Wizard Studio Manual

Rev. 2.02 156 - 13/03/2017

1 Getting started

Wizard is a software application designed to create graphical HMI pages. Wizard has a drag-and-drop interface that makes it easy to create complex pages. Many of the features found in common Windows applications are also available in Wizard.

This document is divided into chapters that describe the key functions of Wizard and explain how to use them. Each chapter is presented in a standalone manner, allowing you to jump from chapter to chapter, depending on the task at hand.

Assumptions

We assume that readers have a basic understanding of computers, Microsoft Windows, and the specific network environment where the application will run.

Installing the application

Wizard installation contains:

- Wizard: an application for designing custom HMI projects in a user-friendly manner, along with a variety of objects in its built-in library, the Widget Gallery.
- HMI Client: a light-weight application that can be used on Windows computers to remotely view and manage a project running on an HMI device.
- HMI Runtime: a standalone application that runs on the HMI devices. The HMI Runtime is installed via Wizard.
- BTM-PCRUNTIME: a standalone application that runs on Win32 platforms (computers instead of HMI devices).

Wizard system requirements

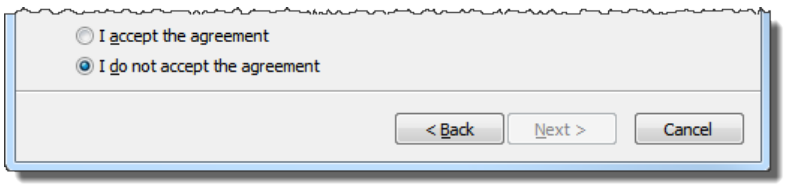
Wizard has the following system requirements:

Operating System	Windows XP (SP2 or SP3) Windows Vista Business/Ultimate Windows 7 Windows 8 Windows 10
Storage	500 MB Minimum
RAM	512 MB
Other	One Ethernet connection

Installation procedure

To install Wizard:

1. Run Wizard setup and click **Next**.
2. Read the Wizard Software License and accept the agreement.



3. Follow the instructions on the screen. The default location for the c software is *C:\Program Files\CG\Wizard*, change path if needed.
4. If the Select Components step is available, select the components you want to install.
5. Select the **Create a desktop icon** option to add a Wizard icon on your desktop. A Wizard group is automatically added to the **Start** menu by the installation procedure.

6. To run the application click the desktop icon or choose **Start** > **All programs** > **Wizard**.

Installing multiple versions of Wizard

You may install different instances of Wizard on the same computer. Each installation has its own settings and can be uninstalled individually.

Three installation scenarios are possible:

Installation scenario	Results
First installation of Wizard in the system	Software is installed in the specified destination folder
System with only one instance of Wizard already installed	Current version can be replaced or maintained.
System with multiple instances of Wizard already installed	Last version installed can be replaced or maintained.

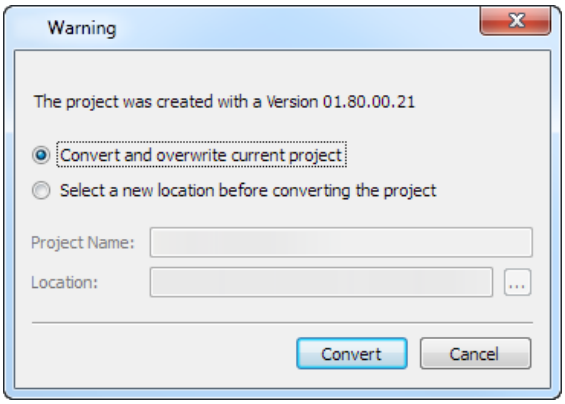
If you try to install a second instance of an already installed version of Wizard, a warning message is displayed.

Multiple Wizard installations share a common workspace folder, each sub-folder includes the version number, for example *C:\Program Files\CG\Wizard 2.5*. Each installed version has its ID and can therefore be removed individually.

Each installation is listed separately in the Windows **Start** menu.

Opening older projects

When opening a Wizard project (.jpr file) created with an older version of the software Wizard asks to convert the project to the current version:



Option	Description
Convert and overwrite current project	The project is converted without a backup copy of the original version
Select a new location before converting the project	The project is copied inside the specified folder and then converted.

WARNING: Do not edit projects with a version of Wizard older than the version used to create them. This will damage the project and may cause runtime instability.

Multilanguage for Wizard

Wizard is available in multiple languages. All languages are installed by default as part of Wizard.

The default language is English. To change it go to **Help** > **Change Language**.

Crash reports

A crash report dialog appears whenever Wizard freezes or crashs.

Important: Always save crash report files since they may contain useful information for technical support.

Note: Crash reports are unavailable in Windows XP.

2 Runtime

HMI Runtime is designed to support different platforms and different operating systems.

HMI device basic settings

HMI devices are delivered from factory without Runtime. If no Runtime is installed on the device, see "The Runtime loader" on page 75 for details.

Runtime modes

The HMI Runtime is composed of two logic units:

- **Server**: runs communication protocols, collects data, monitors alarms, drives trend buffer sampling.
- **Client**: displays data collected by server.

The server unit is responsible for handling the HMI services such as the communication protocols, performing data acquisition, driving trend buffer sampling activities, monitoring alarms, and so on.

The client unit is the part which is responsible for the visualization process: use the data collected by the server to render it on the display as graphical information.

The server unit works in two operating modes:

- **Configuration mode**: server is idle (for example when no project is loaded on the device or some system files are missing).
- **Operation mode**: server is operating according to the settings defined by the system files and by the loaded application project.

Note: Data on client may be displayed even if no activity is running on the server.

Context menu options

On the HMI device press and hold on an empty area of the screen for a few seconds to display the context menu.

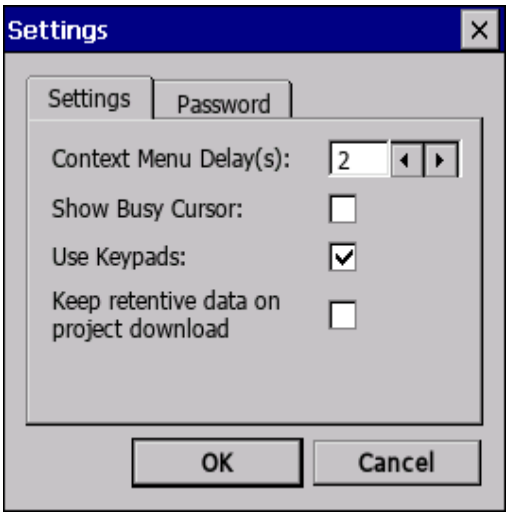
Zoom In/Out

Select view size at run time

Pan Mode

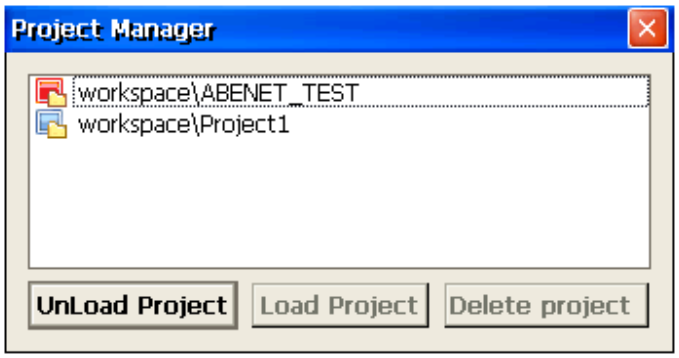
Enables/disables pan mode after a zoom in

Settings



Main parameters	Description
Context Menu Delay (s)	Context menu activation delay. Range: 1–60 seconds.
Show Busy Cursor	Display an hourglass when the system is busy
Use keypads	Display keypads when user touches a data entry field. Set to **disable** when an external USB keyboard is connected to the device.
Keep retentive data on project download	Preserve the content of the retentive data at project download or update.
Password	Define password protected operations amongst the following: • Download Project/Runtime • Upload project • Board management (BSP Update) See "Protecting access to HMI devices" on page 415 for details.

Project Manager



This tool allows you to:

- unload the current project
- load another project
- delete a project.

When you load a new project, the current project is automatically unloaded. You must unload a project before you can delete it.

Update

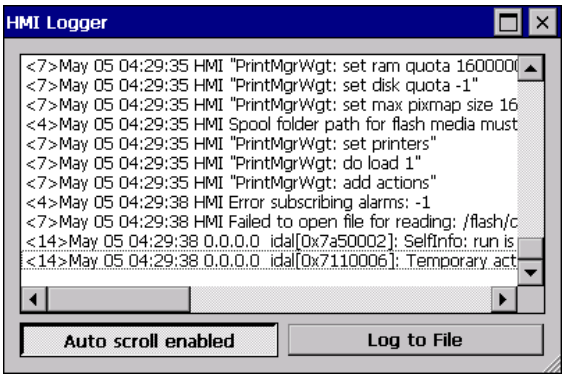
This function loads update packages from an external USB drive. See "Update package" on page 72for details.

Backup

You can create a backup copy of the Runtime and of the project.

Logging

This function displays a log of system operations.



Click **Log to file** to save data: a logger.txt file is saved to the *...\var\log* folder.

This file can be retrieved using an FTP Client and forwarded to technical support.

Note: Once enabled, logging is maintained after power cycles and must be manually disabled.

Show log at boot

This function enables the logger at start up. If the **Log to file** option has been enabled, log files are saved from startup.

Logout

Logs off the current user.

Show system settings

Allow the HMI settings and the management of system components. See "System Settings" on page 373 for details.

Developer tools

Utility functions for debugging at run time. It is visible only if enabled in the Project Properties (see "Developer tools" on page 56 for details)

About

This function shows information about the Runtime version.

WARNING: Context Menu action has no effect if executed from a dialog page.

Built-in SNTP service

The HMI device features an integrated SNTP that synchronizes the internal real-time clock panel whenever the predefined server is available. The system searches the SNTP server when turned on, or once a week if the HMI device is not turned off.

Use HMI device "System Settings" on page 373to configure the service.

Availability: BSP v1.76 ARM / 2.79 MIPS or higher

2 Runtime on PC

BTM-PCRUNTIME for Windows is an HMI platform that combines advanced HMI features and vector graphics with powerful web technologies. You can choose this platform to monitor and control your equipment with tags, alarms, schedulers, recipes, trends, Javascript logic and events.

BTM-PCRUNTIME provides connectivity with factory and building automation protocols, based on Ethernet and serial interfaces.

BTM-PCRUNTIME system requirements

BTM-PCRUNTIME as the following minimum system requirements:

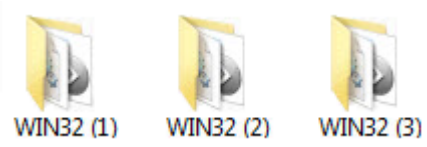
Operating System	Windows XP Professional
	Windows XP Embedded
	Windows Embedded Standard (WES 2009)
	Windows Vista Business/Ultimate
	Windows 7 Professional
	Windows Embedded Standard 7
	Windows 8
	Windows Server 2003
Storage	256 MB Min
RAM	512 MB
CPU	min. 300 MHz Pentium III or similar processors with 500 MHz.
Graphic	min. SVGA
Other	One Ethernet connection

Installing Runtime

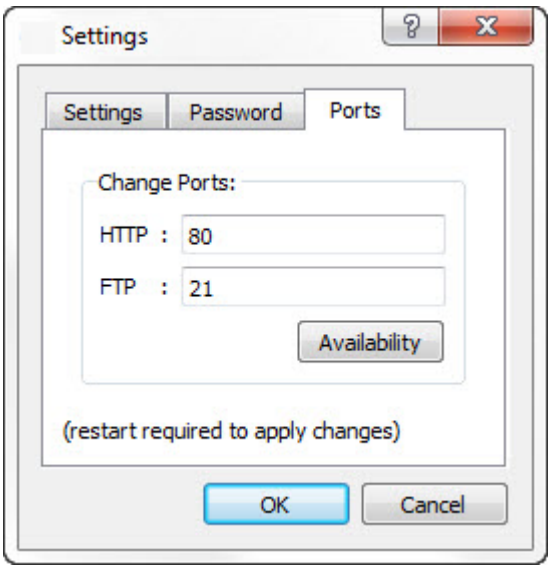
BTM-PCRUNTIME could be distributed as a component of the Wizard or as a standalone application. When installing the software make sure that you select the **Runtime PC** option in the **Select Components** dialog.

Multiple instances of BTM-PCRUNTIME

BTM-PCRUNTIME can run in multiple instances. Copy the installation folder to a writable location and double-click on the HMI application in each folder to start it.



The port used by BTM-PCRUNTIME can be changed from the **Settings** dialog. Restart the application to apply the port change.



Limitations

The following features are not supported in BTM-PCRUNTIME:

Function	Feature NOT supported
Manage Target	Board section
System Mode/ User Mode	Tap sequence and rotating menu
VNC/PDF readers	Non-standard computer software
Backup/Restore	Backup and restore functions. Standard computer software can be used for the purpose.
Protocols	Serial protocols requiring special hardware.

See "Functional specifications and compatibility" on page 427 for more details.

Fullscreen mode

BTM-PCRUNTIME can start in fullscreen mode or in a window.

To switch to full screen:

1. Right click in the BTM-PCRUNTIME main window to display the context menu.
2. Choose **Full Screen**.

The workspace folder

When using BTM-PCRUNTIME, project files are stored in a workspace folder in:

%appdata%\CGC S.p.A.\[build number]\server\workspace

where [build number] is a folder named as build number (for example, 01.90.00.608).

Typical installation problems

Important: Make sure that ports 80/HTTP and 21/FTP are not blocked by the firewall.

If a port is in use and a conflict is detected a dialog is displayed to allow the user to change the default ports.

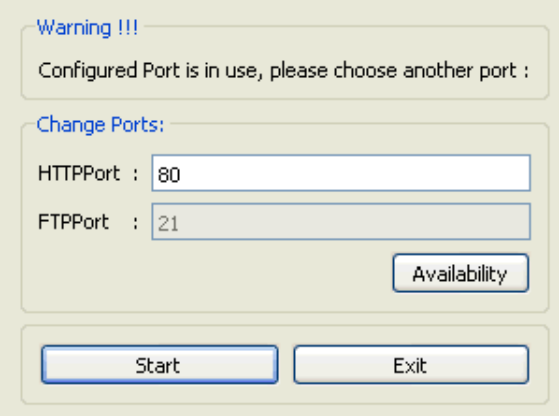
See "Protecting access to HMI devices" on page 415 for details.

In some conditions BTM-PCRUNTIME cannot detect all services running in ports like 80/HTTP and 21/FTP, this forces BTM-PCRUNTIME to be closed automatically. This happens, for example, when IIS or MS SQL Server or other windows services are running on these ports. In these cases, disable window services

If the project download to BTM-PCRUNTIME fails, try one of the following procedures.

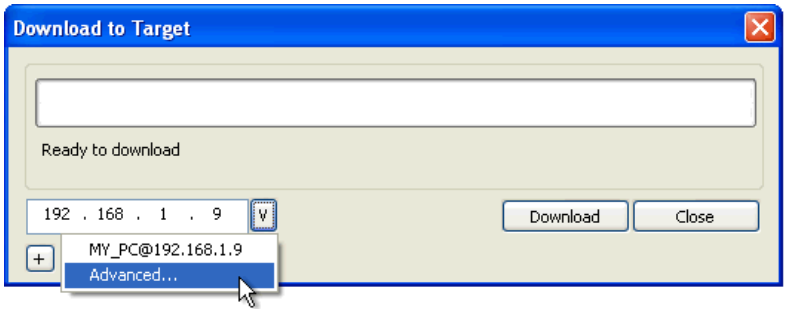
Issues with port numbers

BTM-PCRUNTIME uses ports 80 and 21 by default. If at least one is occupied a warning message is displayed:

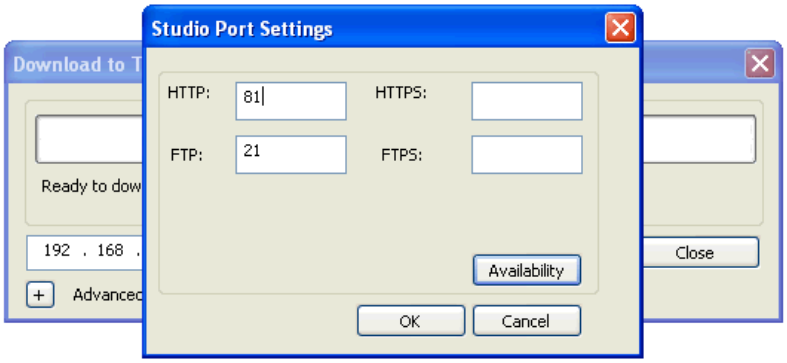


Make sure that when you change this port you also change the port used for download to HMI device in Wizard.

1. From the **Download to Target** dialog select **Advanced**.



2. Modify the port number to match that set on BTM-PCRUNTIME.



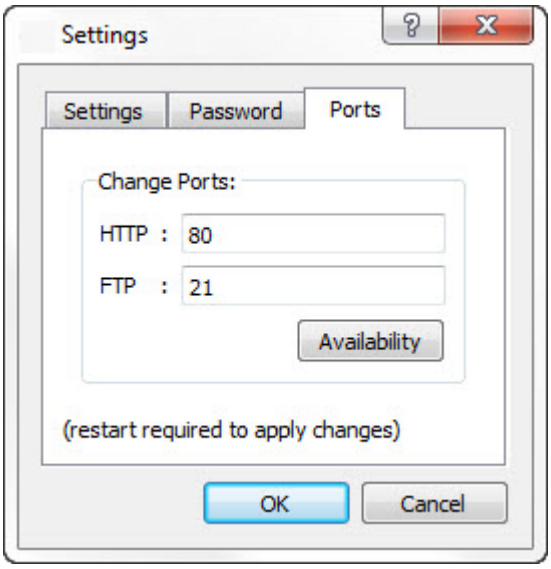
3. Click **OK** to confirm: you can now download you project to the BTM-PCRUNTIME.

Restoring port information

If information about changes made on BTM-PCRUNTIME listening ports has been lost, the following error message is returned:

Impossible to establish communication with Runtime. Please check connection settings and verify the Runtime is properly running on HMI device.

The port used by BTM-PCRUNTIME can be changed from the **Settings** dialog. Restart the application to apply the port change.



Bypassing firewall or antivirus blocks

If Wizard is running on the same machine as the BTM-PCRUNTIME, your firewall or antivirus may block the connection from Wizard to BTM-PCRUNTIME.

1. From the **Download to Target** dialog manually type-in the localhost IP address 127.0.0.1.
2. Click **Download**.

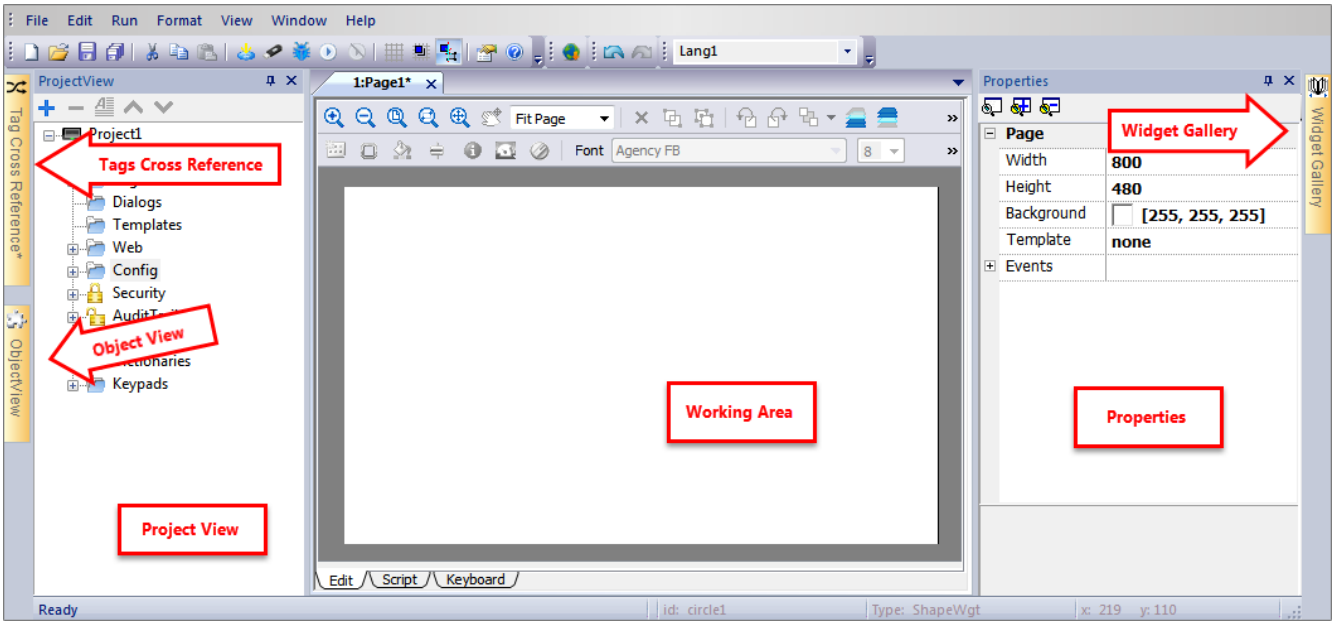
3 My first project

This section describes how to create a simple Wizard project.

The workspace

Workspace areas

Wizard workspace is divided into the following main areas:



Area	Description
Project View	Project elements in hierarchical project tree.
Object View	Tree view of widgets organized by page.
Working Area	Space where pages are edited. Tabs at the top of the area show all open pages.
Properties	Properties of selected object.
Widget Gallery	Library of graphic objects and symbols.
Tag cross reference	List of locations where a given tag is referenced.

Note: The workspace layout can be changed at any time, changes are saved and maintained through working sessions.

Resetting the workspace layout

To restore the default layout, use the **File > Reset and Restart** function.

Creating a project

Path: ***File> New Project***

1. In the **Project Wizard** dialog enter a name for the project and the storage location.
2. Click **Next**: the HMI device selection dialog is displayed.
3. Choose one device from the list of the available models.
4. Choose device orientation.
5. Click **Finish** to complete the Wizard.

Portrait rotation exceptions

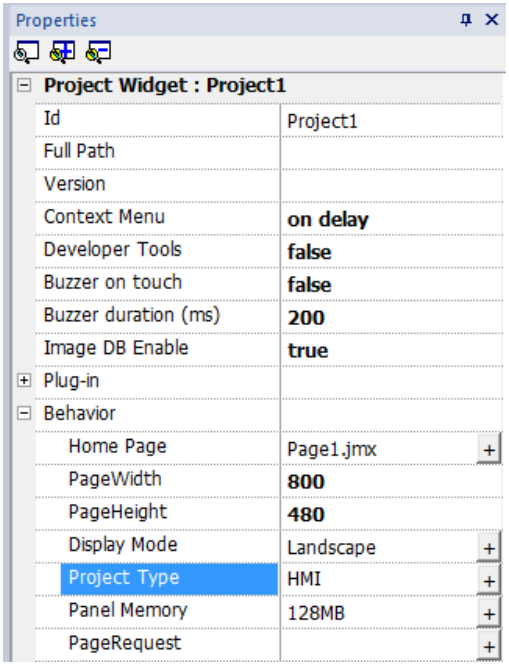
The following elements are not rotated in portrait mode.

Element	Description
Operating system dialogs	System settings and system dialog
ContextMenu and related dialogs	Project Manager, About, Settings, Logging, Backup
Video	IPCamera, MediaPlayer
JavaScript	Alert and Print function
Dialog pages	“Title” of dialog pages
Scheduler	Dialogs for data entry
Macro	ShowMessage, LunchApplication, LunchBrowser
External applications	, VNC

HMI devices based on Linux platform can be rotated from the BSP (see "*Displays*" tab from the "System Settings" on page 373"System Settings on Linux Devices" on page 381page) without these limitations.

Changing the device model

Once you have developed your project you can still change the device model, from the Project Properties pane. This will not resize the widgets, but will relocate them on the screen. A warning will be displayed if some objects cannot be relocated.



Copying, moving, renaming a project

Wizard projects folder contain all the files of the project: to move, copy or backup a project, move or copy the project folder to the desired location.

To rename a project use the **File > Save Project As** function: this operation might take a few minutes.

WARNING: Do not rename the project folders manually.

Communication protocols

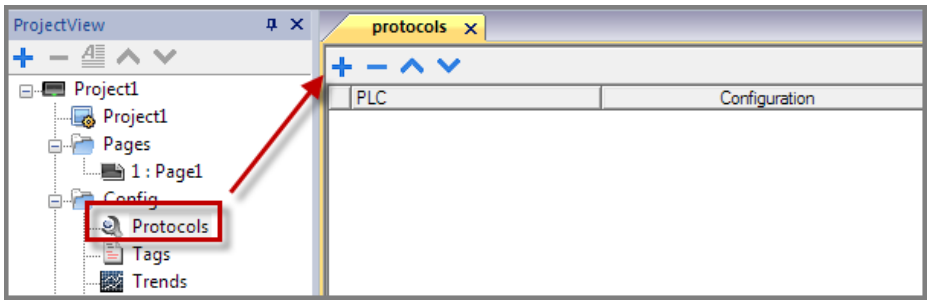
Path: ***ProjectView*** *>* ***Config*** *>* ***Protocols***

Device communication drivers are configured in the **Protocol Editor**.You can add up to the maximum number of protocols as specified in Table of functions and limits. Variable and System Variables are not counted as protocols.

Note: you can run different Ethernet protocols over the same physical Ethernet port, but you cannot run different serial protocols using the same serial port. Some serial protocols support access to multiple controllers, but this option is set within the protocol itself which is still counted as one protocol.

Adding a protocol

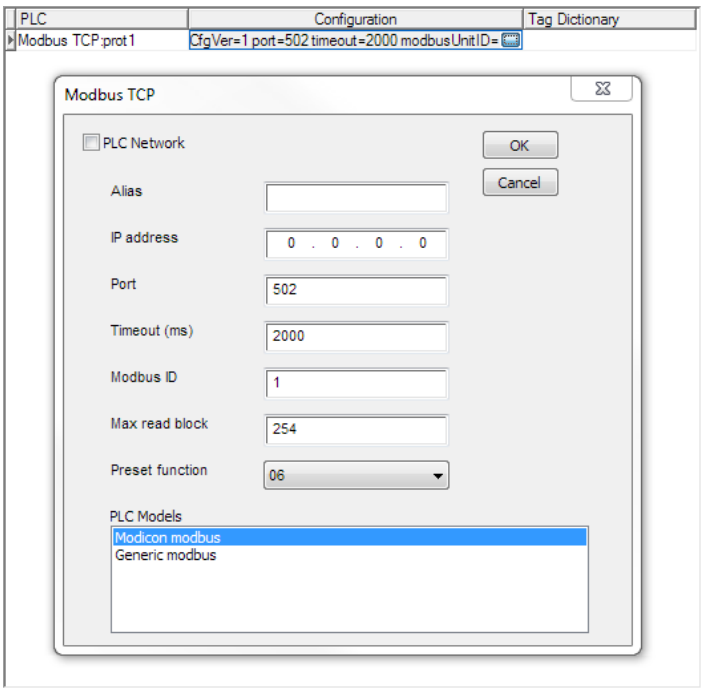
1. Click **+**.



2. Select the protocol from the **PLC** list and enter the required values.

Changing protocol settings

To change configuration parameters, click the browse button in the **Configuration** column.



Protocol parameters

Click **Show Advanced Properties** icon to see all parameters.

Parameter	Description
Dictionaries	Tags imported for the protocol. See "Importing tags" on page 25 for details.
Enable Offline AlgorithmOffline Retry Timeout	See "Automatic offline node detection" on page 207 for details.
Version	Protocol version available in Wizard for selected HMI device.

Designing a page

Path: ***ProjectView*** *>* ***Pages***

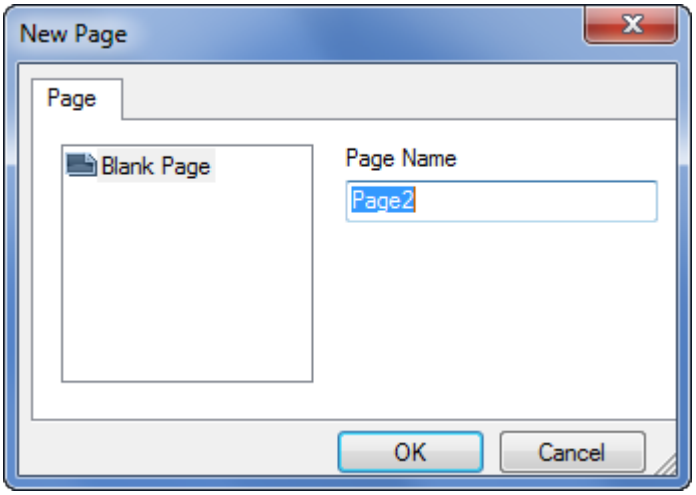
When a project is created, the first page is automatically added and shown in the **Page Editor**.

Adding objects to a page

Drag and drop objects from **Widget Gallery** to the page.

Adding a page

1. Right click the **Pages** node from the project tree and select **Insert new page**.
2. Type a name for the new page.



Importing a page

When importing a page Wizard will import the page layout and the page widgets without importing the actions and data links attached to widgets. You can choose between two different behavior:

- importing only the pages and the widgets: in this case all actions and data link have to be defined
- importing pages with references to actions and data links: used tags must be present in the project for these elements to work properly

Note: Page import can only be performed between projects made using the same software version. Save the older project as the newer version, then try again.

1. Right click the **Pages** node from the project tree and select **Import page**.
2. Choose the page to be imported from the desired project then click **OK**: a warning message is displayed.
3. Click **Yes** to remove all the links to data and actions. Click **No** to maintain the reference to data links and actions. Tags need to be available in the new project.

Group of pages

You can group similar pages for easier maintenance. Grouping pages does not affect how pages appears at run time. To create a group of pages:

1. In **ProjectView** right click **Pages** node and select **Create Group**: a new folder is added
2. To move a page to a group, right click a page and select **Groups** > *groupName.*

The Widget Gallery

Path: ***View> Toolbars and Docking Windows> Widget Gallery***

HMI objects required to build an application are available in the **Widget Gallery**. The gallery is divided into several categories, each containing a collection of widgets.

Adding a widget to a page

1. Select the widget from the **Widget Gallery**.
2. Drag and drop it on the page.

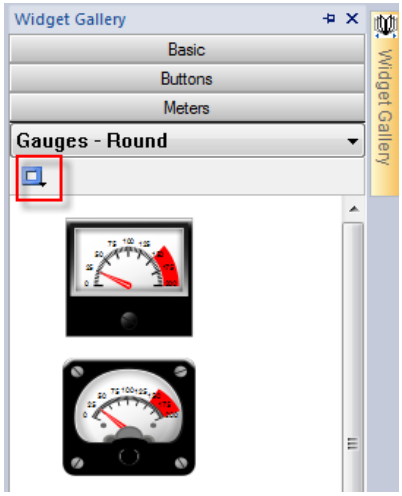
Changing the appearance of a widget

All widgets have properties (**Properties** pane) that can be changed, Some widgets are presented in various styles. You can click the buttons in each category to see available styles.

Example

To set the widget style for round gauges:

1. Click the style button to display the available styles for the widget.

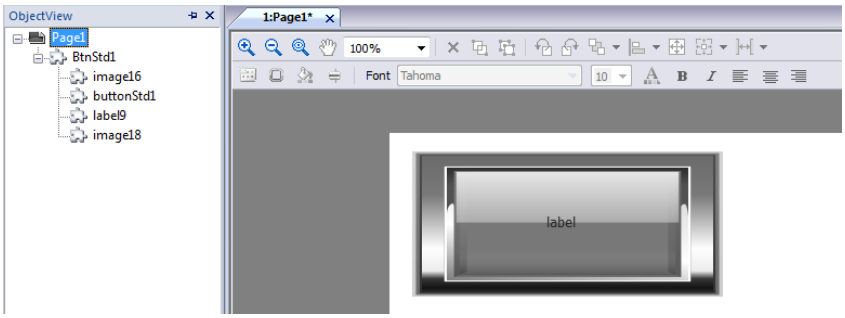


2. Select one of the available styles from the toolbar: depending on the selected widget, different options are available.

Complex widgets

Some widgets are composed of many sub widgets. For example, a button is a complex widget composed by a button widget and a label. The structure of widgets can be seen in the **ObjectView** when the widget is selected.

You can select a sub-widget, such as the label in a button, from the **ObjectView** and modify it without ungrouping the whole widget.



Adding tags

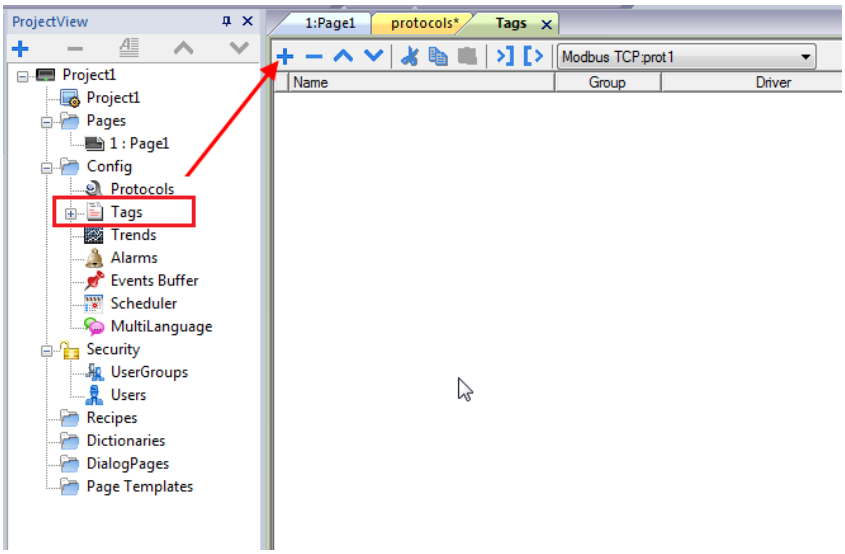
Wizard uses tag names to access all device data. All fields and reference locations in the device need to be assigned a tag name to be used in the HMI project.

Tag Editor can be used to create and manage tags. After the tags have been defined, they can be used in the project by attaching them to widgets' properties.

See ""Attach to" parameters" on page 32 for details.

Tag editor

*Path: **ProjectView** > **Tags***



Adding a tag

1. Click **+** and enter the required data.
2. Select the Address from the communication protocol address dialog: new tags are named Tag1, Tag2, ...
3. Click on the tag name to rename it.

Tag properties

See specific protocol documentation for details.

Property	Description
Name	Unique tag name at project level. Primary key to identify information in the runtime tag database.  **WARNING: Duplicate tag names are not allowed.**
Groups	Group names associated to a tag
Driver	Communication protocol
Address	Controller memory address. To edit click on the right side of the column to get the dialog box where you can enter the address information.
Encoding	Encoding type for string data type (UTF-8, Latin1, UTF-2 and UTF-16)
Comment	Tag description
Simulator	Tag behavior during simulation. Several profiles are available.
Scaling	Conversion applied to tag before database storage. **By formula** = defined as a linear transformation. **By range** = defined as a range conversion.

The below properties will be visible only after select the “Show Advance Columns” mode from the tag editor tollbar..

Property	Description
PLC Tag Name	Original PLC tag name, used to match tags used by HMI application (Tag Name) and tags exported from PLC R/W only in advanced view to allow for adjustments in case tag import errors.
Rate (ms)	Tag refresh time. Default: 500ms.  **WARNING: Tags refresh rate is the maximum refresh rate. Actual refresh rate depends on: communication type (serial, fieldbus, Ethernet), protocol, amount of data exchanged.**
R/W	R/W tag attribute (R/W, R or W).  Note: The content of Write Only tags is always written and never read. When communication is not active, the content of these tags may not be available in widgets.
Active	Update mode. **false** = tags are read from controller only when required by the HMI device. **true** = tags are continuously read even if not required by the displayed page.  **Important: Leave this value set to false for higher communication performance.**

Managing tag names

Tag names must be unique at project level. If the same tags, from the same symbol file have to be used for two different controllers, use the “Alias” feature to add a prefix to the imported tags and make them unique at project level.

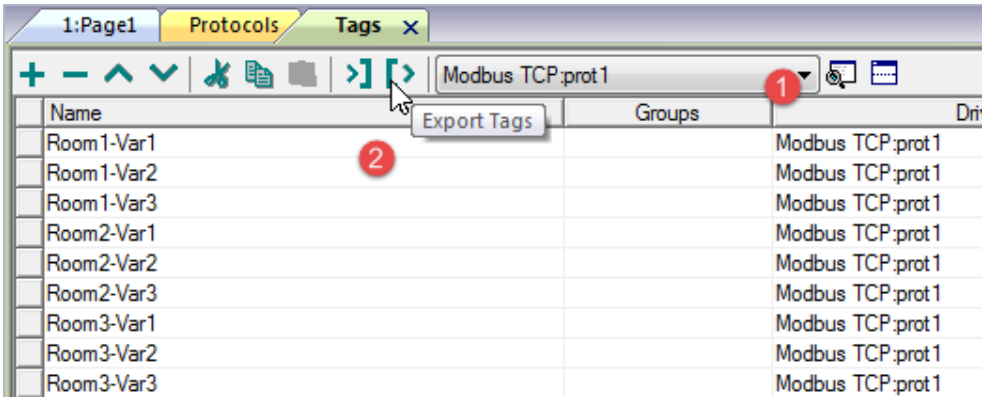
Note: Not all protocols support the “Alias” feature.

Managing tag groups

Tags used in each page are identified as part of a group, so that requests made by the communication protocol to the connected controller(s) can be processed faster: only the tags included in the displayed page are polled from the controller.

Exporting tags

Path: ***ProjectView*** *>* ***Tags***



1. Select the protocol for the tags you want to export.
2. Click the **Export Tags** button: all the tags configurations for the selected protocols are exported into an .xml file.

You can edit the resulting .xml file using third part tools (for example, Microsoft Excel) and then re-import the modified file (see "Importing tags" below for details).

Importing tags

Introduction

Some protocols allow you to import tags stored in a comma separated file (.csv or other formats).

Importing is a two step process:

1. Import of the tag definition into a dictionary
2. Import tags from the dictionary to the project

WARNING: Special characters in tag names such as “&” character cause communication errors. See "Limitations in Unicode support" on page 216

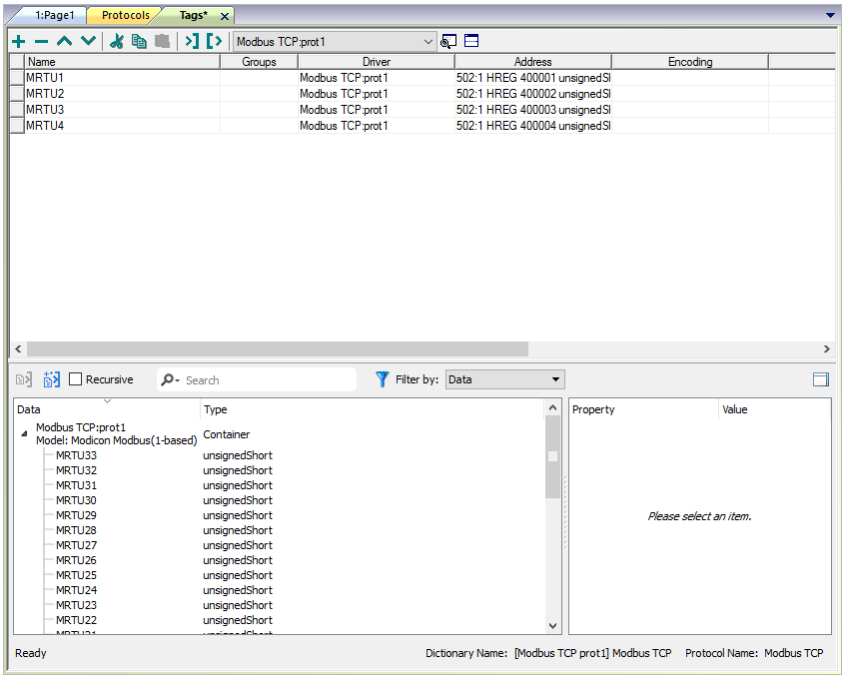
Note: When importing tags, character "." in tag names is replaced with "/" . The protocol will use the correct syntax when communicating to the PLC.

Dictionaries

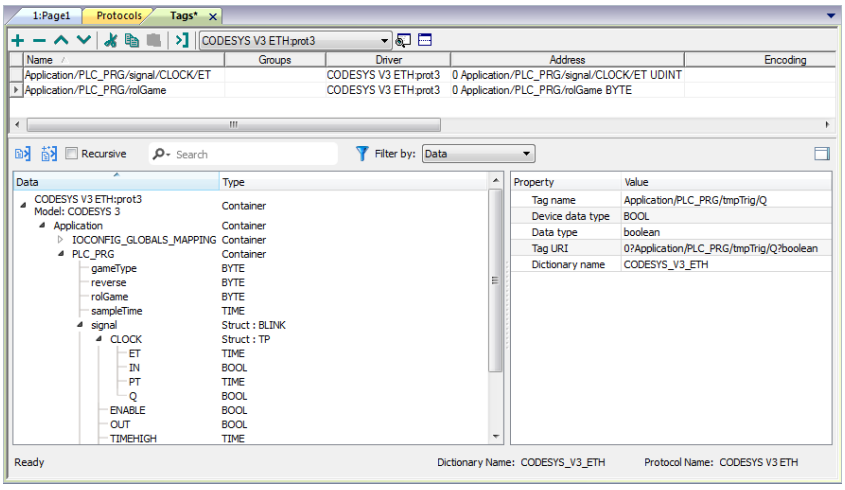
Path: ***ProjectView*** > ***Dictionaries***

A dictionary is a list of tags imported in the Tag Editor for a specific protocol. Depending on the protocol type, tags are shown in linear view or in hierarchical view.

Linear view



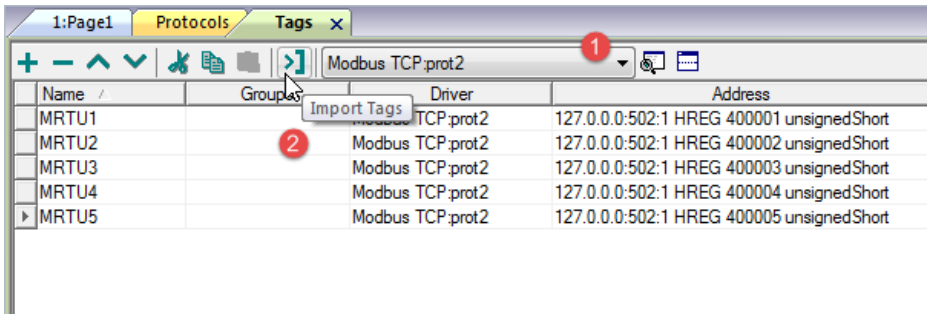
Hierarchical view



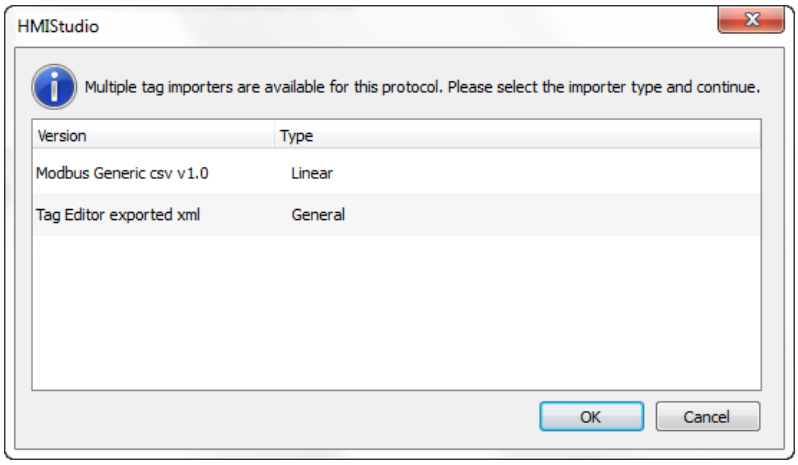
Importing tags

To import tags from an external file:

1. In **ProjectView, Tags** select the protocol from the filter list.



2. Click the **Import Tags** button: the select file dialog appears. A dialog to choose the importer type appears.



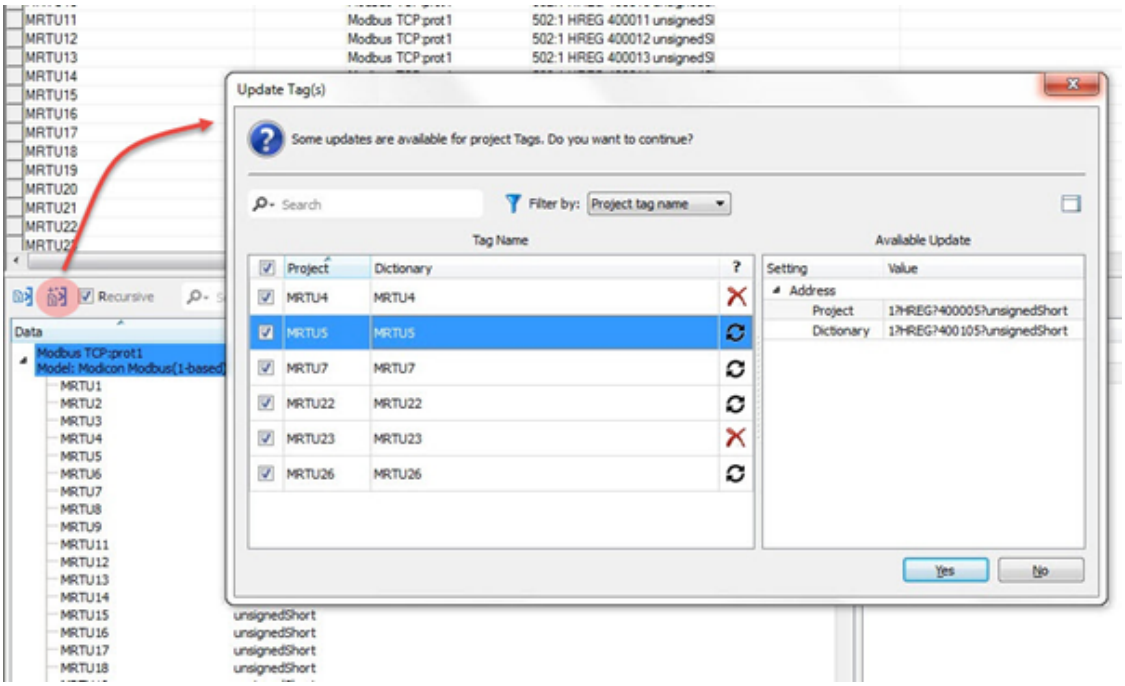
3. Select the file: a list of tags is shown in a linear or hierarchical view.
4. To import tags, select one or more tags or a node (hierarchical view only) and click the **Import tag** button: tags are copied to the project and listed in the upper window section.

Parameter	Description
Recursive	All elements of the structure are imported into separate tags.

Note: When the project is configured to use a protocol network you must also select the protocol node where tags are to be imported. You can import the same tags on multiple protocols. When the tags file contains the node information, you can choose to use the information to filter the tags and import only those matching with the selected nodes.

Updating the imported tags

Using the Update Tag(s) command you can re-import tags. A dialog allows you to select the tags to be reimported:



These tags need to be updated. A list of differences between project and dictionary is displayed.

These tags are no longer available in the dictionary. If updated, these tags will be removed from the project.

Attaching widget to tags

To control a widget and animate it through live data it is possible to bind a specific property to different data sources. For example it is possible to bind the gauge **Value** property to a probe temperature tag, or the **Display** property to a recipe data

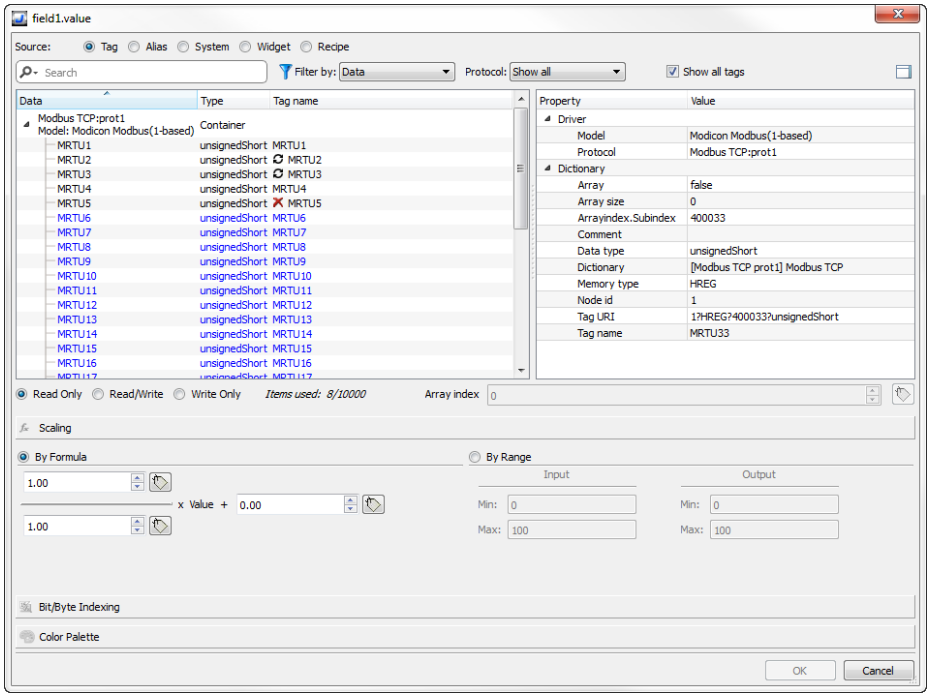
Data sources

Elements to which an object property can be attached:

Data source	Description
Tag	Tag defined in the Tag Editor
Alias	Indexed tag address
System	Predefined system tags (see "System Variables" on page 79)
Widget	Connect to a widget property (for example, value of a slider widget)
Recipe	Data from the Recipe Manager (see "Recipes" on page 181)

Attaching a property to a tag

1. Click **+** in the **Properties** pane.
2. In **Source** choose the data source, in the list choose a protocol and the tag. Use the **Search** box to filter tags.



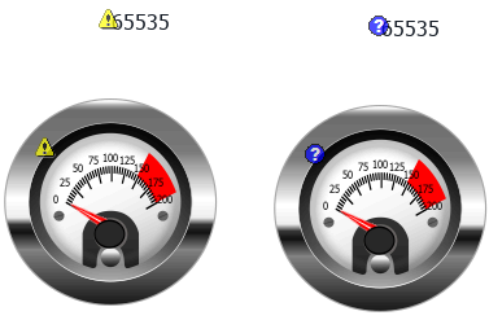
3. Set the access type (for example **Read Only**). The **Array Index** field appears when the selected tag is an array to identify the element of the array to use. The indirect index mode, through an additional tag, is supported.
4. Click **OK** to confirm.

The icons adjacent to the tag name highlight when a definition does not match the tag definition in the dictionary, or when missing. If the **Show all tags** is selected, all the dictionary tags are shown also if not imported within the application. A double-click will import the tags from the dictionary.

See ""Attach to" parameters" on page 32 for details.

Communication Error

Two icons may appear close to widgets that have an attached tag.



- : communication error
- : data not yet available (slow communication protocol)

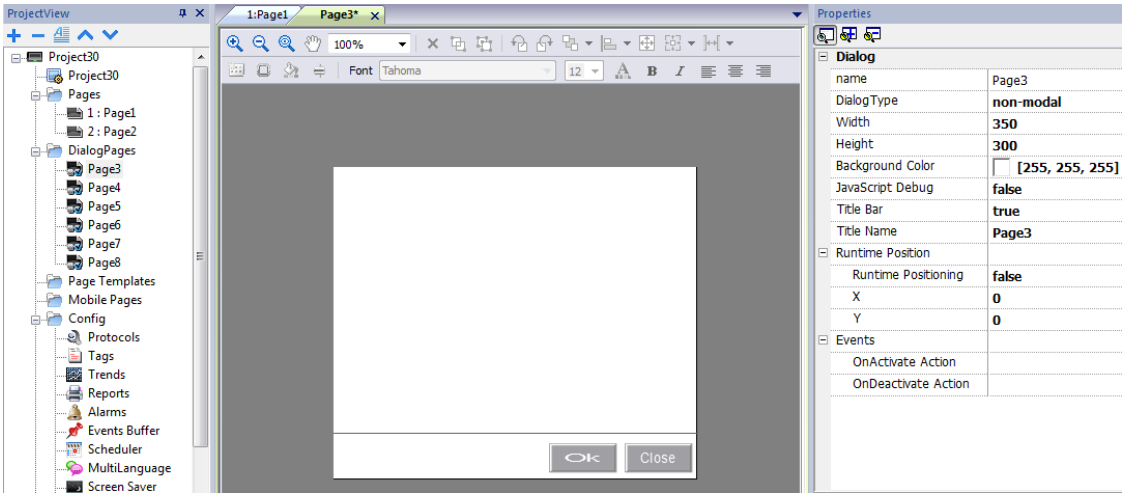
Dialog pages

Path: ***ProjectView*** *>* ***Web*** *>* ***Dialogs***

Dialog pages are opened at run time on top of the current page on project request. They are used to notify alarms, errors or to require user action.

Main dialog properties

Property	Description
Dialog Type	**modal** = user cannot return to main project window/page until dialog is closed. **non-modal** = user can continue to use main project window (or other non- modal dialogs) while a dialog is shown on top of it.
Title Bar	**true** = dialog title displayed **false** = no dialog title displayed
Title Name	Dialog title. Only if **Title Bar**=true.
Runtime Position	Dialog fixed position **false** = Dialog will be placed centered on the screen **true** = Dialog will be placed with upper-left corner at position X and Y



Maximum number of dialogs

Maximum number of open dialogs is defined in "Functional specifications and compatibility" on page 427.

When the maximum number of open dialogs is reached, the oldest dialog is closed to open the new one.

4 Programming concepts

Programming for Wizard is based on a few basic concepts and behaviors.

Data types

When creating a tag you have to specify its properties. Data type are specific to Wizard, memory type are specific to the selected protocol. Choose the value according to the internal representation you need for the selected controller address.

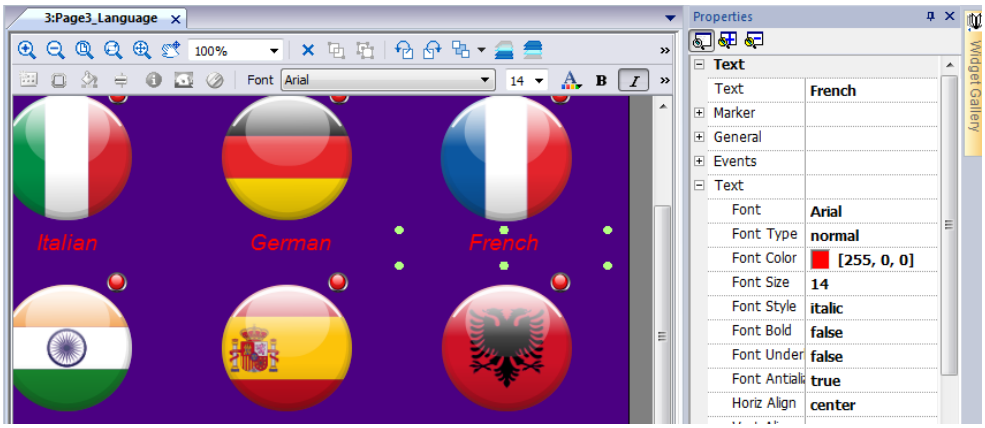
Note: arrays type use the same data type followed by "[ ]" (i.e.: boolean [ ])

Data Type	Description
boolean	One bit data (0..1)
byte	Signed 8 bit data (-128..127)
double	IEEE double-precision 64-bit floating point type (±2.2e-308 ... ±1.79e308)
float	IEEE single-precision 32-bit floating point type (±1.17e-38 ... ±3.40e38)
int	Signed 32 bit data (-2.1e9 ... 2.1e9)
short	Signed 16 bits data (-32768..32767)
string	Characters coded according to selected format
time	Time data
unsignedByte	Unsigned 8 bit data (0..255)
unsignedInt	Unsigned 32 bit data (0 ... 4.2e9)
unsignedShort	Unsigned 16 bit data (0..65535)
uint64	Unsigned 64 bit data (0...264 − 1)

"Attach to" parameters

Object properties

In Wizard the properties of an object placed on a page can be set at programming time or configured to be dynamic. To change a property at programming time use the page toolbar or the property pane. Select the object first to see its properties displayed.

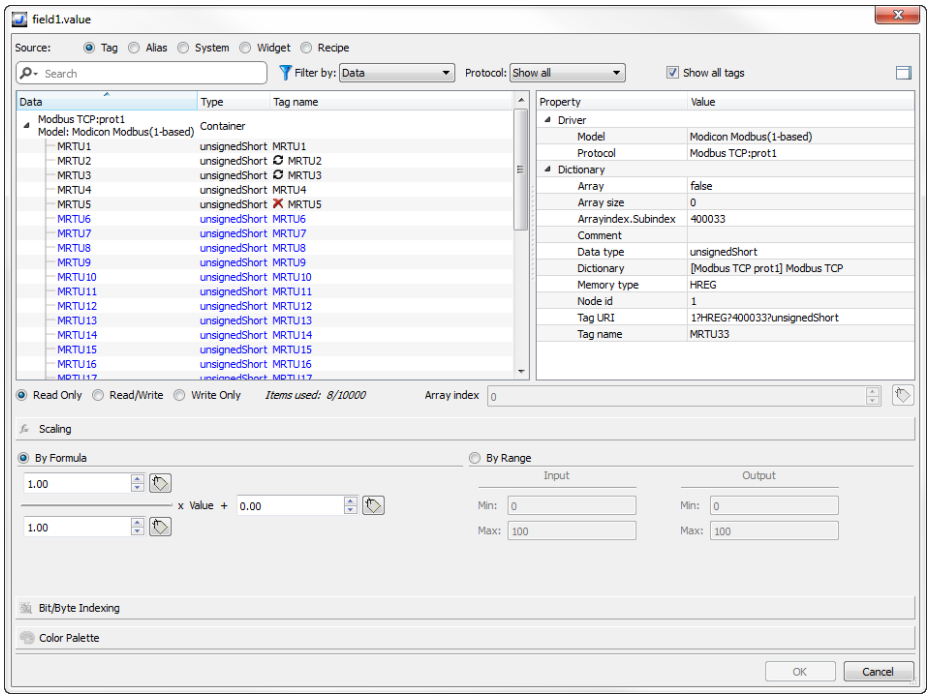


The page toolbar shows only the most common object properties, while the property pane show all the properties in a basic or advanced view.

To change a property value dynamically you can attach it to tags or variables.

Attaching a property to a tag

1. Click **+** in the **Properties** pane.
2. In **Source** choose the data source, in the list choose a protocol and the tag. Use the **Search** box to filter tags.



3. Set the access type (for example **Read Only**). The **Array Index** field appears when the selected tag is an array to identify the element of the array to use. The indirect index mode, through an additional tag, is supported.
4. Click **OK** to confirm.

The icons adjacent to the tag name highlight when a definition does not match the tag definition in the dictionary, or when missing. If the **Show all tags** is selected, all the dictionary tags are shown also if not imported within the application. A double-click will import the tags from the dictionary.

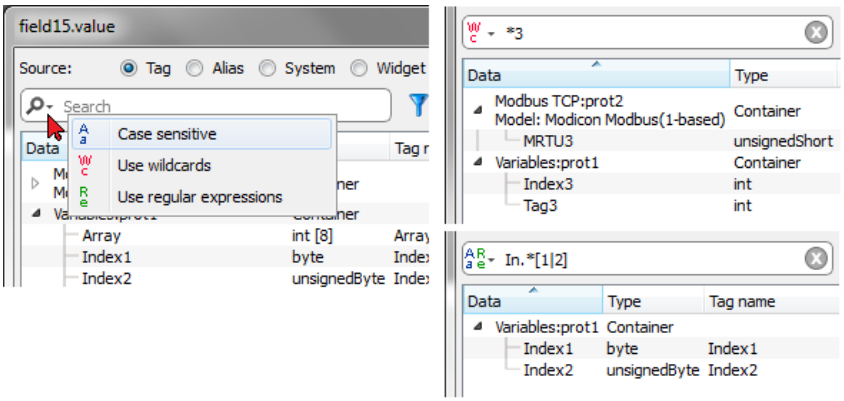
Data sources

Elements to which an object property can be attached:

Data source	Description
Tag	Tag defined in the Tag Editor
Alias	Indexed tag address
System	Predefined system tags (see "System Variables" on page 79)
Widget	Connect to a widget property (for example, value of a slider widget)
Recipe	Data from the Recipe Manager (see "Recipes" on page 181)

Advanced search

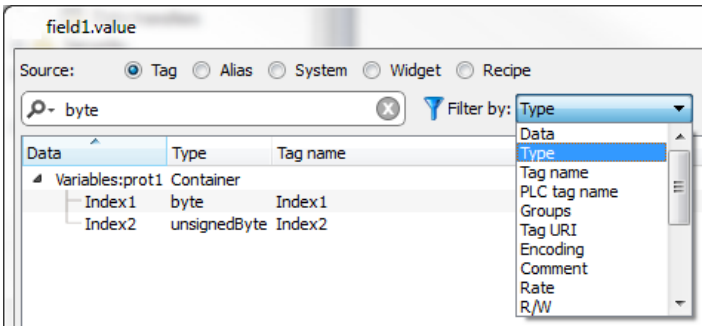
Various syntax options can be applied to search box:



Main options	Function
Wildcards	Search using simple wildcards matching . Character '?': matches any single character. Character ' *': matches zero or more of any characters." […]": sets of characters can be represented in square brackets.
Regular Expression	Describes character pattern. See http://www.regular-expressions.info/

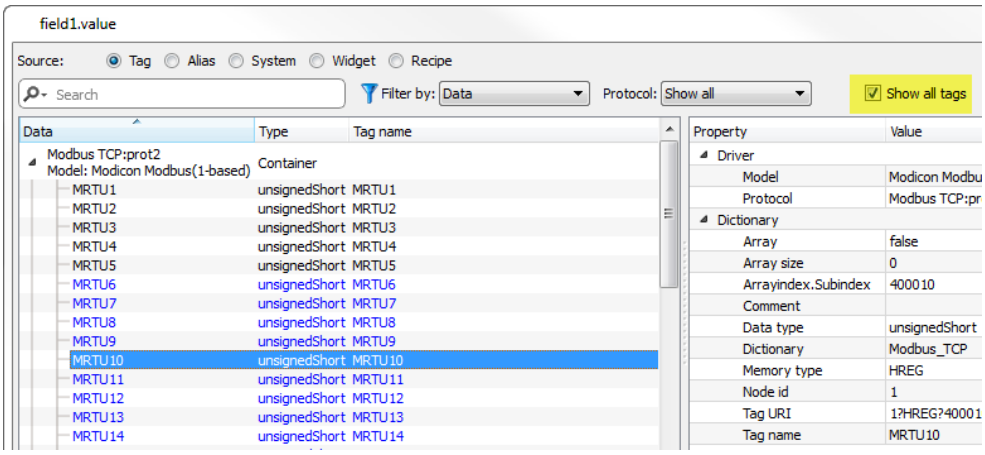
Filtering tags

Choose various tag filter criteria:

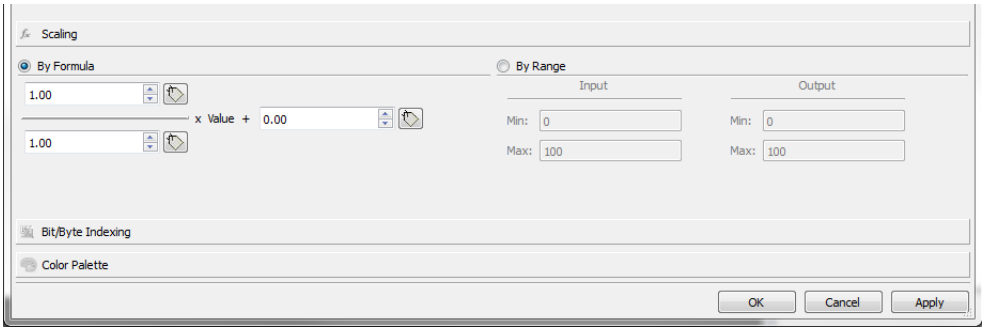


Showing dictionary tags

When **Show all tags** is checked, tags that belong to one dictionary but have not been imported yet, appear in blue color. You can select and double-click a tag to import it into the project.



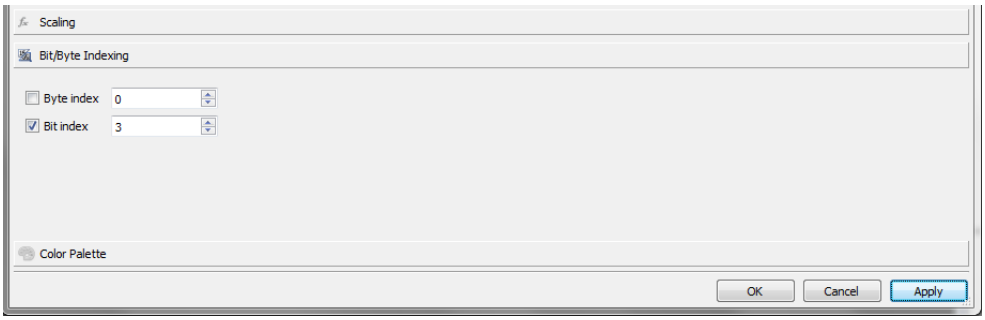
Converting tag value



Scaling tab converts the tag value. In **By Range** section set the input and output range: the system will automatically calculate the scaling factors.

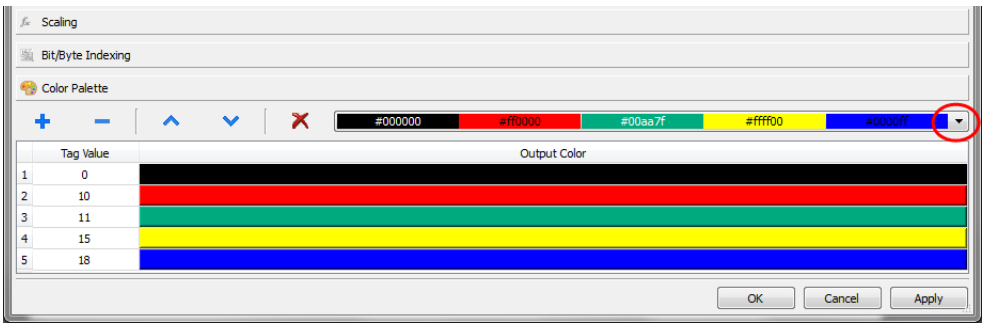
Extract tag bit/byte based on index

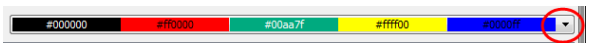
Allows extracting a single bit or byte content from a word depending on the specified bit or byte number



Mapping tag values to color

Allows you mapping numeric tag values to colors. You can use this option to change the color of a button.



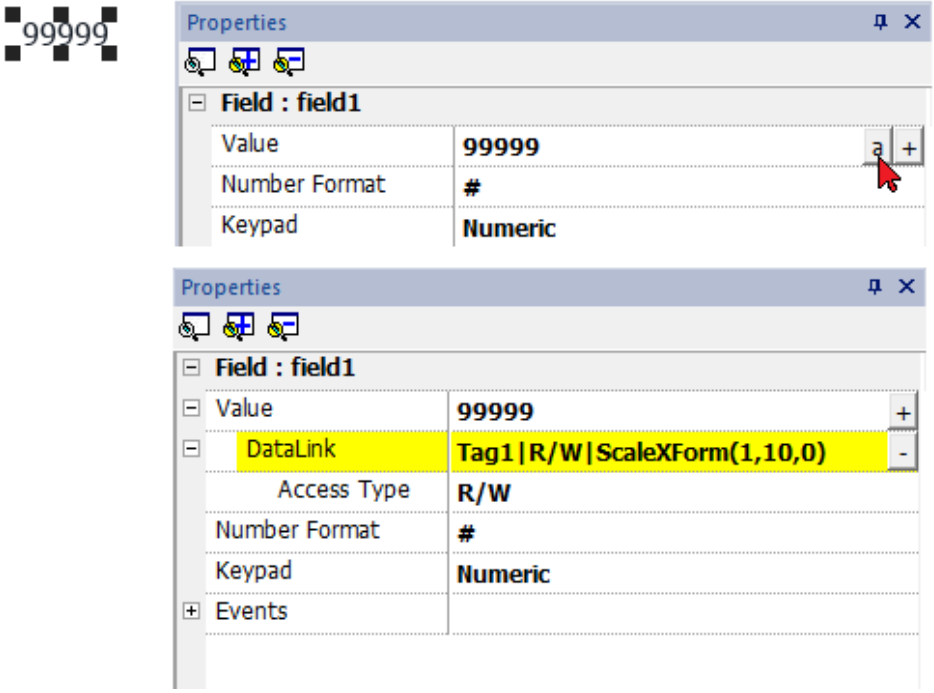
Section	Function
	From the toolbar add/remove or move up/down the colors lines. The tag value is editable and you can modify the sequence values.
	Last defined color combination is saved automatically and can be retrieved from the color toolbar.

Note that the mapping tag value to color will return a string data type (e.g. "#FF0000")

Datalink Serialization

Instead of use the above "Attach to…" dialog box, datalinks can be entered, or modified, manually.

Click **a** button in the **Properties** pane and enter the text that describe the datalink



The data link format is:

```
Tagname [index] | [Atribute] | [XForm] | [XForm] | ...
```

Example:

- arrayTag[2]
- Tag[0|index]
- Alarm triggered:_SysPropMgr
- Tag|R/W|ScaleXForm(1,10,0)
- Tag|R/W|ScaleXForm(1,10,0)|ByteIndexXForm(1)|ColorPaletteCustomXForm(0#00aa7f,1#ff0000)

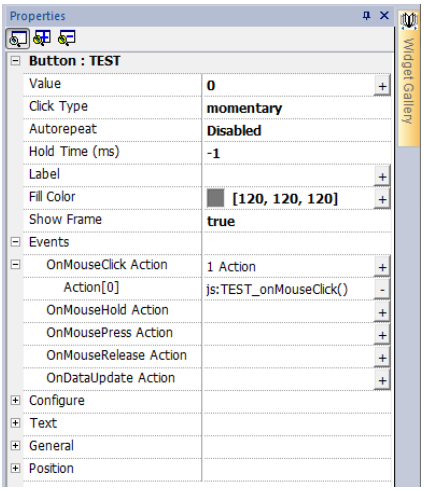
Events

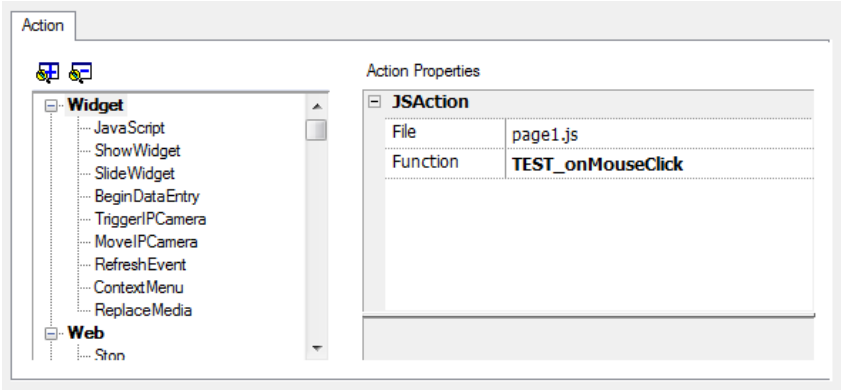
Events are used to trigger actions at project level and can be associated to:

- buttons / touch (click, press, release)
- external input devices like keyboards and mouse (click, press, hold, release, wheel)
- data changes (OnDataUpdate)
- switch of pages (OnActivate, OnDeactivate)
- alarms
- scheduler

You can attach one or more actions to an event, so that they will be executed whenever the event occurs.

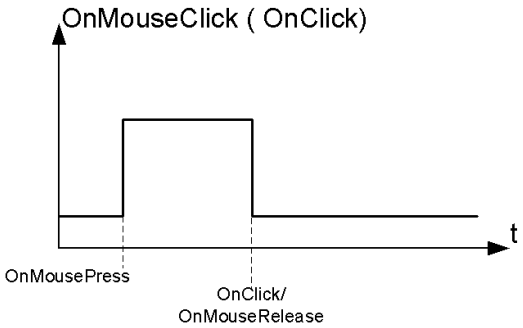
This example shows a JavaScript action activated by pressing a button.





OnClick / OnMouseClick

Triggers the event when the button/key is pressed and released quickly.

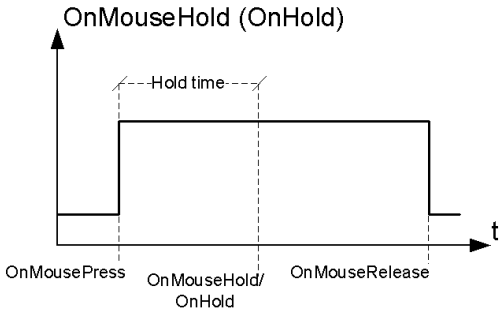


OnHold/OnMouseHold

Triggers the event when the button/key is pressed and held pressed for a certain time set as **Hold Time** in the widget properties. Actions programmed for this event will be executed only after the hold time has expired.

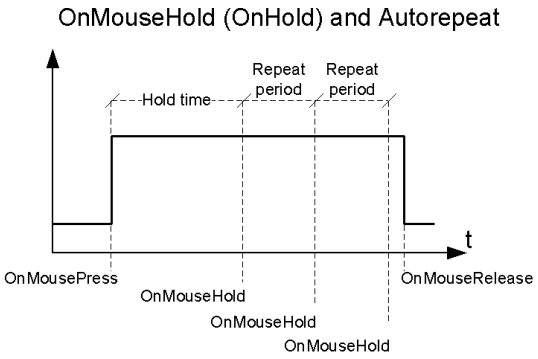
The default **Hold Time** is configured in Project properties but can be redefined for each button/key. See "Project properties" on page 53.

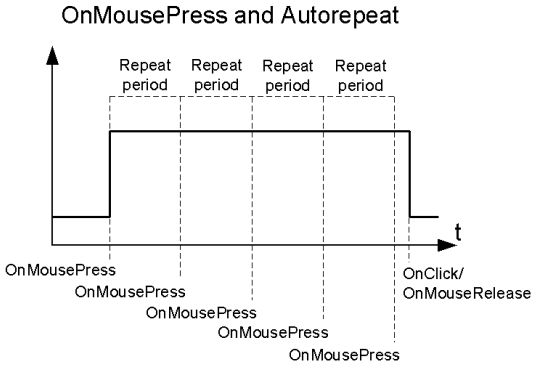
Note: If **Hold Time** is set to -1 for the widget, the project **Hold Time** value will be used.



Autorepeat

Enables auto repeat for a press or hold event of button or key. **Autorepeat Time** is specified in the Project properties but can also be redefined for each button or key





OnWheel

Triggers the event when a wheel (for example a USB mouse wheel) value changes. A wheel usually is used to increase/decrease values in a text box or attached to a tag.

OnActivate

Triggers the event when a page is loaded. The event starts before widgets in the page are initialized.

OnDataUpdate

Triggers the event when the tag value changes. The update moment depend on the time needed by the protocol to finish the update process. For example the **OnDataUpdate** event can be triggered or not, depending on whether data becomes available from protocol respectively after or before widgets being initialized for the first time. In particular, page change notifications are more likely to happen with slow protocols and remote clients.

Note: The value read during **OnActivate** can be the same obtained from a subsequent **OnDataUpdate** event, since **OnDataUpdate** notifications are sent asynchronously.

Widgets positioning

You can position widgets in the page using two methods:

- Snap to Grid
- Snap to Object

To display the grid, on the **View** menu, click **Show Grid**.

Snap to Grid

Path: ***View*** *>* ***Snap to Grid***

When you move or re-size an object, its top left corner will align with the nearest intersection of lines in the grid, even if the grid is not visible.

Setting grid properties

Path: ***View*** *>* ***Properties***

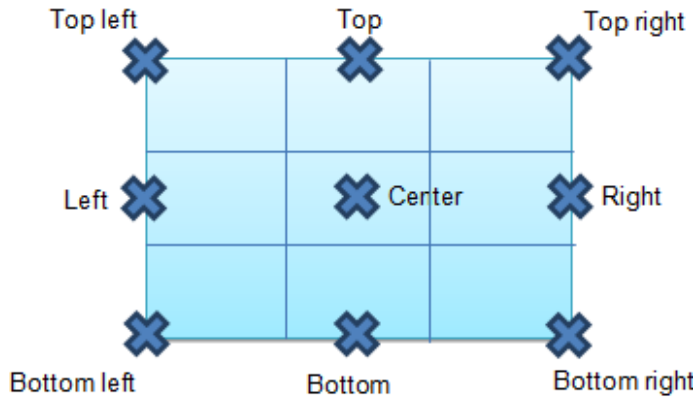
Parameter	Description
Spacing X	Space in pixel between two lines/dots on the X axis
Spacing Y	Space in pixel between two lines/dots on the Y axis
Type	Grid type (dot or line)
Color	Grid color

Snap to Object

Path: ***View*** *>* ***Snap to Object***

When you move an object, it will align with other objects on the page.

When you select an object, one of the following hot points is selected as the source of the snap point, depending on the area you pressed: top, top left, top right, bottom, bottom left, bottom right, left, right, center:

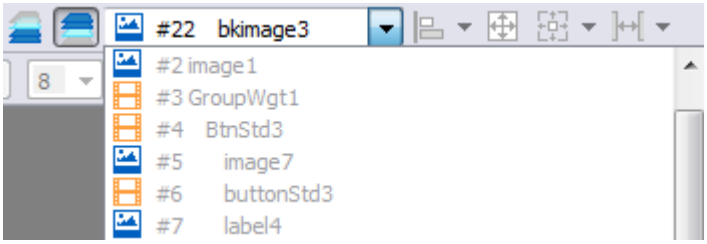


An algorithm finds a matching hot point among the near widgets hot points matching either the x or the y coordinates of the source snap point. For line widgets, the source snap points are the terminal points of the line.

Managing overlapping widgets

When one or more widgets on the page overlap, you can manage their order so that one is displayed on top of the other.

The order of the widget on the page is shown in the combo box. A widget with greater z-order number is in front of an element with a lower z-order number. A picture icon identifies static objects, a movie frame icon identifies dynamic objects.



Important: Correct ordering of widgets is essential for run time performance since overlapping dynamic widgets can invalidate static optimization and reduce performance of HMI applications.

Hiding/showing widget on z-order

To hide widgets above a selected widget:

- On the toolbar click and select a widget: all widgets above this one are hidden

To hide widgets below a selected widget:

- On the toolbar click and select a widget: all widgets below this one are hidden

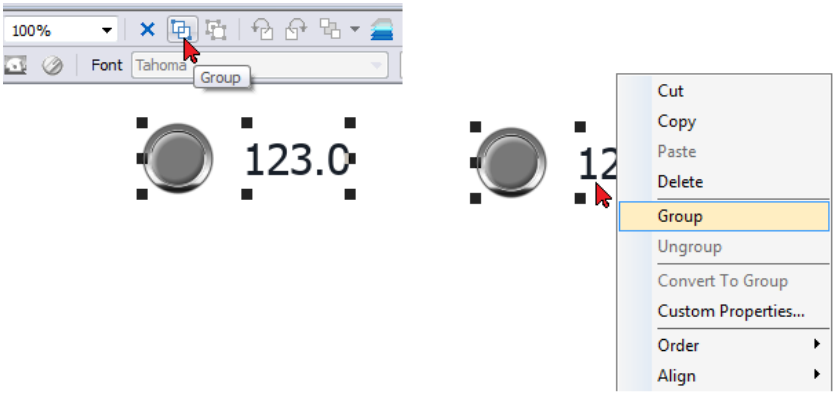
The toolbar allows to:

- hide widgets stacked above and/or below selected widgets
- work on different widgets using the combo box which lists all the widgets in their z-order.

Grouping widgets

To group widgets:

1. Select all the widgets to group.
2. Right-click and then click **Group**.

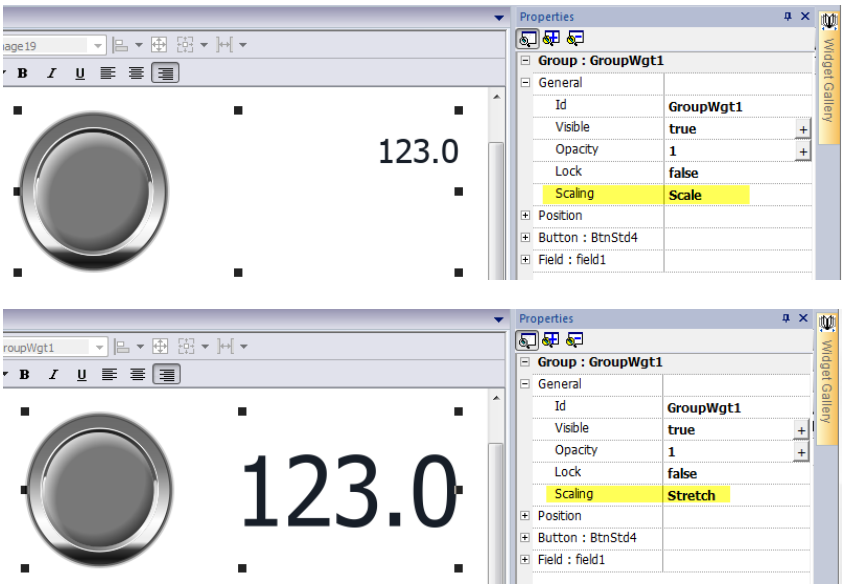


Tip: Double click to enter the group editing mode. In group mode only the group widgets are editable and selectable. All other widgets are partially hidden

Resizing grouped widgets

You can define how object reacts when re-sized. Use the **Scaling** property in **General** section:

- **Scale**: object and text are not re-sized proportionally
- **Stretch**: object and text are re-sized proportionally



Grid Layout Group

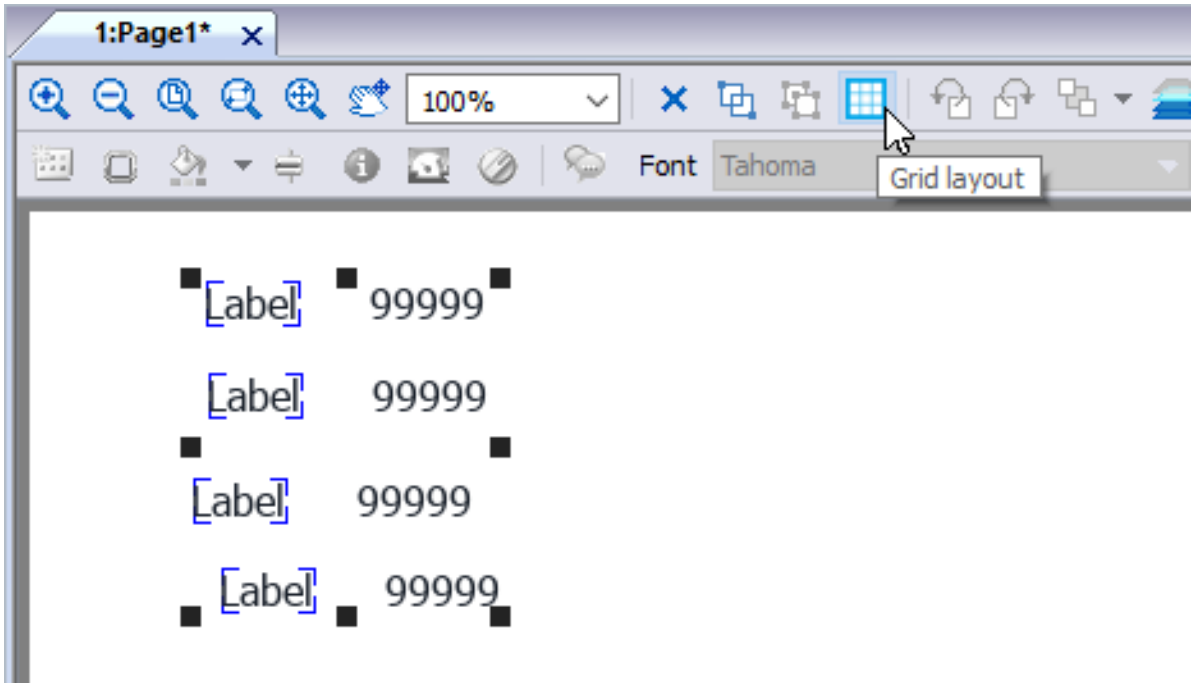
The grid layout add the possibility to configure the spatial relationships among the widgets of the group.

