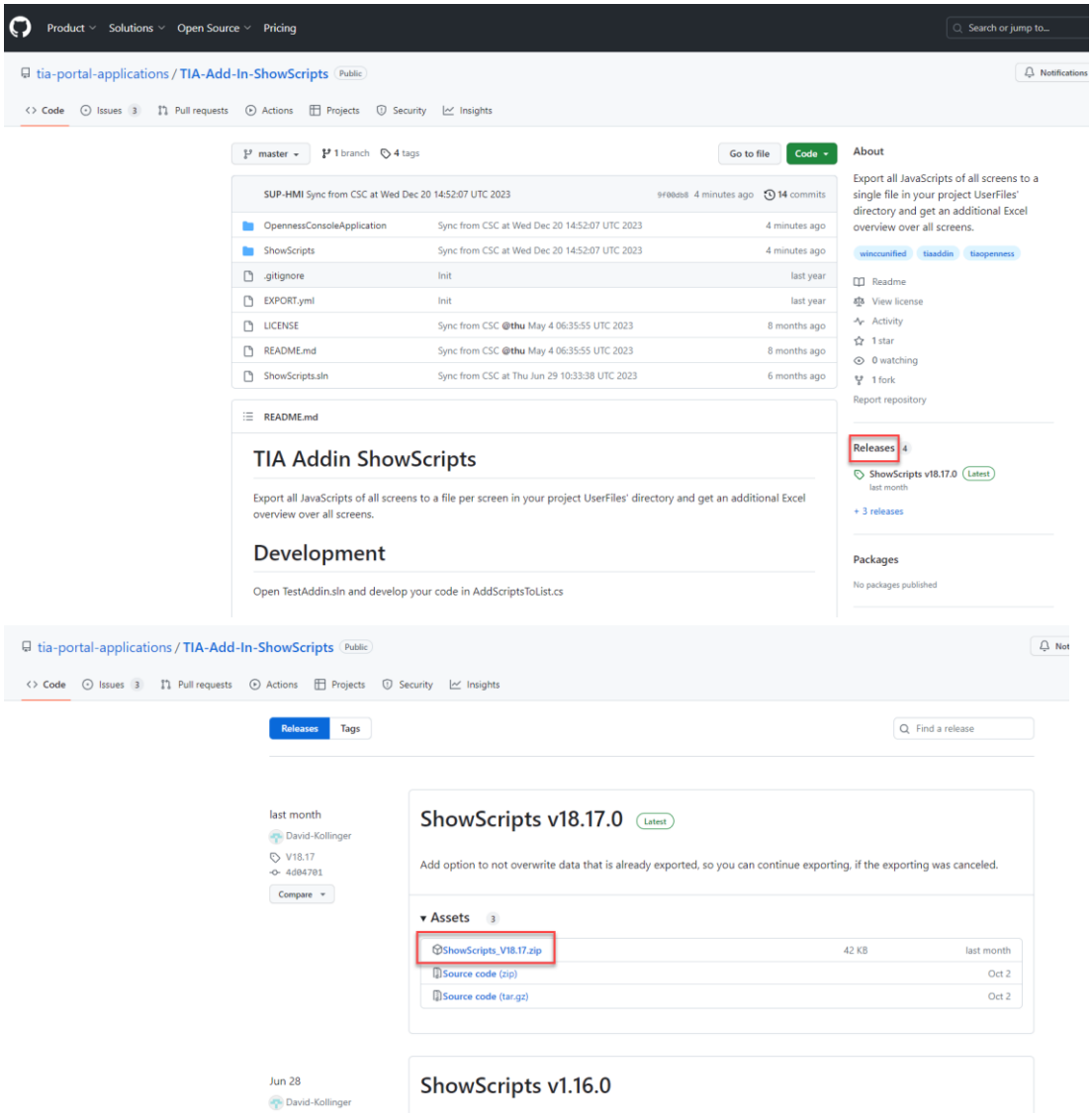


Figure 4-3 ShowScripts add-in on GitHub

NOTE Please be aware to download the most current version of the add-in to take benefit from the latest bugfixes.

© Siemens 2024

After the download of the .zip folder of the ShowScripts add-in, it needs to be installed in TIA Portal. Please refer to the respective application example in the Siemens Industry Online Support for a description how add-ins can be installed and used.

The add-in is ready for use as soon as you can see the green status in the add-ins task card in the top right corner.

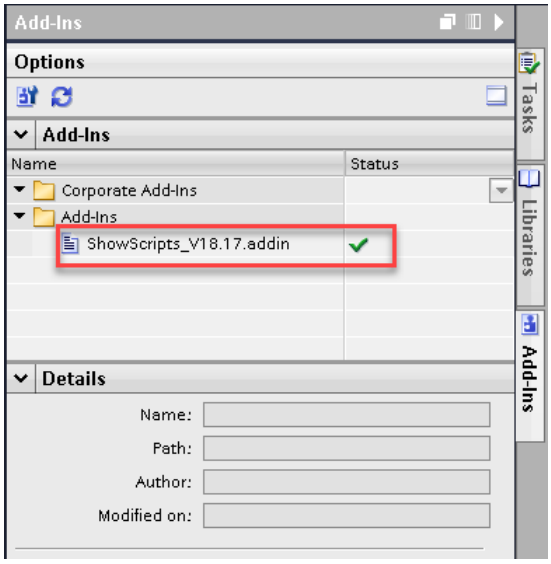


Figure 4-4 ShowScripts add-in installed in TIA Portal

4.1.2. Usage of the ShowScripts Add-in

Function scope

The function scope of the ShowScripts add-in is the export of relevant quantity structures and scripts of a WinCC Unified based device which means:

- All scripts which are used on the screen items within every screen as a js file
- A tabular overview (csv file) of all screens and its respective screen item amount

Starting the analysis with ShowScripts

1. For getting started with the ShowScripts add-in open the to be checked project in TIA Portal and follow the steps below.
2. Right-click on the Unified device you want to analyze with the add-in
3. If the add-in is ready to use (see 4.1.1) the option “ShowScriptCode” will be available at the right-click menu. There are two “Export” and one “Import” option available.
    - **Export all scripts of HMI:** Option for the very first export of scripts and screen overview
    - **Export all scripts of HMI overwrite:** Option for subsequent exports to overwrite existing export files after a previous export failure
    - **Import all scripts to HMI:** Option for the import of adapted scripts back to the project

For the initial analysis of a device in your project select the option “Export all scripts of HMI” to start the project analysis.

© Siemens 2024 |

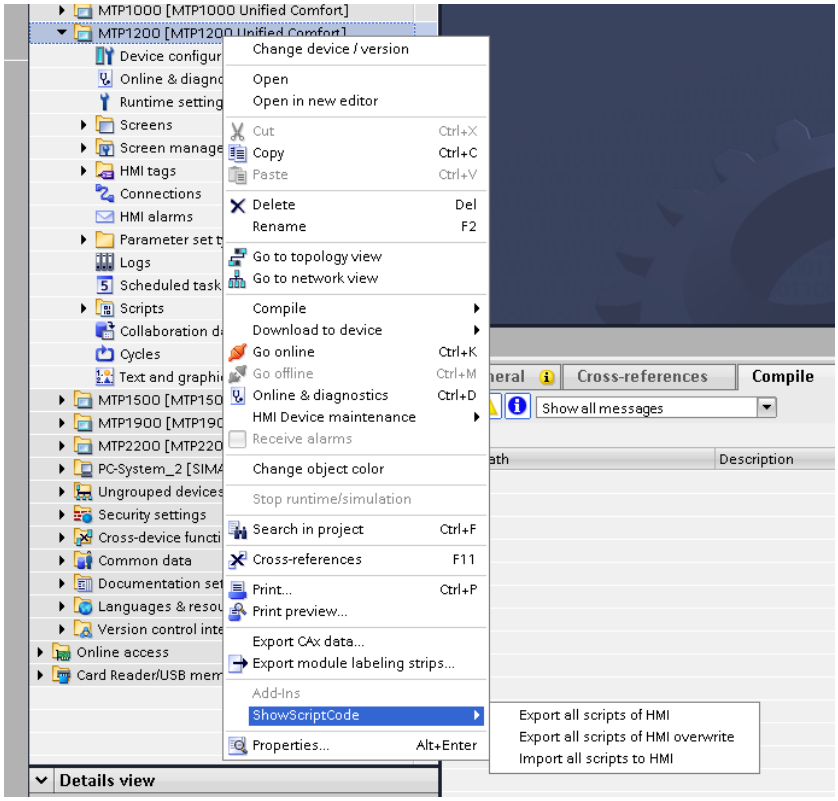


Figure 4-5 Initial export via ShowScripts add-in

As next step the TIA Portal will ask you for confirmation that the add-in will access your project's data via Openness which you need to confirm with "Yes to all".

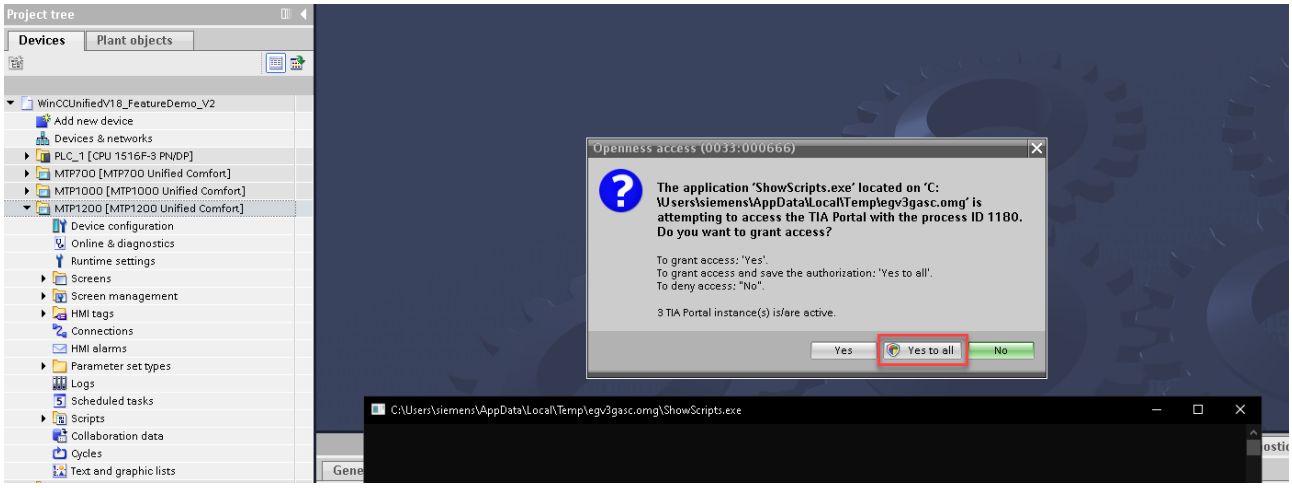


Figure 4-6 Confirmation of the add-in's project access

After "yes to all" has been executed, a window opens in which you have to assign a screen name.