To create a grid layout:

- Enable the "*Grid Layout*" parameter of the group of widgets.

or

- Select the widgets that will be inside the table and click the “Grid Layout” button on page toolbar. The selected widgets will be aligned and collected inside a group with the grid layout property enabled.

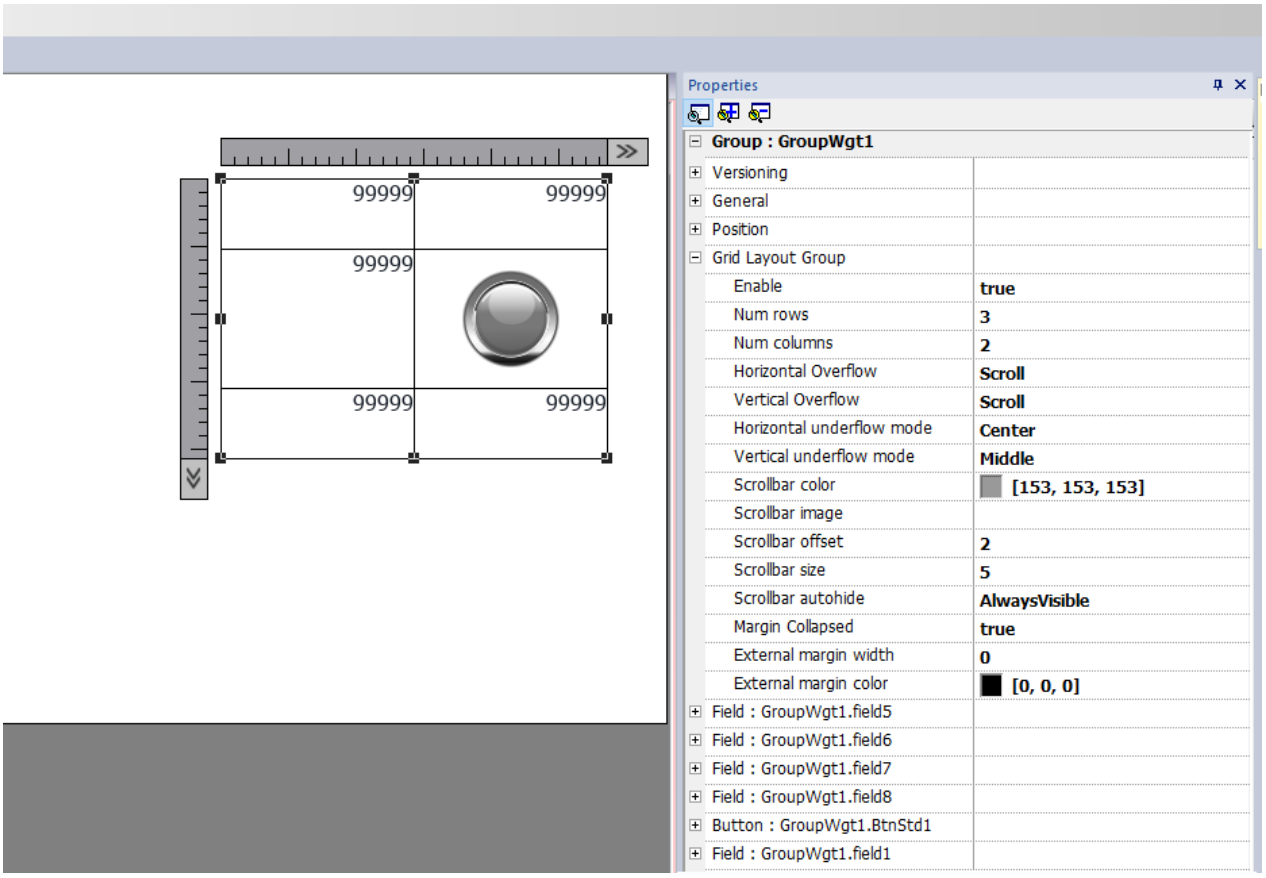


There are several elements associated with the grid layout that can be configured:

- Grid properties
- Rows, Columns Properties
- Cells Properties

Grid Properties

Main grid layout properties are available inside the advance mode of the properties panel when the widget group is selected.

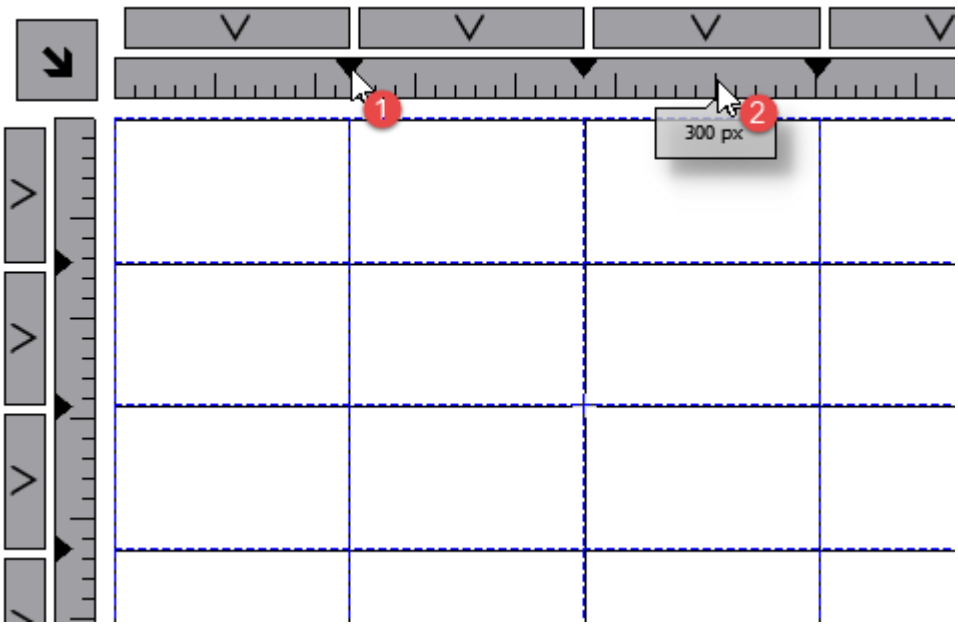


Parameter	Description
Enable	Enable the grid layout. A grid will be generate around the widgets of the group
Num rows **Num columns**	Number of rows and columns of the grids. Rows and columns can be removed only if their cells are empty .
Horizontal overflow **Vertical overflow**	This parameter define the behavior of the grid when it is too small to contain all rows and columns. • Hidden Rows and columns that do not fit into the grid are not displayed • Visible The grid can not be made smaller than the minimum size required to contain all defined rows and columns • Scroll When the grid is too small to hold all the defined rows and columns, the scroll bars can be used to shift the content of the grid.

Parameter	Description
Horizontal underflow **Vertical underflow**	This parameter defines the behavior of the grid when it is larger than the size defined for the rows and columns • Blocked The grid can not be made larger than the maximum size of rows and columns • Left, Center, Right - Top, Middle, Bottom Defines the position of the widgets when cells are bigger than the maximum defined sizes
Scrollbar color **Scrollbar image** **Scrollbar offset** **Scrollbat size** **Scrollbar autohide**	Parameters to define look and position of the scroll bars
Margin collapsed	Collapse all left-right and top-botton margin using the parameters of the stroke with greater width.
External margin width **External margin color**	External margin parameters

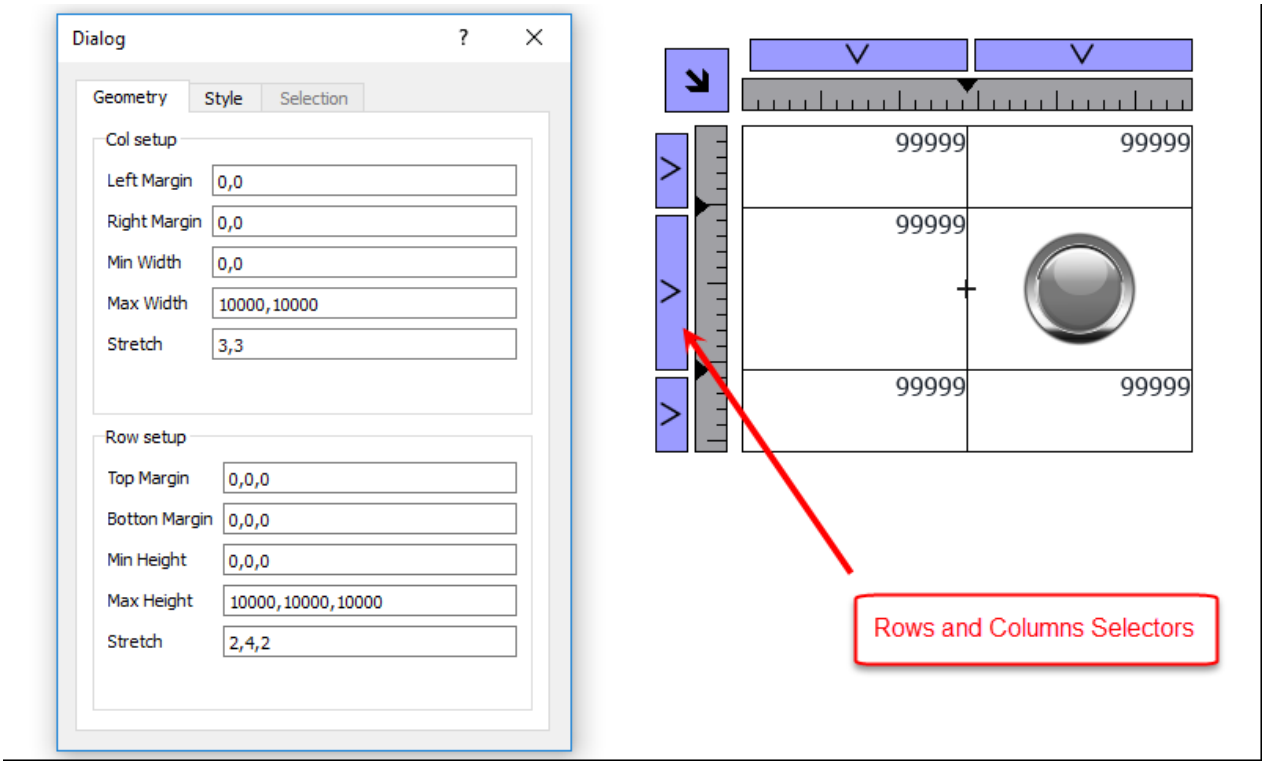
To merge or split rows or columns, double click over the grid, to enter in edit mode, and move the cursor over the ribbons:

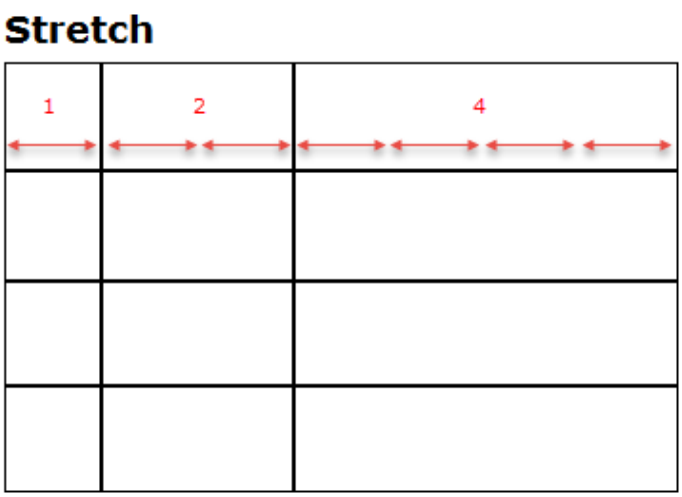
- Double click the black triangle to merge the two adjacent rows or columns (1)
- Double click on ribbon to split the selected row or column (2)

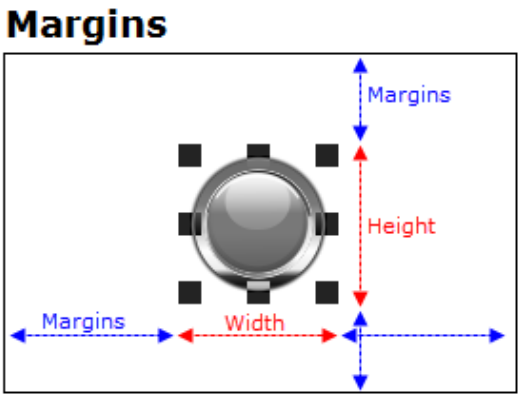


Rows, Columns Properties

Row and columns properties are available inside a pop up dialog after clicking on the row and column selectors, that are visible after double clicking the group of widgets.







Geometry parameters

Parameter	Description
Left margin **Right margin**	Distance of the widget from the border of the cell
Min width **Max width**	Min/Max width that widget can assume when the cell is stretched
Stretch	Defines the relationship between the widths of the columns that will be maintained if the grid is stretched
Top margin	Distance of the widget from the border of the cell

Parameter	Description
Bottom margin	
Min height **Max heighty**	Min/Max height that widget can assume when the cell is stretched
Stretch	Defines the relationship between the heights of the rows that will be maintained if the grid is stretched

Style parameters

Parameter	Description
Left stroke width **Right stroke width** **Top stroke width** **Bottom stroke width**	Strokes width
Left stroke color **Right stroke color** **Top stroke color** **Bottom stroke color**	Strokes color
Background color	Row background color

The list of values that are separated by a comma, are related to rows and columns. Example, the first value is for row 0, the second value for row 1, and so on.

Color format could be #rrggbb or #rrggbbaa, where "*aa*" is the alpha value which defines the opacity of the color.

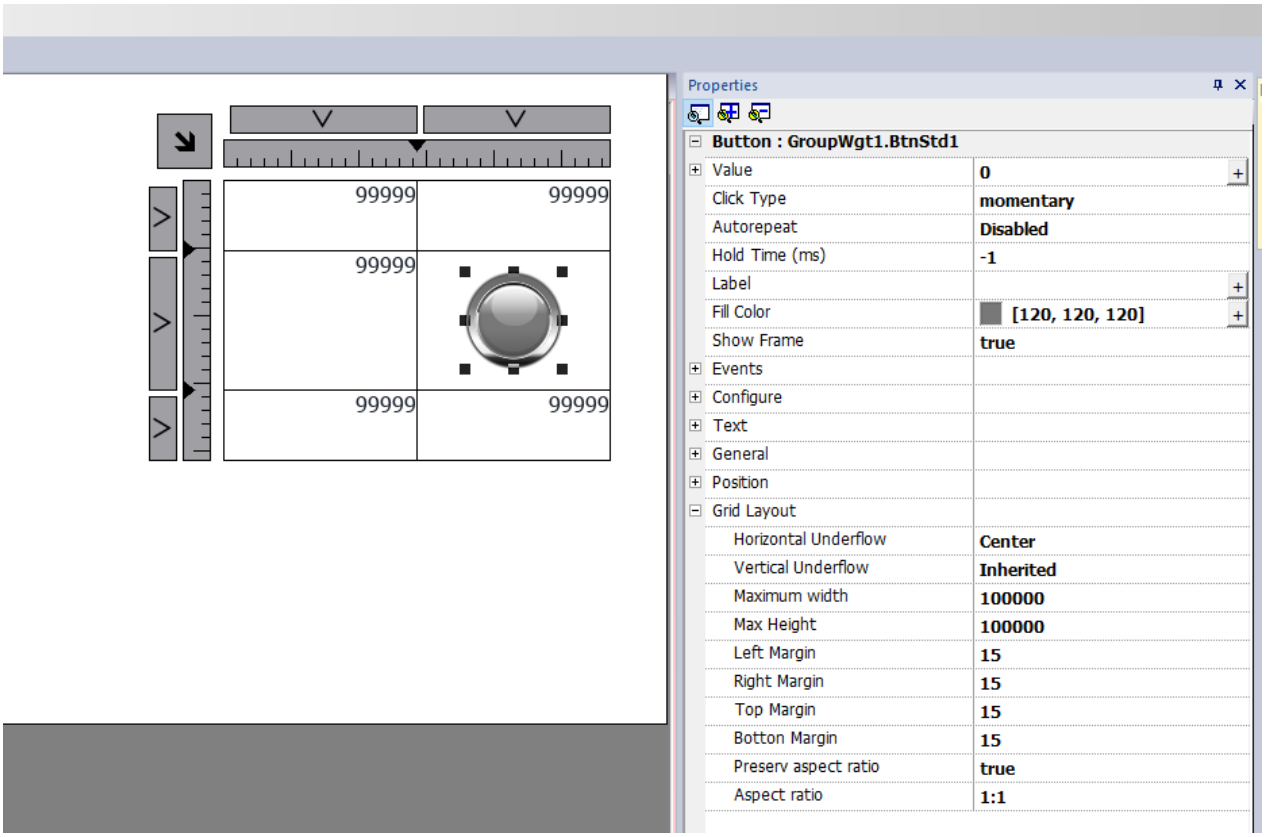
Selection parameters

The selection parameters is available only when the grid is used inside a Table Widget (see "Table widget" on page 302 for details)

Parameter	Description
Forground color **Background color** **Stroke color**	Colors that the row assume when it is selected The list of colors is related with row templates. First color is for row template 0, second color is for row template 1, and so on.

Cells Properties

Properties of a single cell are available inside the properties panel when a cell is selected. To select a cell: first double click the widget group, then click the cell to select.



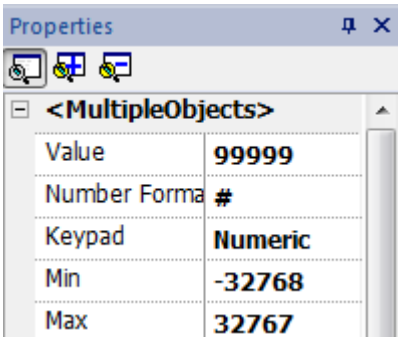
Parameter	Description
Horizontal underflow **Vertical underflow**	This parameter defines the behavior of the widget when the cell is larger than the size defined for widget. • Inherited Inherits the value used for the row or column • Left, Center, Right - Top, Middle, Bottom Defines the position of the widgets when cells are bigger than the maximum defined sizes
Max width **Max height**	Overwrite global grid parameters
Left margin **Right margin** **Top margin** **Bottom margin**	Overwrite global grid parameters Additional pixels that are added to the total margin.
Preserve aspect ratio	Preserve aspect ration of the widget
Aspect ratio	Available only when "*Preserve aspect ratio*" is true

Changing multiple widgets properties

You can set the properties of more widgets of the same type all at once.

To change properties:

1. Select widgets.
2. Set common properties from **Properties** pane.
3. When multiple widgets are selected, the Properties pane title changes to **<MultipleObjects>**: all changes will be applied to all selected widgets.



Note: Not all properties can be modified for multiple widgets simultaneously and must therefore be modified individually.

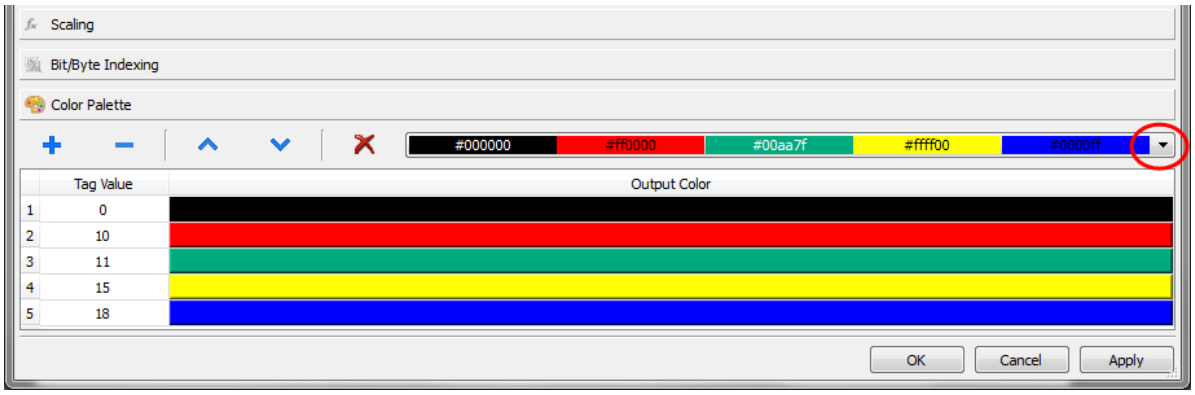
Changing fill color property according to tag values

Wizard allows to change the color property of a widget dynamically, based on tag values in two ways:

- Using ColorPalette
- Connecting the Color property to a String type tag

Changing color property using ColorPalette

1. Create the tag (internal or PLC) that you want to refer to for color management. The tag can be of any data type. On the basis of the value of this tag, the color will change.
2. Attach this tag to the **Fill Color** property of an object (for example, a button).
3. In the same dialog select the **ColorPalette** tab and add the colors that will be used for the object according to the tag value.



Note: The last used colors' tables are saved and can be reused selecting them from the colors list box on the toolbar.

Changing color property connecting Color property to a String type tag

1. Create the tag (internal or PLC) that you want to refer to for color management. On the basis of the value of this tag, the color will change. The tag must be of String type and the **Arraysize** property of the tag must be big enough to contain the string formatted as explained here.
2. Attach this tag to the **Fill Color** property of an object (for example, a button).
3. Write in the **String** tag the RGB color code of the required color. Use one of these formats:

- **#XXYYZZ**, Where XX, YY and ZZ are the RGB components of the needed color expressed in Hexadecimal format (range 00–FF).
- **rgb(XXX,YYY,ZZZ)**, where XXX, YYY and ZZZ are the RGB components of the needed colors expressed in Decimal format (range 0–255).

Note: This feature can be applied to all the objects available in the Widget gallery that have a color property. The run-time change of the color is possible only thanks to the properties of the SVGs that are composing the object. This feature can not be applied to other image formats such as JPEG or BMP files.

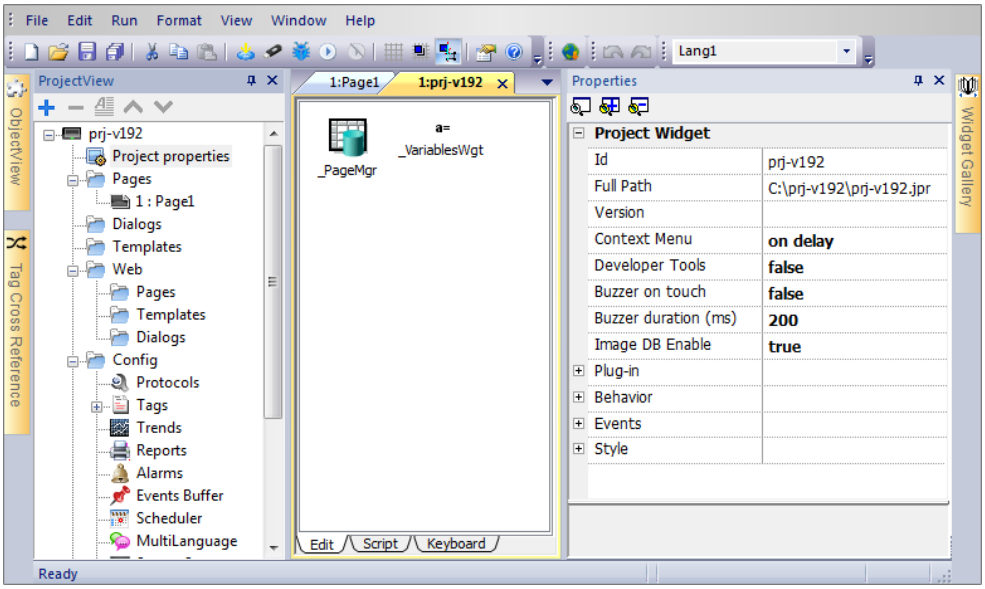
6 Project properties

Project properties contain settings for the project.

Project properties pane

*Path: **ProjectView**> double-click **Project properties**> **Properties** pane*

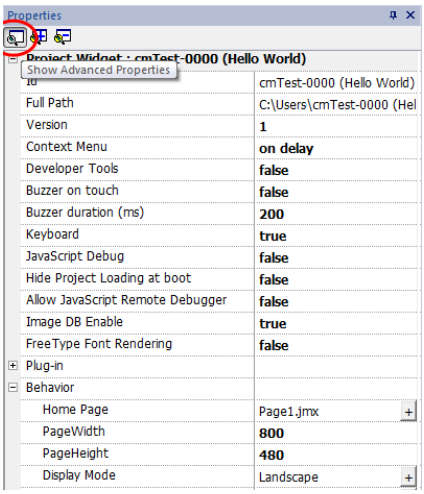
The project **Properties** pane contains a list of project level user-configurable data.



Basic and advanced properties

To view all project properties:

- Click **Show Advanced Properties** button to expand the property view in the **Properties** pane.



Main properties description

Note: Some properties are displayed only in advanced mode.

Property	Description
Version	The Version field is available for users to report the project version.
Context Menu	Define how context menu should appear in the HMI project. **on delay** = context menu appears touching/pressing and holding for a few seconds an empty area of the runtime screen, or via **Context menu** action **on macro command** = context menu appears only via **Context menu** action. See "Widget actions" on page 149 for details.
Developer Tool	Enable/disables a collection of runtime debugging utility tools.
Buzzer on Touch	Enables buzzer when touching a widget on HMI device screen. Supported widgets: • buttons • hotspots • needles • fields • external keys • combo boxes • tables items • control list items  *On Windows CE panels, available for from v1.76 ARM / 2.79 MIPS.*
Buzzer duration	Default 200 ms
Keyboard	Enables the use of keyboard macros at run time when using external keyboards.
JavaScript Debug	Enables the JavaScript debugger at run time for the current project.
Allow JS Remote Debugger	Enables JavaScript remote debugger for current project.  Remote debugging not supported on HMI Client.
Hide Project Loading at boot	When hidden, the splash screen stay on the screen until the application is ready to run.  *Available for Windows CE from v1.99 ARM*
Image DB enable	Activates an engine used by the Runtime to optimize project performance.  **WARNING: This property should only be disabled by technical support for debugging purposes since this might reduce performance at run time.**

Property	Description
FreeType Font Rendering	Switches to FreeType the font rendering used by Wizard and runtime.
Software plug-in modules	Defines which software modules are downloaded to the Runtime with the project. See "Software plug-in modules" on page 59
Behavior	These properties define different aspects of page behavior. See "Behavior" on page 60
Style	Combo Box View Mode (see "Combo Box widget" on page 284 for details) • Context • Full Screen

Developer tools

Collection of runtime debugging functions that can be enabled or disabled.

Enabling developer tools

1. In **Properties** pane, set **Developer Tools** to **true**.
2. Download the project.
3. Open context menu.
4. Select **Developer tools**.

Developer tool list

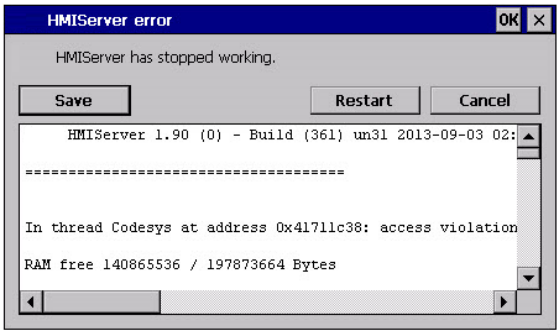
Tool	Description
Show/Hide all	Shows a dialog containing information about device status like CPU load, memory usage, event queues.
CPU statistics	Shows information on CPU load. See "CPU Statistics" on the facing page.
Memory statistics	Shows information about system RAM . A negative value indicates that free memory is decreasing.
Event queues	Shows information on event queues (size, maximum achieved size, number of processed events, last and maximum processing time). Timing statistics are only available for non-UI queue.
Timelog summary	Show page loading time.
Embed window	Allows embedding in runtime the scene or leave the developer tool window as a standalone window (dialog).
Reset queue stats	Resets statistical information on event queues.

Tool	Description
Disable watchdog	Disable the watchdog function and prevents system restart in case of freeze or crash of services.
Ignore exceptions	Disables crash report function, exceptions are not saved in the crash report window.
Launch VNC	Launches the VNC server if available in runtime. VNC server is available as a plugin for Windows CE runtime only.
Profiling	Measures the time spent for loading/rendering the active page. See "Profiling" on the next page

Watchdog

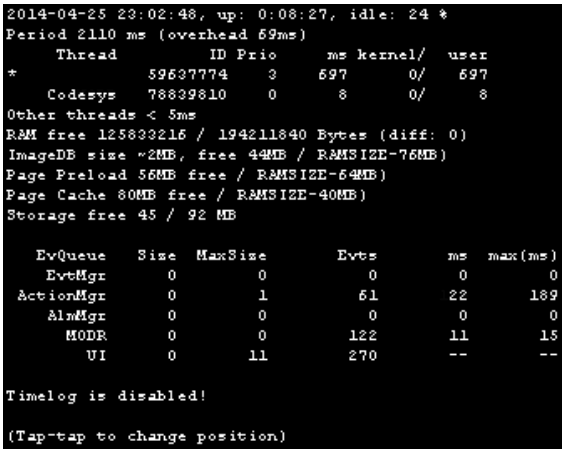
This feature allows you to disable the watchdog. This way you can avoid system restart in case of a runtime crash and have the time to save the crash report or check system status information (for example,memory available, CPU load, events queue size and so on).

The crash report dialog is displayed automatically in case of a system freeze or crash allowing users to save a log file of crash.



Important: Save this file for technical support.

CPU Statistics

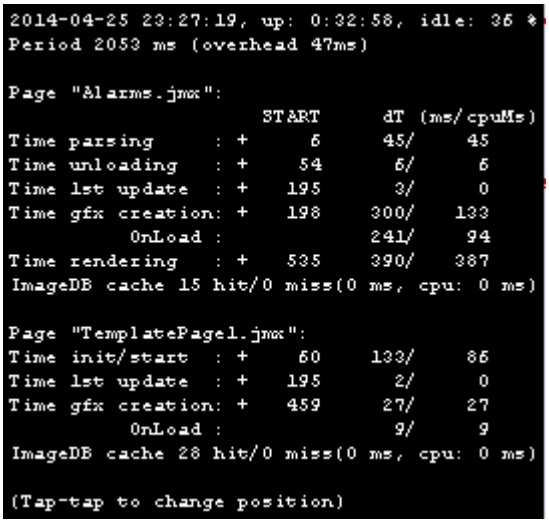


On the top row the current machine time is shown along with the total device uptime.

CPU statistics are collected with a frequency of 2000 milliseconds. The actual period and the overhead required to collect and visualize statistics are displayed as well. The more the actual period is far from the nominal 2000 milliseconds the higher is the system load. CPU consumption of threads is listed reporting the name of the thread (if available, main thread is marked with a *), the thread ID, the thread priority and CPU time spent during the 2000 milliseconds period, divided in user and kernel time.

Profiling

Profiling allows you to check time spent for loading/rendering the active page. Profiling will start from the next page load and will be active only for the first painting of the page to the screen (the configuration is retained).



Profiling option	Description
Enable timelog	Enable timelog capture. Timing will be visible inside the “Timelog summary” window.
Save timelog to file	Saves a report of profile details and the time spent loading a project and its pages into a timelog.txt file. This file can be exported and shared for further analysis. **Important: The execution of this function may reduce page change performance.**
Overlay OnLoad times **Overlay Rendering times**	This view allows displaying time spent on single widgets and is available only for the rendering and OnLoad steps. The view gives an immediate feeling of where time is spent. Red zones represent the most time critical zones. Detailed widget times are visualized by a tooltip window (on Win32 platform attached to mouse over event, on Windows CE press drag and release over the region of interest). In case of out-of-the-scene widgets some arrows allow to navigate to these areas and hovering on them the tooltip will show the area summary
Select overlay color	Select the overlay color to use

Timelog data

Data	Description
Time parsing	Time spent parsing current page. Depends on page complexity/number of widgets.
Time gfx creation	Time spent for image rendering. Mainly related to the *Onload* method.
Time rendering	Time spent rendering the page.
Time unloading	Time spent unloading the page, if current page depends from another page.

Times are provided in couples: wall time/CPU time. Wall time is the absolute time required by this part which can be higher than the actual CPU time required since higher priority threads are also running (for instance protocols). The start time column refers to the page load start time. It can be used to track the actual time required to load a page, since partial times only refer to the most time critical functions and do not include other times that often contribute significantly to the total time.

For example, the actual total wall time required to load a page is rendering (which is the last step) start time + rendering wall time.

FreeType font rendering

New projects use the FreeType font engine as default. Projects created with older versions of Wizard could use an older font engine also after project conversion to avoid any backward compatibility issue.

Switch to FreeType whenever possible for better page rendering.

Once you have switched to the new font rendering, save the project and verify that all texts are displayed correctly in all project pages.

Font rendering issues

When switching to the FreeType font engine a project created with the older font engine, you may experience the following problems:

- text requires more/less pixels for rendering thus changing text layout
- widgets are resized to accommodate text
- better rendering can be obtained using antialiasing (antialiasing is a text widget property)

Software plug-in modules

You can choose which software modules are downloaded to the runtime with the project. Software plug-in has been designed to reduce memory requirements for the HMI application in HMI devices where storage is limited. This option is not supported in Win32 platform

Software plug-in:

- WebKit (module required by browser widget – if available)
- PDF Reader
- VNC Server

Note: Not all software plug-in modules are compatible with all HMI device platform.

Once enabled, software plug-in become part of the runtime. Use Wizard to install it using one of the following procedures:

- install Runtime/update Runtime
- update package

To remove plug-ins from runtime use one of the following functions in System Mode:

- format flash
- restore factory settings

Important: The system cannot detect automatically which software plug-ins are required by the HMI application, make sure you select them all in the Project Properties.

Note: Software plug-in support has been designed for embedded HMI devices where storage is limited. This option is not supported in Win32 platform.

Behavior

These properties define various elements of page behavior.

Home Page

The first page loaded at run time (after log-in page if security is enabled in project).

When security is enabled, you can specify a different homepage for each groups of users. In this case this setting is ignored. See "User management and passwords" on page 227 for details.

Page Width/Page Height

Defines the default size in pixel of an HMI page. Default is the display resolution of the HMI device model selected when creating the project.

Display Mode

Defines HMI device orientation.

Project Type

Defines HMI device type for the project. According to the model, some project features and properties are automatically adjusted.

WARNING: Starting from v2, the HMI Runtime will check if the selected project type is matching with the HMI device model and will advise with a message when the selected type is not matching: “HMI Type mismatch. Convert project and download again.”

Panel Memory

Size of the available internal panel memory.

PageRequest, CurrentPage and SyncOptions

It is possible to have HMI Runtime exchange devices information on the page shown by the HMI. You can synchronize pages shown on the HMI device and on HMI Client or to control an HMI project from a controller such as a PLC.

The following properties can be customized:

Property	Description
PageRequest	Page to be shown on the HMI device and on HMI Client. Attached tag must contain an integer value within the range of the available project pages and must be available at least as a Read resource.
CurrentPage	Page number displayed on the HMI device or on HMI Client or on both. Attached tag must be available at least as a Write resource and must have integer data type.
SyncOptions	Synchronization of project pages with the value contained into the **CurrentPage** property. Options can be: • **disable**: page number value is ignored, • **local**: page number displayed on HMI, • **remote** : page number displayed on HMI Client. • **local + remote**: page number displayed on HMI and on HMI Client, if different pages are displayed the last page loaded is considered.

Example: forced page change from controller/PLC to HMI device and HMI Client

Set properties as follows:

PageRequest	attached to tag "A"
CurrentPage	empty
SyncOptions	disable

Set value of tag "A" to display the requested page on HMI device and HMI Client.

Example: forced page change from controller/PLC to HMI and HMI Client. Read current page loaded on HMI

Set properties as follows:

PageRequest	attached to tag "A"
CurrentPage	attached to a tag "B" as read/write
SyncOptions	local

Set value of tag "A" to display the requested page on HMI device and HMI Client. Tag "B" will contain the number of page currently shown by the device.

Example: forced page change from controller/PLC to HMI device and HMI Client. Read current page loaded on HMI Client.

Set properties as follows:

PageRequest	attached to tag "A"
CurrentPage	attached to a tag "B" as read/write
SyncOptions	remote

Set value of tag "A" to display the requested page on HMI and HMI Client. Tag "B" will contain the number of page currently shown by HMI Client.

Example: forced page change from controller/PLC to HMI device and HMI Client. Force HMI Client page synchronization with HMI device (not vice versa).

Set properties as follows:

PageRequest	attached to a tag "A" as Read/Write
CurrentPage	attached to the same tag "A" as per **PageRequest**
SyncOptions	local

Set value of tag "A" to display the requested page on HMI and HMI Client. Change page on HMI to display the same page on HMI Client.

Example: forced page change from controller/PLC to HMI device and HMI Client. Force HMI page synchronization with HMI Client (not vice-versa).

Set properties as follows:

PageRequest	attached to a tag "A" as read/write
CurrentPage	attached to the same tag "A" as per **PageRequest**
SyncOptions	remote

Change value of tag "A" to display the requested page on HMI and HMI Client. Change page on HMI Client to display the same page on HMI.

Example: synchronize displayed page between HMI device and on HMI Client

Set properties as follows:

PageRequest	attached to a tag "A" as read/write
CurrentPage	attached to the same tag "A" as per **PageRequest**
SyncOptions	local+remote

Changing page on HMI device, same page will be shown on HMI Client and vice-versa.

WebPageRequest

You can synchronize pages shown on the SH4Web Clients from a controller such as a PLC.

The following properties can be customized:

Property	Description
WebPageRequest	Page to be shown on the SH4Web Client. Attached tag must contain an integer value within the range of the available project pages and must be available at least as a Read resource.

Hold Time/Autorepeat Time

Defines the values for hold time and autorepeat time for buttons and external keyboards.

Note: These properties can be redefined for each button or key in their widget property table.

Web Inactivity Timeout

Defines a timeout for SH4Web client. When the timeout expires without any activity the current user is logged out.

Range	1–86400 s (form 1 s to 24 h)
Default value	600 s
Values	0 = disabled

Web Icon

The favorite icon associate at the web pages

Refresh Time

Defines the refresh time for the communication between the runtime and SH4Web clients.

Range	500–10000 ms
Default value	3000 ms

Browser Optimization

true	Web engine optimization enable (default)
false	Web engine optimization disable (useful for old browsers that not support the web engine optimization)

Max Bandwidth (Kbs)

Limit for maximum data sent by server (useful for old slow browsers). Set to 0 to use all the available bandwidth (default)

Web clients connection mode

Auto	The connection mode is selected by the client (default)
SSE	Force the Server-Sent Events mode
Long Polling	Force the Long-polling mode

Target Zoom Factor

It is the zoom factor of the HMI device that will be applied when project is loaded at run time.

Range	0.3–2.9
Default value	1 = no zoom

Background color option

When the defined page is smaller of the entire display area, colorize the area that is not covered from the page (for example when page is Zoom Out)

Property	Description
None	Old mode, color is white (default)
Selected color	Color to use
Page background	Auto adjust color based on background of template or of page

Events

OnWheel

Used only in conjunction with wheel input devices. Normally the wheel is used to increase/decrease the value of a tag without an external keyboard device.

Attach this property to a change of wheel event and use an action like **StepTag** to increase/decrease a tag value.

7 The HMI simulator

HMI simulator allows you testing projects before downloading it to the HMI device. It may be used to test the project when no HMI device is available and to speed up development and debugging activities.

The HMI simulator supports:

- online simulation - in communication with real devices (only for protocols with Ethernet or RS-232 communication),
- offline simulation - simulating tag behavior

The data simulation method is set in the **Simulator** column of the Tag Editor.

Data simulation methods

Set tag simulation behavior in the **Simulator** field of Tag Editor.

Method	Description
Variables	Data is stored in a simulator variable. This variable holds the value of the tag so you can read and write the value.
SawTooth	A count value is incremented from **Offset** to **Amplitude + Offset** value with a **Period** of 60..3600 seconds. When the counter reaches **Amplitude + Offset**, the value is reset to **Offset** and the counter restarts.
Sine Wave	A sine wave value is generated and written to the tag value. **Min, Max** and **Period** values can be defined for each tag.
Triangle Wave	A triangle wave value is generated and written to the tag value. **Min, Max** and **Period** values can be defined for each tag.
Square Wave	A square wave value is generated and written to the tag value. **Min, Max** and **Period** values can be defined for each tag.

See "Adding tags" on page 23 for details.

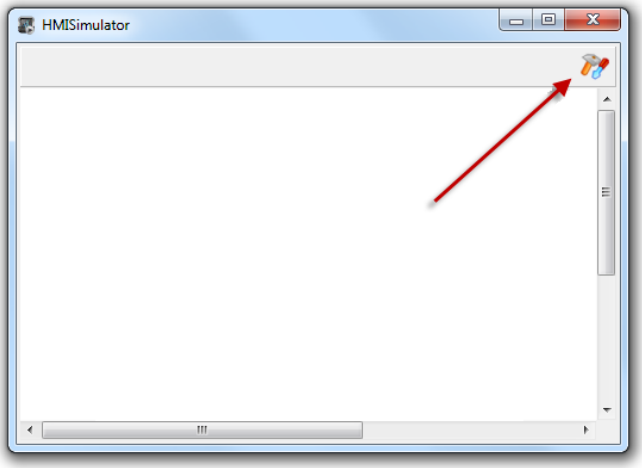
Simulator settings

The Simulator works by default with simulated protocols. It can also work with real protocols (Ethernet or serial protocols)

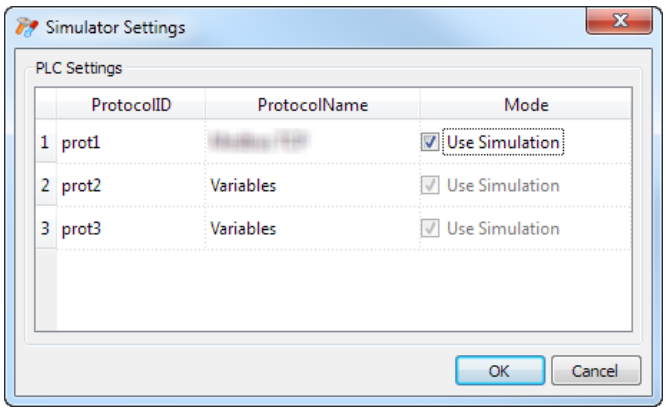
Note: For protocols not supporting communication with external devices, such as the Variables protocol, this option is always disabled.

Changing simulated protocols

1. Click the simulator **Settings** icon.



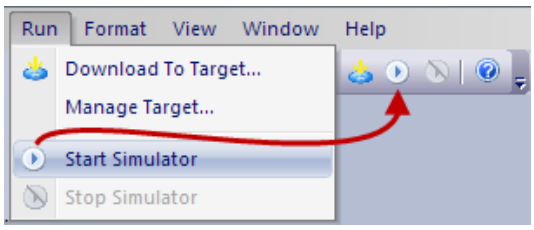
2. Select **Use Simulation** to use simulated protocols, otherwise real protocols will be used for communication with external devices.



Launching and stopping the simulator

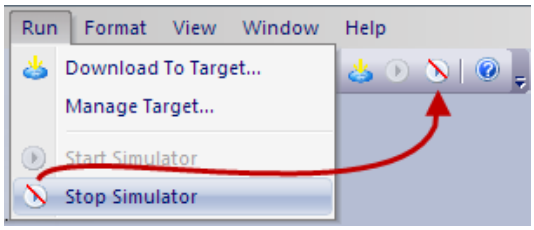
To launch the simulator:

1. On the **Run** menu, click **Start Simulator:** the Simulator runs on the computer in the same way as the server would run on the HMI device.



To stop the simulator:

1. On the **Run** menu, click **Stop Simulator** or on the simulated page double-click the **Exit** button.



8 Transferring the project to HMI device

To transfer the Wizard project to the target HMI device you can use:

- function **Run > Download to Target**
- function **Run > Update Package** with the use of a USB device

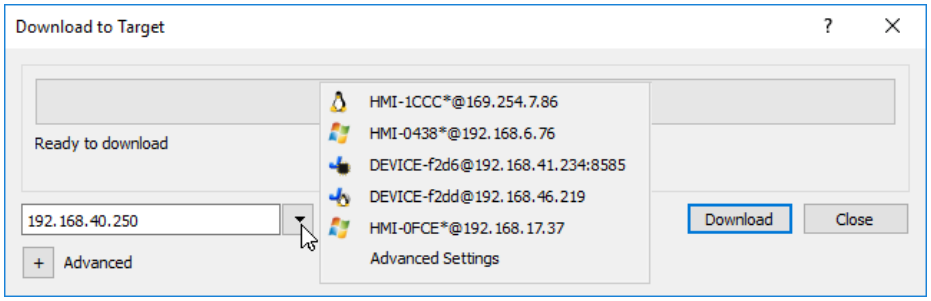
Download to HMI device

Path: ***Run> Download to Target***

This function transfers project and HMI Runtime via Ethernet .

Note: The HMI device must have a valid IP address. See "HMI device basic settings" on page 6 for details on how to assign an IP address.

1. Click the discovery button: a list of the detected IP addresses is displayed.
2. Select the HMI device IP address.

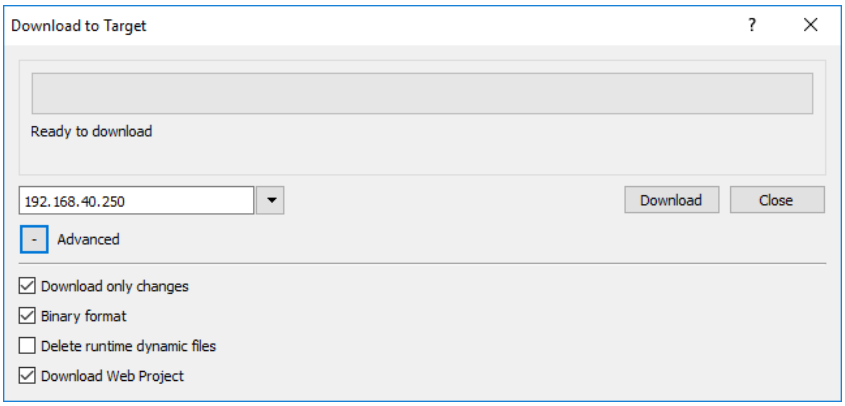


You can even enter the IP address manually or, if available, the host name provided by a DNS server. Using a service tool like Bonjour, Linux-based HMI devices can be discovered using their hostname (e.g HMI-0d37.local). Bonjour is a trademark of Apple inc.

3. Click **Download**: Wizard will switch the HMI device to Configuration Mode and transfer the files.

When the download operation is completed, the HMI device automatically switched back to Operation Mode and the project is started.

Advanced options



Option	Description
Download only changes	Transfers to the HMI device only the modified project files.
Binary format	Download files using binary format.

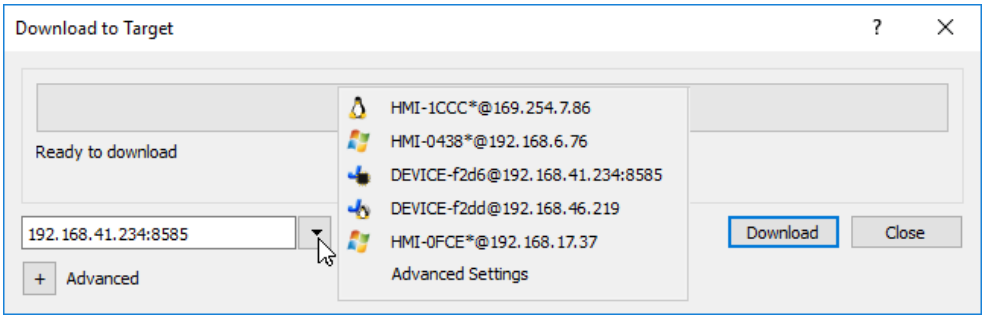
Option	Description
Delete runtime dynamic files	Modified configuration of recipes, users, schedulers, etc. done at run time will be deleted and overwritten by the configuration defined in the project. ***CAUTION: This operation cannot be undone, deleted dynamic files cannot be restored.*** ***CAUTION: Dynamic files are not deleted if stored on external devices (USB or SD Cards).***
Download Web Project	Download the SH4Web pages to HMI device.

When transferring a project, Wizard uses a combination of HTTP and FTP connections:

- HTTP connection - issues the commands to switch to transfer mode or to unload running project,
- FTP session - transfers the files to the flash memory in the HMI device.

Advanced Settings

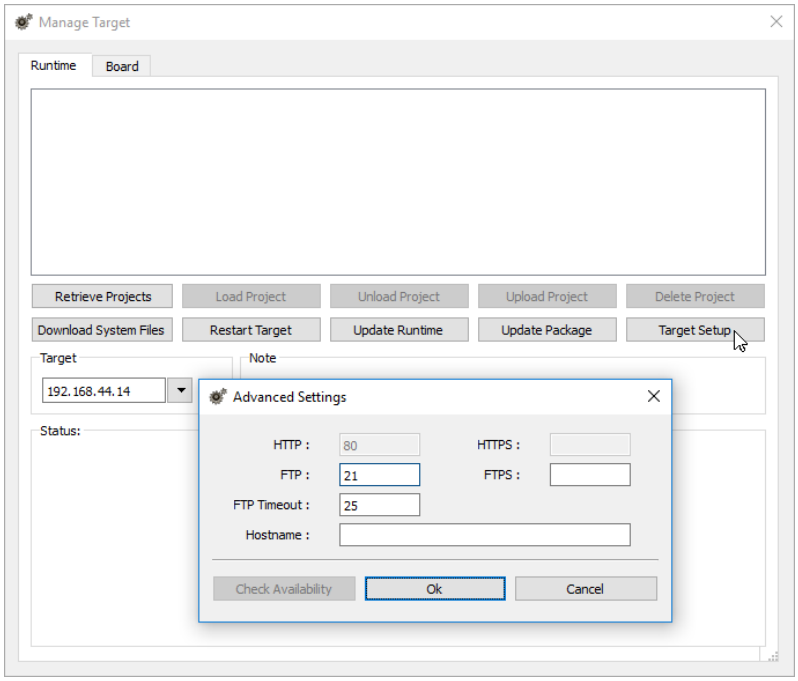
Using the “Advanced Settings” option, you can define the ports to use, but generally, you do not need to enter this information because HMI devices will provide the ports to use inside the panesl list.



Changing HMI device connection settings

Path: ***Run> Manage Target***

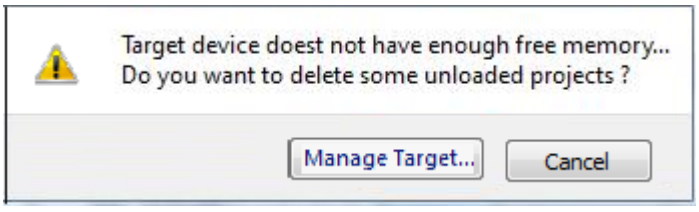
1. Click **Target Setup**: the **Advanced Settings** dialog is displayed. Default port for HTTP connections on the HMI device is port 80.



2. Set correct HTTP, FTP or HTTPS, FTPS ports for the HMI device. (These are the ports used by the system to connect to the HMI device and may need to be modified when default ports are used by other services or applications or if the local network requires specific settings.)
3. Specify **Hostname** to easily identify each device in a network where multiple devices are available. The default hostname is “HMI” for all devices.
4. Click **Download System Files**. At the next download the new ports will be used in the HMI device and new hostname will appear in the drop-down list

Managing big projects

For successful download the project size should be at least 2 MB smaller than the available memory. If not, you run out of flash memory in the HMI device and a warning message is displayed.



To free more memory:

1. Click **Manage Target**.
2. Delete the projects you no longer need t to make more memory available.

Update package

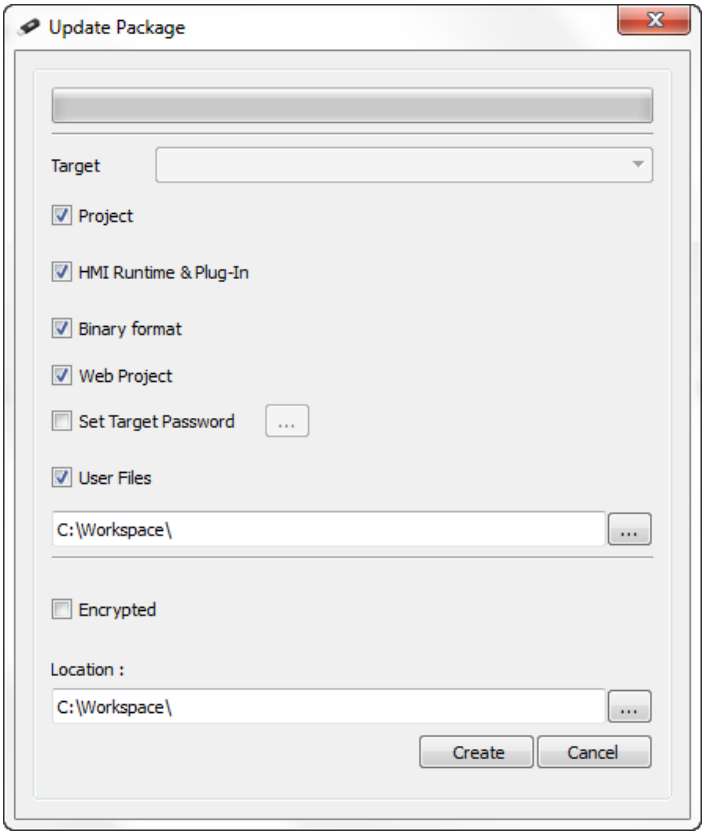
To install or update HMI Runtime and project you may create a package to be loaded via USB.

Important: Always include both project and the Runtime in the update packages.

If you need to use an old project with the latest Runtime version, convert the project first. See "Installing the application" on page 2 for details.

Creating an update package

Path: ***Run> Update Package***



Option	Description
Target	HMI device type. Selected automatically if the project is open.
Project	Adds open project to update package.
HMI Runtime & Plug-In	HMI Runtime is added to the update package. If the project is open the required plugins are also added to update package.
Binary Format	Download files using binary format.
Web Project	Download the SH4Web pages to HMI device.
Set Target Password	Sets password to perform critical tasks (for example, project download/upload , board management) See "Protecting access to HMI devices" on page 415.

Option	Description
User Files	Selects files to be copied to the QTHM folder of HMI device. Max size 5 MB
Encrypted	Enables encryption of update package so that it can only be unzipped by the HMI Runtime.
Location	Location of update package.

Example of user's file location

Computer:

C:\Users\Username\Desktop\myFolder

- *subFolder1/file1*
- *subFolder1/file2*
- *file3*
- *file4*

WinCE devices:

/Flash/QtHmi

- *subFolder1/file1*
- *subFolder1/file2*
- *file3*
- *file4*

Linux devices:

/mnt/data/hmi/qthmi

- *subFolder1/file1*
- *subFolder1/file2*
- *file3*
- *file4*

Note: User files copy is available only from the USB key.

Loading an update package

Path: from the context menu > ***Update***

1. Assuming you have stored the package in the root folder of a USB drive, remove the drive from the computer, plug it in the HMI device, display the context menu by holding your finger for a few seconds on the screen and select **Update**.
2. The system will check for the presence of the update package in the USB drive root and ask confirmation to proceed with the update.

3. Select **Auto select best match** and click **Next**: the procedure is completed automatically. Alternatively use the browser button to select the file to use.

The Runtime loader

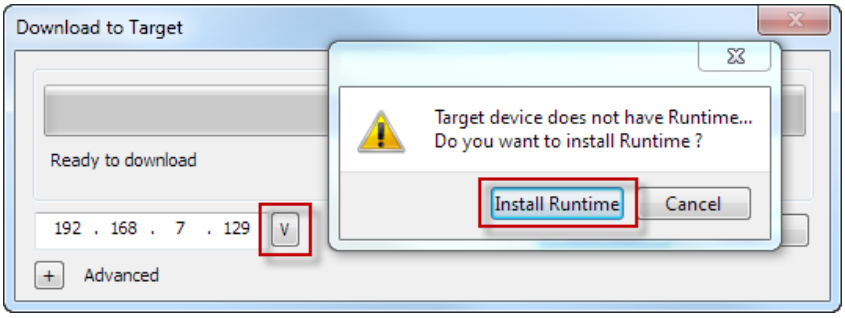
HMI devices are delivered from factory without Runtime.

When you power up the device for the first time, the Runtime Loader window is displayed (see "System Settings" on page 373for details)

The Runtime Loader presence depends on the device Operating System and may not be available on all the units. Old versions of HMI devices may not include the Runtime Loader. Contact technical support if you need further information.

Installing Runtime from Wizard

When you download a project the Runtime is automatically installed if needed.



See "Transferring the project to HMI device" on page 69 for details.

1. Click **Install Runtime**: the procedure is run automatically.

Installing Runtime from a USB drive

1. Prepare the Update Package as described in "Update package" on page 72
2. Plug the USB drive in the device and follow the instructions for the type of device (see "System Settings" on page 373for details)

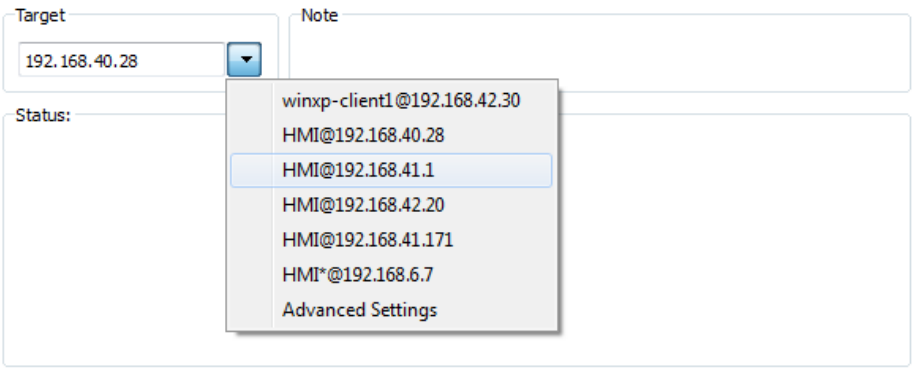
Note: Old versions of HMI devices may not support automatic installation of Runtime. Contact technical support for more information.

Upload projects

Path: ***Run*** *>* ***Manage Target***

You can copy a project from the Runtime to the computer where Wizard is running.

1. In the **Runtime** tab, select the IP address of the device from the drop-down list **Target**.



2. Click **Retrieve Projects**: a list of all the projects available is displayed.
3. Select project to upload
4. Click **Upload Project**

Upload could be password protected. See "Protecting access to HMI devices" on page 415 for details.

5. If required, enter password. The upload process starts.

A copy of the project is saved in:

C:\Users\username\Documents\Wizard\workspace\Uploaded\RuntimeIPAddress\workspace\ProjectName

Note: If the upload operation fails, check firewall settings the computer where Wizard is running.

System Variables

This chapter describes how to access to HMI System Variables.

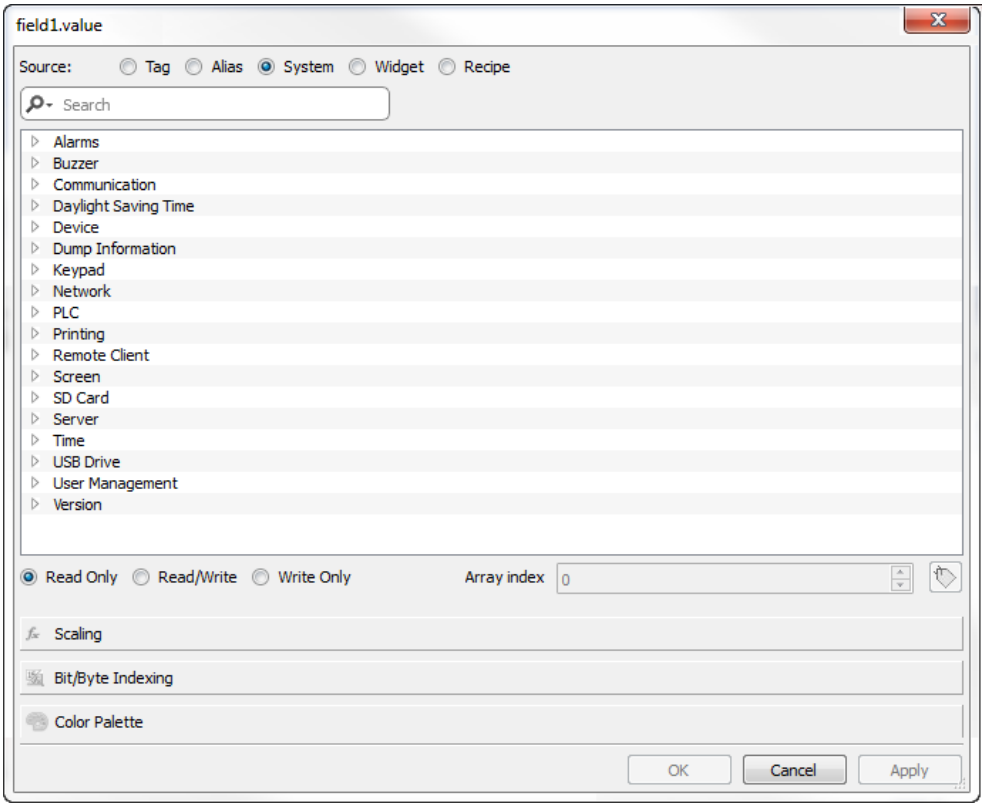
Chapter	Description
Attach To	Explains how widgets can be linked to System Variables by using Attach To window
Protocol	Explains how System Variables can be defined as Tags within a protocol

10 System Variables

Path: ***Source> Attach to***

System variables are special tags containing information about the HMI runtime.

Note: System Variables are available also as a standard protocol in the Protocol Editor. Use System Variables as a protocol when you have to transfer data between system variables and tags from devices, or to select custom refresh rate for a system variable.



Alarms variables

Number of alarms of the requested type.

Variable	Description	Data type
Alarm not acknowledged	True when alarms unacknowledged is pending (Not Triggered Not Acknowledged<>0) OR (Triggered Not Acknowledged<>0)	boolean read only
Alarm triggered	True when at least one alarm is triggered (Triggered Acknowledged<>0) OR (Triggered <>0) OR (Triggered Not Acknowledged<>0)	boolean read only
Number of missed alarm events	Alarms exceeding the event queue. Queue length is defined in the *engineconfig.xml* file.	int read only
Number of not triggered acknowledged	Alarm condition no longer active; alarms already acknowledged	int read only
Number of not triggered not acknowledged	Alarma condition no longer active; awaiting acknowledgment	int read only
Number of triggered acknowledged	Alarm condition active; alarms already acknowledged	int read only
Number of triggered alarms	Alarm active: aknowledgement not required	int read only
Number of triggered not acknowledged	Alarm condition active; awaiting acknowledgment	int read only

Note: For compatibility reasons, the older names are still valid but they usage is deprecated.

Buzzer variables

Adjust buzzer behavior.

Variable	Description	Data type
Buzzer Setup	**0** = disabled **1** = enabled (buzzer sounds as audible on any touchscreen event) **2** = buzzer status controlled by **Buzzer Control** system variable or by **Buzzer on Touch** property inside the "Project properties" on page 53  **Buzzer on touchscreen (Setup=1) is not available on Linux platforms. See "*Buzzer on Touch*" property in alternative.**	int
Buzzer Control	**0** = buzzer off **1** = buzzer on **2** = buzzer blink	int
Buzzer Off Time	Duration in milliseconds of off time when blink has been selected. Default = 1000. Range: 100–5000.	int
Buzzer On Time	Duration in milliseconds of on time when blink has been selected. Default = 1000. Range: 100–5000.	int

Communication variables

Communication status between HMI device and controllers.

Variable	Description	Data type
Protocol Communication Status	Summarize the status of the communication protocols. **0** = No protocol running, protocol drivers might not have been properly downloaded to the HMI device. **1** = Protocols loaded and started, no communication error. **2** = At least one communication protocol is reporting an error.	int Read only
Protocol Error Message	Communication error with error source. For example: "[xxxx]" where "xxxx" is the protocol abbreviation, the error source. Multiple acronyms appear in case of multiple error sources. Blank when no errors are reported.	ASCII string Read only
Protocol Error Count	Number of communication errors occurred since last reset. Reset value with Reset Protocol Error Count action, see "System actions" on page 134.	int Read only

Daylight Saving Time variables

Information on the system clock. The variables contain information on the "local" time. Standard Time (solar time) and Day Light Saving time (DST) are available.

Note: All variables are read only; you cannot use them to update the system clock.

Variable	Description
Standard Offset	Offset in minutes when standard time is set, with respect to GMT (for example: -8x60 = -480 minutes).
Standard Week	Week in which the standard time starts (for example: First = 1).
Standard Month	Month in which the standard time starts. Range: 0–11. (for example: November = 10).
Standard Day	Day of week in which the standard time starts (for example: Sunday = 0).
Standard Hour	Hour in which the standard time starts (for example: 02 = 2).
Standard Minute	Minute in which the standard time starts (for example: 00 = 0).
DST Offset	Offset in minutes when DLS time is set, with respect to GMT
DST Week	Week in which the DLS time starts
DST Month	Month in which the DLS time starts. Range: 0–11.
DST Day	Day of week in which the DLS time starts
DST Hour	Hour in which the DLS time starts
DST Minute	Minute in which the DLS time starts

Device variables

Device settings and operating status information.

Variable	Description	Data type
Available System Memory	Free available RAM memory in bytes.	uint64 read only
Backlight Time	Activation time in hours of the display backlight since production of the device.	unsignedInt read only
Battery LED	Enables/disables the low battery LED indicator (when available). **0** = disabled **1** = enabled Not available on Linux platforms (see "HMI devices capabilities" on page 429 for panels details)	int
Battery	Reserved	int

Variable	Description	Data type
Timeout	Not available on Linux platforms (see "HMI devices capabilities" on page 429 for panels details)	
Display Brightness	Returns and adjusts brightness level. Even when set to 0, the backlight is still on and the **Backlight Time** counter increases. Range: 0–255 On WinCE device only: When set to a low light level (0..3), the backlight stays lit to a higher level for 8 seconds to allow the user to make the adjustments and then is switched-off.	int
External Timeout	Non-operational time after which the display backlight is automatically turned off. The backlight is automatically turned on when the user touches the screen. **-1** = switch off backlight and disable touch (switch display off). **Backlight Time** counter is stopped. **0** = switch backlight on (switch display on) **1..n** = timeout, in seconds, for switch off backlight (screen saver timer) On Linux device (see "HMI devices capabilities" on page 429 for panel details), timeout is managed in minutes. The entered value is converted to minutes rounded to next value, for example, 60, 120, 180.	int
Flash Free Space	Free space left in internal Flash memory.	uint64 read only
Manufacturer Code	Internal code that identify the HMI type	unsignedInt read only
System Font List	List of system fonts	string read only
System Mode	Runtime operation status. **1** = booting **2** = configuration mode **3** = operating mode **4** = restart **5** = shutdown	int
System UpTime	Time the system has been powered since production of the unit (hours).	unsignedInt read only

Dump information variables

Status of the copy process to external drives (USB or SD Card) for trend and event buffers.

Note: If copy time is less then one second, the system variable does not change its value.

Variable	Description	Data type
Dump Archive Status	**1** = event buffer copy in progress	int read only
Dump Error Message	Return error message if any error occurs during the dump operation	string read only
Dump Recipe Status	**1** = recipe buffer copy in progress If the copy duration time is less than 1 second, the system variable does not change its value	int read only
Dump Trend Status	**1** = trend buffer copy in progress	int read only
Reset Recipe Status	**1** = recipe buffer reset in progress If the reset duration time is less than 1 second, the system variable does not change its value	int read only
Restore Recipe Status	Returns information during the copy process of recipes. If the copy duration time is less than 1 second, the system variable does not change its value. **0** = initial default state **1** = operation triggered **2** = operation complete successfully **3** = operation completed with errors	int read only

FTP client variables

The FTP client variables are updated when the FTP actions are used.

Variable	Description	Data type
FTP Current Command	Last FTP command	string read only
FTP Error Message	Last FTP error message	string read only
FTP Progress	Download/upload progress (0/100%)	short read only
FTP Status	Status of last FTP command: • 0 = idle • 1 = active • 2 = done • 3 = error	short read only

Keypad variables

Keypad status.

Variable	Description	Data type
Is keypad open	**0** = no keypad open **1** = keypad open	int read only

Network variables

Device network parameters.

Variable	Description	Data type
Adapters Parameters	This is a JSON string that can be use to read or update the network adapters parameters	string
Gateway	Gateway address of the main Ethernet interface of device	string read only
IP Address	IP address of the main Ethernet interface of device	string read only
Mac ID	MAC ID of the main Ethernet interface of device	string

Variable	Description	Data type
		read only
Status	Contains the result of the last operation required by writing inside the Adapter Parameters. It is updated after each write operation. • Empty string is meaning no errors • Last error descriptions	string read only
Subnet Mask	Subnet Mask of the main Ethernet interface of device	string read only

Printing variables

Information on printing functions.

Variable	Description	Data type
Completion percentage	Percentage of completion of current print job. Range: 0–100	read only
Current disk usage	Folder size in bytes where PDF reports are stored. If *Flash* has been selected as *Spool media type*, this value corresponds to *reportspool*.	read only
Current job	Name of the report the job is processing. Current job is the following: • [report name] for a **Graphic Report** • [first line of text] for a **Text Report**	read only
Current RAM usage	Size in bytes of the RAM used to process the current job	read only
Disk quota	Maximum size in bytes of the folder where PDF reports are stored	read only
Graphic job queue size	Number of available graphic jobs in the printing queue	read only
Last error message	Description of the last returned error	string read only
RAM quota	Maximum size in bytes of the RAM used to generate reports	read only
Status	Printing system status. Values: • **idle**	string read only

Variable	Description	Data type
	• **error** • **paused** • **printing**	
Text job queue size	Number of available text jobs in the printing queue	read only

Remote Client variables

The following system variables are associated to the transferring files to a remote HMI device.

Variable	Description	Data type
Download from HMI error message	Error description	ASCII string read only
Download from HMI percentage	Download progress (0→100)	read only
Download from HMI status	**0** = idle, action is not in use or completed **1**= file download in progress **2** = error	int (32 bit) read only
Upload to HMI error message	Error description	ASCII string read only
Upload to HMI percentage	Upload progress (0→100)	read only
Upload to HMI status	**0** = idle, action is not in use or completed **1**= file upload in progress **2** = error	int (32 bit) read only

Version variables

Operating System and runtime version.

Variable	Description	Data type
Main OS Version	Version of Main OS.	string
Runtime Version	Version of runtime.	string

Screen variables

Screen status.

Variable	Description
Time remaining to unlock	Time remaining to unlock screen (see **LockScreen** action, "Page actions" on page 123)
X Screen resolution	Display horizontal screen size in pixel
Y Screen resolution	Display vertical screen size in pixel

SD card variables

Information on the external SD card.

Variable	Description	Data type
SD Card FreeSpace	Available space on card in bytes	uint64 read only
SD Card Name	Name of SD card	string read only
SD Card Size	Size in bytes of the card plugged in the slot	uint64 read only
SD Card Status	0 = SD card unplugged 1 = SD card plugged	int read only

Server variables

Server status.

Important: All variables refer to server, not to HMI Client.

Variable	Description	Data type
Current page	Name of current page	string
Current project	Name of current project	string
Operating mode time	Seconds elapsed since device started operating mode	uint64
Project load time	Date when the project was loaded on the HMI Runtime as in **System Date** format (milliseconds).	uint64

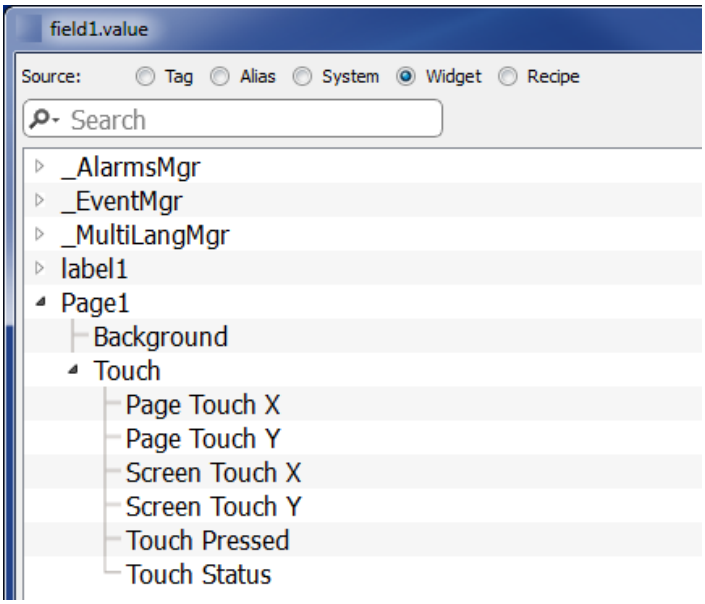
Time variables

System time expressed in UTC format.

Variable	Description	Data type
Day Of Month	Range: 1–31	int
Day of Week	Range: 0 = Sunday, .. , 6 = Saturday	int
Hour	Range: 0–23	int
Minute	Range: 0–59	int
Month	Range: 1–12	int
Second	Range: 0–59	int
System Time	The same as UTC time. It can also be set as date/time for this variable.	unsignedInt
Year	Current Year	int

Touch screen variables

Cursor status and position on the touchscreen. These are properties of the active page and can be selected in the **Widget** section.



Note: Page size can be different than HMI device display size.

Variable	Description	Java Script
Page Touch X **Page Touch Y**	Cursor position related to page	page.primaryTouch.x page.primaryTouch.y
Screen Touch X **Screen Touch Y**	Cursor position related touchscreen	page.primaryTouch.screenX page.primaryTouch.screenY
Touch Press	**0** = screen not pressed **1** = screen pressed	page.primaryTouch.pressed
Touch Status	Generic touch screen changes. This variable contains the concatenation of **Screen Touch X**, **Screen Touch Y** and **Touch Press** values (for example, “924,129,0”). The main usage of this variable is to trigger an event, using the OnDataUpdate feature, when something (x, y or click) is changed.	page.primaryTouchStatus

USB drive variables

Information on the external USB drive connected to the device.

Variable	Description	Data type
USB Drive free space	Available space in bytes	uint64 read only
USB Drive Name	Name of USB device	string read only
USB Drive Size	Size in bytes of the device plugged in the USB port	uint64 read only
USB Drive Status	0 = USB Drive unplugged 1 = USB Drive plugged	int read only

User management variables

Information on users and groups.

Variable	Description	Data type
No of Remote-Clients Alive	Number of HMI Clients connected to the server	short read only
This Client Group-Name	Group of currently logged user	string read only
This Client ID	Only for HMI Clients. Local and remote clients connected to the same server (for example, runtime) get a unique ID.	short read only
This Client User-Name	Name of the user logged to the client where the system variable is displayed.	string read only

System Variables Protocol

System Variables communication driver allows to create Tags that point to system information.

System Variables communication driver is not counted as physical protocol.
Refer to **Table of functions and limits** from main manual in "Number of physical protocols" line.

Protocol Editor Settings

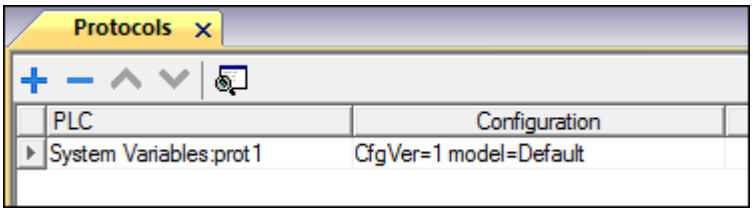
Adding a protocol

To configure the protocol:

1. In the **Config** node double-click **Protocols**.
2. To add a driver, click **+**: a new line is added.
3. Select the protocol from the **PLC** list.

The driver configuration dialog is displayed.

From PLC Model list select the specific System Variables type.



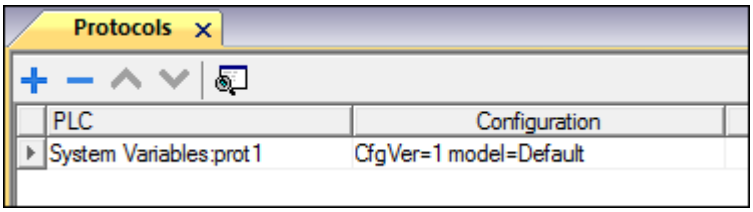
System Variables - Default

System Variables - Default protocol allows to create Tags that point to HMI system variables regarding:

- Alarms
- Buzzer
- Communication
- Database
- Daylight Saving Time
- Device
- Dump information
- Network
- Screen
- SD Card
- Server
- Time
- USB Drive
- Version
- Virtual Com Switch

Protocol Editor Settings

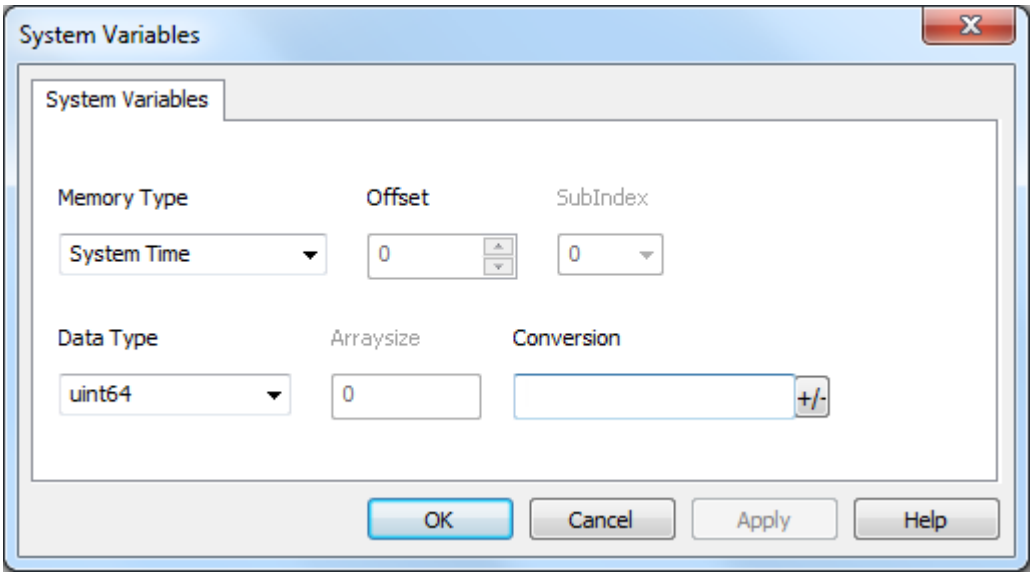
From PLC Model list of Protocol Editor dialog, select Default.



Tag Editor Settings

Path: ***ProjectView*** *>* ***Config*** *> double-click* ***Tags***

1. To add a tag, click **+**: a new line is added.
2. Select **System Variables** from the **Driver** list: tag definition dialog is displayed.



<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td>Memory Type</td><td>Represents the system variable to which the Tag refers to.
The below section shows the full list of possible system variables, grouped by category.
<table>
<tr><th colspan="3">Alarms Variables</th></tr>
<tr><th>Variable Name</th><th>Description</th><th>Data Type</th></tr>
<tr><td>Alarm not acknowledged</td><td>True when alarms unacknowledged is pending
(Not Triggered Not Acknowledged<>0) OR (Triggered Not Acknowledged<>0)</td><td>boolean
read only</td></tr>
<tr><td>Alarm triggered</td><td>True when at least one alarm is triggered
(Triggered Acknowledged<>0) OR (Triggered <>0) OR (Triggered Not Acknowledged<>0)</td><td>boolean
read only</td></tr>
<tr><td>Number of missed alarm events</td><td>Alarms exceeding the event queue. Queue length is defined in the engineconfig.xml file.</td><td>int
read only</td></tr>
<tr><td>Number of not triggered acknowledged</td><td>Alarm condition no longer active; alarms already acknowledged</td><td>int
read only</td></tr>
<tr><td>Number of not triggered not acknowledged</td><td>Alarm condition no longer active; awaiting acknowledgment</td><td>int
read only</td></tr>
<tr><td>Number of triggered acknowledged</td><td>Alarm condition active; alarms already acknowledged</td><td>int
read only</td></tr>
<tr><td>Number of triggered alarms</td><td>Alarm active: acknowledgment not required</td><td>int
read only</td></tr>
<tr><td>Number of triggered not acknowledged</td><td>Alarm condition active; awaiting acknowledgment</td><td>int
read only</td></tr>
</table>
</td></tr>
</table>

Element	Description

Buzzer Variables

Variable Name	Description	Data Type
Buzzer Setup	**0** = disabled **1** = enabled (buzzer sounds as audible on any touchscreen event) **2** = buzzer status controlled by Buzzer Control system variable or by Buzzer on Touch property inside the "Project properties" of main manual  **Buzzer on touchscreen (Setup=1) is not available on Linux platforms. See "Buzzer on Touch" property in alternative.**	int
Buzzer Control	**0** = buzzer off **1** = buzzer on **2** = buzzer blink	int
Buzzer Off Time	Duration in milliseconds of off time when blink has been selected. Default = 1000. Range: 100–5000	int
Buzzer On Time	Duration in milliseconds of on time when blink has been selected. Default = 1000. Range: 100–5000	int

Communication Variables

Variable Name	Description	Data Type
Protocol Communication Status	Summarize the status of the communication protocols. **0** = No protocol running, protocol drivers might not have been properly downloaded to the HMI device **1** = Protocols loaded and started, no communication error **2** = At least one communication protocol is reporting an error	int read only
Protocol Error Message	Communication error with error source. For example: "[xxxx]" where "xxxx" is the protocol abbreviation, the error source.	string read only

Element	Description

Communication Variables

Variable Name	Description	Data Type
	Multiple acronyms appear in case of multiple error sources. Blank when no errors are reported.	
Protocol Error Count	Number of communication errors occurred since last reset. Reset value with Reset Protocol Error Count action, see "System actions" of main manual	int read only

Database Variables

Variable Name	Description	Data Type
Database link error message	Last detected error description	string read only
Database link status	**0** = Undefined (not yet initialized) **1** = OnLine (ready) **2** = OffLine (not available) **3** = Transfer in progress **4** = Error	int read only
Database link error count	Errors counter. Increased after each error	int read only

Each database variable is an array where index select the database link connection (Range 1-10)
Variables are updated only when any database connector action is executed

Daylight Saving Time Variables

Variable Name	Description	Data Type
Standard Offset	Offset in minutes when standard time is set, with respect to GMT (for example: -8x60 = -480 minutes)	int read only
Standard Week	Week in which the standard time starts (for example: First = 1)	int read only

Element	Description

Daylight Saving Time Variables

Variable Name	Description	Data Type
Standard Month	Month in which the standard time starts. Range: 0–11. (for example: November = 10)	int read only
Standard Day	Day of week in which the standard time starts (for example: Sunday = 0)	int read only
Standard Hour	Hour in which the standard time starts (for example: 02 = 2)	int read only
Standard Minute	Minute in which the standard time starts (for example: 00 = 0)	int read only
DST Offset	Offset in minutes when DLS time is set, with respect to GMT	int read only
DST Week	Week in which the DLS time starts	int read only
DST Month	Month in which the DLS time starts. Range: 0–11	int read only
DST Day	Day of week in which the DLS time starts	int read only
DST Hour	Hour in which the DLS time starts	int read only
DST Minute	Minute in which the DLS time starts	int read only

All variables are read only: they cannot be used to update the system clock.