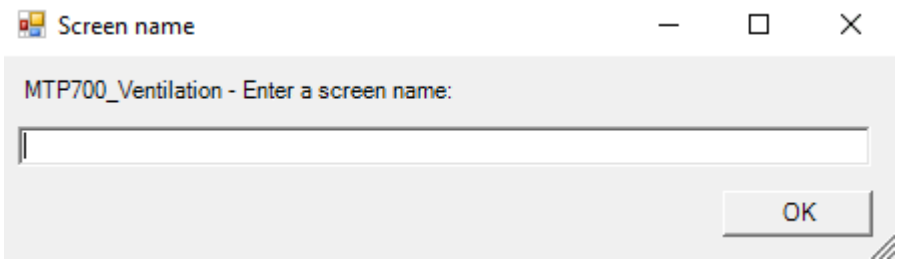


Figure 4-7 Enter a screen name

© Siemens 2024 |

Afterwards you will see the analysis progress of the add-in in the command window. If the tool runs successfully the command window will close itself after running through all the screens in the device. In the folder “UserFiles” of your TIA Portal project folder there will be a new folder created with the respective device name in which you can find the exported scripts and a csv-based screen overview file.

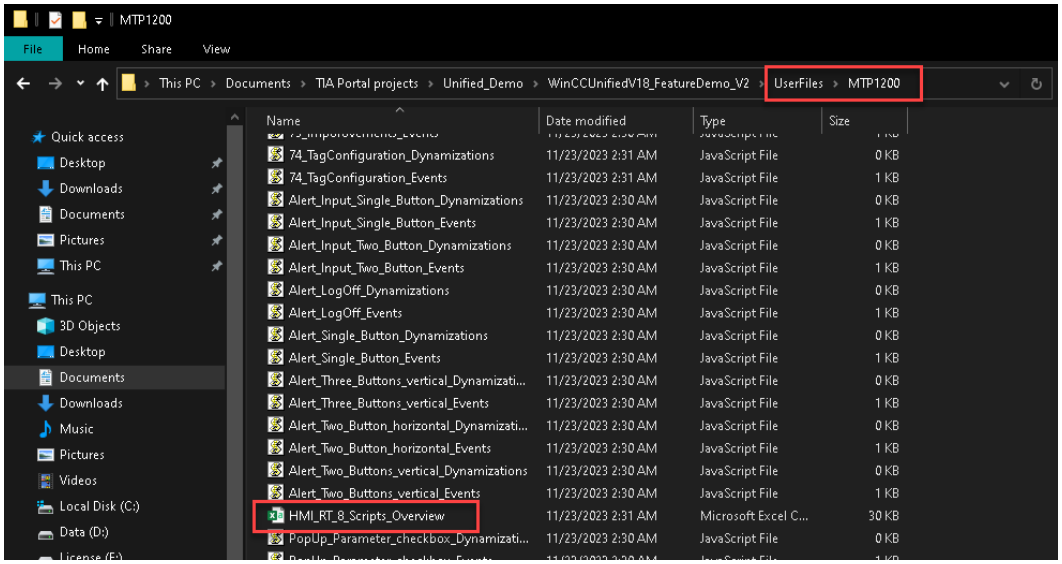
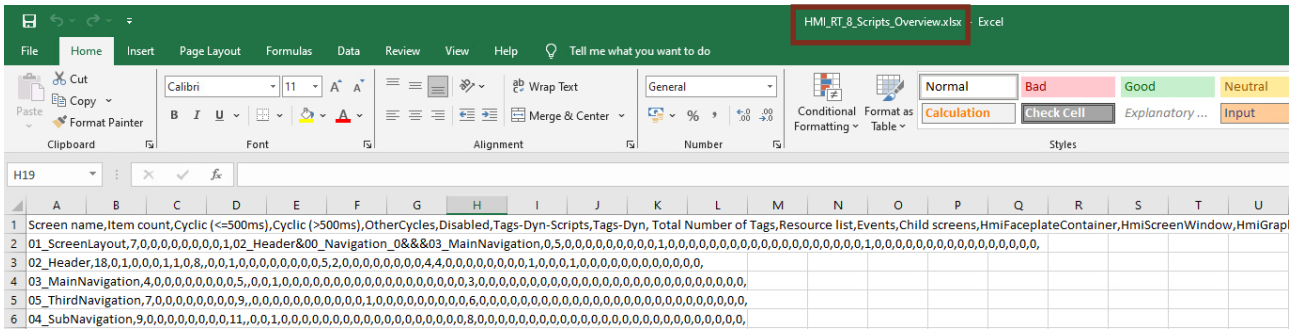


Figure 4-8 Results of the add-in export in UserFiles > “DeviceName”

4.1.3. Preparation of the screen overview file

By default, the exported screen overview file has the format csv. To be able to adapt the file format and save the changes later you need to initially open the file e.g., with Microsoft Excel and save it separately with the file format “.xlsx”. Afterwards you can work on this file and save the shown implemented changes permanently.

After the file gets opened the first time the values are still in comma-separated format and it is hard to evaluate the analysis.


HMI_RT_8_Scripts_Overview.xlsx - Excel
File
Home
Insert
Page Layout
Formulas
Data
Review
View
Help
Tell me what you want to do
Clipboard
Font
Alignment
Number
Styles
Normal
Bad
Good
Neutral
Calculation
Check Cell
Explanatory ...
Input
H19
Screen name,Item count,Cyclic (<=500ms),Cyclic (>500ms),OtherCycles,Disabled,Tags-Dyn-Scripts,Tags-Dyn, Total Number of Tags,Resource list,Events,Child screens,HmiFaceplateContainer,HmiScreenWindow,HmiGrapl
01_ScreenLayout,7,0,0,0,0,0,0,0,1,02_Header&00_Navigation_0&&&03_MainNavigation,0,5,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,
02_Header,18,0,1,0,0,0,1,1,0,8,,0,0,1,0,0,0,0,0,0,0,0,0,5,2,0,0,0,0,0,0,0,0,0,4,4,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,
03_MainNavigation,4,0,0,0,0,0,0,0,0,5,,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

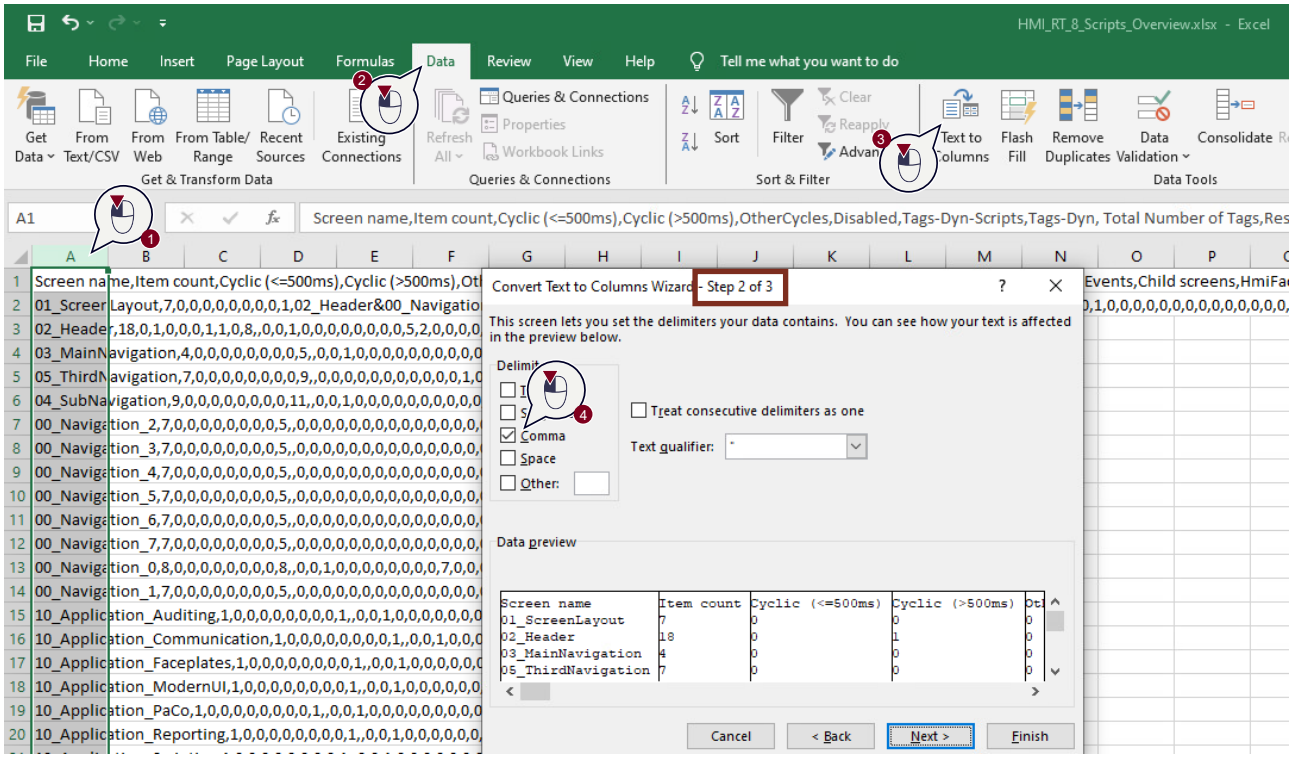


Figure 4-10 Preparation of the analysis data – step 1

4. The readability can be increased even more by auto-fitting the column width depending on the table head texts. Therefore, all columns need to be marked and the option “AutoFit Column Width” in the register card “Home” needs to be selected.