Element	Description

Device Variables

Variable Name	Description	Data Type
Available System Memory	Free available RAM memory in bytes	uint64 read only
Backlight Time	Activation time in hours of the display backlight since production of the device	unsignedInt read only
Battery LED	Enables/disables the low battery LED indicator (when available) **0** = disabled **1** = enabled	int
Battery Timeout	Reserved	int
Display Brightness	Returns and adjusts brightness level. When set to a low light level (0..3), the backlight stays lit to a higher level for 8 seconds to allow the user to make the adjustments and then is switched-off. Even when set to 0, the backlight is still on and the Backlight Time counter increases. Range: 0–255	int
External Timeout	Non-operational time after which the display backlight is automatically turned off. The backlight is automatically turned on when the user touches the screen **-1** = switch off backlight and disable touch (switch display off). Backlight Time counter is stopped **0** = switch backlight on (switch display on) **1..n** = timeout for switch off backlight (screensaver timer)	int
Flash Free Space	Free space left in internal Flash memory	uint64 read only
Manufacturer Code	Code number that identifies the HMI	short read only
System RAM Usage	Current RAM memory used from HMI, expressed in byte	uint64 read only

Element	Description

Device Variables

Variable Name	Description	Data Type
System Font List	List of system fonts	string read only
System Mode	Runtime operation status **1** = booting **2** = configuration mode **3** = operating mode **4** = restart **5** = shutdown	int
System UpTime	Time the system has been powered since production of the unit (hours)	unsignedInt read only

Dump information Variables

Variable Name	Description	Data Type
Dump Archive Status	**1** = event buffer copy in progress	int read only
Dump Recipe Status	**1** = recipe buffer copy in progress If the copy duration time is less than 1 second, the system variable does not change its value	int read only
Dump Trend Status	**1** = trend buffer copy in progress	int read only
Dump Error Message	Return error message if any error occurs during the dump operation	string read only

Element	Description

Dump information Variables

Variable Name	Description	Data Type
Reset Recipe Status	**1** = recipe buffer reset in progress If the reset duration time is less than 1 second, the system variable does not change its value	int read only
Restore Recipe Status	Returns information during the copy process of recipes. If the copy duration time is less than 1 second, the system variable does not change its value. **0** = initial default state **1** = operation triggered **2** = operation complete successfully **3** = operation completed with errors	int read only

Network Variables

Variable Name	Description	Data Type
Gateway	Gateway address of the main Ethernet interface of HMI	string read only
IP Address	IP address of the main Ethernet interface of HMI	string read only
Mac ID	MAC ID of the main Ethernet interface of HMI	string read only
Network Adapter Parameters	JSON string that can be use to read or update the network adapters parameters	string
Network Status	Contains the result of the last operation required by writing inside the Adapter Parameters. It is updated after each write operation. • Empty string is meaning no errors • Last error descriptions	string read only
Subnet Mask	Subnet Mask of the main Ethernet interface of HMI	string read only

Element	Description

Screen Variables

Variable Name	Description	Data Type
X Screen resolution	Display horizontal screen size in pixel	int read only
Y Screen resolution	Display vertical screen size in pixel	int read only

SD Card Variables

Variable Name	Description	Data Type
SD Card FreeSpace	Available space on card in bytes	uint64 read only
SD Card Name	Name of SD card	string read only
SD Card Size	Size in bytes of the card plugged in the slot	uint64 read only
SD Card Status	**0** = SD card unplugged **1** = SD card plugged	int read only

Server Variables

Variable Name	Description	Data Type
Page name	Name of current page	string read only
Current project	Name of current project	string read only
Project load time	Date when the project was loaded on the HMI Runtime as in System Date format (milliseconds)	uint64 read only
Last operating mode start time	Seconds elapsed since device started operating mode	uint64 read only

Element	Description
	 All variables refer to server, not to HMI Client.

Time Variables

Variable Name	Description	Data Type
Day Of Month	Range: 1–31	int
Day of Week	Range: 0 = Sunday, .. , 6 = Saturday	int
Hour	Range: 0–23	int
Minute	Range: 0–59	int
Month	Range: 1–12	int
Second	Range: 0–59	int
System Time	The same as UTC time. It can also be set as date/time for this variable	unsignedInt
Year	Current Year	int

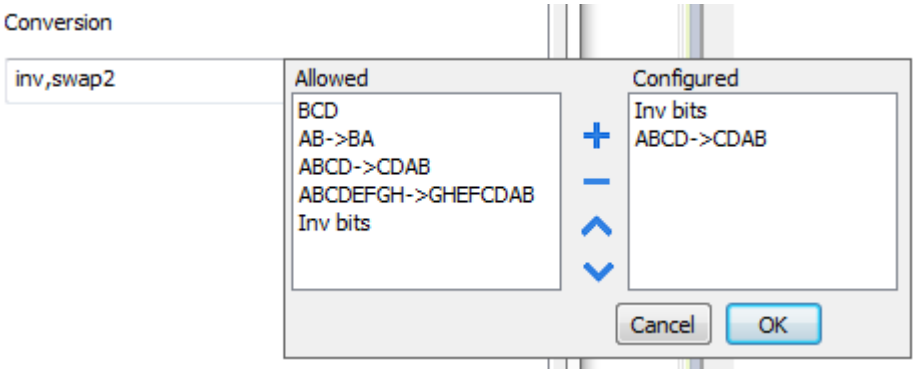
 System time expressed in UTC format

USB Drive Variables

Variable Name	Description	Data Type
USB Drive FreeSpace	Available space in bytes	uint64 read only
USB Drive Name	Name of USB device	string read only
USB Drive Size	Size in bytes of the device plugged in the USB port	uint64 read only
USB Drive Status	**0** = USB Drive unplugged **1** = USB Drive plugged	int read only

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th colspan="3">Version Variables</th></tr>
<tr><th>Variable Name</th><th>Description</th><th>Data Type</th></tr>
<tr><td>Main OS version</td><td>Version of Main OS</td><td>string
read only</td></tr>
<tr><td>Runtime version</td><td>Version of Runtime</td><td>string
read only</td></tr>
</table>
<table>
<tr><th colspan="3">Virtual Com Switch Variables</th></tr>
<tr><th>Variable Name</th><th>Description</th><th>Data Type</th></tr>
<tr><td>VCS status</td><td>Provides status of VCS service.
0 = Service enabled
1 = Client connected in interleaved mode
2 = Client connected in exclusive mode
3 = Service disabled (default)</td><td>unsignedByte
read only</td></tr>
<tr><td>VCS disable</td><td>Provides manual override of VCS service.
0 = VCS service enabled
1 = VCS service disabled (default)</td><td>boolean</td></tr>
<tr><td>VCS port</td><td>Provides current listening TCP port on HMI by VCS service</td><td>unsignedShort</td></tr>
</table>
</td></tr>
<tr><td>Data Type</td><td>Each system variable has a specific data type, described in above tables.
The following table shows the details of any data type used for system variables.
<table>
<tr><th>Data Type</th><th>Memory Space</th><th>Limits</th></tr>
<tr><td>short</td><td>16-bit data</td><td>-32768 ... 32767</td></tr>
<tr><td>int</td><td>32-bit data</td><td>-2.1e9 ... 2.1e9</td></tr>
<tr><td>unsignedByte</td><td>8-bit data</td><td>0 ... 255</td></tr>
<tr><td>unsignedShort</td><td>16-bit data</td><td>0 ... 65535</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Data Type</th><th>Memory Space</th><th>Limits</th></tr>
<tr><td>unsignedInt</td><td>32-bit data</td><td>0 ... 4.2e9</td></tr>
<tr><td>uint64</td><td>64-bit data</td><td>0 ... 1.8e19</td></tr>
<tr><td>string</td><td colspan="2">Array of elements containing character code defined by selected encoding</td></tr>
</table>
</td></tr>
<tr><td>Arraysize</td><td>In case of string Tag, this property represents the maximum number of bytes available in the string Tag.

Note: number of bytes corresponds to number of string chars if Encoding property is set to UTF-8 or Latin1 in Tag Editor.
If Encoding property is set to UCS-2BE, UCS-2LE, UTF-16BE or UTF-16LE one char requires 2 bytes.</td></tr>
<tr><td>Conversion</td><td>Conversion to be applied to the Tag.
 

Depending on data type selected, the Allowed list shows one or more conversions, listed below.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>Inv bits</td><td>Invert all the bits of the tag.

Example:
1001 → 0110 (in binary format)
9 → 6 (in decimal format)</td></tr>
<tr><td>Negate</td><td>Set the opposite of the tag value.

Example:
25.36 → -25.36</td></tr>
<tr><td>AB -> BA</td><td>Swap nibbles of a byte.

Example:
15D4 → 514D (in hexadecimal format)
5588 → 20813 (in decimal format)</td></tr>
<tr><td>ABCD -> CDAB</td><td>Swap bytes of a word.</td></tr>
</table>
</td></tr>
</table>

Element	Description

Value	Description
	Example: 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)
ABCDEFGH -> GHEFCDAB	Swap bytes of a double word. *Example:* 32FCFF54 → 54FFFC32 (in hexadecimal format) 855441236 → 1426062386 (in decimal format)
ABC...NOP -> OPM...DAB	Swap bytes of a long word. Example: 142.366 → -893553517.588905 (in decimal format) 0 10000000110 00011100101110110110010001011010000111001010110000001 → 1 10000011100 10101010000101000101101101101100101101100000100111101 (in binary format)
BCD	Separate the byte in two nibbles, and reads them as decimal (from 0 to 9) *Example:* 23 → 17 (in decimal format) 0001 0111 = 23 0001 = 1 (first nibble) 0111 = 7 (second nibble)

Select the conversion and click on plus button. The selected item will be added on **Configured** list.

If more conversions are configured, they will be applied in order (from top to bottom of **Configured** list).

Use the arrow buttons to order the configured conversions.

System Variables - Retentive Memory

System Variables - Retentive Memory protocol allows to create Tags that point to a memory area whose content is maintained when HMI is powered off.

The physical support for retentive memory is based on FRAM technology.

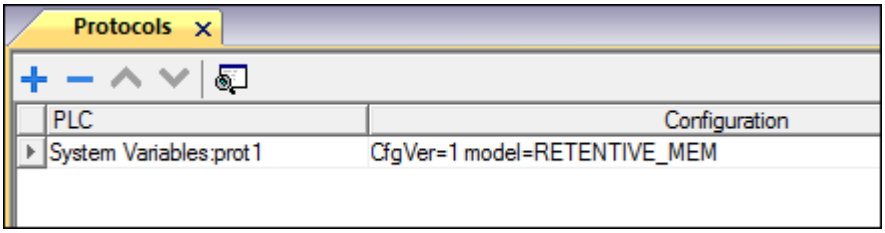
Important: Not all HMI devices include FRAM memory. If FRAM memory is not available, persistency is supported using user memory storage (Flash or hard disk drive). Flash technology has a limitation in the maximum number of write operations. The use of Flash as storage media for retentive memory with frequent write operations may damage the memory components. Check HMI device data for availability of FRAM memory.

Important: Retentive memory is 16 KB flat memory area organized in bytes and accessible through an offset.
Refer to schema below.

Retentive memory vs. recipes storage
Recipe data is saved in flash memory (except forBTM-PCRUNTIME) while retentive data is saved in a FRAM. Flash memory is not suitable for a high number of write operations, while FRAM supports a virtually unlimited number of write operations and should be preferred when frequent write operations are required.

Protocol Editor Settings

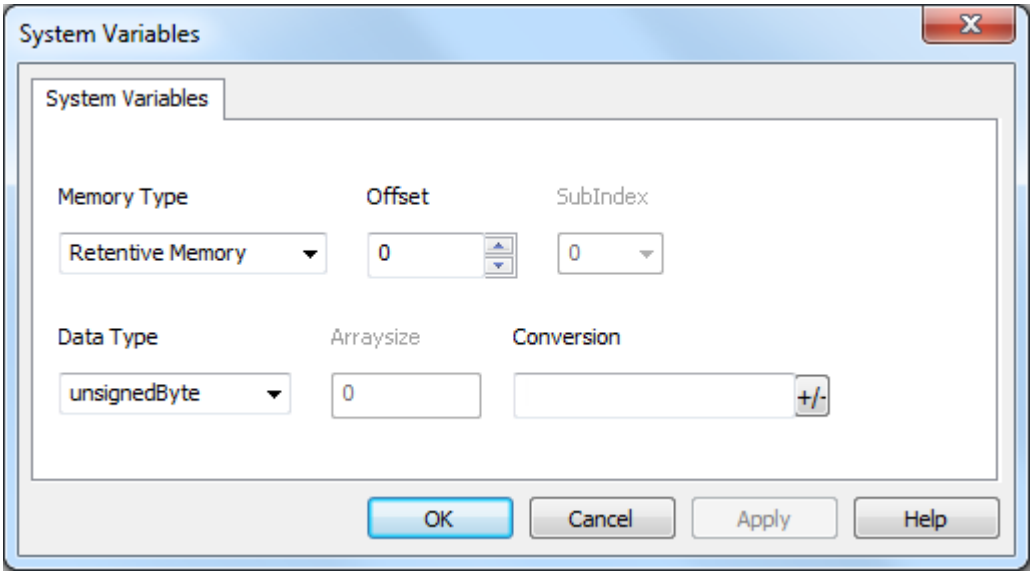
From PLC Model list of Protocol Editor dialog, select Retentive Memory.



Tag Editor Settings

Path: ***ProjectView*** *>* ***Config*** *> double-click* ***Tags***

1. To add a tag, click **+**: a new line is added.
2. Select **System Variables** from the **Driver** list: tag definition dialog is displayed.



Element	Description
Memory Type	Fixed to Retentive Memory
Offset	Offset address where tag is located. Range: 0-16383
SubIndex	This parameter allows resource offset selection based on selected Data Type

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td>Data Type</td><td>
<table>
<tr><th>Data Type</th><th>Memory Space</th><th>Limits</th></tr>
<tr><td>boolean</td><td>1-bit data</td><td>0 ... 1</td></tr>
<tr><td>byte</td><td>8-bit data</td><td>-128 ... 127</td></tr>
<tr><td>short</td><td>16-bit data</td><td>-32768 ... 32767</td></tr>
<tr><td>int</td><td>32-bit data</td><td>-2.1e9 ... 2.1e9</td></tr>
<tr><td>int64</td><td>64-bit data</td><td>-9.2e18 ... 9.2e18</td></tr>
<tr><td>unsignedByte</td><td>8-bit data</td><td>0 ... 255</td></tr>
<tr><td>unsignedShort</td><td>16-bit data</td><td>0 ... 65535</td></tr>
<tr><td>unsignedInt</td><td>32-bit data</td><td>0 ... 4.2e9</td></tr>
<tr><td>uint64</td><td>64-bit data</td><td>0 ... 1.8e19</td></tr>
<tr><td>float</td><td>IEEE single-precision 32-bit floating point type</td><td>1.17e-38 ... 3.4e38</td></tr>
<tr><td>double</td><td>IEEE double-precision 64-bit floating point type</td><td>2.2e-308 ... 1.79e308</td></tr>
<tr><td>string</td><td colspan="2">Array of elements containing character code defined by selected encoding</td></tr>
<tr><td>binary</td><td colspan="2">Arbitrary binary data</td></tr>
</table>

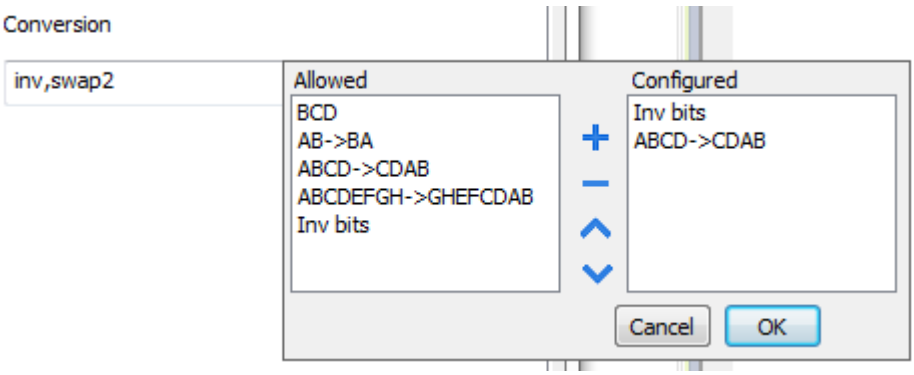
Note: to define arrays. select one of Data Type format followed by square brackets like “byte[]”, “short[]”…
</td></tr>
<tr><td>Arraysize</td><td>

In case of array Tag, this property represents the number of array elements.
In case of string Tag, this property represents the maximum number of bytes available in the string Tag.

Note: number of bytes corresponds to number of string chars if Encoding property is set to UTF-8 or Latin1 in Tag Editor.
If Encoding property is set to UCS-2BE, UCS-2LE, UTF-16BE or UTF-16LE one char requires 2 bytes.
</td></tr>
<tr><td>Conversion</td><td>Conversion to be applied to the Tag.</td></tr>
</table>

Element	Description

Conversion



Depending on data type selected, the **Allowed** list shows one or more conversions, listed below.

Value	Description
Inv bits	Invert all the bits of the tag. *Example:* 1001 → 0110 (in binary format) 9 → 6 (in decimal format)
Negate	Set the opposite of the tag value. *Example:* 25.36 → -25.36
AB -> BA	Swap nibbles of a byte. *Example:* 15D4 → 514D (in hexadecimal format) 5588 → 20813 (in decimal format)
ABCD -> CDAB	Swap bytes of a word. *Example:* 9ACC → CC9A (in hexadecimal format) 39628 → 52378 (in decimal format)
ABCDEFGH -> GHEFCDAB	Swap bytes of a double word. *Example:* 32FCFF54 → 54FFFC32 (in hexadecimal format) 855441236 → 1426062386 (in decimal format)
ABC...NOP -> OPM...DAB	Swap bytes of a long word. Example: 142.366 → -893553517.588905 (in decimal format) 0 10000000110 00011100101110110110010001011010000111001010110000001

<table>
<tr><th>Element</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td></td><td>→
1 10000011100
10101010000101000101101101101100101101100000100111101
(in binary format)</td></tr>
<tr><td>BCD</td><td>Separate the byte in two nibbles, and reads them as decimal (from 0 to 9)

Example:
23 → 17 (in decimal format)
0001 0111 = 23
0001 = 1 (first nibble)
0111 = 7 (second nibble)</td></tr>
</table>
Select the conversion and click on plus button. The selected item will be added on Configured list.
If more conversions are configured, they will be applied in order (from top to bottom of Configured list).
Use the arrow buttons to order the configured conversions.
</td></tr>
</table>

.

Cleaning Retentive Memory

Use the **ClearRetentiveMemory** action to clear the content of the retentive memory.

Tip: Use this action to set the memory content to a known status at any time.

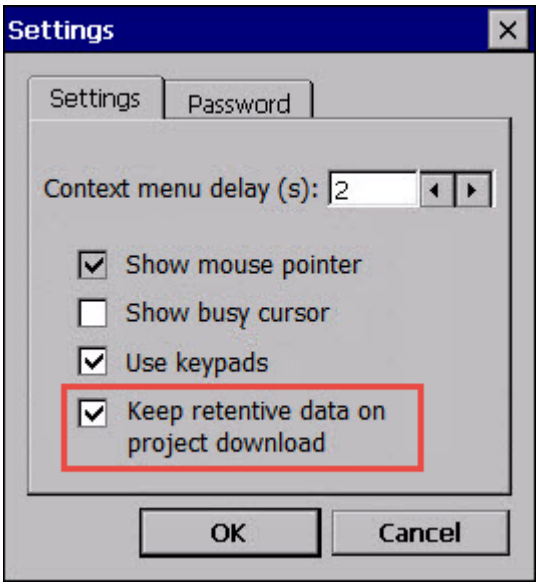
See *Actions* > *Tag Action*s section of main manual for more details.

JavaScript interface for this action is:
project.clearRetentiveMemory();

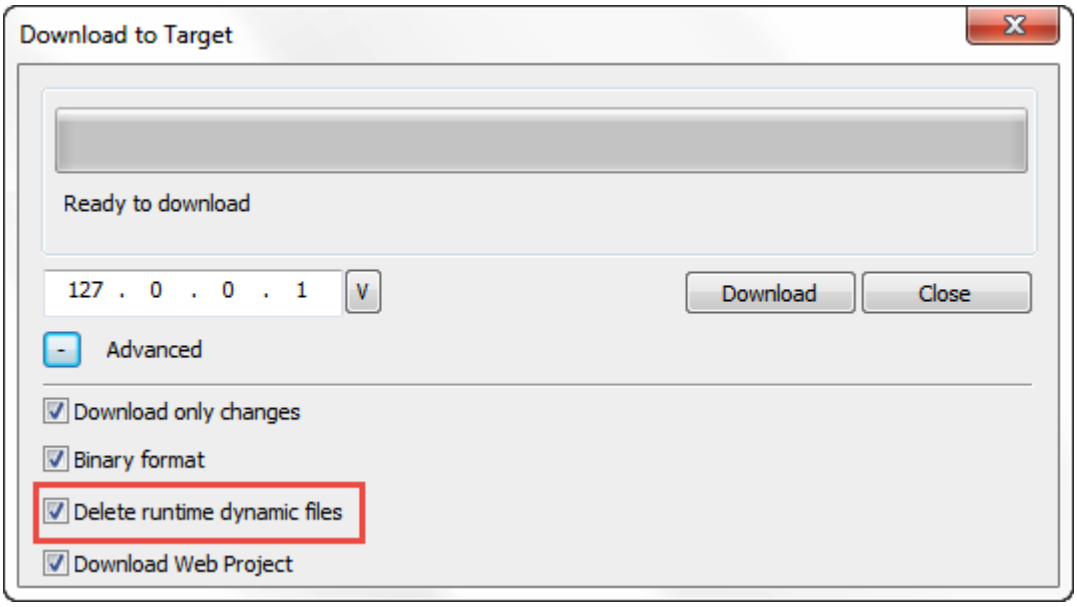
Preserving Retentive Memory at project download

When a project file is downloaded to an HMI, or when the active project is modified, the content of retentive memory is usually deleted.

If is needed to preserve the content of retentive data at project download or update, select the **Keep retentive data on project download** option in the settings tabs of the HMI device.

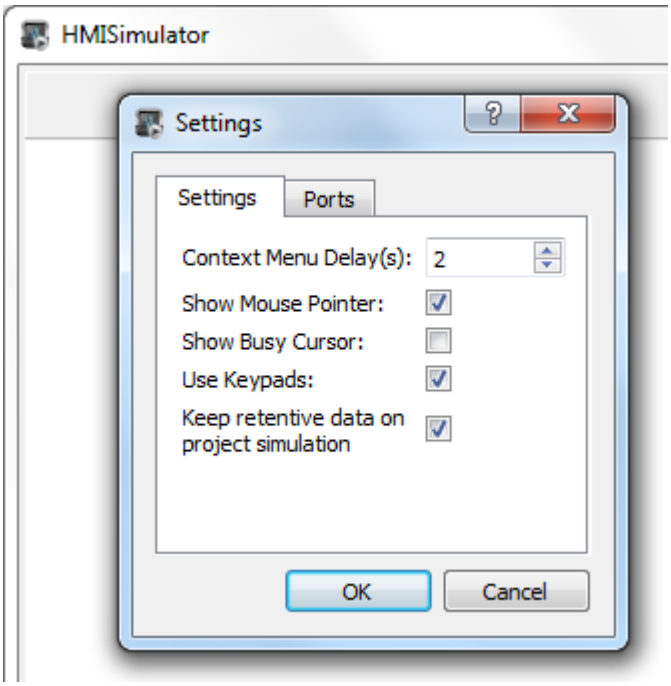


This setting will be ignored if **Delete runtime dynamic files** option is selected from *Download to Target* window.



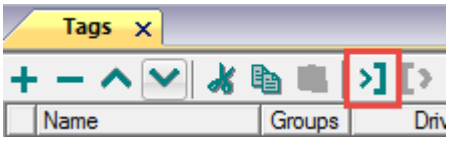
Preserving Retentive Memory in Simulator

Simulator of Wizard supports the retentive memory. To enable retentive memory during project simulation use the option "Keep retentive data on project simulation" in context menu.

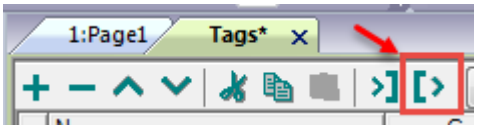


Tag Import

Select the driver in Tag Editor and click on the **Import Tags** button to start the importer.

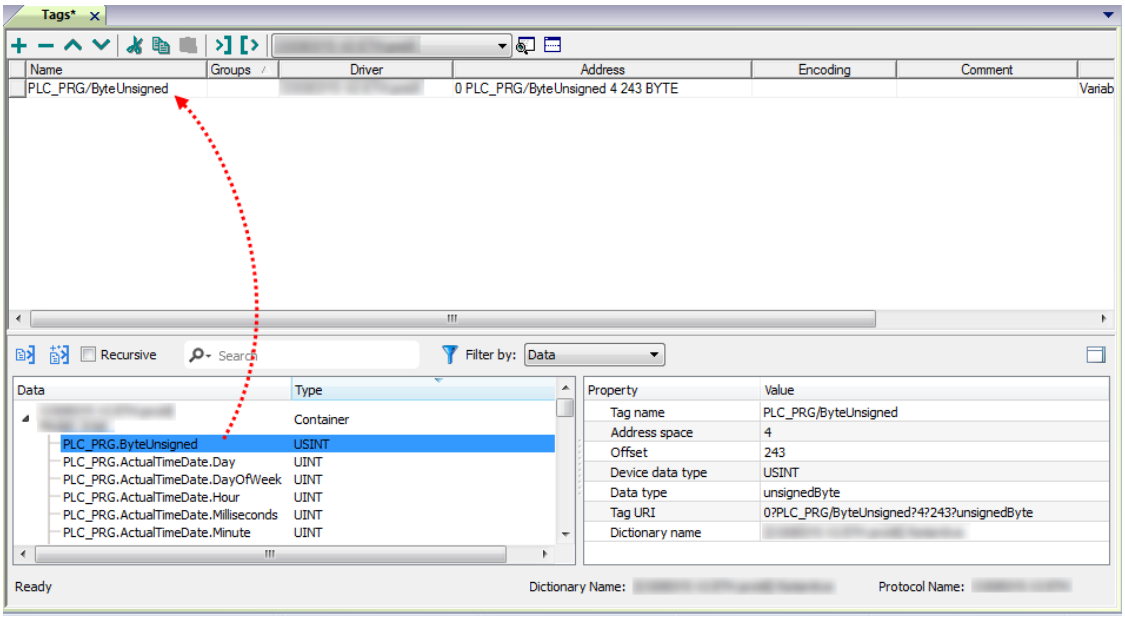


The system will require a generic XML file exported from Tag Editor by appropriate button.

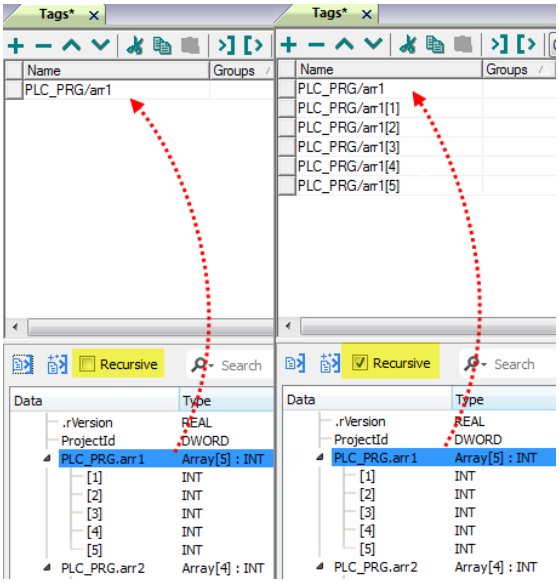


Once the importer has been selected, locate the symbol file and click **Open**.

Tags included in the symbol file are listed in the tag dictionary. The tag dictionary is displayed at the bottom of the screen.



Toolbar item	Description
	**Import Tag(s)**. Select tags to be imported and click on this icon to add tags from tag dictionary to the project
	**Update Tag(s)**. Click on this icon to update the tags in the project, due a new dictionary import.

Toolbar item	Description
Recursive	Check this box to import all sub-elements of a tag. Example of both checked and unchecked result: 
Search Filter by: Tag name	Searches tags in the dictionary basing on filter combo-box item selected.

12 Actions

Actions are functions used to interact with the system and are normally executed when events are triggered.

Events can be triggered by various widgets, for example on press and on release of a button. Not all actions are available for all the events of an object.

Actions are linked to widgets in the **Event** section of the Property pane (Page Editor).

Alarm actions

Used to acknowledge or reset alarms.

SelectAllAlarms

Selects all alarms in the alarm widget.

AckAlarm

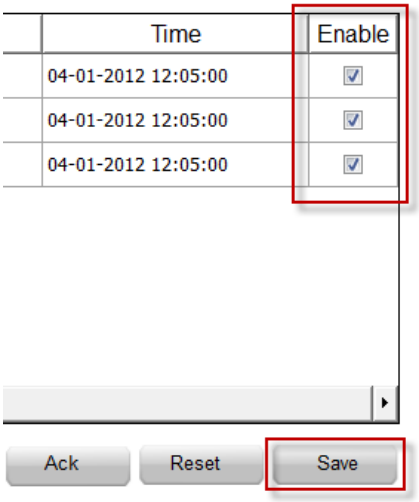
Acknowledges selected alarms.

ResetAlarm

Resets selected acknowledged alarms.

EnableAlarms

Saves changes made in the **Enable** column in the alarm widget. This action is used with the **Save** button in the alarm widget.



Event actions

Used by Alarm History widget to scroll events/alarms backward/forward in table view (event buffer widget).

ScrollEventsBackward

Scrolls events/alarms backward in table view (event buffer widget).

ScrollEventsForward

Scrolls events/alarms forward in table view (event buffer widget).

MultiLanguage actions

Selects the application language.

SetLanguage

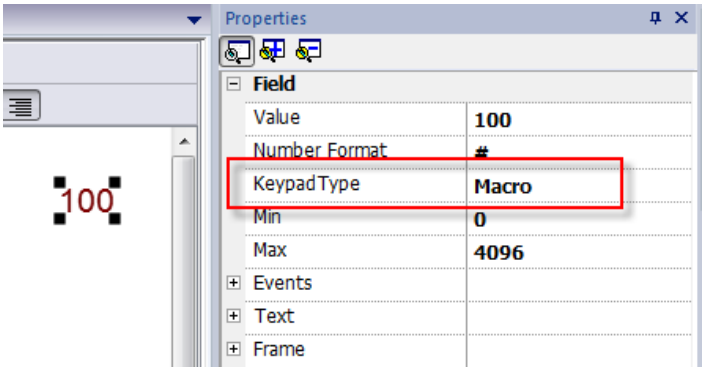
Sets the language used. The selected language will be applied at run time to all applicable widgets.

Keyboard actions

Changes the use of keypads.

SendKey

Sends one character to a numeric widget. The **KeypadType** property of the numeric widget must be set as **Macro**.

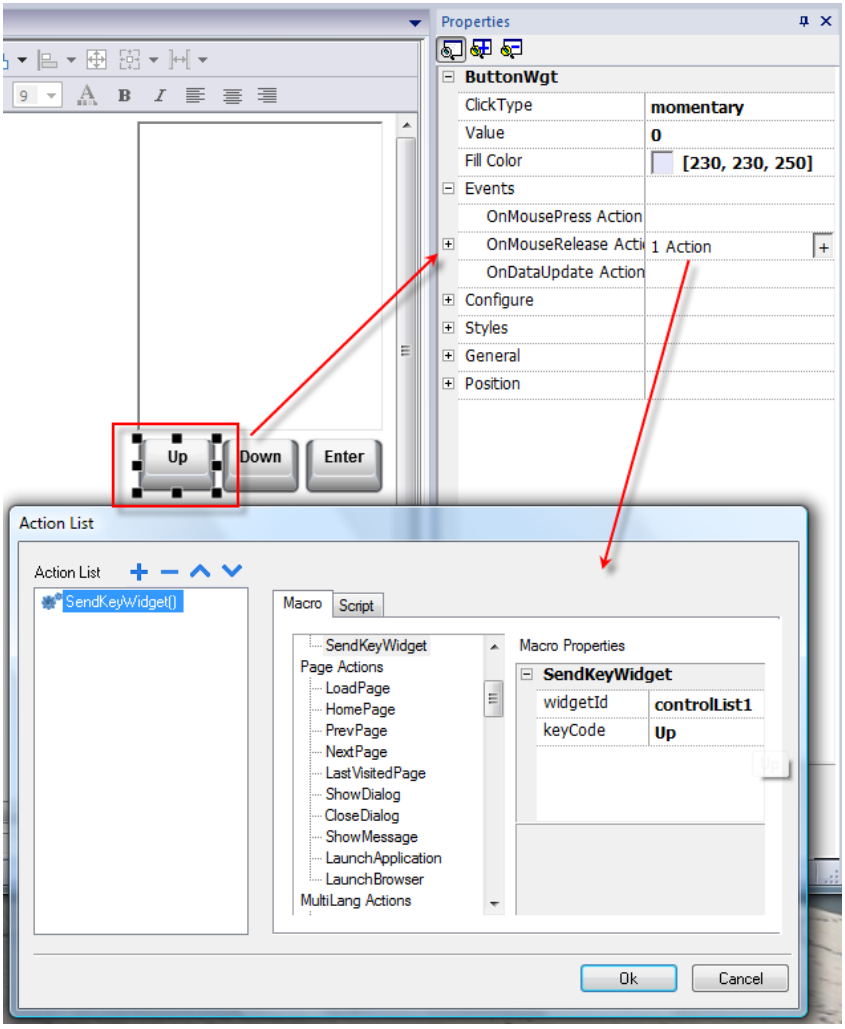


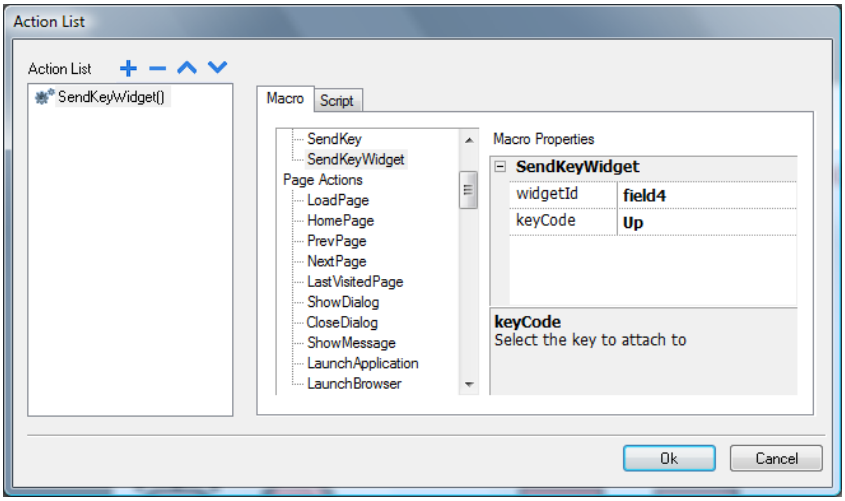
SendKeyWidget

Sends one character to a specific widget.

Example

The **Up** and **Down** buttons use the **SendKeyWidget** action in association with the **Control List Widget**.





ShowKeyPad

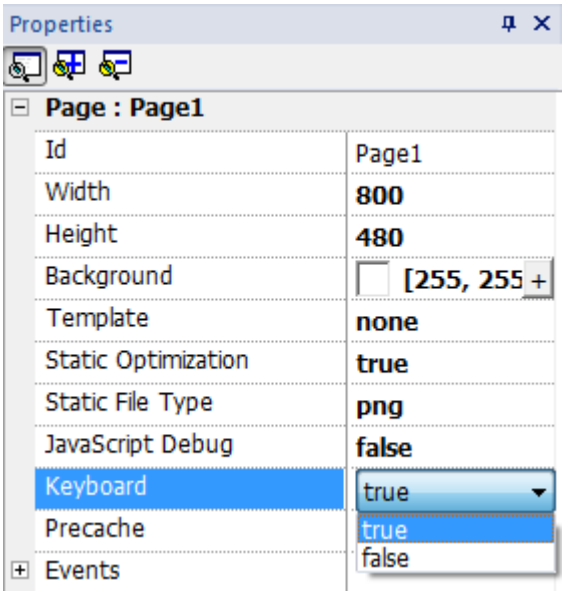
Shows the default operating system touch keypad.

Note: might not be supported by all operating systems.

Keyboard

Enables/disables the use of actions when using external keyboards. Action execution can be enabled/disabled both at project and at page level.

The effect is equivalent to the use of the property Keyboard for project and page.



FTP actions

Used to upload and download files to and from a remote FTP server.

ftpGET

Download files from a remote FTP server

Parameter	Description
FtpConfig	Configuring the FTP parameters
FtpRemoteFileName	File name on the remote FTP server to download (source)
FtpLocalFileName	File name on local HMI device (destination)

ftpPUT

Upload files to a remote FTP server

Parameter	Description
FtpConfig	Configuring the FTP parameters
FtpLocalFileName	File name on local HMI device (source)
FtpRemoteFileName	File name on the remote FTP server to download (Destination)

Filenames can contain wildcards.

When transferred, system variables are updated with the status of ongoing operations (see "FTP client variables" on page 85for details).

FTP Server Configuration

To configure the FTP parameter, enter the following information for the **FtpConfig** setting:

Parameter	Description
FTP Address	FTP server IP Address
Server Port	Port for FTP connection (default = 21).
Authentication	Select the FTP authentication to use: • Normal (Username and password required) • Anonymous
User Name	Username of the remote FTP account
Password	Password of the remote FTP account

Click **+** to add more FTP servers configuration.

Tip: Use tags if you want change the server parameters dynamically from the HMI Runtime.

FTP JavaScript Interface

ftpConfig

```
ftpCONFIG (IPAddress, Port, Authentication, UserName, Password)
```

Set the FTP parameters to use on next FTP calls

Parameter	Description
IPAddress	FTP server IP Address.
Port	Port for FTP connection (default = 21).
Authentication	Select the FTP authentication to use: • Normal (Username and password required) • Anonymous
UserName	Username of the remote FTP account
Password	Password of the remote FTP account

ftpGET

```
ftpGET (remoteFileName, localFileName, [callback])
```

Download files from a remote FTP server

Parameter	Description
remoteFileName	File name on the remote FTP server to download (source)
localFileName	File name on local HMI device (destination)
callback	Function that will be call at the end of the FTP transfer

ftpPUT

```
ftpPUT (remoteFileName, localFileName, [callback])
```

Upload files to a remote FTP server

Parameter	Description
remoteFileName	File name on the remote FTP server to download (source)
localFileName	File name on local HMI device (destination)
callback	Function that will be call at the end of the FTP transfer

Example:

```
project.ftpCONFIG("192.168.0.200", "21", "true", "admin", "admin");

project.ftpGET( "data.txt",
                "\\USBMemory\\data.txt",
                function(ftpStatus) {fnFtpGetFinished(ftpStatus);} );

function fnFtpGetFinished(ftpStatus) {
        alert(ftpStatus);
}
```

Page actions

Page navigation. Page actions can be used with the following events:

- OnMouseClick,
- OnMouseRelease,
- OnMouseHold
- OnActivate
- OnDeactivate
- Alarms
- Schedulers.

LoadPage

Go to the selected page of the project.

HomePage

Go to the home page.

You can set the home page in the **Behavior** section of the **Project Widget**, see "Behavior" on page 60

PrevPage

Go to the previous page.

NextPage

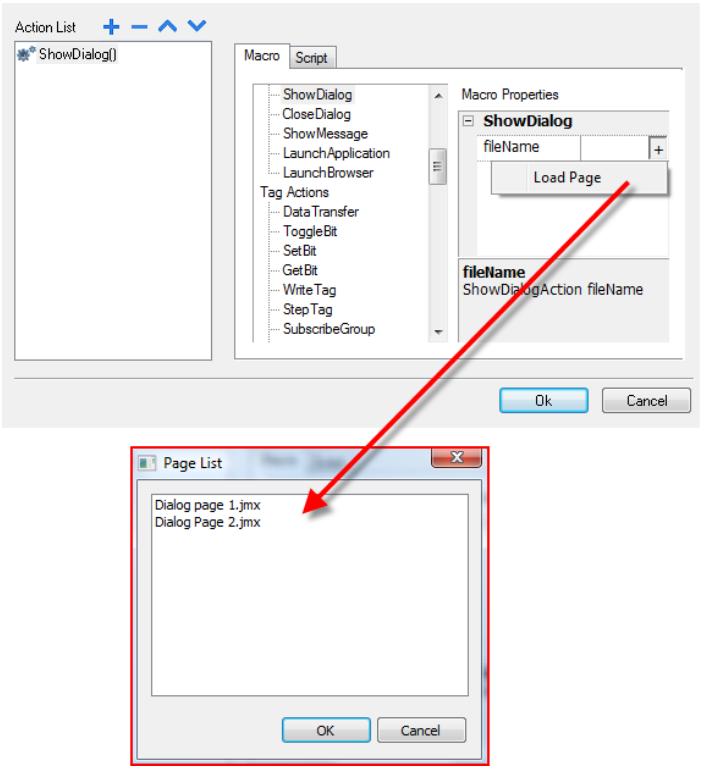
Go to the next page.

LastVisitedPage

Go to the previously displayed page

ShowDialog

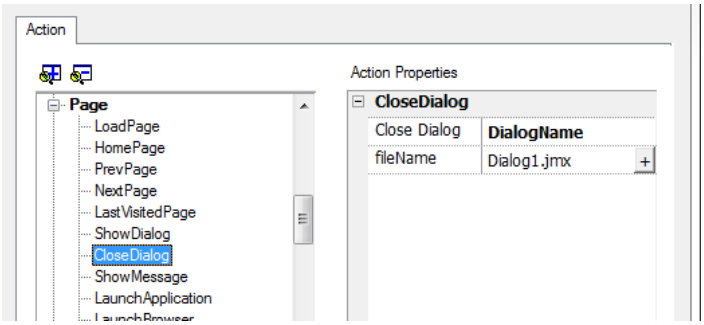
Opens a dialog page defined in the project.



CloseDialog

Close dialog pages.

Note: This action is applicable only to dialog pages.



CloseDialog options

Option	Description
All	Closes all open dialogs
Selected	Closes only active dialog
DialogName	Closes dialog specified as **fileName** property

JavaScript Interface

project.closeDialog(DialogID);

Where *DialogID*:

All	Closes all open dialogs
Selected	Closes only active dialog
DialogName.jmx	Closes dialog specified as **fileName** parameter

Examples

Example	Behavior
project.closeDialog("All");	All open dialogs are closed
project.closeDialog("Selected");	The selected dialog is closed
project.closeDialog("Dialog1.jmx");	All instances of Dialog1 are closed

The function *project.closeDialog();* without parameter works as *project.closeDialog("Selected");*.

ShowMessage

Displays a popup message. Enter the text of the message to be displayed.

LaunchApplication

Launches an external application.

Parameter	Description
App Name	Executable name with extension (for example, "notepad.exe" to run Notepad)
Path	Application path.
Arguments	Application specific arguments (for example, *\flash\qthmi\Manual.pdf* to open the document “Manual.pdf”)
Single Instance	Argument to start the application in a single instance or multiple instances. When single instance is selected, the system first verifies whether the application is already running; if so, then the application is brought to the foreground, if not, then the application is launched.
FlushRuntimeCache	Flush all runtimes cache to free as more ram as possible before running the application.

Note: Arguments with spaces must be quoted (for example, "\Storage Card\Manual.pdf")

Example:

LaunchApplication	
Application Name	\Windows\cmd.exe
Executable path	
arguments	/c "\Flash\New Folder\test.bat" Par1 Par2
Single Instance	true

LaunchBrowser

Opens the default web browser. You can define URL address as argument.

Note: Only works on platforms having a native web browser (for example, on Windows CE PRO with Internet Explorer enabled).

LaunchVNC

Starts VNC server and opens the configuration.

Macro available only for HMI devices based on Windows CE platform.
On HMI devices based on Linux platform the VNC service can be enabled from the "*Service*" tab of the "System Settings on Linux Devices" on page 381

See "Software plug-in modules" on page 59 to include it on Windows CE devices.

LaunchPDFViewer

Starts PDF Viewer.

Note: Only works on devices that include PDF Viewer.
See "Software plug-in modules" on page 59 to include it on Windows CE devices.

LaunchUpdater

Updates project and runtime from an external device.

Use **Path** parameter to specify the folder that will contain the update package file. Leave the path parameter empty if you prefer select the file manually on the HMI device when the macro is invoked.

When the LaunchUpdater macro is executed, the below dialog is showed on HMI device

Note: Not supported in devices based on Win32.

JavaScript Interface

project.launchUpdater(strPath)

Examples

```
project.launchUpdater("\\USBMemory")
```

LockScreen

Temporarily locks the touch screen. Allows cleaning the touch screen.

The system variable **Time remaining to unlock** displays the time remaining to unlock.See "Screen variables" on page 88

Print actions

Manages print tasks.

PrintGraphicReport

Prints a graphic report.

Parameter	Description
reportName	Assigns a name to the report
silent	**false** = allows to set printer properties at run time

PrintText

Prints a string.

Parameter	Description
text	String to be printed
silent	**false** = allows to set printer properties at run time

This action works in line printing mode and uses a standard protocol common to all printers that support it. Text is printed immediately line by line or after a timeout custom for each printer model.

Note: printing could a few minutes for models not designed for line printing.

Not available on Linux platforms (see "HMI devices capabilities" on page 429 for panels details)

PrintBytes

Prints an hexadecimal string representing data to print (for example, "1b30" to print < ESC 0 >.

Parameter	Description
bytes	Exadecimal string to print
silent	**false** = allows to set printer properties at run time

This action works in line printing mode and uses a standard protocol common to all printers that support it. Text is printed immediately line by line or after a timeout custom for each printer model.

Note: printing could a few minutes for models not designed for line printing.

Not available on Linux platforms (see "HMI devices capabilities" on page 429 for panels details)

EmptyPrintQueue

Flushes the current printing queue. If executed while executing a job, the queue is cleared at the end of the job.

PausePrinting

Puts the current printing queue on hold. If executed while executing a job, the queue is paused at the end of the job.

ResumePrinting

Restarts a queue previously put on hold.

AbortPrinting

Stop the execution of the current job and removes it from the queue. If the queue has another job, then, after aborting, the next job starts.

Recipe actions

Used to program recipe management.

DownLoadRecipe

Copy recipe data from HMI device flash memory to the controller (e.g. PLC, local variable, depending on the protocol).

Parameter	Description
RecipeName	Name of recipe to download
RecipeSet	Number of recipe set to copy. **curSet** = download currently selected recipe set

UpLoadRecipe

Saves recipe data from the controller (e.g. PLC, local variable, depending on the protocol) to the device Flash Memory.

Parameter	Description
RecipeName	Name of recipe to upload
RecipeSet	Number of recipe set to copy. **curSet** = upload currently selected recipe set

WriteCurrentRecipeSet

Sets the selected recipe as current recipe set.

Parameter	Description
RecipeName	Name of recipe to set as current recipe
RecipeSet	Recipe set to define as current recipe set

DownLoadCurRecipe

Downloads current set of recipe data to the controller.

No parameter is required.

UploadCurRecipe

Uploads set of controller data to current recipe set.

No parameter is required

ResetRecipe

Restores factory settings for recipe data. Original recipe data will overwrite uploaded recipes

Select the recipe that you want to reset to factory data.

DumpRecipeData

Dumps recipe data to internal or external storage. Data is saved in .csv format.

Parameter	Description
RecipeName	Name of recipe to dump
FilePath	Destination folder • Internal = *\Flash\QTHMI\workspace\Dump* • USB drive = *\USBMemory* • SD Card = *\Storage Card* • Public Network = *\\<hostname or IP>\sharePath*

Parameter	Description
	• Private Network = \\<*username*>:<*password*>@<*hostname or IP*>*sharePath*  Note: supported formats for external memory are FAT or FAT32 (NTFS format is not supported).  Note: Private networks are supported only from Linux devices with BSP 1.0.25 and above.
FileName	Tag that specifies a filename.
DateTimePrefixFileName	**true** = the dumped file will have date and time as prefix to its name (for example D2012_01_01_T10_10_recipe1.csv)
TimeSpec	Time format: • **Local** = the time values exported are the time of the HMI device. • **Global** = the time values exported are in UTC format.

RestoreRecipeData

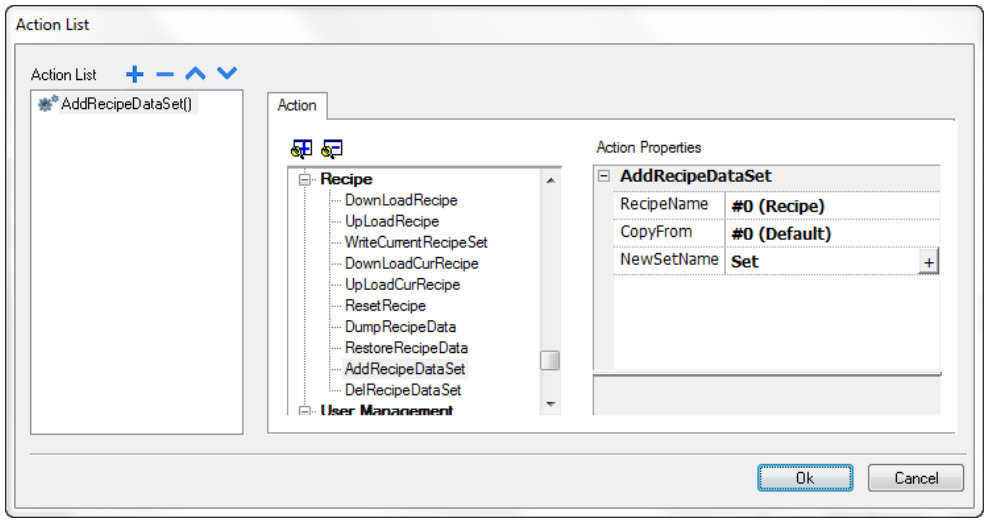
Restores previously saved recipe data.

Parameter	Description
RecipeName	Recipes to restore: • AllRecipes Data of all recipes will replaced with the data read from the external file • CurrentRecipe Only the data of the current selected recipe will replaced with the data read from the external file
RecipeDataSet	Available only when RecipeName=CurrentRecipe. Select the data sets to restore: • AllRecipeDataSet All data set will restored • curSet Only the data set of the current selected data set will restore
Restore Type	Available only when RecipeDataSet=AllRecipeDataSet. This parameter define the behavior when the numbers of data sets inside the file to restore is not matching with the data set number inside the HMI device • Replace All data sets that are inside the device are removed and replaced with the data sets from the csv file • Match Replace only the data set inside the device that have the same data set id

Parameter	Description
	• MatchAndAdd Replace the data set inside the device that have the same data set id and add the additional data set found inside the csv file (Note: data sets that are inside the device but not inside the csv file are not removed from the device)
FilePath	Source folder • Internal = *\Flash\QTHMI\workspace\Dump* • USB drive = *\USBMemory* • SD Card = *\Storage Card* • Public Network = *\\<hostname or IP>\sharePath* • Private Network = *\\<username>:<password>@<hostname or IP>\sharePath* Note: supported formats for external memory are FAT or FAT32 (NTFS format is not supported). Note: Private networks are supported only from Linux devices with BSP 1.0.25 and above.
FileName	Attached tag from which read the file name at run time.
BrowseForFile	**true** = shows the Open dialog to browse the file to read. **false** = no dialog is shown,

AddRecipeDataSet

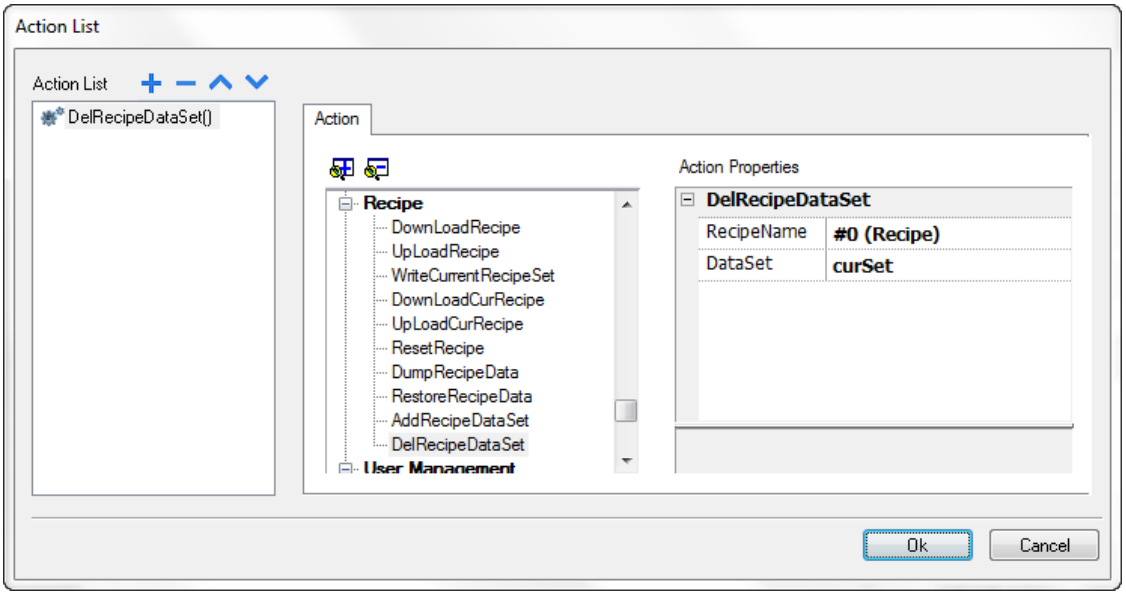
Adds a new dataset to the selected recipe. The new dataset is appended at the end of the already defined datasets.



Parameter	Description
RecipeName	Recipe where the dataset is added.
CopyFrom	Dataset from where parameters values are copied from to initialize the new dataset
NewSetName	Name of new dataset. Here you can you can use a tag reference.

DelRecipeDataSet

Deletes a dataset from the selected recipe. Deleting a dataset will rearrange the position number of the datasets that follow.



Parameter	Description
RecipeName	Recipe where the dataset is to be deleted.
DataSet	Dataset to be deleted.

Remote Client actions

Used to upload and download files to and from a remote HMI device. These actions can only be used from a remote HMI Client to access remote files via FTP.

Important: Enable FTP support and give all necessary user rights to the folders used to transfer files.

UploadToHMI

Opens a file Open dialog to select a file to be uploaded to the remote HMI device.

Parameter	Description
Destination	Destination path on HMI device for file upload
Filter	File extensions of the files to be displayed separated by commas (for example, *.txt)

DownloadFromHMI

Opens a file Open dialog to select a file to be downloaded from the remote HMI device.

Note: Only files matching the set filter are displayed and can be downloaded.

Parameter	Description
Source	Source path on the HMI device for file download
Filter	File extensions of the files to be displayed separated by commas (for example, *.txt)

JavaScript Interface

```
boolean project.uploadToHMI(dirPath, strFilter);
```

```
boolean project.downloadFromHMI(dirPath, strFilter);
```

Parameter	Description
dirPath	Source path on the HMI device for file download/upload
strFilter	File extensions of the files to be displayed separated by commas (for example, *.txt)

Return values:

True	Transfer successful
False	Transfer failed

Note: When transferred, system variables are updated with the status of ongoing operations.

System actions

Used to manage system properties.

Restart

Restarts the runtime.

DumpTrend

Stores historical trend data to external drives (USB drive or SD card).

Parameter	Description
TrendName	Name of historical trend to store
FolderPath	Destination folder: • Internal = *\Flash\QTHMI\workspace\Dump* • USB drive = *\USBMemory* • SD Card = *\Storage Card* • Public Network = *\\<hostname or IP>\sharePath* • Private Network = *\\<username>:<password>@<hostname or IP>\sharePath* Note: supported formats for external memory are FAT or FAT32 (NTFS format is not supported). Note: Private networks are supported only from Linux devices with BSP 1.0.25 and above.
FileFormat	**Binary** = the buffer is dumped in binary format (a .dat file and .inf file). Both these files are then required to convert data in .csv format by an external utility. **Compatibility CSV** = the buffer is dumped to the specified location as a .csv file format compatible with versions 1.xx **Compact CSV** = the buffer is dumped to the specified location as a .csv file using a newer format See "Exporting trend buffer data" on page 189
DateTimePrefixFileName	**true** = the dumped file will have date and time as prefix to its name (for example D2012_01_01_T10_10_Trend1.csv)

Parameter	Description
timeSpec	Time format: • **Local** = the time values exported are the time of the HMI device. • **Global** = the time values exported are in UTC format.
FileName	Enabled when the DateTimePrefixFileName=true The below wildcards are supported • %n = Trend name • %y = Year • %M = Month • %d = Day • %h = Hour • %m = Minutes • %s = Seconds Example: \%n\%y%M%d\%h%m%s

Note: execution of the DumpTrend action will automatically force a flush to disk of the data temporarily maintained in the RAM memory. See "History trends" on page 192 for details on how to save sampled data to disk.

Note: external drives connected to USB port must have format FAT or FAT32. NTFS format is not supported.

WARNING: Be aware there are limits in the max number of files that can create inside a folder. Limits are depending of different factors and are not simple to calculate, you can think as 999 the max number of files that can be use inside a folder.

To convert binary dump files to .csv

The TrendBufferReader.exe tool is stored in the *Utils* folder of the Wizard installation folder.

Use the following syntax:

```
TrendBufferReader -r Trend1 Trend1.csv 1
```

where:

`Trend1` = name of the trend buffer without extension resulting from the dump (original file name is trend1.dat)

`Trend1.csv` = name for the output file.

.csv file structure

The resulting .csv file has five columns

Column	Description
Data Type	Data type of sampled tag: 0 = empty 1 = boolean 2 = byte 3 = short 4 = int 5 = unsignedByte 6 = unsignedShort 7 = unsignedInt 8 = float 9 = double
Value	Value of the sample
Timestamp (UTC)	Timestamp in UTC format
Sampling Time(ms)	Sampling interval time in milliseconds
Quality	Tag value quality. Information coded according the OPC DA standard and stored in a byte data (8 bits) defined in the form of three bit fields; Quality, Sub status and Limit status. The eight quality bits are arranged as follows: QQSSSSLL. For a complete and detailed description of all the single fields, please refer to the OPC DA official documentation.

Commonly quality values

The most commonly used quality values returned by the HMI acquisition engine are:

Quality Code	Quality	Description
0	BAD	The value is bad but no specific reason is given
4	BAD	Specific server problem with the configuration. For example, the tag has been deleted from the configuration file (tags.xml).
8	BAD	No value may be available at this time, for example the value has not been provided by the data source.
12	BAD	Device failure detected
16	BAD	Timeout before device response.
24	BAD	Communication failure

Quality Code	Quality	Description
28	BAD	No data found for upper or lower bound value Trend interface specific flag.
32	BAD	No data collected (for example, archiving not active. Trend interface specific flag. This value is also used to indicate a temporary offline status (for any condition where sampling was stopped).
64	UNCERTAIN	No specific reason.
65	UNCERTAIN	No specific reason. The value has 'pegged' at some lower limit.
66	UNCERTAIN	No specific reason. The value has 'pegged' at some higher limit.
67	UNCERTAIN	No specific reason. The value is a constant and cannot move.
84	UNCERTAIN	Returned value outside its defined limits defined. In this case the **Limits** field indicates which limit has been exceeded but the value can move farther out of this range.
85	UNCERTAIN	Returned value outside its defined limits defined. In this case the **Limits** field indicates which limit has been exceeded but the value can move farther out of this range. The value has 'pegged' at some lower limit.
86	UNCERTAIN	Returned value outside its defined limits defined. In this case the **Limits** field indicates which limit has been exceeded but the value can move farther out of this range. The value has 'pegged' at some higher limit
87	UNCERTAIN	Returned value outside its defined limits defined. In this case the **Limits** field indicates which limit has been exceeded but the value can move farther out of this range. The value is a constant and cannot move.
192	GOOD	-

DeleteTrend

Deletes saved trend data.

Define the name of the trend from which you want to delete logs.

DumpEventArchive

Stores historical alarm log and audit trail data to external drives, such as USB memory or SD card.

Parameter	Description
EventArchive	Name of buffer to dump data
FolderPath	Destination folder • Internal = *\Flash\QTHMI\workspace\Dump* • USB drive = *\USBMemory* • SD Card = *\Storage Card* • Public Network = *\\<hostname or IP>\sharePath* • Private Network = *\\<username>:<password>@<hostname or IP>\sharePath* Note: supported formats for external memory are FAT or FAT32 (NTFS format is not supported). Note: Private networks are supported only from Linux devices with BSP 1.0.25 and above.
DumpConfigFile	Dump the description files of the archives
DumpAsCSV	**true** = the buffer is dumped to the specified location as a .csv file **false** = the buffer is dumped in binary format (a .dat file and .inf file). Both these files are then required to convert data in .csv format by an external utility.
DateTimePrefix	**true** = the dumped file will have date and time as prefix to its name (for example D2012_01_01_T10_10_alarmBuffer1.csv)
timeSpec	Time format: • **Local** = the time values exported are the time of the HMI device. • **Global** = the time values exported are in UTC format.
csv Colums	Select the columns to dump into the .csv file. Available only when the EventArchive is an alarms buffer
FileName	The below wildcards are supported • %n = Event archive name • %y = Year • %M = Month • %d = Day • %h = Hour • %m = Minutes • %s = Seconds

Parameter	Description
	Example: \%n\%y%M%d\%h%m%s Available only when the DateTimePrefixFileName=true
Language	Select the language to use. When empty, dump will execute on all languages. Available only when the EventArchive is an alarms buffer

Example

When exporting Event buffers in binary format and **DumpConfigFile** is set to true (recommended settings), there are two folders:

- **data**, containing data files,
- **config**, containing configuration files for .csv conversion.

Once the two folders are copied from the USB drive to the computer disk, the folder structure will be:

config\

alarms.xml

eventconfig.xml

data\

AlarmBuffer1.dat

AlarmBuffer1.inf

\

AlarmBufferReader.exe

To convert dump files to .csv

The AlarmBufferReader.exe tool is stored in the *Utils* folder of the Wizard installation folder.

Use the following syntax:

```
AlarmBufferReader AlarmBuffer1 FILE ./AlarmBuffer1.csv
```

where:

`AlarmBuffer1` = name of the dumped .dat without extension

`AlarmBuffer1.csv` = name for the output file.

The utility AuditTrailBufferReader.exe is available for Audit Trail buffers.

Note: set DumpConfigFile to **true**.

The result of the dump is a folder structure similar to the one generated for Events.

Use the following syntax:

`AuditTrailBufferReader AuditTrail FILE ./AuditTrail.csv`

where:

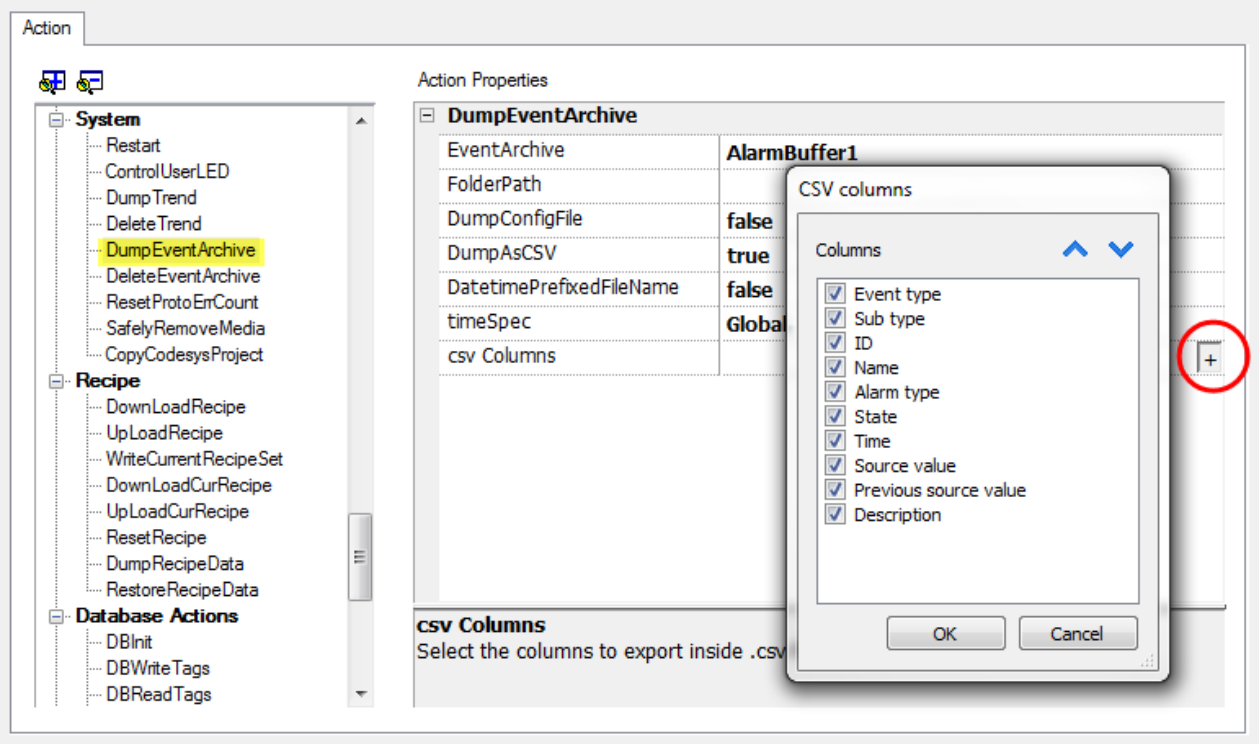
`AuditTrail` = name of the dumped buffer without extension and

`AuditTrail1.csv` = name for the output file.

csv Columns

Note: available only for Alarms buffers.

For Alarms buffers, additional columns can be included in the dump .csv file.



DeleteEventArchive

Deletes saved Event buffers log data.

Specify the name of Event buffer to delete from the Event logs.

ResetProtoErrCount

Resets the Protocol Error Count system variable.

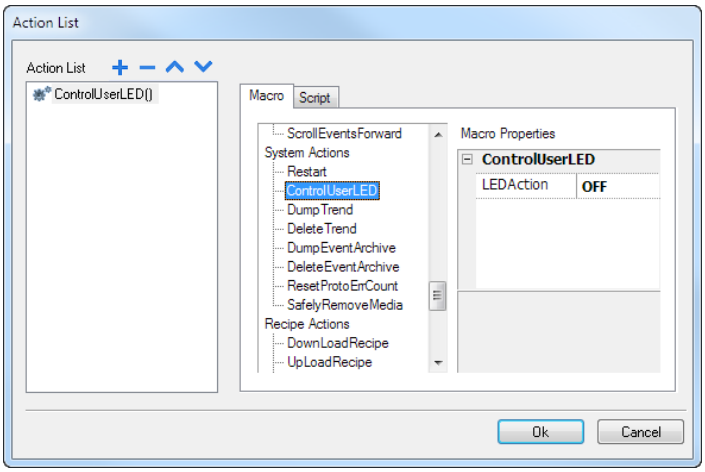
See "System Variables" on page 79 for details.

SafelyRemoveMedia

Provides for safe removal of SD card or USB drive fromHMI.

ControlUserLED

Sets the user LED behavior.



Not available on Linux platforms (see "HMI devices capabilities" on page 429 for panels details)

Tag actions

Interacts with tags.

DataTransfer

Exchanges data between:

- two controllers,
- registers within a controller,
- from system variables to controllers,
- from controllers to system variables

The various tag types include a controller tag, a system variable, a recipe tag and widget property.

ToggleBit

Toggles a bit value of a tag.

BitIndex allows you to select the bit to be toggled: toggling requires a read-modify-write operation; the read value is inverted and then written back to the tag.

SetBit

Sets the selected bit to “1".

BitIndex allows you to select the bit position inside the tag.

ResetBit

Resets the selected bit to “0”

BitIndex allows you to select the bit position inside the tag.

WriteTag

Writes constant values to the controller memory. Specify tag name and value.

StepTag

Increments or decrements tag value.

Parameter	Description
TagName	Name of tag to increase/decrease
Step	Step value
Do not step over limit	Enables step limit
Step Limit	Value of step limit, if enabled.

ActivateGroup

Forces the update of a group of tags.

Tags are updated either when used in the current page or continuously, if defined as active in the Tag Editor. This action forces all the tags of a group to be continuously updated.

DeactivateGroup

Deactivates a group of tags, that is stops forcing the update of a group of tags.

EnableNode

Enable/disables action for offline node management. No communication is done with a disabled node.

Parameter	Description
Protocol ID	Unique identifier of selected protocol
NodeID	Node identifier in selected protocol. Can be attached to a tag.
Enable	Node communication status: **False** = disabled **True** = enabled When attached to a tag, tag = 0 means **False**

BACnetClearPriority

Refer to the BACnet manual inside the “Communication Drivers” folder for a detailed description of BACnet actions.

BACnetClearAllPriorities

Refer to the BACnet manual inside the “Communication Drivers” folder for a detailed description of BACnet actions.

BACnetSetPriority

Refer to the BACnet manual inside the “Communication Drivers” folder for a detailed description of BACnet actions.

ClearRetentiveMemory

When set to 0, clears the content of the Retentive Memory.

ForceReadTag

Force a refresh of the specified tag from the remote controller.

Trend actions

Used for Live Data Trends and Historical Trends Widget.

RefreshTrend

Refreshes the **Trend** window.

It can be used in any Trends/Graphs widgets. Specify the widget as a parameter for the action.

ScrollLeftTrend

Scrolls the **Trend** window to the left side, by one-tenth (1/10) of the page duration.

Note: with the real-time trends pause the trend using the **PauseTrend** action, or the window will be continuously shifted to the current value.

ScrollRightTrend

Scrolls the **Trend** window to the right side, by one-tenth (1/10) of the page duration.

Note: with the real-time trends pause the trend using the **PauseTrend** action, or the window will be continuously shifted to the current value.

PageLeftTrend

Scrolls the **Trend** window by one-page. For example, if the page size is 10 minutes, then use the **PageLeftTrend** action to scroll the trend left for 10 minutes.

PageRightTrend

Scrolls the **Trend** window by one-page. For example, if the page size is 10 minutes, then use the **PageRightTrend** action to scroll the trend right for 10 minutes.

PageDurationTrend

Sets the page duration of the **Trend** window.

Define trend name and page duration.

Note: you can set page duration at run time using a combo box widget.

ZoomInTrend

Reduces page duration.

ZoomOutTrend

Extends page duration.

ZoomResetTrend

Reset the zoom level back to the original zoom level.

ZoomInYAxisTrend

Reduces Y Axis.

ZoomOutYAxisTrend

Extends Y Axis.

ZoomResetYAxisTrend

Reset the Y Axis zoom level back to the original zoom level.

PauseTrend

Stops plotting the trend curves in the **Trend** window.

When used with real time trend the plotting stops when the curve reaches the right border of the graph. This action does not stop trend logging.

ResumeTrend

Resumes trend plotting if paused.

ShowTrendCursor

Shows value of the curve at a given point on the X axis.

It activates the trend cursor. A cursor (vertical line) will be displayed in the trend widget.

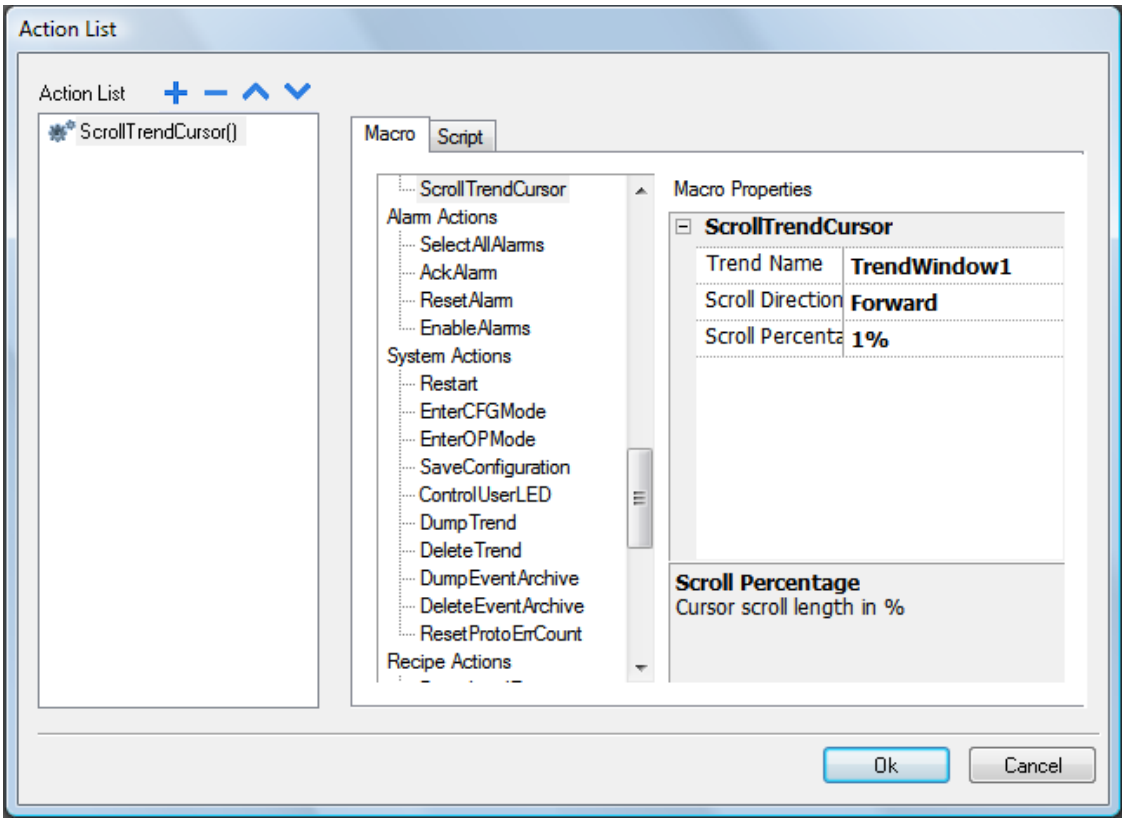
When the graphic cursor is enabled, the scrolling of the trend is stopped.

The **ScrollCursor** action moves the graphic cursor over the curves, or over the entire **Trend** window.

ScrollTrendCursor

Scrolls the trend cursor backward or forward.

The Y cursor value will display the trend value at the point of the cursor. Scrolling percentage can be set at 1% or 10%. The percentage is calculated on the trend window duration.



ScrollTrendToTime

Scrolls the **Trend** window to a specified point in time.

Use this action when you need to scroll to a specific position in a trend window when a specific event occurred.

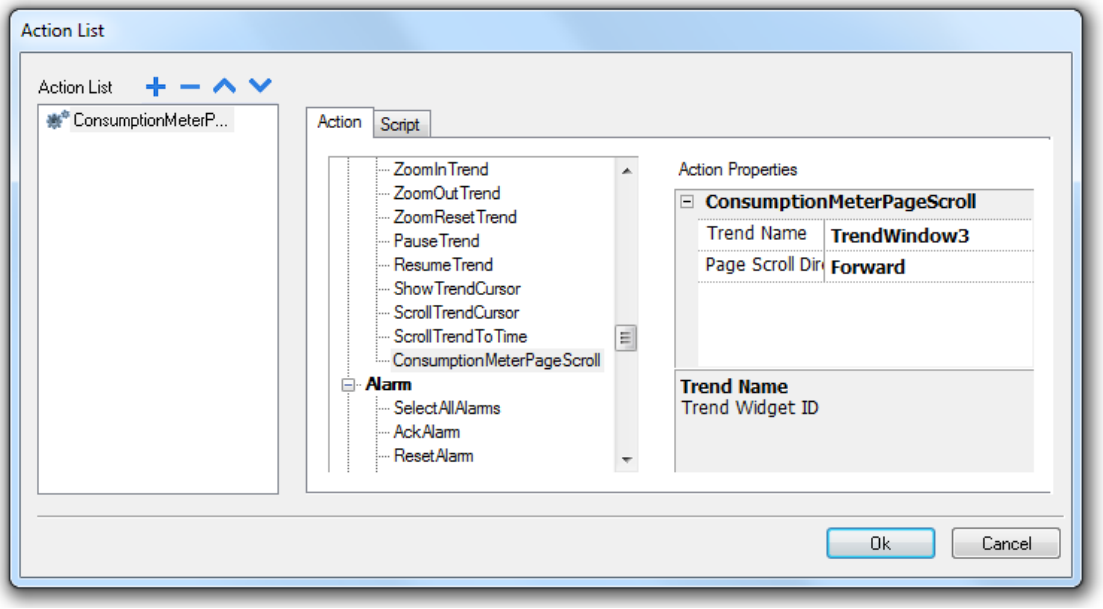
Example

1. Configure an action for an event (for example, an alarm) that executes a data transfer of the system time into a tag.
2. Select that tag as **ScrollTrendtoTime** parameter: the trend windows will be centered at the time when the event was triggered.

ConsumptionMeterPageScroll

Scrolls the page backward or forward in a Consumption Meter widget.

Parameter	Description
Trend Name	Trend widget ID (for example, TrendWindow3)
Page Scroll Direction	Direction of page scrolling (Forward/backward)



User management actions

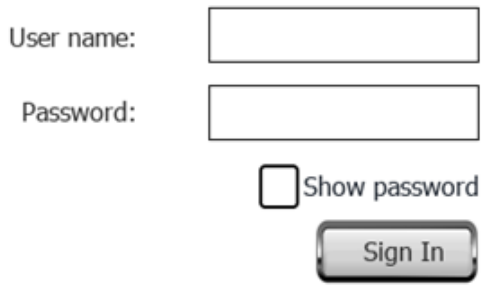
User management and security settings.

LogOut

Logs off the current user. The default user is then automatically logged in. If no default user has been configured, the logon window is displayed.

SwitchUser

Switches between two users without logging off the logged user: the user login dialog appears. User can click **Back** to go back to the previously logged user.



The server continues running with the previously logged user, until the next user logs on. One user is always logged onto the system.

ChangePassword

Change current user password: a dialog appears

No parameter is required.

ResetPassword

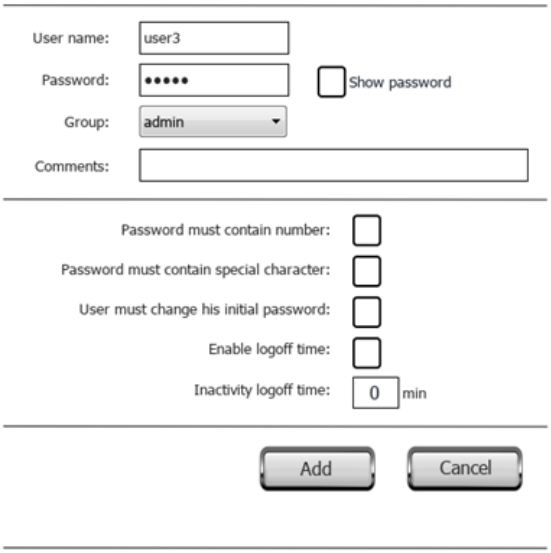
Restores the original password together with the settings specified in the project for the current user.

No parameter is required.

AddUser

*Reserved to users with **Can manage other users** property set.*

Adds a user at run time: a dialog appears.



DeleteUser

*Reserved to users with **Can manage other users** property set.*

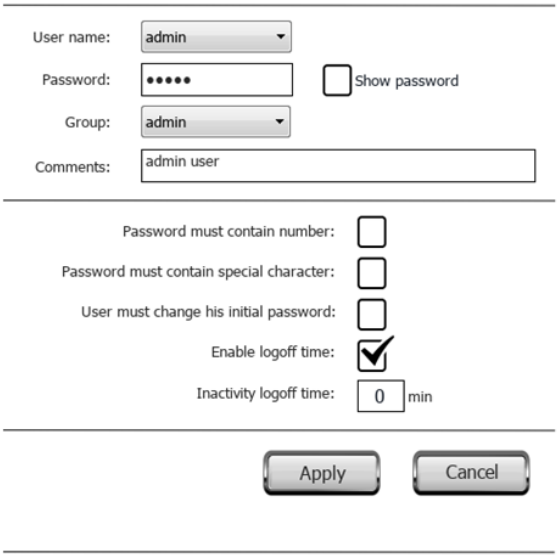
Deletes a user at run time: a dialog appears.

No parameter is required.

EditUsers

*Reserved to users with **Can manage other users** property set.*

Edits user settings.



DeleteUMDynamicFile

Deletes the dynamic user management file. Changes made to users settings at run time are erased. The original settings are restored from the project information.

No parameter is required.

ExportUsers

Exports user settings to an .xml file (*usermgnt_user.xml*) in encrypted format to be restored when needed.

Set destination folder for the export file.

Important: The user file is encrypted and cannot be edited.

Note: supported formats are FAT or FAT32. NTFS format is not supported.

ImportUsers

Imports user settings from a previously saved export .xml file (*usermgnt_user.xml*).

Set source folder for the import file.

Note: supported formats are FAT or FAT32. NTFS format is not supported.

Widget actions

ShowWidget

Shows or hides page widgets.

Property	Description
Widget	Widget to show/hide

SlideWidget

Shows the sliding effect of a widget, or of a widget group.

Note: The widget or grouped widgets can actually be outside of visible part of the page in the project and slide in and out of view.

Property	Description
Widget	Widget to slide
Direction	Sliding direction
Speed	Transition speed of sliding widget
X Distance	Travel distance of X coordinate in pixels
Y Distance	Travel distance of Y coordinate in pixels
Slide Limit	Enable/Disable movement limits of the widget with respect to the x, y coordinates
X Limit	Limit position of slide action for x coordinate
Y Limit	Limit position of slide action for y coordinate
Toggle Visibility	Show/hide widget at the end of each slide action
Image Widget	Image displayed during slide action

BeginDataEntry

Displays a keypad and starts data entry on a data field without touching the widget itself. This action can be used to activate data entry using a barcode scanner.

Java Script Interface

```
project.beginDataEntry(wgtName [, pageName])
```

Parameter	Description
wgtNameWidget	Widget name
pageName	Active page for data entry. Optional parameter. Useful to select a data field inside a non-modal active dialog box.

TriggerIPCamera

Captures an image from an IP Camera. Only works on pages that include an IP Camera widget.

MoveIPCamera

Sends remote commands to a camera that supports them. See "IP Camera widgets" on page 293 for details. Make sure that the IP Camera supports movement commands.

RefreshEvent

Refreshes the event buffer for **Alarm History** widget. See "Alarms History widget" on page 174 for details.

ContextMenu

Displays the context menu.

If **Context Menu** property of Project Widget has been set to **On delay** context menu can appear also touching for a few seconds the background area of the screen. See "Project properties pane" on page 54

ReplaceMedia

Replaces existing media files with new files from USB/SD card. Can be used to replace video files of MediaPlayer widgets, or images of project.

Note: New media files must have same name and format of the files to be replaced.