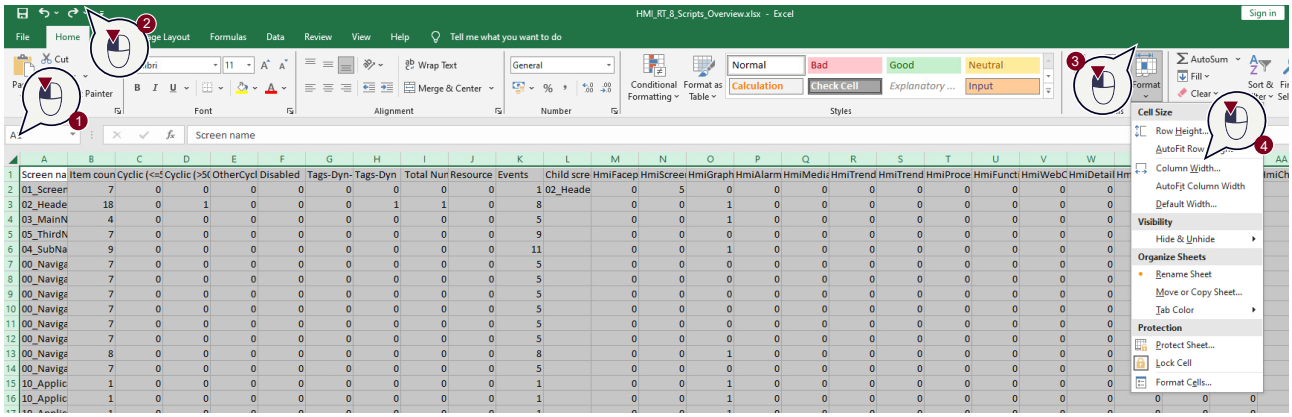
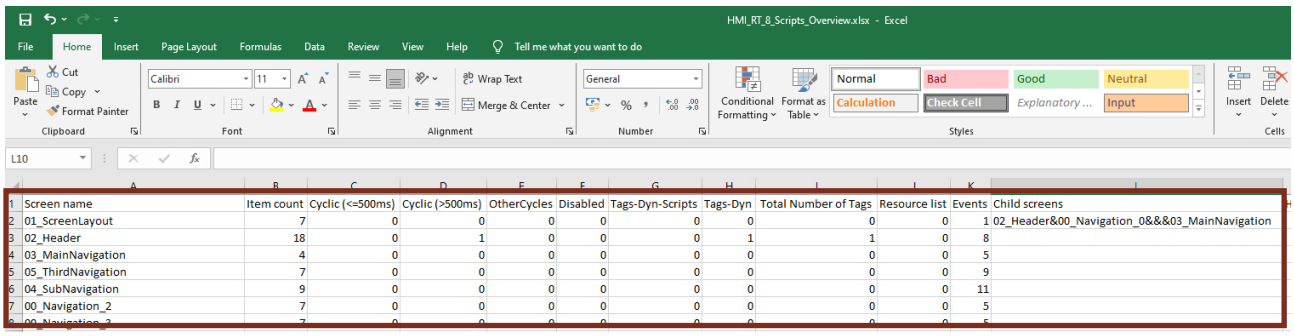


Figure 4-11 Preparation of the analysis data – step 2

After these two steps the readability of the screen analysis data is increased considerably, and the analysis of the project data can be started.



Screen name	Item count	Cyclic (<=500ms)	Cyclic (>500ms)	OtherCycles	Disabled	Tags-Dyn-Scripts	Tags-Dyn	Total Number of Tags	Resource list	Events	Child screens
01_ScreenLayout	7	0	0	0	0	0	0	0	0	1	02_Header&00_Navigation_0&&&03_MainNavigation
02_Header	18	0	1	0	0	0	1	1	0	8	
03_MainNavigation	4	0	0	0	0	0	0	0	0	5	
05_ThirdNavigation	7	0	0	0	0	0	0	0	0	9	
04_SubNavigation	9	0	0	0	0	0	0	0	0	11	
00_Navigation_2	7	0	0	0	0	0	0	0	0	5	

Figure 4-12 Finalized preparation of the screen analysis file

© Siemens 2024 |

4.1.4. Analysis of the screen overview file

This chapter contains information about how the data in the screen overview file can be related to the best practice topics mentioned in chapter 3 and the system limit information of WinCC Unified based devices which can be found in the respective manual of each version (see \9\).

NOTE For the following chapters be aware that the ShowScripts add-in currently does not support the analysis of objects and configurations inside faceplates but only the faceplate containers itself and its properties.

Faceplates must be therefore analyzed manually in the library view of the TIA Portal project.

4.1.4.1. Analysis of screen context

With the formatted screen overview file, it is possible to see the screen context (screens with subordinate screen window levels) very comfortably.

NOTE The screen context describes the total number of simultaneously existing screens windows with its objects, tags and dynamizations in WinCC Unified based applications.

In the example below you can see that the screen “02_ScreenLayout” has several child screens which means that the screen contains several screen windows with or without configured screens.

	Screen name	Item count	Cyclic (<=500ms)	Cyclic (>500ms)	OtherCycles	Disabled	Tags-Dyn-Scripts	Tags-Dyn	Total Number of Tags	Resource list	Events	Child screens	HmiFacepla
2	01_ScreenLayout	7	0	0	0	0	0	0	0	0		02_Header&00_Navigation_0&&&03_MainNavigation	
3	02_Header	18	0	1	0	0	0	1	1	0	8		
4	03_MainNavigation	4	0	0	0	0	0	0	0	0	5		
5	05_ThirdNavigation	7	0	0	0	0	0	0	0	0	9		
6	04_SubNavigation	9	0	0	0	0	0	0	0	0	11		
7	00_Navigation_2	7	0	0	0	0	0	0	0	0	5		
8	00_Navigation_3	7	0	0	0	0	0	0	0	0	5		
9	00_Navigation_4	7	0	0	0	0	0	0	0	0	5		

Figure 4-13 Example for screen window hierarchy in “child screens” column

Every child screen which means a screen window with or without configured screen is separated from another one with an “&” letter. If there are two “&” letters following on each other this means that there is a screen window in the respective screen which has an empty “screen” property like shown below.

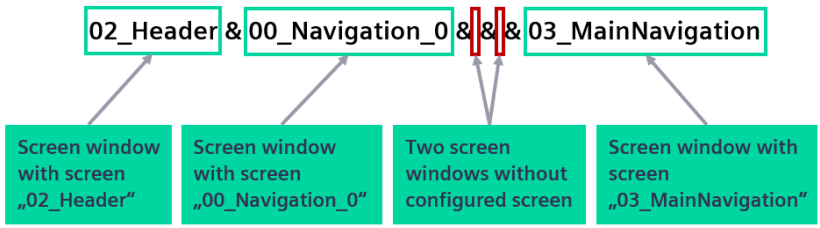


Figure 4-14 Explanation of “child screens” column based on “01_ScreenLayout” screen

© Siemens 2024

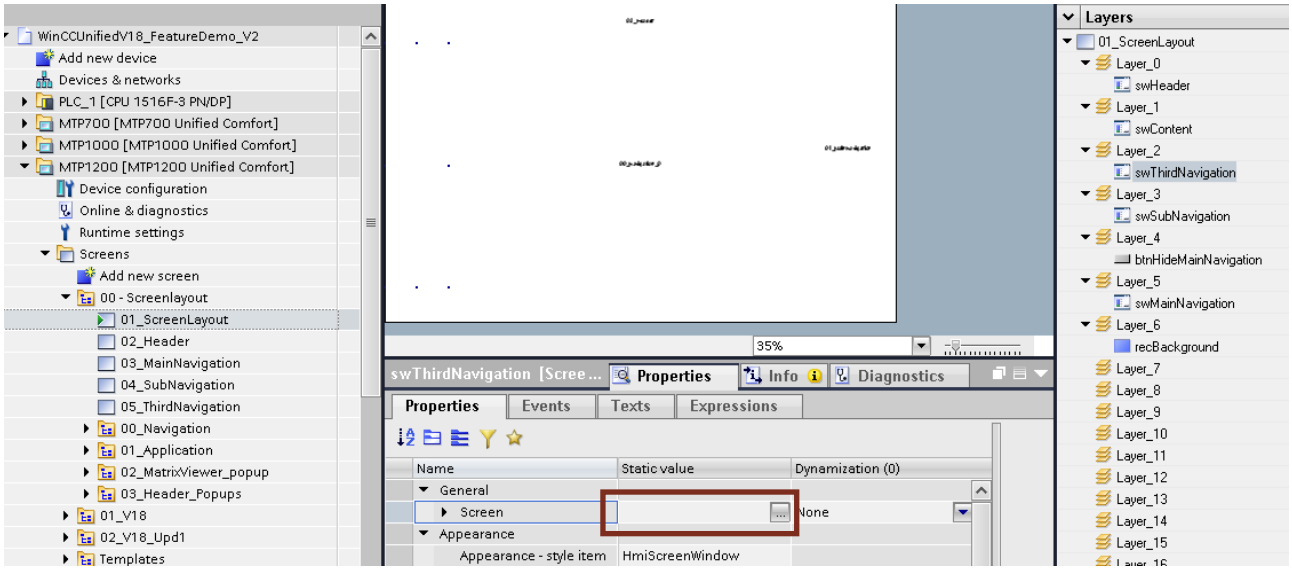


Figure 4-15 Example for screen window with empty screen property

Advantage of knowing the screen context structure

In the WinCC Unified manual further information about the system limits related to screens is listed. An extract was already mentioned in 3.1.2 Tidy up the screen / observe system limits .

NOTE Be aware that the screen related system limits mentioned in the manual always refer to the overall screen context which can contain several further screen window levels. This can result in a larger footprint on the runtime load due to the additional objects, tags and dynamizations.

Screens

	Unified Comfort 7-12"	Unified Comfort 15-22"
Maximum size in the engineering system	20,000 * 20,000 pixels	
Maximum size in runtime	20,000 * 20,000 pixels	
Number of screens	1200	
Number of lower-level screen windows	10	
Number of objects per screen	800	1200
Number of objects from the "Controls" area per screen	40	80
Number of tags per screen	600	800

Figure 4-16 Screen related system limits of a Unified PC station

In complex applications it can be hard to check the whole screen context in the engineering system. Same applies to the runtime application itself where screen windows can be invisible but still contain loaded screens with all its objects, tags and dynamizations.

The ShowScripts add-in provides a better understanding of the screen context in your application with the child screens information. This helps for avoiding overloaded screen contexts in an early stage of the project development or for later implementation of measures to reduce the load footprint by using the best practice recommendations in chapter 3.

Since the content of faceplate instances on screens is not evaluated yet by the ShowScripts add-in, the number of objects generated by them needs to be counted manually.

© Siemens 2024 |

4.1.4.2. Usage of Controls

The exported screen overview file can also give you further hints about screens containing controls which means basic, advanced and custom controls. As already mentioned in chapter 3.1.1 the rendering footprint of controls can be much higher than other objects as basic objects and elements.