Parameter	Description
Media Type	Type of file to update
Device	Device where new media files are supplied
sourcePath	Folder where new media files are stored (for example, "\USBMemory")
Image Resize	Resizes new images to the size of images to be replaced. Not applicable to video files.
Silent	Replaces media automatically. As defau a dialog is displayed for the user to specify file location.

Java Script Interface

```
void replaceMedia(var sourcePath, var bSilent, var Device, var nMediaType, var
bResize)
```

```
project.replaceMedia("Images", true, "\USBMemory", 1, true);
```

13 Using the Client application

HMI Client is a standalone application which provides remote access to the HMI Runtime, and is included in the Wizard. HMI Client uses the same graphic rendering system as the runtime in the HMI devices, it relies on a specified HMI Runtime as server for live data.

To run the HMI Client application:

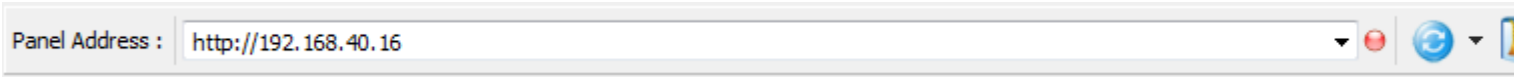
1. From the **Start** menu > **Wizard** >**HMI Client**: the client opens in a browser-like style window.
2. Type the server/device IP address in the address bar (for example: http://192.168.1.12): HMI Client will connect to the server and the same graphical application running on the device will be loaded in the client window.

HMI Client acts as a remote client and communicates to the server, sharing the local visualization with the tag values that are maintained or updated by the communication protocol.

HMI projects contain properties indicating which page is currently displayed on the HMI and can force the HMI to switch to a specific page. You can use these properties to synchronize pages showed on the HMI device and HMI Client or to control an HMI device with a PLC.

See "Behavior" on page 60 for details.

The Client application toolbar



Element	Description
HMI server address	HMI device address
Connection status	Network request status. Red during data exchange.
Reload from cache	Reloads project
BookMark	Bookmarks preferred pages and reload them.
Settings	Opens **Settings** dialog

Reload options

Option	Description
F5	Reloads project from cache
Shift + F5	Downloads project to client

Workspace

Project files are uploaded from the device and stored in HMI Client into the following cache folder.

%appdata%\CGC S.p.A.\[build number]\client\cache

where:

[build number] = folder named as build number, for example 01.90.00.608.

Settings and time zone options

In the **Settings** dialog you can configure client settings and decide how to display project timestamp information.

HTTP settings

Parameter	Description
Protocols	Communication protocol used by HMI Client to communicate with an HMI device.
Update Rate	Polling frequency to synchronize data from server. Default = 1 s.
Timeout	Maximum wait time before a request is repeated by the HMI Client. Default = 5 s.
Reuse connection	Enables reuse of the same TCP connection for multiple HTTP requests to reduce network traffic.

Parameter	Description
	Note: When enabled, this option may cause high latency if the proxy server does not immediately terminate old requests thus saturating connection sockets. This is often the case with 3G connections.
Enable compression	Compresses data to reduce download times. Default = disabled. ***CAUTION: enabling this option could causes excessive CPU overhead.***
Time Settings	Used by the client to adapt the widget time stamp information.

FTP settings

Parameter	Description
Port	FTP communication port

Time settings

Parameter	Description
Use Widget Defaults	Displays time information according to the widget settings.
Local Time	Translates all timestamps in the project into the computer local time where the client is installed.
Global Time	Translates all timestamps in the project into UTC format.
Server Time	Translates all timestamps in the project into the same used by HMI device/server in order to show the same time.

Important: Make sure you set the HMI RTC correct time zone and DST options.

Transferring files to a remote HMI device

You can upload and download files to and from a remote HMI device using two dedicated actions. These actions can only be used from a remote HMI Client and access remote files via FTP.

Important: Enable FTP support and give all necessary user rights to the folders used to transfer files.

See "Remote Client actions" on page 133.

See "Remote Client variables" on page 88.

14 Using the integrated FTP server

HMI Runtime system uses an integrated FTP server.

Connect to the HMI device FTP server using any standard FTP client application. The FTP server responds on the standard port 21 as default.

Important: The server supports only one connection at a time; if you are using a multiple connection FTP client disable this feature on the client program or set the maximum number of connections per session to 1.

FTP settings

FTP default credentials

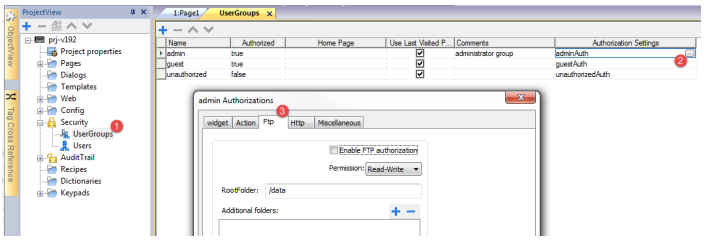
When User Management/Security is disabled use the following credentials for incoming connections:

User name	admin
Password	admin

Changing FTP settings

*Path: **ProjectView*> *Security*> *UserGroups* > *Authorization Settings***

You can change FTP permissions and account information in the **Ftp** tab of the **admin authorizations** dialog.



See "Configuring groups and authorizations" on page 228 for details.

15 Using VNC for remote access

VNC is a remote control software which allows you to see and control the HMI application remotely using your local mouse and keyboard.

Remote access is particularly useful for administration and technical support. In order to use it you need to:

- start a server in the HMI device
- install a viewer on the remote device

Starting VNC server on WinCE devices

VNC server is a plug-in. It can be enabled and downloaded as part of the Runtime. "Software plug-in modules" on page 59.

Installing VNC server

Path: ***ProjectView*** > ***Project properties***

1. In the **Properties** pane set **VNC Server** to **true** to enable the plug-in.
2. Install or update the runtime to add the VNC server.

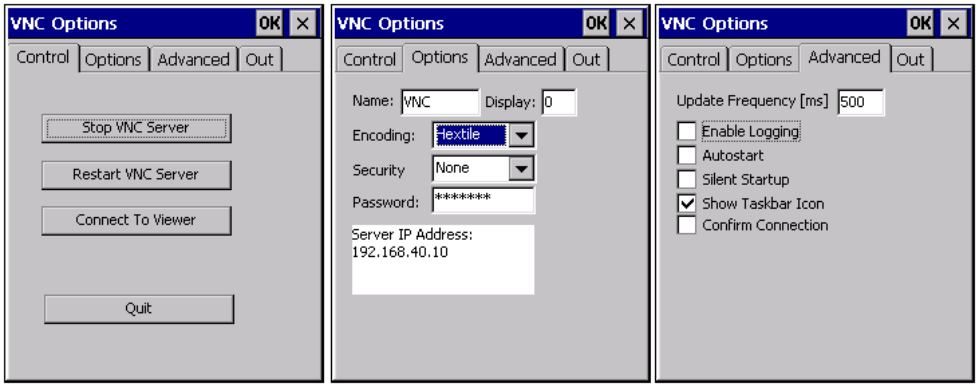
Starting/stopping the VNC server

The VNC server is located in the folder *\Flash\qthmi\VNC* and is activated using the action *launchVNC*. If enabled from the project properties, it can also be activated from the runtime context menu **Developer tools**> **Launch VNC**.

To enable the runtime contextual menu see "Project properties" on page 53

VNC Options dialog

From the **VNC Options** dialog you can perform several tasks.



Tab	Functions
Control	Star/stop the VNC server and connect to viewer
Options	Define security information for server access using a VNC viewer
Advanced	Enable automatic activation of VNC server at HMI device startup.  Select **Silent Startup** to keep the **VNC Options** dialog in the background when **Autostart** is enabled.  *Enable **Show Taskbar Icon** when debugging out of KIOSK mode.*
Out	Contains the configuration settings for an outgoing connection to a listening VNC viewer software.

Important: Settings in the Advanced tab are reserved to expert users and should be modified when the VNC server is used in conjunction with a VNC repeater to overcome firewall problems or optimize VNC performances according to the network configuration.

Connecting to viewer

Many modern VNC viewers offer the possibility to start the software in listening mode. The reason is that mobile devices most of the time do not have a public IP address to refer to. So it is practical to have a public IP address on an Office Computer which runs a listening VNC viewer. A user can then easily call for support by pressing the **Connect to viewer** button on the Control tab.

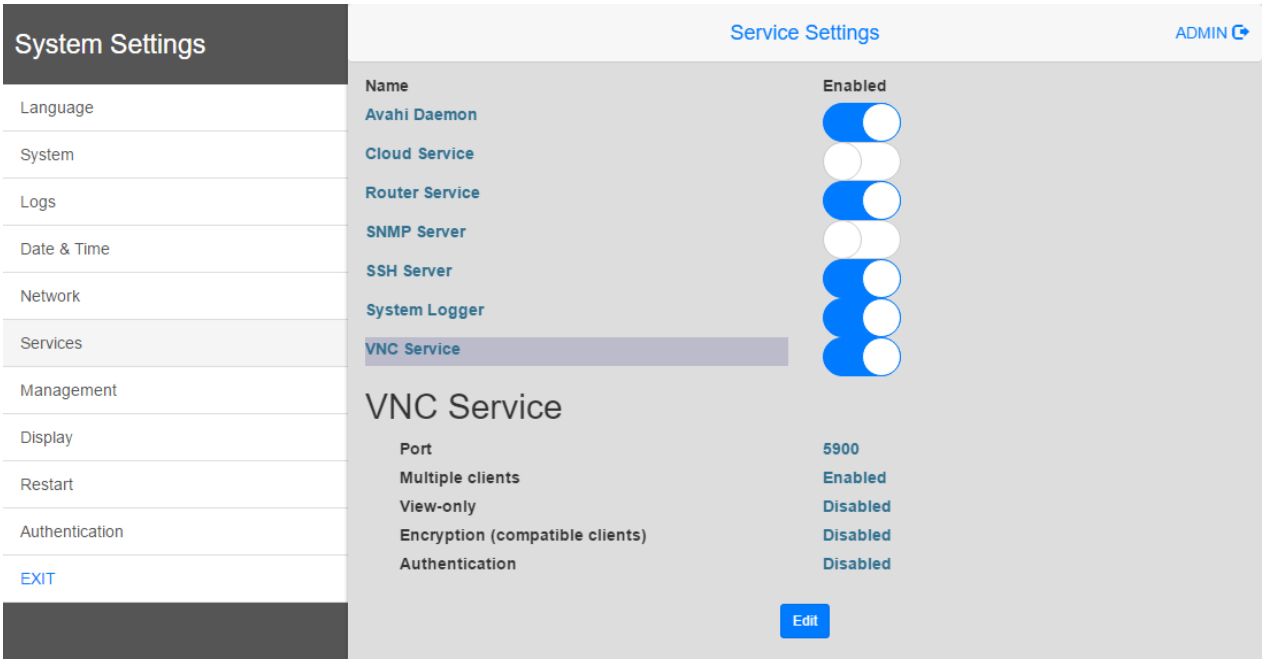
VNC default settings

TCP port	5900
Password	null

Important: The VNC server allows only a single client.

Starting VNC server on Linux devices

VNC server is a service embedded inside the BSP that can be activated from the Services tab of the device System Settings. See "System Settings" on page 384 for details.



Starting VNC viewer

No VNC viewer is provided as part of Wizard.

Many compatible VNC viewers are available for free download (for example, TightVNC).

16 Alarms

The alarms handling system has been designed to provide alerts through pop-up messages, typically to display warning messages indicating any abnormal condition or malfunction in the system under control.

Whenever a bit changes, or the value of a tag exceeds a threshold set in the alarm configuration, a message is displayed. Specific actions can also be programmed to be executed when an alarm is triggered.

Important: No default action is associated with any alarm.

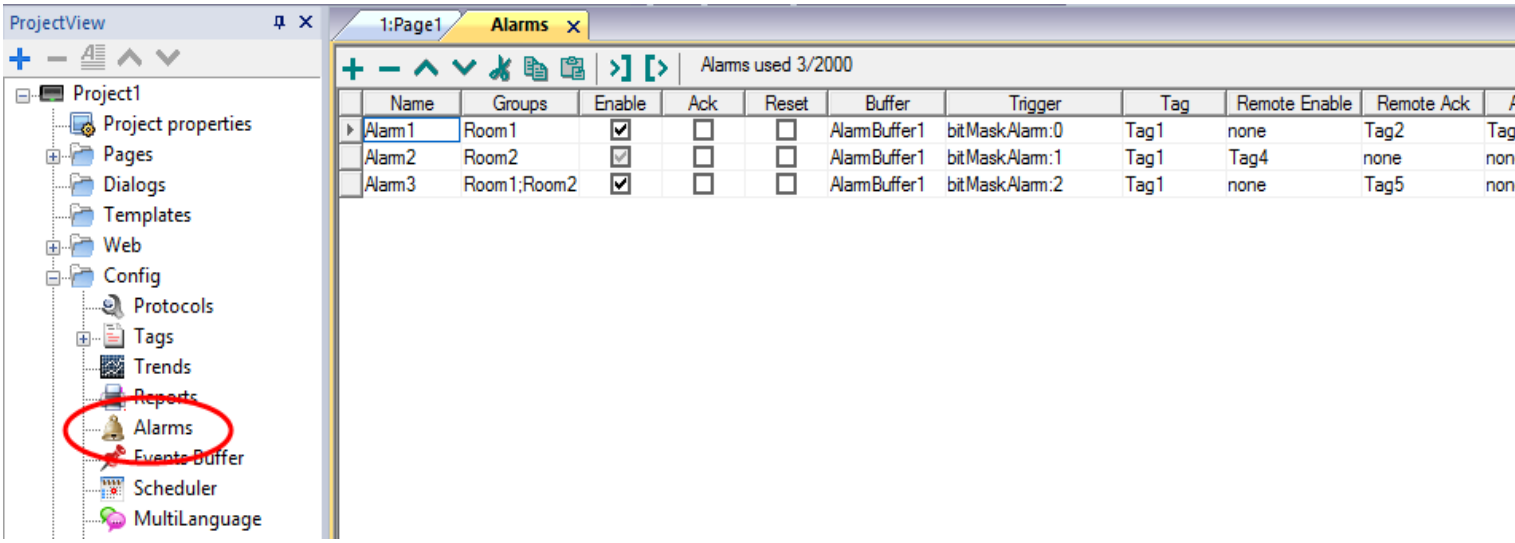
You can define how an alarm is displayed on the HMI device, if it requires user acknowledgment, and if and how it is logged into the event list.

Alarms are configured in the Alarms Configuration Editor and, thus, are available for all the pages of the project. An alarm widget can display more than one alarm at a time, if sized appropriately. You can trigger the opening or closing of the Alarm window with an event.

You work with alarms in the same way as you work with any other event. You may not want to display a dialog when an alarm is triggered and you can associate to it any other available action.

Alarms Editor

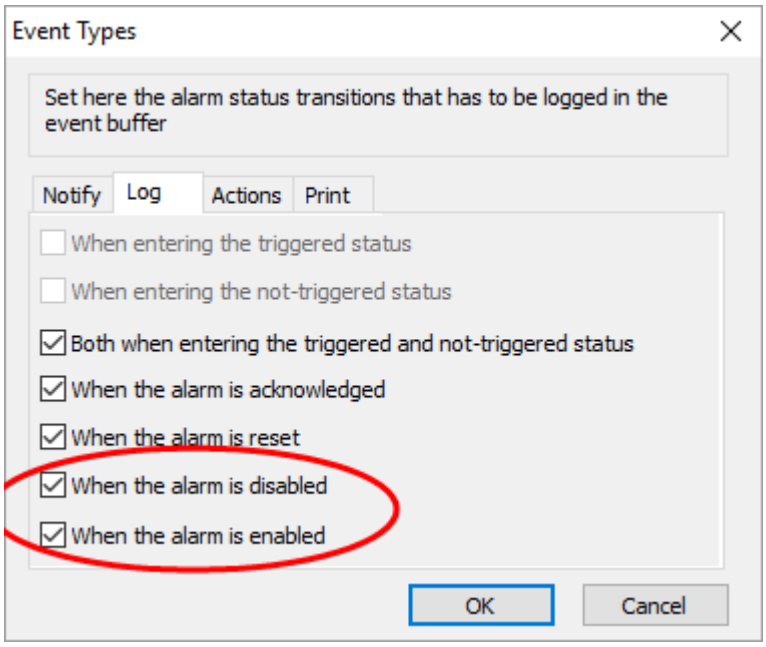
Path: ***ProjectView*** *>* ***Config*** *> double-click* ***Alarms***



Adding an alarm

Click **+** to add an alarm.

Parameter	Description
Name	Name of alarm
Groups	Groups associated with the alarm. They can be used in widgets display filters.
Enable	Enable/disable triggering of alarm. Alarms can be enabled or disabled at run time as well (see "Enable/disable alarms at run time" on page 175 for details).
Ack	Enable/disable acknowledgment of alarm, if selected the operator must acknowledge the alarm once triggered to remove it from the **Active Alarm** widget.
Reset	Used with the **Ack** option, if selected, acknowledged alarms stay in the alarm list, labeled as **Not Triggered Acked**, until the operator presses the **Reset** button in the alarm widget.
Buffer	Buffer file where the alarm history will be saved.
Trigger	Triggering condition depending on alarm type: • **limitAlarm**: alarm triggered when tag value exceeds its limits. The alarm is not triggered if the value reaches the limits. • **valueAlarm** alarm is triggered when tag value is equal to the configured value

Parameter	Description
	• **bitMaskAlarm**: the bitwise AND operator compares each bit of the bitmask with the tag value corresponding to that Alarm. If both bits are on, the alarm is set to true. You can specify one or more bit positions (starting from 0) inside the tag. The Bit position must be given in decimal format; if more bits are specified, each position must be separated by a ",". • **deviationAlarm**: alarm triggered if the percentage of deviation of the tag value from the set point exceeds a set deviation. $\|Value_{now} - SetPoint\| > \left(\frac{deviation}{100} \times SetPoint\right)$
Tag	Tag whose value will trigger the alarm when it exceeds the set limits. The alarm can refer to the value of this tag, or to the state of a bit if **bitMaskAlarm** has been selected as trigger.
Remote Enable	Tag used by the PLC to enable/disable the alarm. • Changing the enable status from the Alarms Widget will change the tag value • When the tag cannot be read (e.g. communication error) the alarm is disabled • No tags related to the alarm are refreshed when alarm is disabled. Tip: It could be useful to enable the logging of the alarm's enable flag 
Remote Ack	Tag used by the PLC to acknowledge the alarm. A transition of this tag from 0 to a non zero value is considered an acknowledgment request. Leave empty if remote acknowledgment is not required. See "Remote alarms acknowledge" on the next page for details.
Ack Notify	Tag used by the HMI device to notify when the alarm is acknowledged from the device or from the PLC.

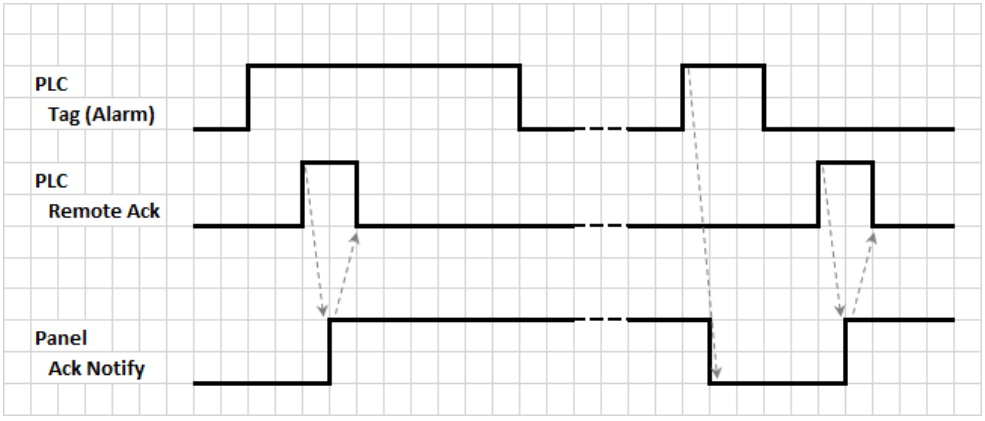
Parameter	Description
	0 = set to this value when alarm is triggered 1 = set to this value when alarm is acknowledged.
Action	Actions executed when the alarm is triggered. Additional conditions can be specified in the **Events** column. See "Setting events" on page 168 for details.
User Action	Actions executed when user press the action button in the active alarm widget. See ""Active Alarms widget" on page 170 for details.
Description	Alarm description. This text supports the multiple language features and can be a combination of static and dynamic parts, where the dynamic portion includes one or more tag values. See "Displaying live alarm data" on page 176 for details.
Color	Foreground and background colors of alarm rows based on the status of alarm.
AckBlink	Blinking for triggered alarms. If selected the alarm rows blinks until acknowledged. Only effective if **Ack** is selected.
Severity	Severity of the alarm. If multiple alarms are triggered simultaneously, actions will be executed based on severity settings. 0 = not important 1 = low 2 = below normal 3 = normal 4 = above normal 5 = high 6 = critical
Events	Conditions in which the alarms are notified, logged or printed. See "Setting events" on page 168 for details.

Remote alarms acknowledge

When the **Remote Ack** parameter is set, an alarm can be acknowledged from a PLC device setting a tag value to a nonzero value. The acknowledged status is notified to the PLC device by the **Ack Notify** flag.

Alarms acknowledgement process

Remote Ack tag is set/reset by the PLC to request the acknowledge, and **Ack Notify** is set/reset by HMI device to notify the execution of the acknowledge.



1. When an alarm condition is detected the HMI device set **Ack Notify** to 0 and all related actions are executed.
2. When the alarm is acknowledged (by HMI device or remotely), **Ack Notify** is set to 1
3. It's up to the controller to set **Remote Ack** to 1 to acknowledge the alarm or reset it to 0 when the HMI device send a notification that the alarm has been acknowledged (**Ack Notify** = 1)

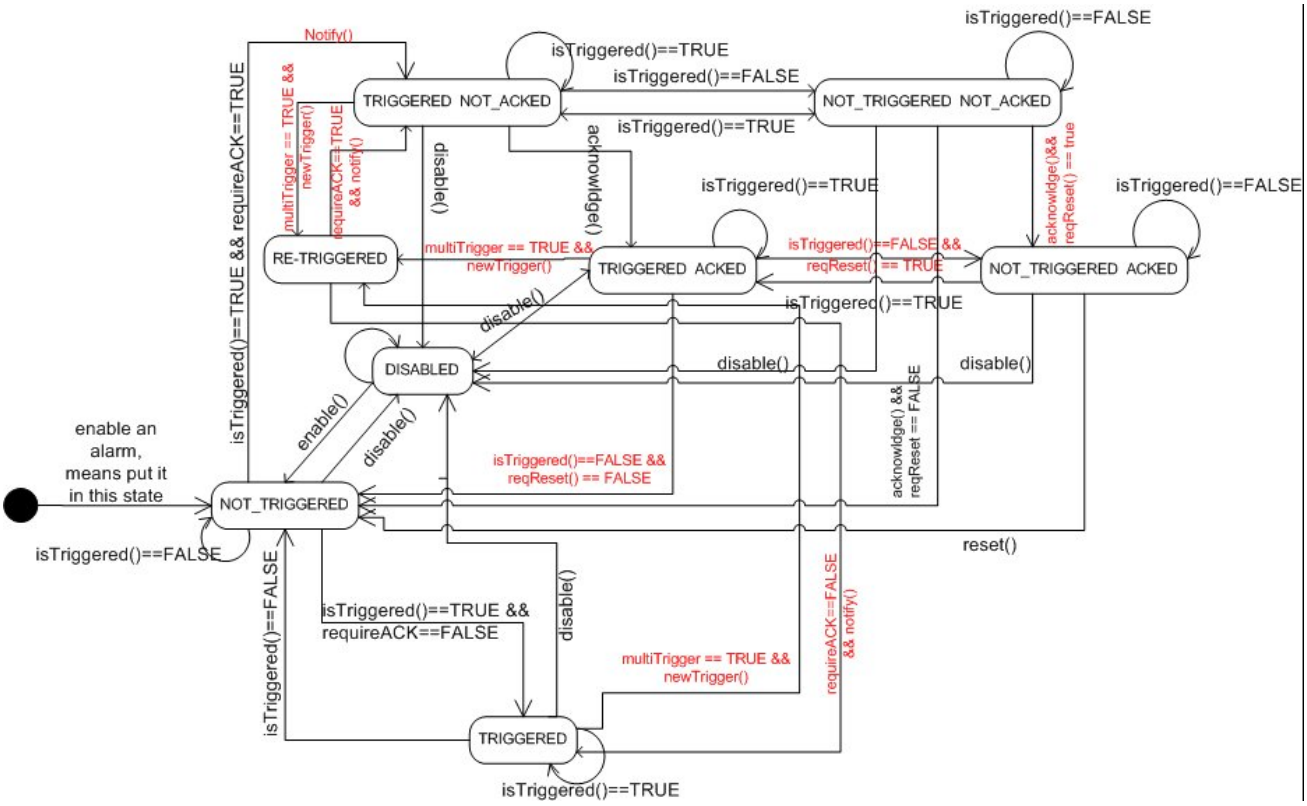
WARNING: When an alarm is triggered, some signals need to be update/communicated through the connected devices. We assume the Acknowledge to be a signal pushed from an operator and not released automatically from a controller device. This allows for time required to communicated the original signals.

Tip: Using the same tag both for **Remote Ack** and **Ack Notify** can connect more devices to the same controller and acknowledge the alarms from any HMI device.

Alarm state machine

The runtime implements the alarm state machine described in this diagram.

States and transitions between states are described according to the selected options and desired behavior.



Setting events

Path: **ProjectView**> **Config** > **Alarms** > **Events** column

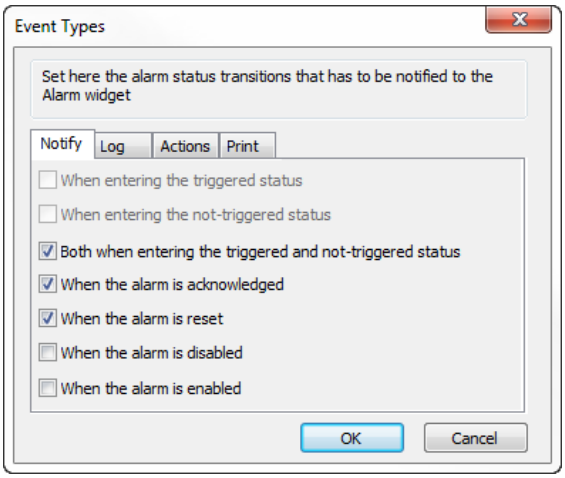
Events are defined using the Alarms Editor.

See "Alarms Editor" on page 164 for details.

Notifying events

Path: **ProjectView**> **Config** > **Alarms** > **Events** column > **Notify** tab

Set conditions under which the alarms will be posted in the alarm widget.



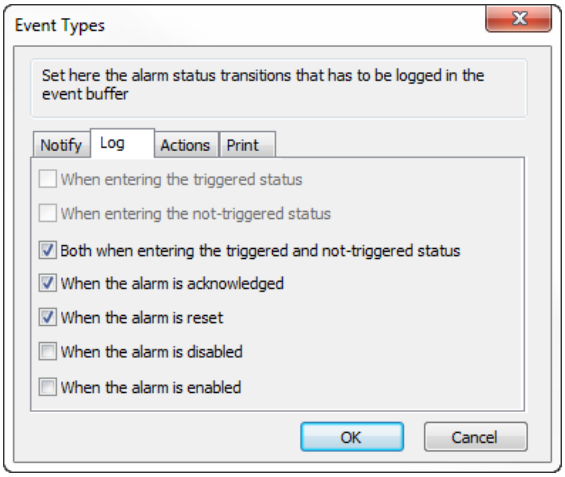
Here you define the behavior of the default alarm widget available in the Widget gallery and decide in which cases the widget is updated by a change in an alarm status.

CAUTION: Make only the adjustments required by the specific application while leaving all other settings as default.

Logging events

Path: **ProjectView**> **Config** > **Alarms** > **Events** column > **Log** tab

Set conditions for which you want to store the specific event in an alarm history buffer.

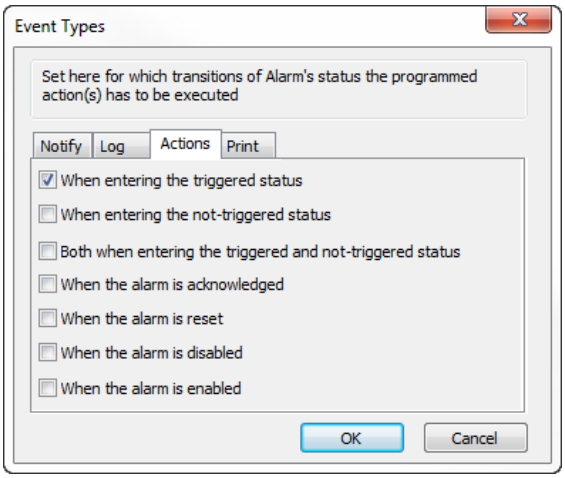


The alarm history is logged in the Event Buffer.

Executing actions

Path: **ProjectView**> **Config** > **Alarms** > **Events** column > **Actions** tab

Set conditions under which the action(s), configured for the specific alarm, must be executed.

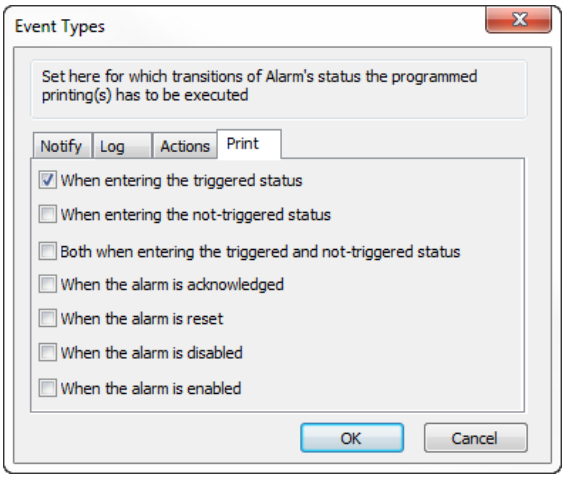


By default, actions are executed only when the alarm is triggered; other alarm states can also be set to execute actions.

Print events

Path: **ProjectView**> **Config** > **Alarms** > **Events** column > **Print** tab

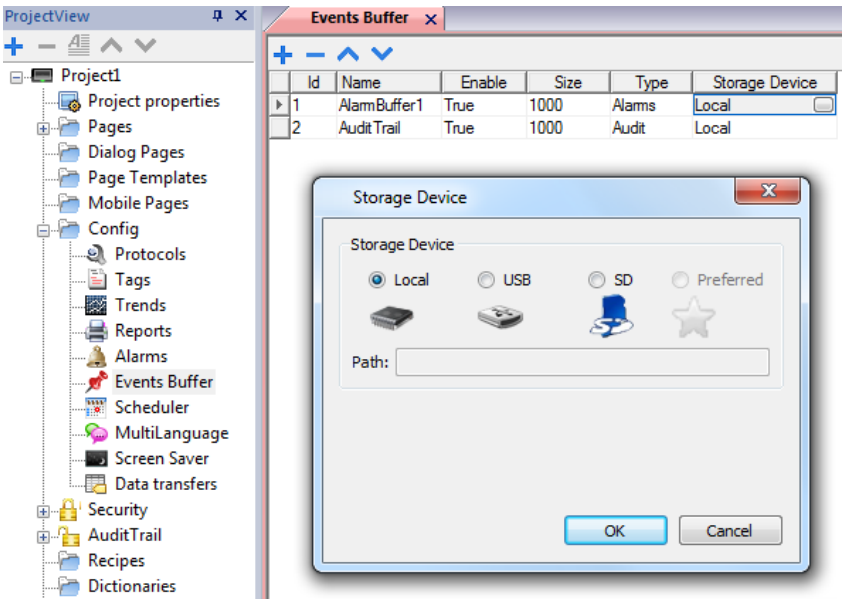
Set conditions for which you want to print the specific event



Setting storage device

Path: ***ProjectView*** *>* ***Config*** *>* ***Events Buffer*** *>* ***Storage Device*** *tab*

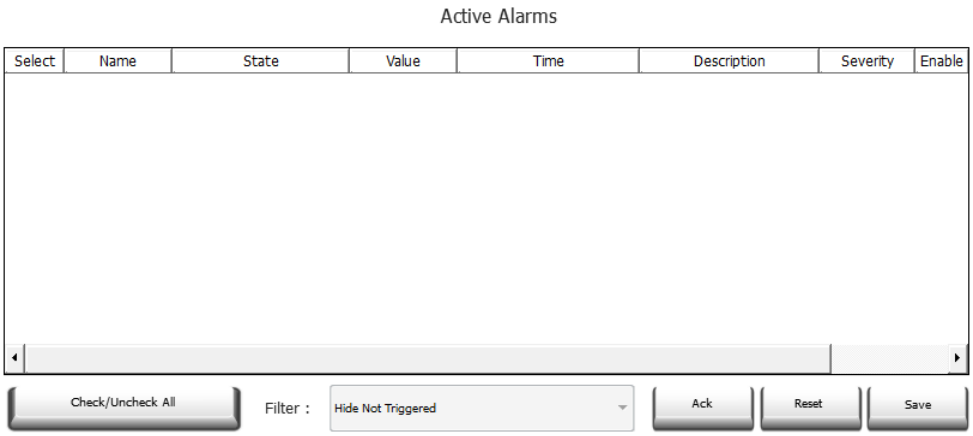
1. Open the **Storage Device** dialog.
2. Select a device for event data storage.



Data is automatically saved every five minutes except for alarm data which is saved immediately.

Active Alarms widget

You can insert the **Active Alarms** widget in a page to display the alarms and to acknowledge, reset or enable/disable them.



Alarm filters

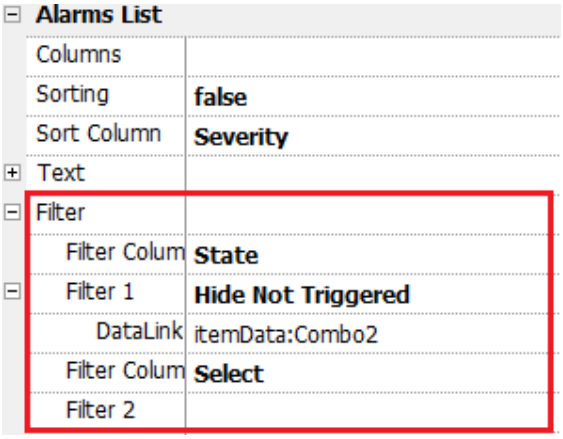
*Path: **ActiveAlarm** widget> **Properties** pane> **Filter***

Define filters used to display only some of the configured alarms. Filters are based on alarm fields, which means you can filter alarms according to name, severity, description and so on.

Filter 1 is the default filter. It's managed by the combo box **Filter 1**, and has two options: **Show all alarms** and **Hide Not Triggered** which, when selected, allows to display only active alarms.

Filter 2 is, by default, not configured and available for customization.

Filter's expressions make use of AWK language, the expressions are applied to the data contained in the selected **Filter** column of the Alarm widget.



Setting filters

*Path: **ActiveAlarm** widget> **Properties** pane> **Filter***

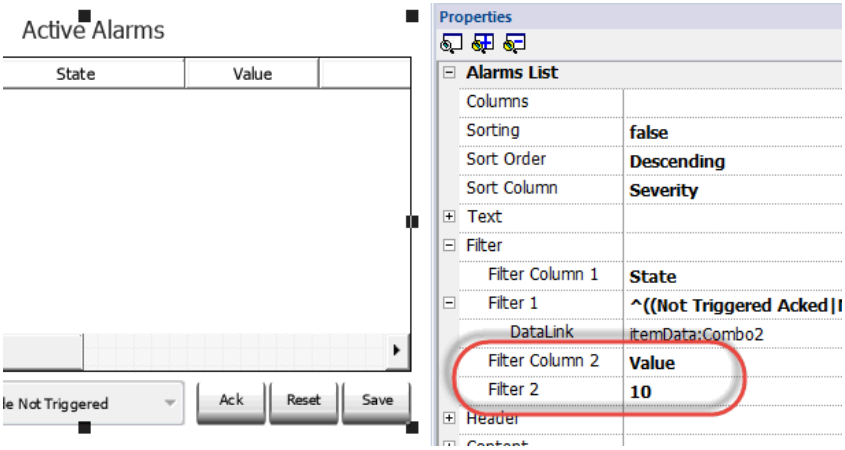
To set one of the two available filters:

1. Select **Filter Column 1** and choose the value to filter for (e.g.: Name, State, Time, Groups)
2. In **DataLink** attach a combo box widget. Use Shift+ left-click to select the combo box.
3. In the **Properties** pane select list property and open dialog to customize combo box values
4. In the combo box configuration dialog, specify **String List** and the regular expression to filter values.

See *http://www.gnu.org/software/gawk/manual/gawk.html* for details on how to use regular expressions.

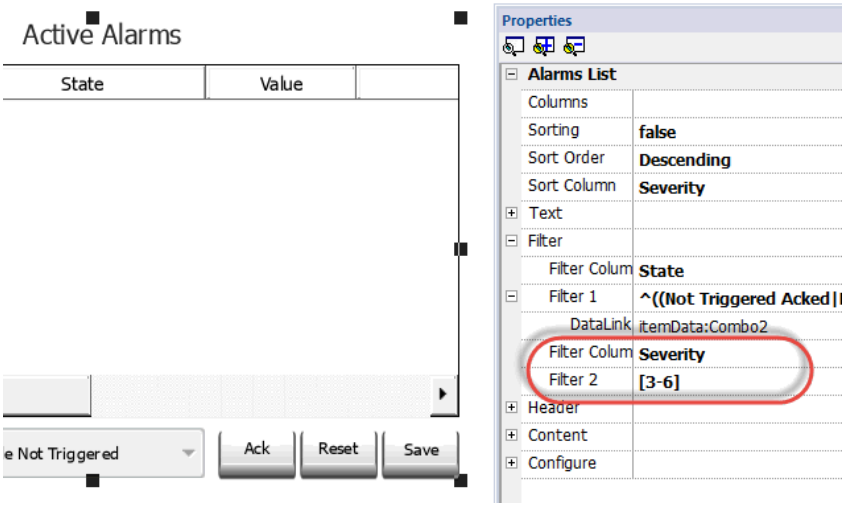
Filters first example

You want to show all alarms matching Filter 1 with value equal to 10. Then properties settings: **Filter column 2** = Value, **Filter 2** = 10



Filters second example

You want to show all alarms matching a Severity value from 3 to 6 (Normal to Critical). Then properties settings: **Filter column 2** = Severity, **Filter 2** = [3-6]



Filters third example

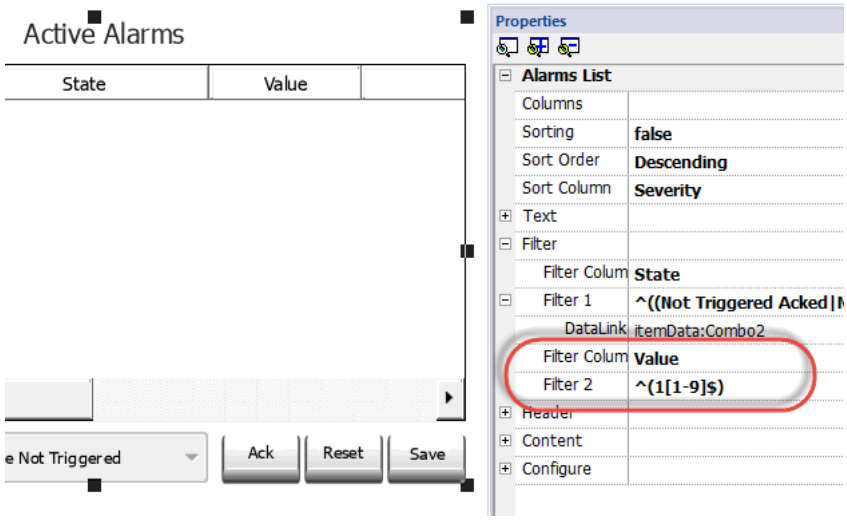
You want to show all alarms matching a value from 11 to 19. Then properties settings: **Filter column 2** = Severity, **Filter 2** = ^(1[1-9]$)

Meaning:

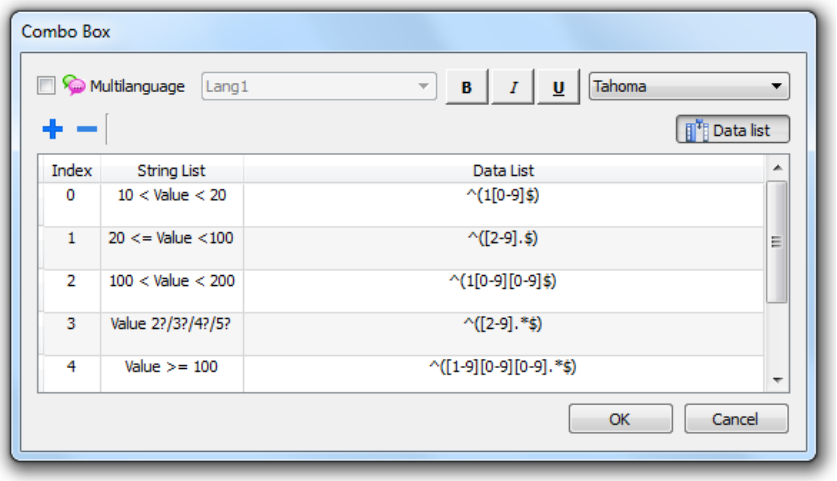
^ = match must starts from the beginning of the string

1[1-9] = first char must be 1 and the second char must be between 1 and 9

$ = end of the comparison.



Filters expression examples



Filter by	String list	Data list
State	Hide Not Triggered	^((Not Triggered Acked|Not Triggered Not Acked|Triggered).*$)
Value	10 < Value < 20	^(1[0-9]$)
Value	20 <= Value <100	^([2-9].$)
Value	100 < Value < 200	^(1[0-9][0-9]$)
Value	Value 2?/3?/4?/5?	^([2-9].*$)
Value	Value >= 100	^([1-9][0-9][0-9].*$)
Value	Value >= 20	^([2-9].*$\|[1-9][0-9][0-9].*$)

Sorting alarms

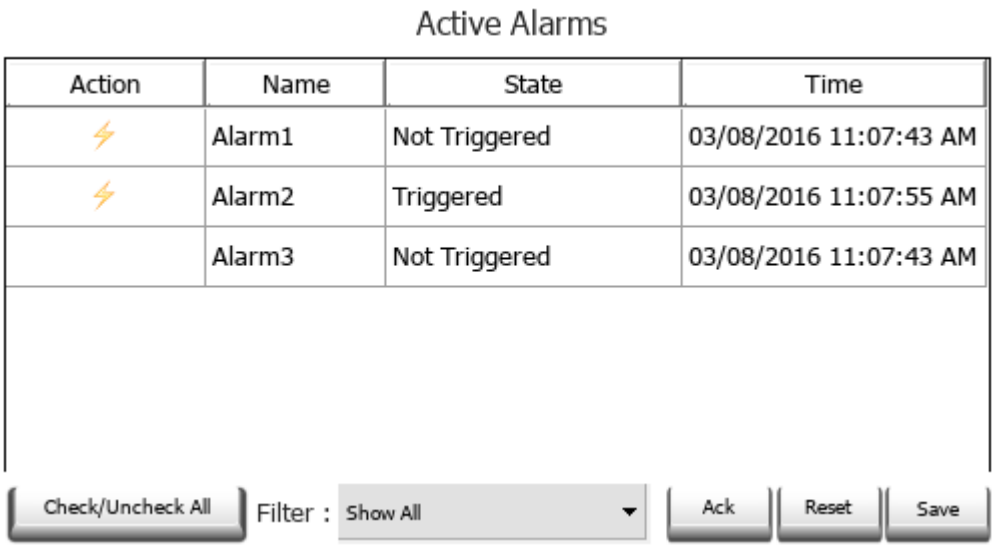
Path: ***ActiveAlarm*** *widget>* ***Properties*** *pane>* ***Sorting***

The sorting function allows you to sort alarms at run time in the alarms widget by clicking on the column header.

Note: The severity value displayed here is set in the Alarm Editor.

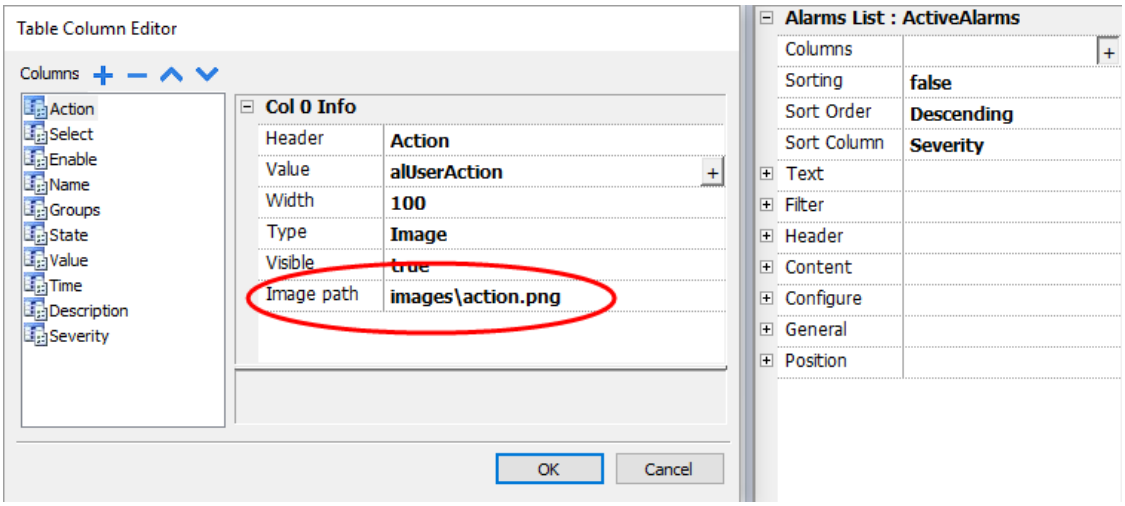
Action

When the "User Action" associate with the alarm (see ""Alarms Editor" on page 164 for details) contains valid actions, the Action icon is showed. Pressing the icon, the configured actions will be executed.



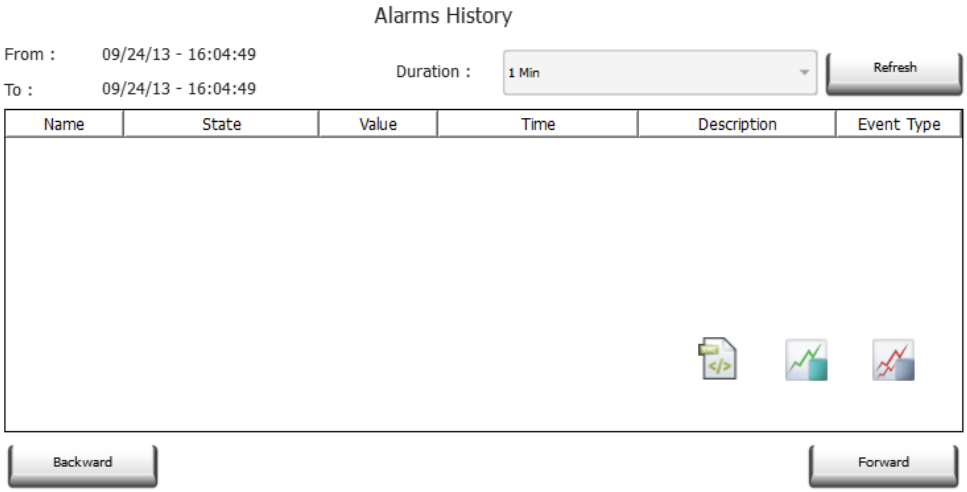
WARNING: If you are using an older converted project, you have to substitute the old Active Alarms Widget with the new one from the Widgets gallery

Note: The image can be modified from the Colums property of the Active Alarms widget



Alarms History widget

Logs and display an alarm list if **Buffer** property in Alarms Configuration Editor is set.



Attaching widget to buffer

Path: **AlarmHistory** widget> **Properties** pane> **Buffer** > **EventBuffer**

In **Properties** pane > **Event** select the **Event Buffer** from which the alarm list is retrieved

Managing alarms at run time

When an alarm is triggered it is displayed in the Active Alarms widget where you can acknowledge and reset it. You can filter the alarms displayed using several filters, for example you can hide not triggered alarms or show all alarms.

See "Active Alarms widget" on page 170 for details.

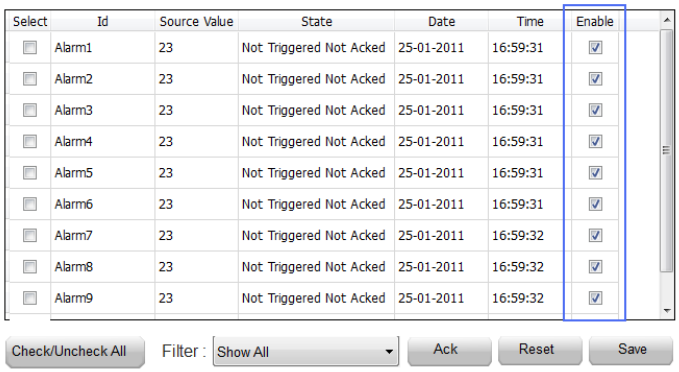
IMPORTANT: The Active Alarms widget is not displayed automatically. You must add a dedicated action that will open the page containing the alarm widget when the alarm is triggered.

Enable/disable alarms at run time

You can enable or disable the alarms at run time.

To enable an alarm select the **Enable** option in the alarm widget.

Disabled alarms are not triggered and therefore not displayed at run time.



Note: Alarms can be configured to be enable/disable even from the PLC.
See Alarm Configuration Editor for details.

Displaying live alarm data

Path: ***ProjectView*** *>* ***Config*** *> double-click* ***Alarms***

Both in the Active Alarms and in the History Alarms widget you can set the alarm description to display live tag data.

Id	Name	Enable	Ack	Reset	Tag	Buffer	Trigger	Action	Description
1	Alarm1	☑	☑	☑	Tag1	AlarmBuffer1	bitMaskAlarm:	ShowDialog	Alarm 1 Tag Value is [Tag1]
2	Alarm2	☑	☑	☑	Tag1	AlarmBuffer1	bitMaskAlarm:1	ShowDialog	Alarm 2 Tag Value is [Tag2]
3	Alarm3	☑	☑	☑	Tag1	AlarmBuffer1	bitMaskAlarm:1	ShowDialog	Alarm 3 Tag Value is [Tag3]

To show the tag value, set a placeholder in **Description** entering the tag name in square brackets, for example "[Tag1]". At run time, in **Description** column of Active Alarms widget the current value of the tag will be displayed. In History Alarms widget or in .csv file the value at the time the alarm was triggered is displayed

Use '\' before '[ ]' if you want to show the '[ ]' in the description string, for example: *\[Tag\[1\]\]* will display the string *"[Tag[1]]"*.

Use '\', even when the tag label contains square brackets. For example, to display the live tag value of tag “TAG]3” or “TAG[3]” use:

- TAG\]3 = **[**TAG\]3**]**
- TAG\[3\] = **[**TAG\[3\]**]**

Example of Alarm widget

Select	Id	Source Value	State	Description	Date
☐	Alarm1	123	Triggered Not Acked	Alarm 1 Tag value is 123	25-01-2011
☐	Alarm2	1234	Triggered Not Acked	Alarm 2 Tag value is 1234	25-01-2011
☐	Alarm3	456	Triggered Not Acked	Alarm 3 Tag value is 456	25-01-2011
☐	Alarm4	987	Triggered Not Acked	Alarm 4 Tag value is 987	25-01-2011
☐	Alarm5	555	Triggered Not Acked		25-01-2011
☐	Alarm6	1234	Triggered Not Acked		25-01-2011
☐	Alarm7	1234	Triggered Not Acked		25-01-2011

Check/Uncheck All Filter : Hide Not Triggered Ack Reset Save

Note: The csv file resulting from the dump of the alarm events list will also display the tag value in the description column.

Exporting alarm buffers to .csv files

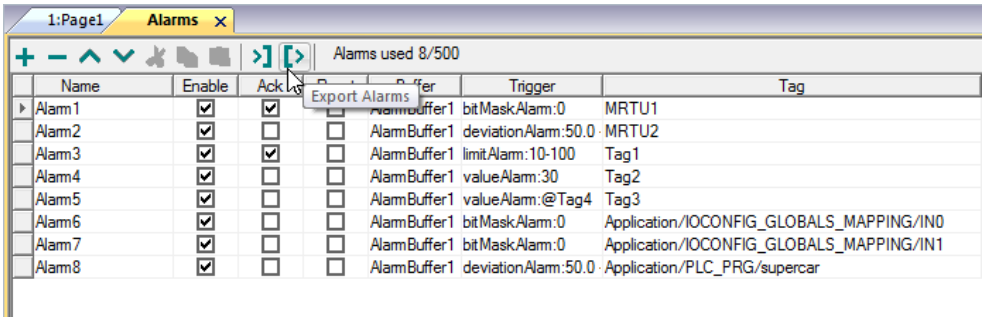
To export an event buffer containing an history alarms list, use the **DumpEventArchive** action.

See "System actions" on page 134 for details.

Note: Tag values displayed in the alarms description are also included in the buffer. Tags are sampled when the alarm is triggered and that value is logged and included in the description.

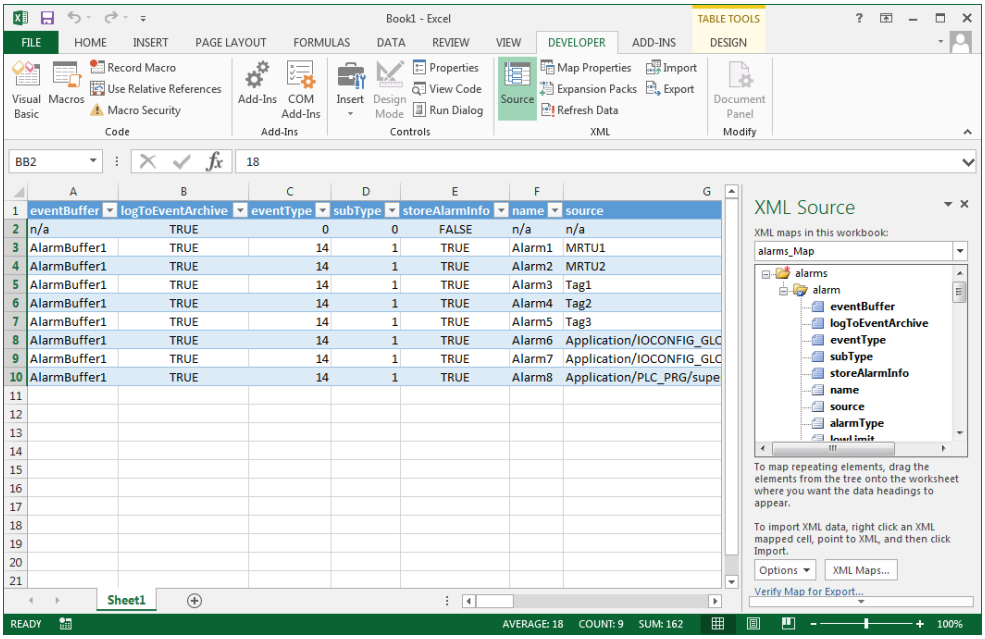
Exporting alarm configuration

Path: ***ProjectView*** *>* ***Config*** *> double-click* ***Alarms***



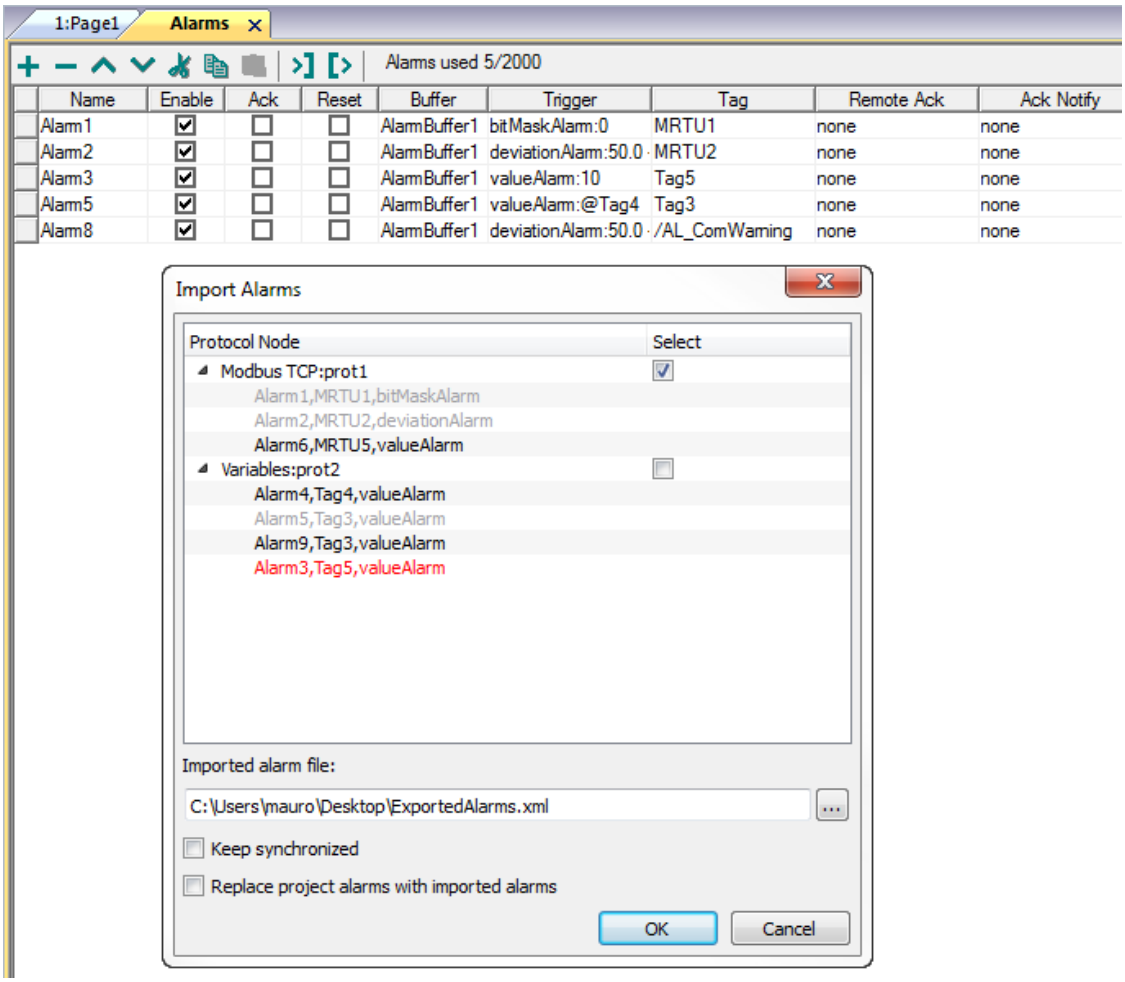
Click the **Export Alarms** button: the alarms configuration table is exported into an .xml file.

You can edit the resulting .xml file using third part tools (for example, Microsoft Excel) .



Importing alarm configuration

Path: ***ProjectView*** *>* ***Config*** *> double-click* ***Alarms***



1. Click the **Import Alarms** button and select the .xml file from which to import the alarms configuration: the **Import Alarms** dialog is displayed.
2. Select the group of alarms to import and click **OK** to confirm.

Differences are highlighted in the **Import Alarms** dialog using different colors

Color	Description
Black	This is a new alarm and it will be imported
Red	This alarm has not been found and will be removed (only if check "Replace project alarms with imported alarms" is checked)
Blue	This alarm has been modified and will be updated.
Gray	This alarm is already part of the project and will be skipped.

Automatic synchronization

Select the **Keep synchronized** option in the **Import Alarms** dialog to enable the automatic synchronization of the alarm configuration file.

Whenever changes occur in the alarms configuration, the file will be automatically updated in silent mode.

Tip: Enable this function when the alarm file is managed by a different tool (for example, PLC programming software) as well as by Wizard.

17 Recipes

Recipes are collections of tag values organized in sets that satisfy specific application requirements.

For example, if you have to control room variables (temperature and humidity) in the morning, afternoon and evening. You will create three sets (morning, afternoon and evening) in which you will set the proper tag values.

Each element of the recipe is associated to a tag and can be indexed into sets for a more effective use. This feature allows you to extend the capabilities of controllers that have limited memory.

You can add controller data to a page using a recipe widget. Recipe data contains all the controller data items; however data is no longer read directly from the controller but rather from the associated recipe element in the HMI device.

Recipe data is configured in Wizard workspace; the user can specify default values for each element of the data records. In HMI Runtime, data can be edited and saved to a new data file, any change to recipe data is therefore stored to disk. With the use of a separate data file HMI Runtime ensures that modified recipe values are retained throughout different project updates. In other words, a subsequent project update does not influence the recipe data modified by the user in the HMI Runtime.

See "Recipe actions" on page 129 for details on how to reset recipe data.

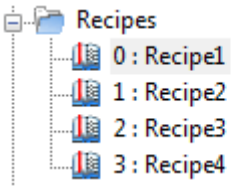
Note: Recipe data can be stored on a Flash memory, on a USB drive or on a SD card.

Managing recipes

Creating a recipe

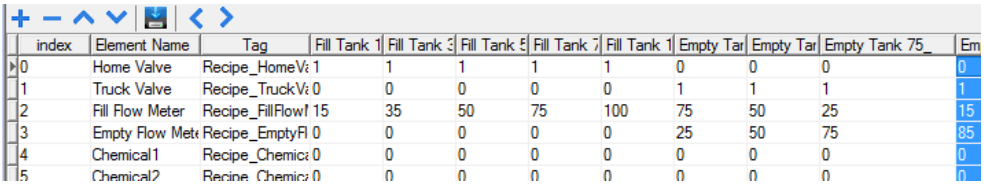
To create a recipe for your project:

1. In **ProjectView** right-click **Recipes** and select **Insert Recipe**: an empty recipe is added. You create and configure recipes using the Recipe Editor.



Recipe editor

Path: ***ProjectView*** *>* ***Recipes*** *> double-click* ***RecipeName***

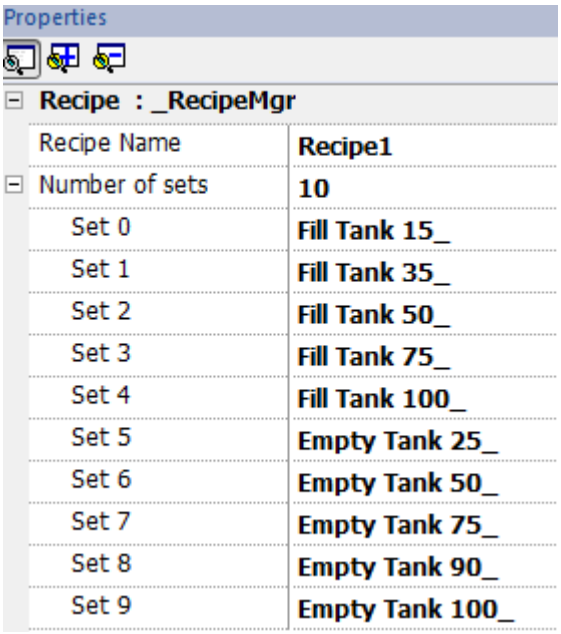


index	Element Name	Tag	Fill Tank 1	Fill Tank 3	Fill Tank 5	Fill Tank 7	Fill Tank 1	Empty Tar	Empty Tar	Empty Tank 75_	Em
0	Home Valve	Recipe_HomeVa	1	1	1	1	1	0	0	0	0
1	Truck Valve	Recipe_TruckVa	0	0	0	0	0	1	1	1	1
2	Fill Flow Meter	Recipe_FillFlowI	15	35	50	75	100	75	50	25	15
3	Empty Flow Met	Recipe_EmptyFl	0	0	0	0	0	25	50	75	85
4	Chemical1	Recipe_Chemica	0	0	0	0	0	0	0	0	0
5	Chemical2	Recipe_Chemica	0	0	0	0	0	0	0	0	0

Configuring recipe properties

In the **Properties** pane of each recipe you set the following parameters:

Parameter	Description
Recipe Name	Name of the recipe
Number of sets	Number of values sets for each recipe element. Each set has a different configurable name.

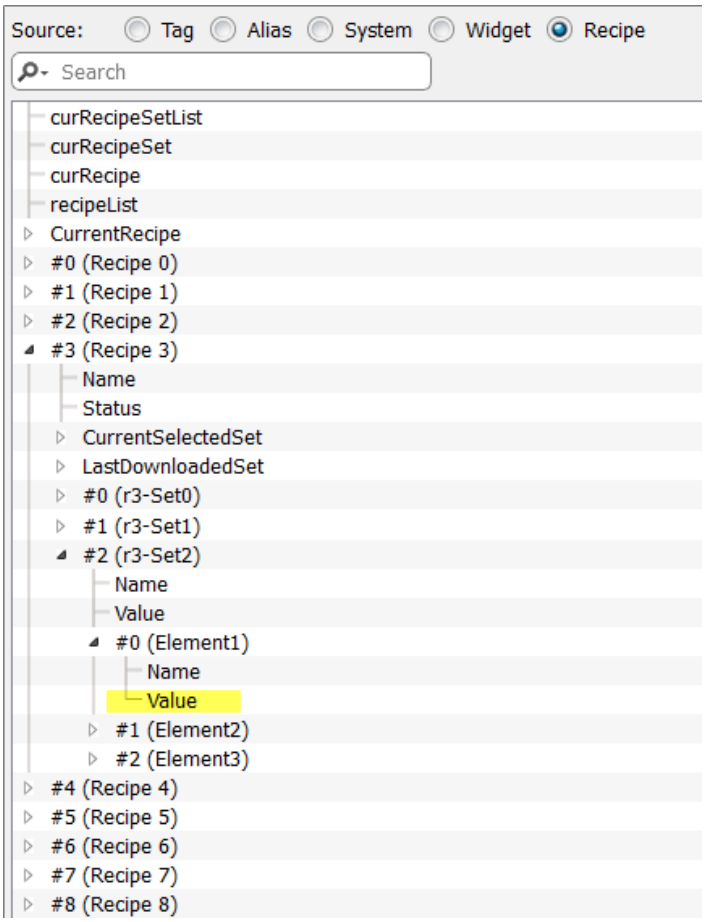


Setting up a recipe

1. Click **+** to add an element of the recipe.
2. Link the tags to each recipe element.

Defining recipe fields

Create a recipe field in the page using a numeric widget and attaching it to a recipe item after selecting Recipe as the Source.



In the **Attach to** dialog you have the choice of all the different recipe variables, such as:

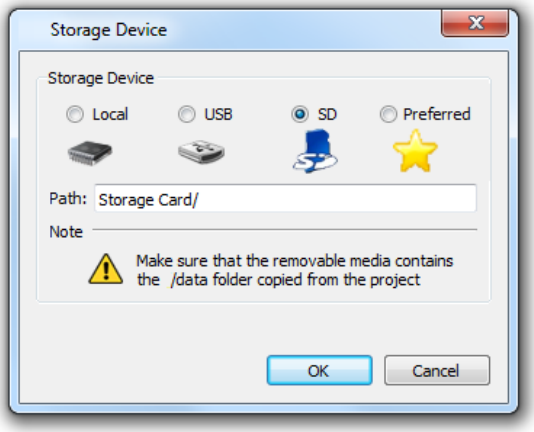
- Current Recipe >Current Selected Recipe Set> Element > Value
- Selected Recipe > Selected Set0 > Element > Value
- recipeList

When numeric widgets are defined as read/write, the default recipe data can be edited at run time. These new values are stored in a separate file as modified recipe data.

Note: Since JavaScript API functions are used, the recipe elements and sets can be referenced by name or by position. To avoid ambiguity between names and index, the names of the recipe elements and sets must include at least one alphanumeric character.

Storing recipe data

In the Recipe Editor click the storage type icon to select where to store recipe data: the **Storage Device** dialog is displayed.



For USB drive and SD card storage you can provide the folder location.

WARNING: Recipe configuration files are created automatically when the project is saved and stored in the data subfolder of the project. To use external storage devices, you need to copy this folder into the external device. Note that you have the responsibility to manage the data folder inside external devices. Even dynamic files are not deleted when project is updated using the "Delete dynamic file" option.

Important: You can add a subfolder but you must not rename the "data" subfolder.

Configuring a recipe widget

You can choose one of the two recipe widgets available in the **Widget Gallery**:

- **Recipe set**: allows you to select a recipe set for upload or download. See "Uploading/downloading a recipe" on the facing page
- **Recipe menu**: when more recipes have been created for a project, use this widget to manage all recipes and select the desired sets for each of them.

Configuring the Recipe Set widget

In the **Properties** pane of each **Recipe Set** widget set the following parameter:

Parameter	Description
Recipe Name	Name of the recipe

Recipe status

After every recipe upload or download, or recipe set modification, the recipe **Status** parameters contain a value with the result of the operation.

Code	Function	Description
0	Set modified	Selected set changed
1	Download triggered	Download request triggered
2	Download Done	Download action completed
3	Download Error	Error during download (for example, unknown set, unknown recipe, controller not ready, Tags write failed etc.)
4	Upload triggered	Upload request triggered
5	Upload done	Upload action completed
6	Upload Error	Error during upload - same as for download
7	General Error	General error (for example, data not available)

Note: On device startup the value of recipe **Status** is 0.

Uploading/downloading a recipe

Uploading a recipe

You upload a recipe to an HMI device using a recipe widget and the **UpLoadRecipe**, **UpLoadCurRecipe** action in one of the following ways:

- attach the action to an event of a button or a switch (see ""Attach to" parameters" on page 32 for details)
- configure the action in an alarm action list (see "Alarm actions" on page 118 for details)
- configure the action in a scheduler action list (see "Scheduling events at run time" on page 224 for details)

Downloading a recipe

You download a recipe from an HMI device using a recipe widget and the **DownloadRecipe**, **DownLoadCurRecipe** action. See "Recipe actions" on page 129

Backup and restore recipes data

The recipe data stored in an HMI device can be exported for backup and later restored. This is done using the **DumpRecipeData** or the **RestoreRecipeData** actions.

See "Recipe actions" on page 129 for details.

18 Trends

Trends allow you to sample and record the values of specified tags according to specific sampling conditions. The trend function includes trend acquisition and trend display.

Trend acquisition parameters are set in the Trend editor so that data can be stored. Stored data can then be displayed in a graphical format using a trend widget.

Data logging

Data can be logged and stored to HMI memory. Data logging allows you to store the values of a group of tags all at the same time to a buffer. Data logging can be triggered by a timer or by a dedicated tag. Logged data can be exported to a .csv file or displayed using the historical trend widget. Logged data can be saved locally on a USB device or SD card, or on any available custom network folder.

WARNING: The operation with removable memory devices (USB Flash drives, SD memory cards) containing a very large number of files may result in a decrease of system performance.

WARNING: The max number of files inside a SD memory card depends on the type of formatting (e.g. FAT32 max 65536 files; FAT max 513 files).

WARNING: Flash cards support a limited number of write operations. We suggest to use only good quality memory cards; in the case your application use intensively the memory card consider a regular substitution of the memory card.

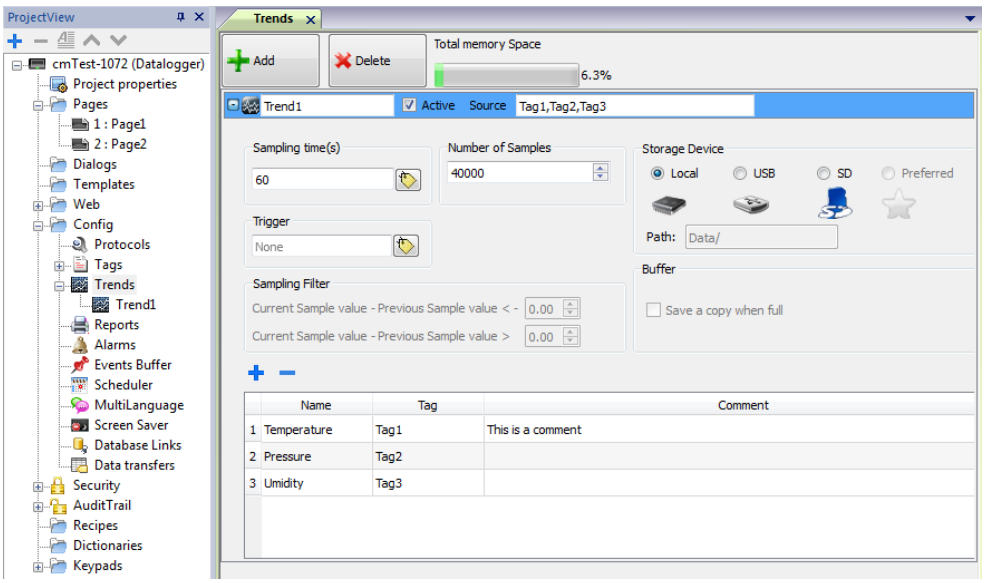
WARNING: If the data/time is moved back, the samples with invalid date/time are removed from the trend buffer. When system detects that data/time is invalid (e.g. battery low), a popup is shown to advise the user and the date/time of the last sample is used to avoid losing data.

Storage is based on trend buffers. Trend buffers are organized as a FIFO queue: when the buffer is full, the oldest values are discarded unless you configure your trend to create a backup copy of the buffer.

Adding a trend buffer

*Path: **ProjectView**> **Config** > double-click **Trends***

1. Click **Add** to add a new buffer.
2. Click **+** next to each trend buffer to display all configuration parameters.



Element	Description
Total memory Space	Memory currently used by the trend buffer. See "Table of functions and limits" on page 428 for maximum number of samples allowed for project. This percentage is calculated as follows: $Total\ Memory\ Space = \frac{Total\ Number\ of\ Samples\ used\ in\ the\ Project}{Max\ Number\ of\ Samples\ allowed\ for\ a\ Project} * 100$
Trend Name	Name of trend that will be displayed in the window property pane.
Active	When enabled, the trend runs by default at system startup. Note: Trends cannot be activated at run time.
Source	Tags sampled by the trend.
Sampling Time (s)	Sampling interval in seconds.
Trigger	Tag triggering the sample. When the value of this tag changes, a sample is collected. Note: Trigger and Source can refer to the same tag.
Number of Samples	Buffer size.
Storage Device	Where trend buffer data will be stored.
Buffer	If **Save a copy when full** option is enabled, a backup copy of the buffer data is created before it is overwritten by newer data.
Sampling Filter / Trigger Filter	If triggering condition is time, a new sample is stored when its value, compared with the last saved value, exceeds the specified limits. If triggering condition is a tag, a new sample is stored at each change of the trigger tag value.
Sampled tags table	**Name**: name of trend **Tag**: tag to be sampled. **Comment**: trend description

Exporting trend buffer data

Use the **DumpTrend** action to export trend buffer data to a .csv file.

Format of trend data exported to a .csv file can be selected from a macro parameter as shown in figure. All tags specified in the trend buffer are exported

Dump normal mode (compatibility mode)

	A	B	C	D	E	F	G	H	I	J	K
1	Type	Value	Time Stamp	Refresh Time	Quality	Type	Value	Quality	Type	Value	Quality
2	4	0	2015-09-18T14:42:22.000Z	1000	192	8	0.00E+00	192	3	0	192
3	4	0	2015-09-18T14:42:23.000Z	1000	192	8	0.00E+00	192	3	0	192
4	4	0	2015-09-18T14:42:24.000Z	1000	192	8	0.00E+00	192	3	0	192
5	4	40	2015-09-18T14:42:25.000Z	1000	192	8	0.00E+00	192	3	0	192
6	4	40	2015-09-18T14:42:26.000Z	1000	192	8	0.00E+00	192	3	0	192
7	4	40	2015-09-18T14:42:27.000Z	1000	192	8	0.00E+00	192	3	0	192
8	4	40	2015-09-18T14:42:28.000Z	1000	192	8	5.00E+01	192	3	0	192
9	4	40	2015-09-18T14:42:29.000Z	1000	192	8	5.00E+01	192	3	0	192
10	4	40	2015-09-18T14:42:30.000Z	1000	192	8	5.00E+01	192	3	0	192

Dump extended mode (compact mode)

	A	B	C	D	E	F	G
1	Timestamp	Tag1	4	Tag2	8	Tag3	3
2		Value	Quality	Value	Quality	Value	Quality
3	2015-09-18T14:42:22.000Z	0	192	0.00E+00	192	0	192
4	2015-09-18T14:42:23.000Z	0	192	0.00E+00	192	0	192
5	2015-09-18T14:42:24.000Z	0	192	0.00E+00	192	0	192
6	2015-09-18T14:42:25.000Z	40	192	0.00E+00	192	0	192
7	2015-09-18T14:42:26.000Z	40	192	0.00E+00	192	0	192
8	2015-09-18T14:42:27.000Z	40	192	0.00E+00	192	0	192
9	2015-09-18T14:42:28.000Z	40	192	5.00E+01	192	0	192
10	2015-09-18T14:42:29.000Z	40	192	5.00E+01	192	0	192

Note: The first row of the header contains the tags names and tags data types

See "System actions" on page 134 for details.

Trend widgets

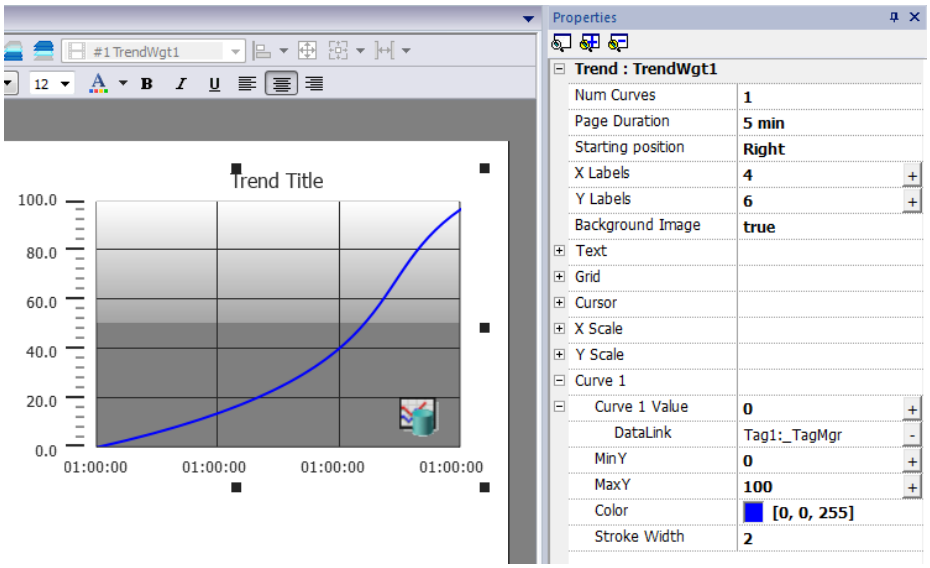
Data logged by the HMI device can be displayed in graphical format using trend widgets.

RealTime trend widget

The real- time trend widget can be used to display the changes of value of a tag. Data is not stored in a trend buffer and cannot be retrieved for later analysis.

To display a real-time trend:

1. Drag and drop the **RealTime Trend** widget from the widget gallery to the page.



2. Attach the tag that you want to sample to the **Curve *n* Value**. Data is always plotted against time.

RealTime trend widget properties

Property	Description
Num Curves	Number of trend curves to be displayed (Max. 5)
Page Duration	Time range of the x-axis. Tip: You can attach a **Date Time** combo widget to the **Page Duration** property and dynamically change page duration at run time.
Starting Position	Specifies the starting point of the curve when the page is opened.
X Labels	Number of ticks on the x-axis scale
Y Labels	Number of ticks in the y-axis scale.
Text	Trend title and font properties (font size, label, etc.)
Grid	Properties of grid presentation (colors)
Cursor	Properties of cursor presentation (enable and color)
X Scale	Properties of X Scale presentation
Y Scale	Properties of Y Scale presentation
Curve "n"	Tag that will be plotted in the trend widget. See "Trend widget properties" on page 193 for details. You can set the minimum and maximum of the curves (**MinY**, **MaxY**). You can attach a tag to minimum and maximum properties. This enhances the ability to change the minimum and maximum values dynamically at run time.

Scaling data

Tag values can be scaled using the X Forms in the **Attach to** dialog. See ""Attach to" parameters" on page 32 for details.

History trends

Trend data stored in trend buffers can be analyzed using the **History Trend** widget.

This is a two-step process:

- first you create a trend buffer to collect data for specified tags at specific points in time,
- then you configure a History Trend widget to display the collected data in a graphical format.

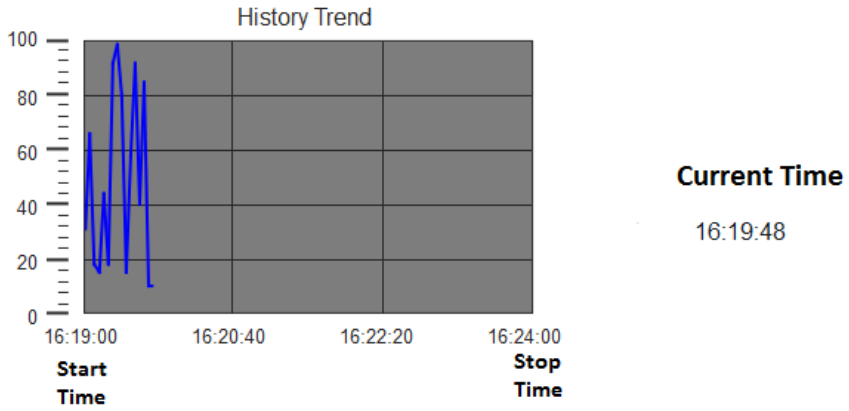
See "Data logging" on page 188 for details on how to create a trend buffer.

History Trend widget

History Trend widget displays in graphical format the content of a trend buffer.

Start time is the current time and stop time will be the current time plus the duration of the window. The curve starts from the left and progresses to the right, data is automatically refreshed during a certain interval time, until the stop time.

When the curve reaches the stop time, the curve will scroll left and the update of the curve will continue until it again reaches the stop time. At that moment a new scroll is automatically performed and the process repeats.

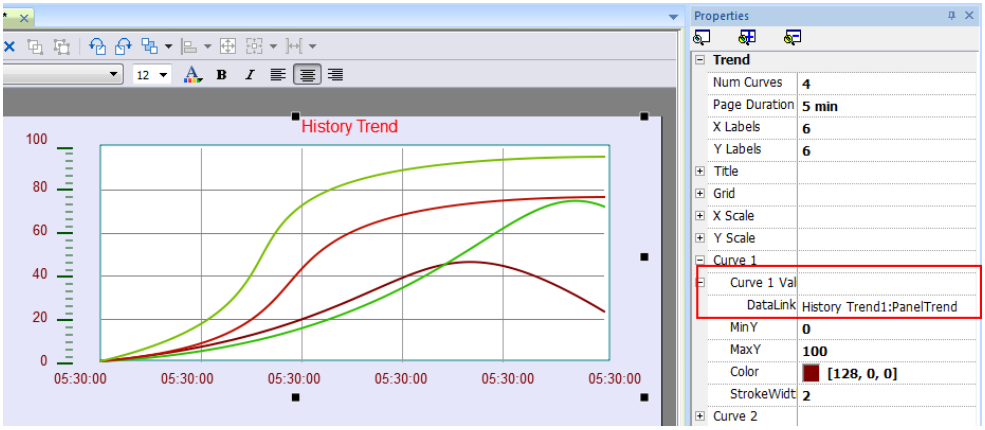


History trends require a proper configuration of trend buffer.