If you detect the usage of several controls within one screen or screen context evaluate a possible reduction of these controls like described in chapter 3.1.5. Wherever possible from operating point of view, splitting controls in different screens or screen contexts can also have a positive impact on the loading time of a screen. Especially two controls of the same type like in the figure below (marked with 1), can often be combined by one.

	Screen name	Item count	Child screens	HmiAlarmControl	HmiMediaControl	HmiTrendControl	Hmi
2	01_ScreenLayout	7	02_Header&00_Navigation_0&&&03_MainNavigation	0	0	0	
3	02_Header	18		0	0	0	
4	03_MainNavigation	4		0	0	0	
5	05_ThirdNavigation	7		0	0	0	
6	04_SubNavigation	9		0	0	0	
7	00_Navigation_2	7		0	0	0	
8	00_Navigation_3	7		0	0	0	
9	00_Navigation_4	7		0	0	0	
10	00_Navigation_5	7		0	0	0	
11	00_Navigation_6	7		0	0	0	
12	00_Navigation_7	7		0	0	0	
13	00_Navigation_0	8		0	0	0	
14	00_Navigation_1	7		0	0	0	
15	10_Application_Auditing	1		0	0	0	
16	10_Application_Communication	1		2	0	1	
17	10_Application_Faceplates	1		0	0	0	
18	10_Application_ModernUI	1		0	0	0	
19	10_Application_PaCo	1		0	0	0	

Figure 4-17 Information about control usage per screen

4.1.4.3. Usage of Dynamizations

There is even more object related information that can be analyzed by use of the screen overview file of the ShowScripts add-in. You can find information about tags, dynamizations and events configured on each screen. This helps you to get an overview on which part of the application to look at when opening the TIA Portal project for further analysis, e.g. when cyclic scripts are used on several screens (refer to section 3.2).

The information can be also taken as reference if you want to have a closer look on the scripts of certain via Visual Studio Code (see 4.1.5).

The related columns in the screen overview file will be elaborated more in detail in the following sections:

Used tags per screen

The "total number of tags" is the sum of all tags that are used as trigger on the screen. These trigger tags will be subscribed to the PLC on screen loading and will influence the screen loading time. Trigger tags are used in dynamizations only.

1	Screen name	Total Number of Tags
165	69_a_Subscription	4
166	69_b_RootWindow	0

Figure 4-18 Total tags column in overview file

NOTE Expressions for the dynamization of objects (available since WinCC Unified V18) cannot be evaluated by the ShowScripts add-in yet and are therefore not included in the counts of the screen overview file.

Examples for the described dynamizations can be found in chapter 3.2 Use of dynamizations.

Tag dynamizations

The "Tags-Dyn" column lists the amount of tag dynamizations configured on a screen.

© Siemens 2024 |

1	Screen name	Tags-Dyn
165	69_a_Subscription	4
166	69_b_RootWindow	0

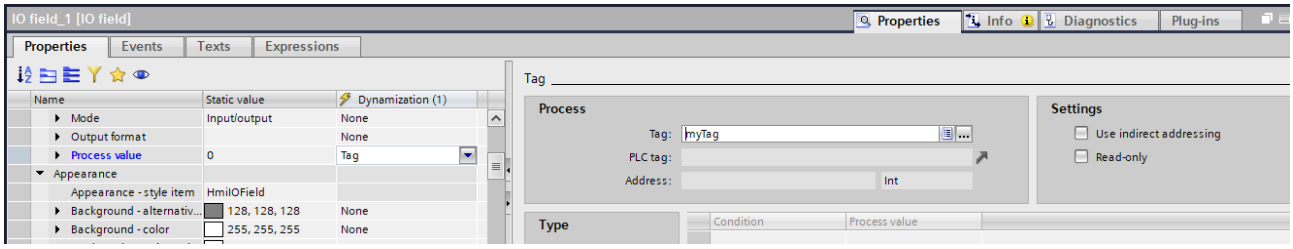


Figure 4-19 Tag dynamization column in overview file

Script dynamizations (tag triggered)

The "Tags-Dyn-Scripts" column lists the amount of script dynamizations based on a tag trigger per screen.

Screen name	Tags-Dyn-Scripts
01_LiveDemo_JS	2
02_HandsOn_JS	2

Figure 4-20 Script dynamization (tag triggered) column in overview file

Script dynamizations (cyclic)

There are three columns referring to cyclic script dynamizations based on default available and own defined cycles.

Screen name	Cyclic (<=500ms)	Cyclic (>500ms)	OtherCycles
01_LiveDemo_JS	0	0	0
02_HandsOn_JS	0	0	0

Figure 4-21 Script dynamization (cyclic) columns in overview file

Events

The "Events" column lists the sum of following configurations:

7. Events on screens and screen items
8. The amount of "on-change" triggered scripts at properties of screen items

Screen name	Events
01_LiveDemo_JS	6
02_HandsOn_JS	1

Figure 4-22 Events columns in overview file

© Siemens 2024 |

4.1.5. Analysis of the exported scripts

Next to the screen overview file the ShowScripts add-in exports all configured scripts within a screen and its objects.

As shown below two export files (js-format) for each screen are generated:

1. “Dynamizations” file: contains all script dynamizations configured on a screen and its objects.
2. “Events” file: contains all “on-change” triggered scripts (on screen object properties) and scripts triggered on events (of screens and objects)

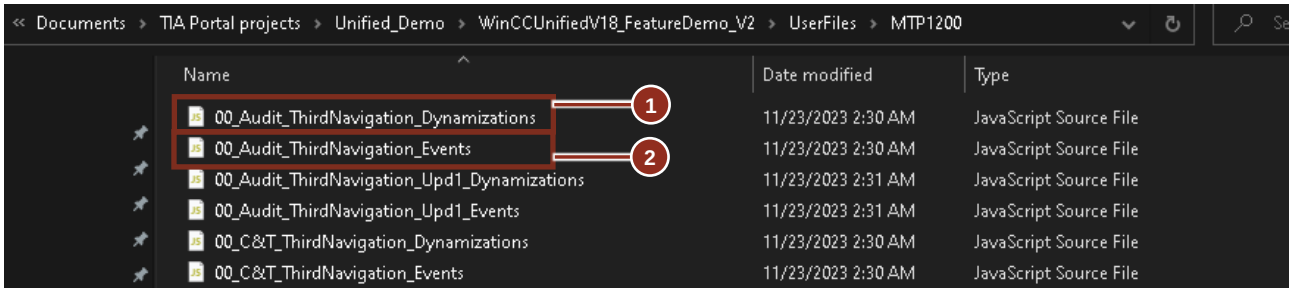


Figure 4-23 Script export files of ShowScripts add-in

NOTE

Currently the ShowScripts add-in does not support the export of scripts inside faceplates, global modules, scheduled tasks and libraries. The scripts inside faceplates and scheduled tasks must be analyzed separately in the TIA Portal project.

For the global module and library scripts there is an alternative since WinCC Unified V19 to retrieve script exports by use of the WinCC Unified JS Connector (see 4.2)

Script analysis using VS Code

All exported scripts can be analyzed quite comfortably according to the best practice recommendations in chapter 3.

This can be done with any editor like Notepad++ for each file separately or if you want to execute a global search through all the script files by use of the free of charge code editor “Visual Studio Code” (VS Code).

Therefore, download VS Code and once it is installed follow the further steps.

1. Open the program and drag & drop the whole export folder of ShowScripts add-in from the Windows Explorer into the VS Code “Explorer” area.

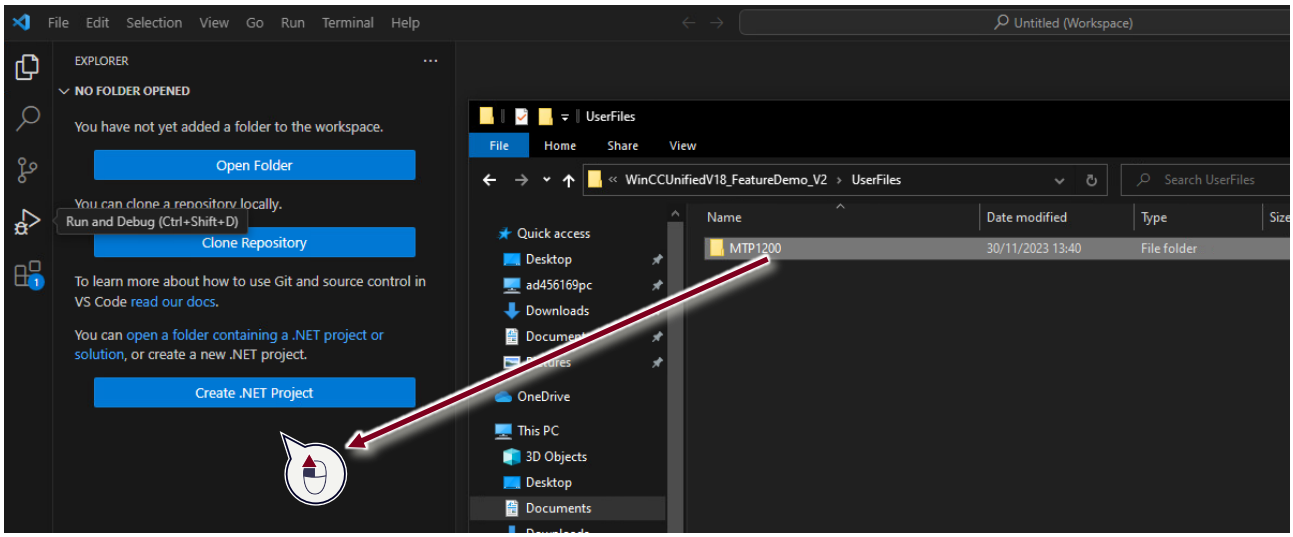


Figure 4-24 Opening script export folder with VS Code

© Siemens 2024 |

2. Afterwards the content of the export folder will get listed in the left area of VS Code (1). As soon as you select any of the script export files the content will be displayed in the work area of the application (2).