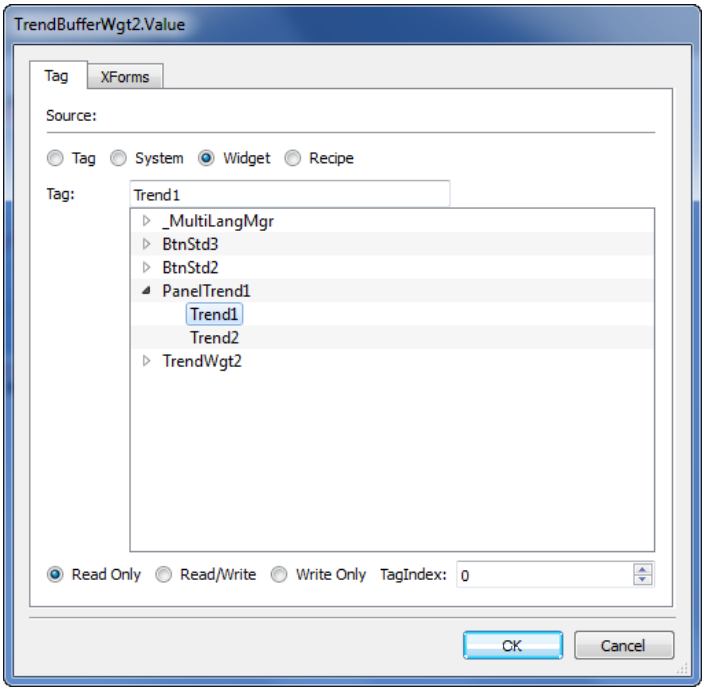
See "Data logging" on page 188 for details on how to work in the Trend editor.

Configuring the History Trend widget

1. From the **Trends/Graphs** section of the **Widget Gallery**, drag and drop the **History Trend** widget to the page.



2. In the **Properties** pane, attach the trend buffer to be plotted in the widget.



Trend widget properties

Some Trend widget properties are only available when the Properties pane is in Advanced view.

Request Samples

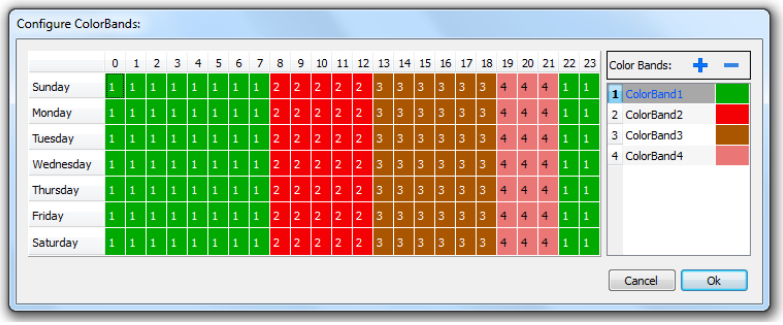
Request Sample property can be set for each curve and indicates the maximum numbers of samples read by the widget at one time from the trend buffer.

Tip: You normally do not need to modify the default value. Adjust it to fine tune performances in the trend widget refresh, especially when working with remote clients.

Color bands

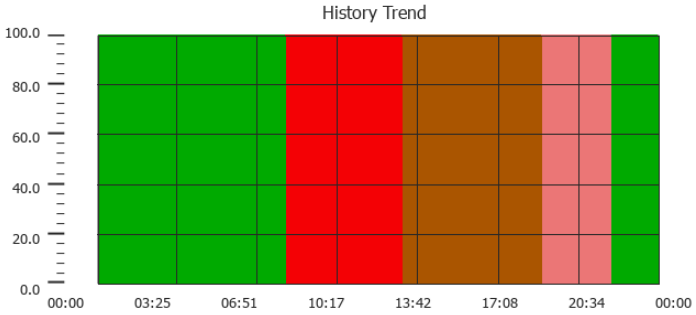
Use the color bands configuration to customize your graphs background, for example to make certain days or hours stand out (weekends, night hours, etc.).

1. In the **Properties** pane, in **Color Bands** property click **+**: the **Configure Bands** window appears.
2. Click **+** to add as many colors you need.
3. Select multiple cells and click on a color band to assign the color to the selected range of cells.



Note: This feature only uses local time in the trend widget, not the global time option.

Calendar color bands example



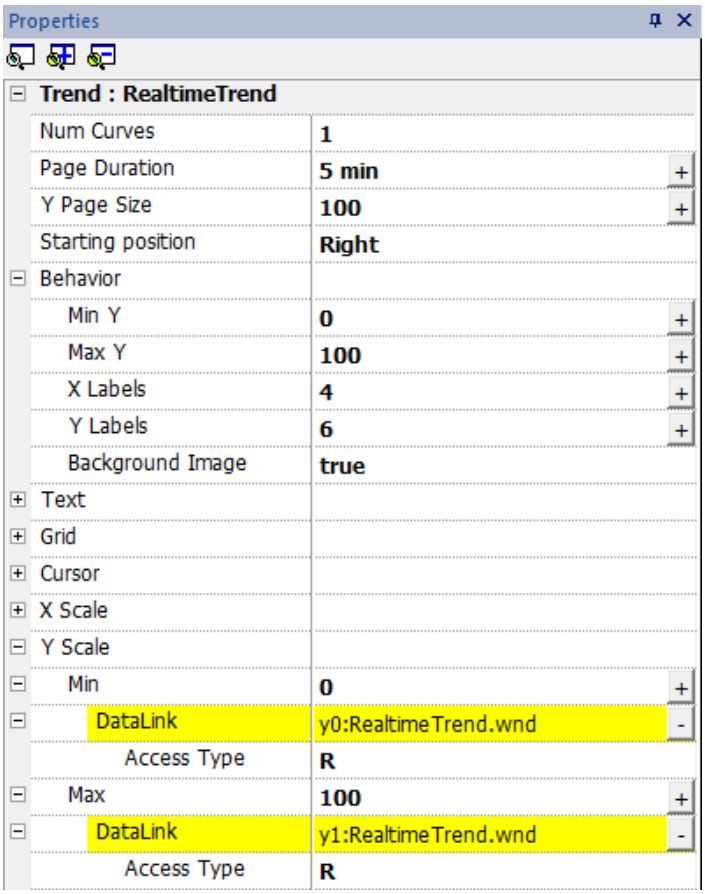
Trend widget gestures

Trend widgets support gesture commands:

Gesture	Description
pan	Touch the widget to scroll the curve within the widget area
pinch	Use two fingers to pinch the curve and perform zoom operations

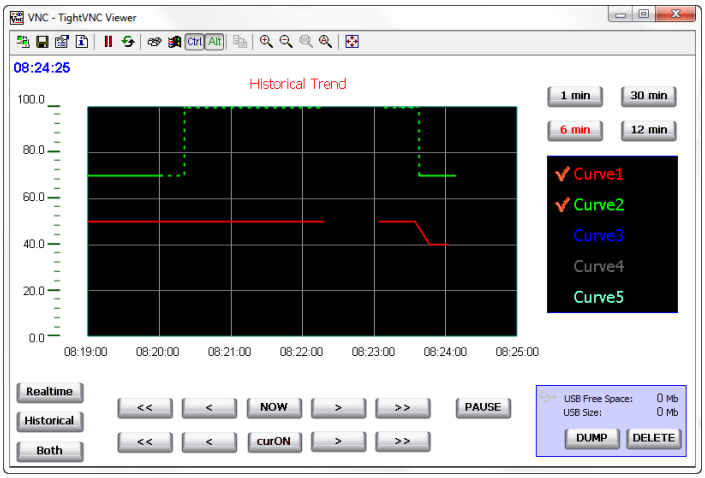
WARNING: Only multi touch HMI devices can generate pinch events

Note: In order to support gestures on Y axis, Min/Max properties of the trend widget must be linked to Min/Max values of Behavior parameters (default for new trends).



Values outside range or invalid

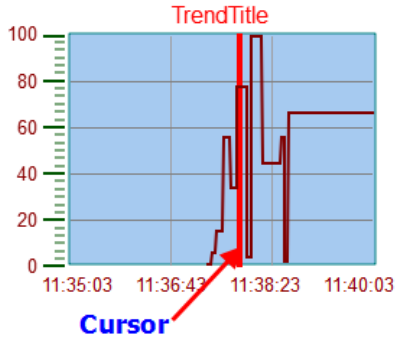
When trend value goes beyond the limits set for the trend widget, a dotted line is displayed. When the value of the tag is not available, for example the controller device is offline, no curve is drawn.



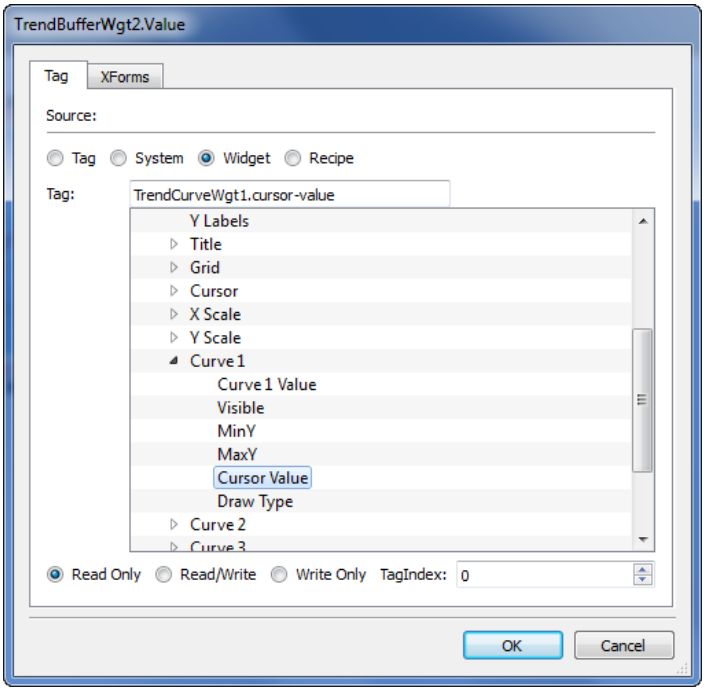
Showing trend values

Trend cursor displays the trend value at a specific point.

Use the actions **ShowTrendCursor** and **ScrollTrendCursor** to enable the trend cursor and move it to the required point to get the value of the curve at that particular point in time.

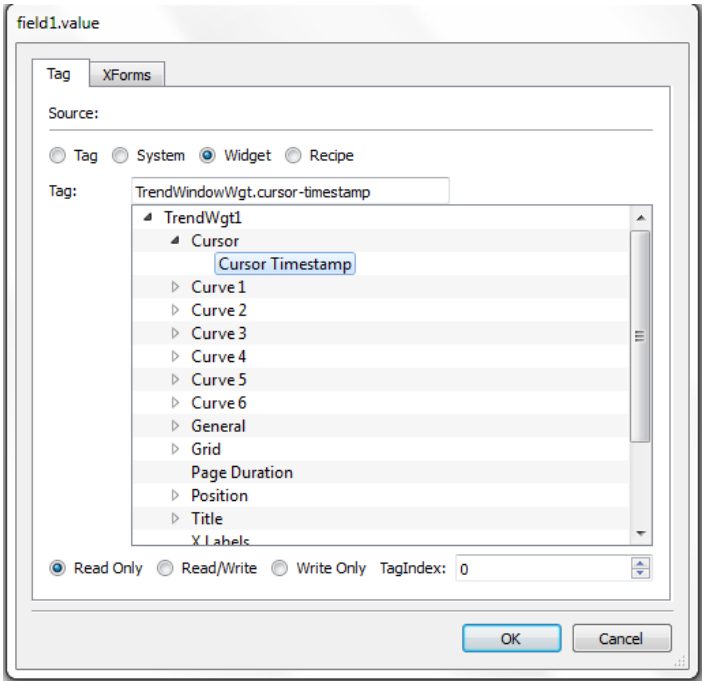


To display the value of the trend cursor on the page, define a numeric field and attach it to the **Cursor Value** widget tag.



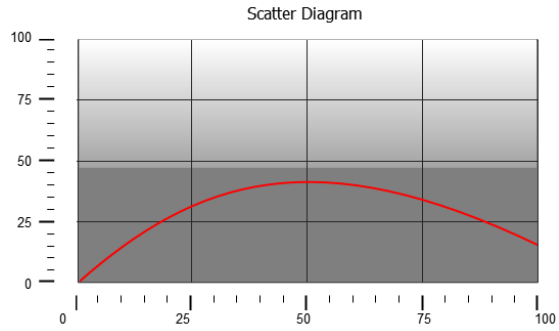
In this example the Y axis value of the cursor is displayed.

To display the trend timestamp at the position of the cursor, define a numeric field and attach it to **Cursor Timestamp** widget tag.



Scatter diagram widget

A scatter diagram is a type of diagram to display values for two variables from a set of data using Cartesian coordinates. The data is displayed as a collection of points, each having the value of one variable determining the position on the horizontal axis and the value of the other variable determining the position on the vertical axis. For this reason it is often called *XY graph*.



Scatter diagram curves are obtained by a linear interpolation of points. To create a new scatter diagram:

1. Add a **Scatter Diagram** widget to the page.
2. Select the number of curves to show: each curve is named as Graph1, Graph2,...
3. Customize the general graph properties such as **X Min**, **X Max**, **Grid** details.
4. Define the max number of samples/values for each curve by setting the **Max Samples** parameter.

   Here you set the max number of values to be displayed in the graph starting from first element in the array.

   For example: Tag1[20] and Max Samples = 10 will show just first 10 elements of the Tag1 array.
5. Define for each curve the two tags of type array to be displayed (**X-Tag** and **Y-Tag**).

When the array tags change, you can force a refresh with the **RefreshTrend** action .

Note: Scatter diagrams support only the **RefreshTrend** action.

19 Data transfer

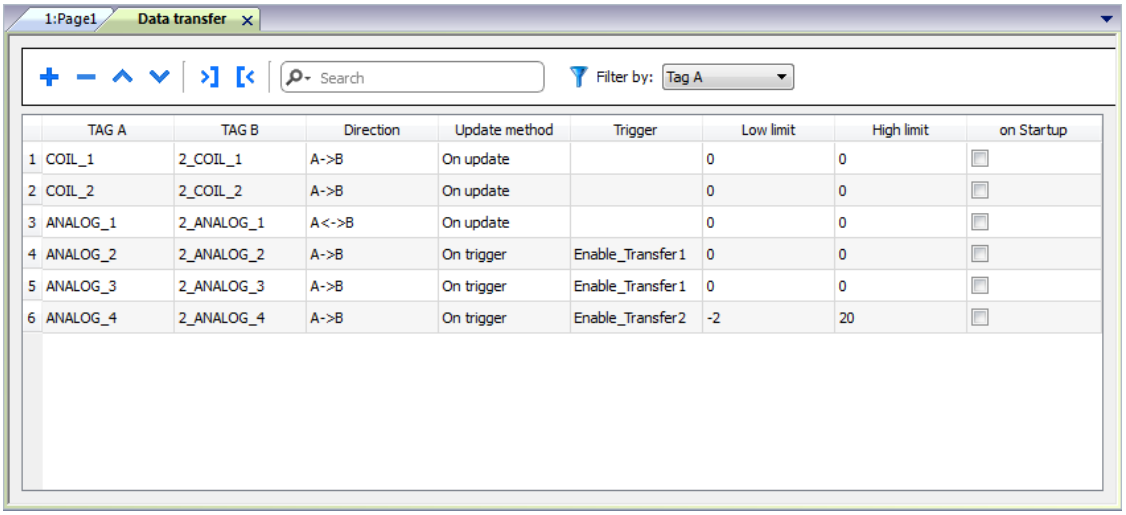
Data transfer allows you transferring variable data from one device to another. Using this feature an HMI device can operate as a gateway between two devices, even if they do not use the same communication protocol.

Data transfer editor

Path: ***ProjectView*** *>* ***Config*** *> double-click* ***Data transfer***

Use the Data transfer editor to map transfer rules.

Each line in the Data transfer editor defines a mapping rule between two tags. Define more mapping rules if you need different direction, update method or trigger.



	TAG A	TAG B	Direction	Update method	Trigger	Low limit	High limit	on Startup
1	COIL_1	2_COIL_1	A->B	On update		0	0	
2	COIL_2	2_COIL_2	A->B	On update		0	0	
3	ANALOG_1	2_ANALOG_1	A<->B	On update		0	0	
4	ANALOG_2	2_ANALOG_2	A->B	On trigger	Enable_Transfer1	0	0	
5	ANALOG_3	2_ANALOG_3	A->B	On trigger	Enable_Transfer1	0	0	
6	ANALOG_4	2_ANALOG_4	A->B	On trigger	Enable_Transfer2	-2	20	

To add a new rule, click **+**: a new tag line is added.

Data transfer toolbar

Prameter	Description
Import/ Export	Imports or exports data transfer settings from or to a .csv file.
Search	Displays only rows containing the search keyword.
Filter by	Display only rows matching filter and search field.

Data transfer parameters

Prameter	Description
TAG A/ TAG B	Pair of tags to be mapped for exchanging through the HMI device.
Direction	Transfer direction. **A->B** and **B->A**: Unidirectional transfers, values are always copied from one tag and sent to the other tag in the specified direction. **A<->B**: Bidirectional transfer, values are transferred to and from both tags.
Update Method	**On trigger**: Data transfer occurs when the value of the tag set as trigger changes above or below the values set as boundaries. Limits are recalculated on the previous tag value, the same that triggered the update.

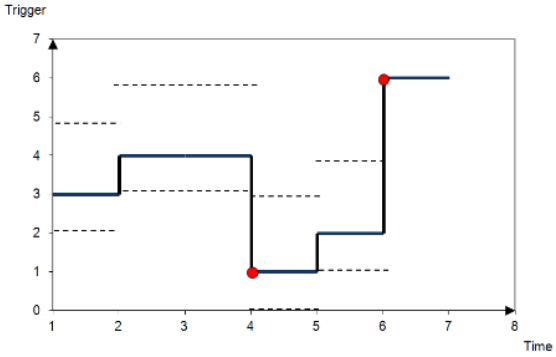
Prameter	Description
	Note: This method applies only to unidirectional transfers (A->B or B->A). **On Update**: Data transfer occurs whenever the value of the source tag changes. Note: This method applies both to unidirectional and to bidirectional transfers (A->B, B->A and A<->B). Note: The Runtime cyclically monitors source tags changes (trigger tag when using On Trigger or tags to transfer when using On Update) based on Tag editor **Rate** parameter. If **Rate** setting for source Tag is 500 ms (default), the system checks for updates every 500 ms. Note: Changes on source tags faster than **Rate** may be not detected .
Trigger, High limit, Low limit	Tag that triggers the data transfer process. When this tag changes its value outside the boundaries set as **High limit** and **Low limit**, data transfer is started. The range of tolerance is recalculated according to the specified limits on the tag value which triggered the previous update. No action is taken if the change falls within the limits. This mechanism allows triggering data transfers only when significant variations of the reference values occur. **Low limit** is less or equal to zero. Note: If both **Low limit** and **High limit** are set to "0", data transfer occurs whenever the value of the trigger tag changes.
on Startup	When selected, data transfer is forced: • on HMI startup if the quality of the source tag is good • after communication errors, when the associate device nodes return active See "Objects" on page 340 for details on quality. **Important: Data transfers executed on startup may have major impact on the HMI device boot time. Enable this option only when necessary.**

Example of limit setting

High limit = 1,9

Low limit = - 0,9

• = points where the data transfer is triggered



Exporting data to .csv files

Configuration information for data transfers can be exported to a .csv file.

Example of data transfer settings in .csv file

A	B	C	D	E	F	G	H	I	J
COIL_1	2_COIL_1	A->B	On update		0	0	data1	true	1
COIL_2	2_COIL_2	A->B	On update		0	0	data2	true	1
ANALOG_1	2_ANALOG_1	A<->B	On update		0	0	data3	true	1
ANALOG_2	2_ANALOG_2	A->B	On trigger	Enable_Transfer1	0	0	data4	true	1
ANALOG_3	2_ANALOG_3	B->A	On trigger	Enable_Transfer1	0	0	data5	true	1
ANALOG_4	2_ANALOG_4	A->B	On trigger	Enable_Transfer2	-10	20	data6	true	1

Column	Description
A to G	Same data as in the Data transfer editor
H	Unique identifier automatically associated to each line. **Important: When you edit the .csv file and you add any extra line, make sure you enter a unique identifier in this column.**
I and J	Reserved for future use.

Import/export use the separator character defined inside Windows Regional Settings.

Data transfer limitations and suggestions

Correct definition of data transfer rules is critical for the good performance of the HMI devices. To guarantee reliability of operation and performance, keep in mind the following rules.

On trigger method

The **On trigger** method allows only unidirectional transfers, (A->B or B->A)

Data transfer based on the **On Trigger** mode should be preferred since it allows you to force the transfer and monitors only the trigger tags and not all the tags involved in the transfer.

On update method

The **On update** method allows changing the values in accordance with the direction settings only when the source value changes.

Using the **On Update** method you force the system to continuously read all the defined source tags to check if there are changes that need to be transferred. The default value of the update rate of each tag is 500 ms and can be modified with Tag editor.

Performance observations

Data transfer performance depends on:

- number of data transfers defined,
- number of data transfers eventually occurring at the same time,
- frequency of the changes of the PLC variables that are monitored,

Important: Always test performance of operation during project development.

Important: If inappropriately set, data transfer tasks can lead to conditions where the tags involved create loops. Identify and avoid such conditions.

Tip: Use the scheduler to calibrate the update rate based on the performance of your entire project.

Tip: Use array type tags to optimize data transfer and reduce workload.

Tip: Reduce the number of data transfers to reduce page change time and boot time.

20 Offline node management

When one of the controllers communicating with the HMI device goes offline, communication performance of the system may eventually decrease.

The offline node management feature recognizes offline controllers and removes them from communication until they come back online.

Additionally, if you know that any of the controllers included in the installation is going to be offline for a certain time, you can manually disable it to maximize system performance.

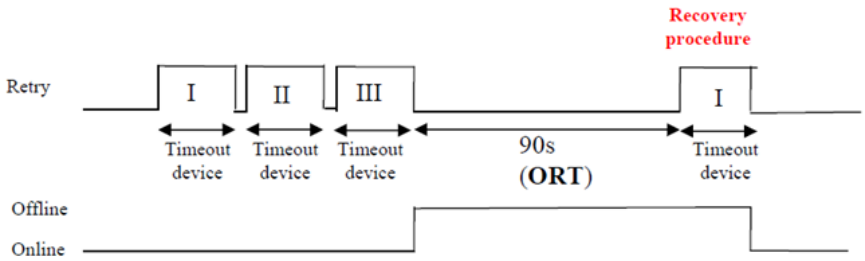
Note: This feature is not supported by all communication protocols. Check protocol documentation to know if it is supported or not.

Offline node management process

Steps of the process are:

- The system communicates normally with a certain device. When the device is not responding to a communication request, the system will repeat the request twice before declaring the device offline.
- When a device is offline, the system sends communication requests to the device with a longer interval, called Offline Retry Timeout. If the device answers to one of these requests, the system declares it online and restarts normal communication.

The diagram shows the three communication attempts and the recovery procedure that starts when the Offline Retry Timeout is elapsed.



Manual offline node management process

Offline node management can be done manually. When a specific device is online and it is communicating normally you can:

- use an action to declare the device offline: the system stops communication with the device.
- use an action to declare the device online: the system restarts normal communication with the device.

Manual offline configuration

When you know that some devices in communication with the HMI device are going to remain offline for a certain period of time, you can exclude them from communication using the **EnableNode** action.

WARNING: All disabled device nodes will remain disabled if the same project is downloaded on the device, on the other hand, if a different project is downloaded, all disabled devices will be re-enabled. The same happens with a package update.

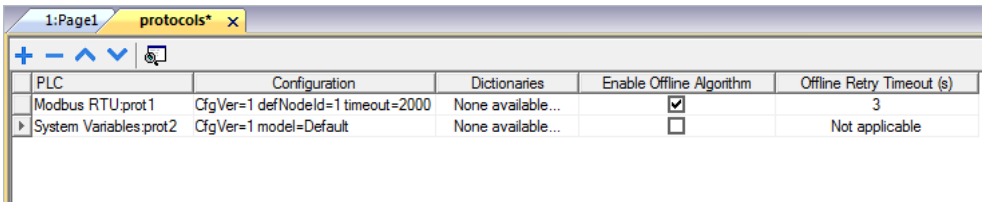
Tip: To make this feature more dynamic, you may decide not to indicate a specific **NodeID** but attach it to the value of a tag or to an internal variable created to identify different devices that might be installed in your network.

Note: When using the action **EnableNode** to force a device node back online, communication will start immediately.

Automatic offline node detection

When a device is not answering to communication requests, it is de-activated. The HMI device stops sending requests to this device. After three seconds, the HMI device sends a single command to check if device is available, if so the communication is restarted, otherwise it is disabled for another timeout interval.

Default settings can be modified in Protocol editor.



Note: Not all protocols support this feature.

Parameter	Description
Enable Offline Algorithm	Enables offline management for the protocol
Offline Retry Timeout	Interval in seconds for the retry cycle after a device has been deactivated. Range: 1–86.400 seconds (24h).

21 Multi-language

Multi-language feature has been designed for creating HMI applications that include texts in more than one language at the same time

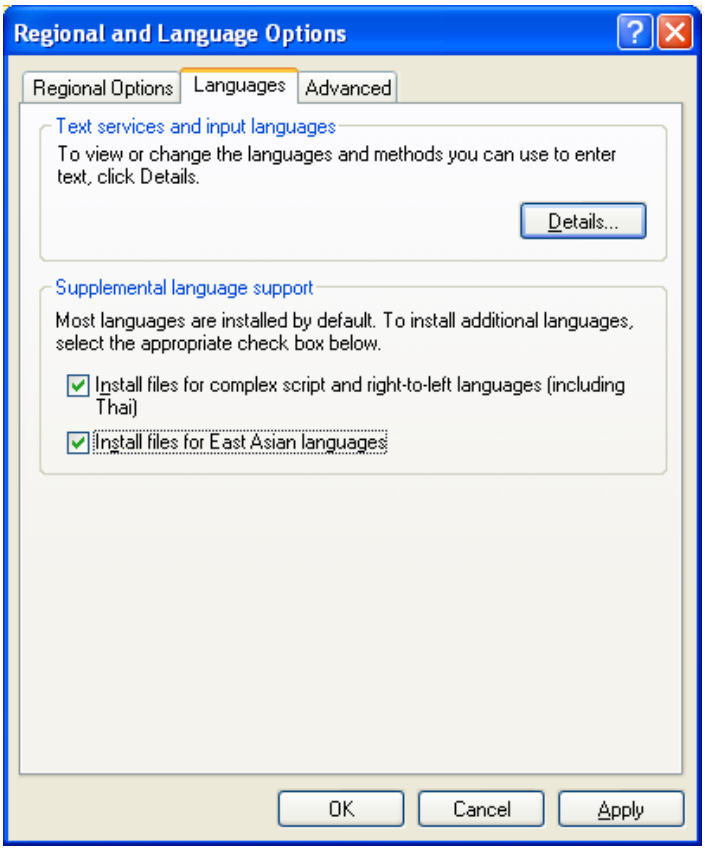
Multi-language feature uses code pages support to handle the different languages. A code page (or a script file) is a collection of letter shapes used inside each language.

Multi-language feature can be used to define languages and character sets in a project. Wizard also extends the TrueType Fonts provided by Windows systems to provide different font faces associated with different character sets.

Wizard also allows you to provide strings for each of the languages supported.

Wizard also allows you to change the display language so that you can see the page look and feel during the design phase.

Important: In Windows XP operating systems you have to install the support for complex script and East Asian languages.



Supported fonts for Simplified Chinese

For Simplified Chinese, the following fonts are supported:

Font name	Font file
Fangsong	simfang.ttf
Arial Unicode MS	ARIALUNI.TTF
Kaiti	simkai.ttf
Microsoft Yahei	msyh.ttf
NSImsun	simsun.ttc
SimHei	simhei.ttf
Simsun	simsun.ttc

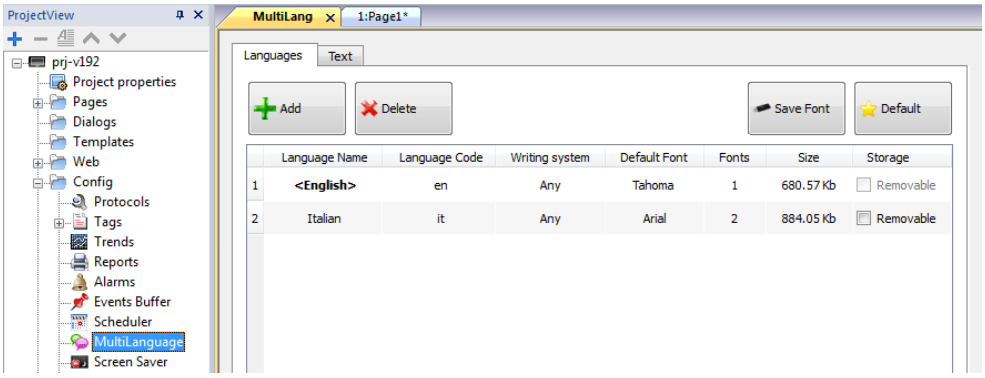
Supported fonts for Traditional Chinese

For Traditional Chinese, the following fonts are supported:

Font name	Font file
DFKai-SB	kaiu.ttf
Microsoft Sheng Hai	msjh.ttf
Arial Unicode MS	ARIALUNI.TTF
MingLiU	mingliu.ttc
PMingLiU	mingliu.ttc
MingLiU_HKSCS	mingliu.ttc

The Multi-language editor

Path: ***ProjectView*** *>* ***Config*** *>* *double-click* ***MultiLanguage***



Language settings

Parameter	Description
Language Name	Name identifying the language in the project.
Language Code	ISO 639 language code identifier, used to match language items when importing resources from external xml files.
Writing system	Select the set of fonts to be used with the language
Default Font	Default font for project's widgets. Note: When you choose a new font you are prompted to replace the font used in the widgets you already created.
Fonts	Number of fonts associated with the selected language.
Size	Memory used to store font files.
Storage	Location of file fonts is a removable external memory. Tip: Store large font files on removable memory to free memory requirements in the HMI device.

Adding a language

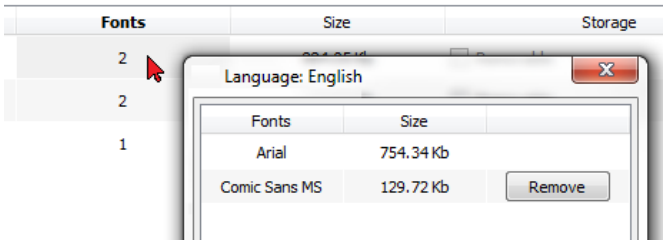
1. In the **Languages** tab, click **+**: a line is added to the table.
2. Enter all language settings.
3. Click **Default** to set the selected language as the default language when the Runtime starts.
4. Click **Save Font** to copy the fonts you marked as **Removable** on an external memory.

Important: Font files configured to be stored on removable memory must be provided to the final user to complete font installation on the HMI device.

Removing fonts

To remove fonts no longer needed:

1. Click on the font number in the Multi-language editor: a dialog with the list of the used fonts is displayed.

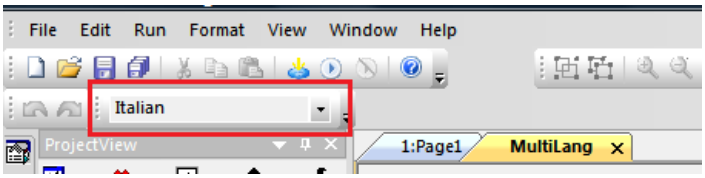


2. Select the fonts to be removed and click **Remove**: removed fonts are replaced with the default font.

Changing language

Changing language during page design

A combo box is available for changing language during page design. If no texts appears, please check **Text** tab in the Multilanguage editor and insert missing string.

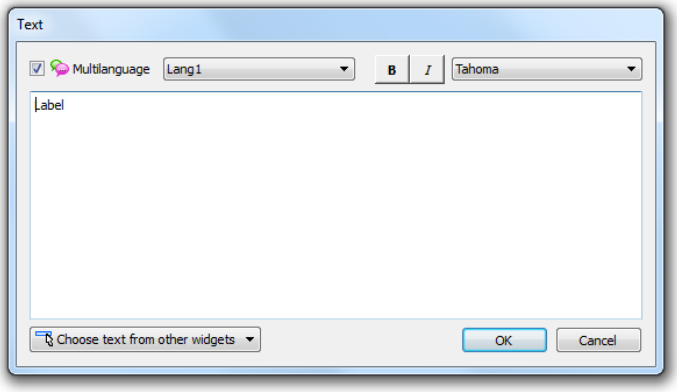


Multi-language widgets

Multi-language support is available for objects such as buttons, static text, messages, alarm descriptions and pop-up messages.

Multi-language for label widgets

Double-click on a text widget in a page to open the **Text** dialog.



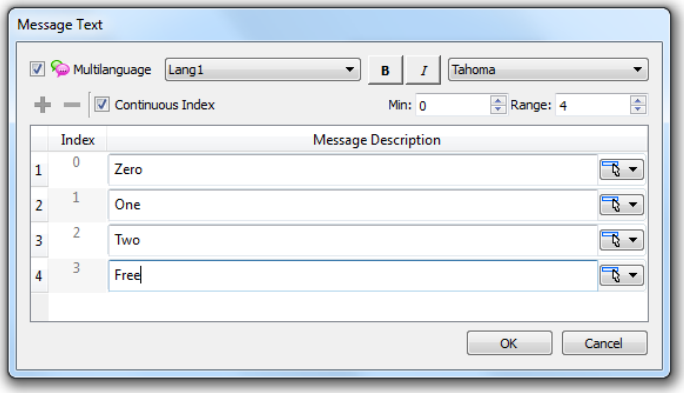
Enable/disable multi-language function, edit the text for the selected language and choose the font.

Note: Bold, italic and color properties set here for the widget are applied to all languages .

Parameter	Description
Multilanguage	Enable/disable multi-language function for the widget.
Choose text from other widget	Click on button to browse existing message strings in project to pick text for the widget.

Multi-language for message widgets

Double-click on a message widget in a page to open the **Message Text** dialog.

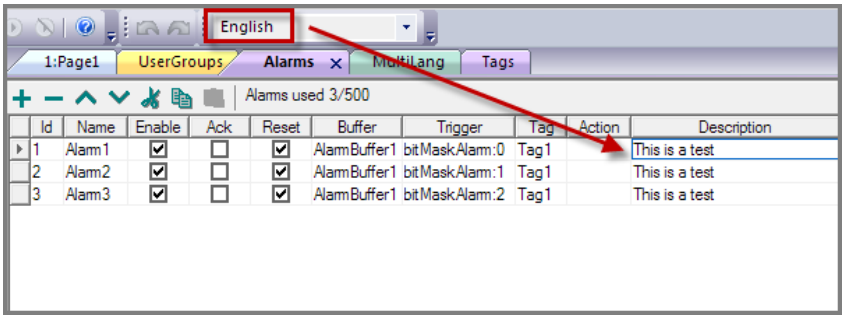


Parameter	Description
Multilanguage	Enable/disable multi-language function for the widget.
Continuous Index	Index for the widget is set of contiguous numbers (example 3, 4,5,6)
Min	Starting number for index
Range	Number of messages
Choose text from other widget	Click on button to browse existing message strings in project to pick text for the widget.

Multi-language for alarm messages

To add a multi-language strings for alarm messages:

1. Open the Alarm editor.
2. Select a language using the language combo box.
3. Enter the text for the alarm in the **Description** column.

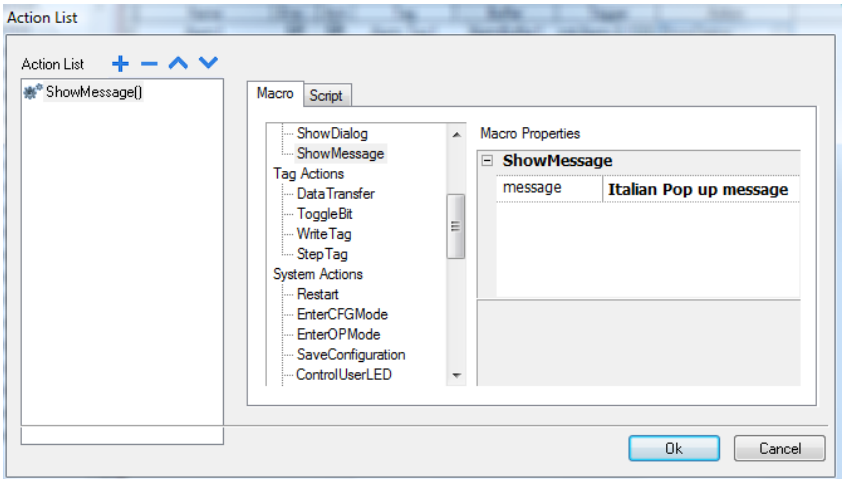


Tip: Text labels with alarm states displayed by alarms widgets can be translated or personalized through the Multilanguage text editor.

Multi-Language for pop-up messages

To add a multi-language pop-up message:

1. Select a language from the language combo box.
2. Add the Page action **ShowMessage** and enter the text in the selected language.



Exporting/importing multi-language strings

The easiest way to translate a project into multiple languages is to export all texts to a .csv file, translate the resulting document and then import the translated text back into the project.

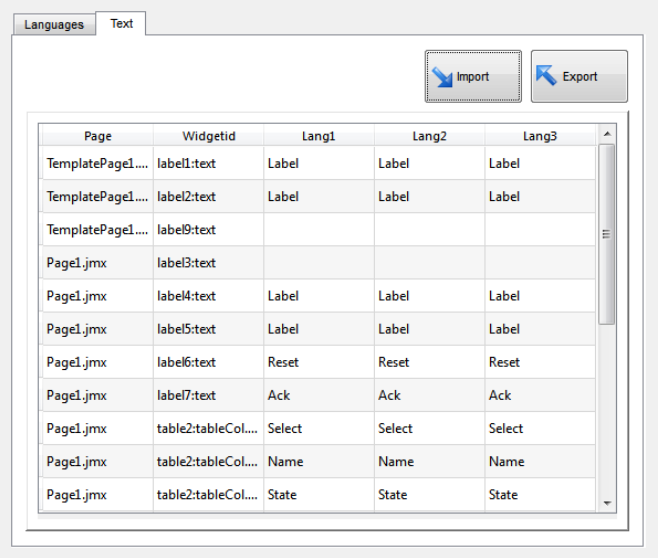
Important: The .csv file exported by Wizard is coded in Unicode, to edit it you need a specific tool supporting Unicode encoded .csv files.

Exporting and reimporting strings

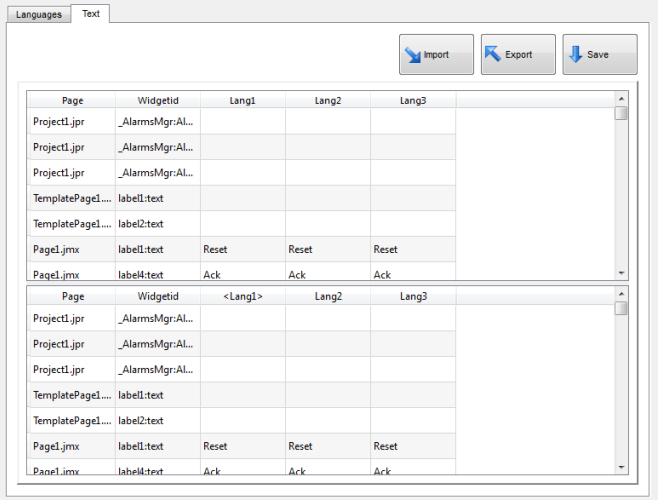
Path: ***ProjectView*** *>* ***Config*** *> double-click* ***MultiLanguage***

To export and re-import multi-language strings:

1. In the **Text** tab, click **Export**: all multi-language strings are exported to a .csv file.



Important: Set all languages that will be used in the project before exporting the file. This will guarantee that the exported file will contain all columns and language definitions.



2. Once the strings have been translated, click **Import** to re-import them into the project: strings are imported matching the widget ID and the page number of each widget.
3. Click **Save** to save the new widget data.

Note: To change the separator used in the exported file, change the regional settings of your computer. When importing, the separator information is retrieved from the file; if not found, the default character "," is used.

Import constraints

The following formats are supported for import:

- Comma Separated Values (.csv)
- Unicode Text (.txt)

Note: Use the Unicode Text file format when you import a file modified using Microsoft® Excel®.

Changing language at run time

Changing language with an action

After the project download, the HMI Runtime will start using the language set as default. You can change the language using the **SetLanguage** action. See "MultiLanguage actions" on page 119.

Note: Once the language has been changed, it will be used also in future sessions.

Missing fonts

When you change language, if the required fonts are not available in the device memory, a pop-up message prompts you to insert the memory card containing the missing fonts. At the end of the operation you can remove the memory card.

Limitations in Unicode support

Wizard has been designed for working with Unicode text. However, for compatibility issues with some platforms, Unicode is supported only in a subset of properties.

Area	Property	Charset Accepted	Reserved Chars/Strings
Protocol editor	Alias	ASCII [32..126]	(space) , ; : . < * >'
Tag editor	Name	ASCII [32..126]	. \ / * ? : > < | " & # %;=
	Group	ASCII [32..126]	<New> \ / * ? : > < | " & # % ;
	Comment	Unicode	

Area	Property	Charset Accepted	Reserved Chars/Strings
Trends	Name	ASCII [32..126]	\ / * ? : > < | " & # % ;
Printing Reports	Name	ASCII [32..126]	\ / * ? : > < | " & # % ;
Alarms	Name	ASCII [36..126]	\ / * ? : > < | " & # % ;
	Description	Unicode	[] - for live tags, \ escape seq for [and \
Events	Buffer Name	ASCII [32..126]	\ / * ? : > < | " & # % ;
Scheduler	Name	ASCII [32..126]	\ / * ? : > < | " & # % ;
Languages	Language Name	ASCII [32..126]	\ / * ? : > < | " & # % ;
	Texts in widgets	Unicode	-
	Texts from import files	Unicode	-
User Group	Group Name	a-z A-Z _	admin,guest,unauthorized
	Comments	Unicode	-
User	Name	ASCII [32..126]	\ / * ? : > < | " & # % ;
	Password	Unicode	-
	Comment	Unicode	-
Recipes	Name	ASCII [32..126]	\ / * ? : > < | " & # % ; ! $' ()+,=@[]{}~`
	Set Name	ASCII [32..126]	\ / * ? : > < | " & # % ; ! $' ()+,=@[]{}~`
	Element name	ASCII [32..126]	\ / * ? : > < | " & # % ; ! $' ()+,=@[]{}~`
General	Project Name	A-Z,a-z,0-9,-,_	“PUBLIC”, “readme”, “index.html”
	Page Name	A-Z,a-z,0-9,-,_	-
	Dialog Page Name	A-Z,a-z,0-9,-,_	-
	Template Page Name	A-Z,a-z,0-9,-,_	-
	Keypad Name	A-Z,a-z,0-9,-,_	-
	Files (Images/Video/etc..)	A-Z,a-z,0-9,-,_	-

Area	Property	Charset Accepted	Reserved Chars/Strings
	Widgets ID	A-Z,a-z,0-9,-,_	-
Runtime	PLC Communication	UTF-8, Latin1, UCS-2BE, UCS-2LE, UTF-16BE, UTF-16LE	-

22 Scheduler

Wizard provides a scheduler engine that can execute specific actions at set intervals, or on a time basis.

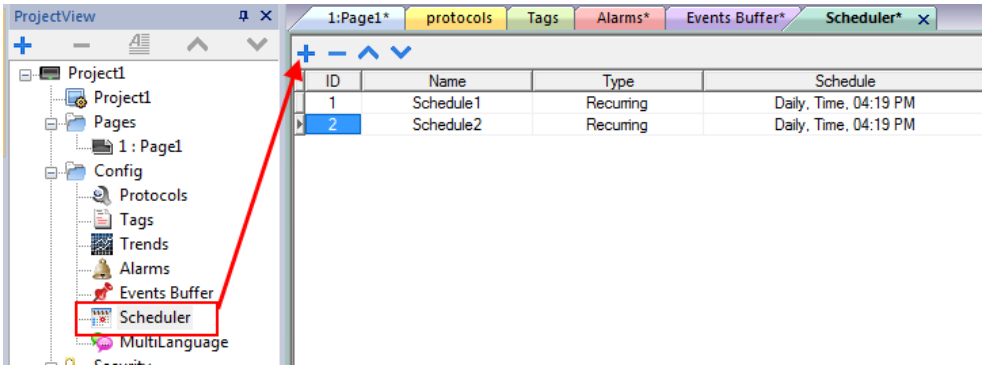
Creating a schedule is typically a two-step process:

1. You create a schedule with a list of actions to be executed when the scheduled event occurs. You do this in the Scheduler editor
2. You create a run-time user interface that allows the end-user to change settings for each schedule. You do this adding a **Scheduler** widget to a page of your project and configuring it to fit user scheduling needs.

Creating a schedule

Path: ***ProjectView*** *>* ***Config*** *> double-click* ***Scheduler***

- Click **+** to add a schedule.



Schedule parameters

Parameter	Description
ID	Unique code assigned automatically to the schedule
Name	Name of schedule
Type	Type of schedule: • **Recurring**, see "Recurring schedule" below for details. • **HighResolution**, see "HighResolution schedule" below for details
Schedule	Scheduler settings and options. See "Recurring schedule" below for details.
Action	Actions to be executed at the scheduled time
Priority	Priority level for the event. If two schedules occur at the same time, the event with the higher priority will be executed first.

HighResolution schedule

The **HighResolution** schedule is used to perform actions that need to be repeated at specified intervals. The interval between executions is set in milliseconds in the **Schedule** column.

Note: You cannot change at run time the settings of this type of schedule. If you need to change the action time settings at run time, choose **Recurring** schedule and set **Type** to **Every**. See "Recurring schedule" below for details.

Recurring schedule

The Recurring schedule is used to perform actions at specified points in time. Settings can be modified at run time.

Recurring scheduler parameters

Parameter	Description
Type	Frequency of the scheduled actions
Mode	Specific settings required by each scheduler type
Condition	Boolean tag (true/false) to activate the specified actions at the moment the timer is triggered. Actions will be executed if tag = true. By default, actions are executed when the timer is triggered.  Note: Only tags attached to the Boolean data type are shown.
Actions	Actions to be executed by the schedule.  **Important: Actions and schedule parameters cannot be modified at run time**
Date	Date when the scheduled actions will be executed
Time/Offset	This field display one of the following: **Time** = when the scheduled actions will be executed **Offset**= delay or advance with respect to the selected mode.
Location	Reference location to calculate sunset/sunrise time.
weekdays	Days of the week in which the scheduled actions will be executed.
On startup	Executes schedule at start up
Enable schedule	Enables/disables the schedule
Execute only at startup	Executes the schedule only once at start up

Schedule type options

Option	Description
By Date	Actions are executed on the specified date and time.
Daily	Actions are executed daily at the specified time.
Every	Actions are executed with the specified interval (Range: 1 s–1 day)
Hourly	Actions are executed every hour at the specified minute.
Monthly	Actions are executed every month at the specified date and time.

Option	Description
Weekly	Actions are executed every week on the specified weekday(s) and time.
Yearly	Actions are executed every year at the specified date and time.

Schedule mode options

Option	Description
Time	Depends on the schedule type. Allows you to specify date/time/week data.
Random10	Actions are executed in the time interval of 10 minutes before or after the set time. For example, if set time is 10:30, actions are executed any time between 10:20 and 10:40.
Random20	Actions are executed in the time interval of 20 minutes before or after the set time. For example, if set time is 10:30, actions are executed any time between 10:10 and 10:50.
Sunrise+	Actions are executed with a specified delay after sunrise. The delay is set in minutes/hours and sunrise time is location specific.
Sunrise-	Actions are executed with a specified advance before sunrise. The advance is set in minutes/hours and sunrise time is location specific.
Sunset+	Actions are executed with a specified delay after sunset. The delay is set in minutes/hours and sunset time is location specific.
Sunset-	Actions are executed with a specified advance before sunset. The advance is set in minutes/hours and sunset time is location specific.

See "Configuring location for schedules" below for details on sunset and sunrise settings.

Note: **Mode** options are not available for all schedule types.

Configuring location for schedules

Scheduled actions can be configured to be executed at a specific time with respect to sunrise and/or sunset. To do this you need to define the correct location, based on UTC information. The system will the automatically calculate the sunrise and sunset time.

Only a few locations are available by default. If your location is not listed, you can add it by entering latitude, longitude and UTC information in the Target_Location.xml file.

Important: Each platform has its own Target_Location.xml file.

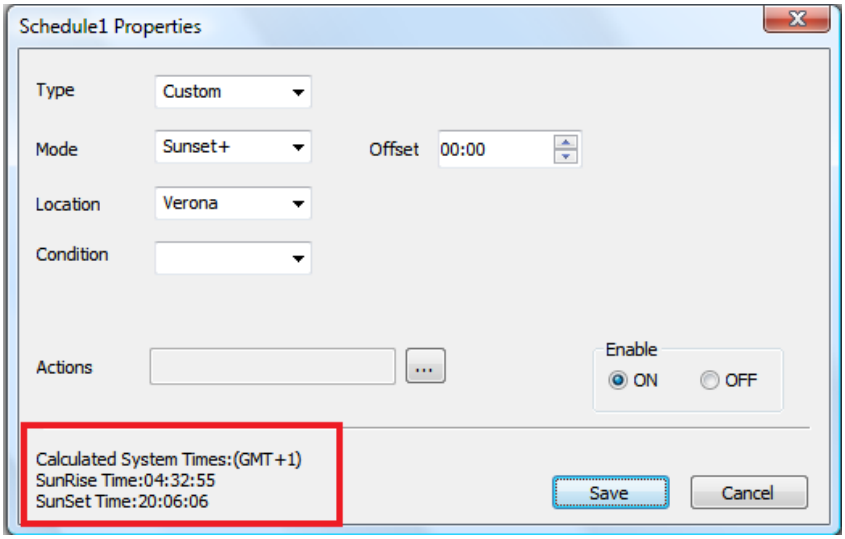
Location files position

Application	Location file path
Wizard	*CG\Wizard\languages\shared\studio\config\Target_Location.xml*
HMI Devices	*CG\Wizard\runtime\<HW Platform>\config\Target_Location.xml*
Simulator	*CG\Wizard\simulator\config\Target_Location.xml*
BTM-PCRUNTIME	*CG\Wizard\server\config\Target_Location.xml*

For example, the information for the city of Verona (IT) is shown below:

```
<file city="Verona" latitude="45.44" longitude="10.99" utc="1"/>
```

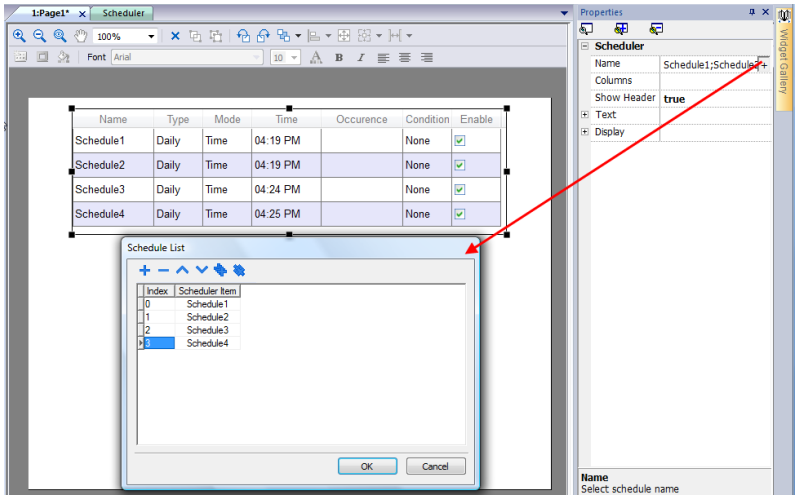
Location information is also displayed in the dialog together with sunset and sunrise times.



Configuring the Scheduler widget

To display scheduler data on a page:

1. Drag and drop a **Scheduler** widget from the widget gallery into the page.
2. In the **Properties** pane, click **+** for the **Name** parameter: the **Schedule List** dialog is displayed.
3. Add all the schedules you want to display in the page.



4. In the **Properties** pane, customize all settings.

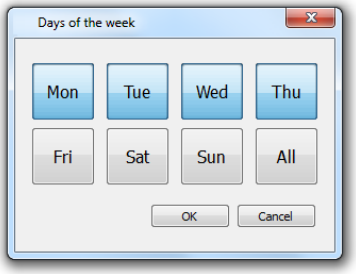
Scheduler settings

Parameter	Description
Name	Schedule to be displayed
Columns	Columns to be displayed and their characteristics
Show Header	Shows/hides column headers
Time Spec	Time to be displayed at run time
Text	Font used for text
Display	Table styles

Scheduling events at run time

At run time you can modify the following scheduling parameters.

Name	Type	Mode	Time	Occurence	Condition	Enable
Schedule1	By Date	Time	11:01	JUN 20,2013	None	☑
Schedule3	Monthly	Sunrise+	11:01	Day : 3	None	☑
Schedule4	Weekly	Rando...	16:19	M T W T F S S	None	☑
Schedule5	Yearly	Time	01:00			
Schedule6	Custom	Time	01:16			



Parameter	Description
Occurrence	Information on the type of schedule and time of execution
Condition	Condition applied to action execution
Enable	Enabels/disables the execution of the scheduled actions without deleting the schedule.

See "Recurring schedule" on page 220 for details on schedule parameters.

23 User management and passwords

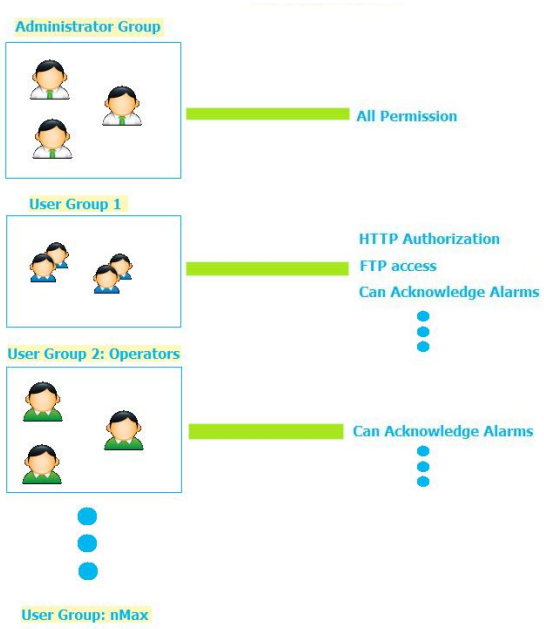
You can restrict access to various widgets and operations by configuring users, users groups and assigning specific authorizations to each group.

Each user must be member of one and only one group. Each group has specific authorizations and permissions.

Authorizations and permissions are divided in two categories:

- Widget permissions: hide, read only, full access
- Action permissions: allowed or not allowed.

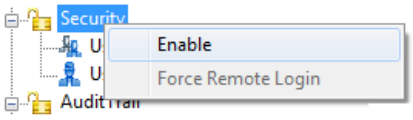
By organizing permissions and groups you can define the security options of a project.



Enable/disable security management

Path: ***ProjectView*** *>* *right-click* ***Security*** *>* ***Enable***

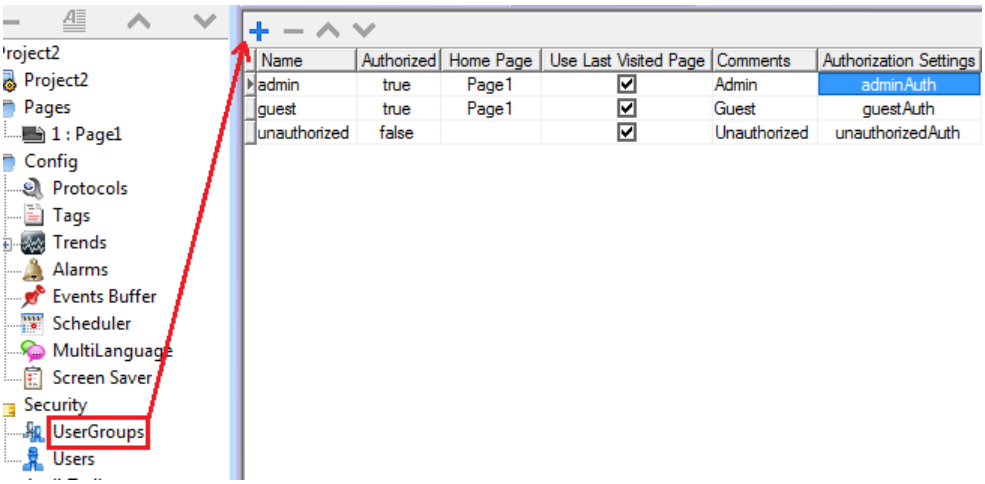
The padlock symbol indicates whether the function is enabled or disabled.



Important: Security settings are effective only if the security function is enabled.

Configuring groups and authorizations

Path: ***ProjectView*** *>* ***Security*** *>* *double-click* ***UserGroups***



Three predefined groups are available by default (**admin**, **guest** and **unauthorized**): they cannot be deleted nor renamed. You can, however, modify authorizations and other settings.

Adding a user group

Click **+** to add user group.

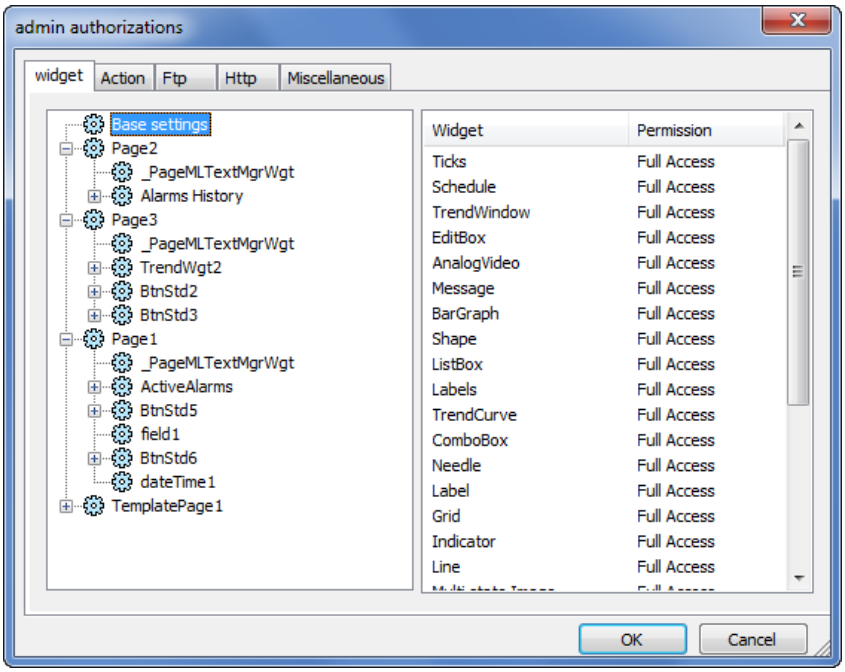
Parameter	Description
Name	Name of users group
Authorized	Authorization granted
Home Page	Page displayed when users belonging to this group log in
Use Last Visited Page	When selected, the last page displayed by the previous user will be displayed when users belonging to this group log in

Parameter	Description
Comments	Any comment or description for the group
Authorization Settings	Opens the Admin Authorization dialog to set access permissions. See "Modifying access permissions" below for details.

Modifying access permissions

Path: ***ProjectView*** *>* ***Security*** *> double-click* ***UserGroups*** *>* ***Authorization Settings*** *column*

Click the button: a dialog appears with a list of widgets and actions. You can modify access permissions for each one in the list.



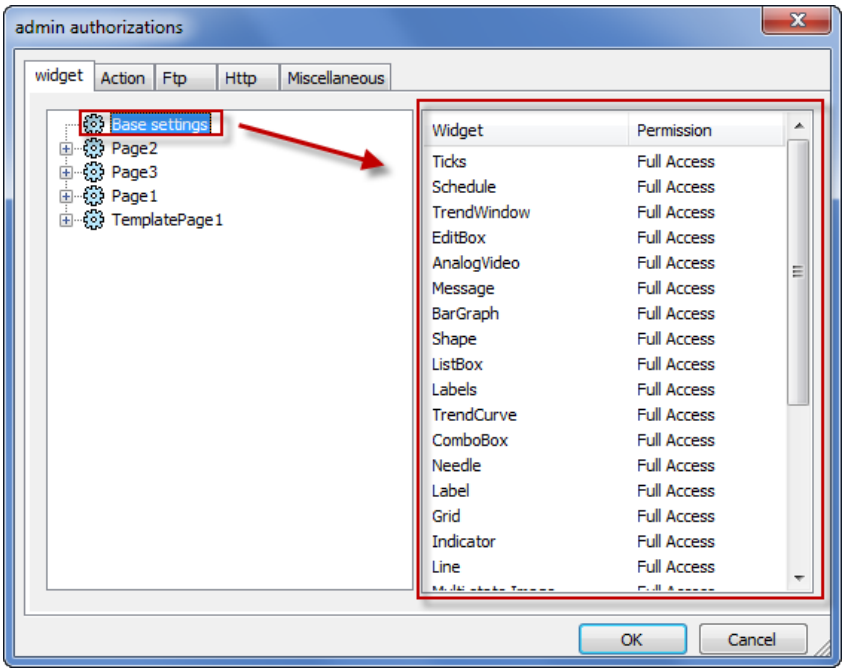
Widget permissions

In the **Widget** tab you can define widget access options at project level, at page level or at widget level for all the widgets used in the project. Lower levels permission (for example, widget level) overrides higher levels (that is, page and project levels).

Use **Base settings** to set default permissions at project level.

Possible settings are:

- **Full Access** to enable read/write access to the widget
- **Read Only** to enable readonly access to the widget
- **Hide** to hide widget for selected group



Changing a widget permission

To change access permission for an individual widget in a page of the project, navigate to that widget within its page on the right pane and customize its access options. Otherwise, all widgets take the permissions set at project or page level.

For example, if page permission for a widget is set at project level to **Read Only**, then all the same widgets will have permission **Read Only**. When you select a widget inside a page from the tree structure, permission is actually set to **Use Base Settings**. You can change this setting and modify access permissions only for this widget in this page.

Access priority

Widget permissions are considered with the following priority:

Permission level	Priority
Project level - Basic settings	Low
Page level	Medium
Widget level	High

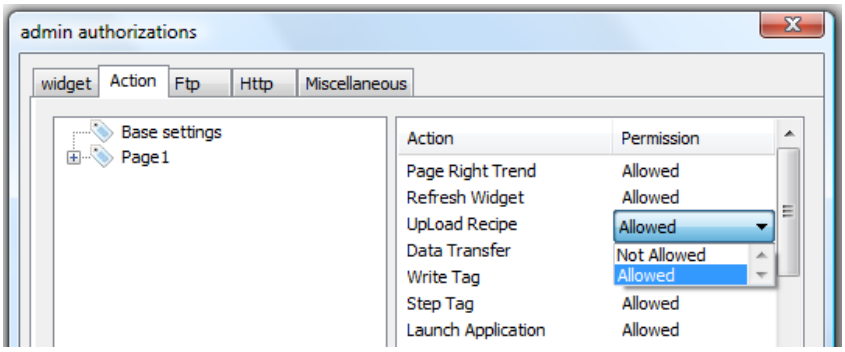
This allows you to specify exceptions for an action or a widget directly from the page view.

For example, if you set permissions for a widget at project level to Read Only and to Full Access at page level then the page level settings will prevail.

Access permissions can be modified directly from the project page. See "Assigning widget permissions from page view" on page 234 for details.

Action permissions

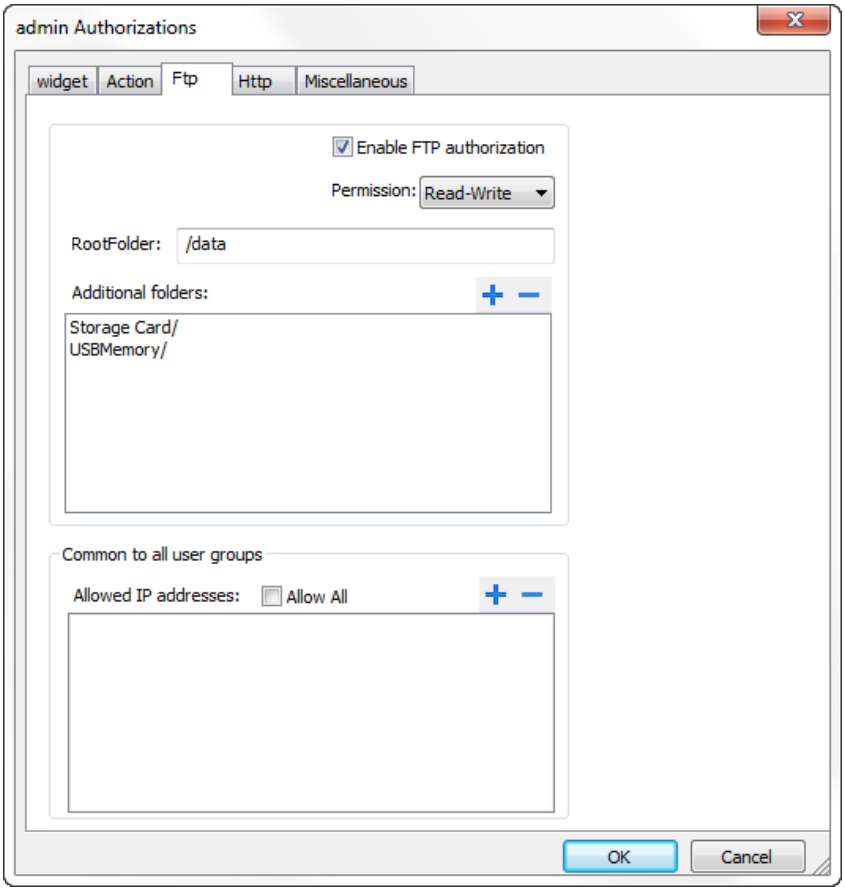
In the **Action** tab you can define action authorizations at project level, at page level or at widget level. Actions can be either **Allowed** or **Not Allowed**.



Action permissions can be modified directly from the project page. See "Assigning widget permissions from page view" on page 234 for details.

FTP authorizations

In the **Ftp** tab you can set specific authorizations for the FTP server.



Element	Description
Enable FTP authorization	Enables the FTP function for the specific group
Permission	Type of permission: • **Read-Only**

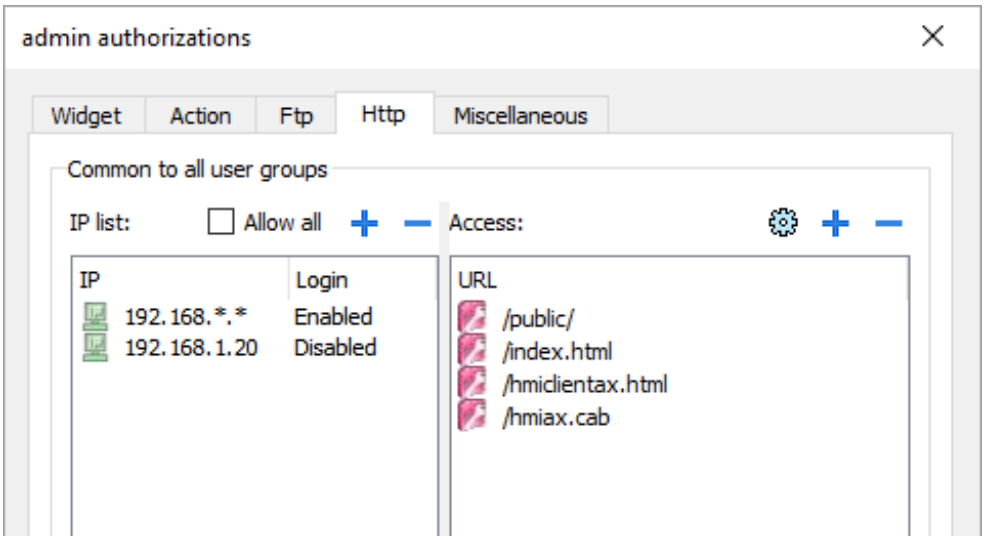
Element	Description
	• **Read-Write**
Root Folder	Folder to be used as root for FTP access. This is a relative path.
Additional folder	Extra folders to be used as root for FTP access (for example, on USB drive or SD card)
Allowed IP Addresses	List of IP addresses from which FTP connection can be accepted. **Important: This setting is common to all users groups.**

HTTP authorizations

In the **HTTP** tab you set restrictions to HTTP access to the web server integrated in HMI Runtime.

Wildcards can be used to identify a range of IP addresses.

For example, the two following rules set the HMI device unit can only be accessed by all the IP addresses 192.168.*.* on your local network in which only the IP address of 192.168.1.20 can access the device without entering a login name.



Element	Description
IP list	IP addresses authorized to access the HTTP server. By default all.
Login	When disabled, the username and password are not required.
Access limits	List of resources for which access is limited

Effect of these settings depends on whether the option **Force Remote Login** has been selected. See "Force remote login" on page 236 for details.

Force Remote Login	Default Access to workspace	Access limits
-	Full	-
Disable	Full	Can be used to block access to some files/folders or to require authorization
Enable	No Access	Can be used to open access to files/folders

Important: This setting is common to all users groups.

Adding an HTTP configuration

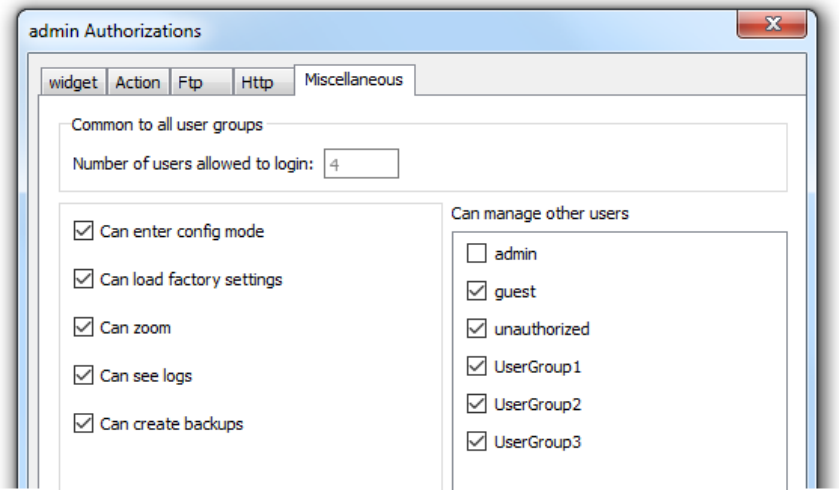
To add and configure a new access click **+**: the **Access limits** dialog is displayed.

To restore the default configuration click the **Set default access limits** icon. Default configuration allows access to the following:

- PUBLIC folder and Index.html

Miscellaneous settings

In the **Miscellaneous** tab you can define various authorization settings.



Note: Some of these settings are group specific, while other are common to all groups.

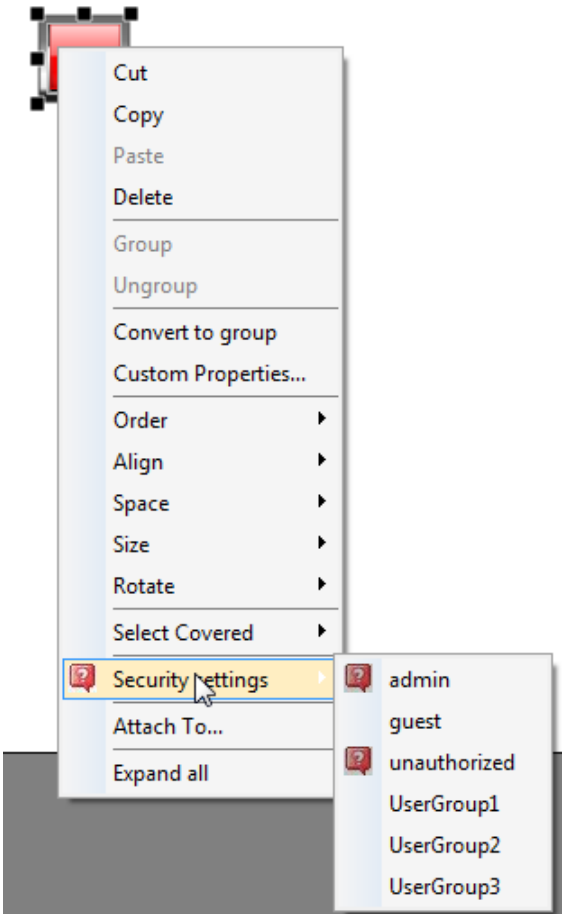
Option	Description
Can enter config mode	Enables switching from runtime to configuration mode. Normally used for maintenance.
Can manage other users	Gives super user privileges at run time to manage the select groups. Allows adding, deleting and modifying users' permissions.

Option	Description
Can load factory settings	Restores factory settings.
Can zoom	Enables zoom in/out in context menu at run time
Can see log	Allows user to see logs at run time
Can create backup	Allows user to backup project.
Number of users allowed to login	Maximum number of users that can be connected to the HMI Runtime at the same time. Default is 3.

Assigning widget permissions from page view

You can assign different levels of security, to different user groups, on a single widget, directly from the project pages.

1. Right-click on the widget and select **Security settings**.
2. Choose the group: the authorization dialog for the group is displayed.
3. Set the security properties to access the widget.

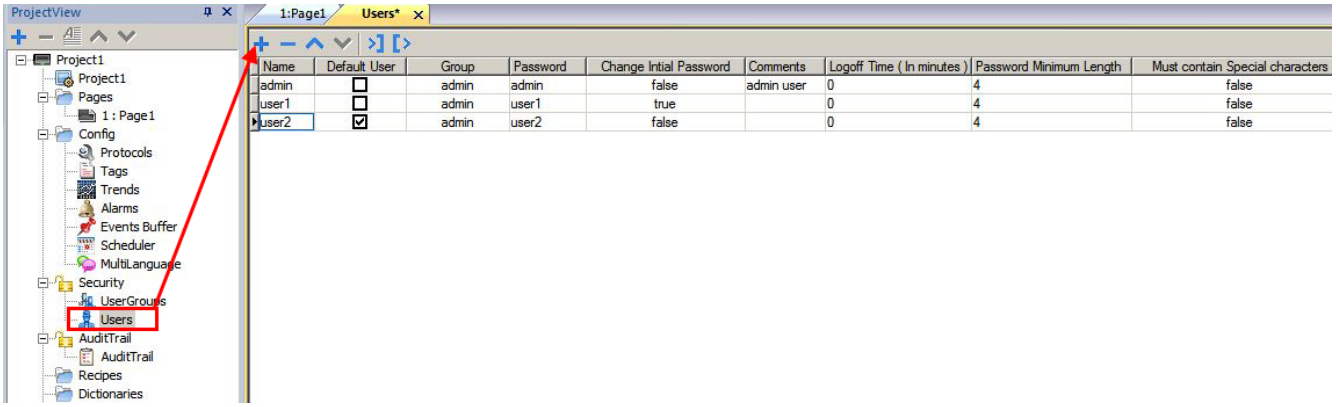


See "Modifying access permissions" on page 229 for details.

Configuring users

Path: ***ProjectView*** *>* ***Security*** *> double-click* ***Users***

In the Users editor, click **+** to add a user: one row is added to the table.



Parameter	Description
Name	User name
Default User	This user is automatically logged in when the system is started or after another user has logged off. Only one Default user can be set.
Group	User group
Password	User password
Change Initial Password	This user is forced to change his password at first log in.
Comments	Further user description
Logoff time	Minutes of inactivity after which the user is logged off. Set to 0 to disable.
Password Minimum Length	Minimum length of password
Must Contain Special Characters	Password must contain at least one special character.
Must Contain Numbers	Password must contain at least one numeric digit.

Default user

You can define only one default user in a project. This is the user automatically logged in at system start up and when the currently logged user logs out or is logged out after time-out.

To log into HMI Runtime with a different user, use one of the actions:

- **SwitchUser**
- **LogOut**

See "User management actions" on page 147for details.

Managing users at run time

The default user, if any, is automatically logged in when the HMI Runtime is started. If no default user is configured, the system requires a user name and password. See "User management actions" on page 147for details on the actions that can be executed on users.

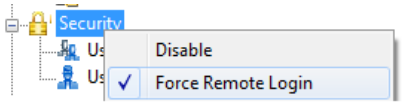
Removing user data

All the user information modified at run time is stored in dedicated files. To remove these dynamic files and all the changes applied to user configuration at run time you can:

- on HMI Runtime: execute the action DeleteUMDynamicFile
- with Wizard: select the **Delete Dynamic Files** in the download dialog.

Force remote login

Path: ***ProjectView*** *>* *right-click* ***Security*** *>* ***Force Remote Login***



Select this option to force user to log in when using remote access viaHMI Client. If not selected, remote access will use the same level of protection of local access.

Important: This function only works when user management is enabled.

Tip: Use this option when you have a default user but at the same time you want to protect remote access.

See "Enable/disable security management" on page 228for details.

The only files/folders still accessible when this flag is enabled are:

- PUBLIC folder and Index.html.

See "Modifying access permissions" on page 229 for details on HTTP access limits.

24 Audit trails

The Audit trail is a chronological sequence of audit records. Each record contains information on the actions executed and the user that performed them.

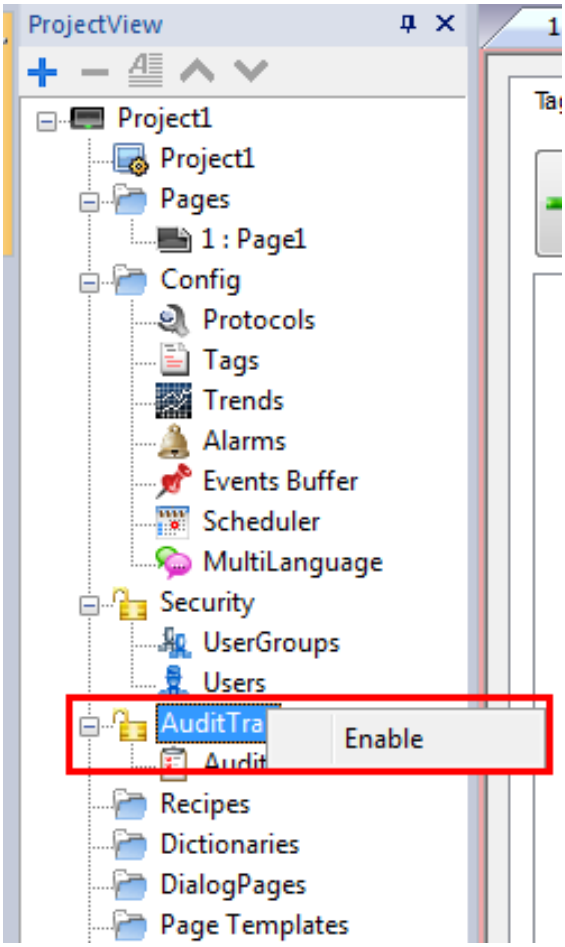
This function provides process tracking and user identification with time stamp for events.

If User Management is enabled, the actions are traced together with the name of the user. Only administrator user can modify this setting.

Enable/disable audit trail

Path: ***ProjectView*** *>* *right-click* ***AuditTrail*** *>* ***Enable***

The padlock symbol indicates status of the function.

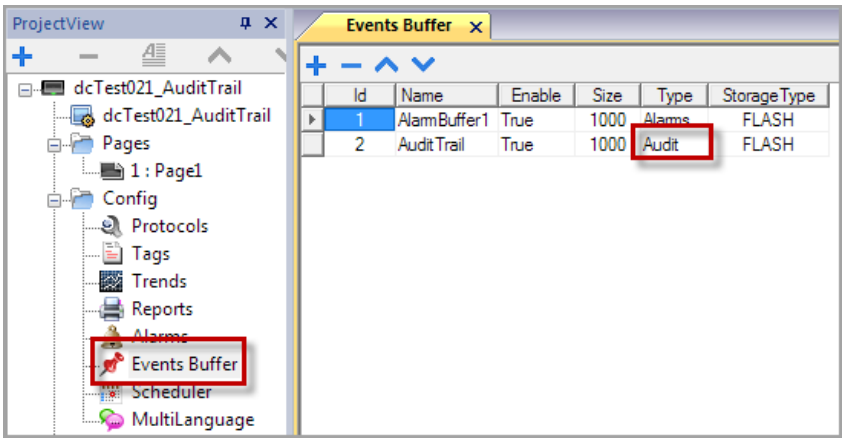


Configure audit events

You can have more than one set of audit records. You need to configure a dedicated event buffer.

Creating an event buffer

Path: ***ProjectView*** *>* ***Config*** *>* *double-click* ***Event Buffer***



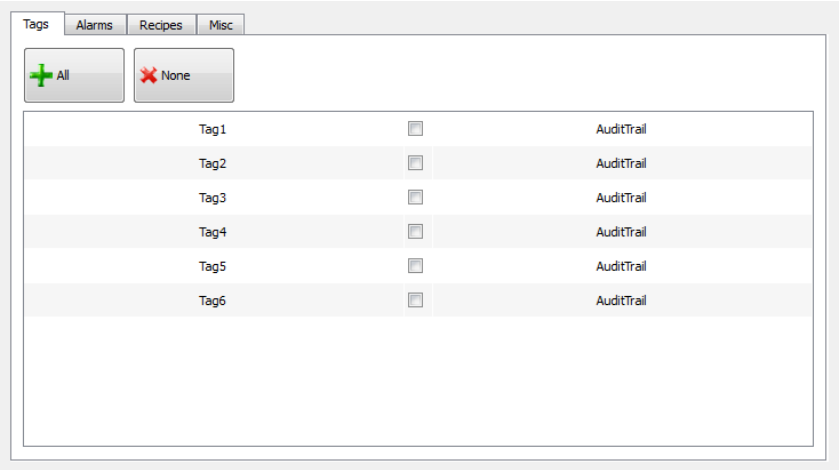
1. In the **Event Buffer** editor, click **+**: a row is added to the table.
2. Select **Audit** for **Type**.
3. Configure buffer parameters.

Parameter	Description
Id	Buffer identification number
Name	Buffer name
Enable	Enable/disable logging
Size	Size of log file. Data is automatically saved to disk every 5 minutes.
Type	Type of events logged: • **Alarms** • **Audit** • **Generic**
Storage Device	Device where audit data will be stored

Configure tags for audit trail

Path: ***ProjectView*** *>* ***AuditTrail*** *> click* ***AuditTrail***

Track only the tags related to actions that you want to keep under control. For tracked tags, all write operations will be logged together with the time stamp and user that performed the operation.