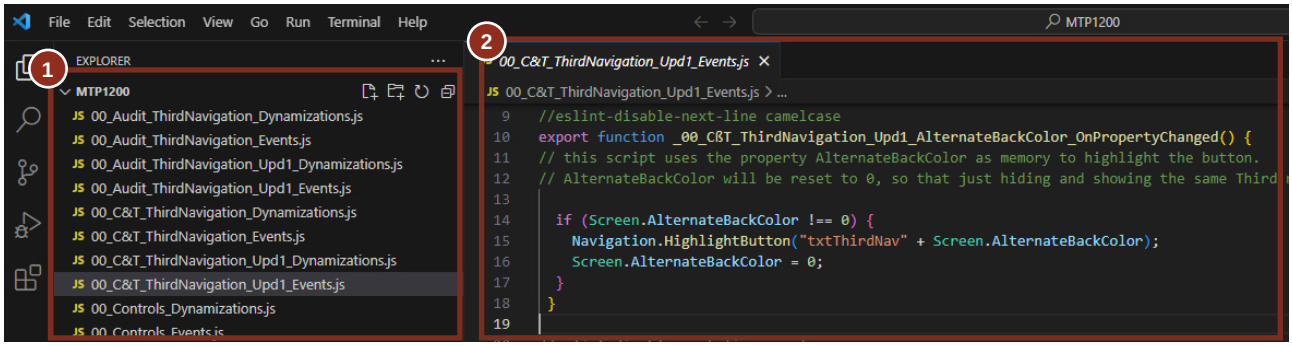


Figure 4-25 Opened script folder in VS Code

3. The identification of the script type within WinCC Unified is also possible via the function title. In the figure below two examples are shown:

   a. "On-change" triggered script on the property *"AlternateBackColor"* of the screen *"00_C&T_ThirdNavigation_Upd1"*
   b. "Click left mouse button" event on the object *"txtThirdNav1"*

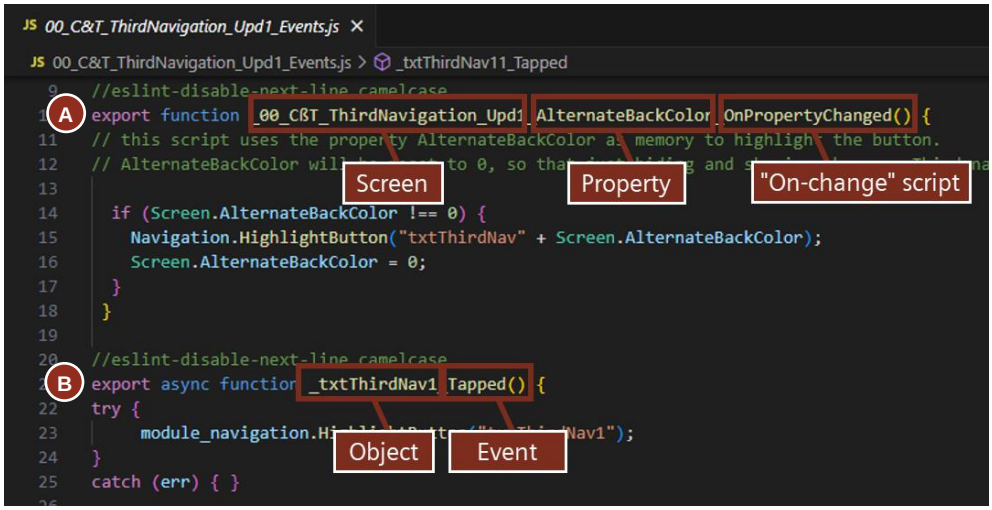


Figure 4-26 Extract of script export in VS Code

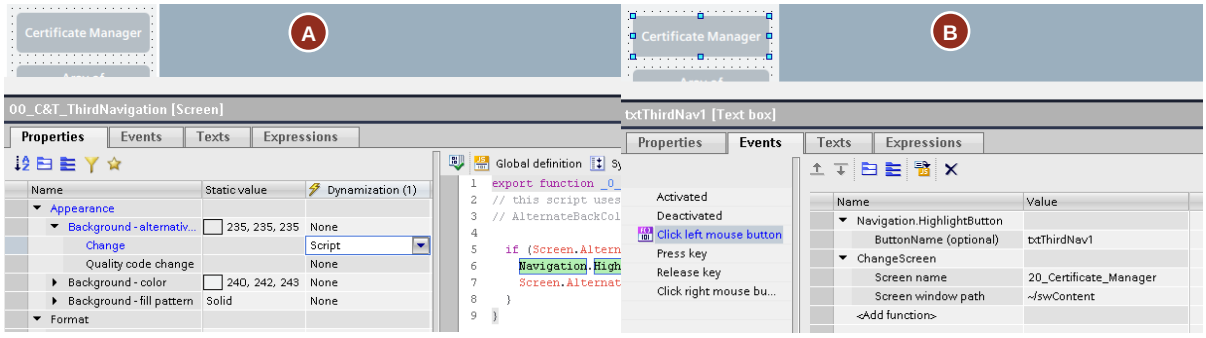


Figure 4-27 Location of the exported scripts in TIA Portal project

NOTE Be aware that the ShowScripts add-in also exports function lists as scripts (see example B). The reason is that system functions are only engineering related but during runtime they get executed as script code as well.

4. By the use of the global search function of VS Code the exported scripts can be analyzed according to the best practice recommendations in chapter 3.
5. In the example below the scripts get scanned for example for the keyword "SetTagValue" to be able to find relevant scripts for the replacement with tag sets as described in chapter 3.4.2.1.

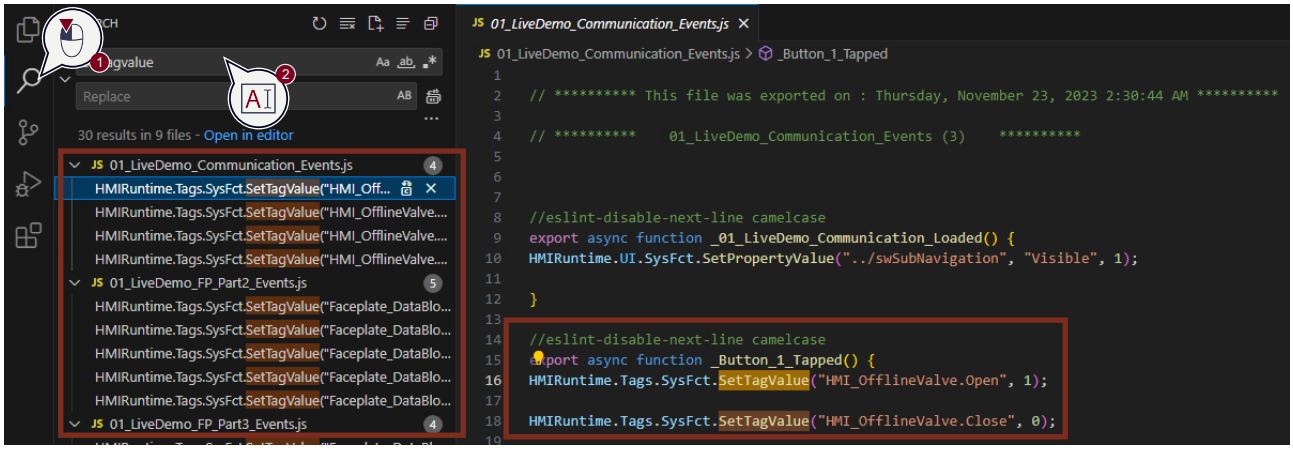


Figure 4-28 Keyword search in VS Code for "SetTagValue"

Other examples to search for could be among others

- « for » to find unnecessary loop runs
- « alarm » for the usage of computing-intensive methods, e.g. GetActiveAlarms()
- « await » Numerous uses of await in connection with asynchonous function calls

NOTE Furthermore, Visual Studio Code can use the style guide configuration, that offers verification and automatic correction of script code inside the Visual Studio Code editor according to the programming style guide specifications. The style guide configuration can be found also on SIOS

© Siemens 2024 |

4.2. WinCC Unified JS Connector V19

A first step for the screen related script export and analysis can be done by use of the ShowScripts add-in. If you detect the usage of global module scripts in the screen related scripts you can make use of the “Simatic WinCC Unified JS Connector” which is a Visual Studio Code extension to export, edit and import these scripts of the global modules.

NOTE The WinCC Unified JS Connector extension is supported since WinCC Unified ES V19.

Below you can see an example for the usage of global modules. If you want to see the content of the used scripts in the imported module the WinCC Unified JS Connector can be used.

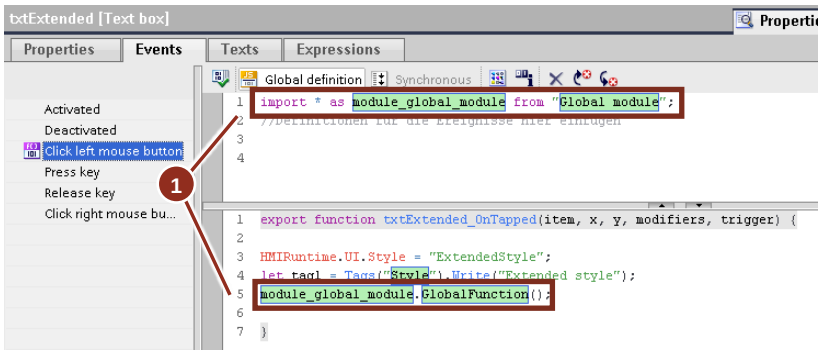


Figure 4-29 Usage of a global function in WinCC Unified script editor

The WinCC Unified JS Connector extension is free of charge and available in a separate entry of the Siemens Industry Online Support (SIOS). With the extension following workflow can be applied by users for project analysis and optimization:

1. Export of scripts in global modules from the TIA Portal project

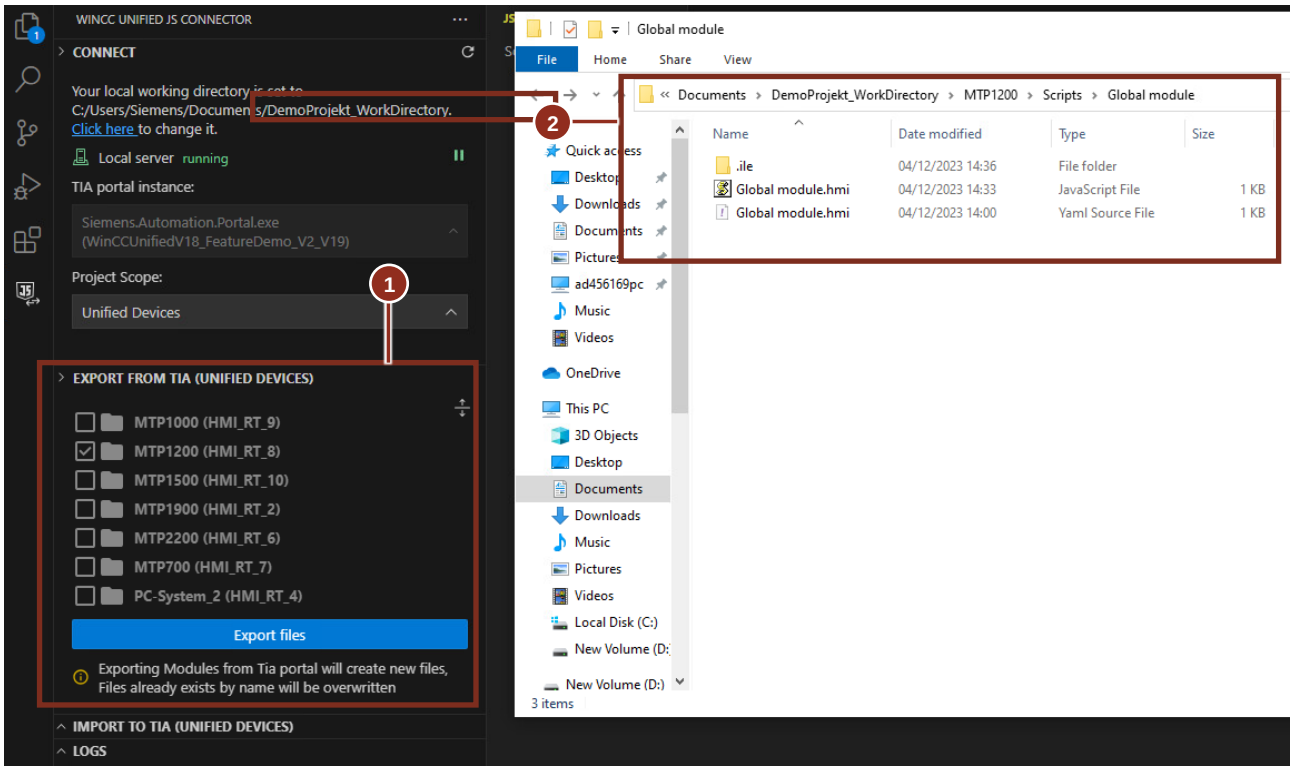


Figure 4-30 Export function of the extension (1) with the exported files in the work directory (2)

© Siemens 2024

2. Analysis and adaption of the scripts in VS Code (according to the best practices in chapter 3)

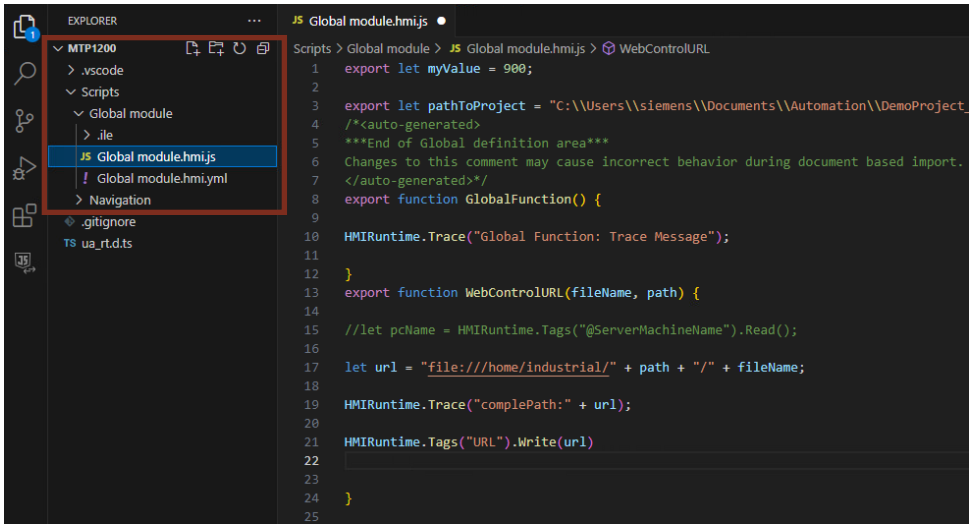


Figure 4-31 Analysis of an exported global module script via VS Code

3. Import of the adapted scripts to the TIA Portal project

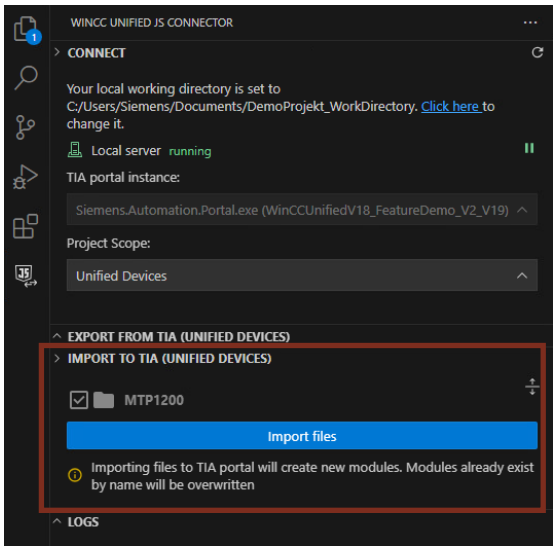


Figure 4-32

For more detailed information about the installation and usage of the WinCC Unified JS Connector please refer to the provided manual in the extension related SIOS entry.

© Siemens 2024

4.3. Runtime analysis

If you have successfully analyzed the project using the above mentioned ShowScripts add-in and the WinCC Unified JS connector and applied changes and optimizations in the screen engineering or script code, it is possible that there are still misbehaviors during screen change or the screen loading time is still not sufficient enough.

To find further potential for project optimization it can be useful to analyze problematic screen change more in detail by use of following tools:

1. RTIL Trace Viewer (for all Unified based devices)
2. Script Debugger for WinCC Unified (only for Unified PC Runtime and Panel Simulation at the Engineering PC)

4.3.1. RTIL Trace Viewer

The RTIL Trace Viewer is an external application provided with the WinCC Unified installation. It displays all available trace messages that a Unified Basic/Comfort Panel or PC Runtime provides during runtime operation.

NOTE

The setup procedure for using the RTIL Trace Viewer on WinCC Unified based devices is described more detailed in the following FAQ:

Use of Trace Viewer with WinCC Unified Comfort Panel or WinCC Unified PC Runtime

The procedure described for the Unified Comfort Panels also applies the identically for the Unified Basic Panels.

Usage of the RTIL Trace Viewer for project analysis

The Trace Viewer can be used for further analysis of following scenarios:

1. When capturing a screen change there can be further errors or warnings which are related to script execution or in general to the project configuration.
   These trace messages can show you a certain influence on the overall screen change performance and need to be resolved as far as possible.

   An example for such errors is visible below in which a configured script tries to access an object (“Button_4”) in the runtime which does not exist and therefore error messages are shown.

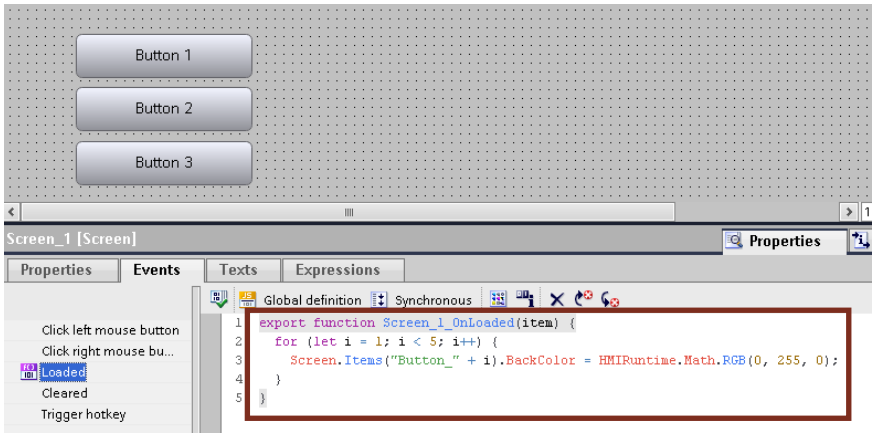


Figure 4-33 Script access to buttons on a screen

   The execution result of the script is shown understandably in the RTIL Trace Viewer and needs to be resolved by adapting the loop in the script.

© Siemens 2024

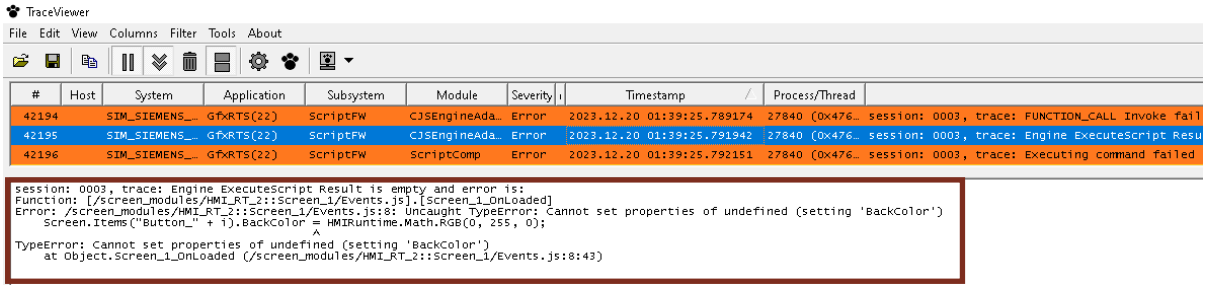


Figure 4-34 Result of script execution in the trace viewer application

NOTE For better analyzability of the traces in the RTIL Trace Viewer, special display filters can be applied to the default view. In order to see just the script related traces you can set the filter for “Subsystem” to “ScriptFW”.

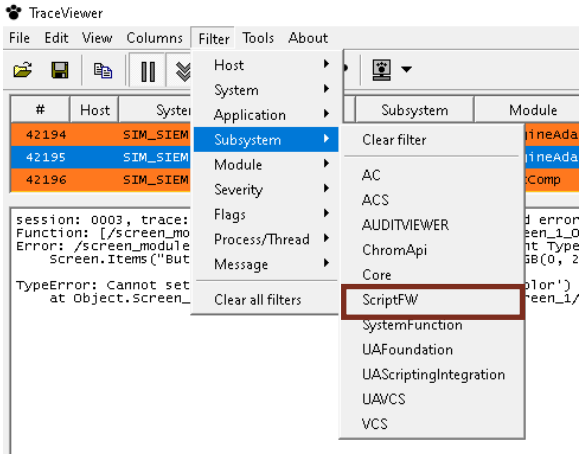


Figure 4-35 Script filter in RTIL TraceViewer

2. The Trace Viewer can be an alternative to detect and understand possible failures of scripts in the project. As there is no further debugging possibility on the Unified Basic/Comfort Panel hardware the usage of trace messages in conjunction with the RTIL Trace Viewer can help to:

- See which scripts get executed in certain operating scenarios
- Check how often these scripts are executed
- How long the scripts are being processed

For the Unified PC Runtime, a useful alternative to the Trace Viewer could be the Script Debugger which is described in the next chapter.