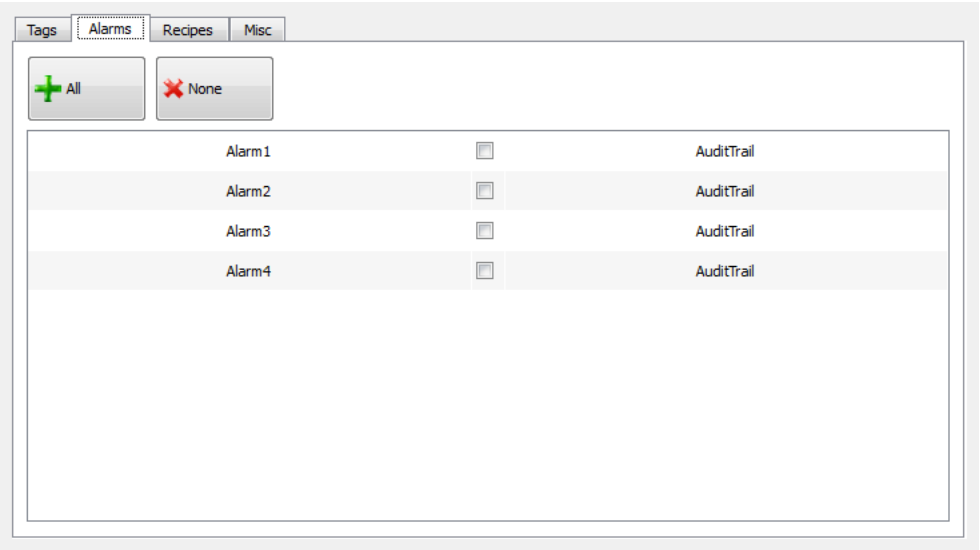


Configure alarms for audit trail

Path: ***ProjectView*** *>* ***AuditTrail*** *> click* ***AuditTrail***

You can specify the alarms to be tracked by the audit trail.

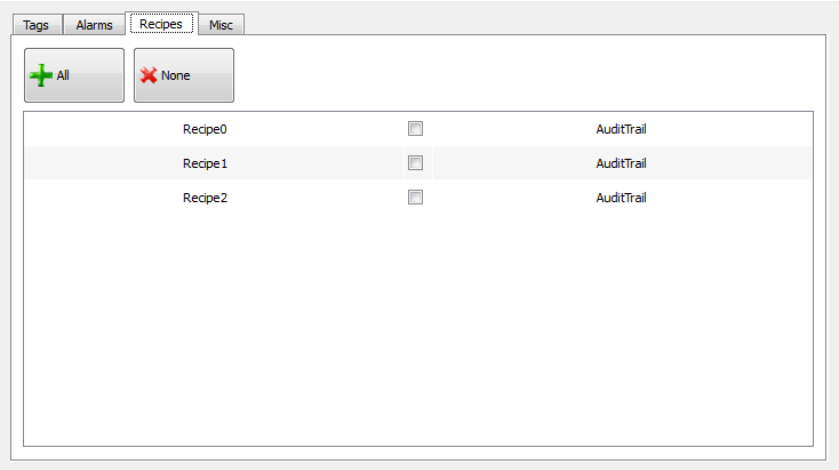
1. In Audit Trail editor, select the **Alarms** tab.
2. Select all the alarms to log in the audit trail: all operations performed on the specified alarms will be logged.



Configure recipes for audit trail

Path: ***ProjectView*** *>* ***AuditTrail*** *> click* ***AuditTrail***

Track only the recipes related to actions that you want to keep under control. For tracked recipes, all transfer operations will be logged together with the time stamp and user that performed the operation.

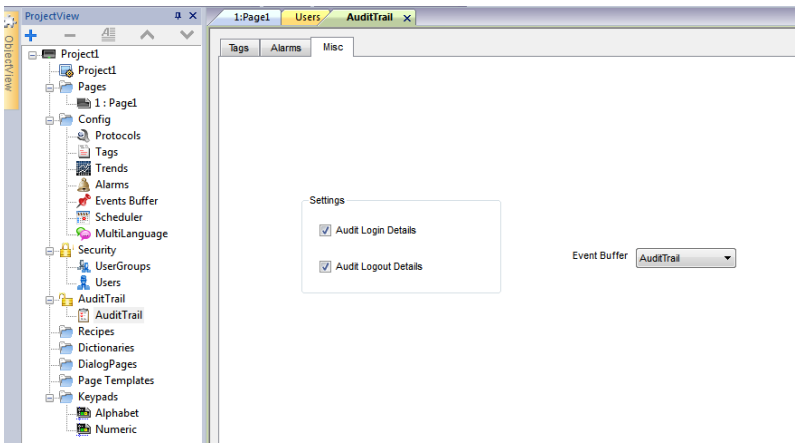


Configure login/logout details

Path: ***ProjectView*** *>* ***AuditTrail*** *> click* ***AuditTrail***

Audit trail can trace information about user login and user logout events.

1. In Audit Trail editor, select the **Misc** tab.

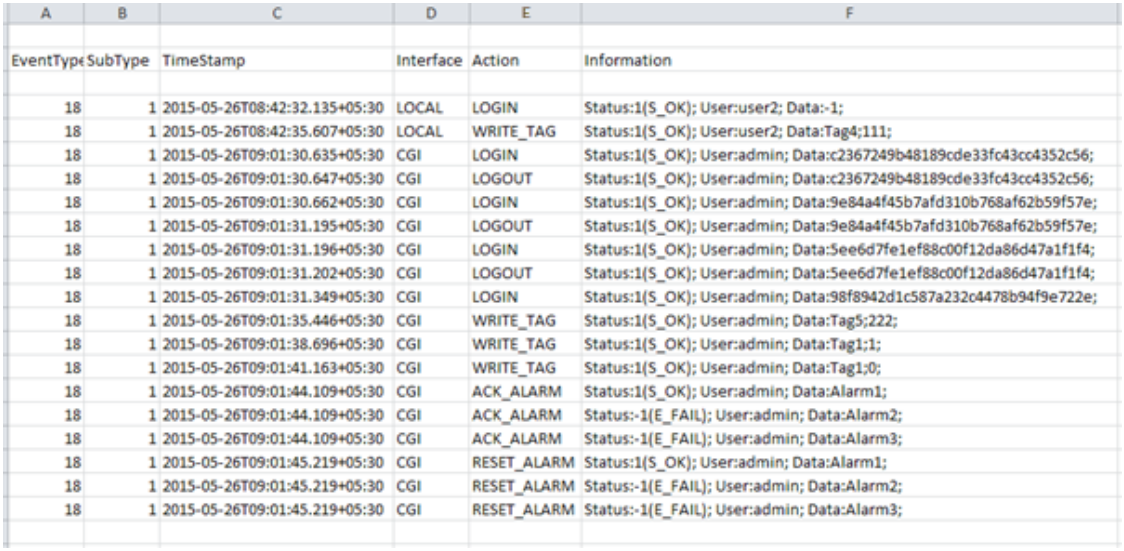


2. Select the information you want to log.
3. If you created additional event buffers of type **Audit**, then you can choose them from the **Event Buffer** combo box or you can leave the value **AuditTrail** that will use the default buffer.

Exporting audit trail as .csv files

To view audit trail data you have to export it o a csv file using the **DumpEventArchive** action. See "System actions" on page 134 for details.

File structure



A	B	C	D	E	F
EventType	SubType	TimeStamp	Interface	Action	Information
18	1	2015-05-26T08:42:32.135+05:30	LOCAL	LOGIN	Status:1(S_OK); User:user2; Data:-1;
18	1	2015-05-26T08:42:35.607+05:30	LOCAL	WRITE_TAG	Status:1(S_OK); User:user2; Data:Tag4;111;
18	1	2015-05-26T09:01:30.635+05:30	CGI	LOGIN	Status:1(S_OK); User:admin; Data:c2367249b48189cde33fc43cc4352c56;
18	1	2015-05-26T09:01:30.647+05:30	CGI	LOGOUT	Status:1(S_OK); User:admin; Data:c2367249b48189cde33fc43cc4352c56;
18	1	2015-05-26T09:01:30.662+05:30	CGI	LOGIN	Status:1(S_OK); User:admin; Data:9e84a4f45b7afd310b768af62b59f57e;
18	1	2015-05-26T09:01:31.195+05:30	CGI	LOGOUT	Status:1(S_OK); User:admin; Data:9e84a4f45b7afd310b768af62b59f57e;
18	1	2015-05-26T09:01:31.196+05:30	CGI	LOGIN	Status:1(S_OK); User:admin; Data:5ee6d7fe1ef88c00f12da86d47a1f1f4;
18	1	2015-05-26T09:01:31.202+05:30	CGI	LOGOUT	Status:1(S_OK); User:admin; Data:5ee6d7fe1ef88c00f12da86d47a1f1f4;
18	1	2015-05-26T09:01:31.349+05:30	CGI	LOGIN	Status:1(S_OK); User:admin; Data:98f8942d1c587a232c4478b94f9e722e;
18	1	2015-05-26T09:01:35.446+05:30	CGI	WRITE_TAG	Status:1(S_OK); User:admin; Data:Tag5;222;
18	1	2015-05-26T09:01:38.696+05:30	CGI	WRITE_TAG	Status:1(S_OK); User:admin; Data:Tag1;1;
18	1	2015-05-26T09:01:41.163+05:30	CGI	WRITE_TAG	Status:1(S_OK); User:admin; Data:Tag1;0;
18	1	2015-05-26T09:01:44.109+05:30	CGI	ACK_ALARM	Status:1(S_OK); User:admin; Data:Alarm1;
18	1	2015-05-26T09:01:44.109+05:30	CGI	ACK_ALARM	Status:-1(E_FAIL); User:admin; Data:Alarm2;
18	1	2015-05-26T09:01:44.109+05:30	CGI	ACK_ALARM	Status:-1(E_FAIL); User:admin; Data:Alarm3;
18	1	2015-05-26T09:01:45.219+05:30	CGI	RESET_ALARM	Status:1(S_OK); User:admin; Data:Alarm1;
18	1	2015-05-26T09:01:45.219+05:30	CGI	RESET_ALARM	Status:-1(E_FAIL); User:admin; Data:Alarm2;
18	1	2015-05-26T09:01:45.219+05:30	CGI	RESET_ALARM	Status:-1(E_FAIL); User:admin; Data:Alarm3;

Exported data file has the following content:

EventType	For internal use
SubType	
TimeStamp	Event time stamp. Time can be configured as local or global from the dump action.
Interface	LOCAL, when the action is performed in HMI Runtime. CGI, when the action is performed by a remote client.
Action	Action executed.
Information	Action status and operation executed. For example, write Tag - Tag1:50

Viewing audit trails

Audit trail data must be exported as a data file for viewing.

See "Exporting audit trail as .csv files" on the previous page for details.

25 Reports

A report is a collection of information that will be printed when triggered by an event. When the programmed event is triggered, the printing starts in background.

You can configure reports, their contents, trigger conditions and output printer in the Reports editor.

Not all widgets can be used in reports. When configuring reports, Wizard provides access to a dedicated widget gallery featuring only widgets available for reports.

Reports format can be customized using predefined templates for page layout.

Note: Report printing is not supported by HMI Client.

Adding a report

Path: ***ProjectView*** *>* ***Config*** *> double-click* ***Reports***

In **Reports** editor, click **Graphic Report** or **Text Report**: one new row is added to the table.

Report types

Report type	Description
Text Reports	Use for line-by-line printing of alarms. Only used for line printers. Text is sent to the printer without using any special driver. **Important: This printing mode requires using a physical port and only works on Windows CE platforms.**
Graphic Reports	Contain graphical elements and may include complex widgets such as screenshots or alarms. **Important: Each printer requires a specific printer driver. See "Configuring graphic reports" on the facing page for a list of supported printer drivers.**

Configuring text reports

Use the **Reports** editor . **Paper Size** in number of characters.

Setting printer options

Use printer options to control flush of pages on printer.

Printing starts either immediately or after a timeout. In printer options you can force flush as soon as a specific condition occurs, after a specified number of events, lines or seconds.

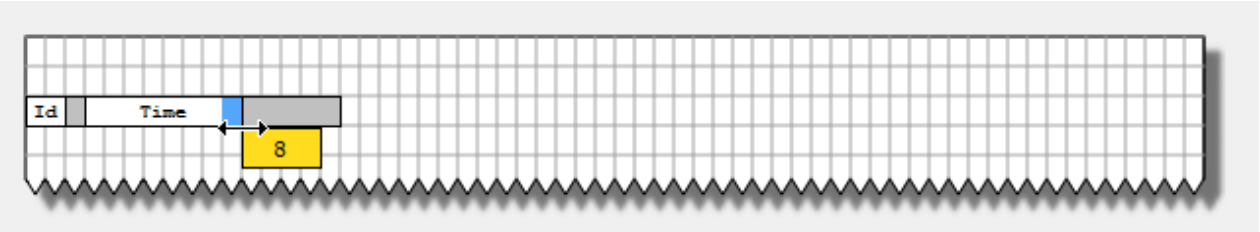
Note: Text reports do not support PDF format.

Setting alarms layout

Paper Size is the width of paper in number of characters.

Adding fields to the report

To add an item to the report, drag and drop it on the template page from the **Available fields** list.

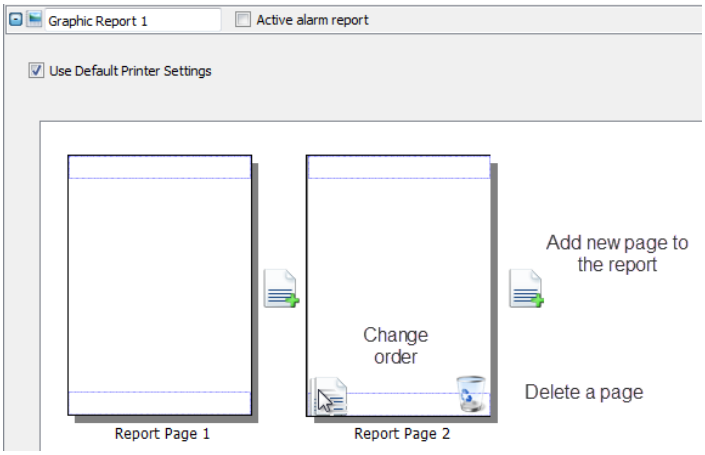


Re-size the field using the mouse, a tool tip shows the dimension in number of characters.

Note: If the text does not fit in the dedicated space, the auto wrap is applied.

Configuring graphic reports

Use the **Report** editor to configure graphic reports.



Adding a report page

Click **+** to add a new page to the report layout.

When the mouse goes over a page, two icons are displayed and allow you to reorder or delete the pages.

Modifying report page content

1. Double click on a page to edit its content: the **Graphic Report** editor appears.

   Each page is divided in: header, footer and page body.

2. Double click on the area you want to edit: the edit area is shown in white, others are grayed out.

The Widget Gallery is context-sensitive and displays only the widgets available for the area you are editing.

Widgets available for reports

Widgets that can be used for a graphic report:

Widget	Function
Page Number	Automatic page numbering
Screenshot	Screen capture of the page currently displayed by the HMI device. The report page is automatically resized to fit the HMI device page. Note: The full screen is printed, including all open dialogs.
Alarm	Entire contents of the event buffer (default buffer is Alarm Buffer1).
Text	Widgets such as labels and numeric fields

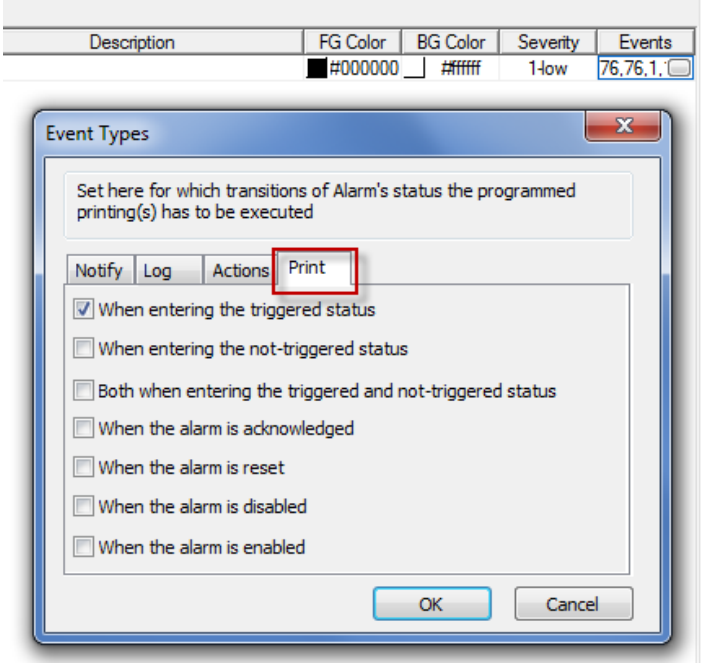
Print triggering events

Report printing can be triggered by events.

Configuring alarm printing

Path: ***ProjectView*** *>* ***Config*** *> double-click* ***Alarms***

1. In the Alarms editor, open the **Event Types** dialog from the **Events** column.
2. In **Print** tab select all the conditions for which you want to trigger printing.

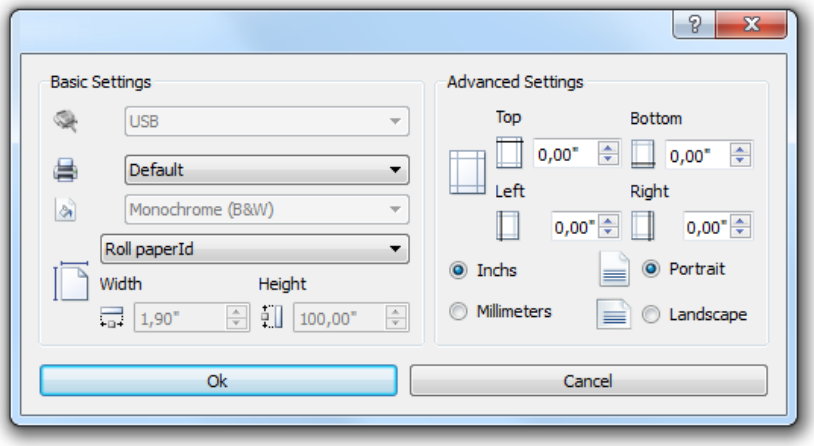


Important: Only one report can be set as Active alarm report in a project and it can be either a text report or a graphic report.

Adjusting printer settings at run time

A graphic report printing can be started also using the action **PrintGraphicReport**.

Set the action property **silent** to **false** to have a pop-up dialog.



Default printer

Printer setting

You can set a default printer for all graphic reports. Each report can then be configured to use the default printer or any other printer available. Click **Printer Setting** button to set printer parameters.

For PDF printers you also define the folder where files are saved by using **Printed Files Location**.

Supported printers

List of printers and printer languages supported by the Windows CE driver printCE.dll. Printers not available in the list but compatible with these languages are supported.

Printer	Languages
HP PCL 3, HP PCL 5e, HP PCL3GUI	HP PCL3/PCL5e/PCL3GUI, including DeskJet, LaserJet, DesignJet
Epson ESC/P2	ESC/P2, LQ
Epson Stylus Color	Epson Stylus Color
Epson LX (9-pin)	9-pin printers, Epson LX, FX, PocketJet
Cannon iP100, iP90, BubbleJet	BubbleJet, iP90, iP100
PocketJet II, 200, 3	Pocket Jet
MTE Mobile Pro Spectrum	MTE Mobile Pro Spectrum
Adobe PDF File	Adobe PDF file

Printer	Languages
SPT-8	SPT-8
M1POS	M1POS
MP300	MP300
Zebra	Zebra CPCL language
Intermec PB42, PB50, PB51, PB2, PB3	Intermect PB42/50/51/2/3 with ESC/P language
Datamax Apex	Datamax Apex

Supported ports

The following ports are supported:

- LPT1 (USB printers)
- File (PDF)

Note: On Win32 platform, only PDF and default printers are supported. Default printer is the default OS printer and it can be connected with any kind of port (not only USB).

Tested printers

The following printers have been tested with printCE drivers in Windows CE HMI devices.

Driver	Printer Model	Graphic	Line
Custom	Plus 4 Kube II	Yes	Yes
Epson ESC/P 2	Epson AcuLaser M2310	Yes	Simulate
Epson LX (9-pin)	Epson LX-300+II	No	Yes
HP PCL 3	HP LaserJet P2015dm	Yes	Simulate
	HP LaserJet 4700dtn	Yes	Yes
HP PCL 3 GUI	HP Deskjet 1010	Yes	No
	HP Deskjet D5560	Yes	No
	HP LaserJet 4700dtn	No	Yes
HP PCL 5e	HP LaserJet P2015dm	Yes	Simulate
	HP LaserJet 4700dtn		
INTERMEC	Intermec PB50 with ESC/P language with 4 inch roll paper. Note: The HMI device crashes when trying to print on	Yes	Yes

Driver	Printer Model	Graphic	Line
	Intermec PB50 printers in standby mode after a first successful print job.		
PDF	-	Yes	No

26 Screen saver

Screen saver can be used to display a slide show when the HMI device is not in use. The slide show starts after a timeout if none of the following events occur:

- touch of display
- mouse movement
- external keyboard key pressed

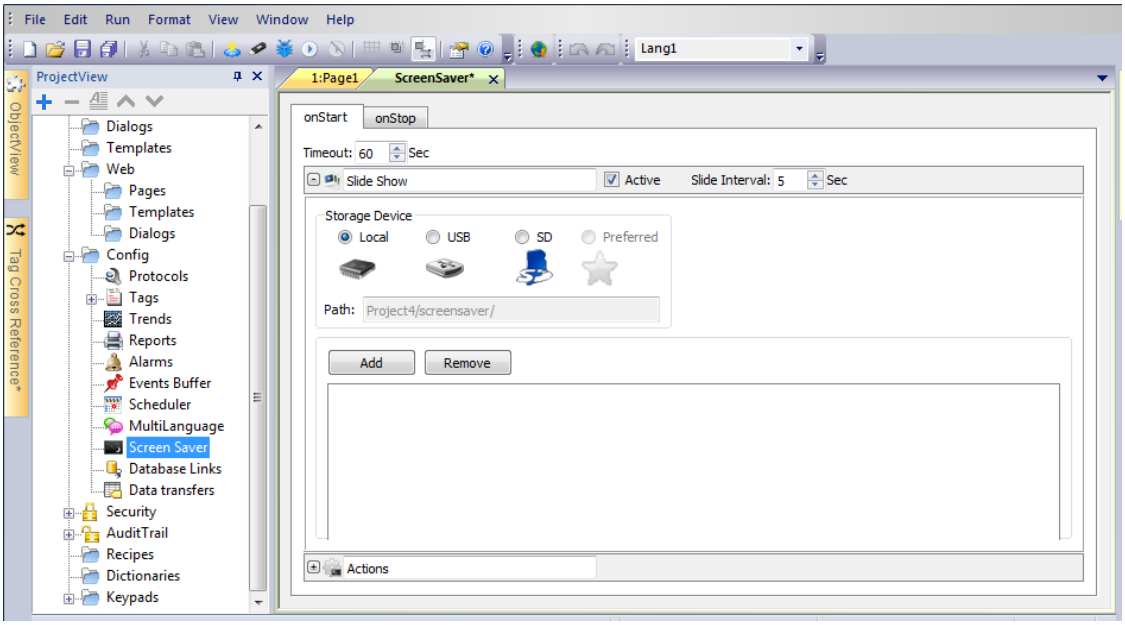
Enabling the screen saver function

Path: ***ProjectView*** *>* ***Config*** *> right-click* ***Screen Saver*** *>* ***Enable***

Important: You must enable the screen saver before you can configure it.

Configuring a screen saver

Path: ***ProjectView*** *>* ***Config*** *> double-click* ***Screen Saver***



Slide show parameters

Parameter	Description
Timeout	Time after which the slide show starts
Slide Interval	Interval between slides

Parameter	Description
Storage Device	Location of the images used in the slide show. Images stored locally are saved in *workspace\projectname\screensaver* and can be downloaded to the HMI device when the project is downloaded. Images stored on USB or SD devices are saved in a screensaver folder on the device itself. **Important: Only JPEG and PNG images are supported.**

Associating actions to the screen saver

Actions can be triggered by the screen saver start and/or stop.

- Click **+** next to **Actions** in the **onStart** tab to configure actions to be executed when the screen saver starts.
- Click **+** next to **Actions** in the **onStop** tab to configure actions to be executed when the screen saver stops.

27 Backup/restore of Runtime and project

You can backup all the content of the HMI device, including HMI Runtime and project, to an external memory. This backup copy can be used to restore the content of the HMI device at a later time or copy it to a new HMI device.

The backup function is available only if enabled for the logged user. See "Modifying access permissions" on page 229 for details.

Note: Backup is not supported in Win32 / HMI Client.

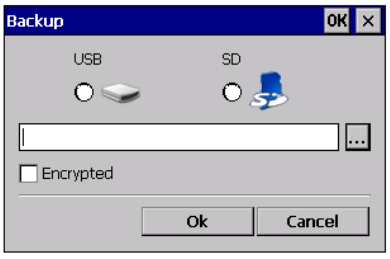
Backup function

The backup function automatically performs the following procedure:

1. Unloads the current project to unlock files in use.
2. Archives the content of the \QTHMI folder (containing HMI Runtime, projects, dynamic files such as recipes, alarms, trends and so on) to a .zip file (standard or encrypted).
3. Reloads the project.

To start the backup procedure:

1. In HMI Runtime right click to open the context menu.
2. Select **Backup:** the **Backup** dialog is displayed.



3. Select the path for storing the backup file.

Note: The backup process does not include files stored in USB and SD cards. Dynamic data such as recipes, trends, events stored in these devices will not be included in the backup.

Restore function

Restore the backup package can be perform on HMI device

- from the Context Menu (see "Update package" on page 72 for details)
- or from the System Settings (see "System Settings" on page 373 for details)

28 Keypads

Several keypads are provided by default in the Wizard so that they can be used for data entry.

The alphabet keypad can be use associate with a string data type

The numeric keypad can be use associate with a numeric data type

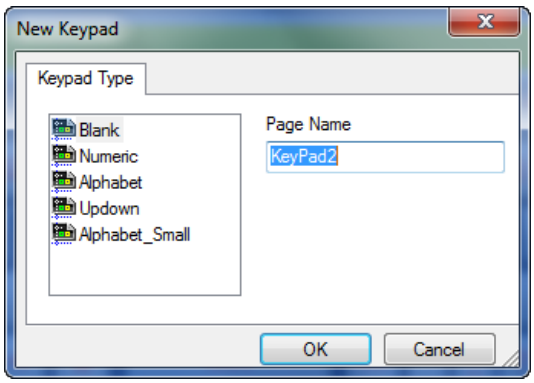
The calendar keypad can be use associate with a date data type

Creating and using custom keypads

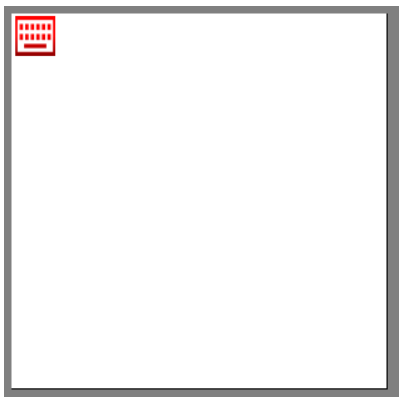
You can either create a new keypad or customize an existing one.

Creating a keypad

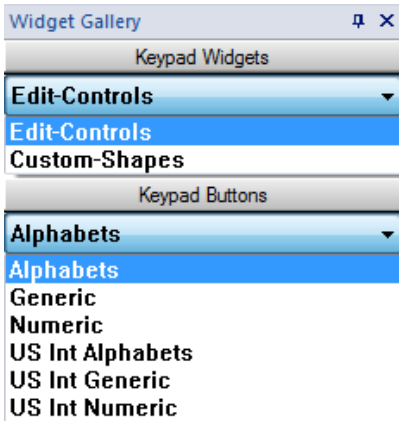
1. In **ProjectView**, right-click **Keypads** and select **Insert Keypad**: the **New Keypad** dialog is displayed.



2. Select one of the available keypads, or **Blank** to create a keypad from scratch. In this case a blank keypad is displayed.



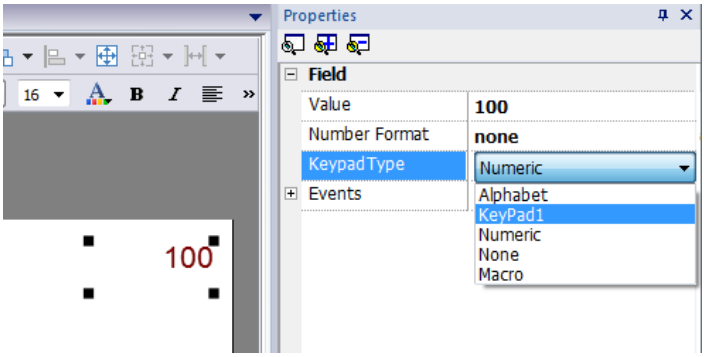
3. Use the **Keypad Widgets** and **Keypad Buttons** from the Widget Gallery to create your custom keypad.



The keypad you create, as in this example, will be saved in the project folder.

Attaching custom keypads to fields

Custom keypads can then be reused for any field where the **Keypad** property points to it as in this example.



Tips and tricks with custom keypads

By default, any numeric widget (read/write numeric field) are assigned the numeric keypad.

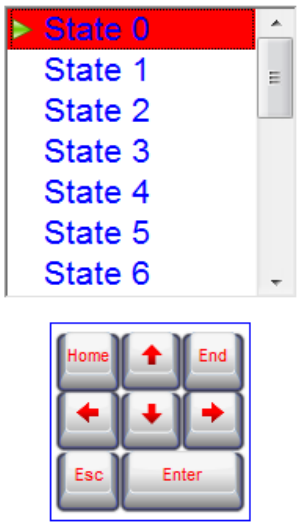
If you want to apply a customized version of the numeric keypad to all the numeric widgets you add to your project proceed as follows:

1. Create a new keypad and select **Numeric** as **Keypad** type. This will be a backup of the original settings for the numeric keypad.
2. Customize the default numeric keypad and save it. This customized version of the numeric keypad will now be assigned as default in the project.

See "Deleting or renaming custom keypads" on the facing page for details on how to rename a custom keypad.

Up-down arrows keypad

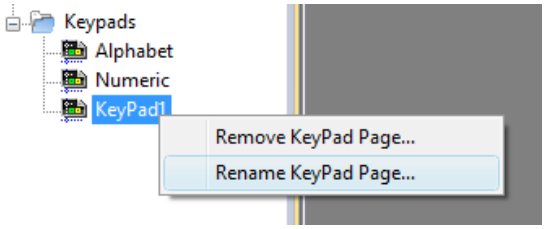
This type of keypad is particularly useful to move the cursor up and down within widget requiring this functionality. Here an example using a **Control List** widget. See "Control list widgets" on page 289 for details.



Deleting or renaming custom keypads

In **ProjectView**, right-click on a custom keypad and select one of the options:

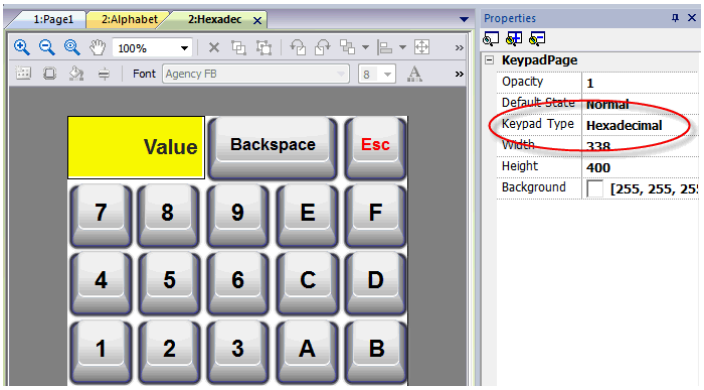
- **Remove KeyPad Page** to remove the keypad from the project
- **Rename Keypad Page** to rename the keypad.



Keypad type

Path: ***ProjectView*** *>* ***Keypads*** *> double-click a keypad >* ***Properties***

Set **Keypad Type** parameter for a keypad to define the type of data entry.



Keypad Type	Description
Auto	Default setting
Decimal	Only numeric keys are accepted. Entering 10, the keypad returns 10 that will be displayed as "10" if the attached field is numeric or ASCII, as 'A' if the attached filed is hexadecimal.
Hexadecimal	Only hexadecimal keys are accepted. Entering 10, the keypad returns 16 that will be displayed as "16" if the attached field is numeric or ASCII, as "10" if the attached field is hexadecimal.
Ascii	All keys are enabled. Entering 1A, the keypad returns 1A that will be displayed as '1' if the attached field is numeric, as "1A" if the attached field is ASCII or if the attached field is hexadecimal.

Keypad position

Runtime Positioning property of keypads can be used to define where keypads will appear in the screen.

Option	Description
Automatic	The best position is selected according to here data entry is required.
Absolute	X,Y coordinates are entered to identify the exact position
Left-top **Left-center** **Left-bottom** **Center-top** **Center-center** **Center-bottom** **Right-top** **Right-cente** **Right-bottom**	Predefined screen positions

Select the **Lock Keypad position** option if you don't want the keypad to be moved by dragging.

29 External keyboards

HMI Runtime has been designed to work with external keyboards connected via USB.

Keyboards can be used for:

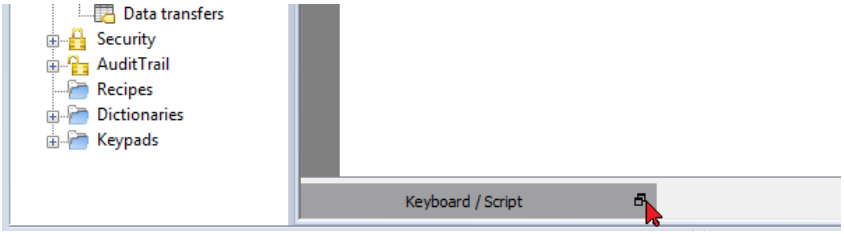
- data entry (default)
- execute actions mapped on specific keys

For example, the right arrow key **OnClick** event can be mapped to the **LoadPage** action.

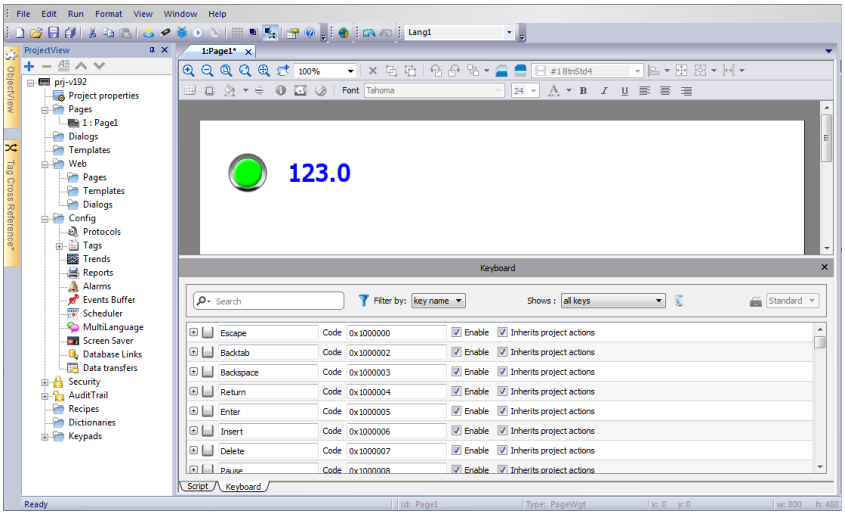
Keyboard can be programmed at project level so that settings will be inherited by all the pages. In each page you can then choose which key setting will be inherited from the project and which one you will customize for the specific page.

Opening external keyboards

1. In the Page Editor, click on the icon on the right of **Keyboard/Script** at the bottom of the workspace: the Keyboard/Script Editor is displayed.
2. Select the **Keyboard** tab.



Each row in the Keyboard Editor corresponds to a key.



For each key, the following information is displayed:

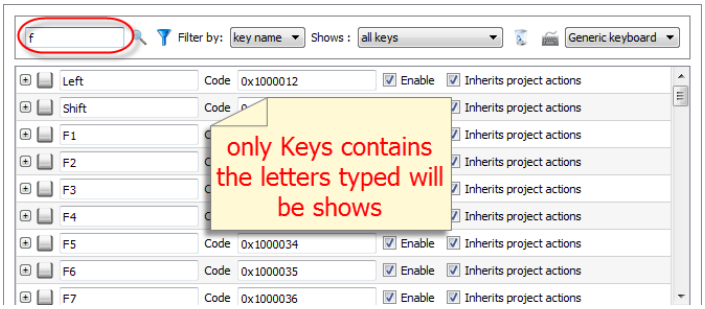
Element	Description
Label	Key name
Code	Key code
Enable	Key enable status
Inherits project actions	Defines whether the key is inheriting the action programmed at the project level

Here the possible configurations:

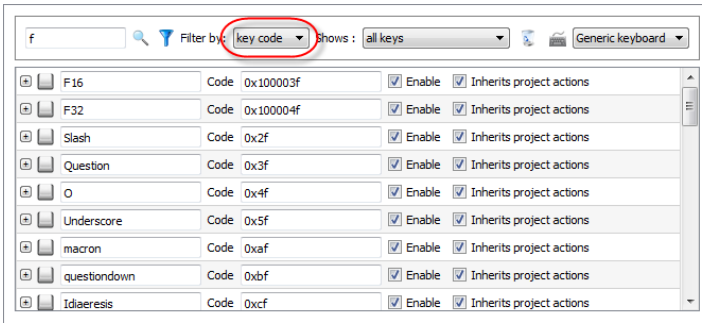
Enable	Inherits project actions	Editor appearance	HMI Runtime behavior
Checked	Unchecked	Action lists show the page actions (or nothing if the list is empty)	Only the page actions (if any) will be executed.
Checked	Checked	Action lists show the project actions only and cannot be edited	Only the configured project actions (if any) will be executed.
Unchecked	Checked	Inherits project actions check box and all action lists are disabled. Action lists show the project actions only.	No page or project action will be executed.
Unchecked	Unchecked	Inherits project actions check box and all action lists are disabled. Action lists show the project actions only.	No page or project action will be executed.

Search and filter

To display a filtered set of keys, in **Filter by** select **key name** and type a letter in the search field: only the keys containing that letter in their name will be displayed in the Keyboard editor.



Alternatively, in **Filter by** select **key code** and type a letter in the search field: only the key containing that letter in their code will be displayed in the Keyboard editor.



Displayed keys

You can easily select what keys will be listed in the Keyboard editor window. To display a limited set of keys, select an option in **Shows**.

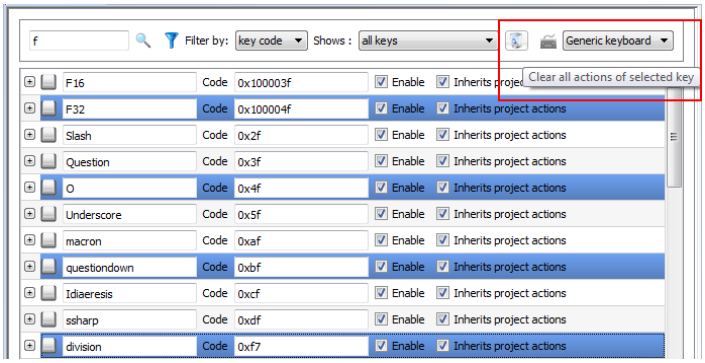
Option	Description
all keys	All keys available in the keyboard layout are listed
modified keys	Only the keys associated with actions at the page level are listed
modified keys in project	Only the keys associated with actions at project level are listed

Removing action associations

To remove all the associations you created between keys and actions:

1. Select the keys for which you want to remove the association.
2. Click the **Clear all actions of selected keys** button.

If you are working at page level, page actions will be removed, if you are working a project level, project actions will be removed.

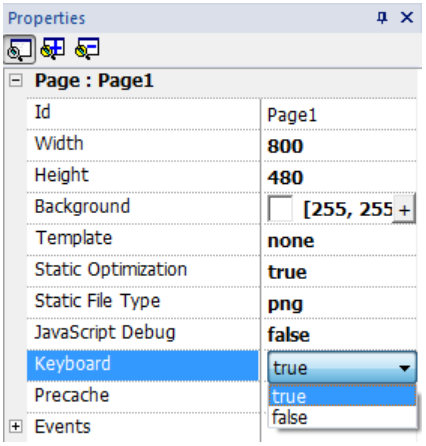


Keyboard layout

Select the layout of the keyboard from the **Keyboard Layout** combo box. **Generic Keyboard** refers to a generic international keyboard layout.

Enable/disable keyboard

You can enable/disable keyboard actions both at project and at page level. To enable keyboard actions, in the **Properties** pane set **Keboard macro** to **true**.

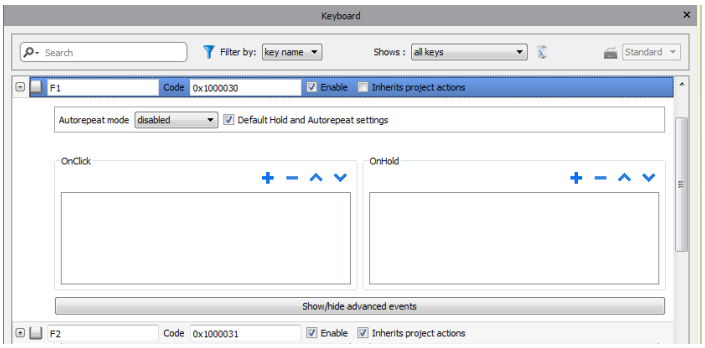


You can enable/disable keyboard actions also at run time using the KeyboardMacros action. See "Keyboard actions" on page 119 for details.

Associating actions to keys

You associate actions to a keys from the Keyboard editor.

1. Click **+** next to the key you want to program: the fields for key configuration are displayed.

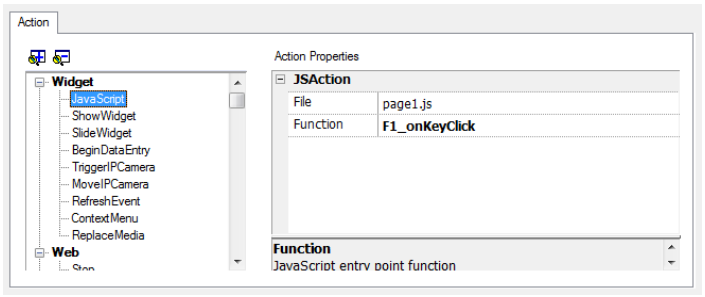


1. Click **+** to add actions.

You can associate actions both to the **OnClick** event and toe the **OnHold** event.

See "Events" on page 37 for details.

Note: Also JavaScript code can be associated to a key event.



30 Tag cross reference

The **Tag Cross Reference** pane displays a list of tag names used in current project organized according to their location and use.

From this pane you can:

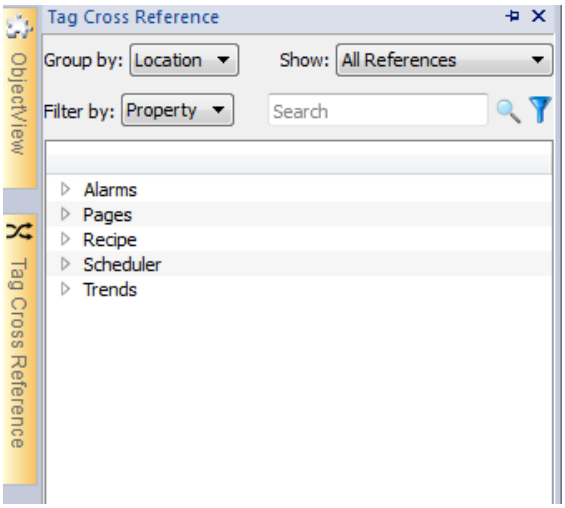
- verify where each tag is used (alarms, pages, recipes, schedulers, trends, and so on)
- identify invalid tag references (references to tags not defined in the tag editor)
- identify tags not used in the project

Note: The Tag Cross Reference pane does not list tags used in JavaScript code.

Opening the Tag Cross Reference pane

Path: ***View*** *>* ***Toolbars and docking windows*** *>* ***Tag Cross Reference***

Click the **Tag Cross Reference** tab to open the Tag Cross Reference pane.



Working in the Tag Cross Reference pane

The Tag Cross Reference pane provides a set of standard functions.

Element	Function
Group by	Groups tags by **Location** (alarms, pages, trends and so on) or **Tag** name
Show	Filters tags and displays: • **All Reference**: all tags • **Invalid Tag Reference**: tags not listed in the Tag Editor. • **Unused Tags**: tags listed in the Tag Editor but not used in project.
Search field	Applies a filter to display a limited number of tags
Filter by	Filters tags by **Location**, **Tag** or **Property**.

Navigate the listed tags to find where they are used inside the project.

Double-click on a tag to open the editor or page where it is used.

Updating data in the Tag Cross Reference pane

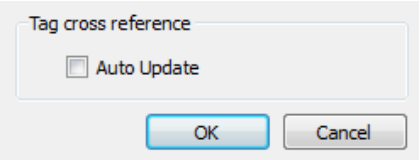
Manual update

By default, the information displayed in the Tag Cross Reference pane must be updated manually. To do this, click the refresh button  . A warning sign is displayed when a refresh is needed.

Automatic update

Path: ***View> Properties***

You enable the automatic update of the Tag Cross Reference pane from the Wizard **Properties** page.



Select the **Auto Update** option.

Exporting data

Data displayed in the Tag Cross Reference pane can be exported in .csv file.

Data is organized in the exported file according to how it was grouped in the pane.

Grouped by	File format
Location	RESOURCE, RESOURCE DESC, WIDGET-ID, ATTRIBUTE, TAG
Tag	TAG, RESOURCE, RESOURCE DESC, WIDGET-ID, ATTRIBUTE

Note: The separators used in export operation depends on regional settings of your computer.

31 Indexed addressing

Indexed addressing allows you to select a set of tags depending on the value of another tag. This is very useful, for example, to use the same graphics to visualize a set of data coming from different sources, all the user has to do is pick the source to monitor from a list.

Creating an indexed addressing set

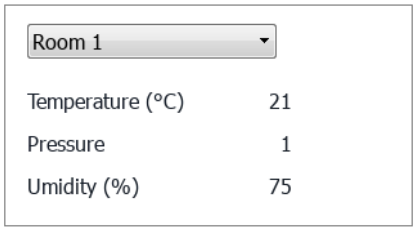
Scenario

In this scenario, environment data is collected from with four rooms, each equipped with temperature, pressure, and humidity sensors. Data is available as follows:

Room Number	Temperature	Pressure	Humidity
1	Room1-Temperature	Room1-Pressure	Room1-Humidity
2	Room2-Temperature	Room2-Pressure	Room2-Humidity
3	Room3-Temperature	Room3-Pressure	Room3-Humidity
4	Room4-Temperature	Room4-Pressure	Room4-Humidity

Using the indexed addressing feature, you can use a single table format to arrange all data in the HMI device.

Data from the three different sensors can be displayed in a single page where the room number is used as a selector (combo box) to pick the correct set of tags.



How to create an indexed tag set

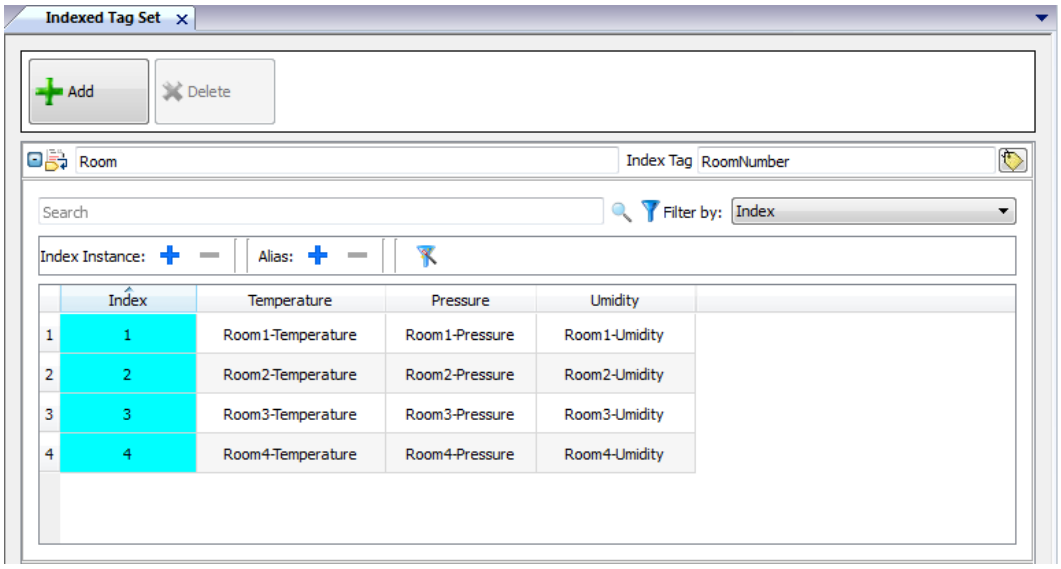
Path: ***ProjectView*** *>* ***Tags***

To do this you need to create an indexed tag set.

1. In the Tag Editor, define protocols and tag. Define a tag for each data to be indexed, in this example you must create a tag for each sensor in each room.

Name	Group	Driver	Address
Room1-Temperature		Modbus TCP:prot1	192.168.0.34:502:1 HREG 400001 unsignedShort
Room1-Pressure		Modbus TCP:prot1	192.168.0.34:502:1 HREG 400002 unsignedShort
Room1-Umidity		Modbus TCP:prot1	192.168.0.34:502:1 HREG 400003 unsignedShort
Room2-Temperature		Modbus TCP:prot1	192.168.0.34:502:1 HREG 400004 unsignedShort
Room2-Pressure		Modbus TCP:prot1	192.168.0.34:502:1 HREG 400005 unsignedShort
Room2-Umidity		Modbus TCP:prot1	192.168.0.34:502:1 HREG 400006 unsignedShort
Room3-Temperature		Modbus TCP:prot1	192.168.0.34:502:1 HREG 400007 unsignedShort
Room3-Pressure		Modbus TCP:prot1	192.168.0.34:502:1 HREG 400008 unsignedShort
Room3-Umidity		Modbus TCP:prot1	192.168.0.34:502:1 HREG 400009 unsignedShort
Room4-Temperature		Modbus TCP:prot1	192.168.0.34:502:1 HREG 400010 unsignedShort
Room4-Pressure		Modbus TCP:prot1	192.168.0.34:502:1 HREG 400011 unsignedShort
Room4-Umidity		Modbus TCP:prot1	192.168.0.34:502:1 HREG 400012 unsignedShort

2. Create a tag to be used as index tag. In this example you create a "RoomNumber" tag that could be of type UnsignedInt using Variable protocol.
3. From **ProjectView**, select **Config**> **Tags**, double-click **Indexed Tag Set**: the Indexed Tag Set editor is displayed.
4. Click + to add an Indexed Tag Set. In this example you will call it "Room".
5. Select the tag "RoomNumber" to use as a selector for the room number.
6. Create an **Index Instance** for each set of data. In this example, one for each room.
7. Create an **Alias** for each type of data and rename the table columns appropriately. In this example "Temperature", "Pressure" and "Humidity".
8. Double-click on each cell to associate the correct tag.



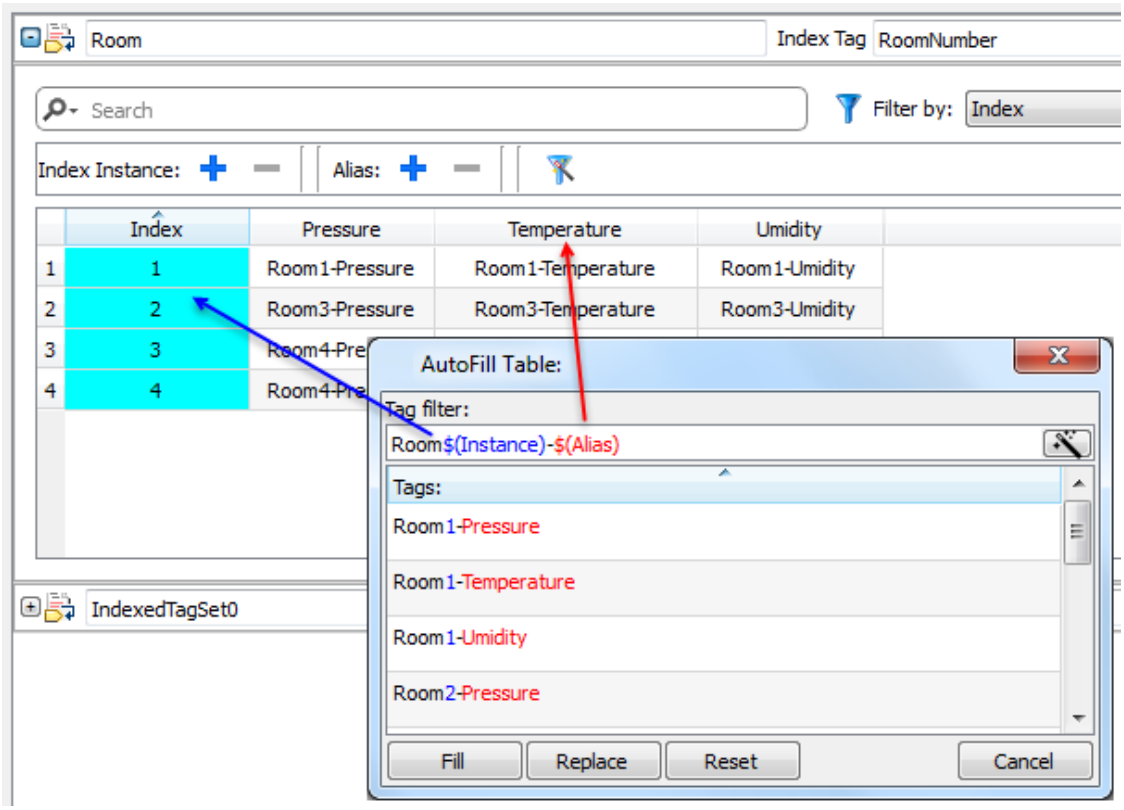
Note: The Index Tag datatype can be a number, a string or any type of simple data types.

Note: To reference an array data type use the array index = -1

Autofill function

An Indexed Tag Set table may become very complex and filling it may be an error prone procedure. Enable the Autofill feature to make sure aliases are entered correctly.

Click to enable the Autofill feature: the **Autofill Table** is displayed.



This function uses regular expression for populating the table with tags trying to match the filter where the keyword $(Instance) will be replaced with the defined Index values and the keyword $(Alias) with the defined alias labels.

Autofill example

“Room$(Instance)-$(Alias)” will match all tag names:

Room1-Temperature,

Room1-Pressure,

Room1-Humidity,

Room2-Temperature,

…

“Room0*$(Instance)-$(Alias)” will match all tag names:

Room1-Temperature,

Room01-Pressure,

Room001-Humidity,

Room2-Temperature,

Room02-Pressure,

Room002-Humidity,

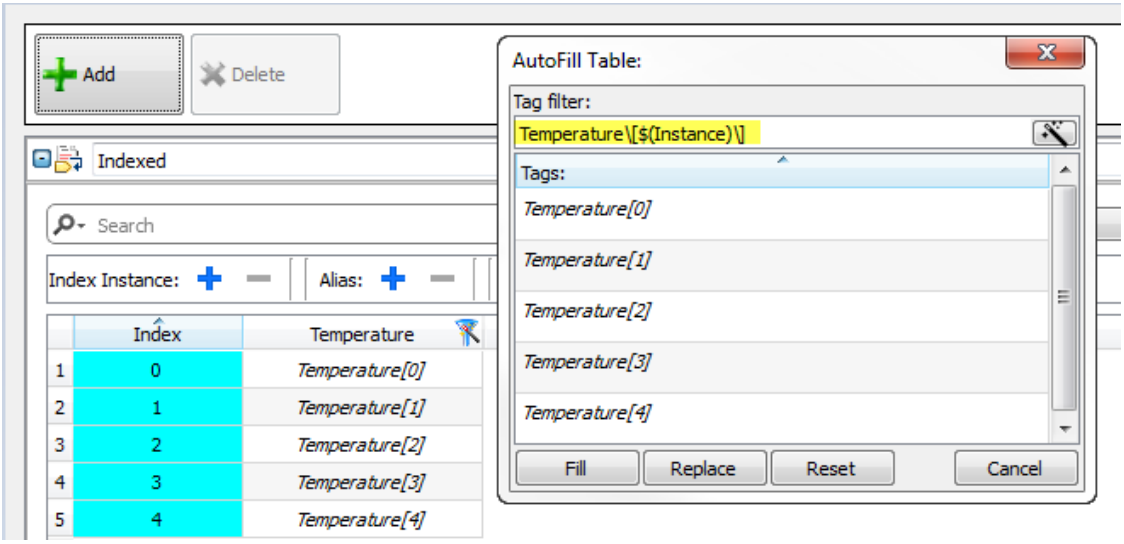
…

Autofill table elements

Element	Description
Fill	Fills in missing entries in the tag table using the set filter (if any). For example, when new instances or new aliases are added you can use this option to fill in the new entries.
Replace	Replace all table entries with those provided by the Autofill table.
Reset	Resets the tag filter to empty, no automatic fill is done.
	Suggests a valid filter expression for your project.

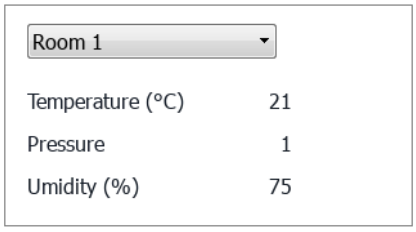
Note: Filters are saved as project preferences and can be set for the entire table or for a column. Once a filter is set for a column, the table filter is ignored. You can therefore selectively change the filter for handling a particular alias only.

Note: To reference the elements of an array use the \ character to disable the regular expression interpretation of the square brackets (array tags are differentiated by Italic).



Using indexed tag set in pages

Once an indexed tag set has been created, you can use it to create a page for the HMI device as in this example.

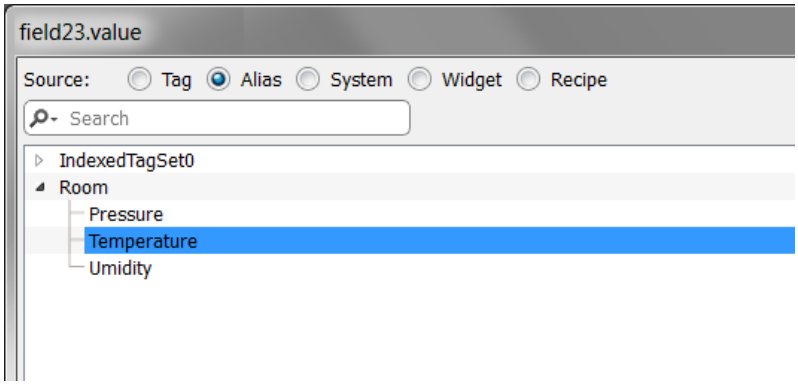


To create this page:

1. Create a page and add a combo box, three labels and three numeric fields.
2. Use the index tag created for the room number for the combo box, "RoomNumber" in this example. This will be the selector for the room number.
3. Create a list for the combo box. In this example use the following list.

Index	String List
0	Room Number
1	Room 1
2	Room 2
3	Room 3
4	Room 4

4. Attach to each numeric field value the corresponding Alias variable (**Room > Temperature, Room > Humidity, Room > Pressure**).



32 OPC UA Server

Path: ***ProjectView*** *>* ***Config*** *>* ***Interfaces*** *> double-click* ***OPC UA***

Use the OPC UA Server to publish data according to the OPC UA standard..

Features

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>Enable OPC UA Server</td><td>Activates operation of the OPC UA Server.
Data values defined in the HMI are published by the OPC UA Server.</td></tr>
<tr><td>Enable alarms</td><td>Activates publication of real-time alarm data (Active Alarms).
The following properties of alarms are published:
• Enabled
• Severity
• Status
The alarm states are mapped to OPC UA states according to the following rules:
<table>
<tr><th>OPC UA Alarm state</th><th>Wizard Alarm state</th></tr>
<tr><td>Opcua.Alarm.Active</td><td>TRIGGERED | TRIGGERED_NOT_ACKED |
TRIGGERED_ACKED</td></tr>
<tr><td>Opcua.Alarm.Acked</td><td>TRIGGERED_ACKED |
NOT_TRIGGERED_ACKED</td></tr>
<tr><td>Opcua.Alarm.Retained</td><td>TRIGGERED |
TRIGGERED_NOT_ACKED |
TRIGGERED_ACKED |
(alarm requires reset flag & state != NOT_TRIGGERED)</td></tr>
<tr><td>Opcua.Alarm.Confirmed</td><td>when alarm requires reset flag & state is NOT_TRIGGERED</td></tr>
</table>
OPC UA confirm operation is mapped to the reset operation. Confirmation only works if the alarm is currently active, otherwise return OpcUa_BadInvalidState (since the reset operation only works in this condition).
The acknowledge/enable/disable/confirm operations done via OPC UA are audited as "OPC UA Server" domain. If the authentication is user/pass the user is logged as well.</td></tr>
</table>

Parameter	Description
	BranchId is not supported (it is always “Null”).
Enable historical alarms	Activates publication of historical alarm data
Enable trends	Activates publication of trend data.

Network

Parameter	Description
Node Name	Enter node name or leave empty to use the host name.
Port	The enter port number of OPC UA Server.
Produc URI	A globally unique identifier for the server.

Authentication

Parameter	Description
Anonymous	Anonymous clients accepted
User/Password	Authentication with user name is accepted
Certificates	Certificate-based authentication is accepted.

Server can support all options simultaneously.

For example, suppose there are 3 clients. Let Client 1 has only anonymous support. Client 2 has only user/password support. And, Client 3 has only certificate support. All three can connect if all checkboxs are checked in server config editor.

Server Identity

Parameter	Description
Manufacturer name	A human readable name for manufacturer of the product.
Product name	A human readable name for the product running the server..
Server's Certificate	Server certificate can be either generated automatically or by adding existing certificate files.

33 Special widgets

Widgets designed for special purposes are called special widgets and include control lists, date and time widgets, variable widgets and so on.

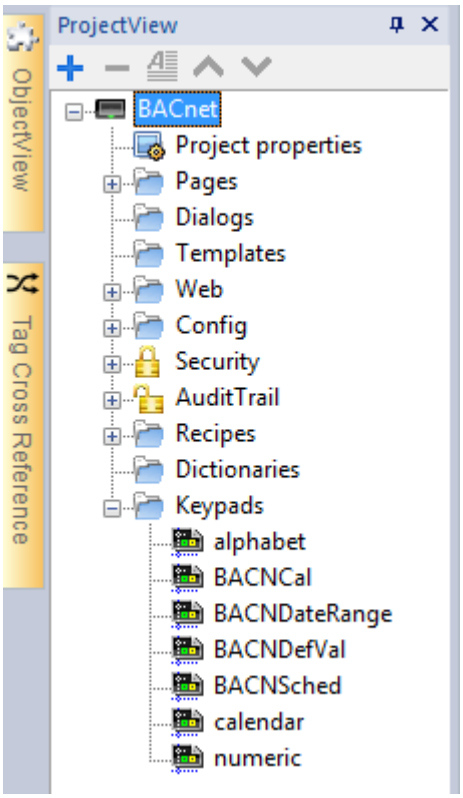
BACnet widget

Path: **Widget Gallery> BACnet**

BACnet widgets are special widgets that let you interact with native BACnet objects.

- BACnet Calendar
- BACnet Scheduler
- BACnet Effective Period

These widgets are using special keypads that are added into the keypads folder when widgets are used. Generally, you do not need to take care of these keypads unless you want customized them.



Refer to the BACnet manual inside the “Communication Drivers” folder for a detailed description of BACnet special widgets.

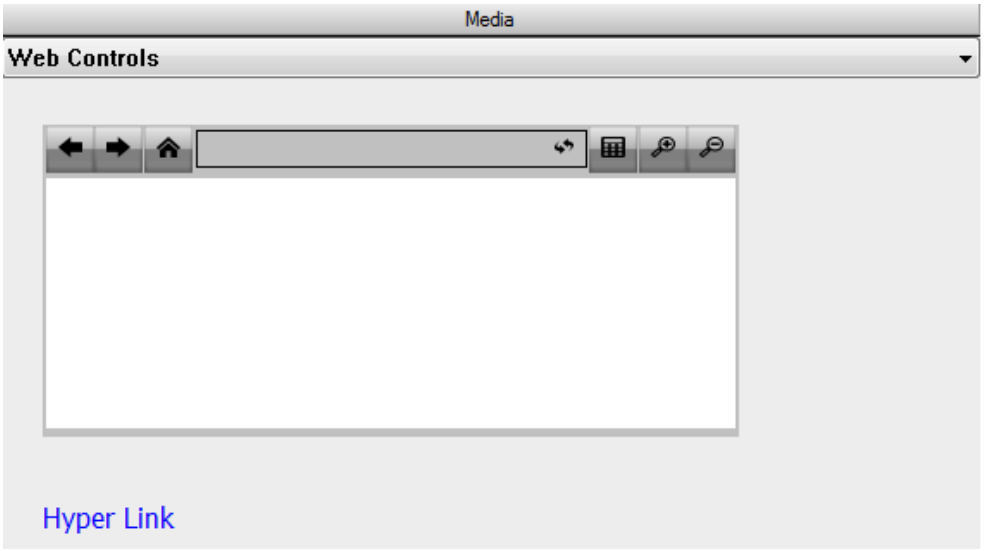
Browser widget

*Path: **Widget Gallery> Media> Web Controls***

Use this widget to embed web pages into your HMI device pages. This is an HTML5 compatible browser widget based on the WebKit engine.

Note: For Windows CE based embedded HMI devices, the WebKit library is available as a plugin (see "Software plug-in modules" on page 59 for details) to download to the HMI Runtime only when required.

Important: This widget is not supported by MIPS based devices.



Parameter	Description
Home Page	Default URL to open when widget is shown on the page.
Zoom to Fit	Automatically scales content to the size of view area.
Time out	Page load timeout in seconds.
Clear History	Automatic history clear on load
Scroll	Shows/hides scrollbars
Show Progress cursor	Shows/hides loading cursor

This allows you to save around 3 MB of space if the widget is not required in your project.

An **Hyper Link** widget is available to create pages hyperlinks. Once clicked these links notify to the browser widget that a particular web page is to be loaded.

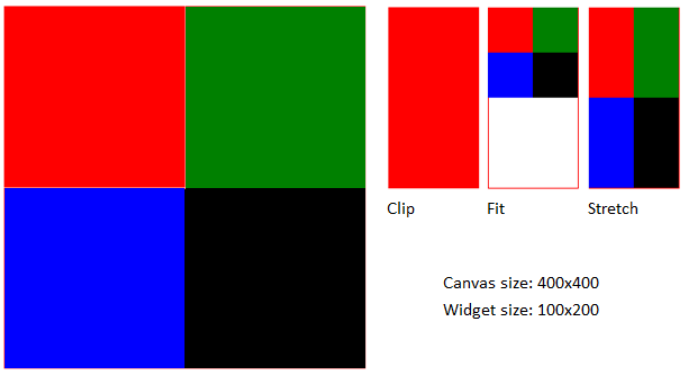
Important: HTTPs protocol is not supported.

Canvas Widget

Path: **Widget Gallery**> **Basic**> **Generic Canvas**

Canvas widget can be used to draw graphic via JavaScript scripting.

Note: the JavaScript methods are the same that are available for the HTML5 <canvas> tag

Parameter	Description
Canvas Width **Canvas Height**	Canvas size. Note this is not the widget size. For example, the canvas size could be 500x500 pixels where the widget size could be 100x100 pixels. Draw Hint parameter will define how to stretch the canvas size to fit the widget size.
Draw Hint	Define how fit the canvas inside the widget size • Clip No Transformation is applied, coordinate system is not scaled and drawing is clipped inside the widget bounding rectangle. • Fit to size Fit to the widget size preserving the canvas model aspect ratio. • Stretch Fit to the widget size ignoring the canvas model aspect ratio. Example using a Canvas size larger than the widget size: 
Design Time Preview	Canvas preview inside Wizard Note the JavaScript code could use data not available inside Wizard but only inside the HMI device
OnDraw Action	The OnDraw event is executed when the page is painted. This event has to be linked with the JavaScript code that draws the canvas graphic.
OnMousePress Action **onMouseRelease Actions**	Mouse events

Available Canvas Methods

// Painter Save/Restore

- void save(); // calls painter save
- void restore(); // calls painter restore

// Scale/Transform

- void scale(qreal x, qreal y);
- void rotate(qreal angle);
- void translate(qreal x, qreal y);
- void transform(qreal m11, qreal m12, qreal m21, qreal m22, qreal dx, qreal dy);
- void setTransform(qreal m11, qreal m12, qreal m21, qreal m22, qreal dx, qreal dy);

// Gradient

- CanvasGradient createLinearGradient(qreal x0, qreal y0, qreal x1, qreal y1);
- CanvasGradient createRadialGradient(qreal x0, qreal y0, qreal r0, qreal x1, qreal y1, qreal r1);

// Rectangle Functions

- void clearRect(qreal x, qreal y, qreal w, qreal h);
- void fillRect(qreal x, qreal y, qreal w, qreal h);
- void strokeRect(qreal x, qreal y, qreal w, qreal h);
- void rect(qreal x, qreal y, qreal w, qreal h);

// Path

- void beginPath();
- void closePath();
- void moveTo(qreal x, qreal y);
- void lineTo(qreal x, qreal y);
- void quadraticCurveTo(qreal cpx, qreal cpy, qreal x, qreal y);
- void bezierCurveTo(qreal cp1x, qreal cp1y, qreal cp2x, qreal cp2y, qreal x, qreal y);

// Drawing Text

- void fillText(const QString &text,qreal x, qreal y);

// Arc

- void arcTo(qreal x1, qreal y1, qreal x2, qreal y2, qreal radius);
- void arc(qreal x, qreal y, qreal radius, qreal startAngle, qreal endAngle, bool anticlockwise);

// Fill/Stroke

- void fill();
- void stroke();
- void clip();
- bool isPointInPath(qreal x, qreal y) const;

// Image manipulation (Draw CImageWgt using target and source rect)

- void drawImage(QObject *pObjImage, qreal sx, qreal sy,qreal sw, qreal sh, qreal dx, qreal dy,qreal dw, qreal dh);
- void drawImage(QObject *pObjImage, qreal dx, qreal dy);
- void drawImage(QObject *pObjImage, qreal dx, qreal dy, qreal dw, qreal dh);
- void drawImage(const QVariant& image, int width, int height, const QString& format, qreal sx, qreal sy,qreal sw, qreal sh, qreal dx, qreal dy,qreal dw, qreal dh);

// Pixel manipulation

- ImageData createImageData(double sw, double sh);//Empty Image
- ImageData createImageData(ImageData fromImage);//from another Image
- ImageData createImageData(ArrayBuffer value); //From arraybuffer
- void putImageData(ImageData imgData,double dx, double dy);
- void putImageData(ImageData imagedata, double dx, double dy, double dirtyX, double dirtyY, double dirtyWidth, double dirtyHeight);
- ImageData getImageData(qreal sx, qreal sy, qreal sw, qreal sh);

Canvas JavaScript Example

The canvas is initially blank. To display something, a script first needs to access the rendering context and draw on it:

```
var ctx = me.context2d;
```

then you can use the canvas methods, as in the below example

```
function GenericCanvasWgt1_onDraw(me, eventInfo)
{
    var ctx = me.context2d;
    ctx.fillStyle = 'red';
    ctx.fillRect(0,0,250,250);
    ctx.fillStyle = 'green';
    ctx.fillRect(250,0,250,250);
    ctx.fillStyle = 'blue';
    ctx.fillRect(0,250,250,250);
    ctx.fillStyle = 'black';
    ctx.fillRect(250,250,250,250);
}

function GenericCanvasWgt1_onMouseDown(me, eventInfo)
{
    alert("X = " + eventInfo.posX + "\nY = " + eventInfo.posY );
}
```

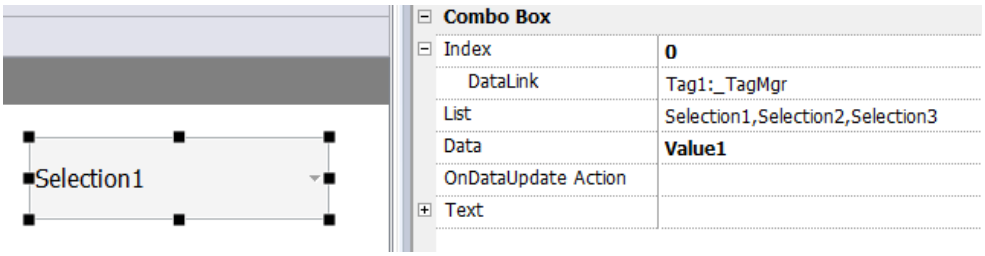
The update method can be used to dynamically redraw a canvas widget

```
function BtnStd1_btn_onMouseClick(me, eventInfo)
{
    var myCanvasWidget = page.getWidget("GenericCanvasWgt1");
    myCanvasWidget.update()
}
```

Combo Box widget

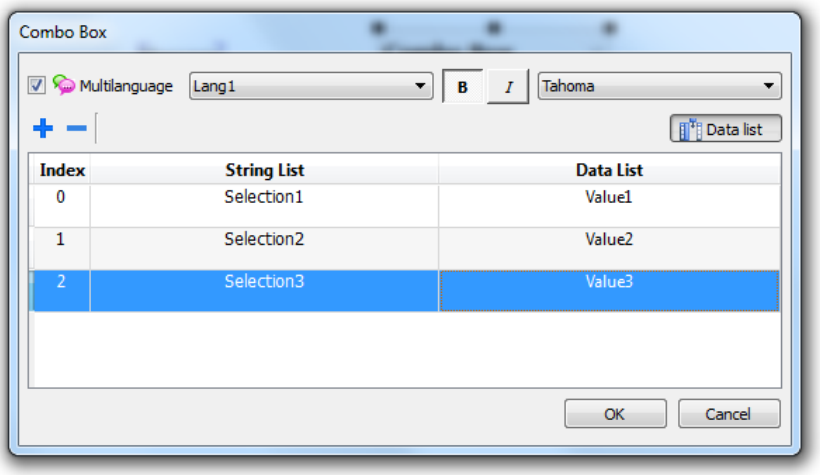
Path: ***Widget Gallery> Basic> Controls***

Use this widget as a selector widget or to filter rows in a table to display only the values selected in the combo box.



Parameter	Description
Index	Index of the selected item.
List / String List	Item strings in the combo box. Note: This field is multi-language.
Data / Data List	Returns the value in the Data List column (as string) in the Data field of the widget. Tip: Use this parameter to return a custom value based on an item selected in the combo box.
Text	Format of displayed text.

Attaching data vs. attaching indexes



In many projects you may need to attach fields such as **Index** or **Data** to tags to know the values of the selected item in the combo box. Use:

- **Index**: to display the index (integer) of the selected item (0…n).
- **Data**: to display the data value (string) specified in the Data List column.

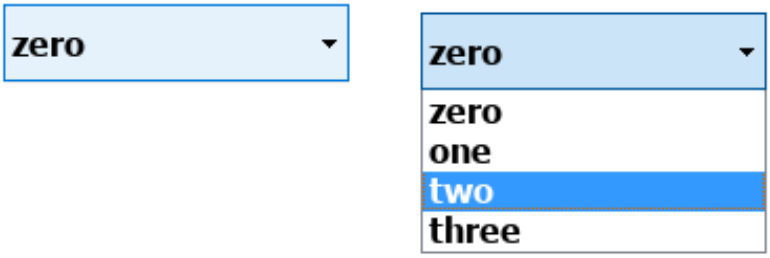
Combo Box widget “full screen” mode with images

From the "Project properties pane" on page 54 the look and behavior of Combo Boxes can switches from Context mode to Full Screen mode

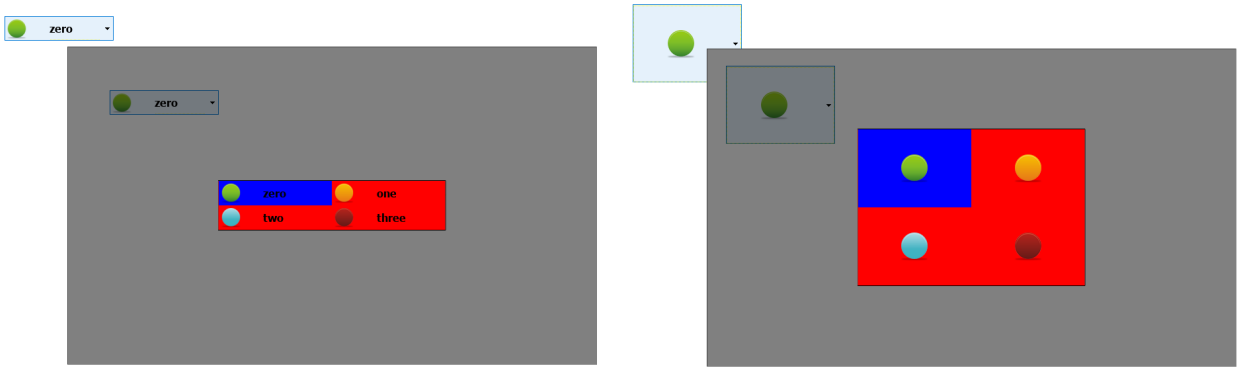
Path: **ProjectView**> double-click **Project properties**> **Properties pane**> **Style**> **ComboBox View Mode**

Parameter	Description
ComboBox View Mode	Select the visualization mode of all the Combo Box widgets of the project **Context** Classic view with drop-down menus **Full screen** Enhanced view with configurable texts and images that will pop up in the middle of the screen for easy scroll and selection.

Context view example

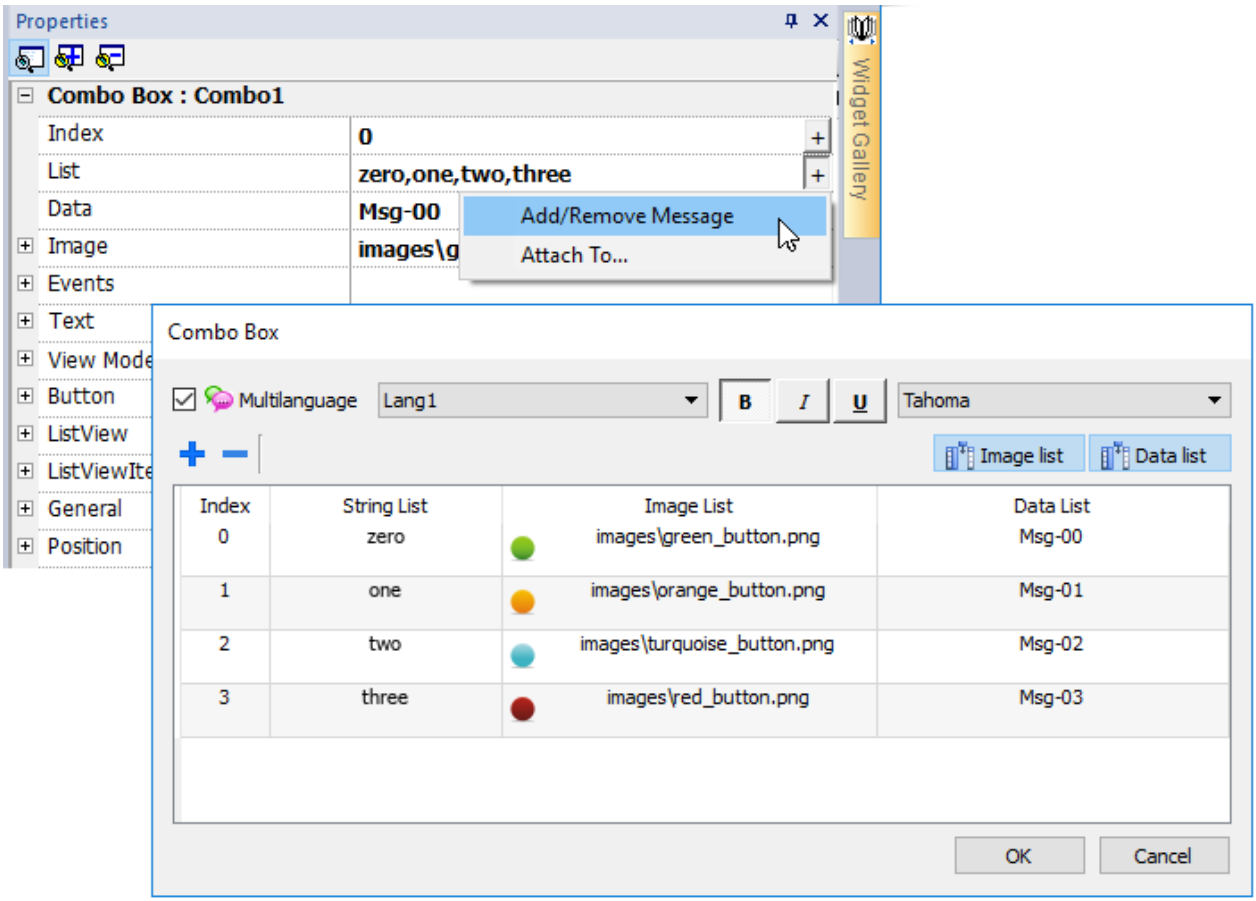


Full screen view example



Additional parameters available in full screen mode

The additional "*Image List*" column will be available inside **Combo Box**> **List** parameter:



Note: Some properties are displayed only in advanced mode.

Parameter	Description
Image	Return, inside the attached tag, the file name of the selected image
Button	Define the look of the Combo Box • Show background = true Combo Box button is showed • Show background = false Only image or text is showed
ListView	Layout parameters of the Combo Box in edit mode
ListViewItems	Define the items type that will be inside the Combo Box Image Mode: • Only Text • Only Images • Text and Images

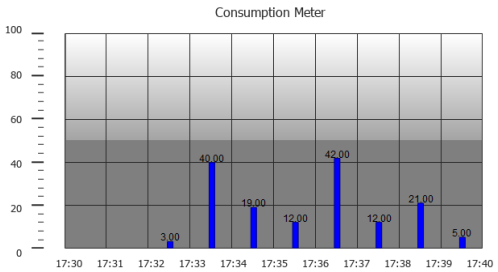
Consumption Meter widget

Path: ***Widget Gallery> Basic> Trends/Graphs***

Use this widget to monitor a resource which is continuously increasing. The system reads the value of the resource and calculates the increment in a set range of time, the increment is then displayed in a bar-graph in a trend-like window.

Different colors can be used to used in the graph based on the time frame.

Tip: Use this widget to calculate the power consumption of a system.

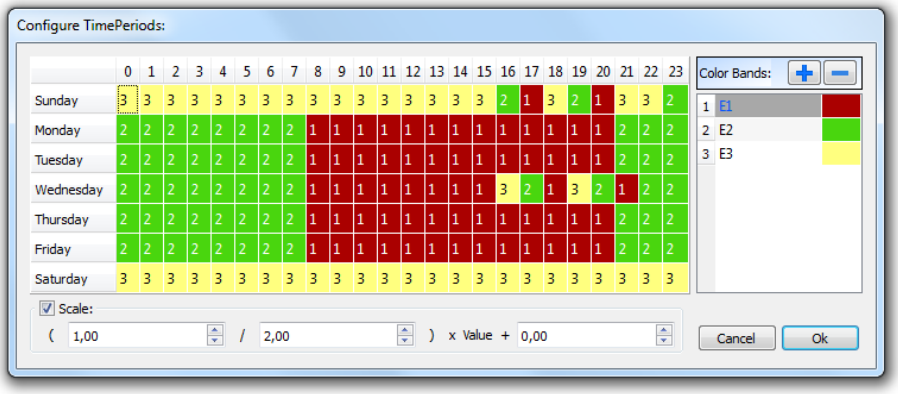


Parameter	Description
Value	Resource monitored
Graph Duration **Graph Duration Units**	Time period displayed in the window
Bar Duration **Bar Duration Units**	Time period represented by each bar in the graph
Time Periods	Assigns a specific color to highlight the increment of the monitored resource in a specified time period (minimum resolution = 1 hour).
Color **Bar Width**	Bar color and width
Bar Value	Show/Hide the value of each bar
Consumption Meter	Number of labels to be displayed on graph.

Example: how to monitor energy consumption

In the following example a widget is design tho monitor energy consumption with a weekly scale and a daily unit.