NOTE Further information about the usage of the trace viewer and configuration of trace messages can be found in the document SIMATIC WinCC Unified - Tips and Tricks for Scripting (JavaScript) in chapter 5.16.2 “Diagnosis with RTIL Trace Viewer”.

4.3.2. Script Debugger

On a Unified PC Runtime and Panel Simulation at Engineering PC there is the possibility to use the Script Debugger in combination with Google Chrome.

The debugger offers you further possibilities for script analysis offering typical functions like setting breakpoints and step-by-step execution.

NOTE A detailed description about the setup and usage of the Script Debugger in WinCC Unified is available in the WinCC Unified Engineering (V18) manual.

Usage of the Script Debugger for screen change analysis

With the Script Debugger of the Unified PC Runtime certain operation scenarios can be further evaluated to:

- See which scripts get executed in certain operating scenarios. When a breakpoint is set e.g. in a loaded event, by clicking through the single steps, also all following script executions are shown.
- Check how often these scripts are executed, by counting how often a breakpoint is reached in a script.

If you want to debug the script executions during the screen change the project needs to be prepared in a certain manner to be able to debug the simultaneously executed scripts. The following steps show how to configure a breakpoint in the load event of the base screen and trigger the loading again for stepping through all scripts that are executed during the initial screen load as this is the more difficult configuration from the two mentioned above.

1. Activate the script debugger in the runtime manager

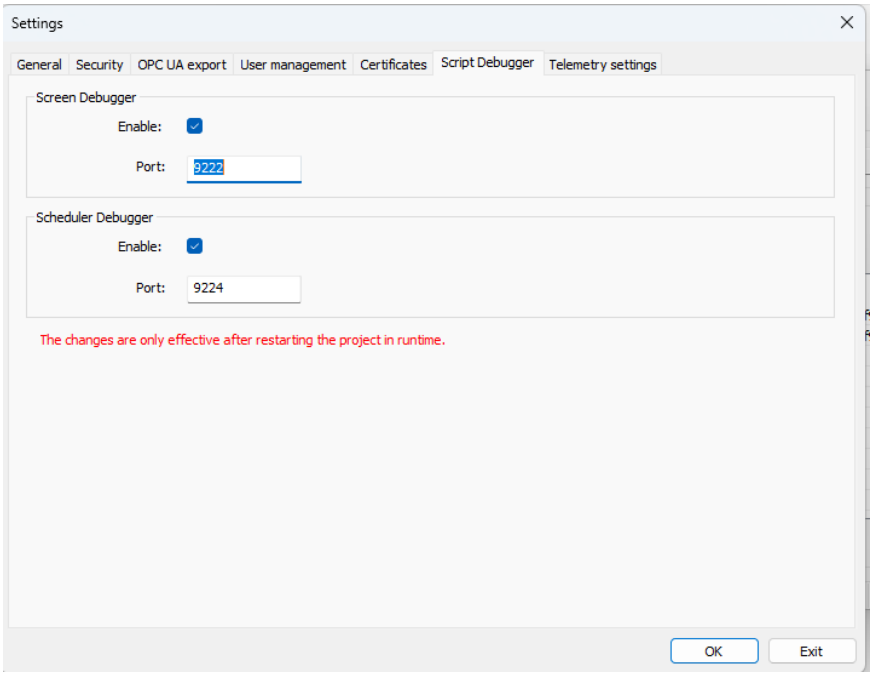


Figure 4-36 Enable script debugger in runtime manager settings

2. Configure a code in the loaded event of the projects base screen. This is used later to set a break point at the position where the base screen starts loading. Define an event that you can trigger during runtime (e.g., through a mouse click event of a screen item), that toggles the base screen from one that is not your current base screen and then back to the original base screen. If you only set the base screen again to the original one, the loaded event will not be triggered.

© Siemens 2024

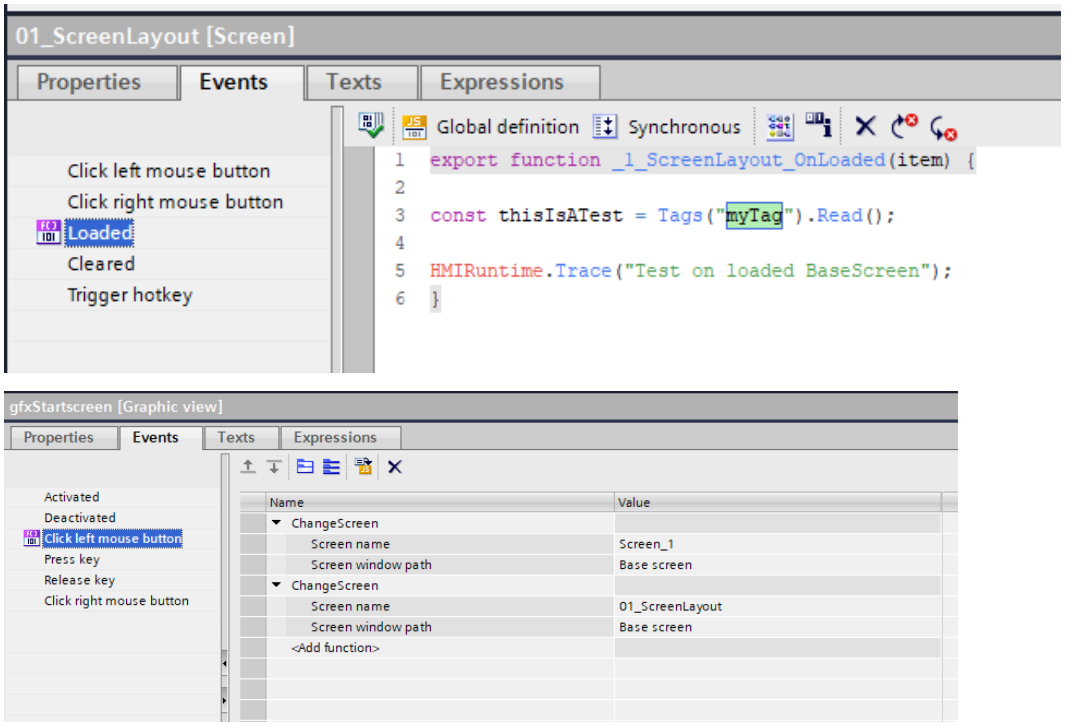


Figure 4-37 Refresh of the base screen and code for the loaded event

3. Start the Runtime
4. Open an additional browser window and type: chrome://inspect/

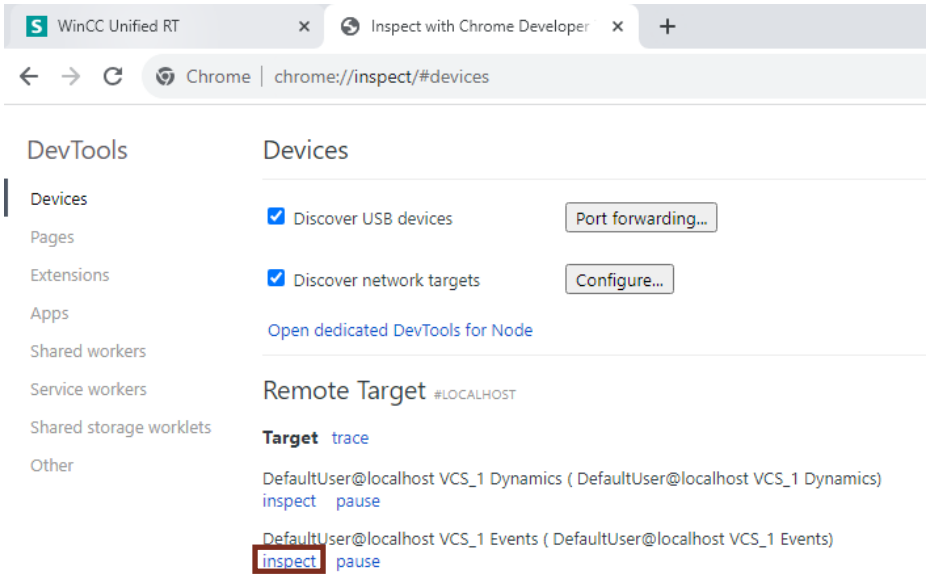


Figure 4-38 Chrome dev tools page

5. Click on the inspect link of the event context. With that, a new browser client is started

© Siemens 2024 |

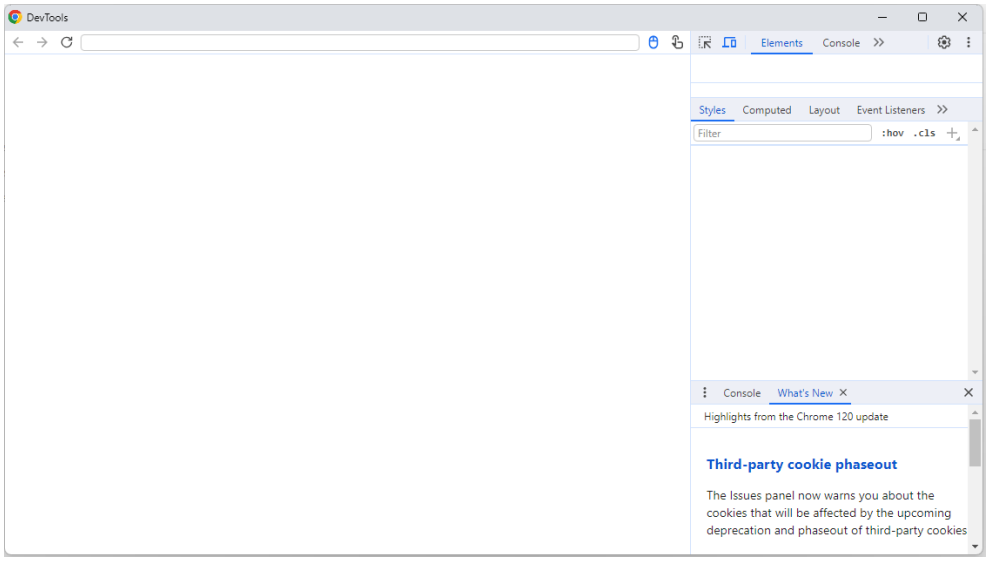


Figure 4-39 DevTools browser window

6. Configure the browser window by dragging the right window more to the left and switch to the source tab. Unfold the (no domain) directory and select your base screen Event.js. There is the previous defined code in the loaded event of the base screen. The list shows all current loaded scripts. The list therefore differs depending on the current opened screens.