1. Attach a tag to the physical variable to monitor. In this example, to the total energy consumed (Tag KWh). This tag contains an incremental number that indicates how many KW/h have been consumed from when energy consumption started.
2. Add a Trend and link it to the tag to be monitored, Tag KWh.
3. Add a **Consumption Meter** widget to a page.
4. Attach the **Value** property of the Consumption Meter to the Trend you created in step 2.
5. Set **Graph Duration/Units** to 1 week: this will give you a weekly graph of consumed energy.
6. Set **Bar Duration/Units** to 1 day, this is the time range when energy consumption is calculated.
7. In **Consumption Meter** set the number of labels to show in the bar graph, in this case 7 to display a weekly graph.
8. From the **Time Periods** property open the **Configure Time Periods** dialog: set the different colors for different values of Tag KWh in each bar.



Tip: To assign the color to the cells of the table, select the cells and click on the desired color, or enter the index value of the band (1, 2, 3) into the cell.

9. Add as many color bands as you need, in this example 3 color bands.
10. Assign a band to each hour in the weekly table, in this example a red band (E1) is used to indicate the range of time in the day/week where the cost of energy is the highest.

Note: You can apply a scale factor to each color band, if needed.

The result is a bar graph consumption meter showing daily consumption of energy in KW/h, with colors indicating the different energy costs. The height of each bar represents the amount of energy in the time range considered, 1 day in this example.

Use the action ConsumptionMeterPageScroll to scroll the bar graph back and forth and the action RefreshTrend to refresh the bar graph since data is not refreshed automatically.

Important: No other Trend action is currently supported by the Consumption Meter widget.

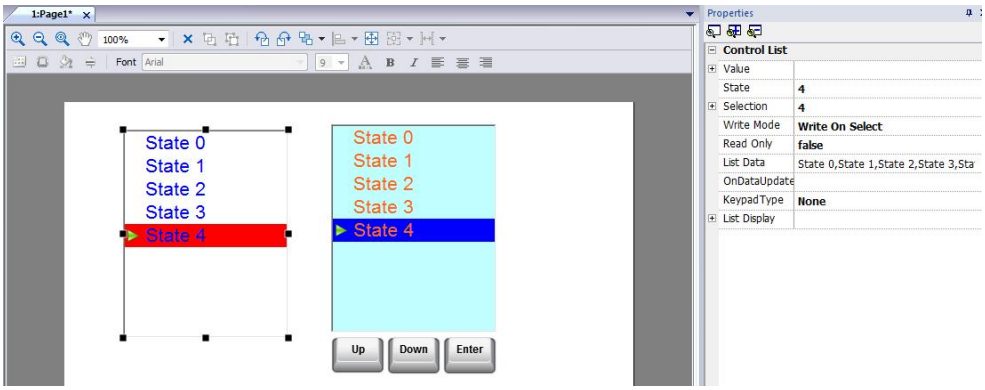
Control list widgets

*Path: **Widget Gallery> Advanced> Control List***

Use these widgets to represent the status associated with a particular process and to control that process from the same widget.

Two types of control lists are available:

- a group control list, with a limited set of navigation button already included, and
- a basic control list with no pre-configured button to be navigated using the touch screen feature.

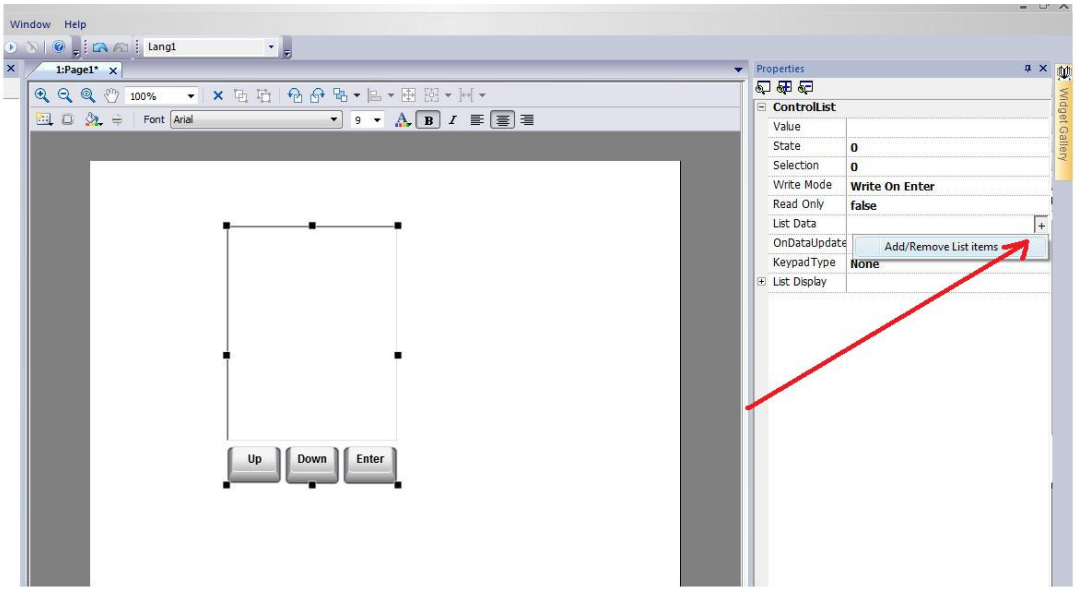


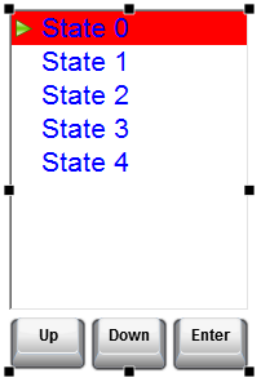
Parameter	Description
Value	If **Write mode** is **Write On Select**: value of the item selected. If **Write mode** is **Write On Enter**: value of item selected and confirmed pressing enter button. This field can be attached to a tag to control selected and confirmed item.
State	Default state when widget is loaded.
Selection	Currently selected item, displayed as a highlight cursor moving up and down. This property can be attached to a tag.
Write Mode	**Write On Select**: the value is automatically written to the tag when one of the items is selected. **Write On Enter**: the value is written to the tag only when one of the items is selected and the enter key is pressed.
Read Only	Defines whether the list is only an indicator.
List Data	Adds/removes list items.

Defining states

Add/remove states, that is items in the list, from the **List Data** property.

Any value can be assigned to a state. When you activate the state, by selecting the related item if in **WriteOnSelect** mode or selecting it and confirming with enter if **Write On Enter**, this will write the value assigned to state to the tag linked to the Control List widget **Value** .



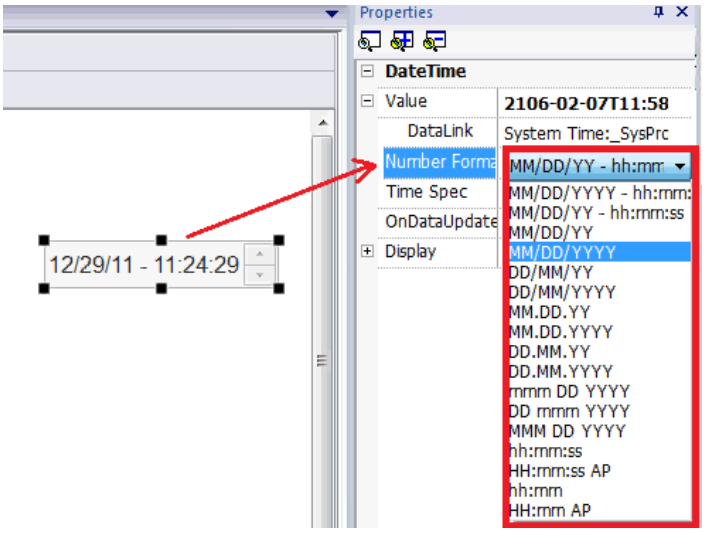


DateTime widget

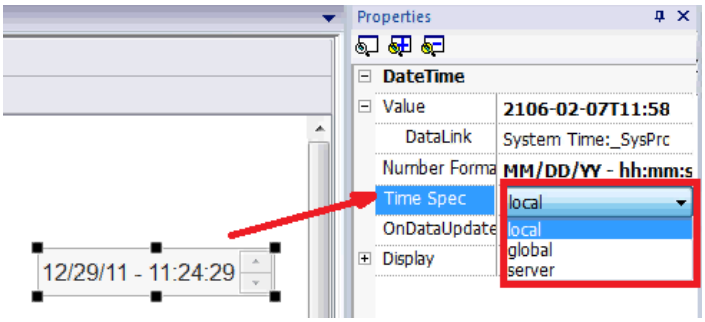
Path: ***Widget Gallery> Basic> Controls***

Use this widget to display and edit current date and time .

In the **Properties** pane different formats are available for representing date and time.



For the **Time Spec** property select which time the widget will show at run time.



Time options

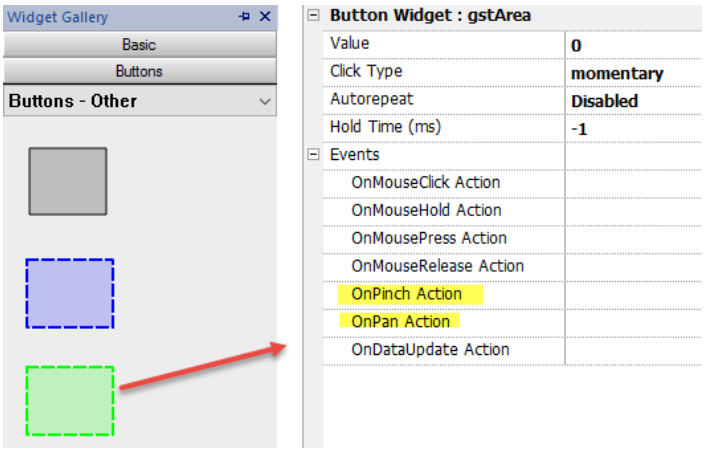
Option	Description
local	shows local time, the time of the HMI device where the project is running
global	shows Global Time (GMT)
server	shows time information as handled by the server side of the HMI device

Gesture area widget

Path: ***Widget Gallery> Buttons> Others***

Gesture Area Widget is a hotspot button that generates pan and pinch gesture events.

WARNING: Only multi touch HMI devices can generate pinch events



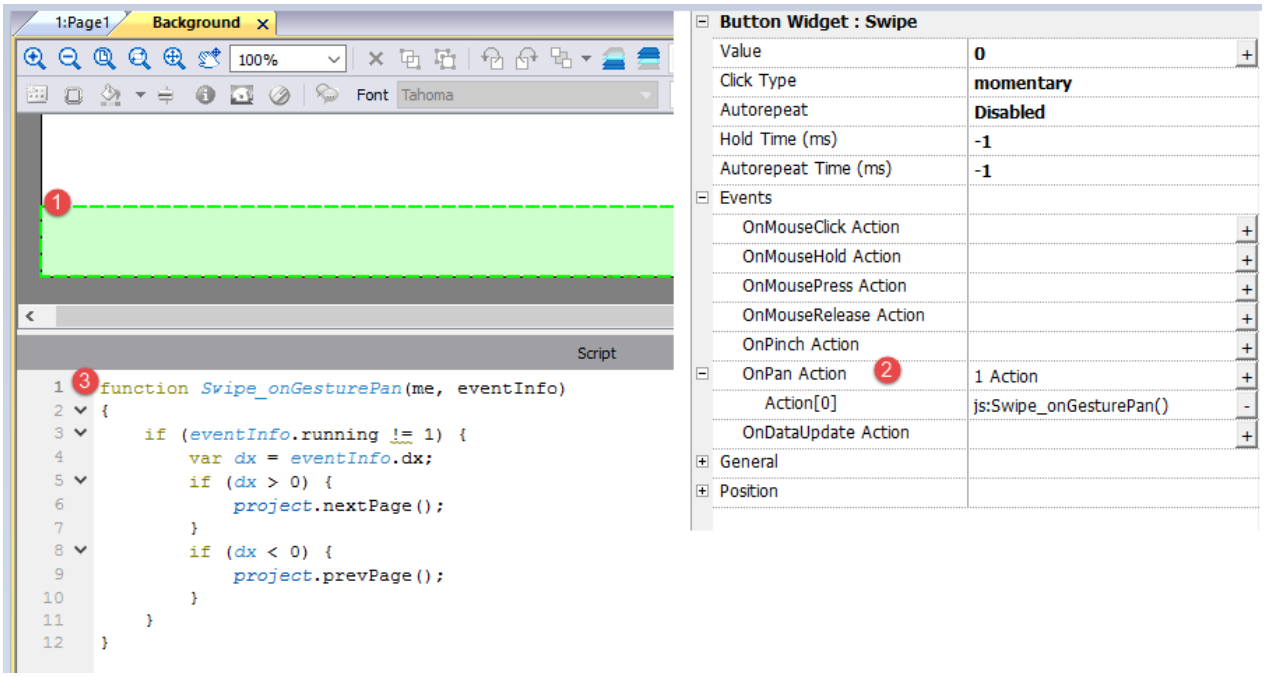
Use OnPan and OnPinch events in association with JavaScript code to identify gestures and program the requested actions

See "Widget events" on page 334for details on these event types

Swipe Gesture

How to recognize a "swipe" gesture to change page in the application.

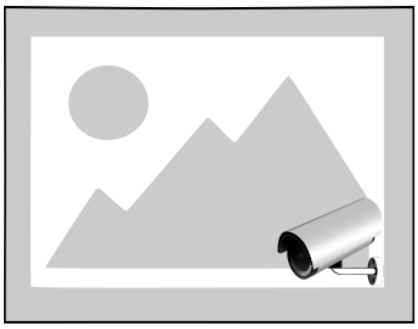
1. Put a Gesture area widget into the page
2. Configure the OnPan Action to trigger a JavaScript function
3. Write the JavaScript code that recognize the swipe gesture



IP Camera widgets

Path: ***Widget Gallery> Media> IP Camera***

Use these these widgets to show images captured from an IP Camera or a video stream.



Parameter	Description
Camera URL	URL of the IP Camera when used in JPEG format.
Refresh Rate	Number of JPEG images for second allowed. Max rate = 1 fps.
User Name	Name of user allowed to access the camera. Set this parameter when access to the camera is password protected.
Password	Password to access the camera.
MJPEG Camera URL	URL of MJPEG streaming (for example, http://192.168.0.1/video.cgi)

When this widget is used to stream HTTP MJPEG, **Camera URL** and **Refresh Rate** are ignored.

Performance of streaming is not fixed and depends on many factors such as: frame size, frame compression level, CPU of HMI device, quality of IPCamera. Based on these factors the widget can reach up to 25 fps.

You can add multiple IP Camera widgets, but this will reduce the frame rate for each widget.

Supported IPCameras

The following IP Cameras have been tested so far:

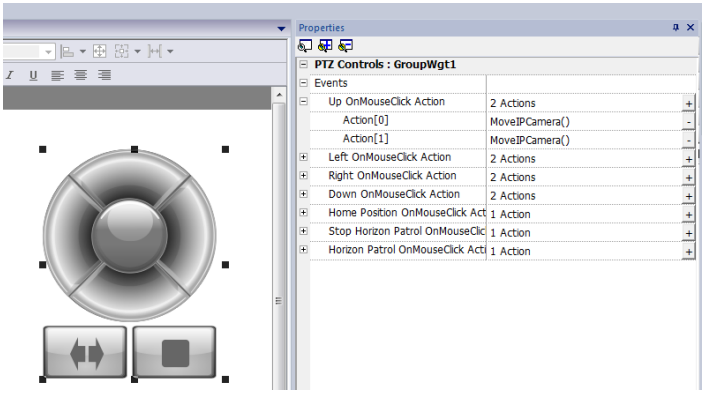
IPCamera	Protocol	URL
Apexis APM-J901-Z-WS PTZ IP Camera	MJPEG HTTP	http://{ip_address}/videostream.cgi http://{ip_address}/snapshot.cgi
AXIS M3027-PVE Network Camera	MJPEG HTTP	http://{ip_address}/axis-cgi/mjpg/video.cgi http://{ip_address}/axis-cgi/jpg/image.cgi
DAHUA DH-IPC-HD2100P-080B 1.3mp Outdoor Vandalproof	HTTP	http://{ip_address}:9988/onvif/media_service/snapshot
D-Link DCS-5605 PTZ	MJPEG	http://{ip_address}/video/mjpg.cgi
D-Link DCS-900W IP Camera	MJPEG	http://{ip_address}/video.cgi
D-Link DCS-932L	MJPEG	http://{ip_address}/video.cgi

IPCamera	Protocol	URL
Edimax IC-7100P PTZ	MJPEG HTTP	http://{ip_address}/mjpg/video.mjpg http://{ip_address}/picture.jpg
Foscam FI8916W	MJPEG HTTP	http://{ip_address}/videostream.cgi http://{ip_address}/snapshot.cgi
Foscam FI9803 EP	MJPEG	http://{ip_address}:88/cgi-bin/CGIStream.cgi?cmd=GetMJStream&usr={user}&pwd={pass} NOTE: • port 88 may be different as per IP Camera settings • {user} = username defined into IP Camera settings • {pass} = password defined into IP Camera settings
Hamlet HNIPCAM IP Camera	MJPEG HTTP	http://{ip_address}/video.cgi http://{ip_address}/image.jpg
MOXA VPort 254 (Rugged 4-channel MJPEG/MPEG4 industrial video encoder)	MJPEG HTTP	http://{ip_address}/moxa-cgi/mjpeg.cgi http://{ip_address}/moxa-cgi/getSnapShot.cgi?chindex=1
NVS30 network video server	MJPEG HTTP	http://{ip_address}:8070/video.mjpeg http://{ip_address}/jpg/image.jpg
Panasonic WV-Series Network Camera	MJPEG	http://{ip_address}/cgi-bin/mjpeg
Ubiquiti UniFi Video Camera	HTTP	http://{ip_address}:7080/images/snapshot/camera/{camera_guid}?force=true NOTE: • {camera_guiID} can be found into IP Camera Webpage • port 7080 may be different as per IP Camera settings
Zavio F3210 2MP Day & Night Compact IP Came	MJPEG HTTP	http://{ip_address}/stream?uri=video.pro3 http://{ip_address}/cgi-bin/view/image?pro_0 NOTE: • MJPEG video streaming can be configured selecting "video profile 3" with 640x480 resolution into IP Camera settings.

PTZ Controls widget

PTZ (pan–tilt–zoom) cameras are cameras capable of remote directional and zoom control.

The PTZ Controls widget uses the MoveIPCamera action to send HTTP/cgi commands to the PTZ IP Camera.



Parameter	Description
Camera URL	URL of IP Camera
User Name	Name of user allowed to access the camera. Set this parameter when access to the camera is password protected.
Password	Password to access the camera.
Command	Command to send to the PTZ controller (for example, decoder_control.cgi?command=0)

Authentication methods

The authentication method is automatically set by the camera web server to which the widget connects. Authentication methods supported are:

- Basic
- NTLM version 1
- Digest-MD5

Javascript function block widget

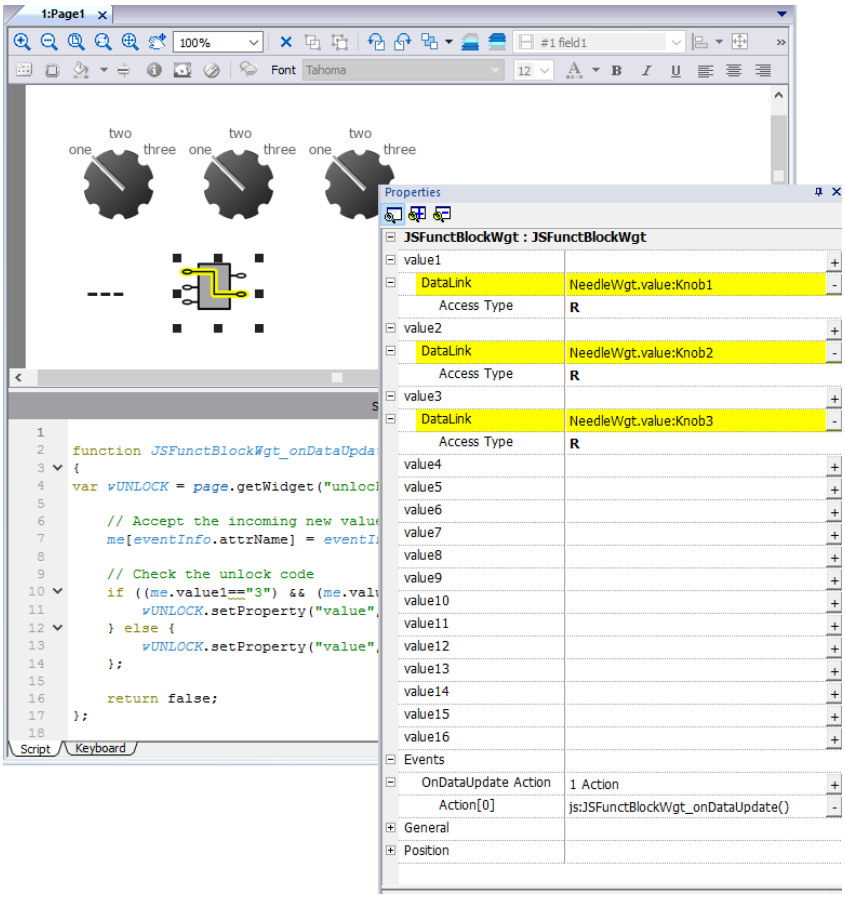
Path: ***Widget Gallery> Basic> JSFunctionBlock***

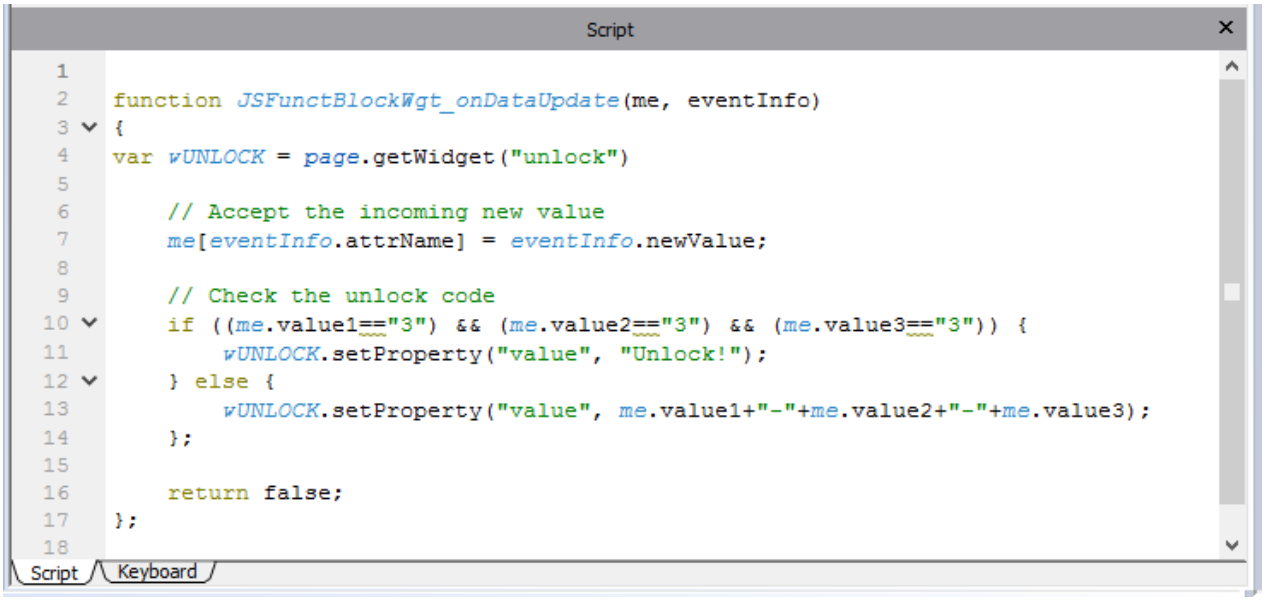
Javascript Function Block is a widget that contains Javascript logic that is executed when tags values change.

Parameter	Description
value1 **...** **value16**	Objects that will trigger the OnDataUpdate action.
OnDataUpdate	Action that will be executed when a change of an associated value is detected

Note: This widget is rendered only in Wizard, and it is not rendered in the HMI device.

Example:
A Javascript code that check the combination lock of three selectors



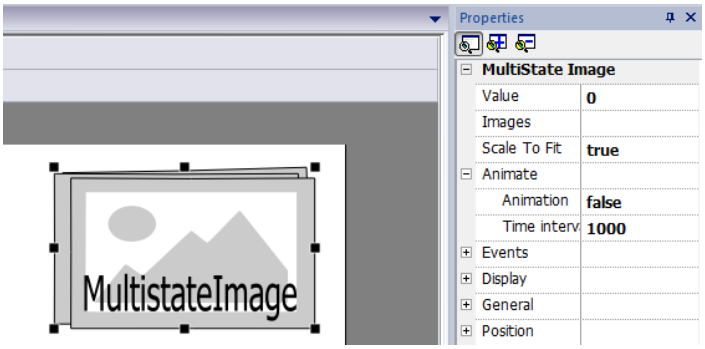


See "Widget events" on page 334for the description of the onDataUpdate parameters

Multistate Image widget

Path: ***Widget Gallery> Basic> Images***

Use this widget to display an image from a collection based on the value of a tag used as Index. You can use this widget also for simple animations.

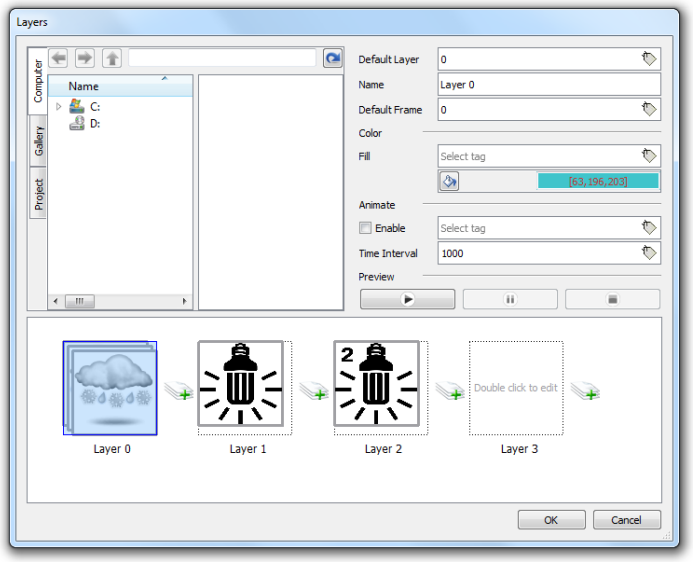


Parameter	Description
Value	Index of image to display. For example, set Value=0, to display the image with index 0 in the image collection.
Images	Images collection with associated index.
Animate	Set to true, to enable a slide show.
Time interval	Interval between images in the slide show.

Multistate Image Multilayer widget

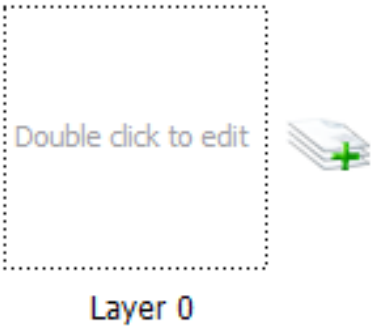
Path: ***Widget Gallery> Basic> Images***

Use this widget to create different animations and select the most suitable at run time.

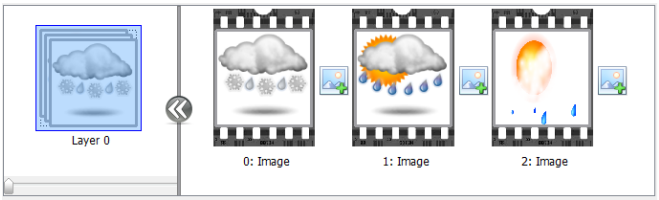


Setting up widget layers

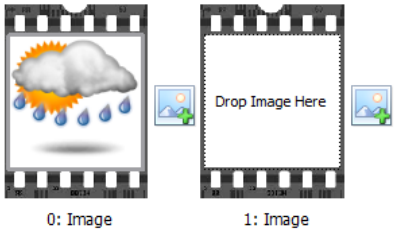
1. Open the **Layers** dialog from the **Properties** pane.
2. Click **+** to add as many layers as you need.



3. Double click on each layer to add as many images as you want to include in the layer.



4. Drag and drop images into the frame to add it to current layer.



5. Define widget properties.

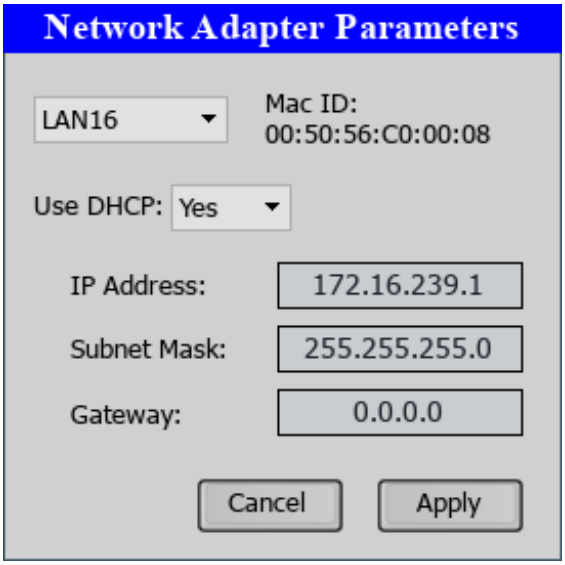
Parameter	Description
Default Layer	Layer shown at run time.
Name	Name of selected layer.
Default Frame	Frame shown when current layer is displayed.
Color / Fill	Fill color for images of current layer.
Animate	Enables slide show for active layer. Animations can be started/stopped at run time attaching it to a tag.
Time Interval	Time interval of slide show, if enabled.
Preview	Slide show simulation.

Note: **Default Layer**, **Default Frame**, **Color** and **Fill** can be changed at run time, attaching the to a tag.

Network Adapters widget

Path: ***Widget Gallery> Basic> Control***

Use the IP Widget to set the network adapters parameters.



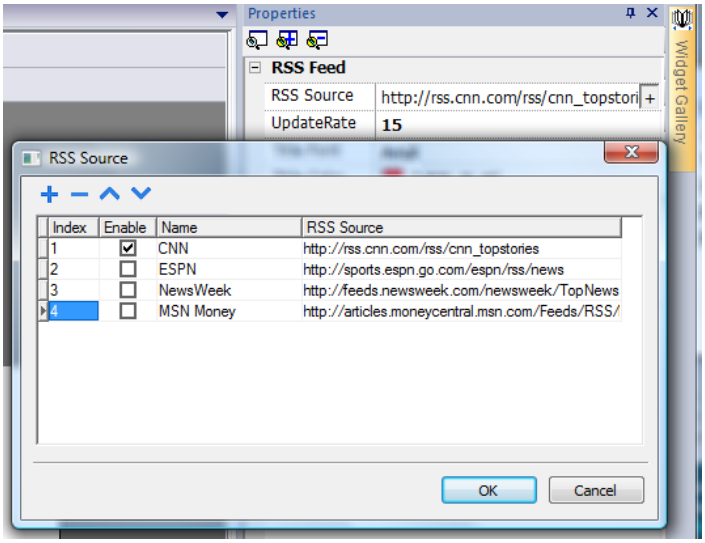
The system variable Network->Status contains the result of the last operation performed by the IP Widget (see "Network variables" on page 86for details)

RSS Feed widget

Path: ***Widget Gallery> Media> RSSFeed Source***

Use this widget to display on the HMI device your favorite RSS feeds directly from the Internet.

Parameter	Description
RSS Source	Feed URL Note: Feed sources cannot be modified at run time.
UpdateRate	Refresh time



The RSS Feed widget has been specifically designed to work with Pocket Internet Explorer.

Scrolling RSS Feed widget

Path: ***Widget Gallery> Media> RSSFeed Scroll***

Use this version of the main RSS Feed widget to display highlights inside a text line using a smoothing scrolling text.

RSS Scroll Widget : RSSScrollWgt	
RSS Source	http://rss.cnn.com/rss/cnn_topstories
UpdateRate	15
Title Separator	
Title Font	Tahoma
Title Color	[23, 30, 40]
Title Size	12
Scrolling	Normal

This widget has additional properties.

Parameter	Description
Scrolling	Scrolling speed
Title Separator	Separator character between highlights

Table widget

Path: **Widget Gallery> Basic> Table**

Use this widget to create a table with data provided from a data source.

To configure a table:

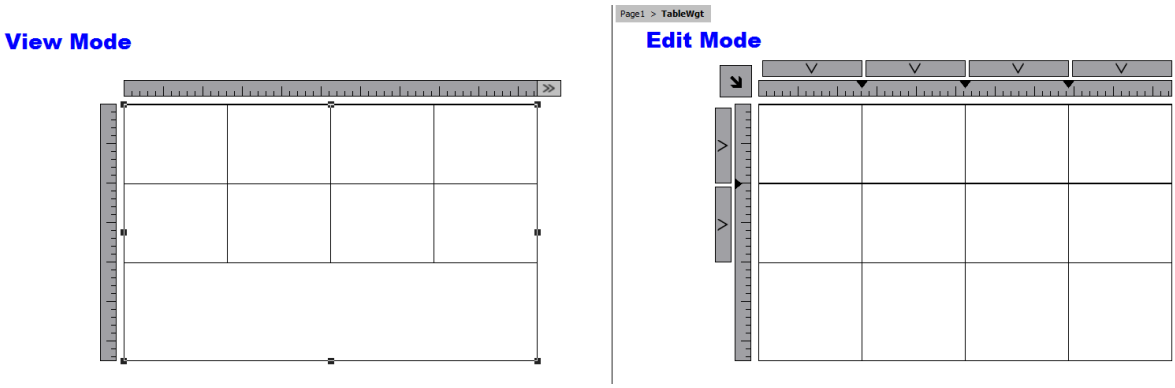
1. Put a table widget on the screen and configure the template of the table.
2. Add widgets into cells to configure one or more rows that will be used as row templates when the table will be filled with data provided from the data source.
3. Select a data source that will be used to fill the rows of the table
4. Define the links from widgets and data source.

Configure the table widget

Table widget has two states:

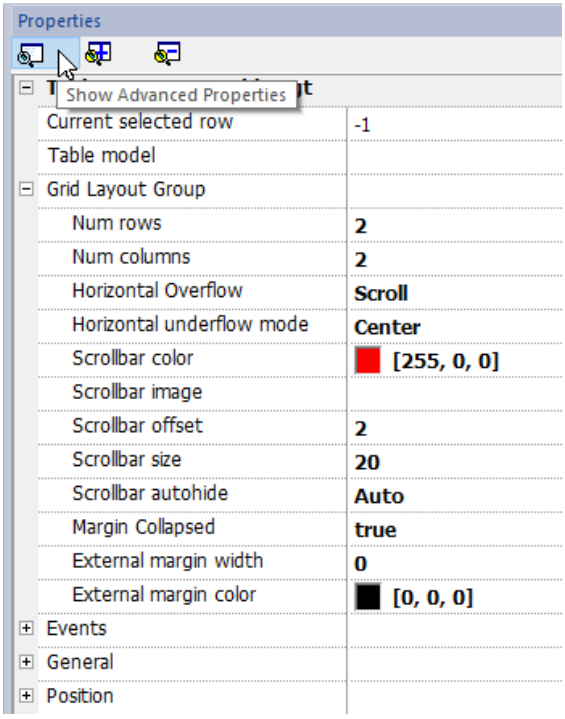
- View mode
- Edit mode.

Click on the table to manage the widget in view mode, double click to enter in the edit mode. To exit and return to view mode click outside the table.



View Mode

In view mode, you can configure the table layout. Drag and drop the table onto the page, resize the table, define number of template rows, number of columns and the main table properties.

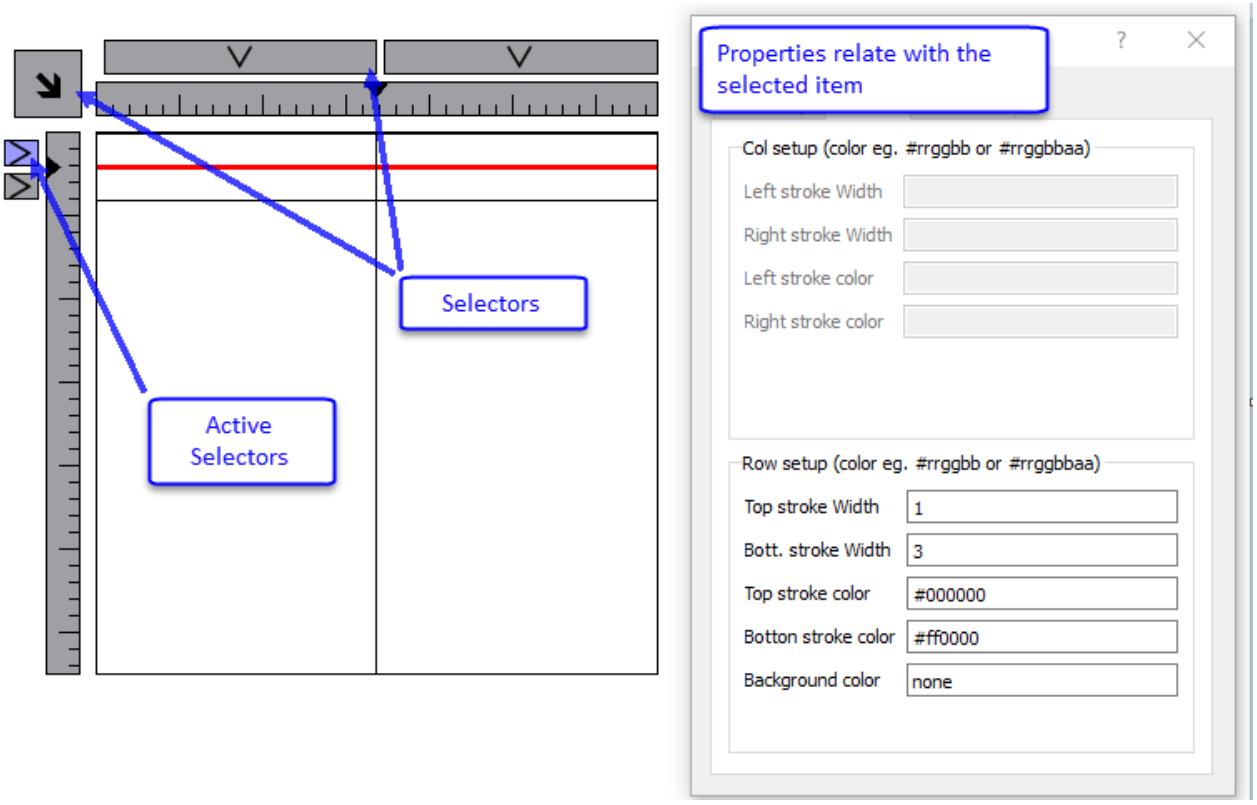


The "*Grid Layout Group*" properties are visible only in advance view. See "Grouping widgets" on page 42 for details.

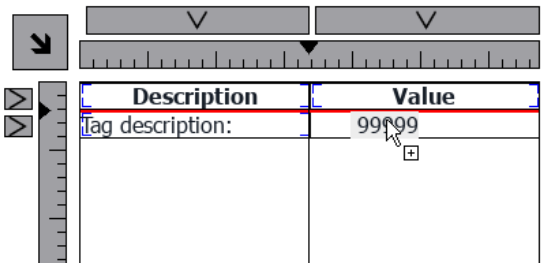
Edit Mode

In edit mode, it is possible configure the format and the content of each cell of the table. Each row of the table will act as a row template.

To configure the look of the table, click on table's selectors to select the item to configure.



To configure the contents of cells, drag and drop the widgets inside the cells.



If you need more widgets inside a single cell, create a group of widgets and copy the group from the page to the cell.

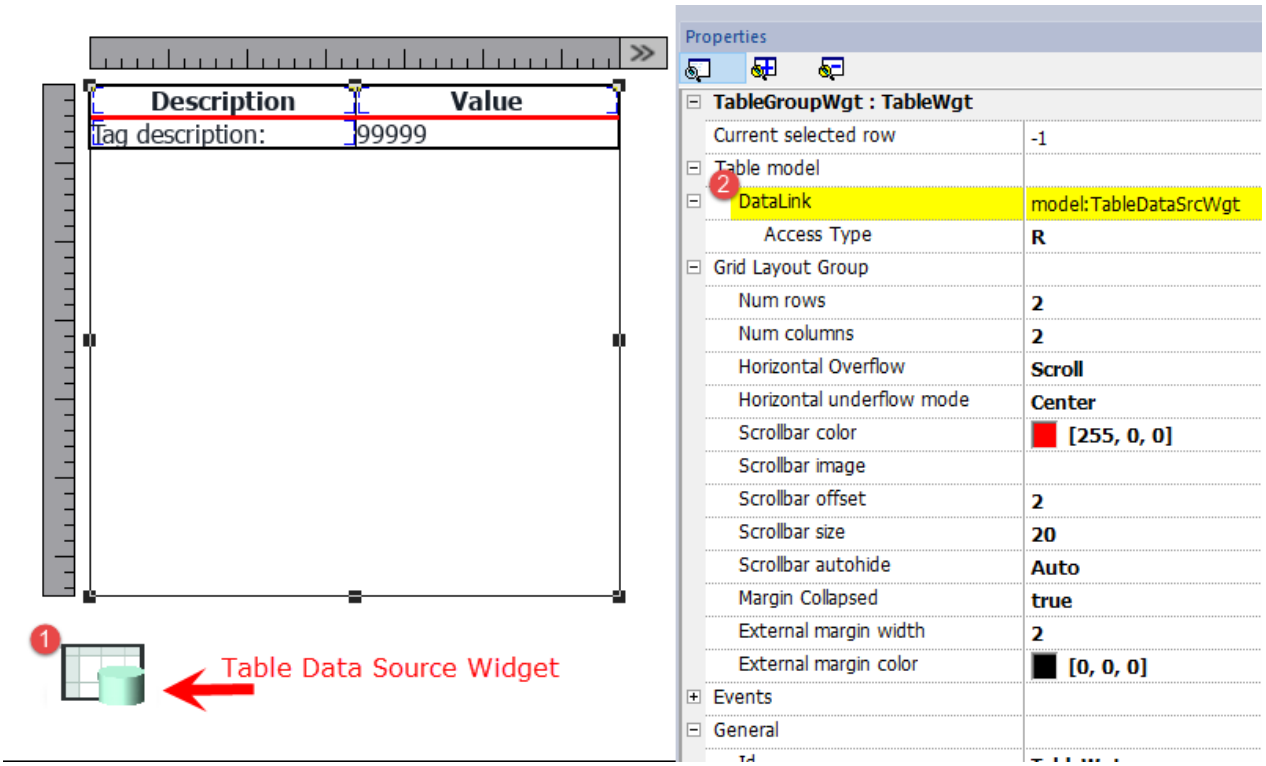
Configuring the data source

The data source, that provide the data to fill the table, could be a Table Data Source Widget or a JavaScript JSON table.

Table Data Source Widget

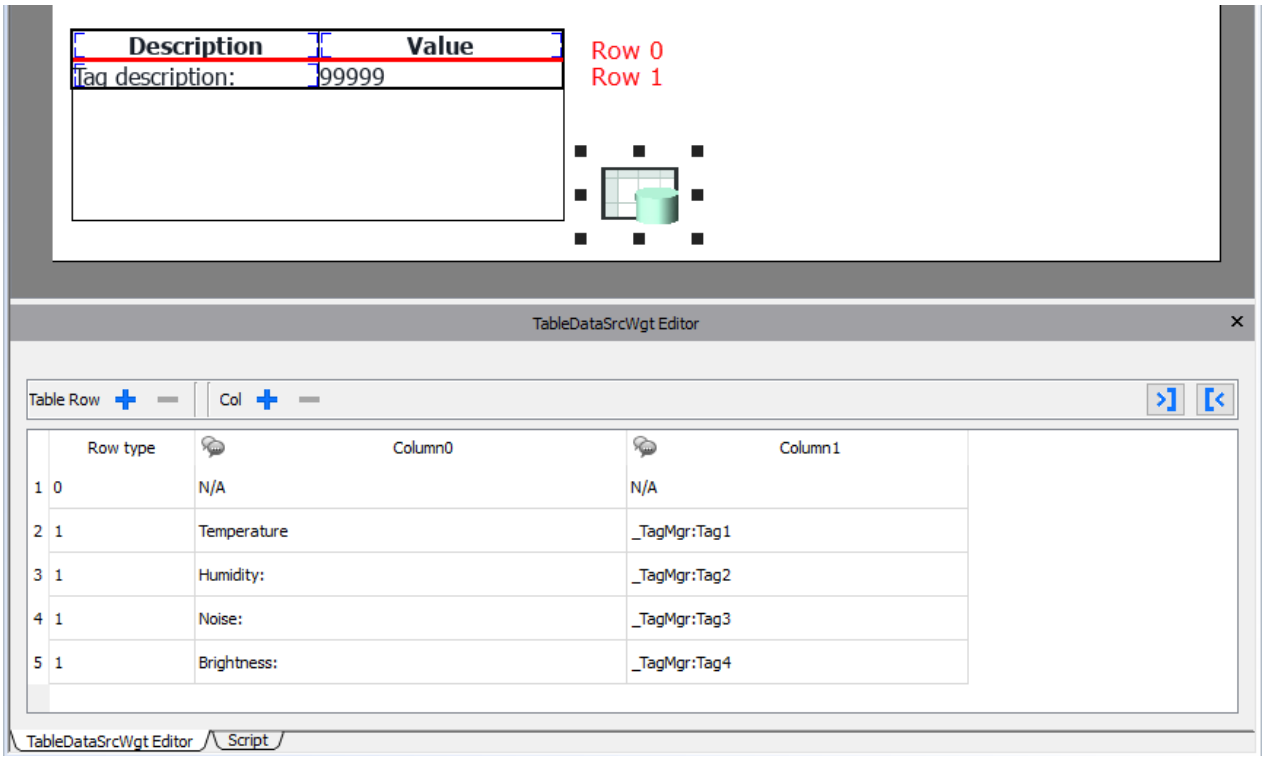
Path: **Widget Gallery> Basic> Table**

1. Drag and drop a *Table Data Source Widget* onto the page
2. Set the *Table Model* parameter to link at the data source.



Select the Data Source and inside the TableDataSrcWgt Editor add the rows and columns that are needed. In the following example, we have defined two row templates:

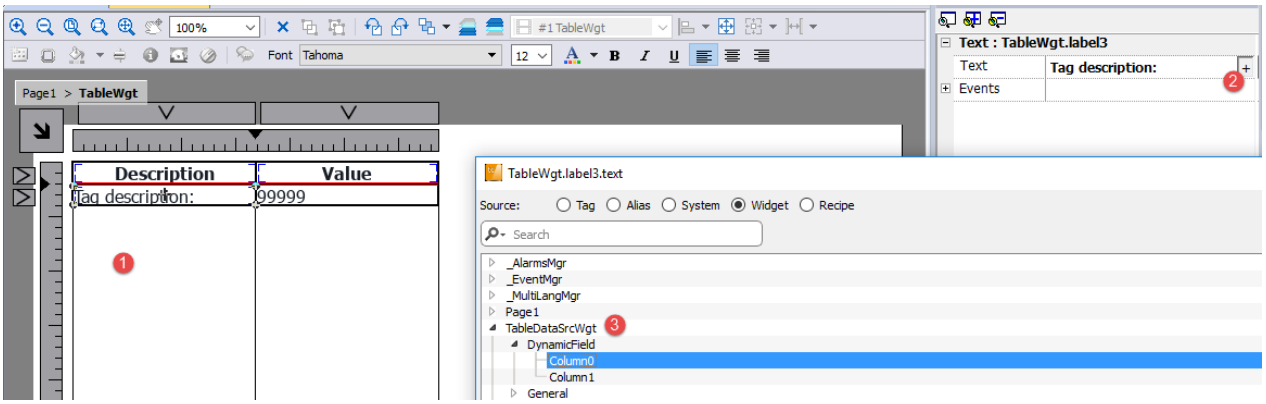
- Row 0
  Header of the table. Contains only static text.
- Row 1
  Template of rows with data. On the first column we added a label that will contain the description and on the second column a field that will contain the value.



Each row must be assigned a row type. The row will take on the format of the corresponding row template. Widgets that were placed in each cell of the row template will appear in rows of that type.

Define links with data source

1. Double click over the Table widget to enter in edit mode and select a widget
2. Select the property that is to be read from the data source
3. Select the column of the data source that will provide the data



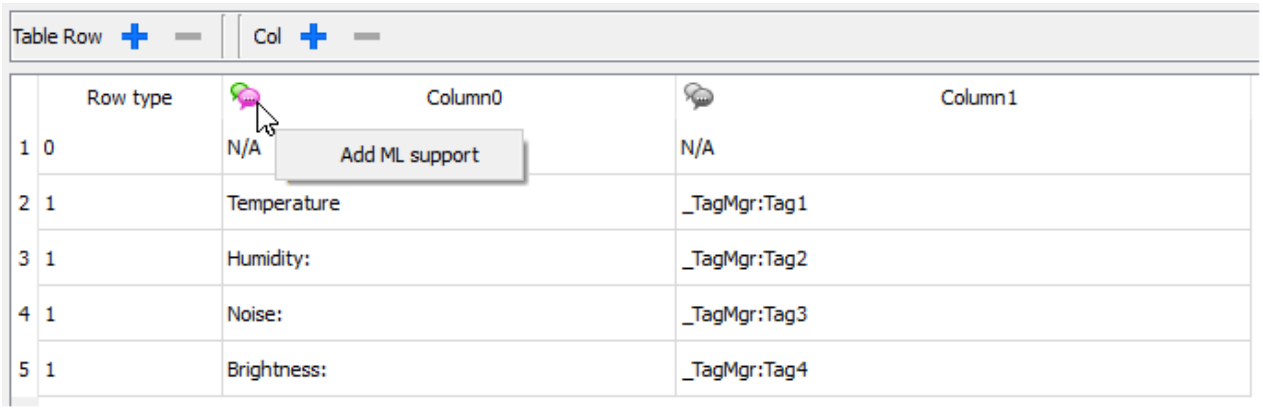
The below picture is showing how our example will be rendered at runtime

Description	Value
Temperature	111
Humidity:	222
Noise:	333
Brightness:	444

Multilanguage

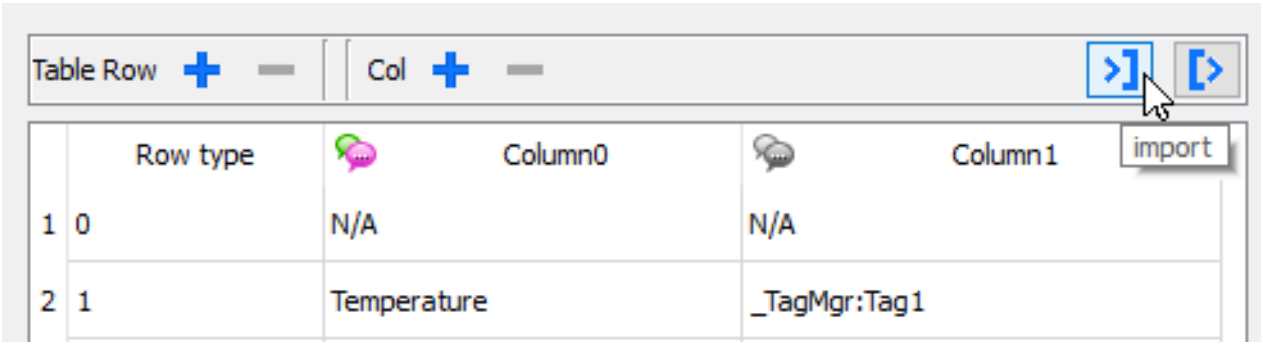
To enable the Multilanguage support right click on the Multilanguage icon of the column. The icon will change color to indicate that the support is enabled.

Avoid enabling the Multilanguage support when not necessary to better performance.



Import/Export Data Source

The configuration of the Data Source can be imported/exported using xml files



JavaScript JSON table

In alternative to the Data Source Widget, for data to fill the table could be provided from a JavaScript code using a JSON table. In this case, we have to fill an array of JSON elements with the data to use and assign the array to the table widget.

```
var myTable = page.getWidget("TableWgt1");
```

```
myTable.model = model;
```

model is an array of JSON elements with the table definition and data. The first element of the array will contain the template of the rows while the other elements will contain the data to fill in the rows of the table

```
model[0] = row_templates;     // row templates
model[1] = row_data1;         // data of the row1
model[2] = row_data2;         // data of the row2
model[3] = row_data3;         // data of the row3
model[4] = row_data4;         // data of the row4
model[5] = row_data4;         // data of the row5
```

The **row templates** is a multi dimensional array where each array defines the datalink of one template row.

On the below example, we have a template for two rows.

```
var row_templates = {
    _h : [
            [ []       , []        ],   //rowType = 0
            [ ["text"] , ["value"] ]    //rowType = 1
        ]
}
```

The first row has two columns that do not contain data links. We use this template for the header on the first row of our table.

The second row defines the template of one row with the “text” property of the widget into the first column and the “value” property of the widget into the second column. They will be dynamically filled using the data provided inside the model variable.

On the below example we define a **row of data**

```
var row_data = {
    _t : 1,
    _v : ["Temperature:", { _c : "dl" , s : "_TagMgr", a : "Tag1", i: 0, m : 2 }]
}
```

The first element is the row template to use while the second element is the array with the data to use. In our example “Temperature:” is the text to use inside the widget on the first column, while the other element is a datalink that will provide the value to fill the value property of the widget into the second column.

The JSON datalink element:

Parameter	Description
_c : "dl"	Identify the JSON element as a Datalink
s : "_TagMgr"	Specify the source of data is the Tag Manager
a : "Tag1", i: 0, m:2	Specify tag name and index (necessary when the tag is an array) and the read/write mode

Parameter	Description
	• m=0 is Read Only • m=1 is Write Only • m=2 is Read/Write

The below JavaScript code will generate the same table of the previous example using the Table Data Source Widget

```
var model = [];

var row_templates = {
    _h : [
            [ []       , []        ],   //rowType = 0
            [ ["text"] , ["value"] ]    //rowType = 1
        ]
}

var row_data1 = {
    _t : 0,
    _v : []
}

var row_data2 = {
    _t : 1,
    _v : ["Temperature:", { _c : "dl" , s : "_TagMgr", a : "Tag1", i: 0, m : 2 }]
}

var row_data3 = {
    _t : 1,
    _v : ["Humidity:",    { _c : "dl" , s : "_TagMgr", a : "Tag2", i: 0, m : 2 }]
}

var row_data4 = {
    _t : 1,
    _v : ["Noise:",       { _c : "dl" , s : "_TagMgr", a : "Tag3", i: 0, m : 2 }]
}

var row_data5 = {
    _t : 1,
    _v : ["Brightness:",  { _c : "dl" , s : "_TagMgr", a : "Tag4", i: 0, m : 2 }]
}

model[0] = row_templates;
model[1] = row_data1;
model[2] = row_data2;
model[3] = row_data3;
model[4] = row_data4;
model[5] = row_data5;
```

```
var myTable = page.getWidget("TableWgt1");
myTable.model = model;
```

Multilanguage

A multi languages text can be entered using the below JSON element:

```
{ _c : "ml" , mltext : {"en-US" : "Temperature:" , "it-IT" : "Temperatura:"} }
```

Parameter	Description
_c : "ml"	Identify the JSON element as a Multilanguage text
mltext : { ... }	List of couples: "ID Language":"Text" Example: • "en-US" : "Temperature:" • "it-IT" : "Temperatura:"

Example:

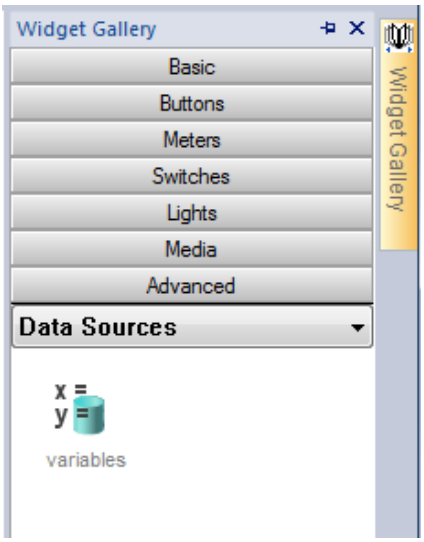
```
var row_data2 = {
    _t : 1,
    _v : [ { _c : "ml" , mltext : { "en-US" : "Temperature:",
                                    "it-IT" : "Temperatura:"}   },
           { _c : "dl" , s : "_TagMgr", a : "Tag1", i: 0, m : 2 }
         ]
}
```

Variables widget

Path: ***Widget Gallery> Advanced> Data Sources***

Use this widget to add internal variables for operations such as data transfer or to be used in JavaScript programs.

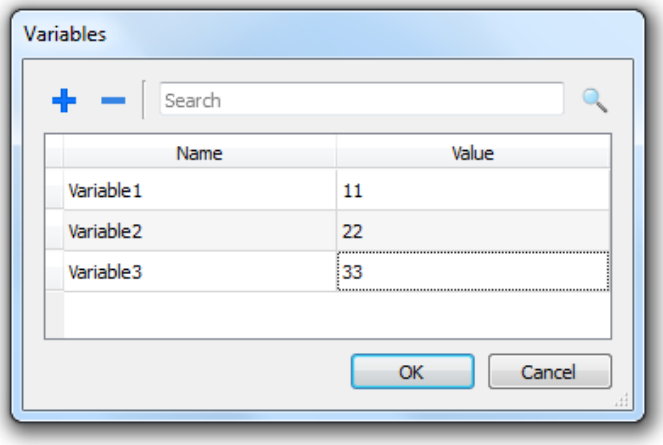
Note: The variables are local to the page where the widget has been inserted.



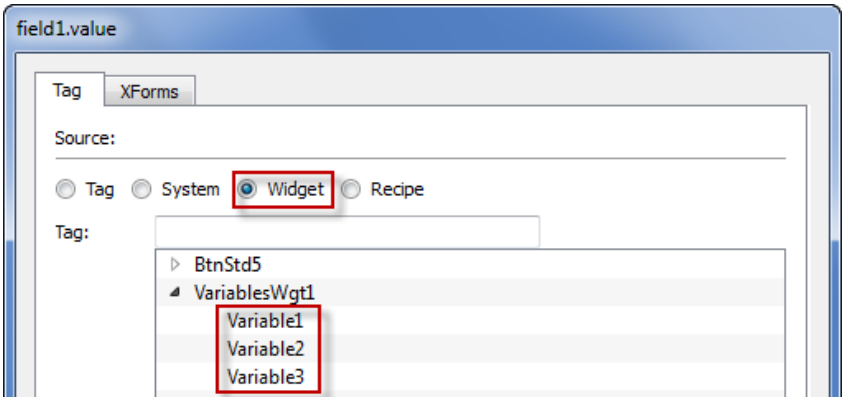
When you drag and drop this widget into you page, a place holder will be displayed to indicate the widget location, but it will not be visible at run time.

Setting the widget

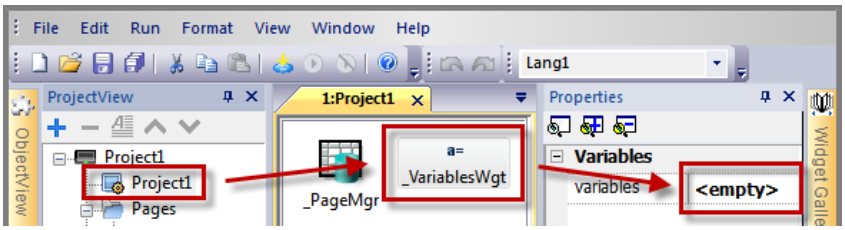
To create variables and assign values to them, open the **Variables** dialog from the **Variables** property in the **Properties** pane.



These variables can then be referenced from the **Attach tag** dialog, from the Page Editor.



If you need global variables, configure them at project level, adding the desired variables to the global variable widget.



Using variables in JavaScript

Variables can be also referenced in JavaScript programs with the following syntax:

For local variables:

```
var varWgt = page.getWidget("_VariablesWgt");
var compVar = varWgt.getProperty("VariableName");
```

For global variables:

```
var varWgt = project.getWidget("_VariablesWgt");
var compVar = varWgt.getProperty("VariableName");
```

34 Custom widgets

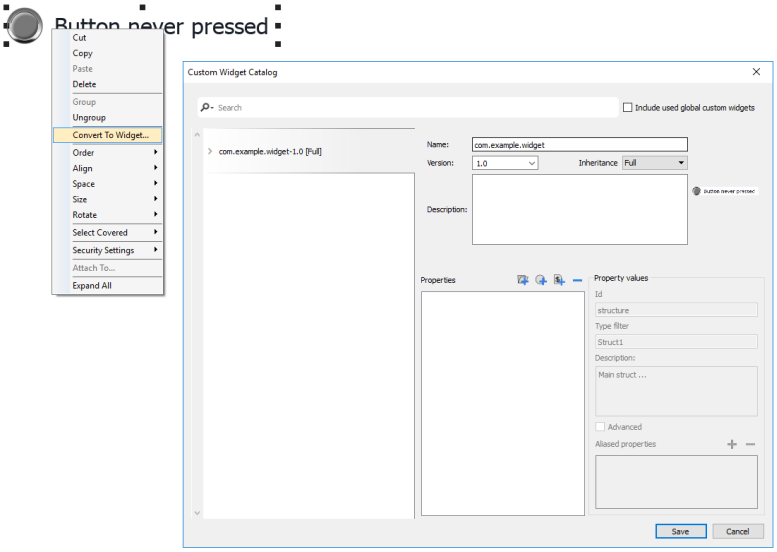
Wizard has a large widget library which includes predefined dynamic widgets (buttons, lights, gauges, switches, trends, recipes, and dialog items), as well as static images (shapes, pipes, tanks, motors).

You can drag and drop an object from the gallery to the page, and then size, move, rotate or transform it. All widgets in the gallery are vector based, so they do not loose definition when resized.

You can, however, modify any of the pre-defined widgets to create your own custom widget. Custom widgets can be made up of several elements only including the properties needed to their purpose.

Creating a custom widget

1. Drag and drop on a page all the widget you want to use to compose your custom widget.
2. Select and group them.
3. Right-click on the grouped object and select **Convert To Widget**: the **Custom Widgets Catalog** dialog is displayed.

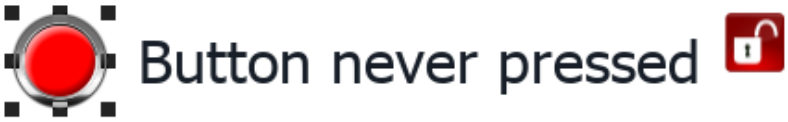


Parameter	Description
Include used custom widgets	When checked, list all the widgets used inside the project. Even system widgets.
Name	You can define everything you prefer, but is common keep a name structure. The folder com.hmi is reserved for the system widgets
Description	Widget description.
Version	Widget version. All widgets that share the same version share the properties defined from the Inheritance parameter.
Inheritance	Properties shared between widgets with the same version • Full (both Graphic and Logic) • Only Graphic • Only Logic • Disable **Inheritance is supported from version 2.7**

Modify a custom widget

Double click to select the custom widget in edit mode. The icon of the green padlock indicates that you are going to edit a custom widget, rather than just a group of widgets. The difference is that the modified will be propagate to all the other custom widgets with the same version that are configure to inherit the widget properties.

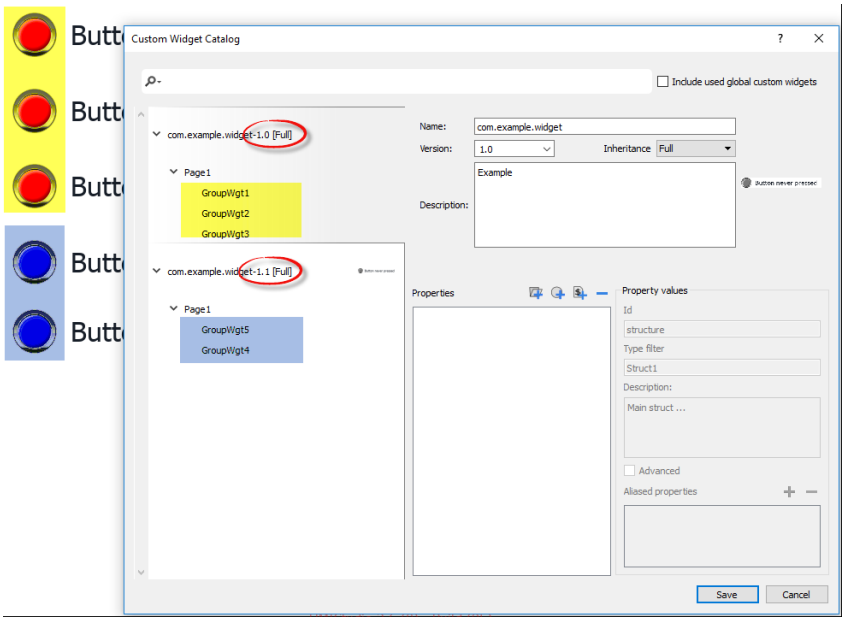
Click the padlock icon to enable the edit mode, padlock will be open. Click again when modifies are done.



Padlock is showed only when the Inheritance is enabled.

Share properties

When a custom widget is modified, all the modifies will be propagated to all the other custom widgets that share the same version and that are configured to inherit the widget properties.

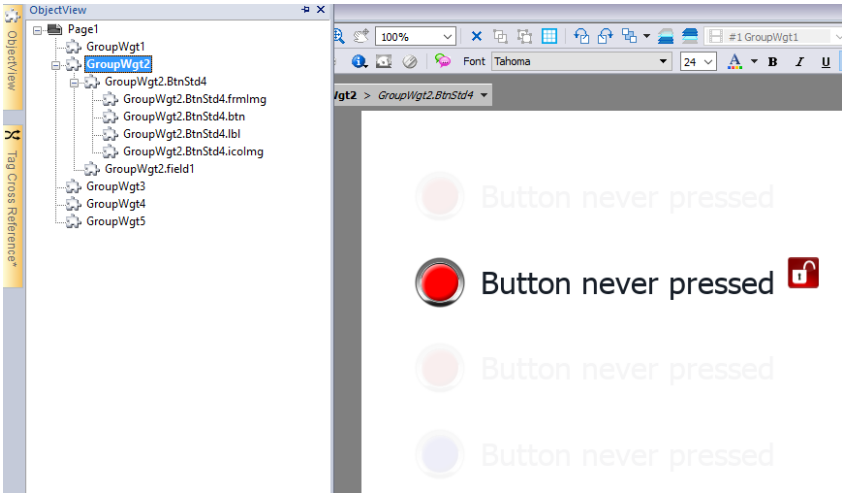


Using widgets components

Widgets are usually made up of many parts, for example a button is a complex widget including two image widgets, a button widget and label.

To display a list of all the elements that are part of a widget, select the widget, open the padlock and open the **ObjectView** pane: all the element making up a complex widget are listed in hierarchical order.

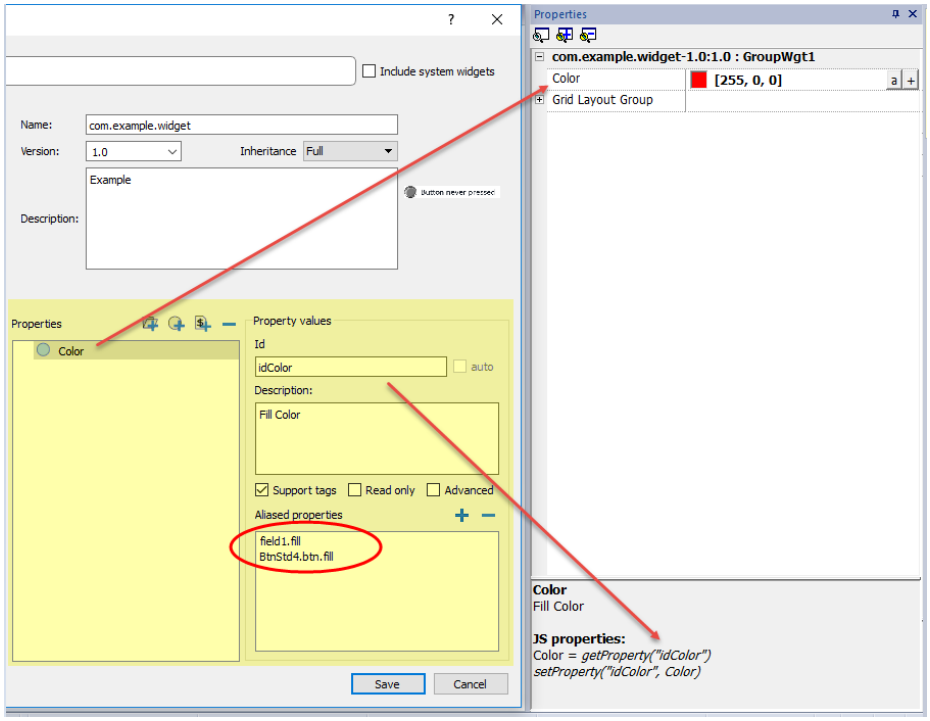
To select a single widget, select it directly form the **ObjectView** pane.



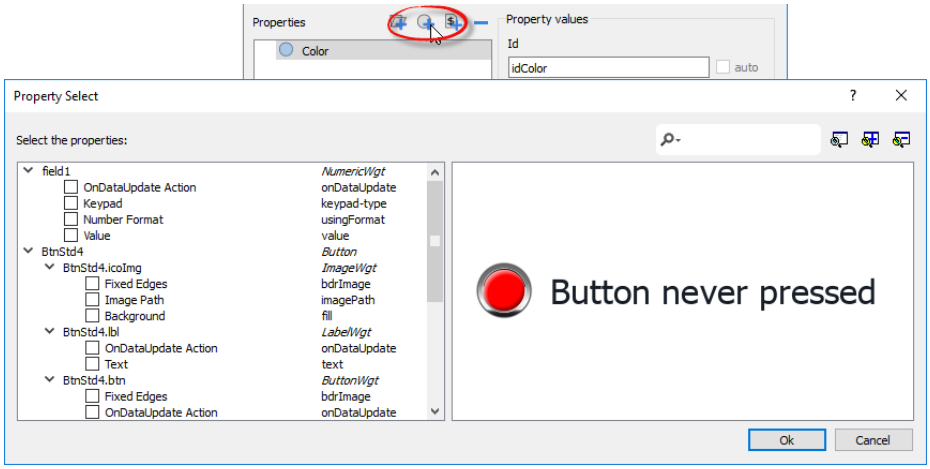
Adding properties to a custom widget

When you create a custom widget, you need to define the properties that will be displayed for it in the **Properties** pane.

1. Right-click on the grouped object and select **Widget catalog**: the properties dialog is displayed.



2. Click **+** to open the **Property Select** dialog: this lists all the properties of all the grouped widgets.



3. Select the properties you want to define for your custom widget.
4. Define each property's details.

Note that you can create folders and use drag & drop to move or reorganize the **Properties** list

Parameter	Description
Properties	Name shown in the **Properties** pane.
Description	Any comment on the property to be displayed in the **Properties** pane.
Id	The name exposed by Wizard, to JavaScript functions and Attach Tag dialog.
Support Tags	Specifies if the property supports the “Attach to" attribute.
Read only	Property exposed only in read mode
Advanced	Specifies whether each property should appear in the advanced, or in the simple view mode of the **Properties** pane.
Aliased properties	Internal properties linked with the exposed property

Combining properties

To combine two or more properties:

1. Select the primary property in the **Properties** list dialog.
2. Click **+** in the **Aliased properties** toolbar: the **Property Select** dialog is displayed.
3. Select the properties you want to combine.
4. Click **OK:** the combined attributes will be shown in the **Aliased properties** list box.

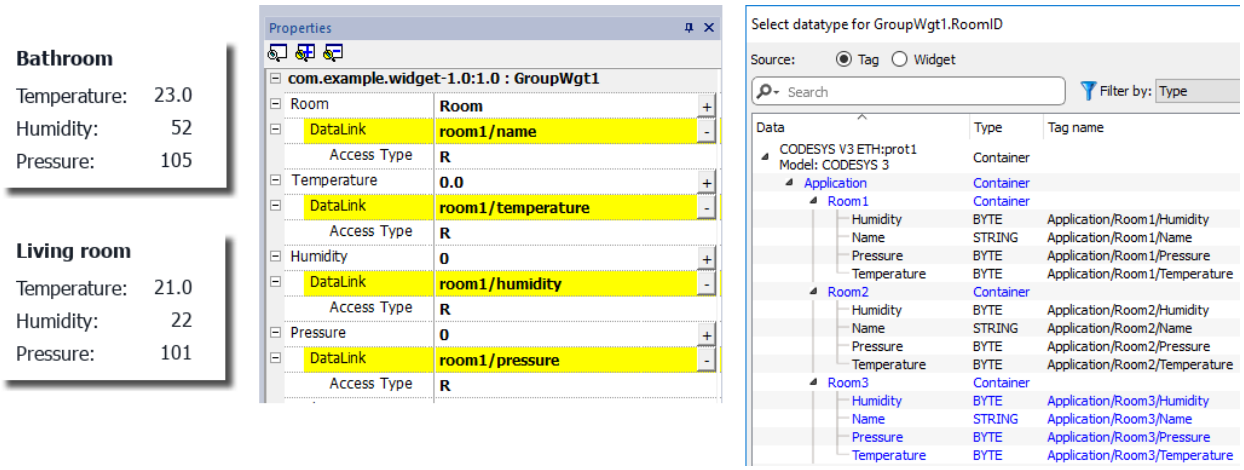
Example

If you insert into a “Color” property the fill color of all widgets (e.g. filed1.fill and BtnStd4.btn.fill) when you set the exposed Color property of the custom widget all colors of the included widgets will changes.

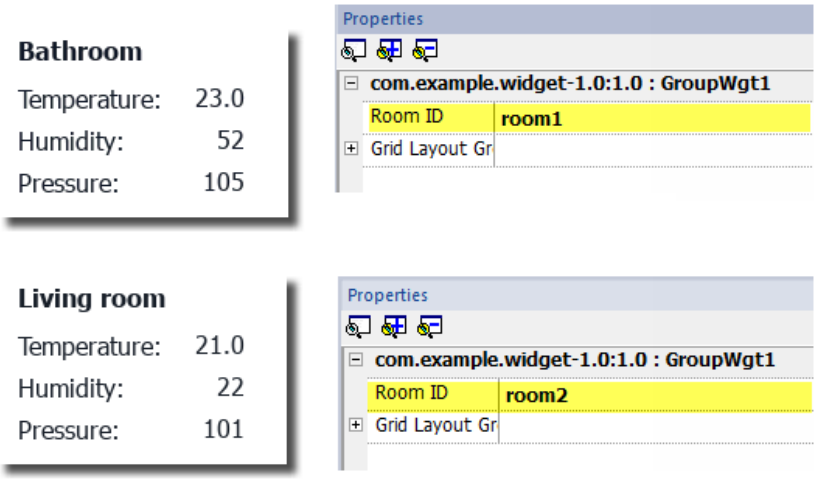
Using structured tags

A common problem using a widget that use many tags is the need to create instances of the widget by giving only the tag name of the structure that contains all the tags instead to configure each single tag.

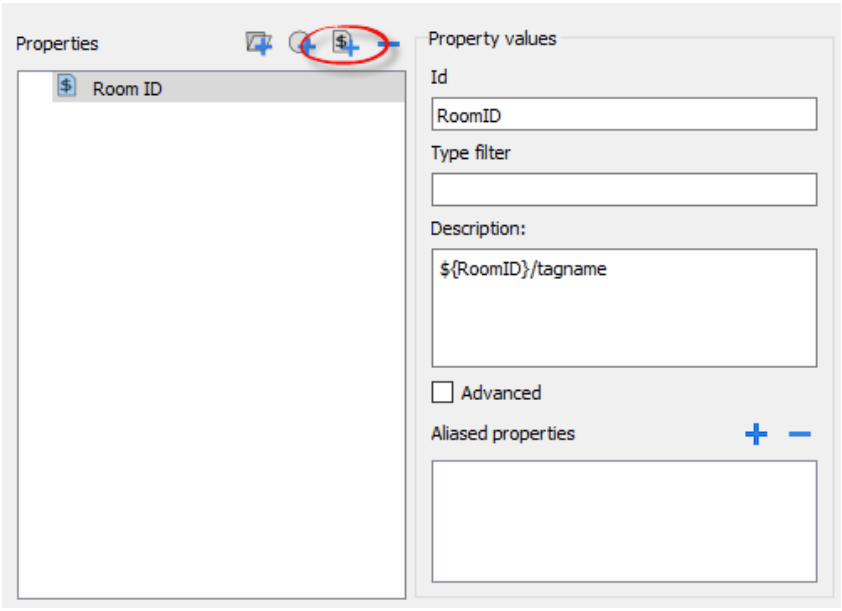
For example, think about the below widget. It use four tags, the room name, temperature, humidity and pressure. If we want use two instances of this widget for two different rooms we have to configure eight tags, four tags for each room.



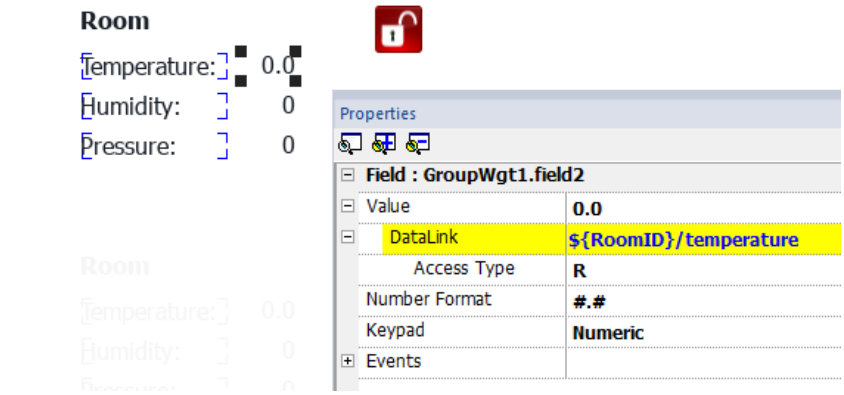
By using a **Parameter** property, is possible to set all the data links of the widget by giving only the name of the structure.



A “Parameter” field can be added inside the custom widget using the *"Add Parameter"* icon:

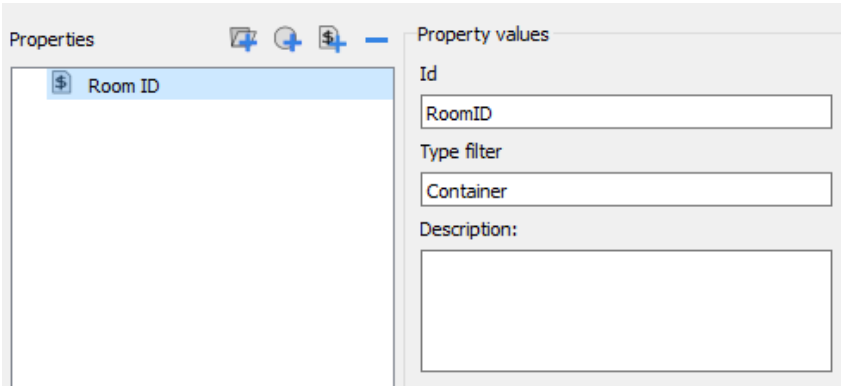


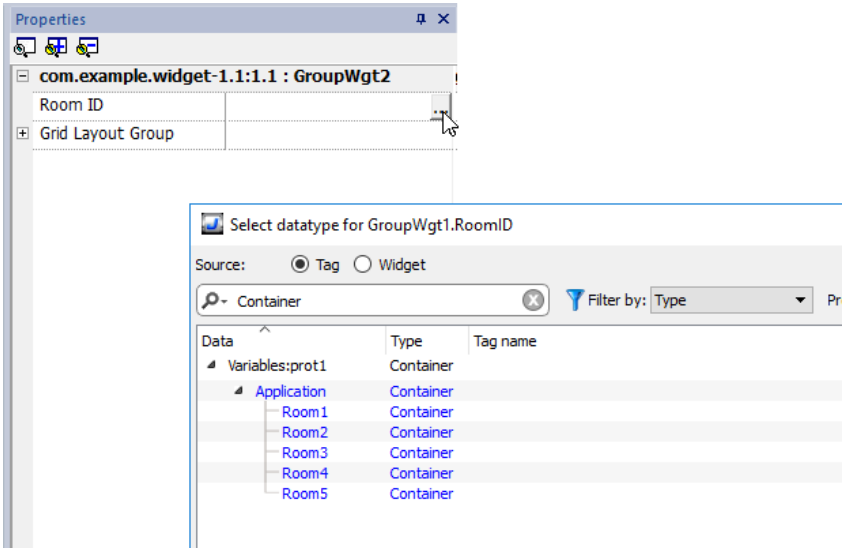
To configure the data links of the custom widget the keyword ${RoomID} can be used to reference at the structure instance



Type filter

Using *"Type filter"* parameter, when attach to tag is opened the listed tags will be filtered using the filter value.

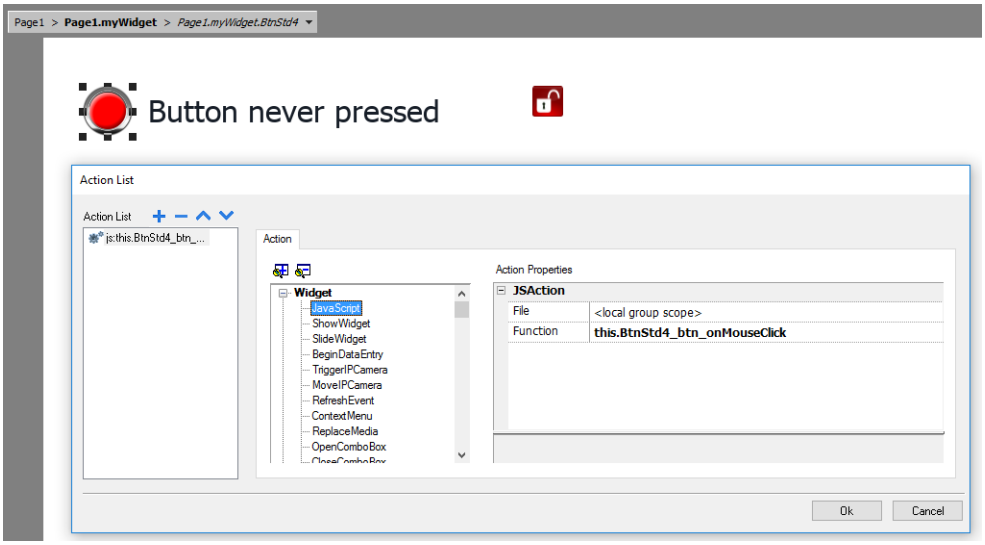




JavaScript in custom widgets

JavaScript functions can be embedded in custom widgets.

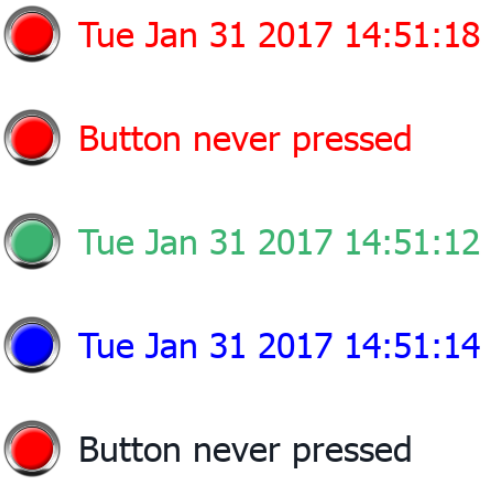
After doing a double click on the custom widget and clicked on the padlock, the edit mode is active and it is possible to associate the JavaScript code to the available events.