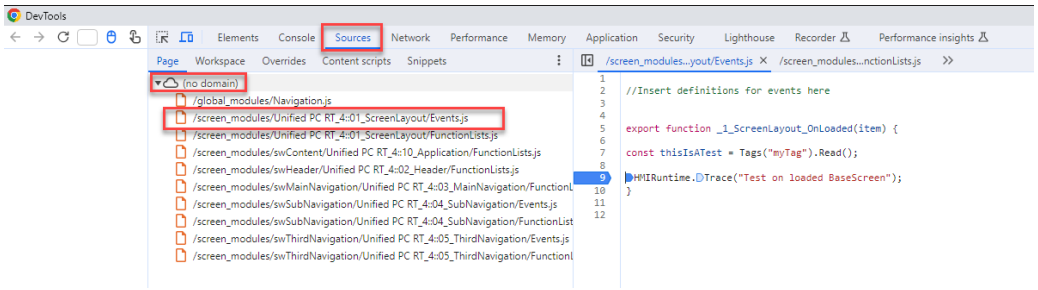


Figure 4-40 Script debugger - set break point

7. Set a break point in the loaded event
8. Go back to the Runtime and trigger the event for the base screen change
9. The debugger browser windows will automatically open and jump to the breakpoint. You can also see on the right side the values of the already executed lines. Klick on the “Step into next function” button to go step by step through the script

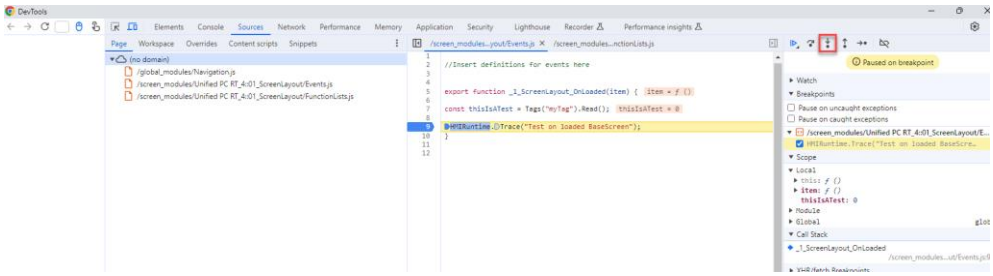


Figure 4-41: Script debugger - jump into break point

You can follow that global modules are loaded

© Siemens 2024 |

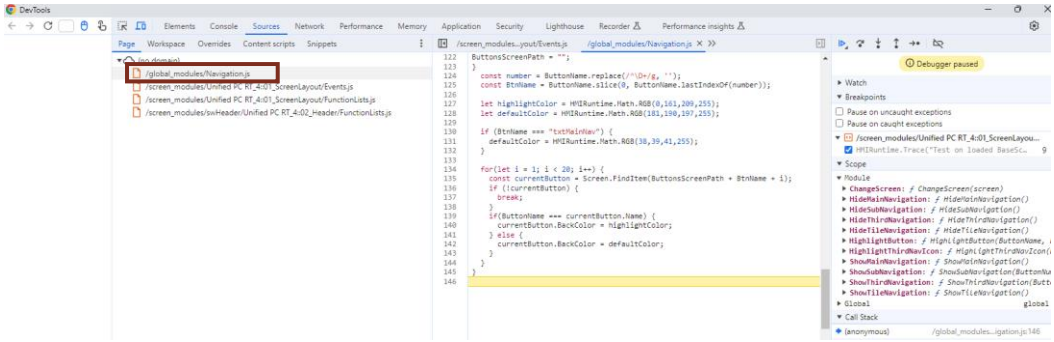


Figure 4-42 Script debugger - loading script modules

And loaded event of nested screen windows and faceplates

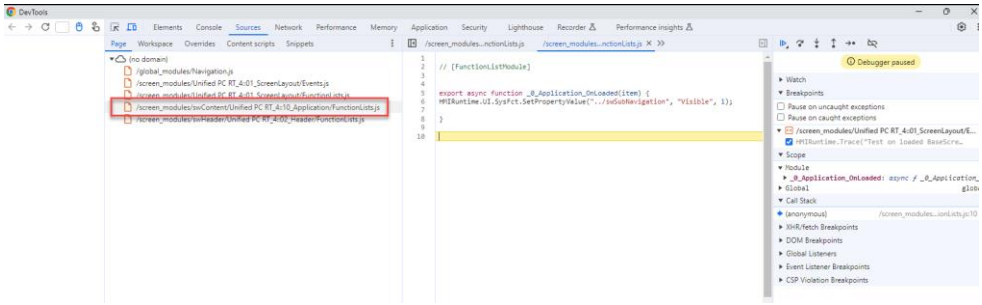


Figure 4-43 Script debugger - loaded event screen window

Every script that is executed will be shown. Once you want to end the debugging and execute the rest of the scripts at once, click on the “Resume script execution” button. If there are more breakpoints defined, the debugger will jump to the next breakpoint.

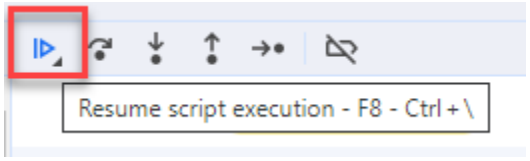


Figure 4-44: Resume script execution

10. For every new download a new DevTools session needs to be opened. That is the reason for the manual trigger of the loaded event of the base screen

© Siemens 2024 |

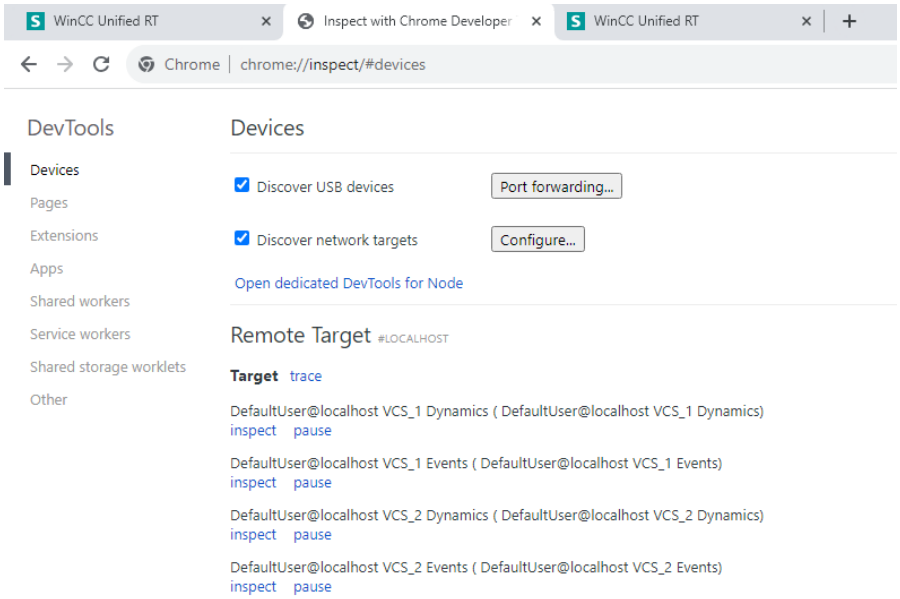


Figure 4-45 Google Chrome inspect page for selection runtime session

© Siemens 2024 |

5. Appendix

5.1. Service and support

SiePortal

The integrated platform for product selection, purchasing and support - and connection of Industry Mall and Online support. The SiePortal home page replaces the previous home pages of the Industry Mall and the Online Support Portal (SIOS) and combines them.

- Products & Services
  In Products & Services, you can find all our offerings as previously available in Mall Catalog.
- Support
  In Support, you can find all information helpful for resolving technical issues with our products.
- mySieportal
  mySiePortal collects all your personal data and processes, from your account to current orders, service requests and more. You can only see the full range of functions here after you have logged in.

You can access SiePortal via this address: sieportal.siemens.com

Technical Support

The Technical Support of Siemens Industry provides you fast and competent support regarding all technical queries with numerous tailor-made offers – ranging from basic support to individual support contracts.
Please send queries to Technical Support via Web form: support.industry.siemens.com/cs/my/src

SITRAIN – Digital Industry Academy

We support you with our globally available training courses for industry with practical experience, innovative learning methods and a concept that's tailored to the customer's specific needs.
For more information on our offered trainings and courses, as well as their locations and dates, refer to our web page: siemens.com/sitrain

Industry Online Support app

You will receive optimum support wherever you are with the "Industry Online Support" app. The app is available for iOS and Android:

© Siemens 2024 |

5.2. Links and literature

Nr.	Thema
\1\	Siemens Industry Online Support https://support.industry.siemens.com
\2\	Link to this entry page of this application example https://support.industry.siemens.com/cs/ww/en/view/109827603
\3\	Beginners Guide: Quick entry and conversion with WinCC Unified https://support.industry.siemens.com/cs/ww/en/view/109810917
\4\	HMI design with the HMI Template Suite https://support.industry.siemens.com/cs/ww/en/view/91174767
\5\	SIMATIC WinCC Unified Tutorial Center (Videos) https://support.industry.siemens.com/cs/ww/en/view/109782433
\6\	SIMATC WinCC Unified – Tips and Tricks for Scripting (JavaScript) https://support.industry.siemens.com/cs/ww/en/view/109758536
\7\	TIA-Add-In-ShowScripts https://github.com/tia-portal-applications/TIA-Add-In-ShowScripts
\8\	Developing WinCC Unified JavaScript code and checking style guide with Visual Studio Code https://support.industry.siemens.com/cs/ww/en/view/109801600
\9\	SIMATIC HMI WinCC Unified Engineering for system limits https://support.industry.siemens.com/cs/ww/en/view/109813308/160577778571
\10\	SIMATIC WinCC Unified Corporate Designer V19 https://support.industry.siemens.com/cs/ww/en/view/109824234

5.3. Change documentation

Version	Date	Modification
V1.0	01/2024	First version

© Siemens 2024 |