Note the usage of the operator **this.** that is necessary to allow the multiple instance of the custom widget.

If you need to reference to an element of the widget, you can use the keyword **wgt.it** to reference at the id of the active widget instance, as for the below example:

If you cut and paste some instances of the custom widget of the above example, you will obtain the below result.



Note that the custom widget can also past inside the User's Gallery for later reuse.

User's Gallery

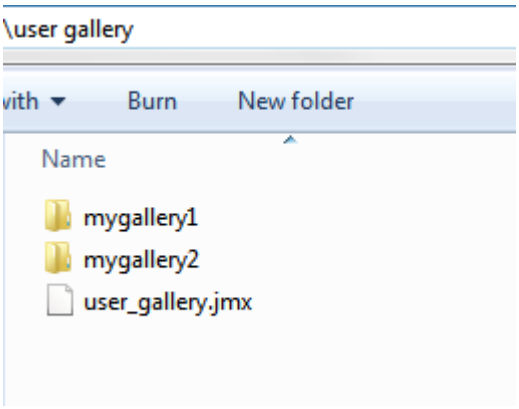
Widgets created from the developers can be saved inside the Widgets Gallery to be available during development of new projects.

User widgets toolbar

Command	Description
	Open the selected widgets folder into the Wizard editor
	Add a new widgets folder
	Delete current selected folder
	Select the user widgets folder

To add a new widget into the user gallery, open the widget folder and then edit the gallery page creating or adding the new widget.

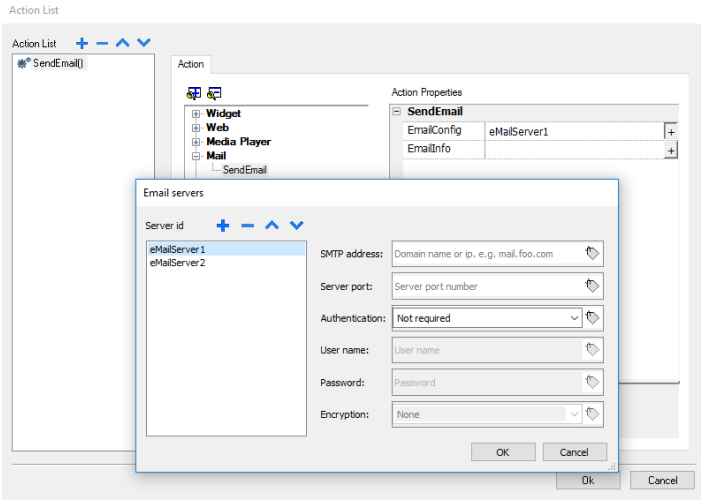
Tip: To import a user gallery sub folder, simply copy the folder to import inside the main user gallery folder.


\user gallery
vith
Burn
New folder
Name
mygallery1
mygallery2
user_gallery.jmx

35 Sending an email message

Send emails using the SendMail action, including tags in the email body and attachments.

The SendMail action has been created for working with alarms and schedulers but can be triggered and executed by many other events.



Configuring the email server

To configure the email server, enter the following information for the **EmailConfig** setting:

Parameter	Description
SMTP Address	SMTP server address.
Server Port	Port for SMTP server connection (default = 25).
Require Auth	Select if the SMTP server requires authentication.
User Name	Username for sending mail using SMTP server.
Password	Password for sending mails using SMTP server.
Encryption	Encryption type (none or SSL).

Click **+** to add more email servers.

Tip: Use tags if you want change the server parameters dynamically from the HMI Runtime.

Configure emails

Enter the following information for the **EmailInfo** setting:

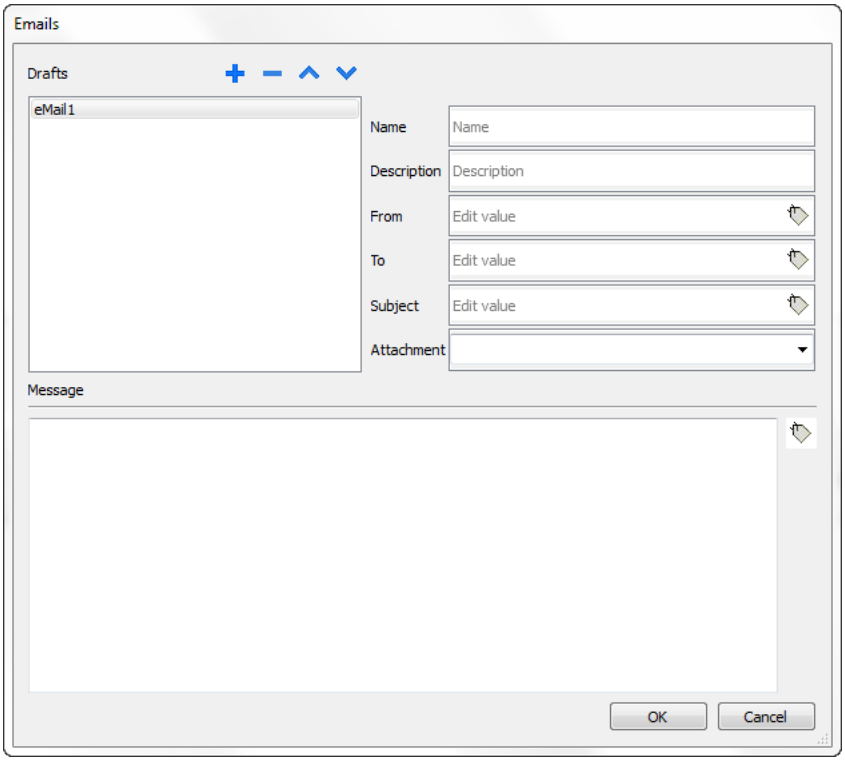
Parameter	Description
Name	Optional, this information is only for the log.
Description	Optional, this information is only for the log.
From	Optional, sender email address (for example, John@domain.com).
To	Recipient e-mail addresses. To enter multiple addresses, separate them with a semi-colon.
Subject	Subject of email.
Attachment	Path of the file to be sent as attachment. Only one attachment at a time can be sent. Note: The maximum size of the attachments is usually set by the SMTP server.
Body	Main content of the email. Here you can insert live tags if you include them in square brackets. For example, a message body as “Tag1 value is [Tag1]”, will be sent as “Tag1 value is 45”, if the current value of Tag1 is 45.

Tip: Attach a string tag to the **From**, **To** and **Subject** fields so that their value can be changed in the HMI Runtime.

WARNING: The maximum size for the message body is 4096 bytes, the exceeding text will be truncated.

Adding email templates

Click **+** to add more templates.



36 JavaScript

The purpose of this section is to describe how JavaScript is used in the Wizard applications, not to explain the JavaScript language.

Wizard JavaScript is based on the ECMAScript programming language http://www.ecmascript.org , as defined in standard ECMA-262.

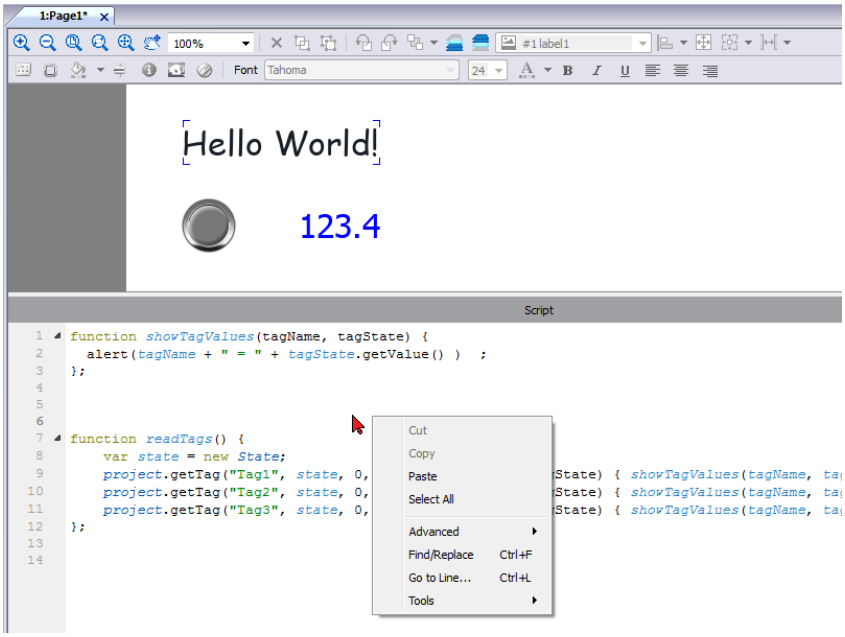
If you are familiar with JavaScript, you can use the same type of commands in Wizard as you do in a web browser. If you are not familiar with the ECMAScript language, refer to:

https://developer.mozilla.org/en/JavaScript

JavaScript editor

Wizard includes a powerful JavaScript editor.

Right-click in the editor to display available commands.



Execution of JavaScript functions

JavaScript functions are executed when events occur. For example, a user can define a script for the OnMouseClick event and the JavaScript script will be executed when the button is pressed on the HMI device.

JavaScript functions are executed only when the programmed event occurs and not cyclically. This approach minimizes the overhead required to execute logic in the HMI device.

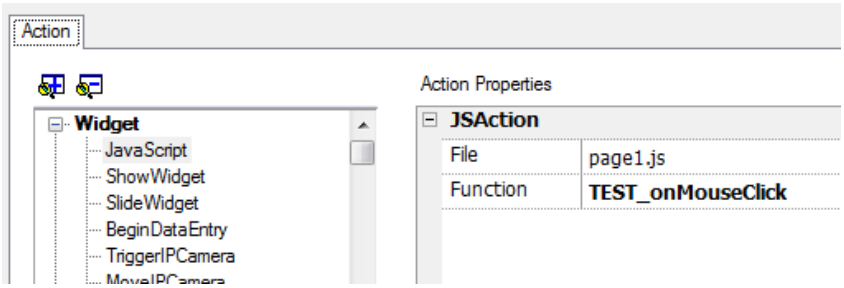
Wizard provides a JavaScript engine running on the client side. Each project page can contain scripts having a scope local to the page where they are added; global scripts can be created to be executed by scheduler events or alarm events.

In both cases scripts are executed on the client. This means that if more than one client is connected to the HMI device (for external computer running the HMI Client), each client will run the same script, providing different output results depending on the input, since inputs provided to different clients may be different.

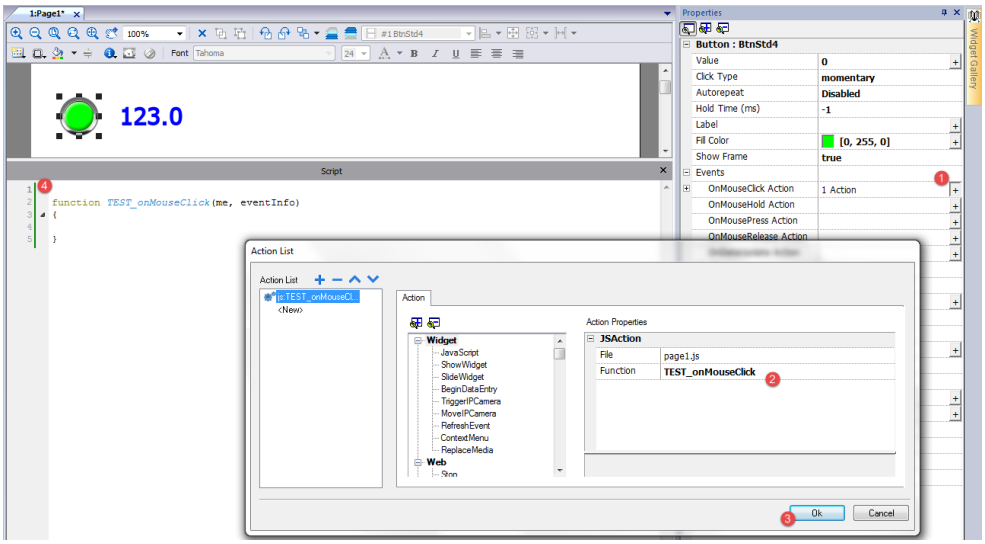
For example, if a script acts according to the position of a slider and this position is different on the different clients, the result of the script will be different on each client.

JavaScript functions for page events

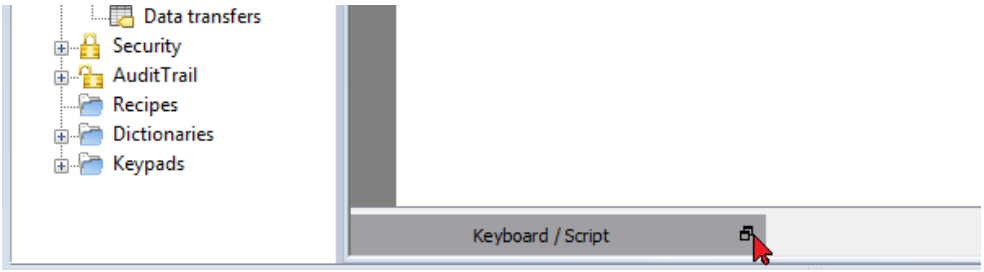
JavaScript editor will open when you add a JavaScript action inside an action list.



1. Select the even that will execute the action.
2. Add a **JavaScript** action from the **Widget** category.
3. Either leave the default function name, or type a new one.
4. Click **OK** to confirm: the JavaScript editor displays your function structure.



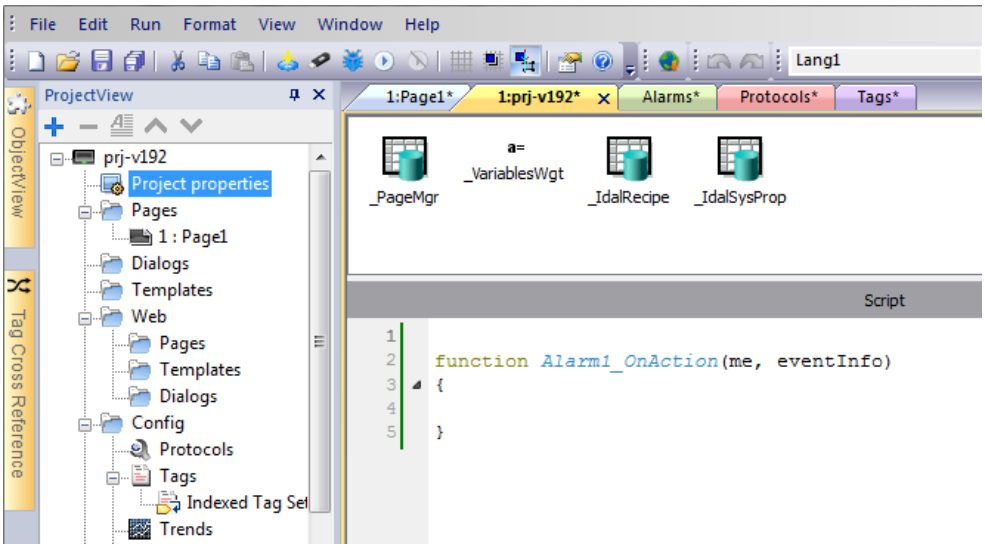
You can also open the JavaScript editor from the **Script** tab at the bottom of the workspace.



JavaScript functions for alarms and scheduled events

JavaScript code associated with alarms and scheduled events and not associated with a specific page, can be edited from the main **Project properties** page.

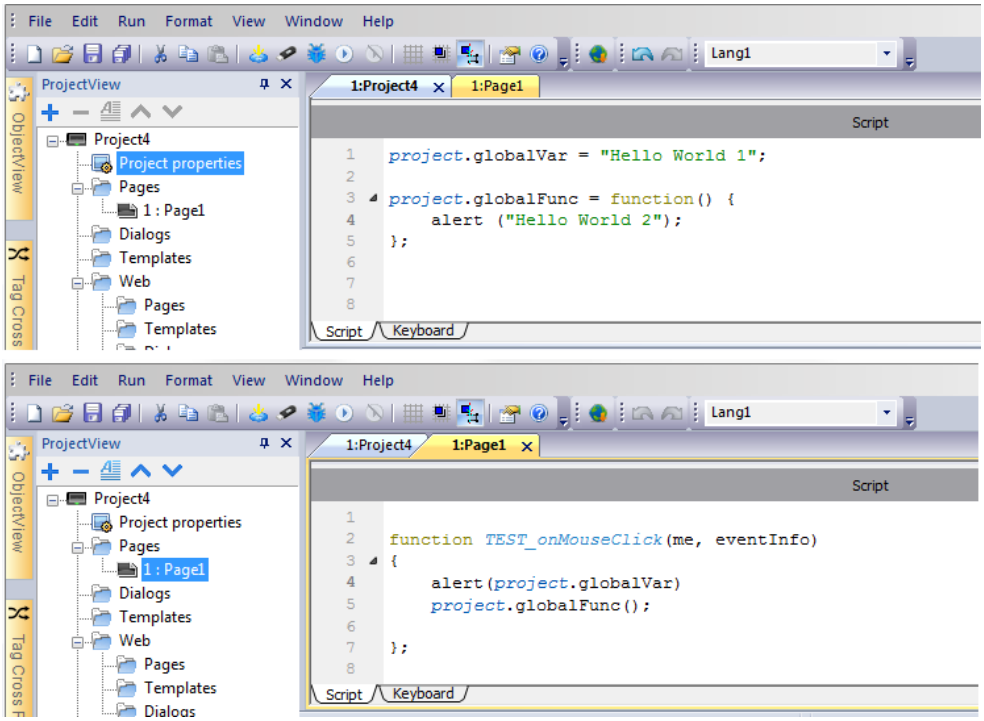
*Path: **ProjectView**> double-click **Project properties***



Note: JavaScript actions are client actions so they are executed only when a client is logged in.

Shared JavaScript code

The **project** global variable can be used to share JavaScript code between the pages. Variables are created/initialized from the main JavaScript code from the main **Project properties** page and can then be used from the project pages.



Events

You can add JavaScript to the following categories of events:

- Widget events
- Page events
- System events

For events of type:

- OnMousePress
- OnMouseRelease
- OnMouseClick
- OnWheel

JavaScript **eventinfo** parameter contains the following additional properties:

Parameter	Description
eventInfo.posX	Local mouse/touch X coordinate with respect to widget coordinates
eventInfo.posY	Local mouse/touch Y coordinate with respect to widget coordinates
eventInfo.pagePosX	Page X mouse/touch coordinate
eventInfo.pagePosY	Page Y mouse/touch coordinate
eventInfo.wheelDelta	Mouse wheel delta. Integer value with sign representing the rotation direction. The actual value is the rotation amount in eighths of a degree. The smallest value depends on the mouse resolution. Typically this is 120, corresponding to 15 degrees.

Widget events

onMouseClick

```
void onMouseClick (me, eventInfo)
```

This event is available only for buttons and it occurs when the button is pressed and released quickly.

Parameter	Description
me	Object triggering the event
eventInfo	Details of triggered event

```
function buttonStd1_onMouseClick(me, eventInfo) {
     //do something…
}
```

onMouseHold

```
void onMouseHold (me, eventInfo)
```

This event is available only for buttons and it occurs when the button is pressed and released after the number of seconds set as **Hold Time** in the widget properties.

Parameter	Description
me	Object triggering the event
eventInfo	Details of triggered event

```
function buttonStd1_onMouseHold(me, eventInfo) {
     //do something…
}
```

onMousePress

```
void onMousePress(me, eventInfo)
```

This event is available only for buttons and it occurs when the button is pressed.

Parameter	Description
me	Object triggering the event
eventInfo	Details of triggered event

```
function buttonStd1_onMousePress(me, eventInfo) {
     //do something…
}
```

onMouseRelease

```
void onMouseRelease (me, eventInfo)
```

This event is available only for buttons and it occurs when the button is released.

Parameter	Description
me	Object triggering the event
eventInfo	Details of triggered event

```
function buttonStd1_onMouseRelease(me, eventInfo) {
     //do something…
}
```

onDataUpdate

```
boolean onDataUpdate (me, eventInfo)
```

This event occurs when data attached to the widget changes.

Parameter	Description
me	Object triggering the event
eventInfo	An object with the fields listed below (you can refer fields using "." - dot notation) **oldValue** = Widget value before the change **newValue** = Value which will be updated to the widget **attrName** = Attribute on which the event is generated **index** = Integer attribute index if any, default = 0 **mode =** W when the user is writing to the widget. R in all others status.

The event is triggered before the value is passed to the widget, this means the JavaScript code can modify the value before it is actually passed to the widget.

The code can terminate with a return true or return false. After terminating the code with return false, control is returned to the calling widget that may launch other actions.

After terminating the code with true, the control is not returned to the widget and this makes sure that no additional actions are executed following the calling event.

```
function buttonStd1_onDataUpdate(me, eventInfo) {
if ( eventInfo.oldValue < 0) {
     //do something…
}
     return false;
}
```

OnPan

```
boolean onGesturePan(me, eventInfo)
```

This event is only available for gesture area buttons; it occurs when one point inside the area has pressed and a movement has been detected.

Parameter	Description
me	Object triggering the event.
eventInfo	**id** = Gesture id; it is used to identify different gestures. **running** = True except for last event delivered to notify gesture completion. **dx** = Total X axis movement in screen pixel units from initial touch position . **dy** = Total Y axis movement in screen pixel units from initial touch position.

```
function gstArea_onGesturePan(me, eventInfo)
{
     wTYPE.setProperty("value","PAN");
```

```
    wID.setProperty("value",eventInfo.id);
    wDX.setProperty("value",eventInfo.dx);
    wDY.setProperty("value",eventInfo.dy);
    wRUN.setProperty("value",eventInfo.running);
}
```

OnPinch

```
boolean onGesturePinch(me, eventInfo)
```

This event is only available for gesture area buttons; it occurs when two points inside the area have been pressed and a movement has been detected.

WARNING: Only multi touch HMI devices can generate pinch events

Parameter	Description
me	Object triggering the event
eventInfo	**id** = Gesture id; it is used to identify different gestures. **running** = True except for last event delivered to notify gesture completion. **dx** = Total X axis movement in screen pixel units from initial touch position. It represents the distance change between fingers. Positive value means that the distance is increasing; negative value means that the distance is decreasing. This amount may be used to control a zoom value. **dy** = Total Y axis movement in screen pixel units (see dx).

```
function gstArea_onGesturePinch(me, eventInfo)
{
    wTYPE.setProperty("value","PINCH");
    wID.setProperty("value",eventInfo.id);
    wDX.setProperty("value",eventInfo.dx);
    wDY.setProperty("value",eventInfo.dy);
    wRUN.setProperty("value",eventInfo.running);
}
```

Page events

onActivate

```
void onActivate( me, eventInfo )
```

This event occurs each time the page is displayed.

Parameter	Description
me	Object triggering the event
eventInfo	Reserved for future use

JavaScript will be executed when the page is active, that is when the page is loaded.

```
function Page1_onActivate(me, eventInfo) {
     //do something…
}
```

onDeactivate

`void onDeactivate( me, eventInfo )`

This event occurs when leaving the page.

Parameter	Description
me	Object triggering the event
eventInfo	Reserved for future use

```
function Page1_onDeactivate(me, eventInfo) {
     //do something…
}
```

onWheel

`void onMouseWheelClock( me, eventInfo )`

This event occurs when a wheel device is moving (for example, a mouse wheel).

Parameter	Description
me	Object triggering the event
eventInfo	Details of triggered event

```
function Page1_onMouseWheelClock(me, eventInfo) {
     //do something…
}
```

System events

System events can be related to:

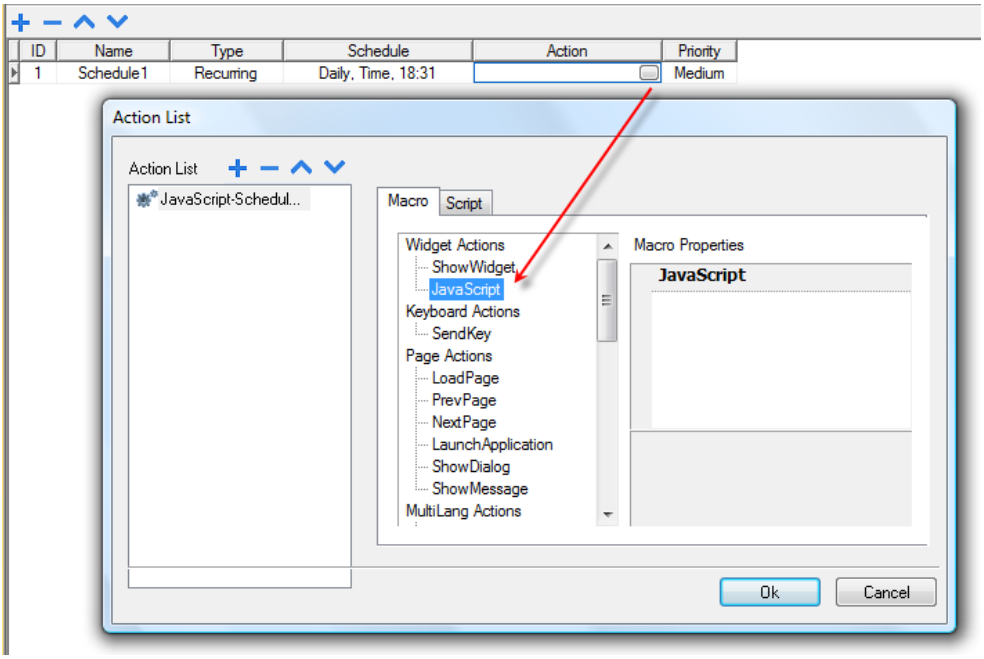
- scheduler
- alarms
- a wheel device

Important: Make sure you do not duplicate JavaScript function names at page and project level. When a conflict happens, that is two functions with the same name in current page and at project level, the system execute the JavaScript callback at page level.

When a JavaScript callback is not found in the current page, the system automatically searches for it at project level.

Scheduler events

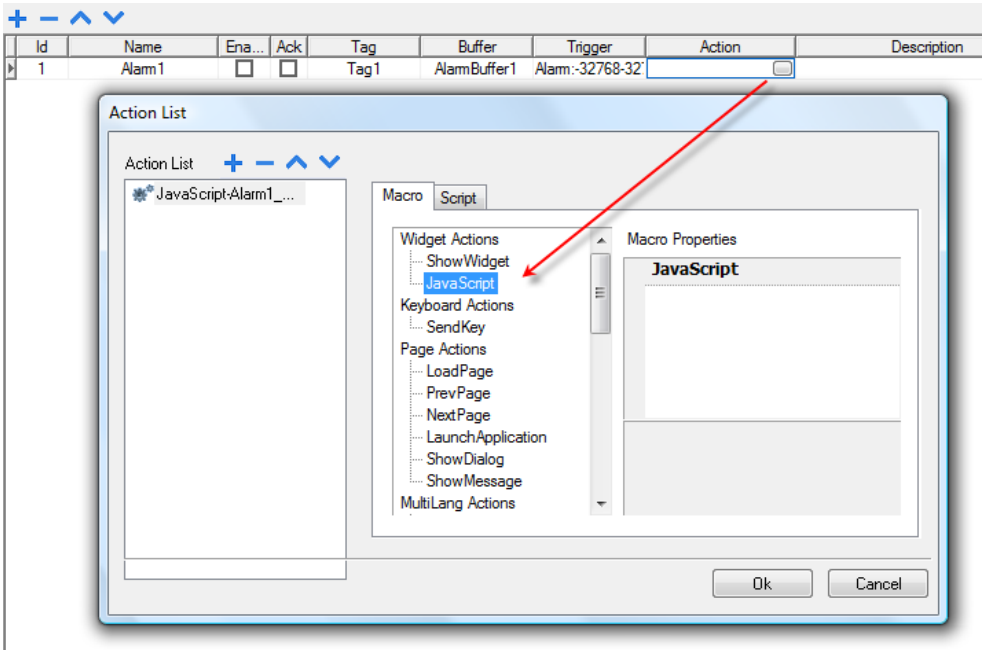
These events occur when triggered by the associated action in the scheduler.



You can edit the JavaScript from the **Project Properties** tab.

Alarm events

These events occur when triggered by the associated alarm condition.



You can edit the JavaScript from the **Project Properties** tab.

onWheel

```
void onMouseWheelClock( me, eventInfo )
```

This event occurs when a wheel device is moving (for example, a mouse wheel).

Parameter	Description
me	Object triggering the event
eventInfo	Details of triggered event

```
function Project1_onMouseWheelClock(me, eventInfo) {
     //do something…
}
```

Objects

Wizard uses JavaScript objects to access the elements of the page. Each object is composed of properties and methods that are used to define the operation and appearance of the page element. The following objects are used to interact with elements of the HMI device page:

Object	Description
Widget	This is the base class for all elements on the page including the page element
Page	This object references the current HMI device page.

Object	Description
	The page is the top-level object of the screen.
Group	This object associates a set of tags to allow uniform operation on a set of logically connected tags
Project	This object defines the project widget. The project widget is used to retrieve data about the project such as tags, alarms, recipes, schedules, tags and so on. There is only one widget for the project and it can be referenced through the project variable.
State	This object is the class holding the state of a variable acquired from the controlled environment. Beside the value itself, it contains the timestamp indicating when the value was collected and flags marking the quality of the value.

Widget class objects

The Widget class is the base class for all the elements on a page including the page element.

Widget, in this case, is not used to indicate a specific screen object but a JavaScript class.

Changing widget properties with JavaScript

If you want to change the properties of widgets with JavaScript set the widget property **Static Optimization** to **Dynamic**.

Important: If the widget property Static Optimization is not set to Dynamic, changes to properties will be ignored.

Whenever a call to `getWidget` fails, the remote debugger reports the following error:

“Trying to access static optimized widget "label1". Disable widget static optimization to access widget from script.”.

This error is visible also using following code fragment:

```
var wgt;
try {
wgt = page.getWidget('label1');
} catch(err) {
alert("" + err);
}
```

Widget properties

Some properties are common to all widgets.

objectName

```
string objectName
```

Gets the name of the widget, a unique id.

```
function btnStd04_onMouseRelease(me) {
     var wgt = page.getWidget("rect1");
     var name = wgt.objectName;
}
```

(Available on web pages)

x

```
number x
```

Gets or sets the widget 'x' position in pixels.

```
function btnStd1_onMouseRelease(me) {
     var wgt = page.getWidget("rect1");
     wgt.x = 10;
}
```

(Available on web pages)

y

```
number y
```

Gets or sets the widget 'y' position in pixels.

```
function btnStd1_onMouseRelease(me) {
     var wgt = page.getWidget("rect1");
     wgt.y = 10;
}
```

(Available on web pages)

width

```
number width
```

Gets or sets the widget width in pixels.

```
function btnStd1_onMouseRelease(me) {
     var wgt = page.getWidget("rect1");
     wgt.width = 10;
}
```

(Available on web pages)

height

`number height`

Gets or sets the widget height in pixels.

```
function btnStd1_onMouseRelease(me) {
     var wgt = page.getWidget("rect1");
     wgt.height = 10;
}
```

(Available on web pages)

visible

`boolean visible`

Gets or sets the widget visible state.

```
function btnStd4_onMouseRelease(me) {
     var wgt = page.getWidget("rect1");
     wgt.visible = false;
}

function btnStd5_onMouseRelease(me) {
     var wgt = page.getWidget("rect1");
     wgt.visible = true;
}
```

value

`number value`

Gets or sets the widget value.

```
function btnStd6_onMouseRelease(me) {
     var wgt = page.getWidget("field1");
     wgt.value = 100;
}
```

opacity

`number opacity (range from 0 to 1)`

Gets or sets the widget opacity. Values are decimals from 0 to 1, where 1 is 100% opaque.

```
function btnStd8_onMouseRelease(me) {
     var wgt = page.getWidget("rect1");
```

```
    wgt.opacity = 0.5;
}
```

(Available on web pages)

rotation

```
number rotation (in degrees)
```

Gets or sets the rotation angle for the widget. The rotation is done clockwise and by degrees, starting at the East position.

```
function btnStd9_onMouseRelease(me) {
    var wgt = page.getWidget("rect1");
    wgt.rotation = 45;
}
```

(Available on web pages)

userValue

```
string userValue
```

Gets or sets a user-defined value for the widget. This field can be used by JavaScript functions to store additional data with the widget.

```
function btnStd9_onMouseRelease(me) {
    var wgt = page.getWidget("rect1");
    wgt.userValue = "Here I can store custom data";
}
```

Every widget has some specific properties that you can access using dot notation. For an up-to-date and detailed list of properties you can use the JavaScript Debugger inspecting the widget methods and properties.

Widget methods

Some methods are common to all widgets.

getProperty

```
object getProperty( propertyName, [index] )
```

Returns a property.

Parameter	Description
propertyName	String containing the name of property to get
index	Index of the element to get from the array (default = 0)

Almost all properties that are shown in the Wizard **Properties** pane can be retrieved using the `getProperty` method. The index value is optional and only used for widgets that support arrays.

```
function buttonStd1_onMouseRelease(me, eventInfo) {
     var shape = page.getWidget("rect2");
     var y_position = shape.getProperty("y");
}
```

```
function buttonStd2_onMouseRelease(me, eventInfo) {
     var image = page.getWidget("multistate1");
     var image3 = image.getProperty("imageList", 2);
     //…
}
```

(Available on web pages)

setProperty

```
boolean setProperty( propertyName, value, [index] )
```

Sets a property for the widget.

Parameters

Parameter	Description
propertyName	String containing the name of property to set
value	String containing the value to set the property.
index	Index of the element to set in the array (default = 0)

Almost all properties that are shown in the Wizard **Properties** pane can be set by this method. The index value is optional and only used for Widgets that support arrays (for example, a MultiState Image widget). The `setProperty` method returns a boolean value (true or false) to indicate if the property was set or not.

```
function buttonStd1_onMouseRelease(me, eventInfo) {
     var setting_result = shape.setProperty("y", 128);
     if (setting_result)
     alert("Shape returned to start position");
}
```

```
function buttonStd2_onMouseRelease(me, eventInfo) {
     var image = page.getWidget("multistate1");
     var result = image.setProperty("imageList", "Fract004.png", 2);
     //…
}
```

(Available on web pages)

Page object

This object references the current HMI device page. The page is the top-level object of the screen.

Page object properties

Properties available at page level.

backgroundColor

```
string backgroundColor (in format rgb(xxx, xxx, xxx) where xxx range from 0 to 255)
```

Page background color.

```
function btnStd11_onMouseRelease(me) {
     page.backgroundColor = "rgb(128,0,0)";
}
```

(Available on web pages)

width

```
number width
```

Page width in pixels.

```
function btnStd05_onMouseRelease(me) {
     var middle_x = page.width / 2;
}
```

(Available on web pages, get only)

height

```
number height
```

Page height in pixels.

```
function btnStd05_onMouseRelease(me) {
     var middle_y = page.height / 2;
}
```

(Available on web pages, get only)

userValue

```
string userValue
```

Gets or sets a user-defined value for the widget. This field can be used by JavaScript functions to store additional data with the page.

```
function btnStd9_onMouseRelease(me) {
     page.userValue = "Here I can store custom data";
}
```

(Available on web pages)

Page object methods

Methods that can be used at page level.

getWidget

```
object getWidget( wgtName )
```

Returns the widget with the given name.

Parameter	Description
wgtName	String containing the widget name

Return value

An object representing the widget. If the widget does not exist, null is returned.

```
function btnStd1_onMouseRelease(me) {
     var my_button = page.getWidget("btnStd1");
}
```

(Available on web pages)

setTimeout

```
number setTimeout( functionName, delay )
```

Starts a timer to call a given function after a given delay.

Parameter	Description
functionName	String containing the name of function to call
delay	Delay in milliseconds

Return value

A number corresponding to the timerID.

```
var duration = 3000;
var myTimer = page.setTimeout("innerChangeWidth()", duration);
```

(Available on web pages)

clearTimeout

```
void clearTimeout( timerID )
```

Stops and clears the timeout timer with the given timer.

Parameter	Description
timerID	Timer to be cleared and stopped

```
var duration = 3000;
var myTimer = page.setTimeout("innerChangeWidth()", duration);
// do something
page.clearTimeout(myTimer);
```

(Available on web pages)

setInterval

```
number setInterval( functionName, interval )
```

Starts a timer that executes the given function with the given interval.

Parameter	Description
functionName	String containing the name of function to call
interval	Interval in milliseconds

Return value

A number corresponding to the timerID.

```
var interval = 3000;
var myTimer = page.setInterval("innerChangeWidth()", interval);
```

(Available on web pages)

clearInterval

```
void clearInterval( timerID )
```

Stops and clears the interval timer with the given timer.

Parameter	Description
timerID	Timer to be cleared and stopped

```
var interval = 3000;
var myTimer = page.setInterval("innerChangeWidth()", interval);
// do something
```

```
page.clearInterval(myTimer);
```

(Available on web pages)

clearAllTimeouts

```
void clearAllTimeouts()
```

Clears all the timers started.

```
page.clearAllTimeouts();
```

(Available on web pages)

Group object

A group is a basic logical element that associates a set of logical tags.

Group object methods

Methods that can be used with group objects.

getTag

```
object getTag( TagName )
```

Gets the tag specified by TagName from the group object.

Parameter	Description
TagName	String representing the tag name

Return value

An object that is the value of the tag or, if tag value is an array, the complete array. If you need to retrieve an element of the array, check the method `getTag` available in the project object. Undefined is returned if tag is invalid.

```
var group = new Group();
project.getGroup("GroupName", group);
var value = group.getTag("Tag1");
```

(Available on web pages)

getCount

```
number getCount()
```

Returns total number of tags in this group.

```
var group = new Group();
```

```
project.getGroup("GroupName", group);
var value = group.getCount();
```

(Available on web pages)

getTags

```
object getTags()
```

Returns the list of all tags in group.

```
function {
var group = new Group();
project.getGroup("enginesettings", group);
var tagList = group.getTags();
for(var i = 0; i < tagList.length; i++){
     var tagName = tagList[i];
     //do something…
};
```

(Available on web pages)

Project object

This object defines the project widget. The project widget is used to retrieve data about the project such as tags, alarms, recipes, schedules, tags and so on. There is only one widget for the project and it can be referenced through the project variable.

Project object properties

Properties to be set at project level.

startPage

```
string startPage
```

Page shown when the project is started.

```
var startPage = project.startPage;
project.startPage = "Page2.jmx";
```

Project object methods

Methods to be used at project level.

nextPage

```
void nextPage()
```

The script executes the Next page action.

```
project.nextPage();
```

(Available on web pages)

prevPage

```
void prevPage()
```

The script executes the previous page action.

```
project.prevPage();
```

(Available on web pages)

lastVisitedPage

```
void lastVisitedPage()
```

The script executes the last visited page action.

```
project.lastVisitedPage();
```

(Available on web pages)

homepage

```
void homePage()
```

The script executes the Home page action.

```
project.homePage();
```

(Available on web pages)

loadPage

```
void loadPage(pageName)
```

The script executes to load the set page defined in the script.

```
project.loadPage("Page5.jmx");
```

(Available on web pages)

WARNING: When page change, all active time events are forced to removed and the JavaScript procedure will run until the end before switch to the new page.

showDialog

```
void showDialog(pageName)
```

The script executes to show the dialog page.

```
project.showDialog("Dialog.jmx");
```

(Available on web pages)

closeDialog

```
void closeDialog()
```

The script executes to close the currently-opened dialog page.

```
project.closeDialog();
```

(Available on web pages)

showMessage

```
void showMessage( message )
```

The script executes to display the message popup.

```
project.showMessage("Hi This is test message");
```

(Available on web pages)

getGroup

```
number getGroup( groupName, groupInstance, [callback] )
```

Fast read method; this gets the values of all tags in a group.

Parameter	Description
groupName	String containing the name of the group
groupInstance	Group element to be filled
callback	String containing the name of the function to be called when the group is ready

Return value

A number value that is the status: 1 for success, 0 for fail.

```
var group = new Group();
var status = project.getGroup ("enginesettings", group);
if (status == 1) {
     var value = group.getTag("Tag1");
     if (value!=undefined) {
```

```
      // do something with the value
      }
}
```

```
var g = new Group();
var status = project.getGroup ("enginesettings", g,
      function (groupName, group) { fnGroupReady(groupName, group);} );

function fnGroupReady(groupName, group) {
      var val = group.getTag("Tag1");
      if (val!=undefined) {
      // do something with the value
      }
}
```

(Available on web pages)

getTag

```
object getTag( tagName, state, index, forceRefresh)
```

```
void getTag( tagName, state, index, callback, forceRefresh)
```

It returns the tag value or the complete array if index value is -1 of the given tagName.

Parameter	Description
tagName	String of tag name
state	State element to be filled
index	Index if the tag is of array type. -1 returns the complete array. Default = 0.
callback	Function name if an asynchronous read is required. Default = “”.
forceRefresh	(Optional parameter) True = the Runtime will read an updated value of the tag directly from the device. Default is false.

Return value

Tags value is returned. If tag is array type and index = -1 then the complete array is returned. For non-array tags provide index as 0.

```
var state = new State();
var value = project.getTag("Tag1", state, 0);
//
//for non array type
//tags index is not considered, so can be left as 0
```

```
//
if (value!=undefined) {
//...do something with s
}
```

```
var state = new State();
project.getTag("Tag1", state, -1,
     function(tagName, tagState) { fnTagReady(tagName, tagState); });
function fnTagReady(tagName, tagState) {
     if (tagName=="Tag1") {
     var myValue = tagState.getValue();
     }
}
```

(Available on web pages)

setTag

```
number setTag( tagName, tagValue, [index], [forceWrite] )
```

Sets the given tag in the project. Name and value are in strings.

Parameter	Description
tagName	String of tag name
tagValue	Object containing the value to write
index	Index if the tag is of array type. -1 pass the complete array. Default = 0.
forceWrite	Boolean value for enabling force write of tags, the function will wait for the value to be written before it returns back. Default = false.

Return value

Interger value for denoting success and failure of action when forceWrite is true. 0 means success and -1 means failure. If forceWrite is false, returned value will be undefined.

```
var val = [1,2,3,4,5];
var status = project.setTag("Tag1", val, -1, true);
if (status == 0) {
     // Success
} else {
     // Failure
}
```

```
var val = "value";
project.setTag("Tag1", val);
```

(Available on web pages)

updateSystemVariables

```
void project.updateSystemVariables()
```

Force system variables to refresh.

```
project.updateSystemVariables()
```

selectAllAlarms

void project.selectAllAlarms(bool selected)

Select/unselect all alarms

```
project.selectAllAlarms(true)
```

(Available on web pages)

ackAlarms

void project.ackAlarms()

Acknowledge all selected alarms

```
project.selectAllAlarms(true);
project.ackAlarms();
project.selectAllAlarms(true);
```

(Available on web pages)

resetAlarms

void project.resetAlarms()

Reset all selected alarms

```
project.selectAllAlarms(true);
project.resetAlarms();
project.selectAllAlarms(true);
```

(Available on web pages)

enableAlarms

void project.enableAlarms()

Enable all selected alarms

```
project.selectAllAlarms(true);
project.enableAlarms();
project.selectAllAlarms(true);
```

(Available on web pages)

getRecipeItem

```
object getRecipeItem (recipeName, recipeSet, recipeElement)
```

Gets the value of the given recipe set element.

Parameter	Description
recipeName	String representing the recipe name
recipeSet	String representing the recipe set, can be either the recipe set name or 0 based set index.
recipeElement	String representing the recipe Element, can be either the element name or 0 based element index.

Return value

An object with the value of the recipe. undefined is returned if invalid. If of type array, an array object type is returned.

```
var value = project.getRecipeItem("recipeName", "Set", "Element");
```

setRecipeItem

```
number setRecipeItem (recipeName, recipeSet, recipeElement, value )
```

Gets the value of the given recipe set element.

Parameter	Description
recipeName	String representing the recipe name
recipeSet	String representing the recipe set, can be either the recipe set name or 0 based set index.
recipeElement	String representing the recipe Element, can be either the element name or 0 based element index.
value	An object containing the value to store in the recipe. It can be an array type.

Return value

Interger value for denoting success and failure of action. A ‘0’ means success and ‘-1’ means failure.

```
var val = [2,3,4];
project.setRecipeItem("recipeName", "Set", "Element", val);
if (status == 0) {
     // Success
} else {
     // Failure
}
```

downloadRecipe

```
void downloadRecipe (recipeName, recipeSet )
```

Downloads the recipe set to the corresponding tag.

Parameter	Description
recipeName	String representing the recipe name
recipeSet	String representing the recipe set, can be either the recipe set name or 0 based set index.

```
project.downloadRecipe("recipeName", "Set");
```

uploadRecipe

```
void uploadRecipe (recipeName, recipeSet )
```

Uploads the value of tags into the provided recipe set.

Parameter	Description
recipeName	String representing the recipe name
recipeSet	String representing the recipe set, can be either the recipe set name or 0 based set index.

```
project.uploadRecipe("recipeName", "Set");
```

launchApp

```
void launchApp( appName, appPath, arguments, singleInstance)
```

Executes an external application.

Parameter	Description
appName	String containing the application name
appPath	String containing the application absolute path
Arguments	String containing the arguments to be sent to application
singleInstance	true = only single instance allowed, false = multiple instances allowed

```
project.launchApp("PDF.exe","\\Flash\\QTHMI\\PDF","\\USBMemory\\file.pdf","true");
```

printGfxReport

```
void printGfxReport( reportName, silentMode)
```

Prints the graphic report specified by reportName.

Parameter	Description
reportName	String containing the report name
silentMode	True = silent mode enabled. No printer settings dialog is displayed.

```
project.printGfxReport("Report Graphics 1", true);
```

printText

```
void printText( text, silentMode)
```

Prints a fixed text.

Parameter	Description
text	String to print
silentMode	True = silent mode enabled. No printer settings dialog is displayed.

```
project.printText("Hello I Am Text Printing",true);
```

printBytes

```
void printBytes( text, silentMode)
```

Prints a hexadecimal string representing data to print. For example, "1b30" to print < ESC 0 >

Parameter	Description
text	Hexadecimal string to print
silentMode	True = silent mode enabled. No printer settings dialog is displayed.

```
project.printText("Hello I Am Text Printing",true);
```

emptyPrintQueue

```
void emptyPrintQueue()
```

Empties the print queue. Current job will not be aborted.

```
project.emptyPrintQueue();
```

pausePrinting

```
void pausePrinting();
```

Suspends printing operations. Will not suspend the print of a page already sent to the printer.

```
project.pausePrinting();
```

resumePrinting

```
void resumePrinting();
```

Resumes previously suspended printing.

```
project.resumePrinting();
```

abortPrinting

```
void abortPrinting();
```

Aborts current print operation and proceed with the next one in queue. This command will not abort the print of a page already sent to the printer.

```
project.abortPrinting();
```

printStatus

```
project.printStatus;
```

Returns a string representing current printing status.

Status string	Description
error	An error occurred during printing
printing	Ongoing printing
idle	System is ready to accept new jobs
paused	Printing has be suspended

```
var status = project.printStatus;
project.setTag("PrintStatus",status);
```

printGfxJobQueueSize

```
project.printGfxJobQueueSize;
```

Returns the number of graphic reports in queue for printing.

```
var gfxqueuesize = project.printGfxJobQueueSize;
project.setTag("printGfxJobQueueSize",gfxqueuesize);
```

printTextJobQueueSize

```
project.printTextJobQueueSize;
```

Returns the number of text reports in queue for printing.

```
var textjobqueuesize = project.printTextJobQueueSize;
project.setTag("printTextJobQueueSize",textjobqueuesize);
```

printCurrentJob

`project.printCurrentJob;`

Returns a string representing current job being printed

```
var currentjob = project.printCurrentJob;
project.setTag("printCurrentJob",currentjob);
```

printActualRAMUsage

`project.printActualRAMUsage;`

Returns an estimate of RAM usage for printing queues

```
var myVar = project.printActualRAMUsage;
alert(" actual ram usage is   "+ myVar);
```

printRAMQuota

`project.printRAMQuota;`

Returns the maximum allowed RAM usage for printing queues

```
var ramquota = project.printRAMQuota;
project.setTag("printRAMQuota",ramquota);
```

printActualDiskUsage

`project.printActualDiskUsage;`

Returns the spool folder disk usage (for PDF printouts)

```
var myVar1 = project.printActualDiskUsage;
alert(" actual disk usage is   "+ myVar1);
```

printDiskQuota

`project.printDiskQuota;`

Returns the maximum allowed size of spool folder (for PDF printouts).

```
var ramquota = project.printRAMQuota;
var diskquota = project.printDiskQuota;
```

printSpoolFolder

```
project.printSpoolFolder;
```

Returns current spool folder path (for PDF printouts).

```
var spoolfolder = project.printSpoolFolder;
project.setTag("printSpoolFolder",spoolfolder);
```

printPercentage

```
project.printPercentage;
```

Returns current job completion percentage (meaningful only for multipage graphic reports)

```
var percentage = project.printPercentage;
project.setTag("printPercentage",percentage);
```

Project object widgets

getCurrentPageName

```
string getCurrentPageName()
```

Return the name of current active page

```
// Get PageMgr widget
var pageMgr = project.getWidget( "_PageMgr" );

// Show Current Page
var currentPageName = pageMgr.getCurrentPageName();
project.showMessage( "Current active page is: " + currentPageName );
```

(Available on web pages)

hasPage

```
boolean hasPage(string pageName)
```

Return true if the page exist, false otherwise

```
// Get PageMgr widget
var pageMgr = project.getWidget( "_PageMgr" );

//Page exists
var pageExists = pageMgr.hasPage( "Page10" );
if (pageExists) {
    project.showMessage( "Page10 exists" );
} else {
```

```
    project.showMessage( "Hei Page10 not exists!" );
}
```

(Available on web pages)

State object

This is the class holding the state of a tag acquired from the controlled environment.

State object methods

Methods to be used with state objects.

getQualityBits

```
number getQualityBits()
```

Returns an integer - a combination of bits indicating tag value quality.

```
var state = new State();
var value = project.getTag("Tag1", state, 0);
var qbits = state.getQualityBits();
```

(Available on web pages)

getTimestamp

```
number getTimestamp()
```

Returns time the value was sampled.

Return value

A number containing the timestamp (for example 1315570524492).

Note: Date is a native JavaScript data type.

```
var state = new State();
var value = project.getTag("Tag1", state, 0);
var ts = state.getTimestamp();
```

isQualityGood

```
boolean isQualityGood()
```

Returns whether the value contained in this state object is reliable.

Return value

A Boolean true if quality is good, false otherwise.

```
var state = new State();
var value = project.getTag("Tag1", state, 0);
if (state.isQualityGood()) {
     // do something…
}
```

(Available on web pages)

Keywords

Global objects are predefined and can be referenced by the following names.

page

`object page`

References the page object for the current page.

```
function btnStd04_onMouseRelease(me) {
     var wgt = page.getWidget("rect1");
     var name = wgt.objectName;
}
```

project

`object project`

References the project widget.

```
var group = new Group();
project.getGroup("GroupName", group);
var value = group.getCount("Tag1");
```

Global functions

print

`void print( message )`

Prints a message to the HMI Logger window.

Parameter	Description
message	Message string

```
print("Test message");
```

alert

```
void alert( message )
```

Displays a pop-up dialog with the given message. The user must press the **OK** button in the dialog to continue with the execution of the script.

Parameter	Description
message	Message string

Note: The alert function may be used for debugging JavaScript functions.

```
alert("Test message");
```

(Available on web pages)

Handling read/write files

Create folder

```
boolean fs.mkdir(strPath);
```

Creates a folder, if not already existing, in the specified path. Returns true on success and false if it fails.

Parameter	Description
strPath	Path string

Remove folder

```
boolean fs.rmdir(dirPath);
```

Remove directory at strPath if exists and empty. Returns true on success and false if it fails.

Parameter	Description
dirPath	Folder string

Read folder content

```
object fs.readdir(dirPath);
```

Reads the contents of a folder. Returns an array of the names of the files in the folder excluding '.' and '..'. Returns empty list if it fails.

Parameter	Description
dirPath	Folder string

Read file

```
object fs.readFile(strfile [,strFlag]);
```

Opens the strFile file in read mode, reads its contents and returns it.

Parameter	Description
strFile	File name string
strFlag	Read file mode: “b” reads and returns as binary file (otherwise returns a text file)

Write file

```
fs.writeFile(strFile, fileData, [strFlag]);
```

Creates the strFile file if not present. Opens the strFile file in write mode and writes the data fileData to the file.

Parameter	Description
strFile	File name string
fileData	Data to be write on the file in byte array
strFlag	Write file mode: • “a”: appends fileData to the end of the text file • “r”: replaces the contents of the file with fileData • “ab”: appends fileData to the end of the binary file • “rb”: replaces the contents of the binary file with fileData

Default flag is for writing text file in append and write mode. File path will be created if not present.

Returns -1 if write error occurs.

Append file

```
int fs.appendFile(strFile, fileData);
```

If the files does not exist creates it, otherwise append to existing file. Returns the number of character written or -1 on error.

Parameter	Description
strFile	File name string
fileData	Data to be write on the file in byte array

File exists

```
boolean fs.exists(strPath)
```

Returns true if the file or folder exists at strPath.

Parameter	Description
strPath	Path string

Remove file

```
boolean fs.unlink(strPath)
```

Removes the given file at strPath from filesystem if exists. Returns true on success and false if it fails.

Parameter	Description
strPath	Path string

File status

```
object fs.stat(strPath)
```

Retrieves information on the file/folder present at the specified path.

Parameter	Description
strPath	File/folder path string

```
var fileStats = var fs.stat(strPath)
```

fileStats.isFile	True if path is a file
fileStats.isDir	True if path is a folder
fileStats.size	Size in bytes of that file
fileStats.atime	Date object representing the last read access time
fileStats.mtime	Date object representing the last write access time
fileStats.ctime	Date object representing the creation time
fileStats.perm	File permissions

If path is invalid both isFile and isDir fields return false.

File permission table

0x4000	File is readable by the owner of the file
0x2000	File is writable by the owner of the file
0x1000	File is executable by the owner of the file
0x0400	File is readable by the user
0x0200	File is writable by the user

0x0100	File is executable by the user
0x0040	File is readable by the group
0x0020	File is writable by the group
0x0010	File is executable by the group
0x0004	File is readable by anyone
0x0002	File is writable by anyone

Important notes on file handling

Path for files and folders are expected to be UNIX style. This means the backslash character (\) is not recognized. Use slash character (/) instead.

File system object is a client side object. So operations are performed on local file system, not on server file system.

Current JavaScript API to get access at the device file system has been designed to manipulate small files. When a file is read, the entire file contents is temporarily stored inside the RAM available for JavaScript environment (16MB) and an exception is raised when there is not enough available memory. Good programming practice is to include the `fs.readFile()` call inside a try/catch block.

Limitations in working with widgets in JavaScript

Widgets cannot be instantiated by JavaScript, they can only be accessed and changed. If you need additional widgets on the page, you can add hidden widgets on the page, and then display or position them using JavaScript.

Debugging of JavaScript

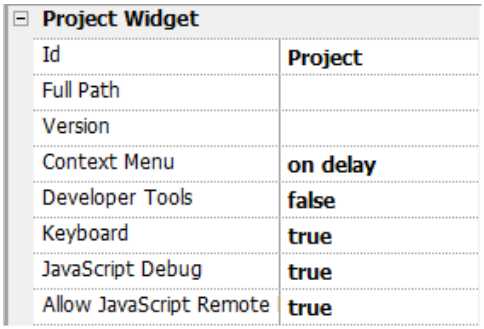
Wizard and HMI Runtime include a JavaScript debugger.

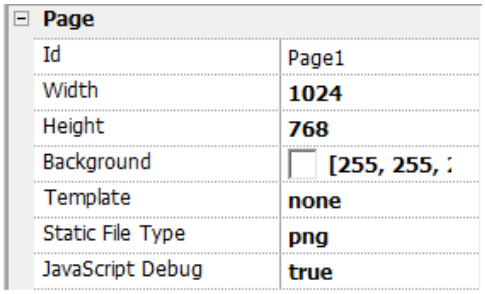
Two types of debuggers are available:

- Runtime debugger: a debugger running directly on the HMI device
- Remote debugger: a debugger running on a remote computer connected to the HMI device via Ethernet (usually computer running Wizard)

Enabling debugging

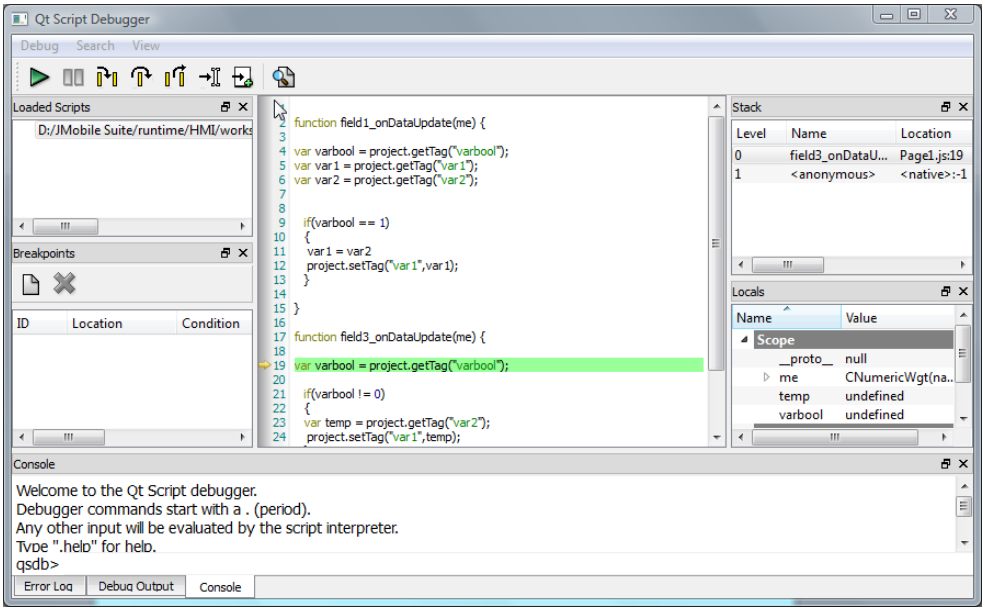
In the **Properties** pane of a page, set **JavaScript Debug** to **true**.





For schedulers and alarms debugging, enable JavaScript Debug in Project properties.

In the HMI Runtime, when the events are called, the debugger will show the debug information. In the **Locals** pane you can inspect all variables and elements.



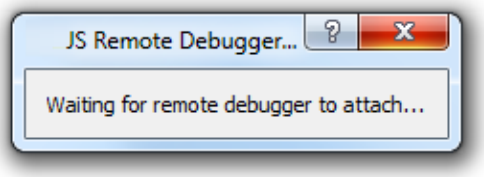
For a complete reference guide about JavaScript Debugger refer to :

http://qt-project.org/doc/qt-4.8/qtscriptdebugger-manual.html

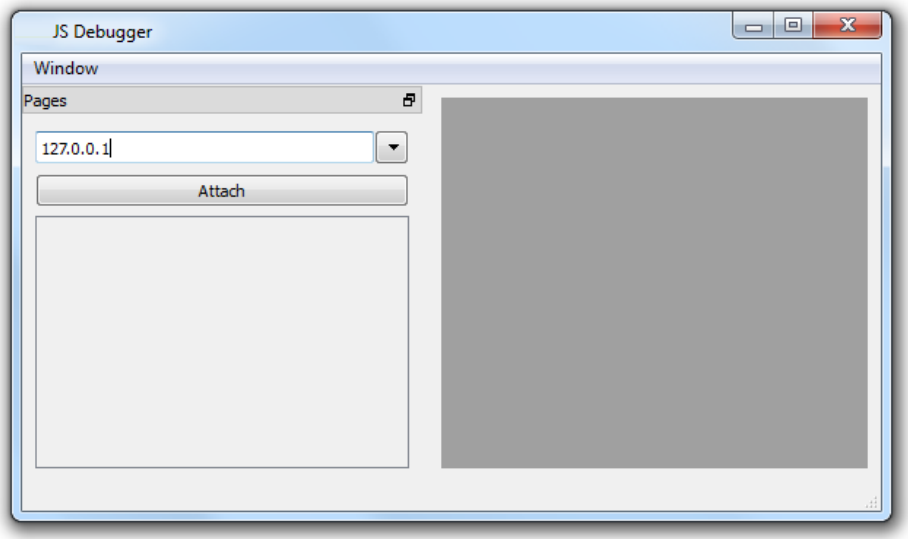
Remote JavaScript Debugger

Path: ***Run> Start JS Remote Debugger***

1. Set the **Allow JavaScript Remote** and the **JavaScript Debug** parameters in the project Properties to true in all the pages where debugging is required.
2. Download the project: the following message is displayed on the runtime.



3. In the **JS Debugger** window, select the IP of the HMI device and click **Attach** to connect the debugger to the HMI device.



Remote JavaScript debugger connects to HMI Runtime using port 5100/TCP.

Note: The Remote JavaScript debugger tool is not supported in HMI Client.

JavaScript Memory Usage

When the memory exceeds the maximum, an out of memory exception is thrown with a custom message. Please note that we don't have a fine control over the actual memory usage so it is mainly a soft limit. Moreover we can't forbid the allocation (this will break the engine implementation), so exception is thrown only when the memory is already over the limit. Before raising the exception, a garbage collection is forced to see if some memory can be freed.

JavaScrip memory limit can be accessed from the global object **$EngineMemory**. The default is 16MB, which should be enough for the typical JavaScript usage (mainly control, without many allocations).

- $EngineMemory.setLimit()
  set maximum memory allowed for JavaScript (the default limit is 0x00FFFFFF)
- $EngineMemory.getLimit()
  get maximum memory allowed for JavaScript
- $EngineMemory.getSize()
  get currently used memory from JS (fastMallocStat)

Test memory exception

To generate and test memory exception you can use the following snipped. Please note that we need to reset the memory limit to 0xffffffff to be able to run the alert, otherwise the memory allocations required to pop up the alert would fail.

```
try
{
    // Generate out-of-memory error
    var a = [];
    while(1)
    {
        a.push("a");
    };
} catch(e)
{
    // Ensure there is enough memory to pop up error message
    $EngineMemory.setLimit(0xffffffff);
    alert("Exception: " + e);
};
```

37 Handling Gestures

Some widgets have the capability to detect and manage pan and pinch gestures.

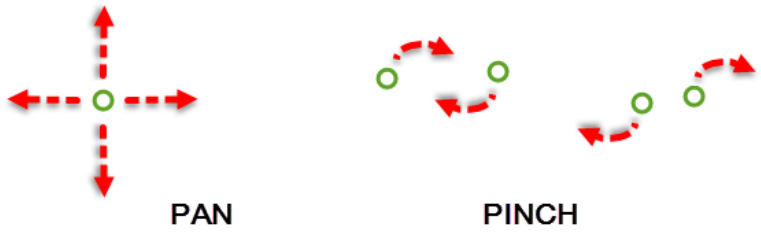
- Trends (see "Trend widget gestures" on page 194 for details)
- Gesture Area Widget. Special widget designed to customize handling of gesture events (see "Gesture area widget" on page 292 for details)

For widgets based on table presentation, when the **ScrollType** parameter has been set to Dynamic, the pan gesture is used to smoothly scroll the table.

- Alarms
- Control List

WARNING: pinch gesture requires two fingers. It is available only with HMI devices supporting multi touch operation (see "HMI devices capabilities" on page 429

Tip:Using multi touch HMI device you can implement safe commands by programming a command to be executed only when two buttons are pressed at the same time.



38 System Settings

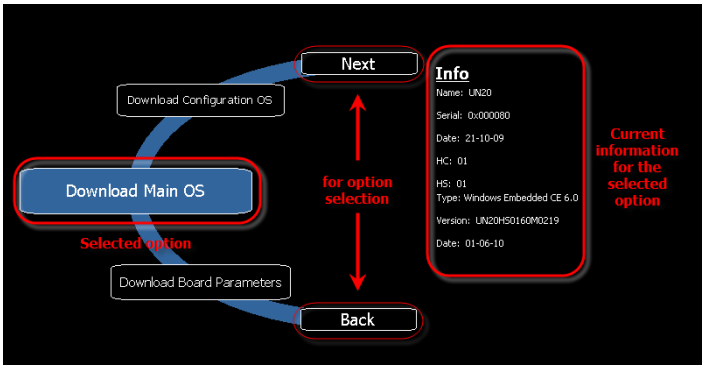
System Settings is an internal tool of the HMI device that can be used for the basic device settings or for the system components update.

Note: the system components can be update even from the Wizard (see "Updating system components in HMI devices" on page 411 for details)

CAUTION: Working with the System Settings tool is a critical operation and, when not performed correctly, may cause product damages requiring service of the product. Contact technical support for assistance.

System Settings on WinCE Devices

The System Settings tool includes a rotating menu, and navigation buttons to scroll between the available options.



For each function and component on the left, the **Info** pane on the right displays all available information. In the example the version of the Main OS component is shown.

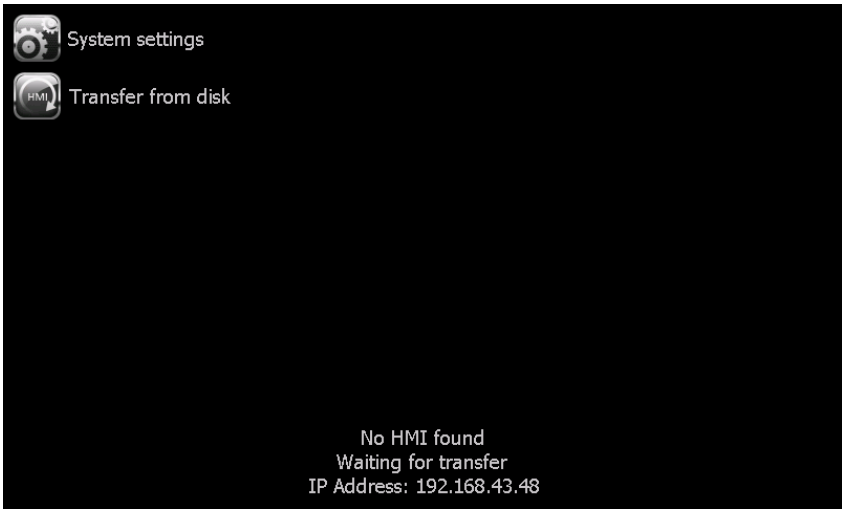
The System Settings tool can be used in two operating modes:

- User mode
- System mode.

For each mode different options are available.

Runtime Installation

HMI devices are delivered from factory without Runtime, at first power up HMI shows the “Runtime Loader” screen.



Runtime can be installed:

- Automatically, via Ethernet on first project download with Wizard
- Manually via USB Memory, creating an “Update Package”

Install Runtime via Ethernet

To install Runtime via Ethernet follow the "Download to HMI device" on page 70procedure.

WARNING: Runtime installation via Ethernet download requires the HMI to have a valid IP address.

The IP address can be assigned in three ways:

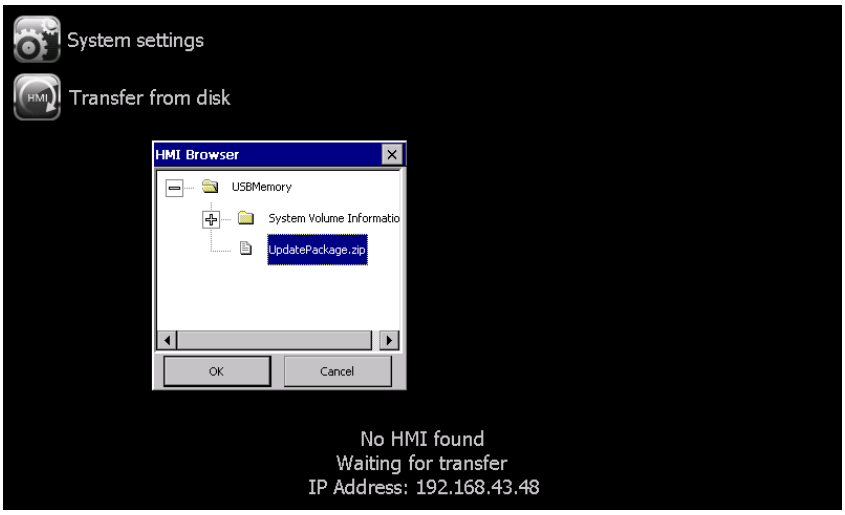
- *Automatically via DHCP server.* This option is enabled by default. If a DHCP server is available on the network IP address will be assigned automatically by the server.
- *Automatically via Auto-IP feature.* If DHCP assignment is enabled but no DHCP server is available on the network the HMI assigns itself an IP Address into range 169.254.x.x with subnet mask 255.255.0.0
- *Manually via System Settings.* From System Settings menu, in Network section the IP address can be manually assigned, disabling the DHCP server assignment feature.

Install Runtime via USB Memory

To install Runtime, UpdatePackage or Backup Package via USB device follow this procedure:

1. Create an Update Package from Wizard and copy into an empty USB memory stick

2. On HMI select [Transfer from disk] and select the UpdatePackage.zip to load.



System Settings

System Settings has two operating modes:

- User Mode
  a simplified interface that gives users access to the basic settings of the HMI device.
- System Mode
  a full interface that gives users access to all the tool's options.

When you access the tool at runtime selecting *"Show system settings"* from the context menu, the tool is started by default in User Mode.

Note: Press and hold on a screen area without buttons or other touch sensitive elements to display the context menu.

To access System Mode:

- Execute a tap sequence on the touch screen during the power-up phase. A tap sequence is a high frequency sequence of touch activations executed immediately after the device has been powered.
- From the System Setting page in User Mode, restart the panel in Configuration OS mode

Elements available in User Mode

Element	Description
Calibrate Touch	Calibrate the touch screen
Display settings	Control backlight inactivity timeout and brightness
Time	Set HMI device date and time manually or configure NTP servers
Regional Settings	Select or customize the regional setting parameters
BSP Settings	Display operating system version and unit operating timers to control buzzer and battery led.
Network	Sets IP address and other network settings
Plug-in List	List the plug-in modules installed and recognized by the system.  Note: this option may not be supported by all platforms and all versions.
Close	Closes the system setting page
Restart	Restart the HMI device • Main OS Restart the HMI device in the operating mode • Configuration OS Restart the HIM device with System Setting tool active in System Mode

Elements available in System Mode

In addition to those available in User Mode, the following features are also available:

Element	Description
Format Flash	Formats the internal device flash disk. All projects and the HMI Runtime will be erased, returning the device to its factory settings.
Restore Factory Settings	Restores factory settings as an alternative to Format Flash, in a more flexible way. The following options are available: **Uninstall HMI**: removes the HMI Runtime (entire qthmi folder) at the next start the device will behave as a brand new unit. This command does not reset settings such as IP address, brightness or RTC. **Clear System Settings**: resets system parameters (registry settings) and deletes the following files: *\\Flash\\Documents and Settings\\system.hv*

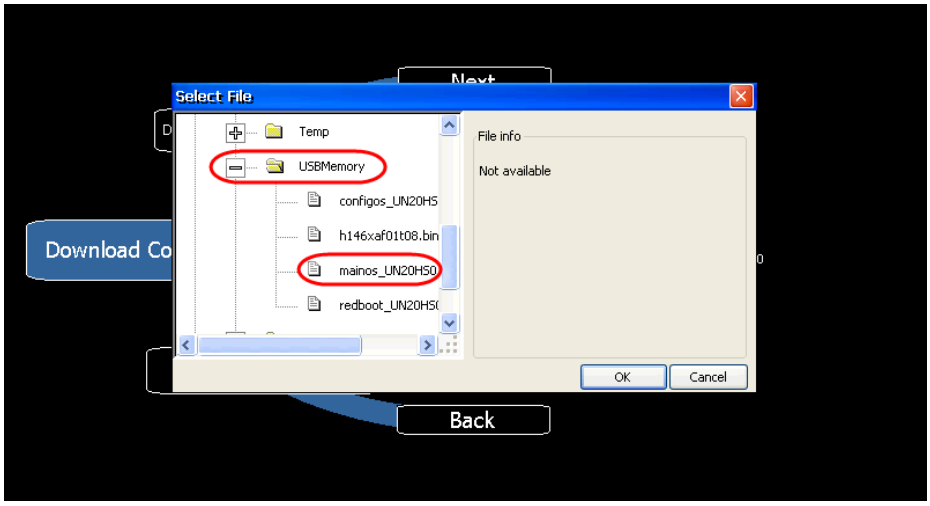
Element	Description
	*Flash**Documents and Settings**default**user.hv* *Flash**Documents and Settings**default.mky* *Flash**Documents and Settings**default.vol* System Mode password is also reset. **Clear sysdata settings**: clears \Flash\$SysData$ folder  *Service call: To be used only by technical support to fix display problems.*  Note: Not all these options are available for all HMI devices and BSPs.
Resize Image Area	Resizes the flash memory reserved to store the splash screen image displayed at power up. Default settings are normally suitable for all units.
Download Configuration OS	Checks and upgrades the current version of the operating system used in System Mode
Download Main OS	Checks and upgrades the current version of the main operating system
Download Splash Image	Loads a new file for the splash screen image displayed by the unit at power up.  Tip: Update the splash screen image directly from the Wizard programming software. See "Update of system components from the application" on page 412 for details.
Download Bootloader	Checks and upgrades the current version of the system boot loader.
Download Main FPGA	Checks and upgrades the current version of the main FPGA file. This function may not be available for all platforms and versions.
Download Safe FPGA	Checks and upgrades the current version of the backup copy of the FPGA file. This function may not be available for all platforms and versions.
Download System Supervisor	Checks and upgrades the current version of the system supervisor firmware (used for the RTC and power supply handling).
Upload Configuration OS **Upload Main OS** **Upload Splash Image** **Upload Bootloader** **Upload Main FPGA** **Upload Safe FPGA** **Upload System Supervisor**	Copy the system files from the operator panel on the external device (usually an USB stick).

Update System Components

System components can be updated using a USB flash drives. For each component, a couple of specific update files are provided.

Note: Upgrading procedures depend on hardware and operating system versions. Contact technical support for assistance.

1. Copy all the upgrade files you need to a USB drive and plug it into the USB port of the HMI device.
2. Start the System Settings tool in System Mode (see "System Settings" on page 375for details).
3. Click on the desired download function.
4. Browse the content of the USB drive to the files to download. The example shows Main OS components.



5. Click **Download** to transfer files to the HMI device.

   Note: From this dialog click **Upload** to transfer files to the USB device.

6. Follow the instructions displayed to complete the update: the progress of the operation in displayed in a progress bar.

   This operation may require a few minutes.

Important: Do not turn off the device while a system component is being upgraded.

List of upgradable components

The HMI devices support the upgrade of the following components:

Component	Description
Application	The HMI Application and the HMI Runtime generated from the **Run> Update Package** command
Main OS	Main Operating System
Configuration OS	Backup operating system that ensures units recovery in case of main operating system corruption
Splash	The initial screen shown during the startup of the HMI device
Bootloader	Loader to handle device startup
Main FPGA	FPGA firmware
Safe FPGA	Backup copy of the Main FPGA that ensures unit booting in case of main FPGA corruption  **Important: Use the same file for updating Main and Safe FPGA components.**
System Supervisor	Firmware of the system supervisor controller (for example: packaged_GekkoZigBee_v4.13.bin). *The System Supervisor component can be upgraded from v4.13 or above.*  **Important: Do not try to update versions V4.08, V4.09, V4.10 and V4.11 since they do not support automatic update from System Settings.**

Touchscreen calibration

System Setting Calibration allows to calibrate Touchscreen device, can be accessed from System Settings

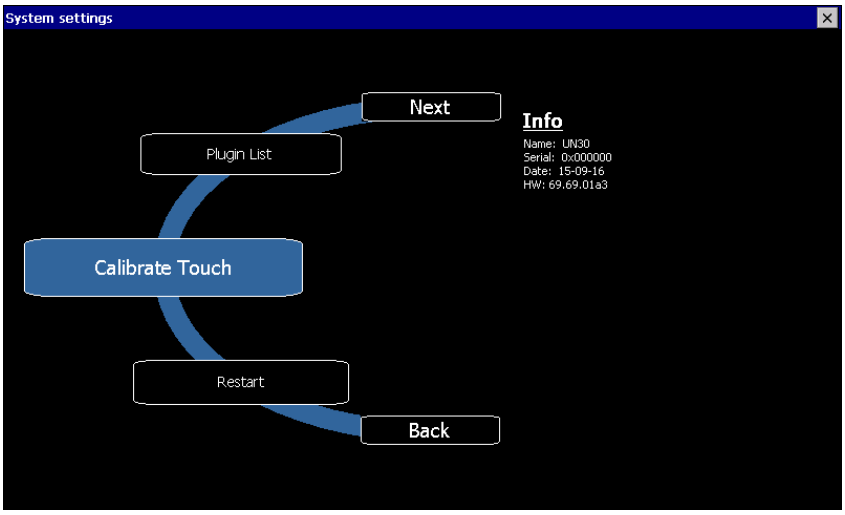
To access System Settings:

- Execute a tap sequence on the touch screen during the power-up phase. A tap sequence is a high frequency sequence of touch activations executed immediately after the device has been powered.

or

- Press and hold on an empty area of the screen for a few seconds to display the context menu.

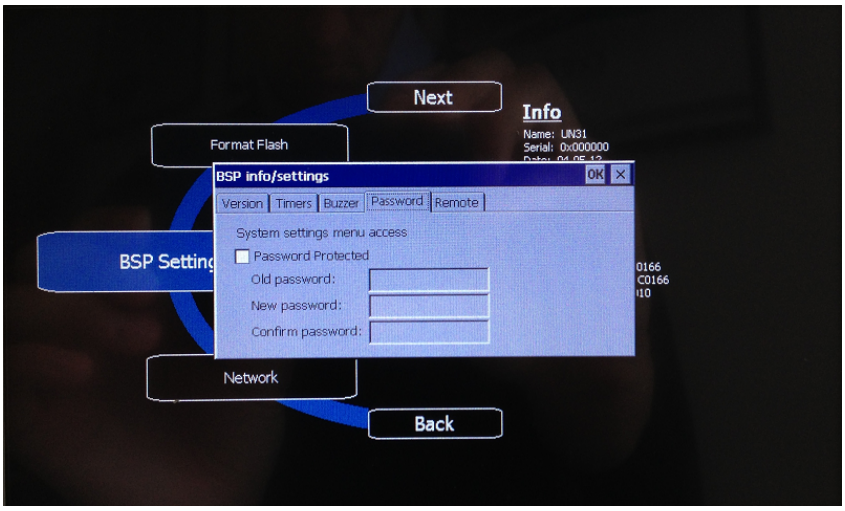
From the rotating menu, select "*Calibrate Touch*" and follow the instructions on screen to complete the calibration procedure, system will prompt to touch specific points to calibrate the touchscreen device.



Password protection

Internal password of the HMI device can be defined from the System Settings in System Mode (see "System Settings" on page 375 for details)

From the rotating menu, select "*BSP Settings*" and then the Password tab to open the set password dialog.



The password must be at least 5 characters long.

Leave "Old password" empty as default if target password is not set.

This feature is available from BSP versions V1.64 ARM UN30/31 and V2.73 MIPS UN20 based on WCE OS.

Factory restore

If you're having problems with the HMI device, try and restore factory default settings from System Mode.

1. Enter **System Mode**.
2. Use one of the following operations available in rotating menu:
   - **Format Flash**, to clean the flash drive and registry configuration.
   - **Restore Factory Settings**, to clean only the select components.

Note: Both operations do not involve firmware factory restore (MainOS, ConfigOS, Bootloader, FPGA images, etc).

See "System Settings" on page 375 for details.

System Settings on Linux Devices

Linux products offer a powerful integrated tool called System Settings that allows management and upgrade of system components. Operations can be done directly on HMI or remotely using web browser.

Runtime Installation

HMI devices are delivered from factory without Runtime, at first power up HMI shows the “Runtime Loader” screen.

Runtime can be installed:

- Automatically, via Ethernet on first project download with Wizard
- Manually via USB Memory, creating an “Update Package”

Install Runtime via Ethernet

To install Runtime via Ethernet follow the "Download to HMI device" on page 70procedure.

WARNING: Runtime installation via Ethernet download requires the HMI to have a valid IP address.

The IP address can be assigned in three ways:

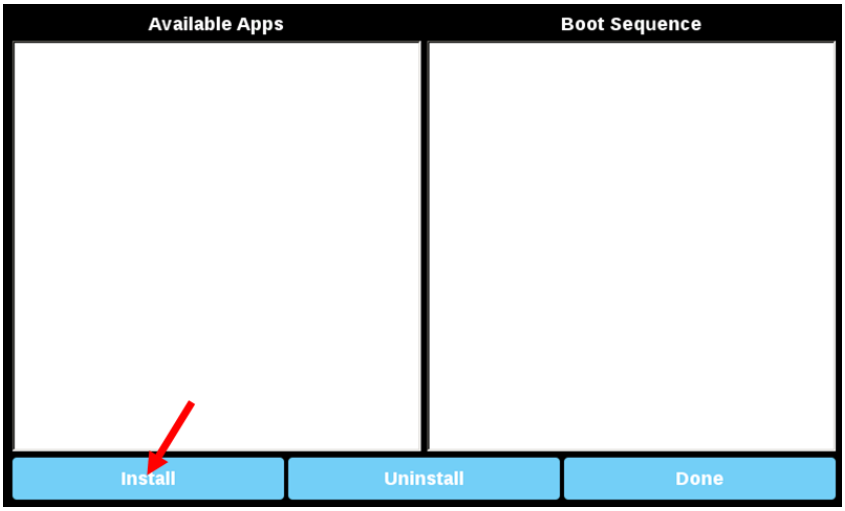
- *Automatically via DHCP server.* This option is enabled by default. If a DHCP server is available on the network IP address will be assigned automatically by the server.
- *Automatically via Auto-IP feature.* If DHCP assignment is enabled but no DHCP server is available on the network the HMI assigns itself an IP Address into range 169.254.x.x with subnet mask 255.255.0.0
- *Manually via System Settings.* From System Settings menu, in Network section the IP address can be manually assigned, disabling the DHCP server assignment feature.

Install Runtime via USB Memory

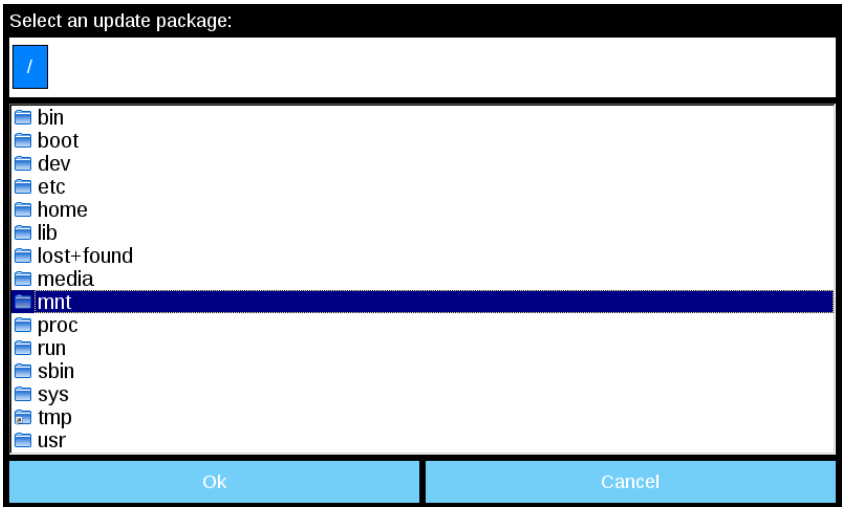
To install Runtime, UpdatePackage or Backup Package via USB device follow this procedure:

1. Create an Update Package from Wizard and copy into an empty USB memory stick

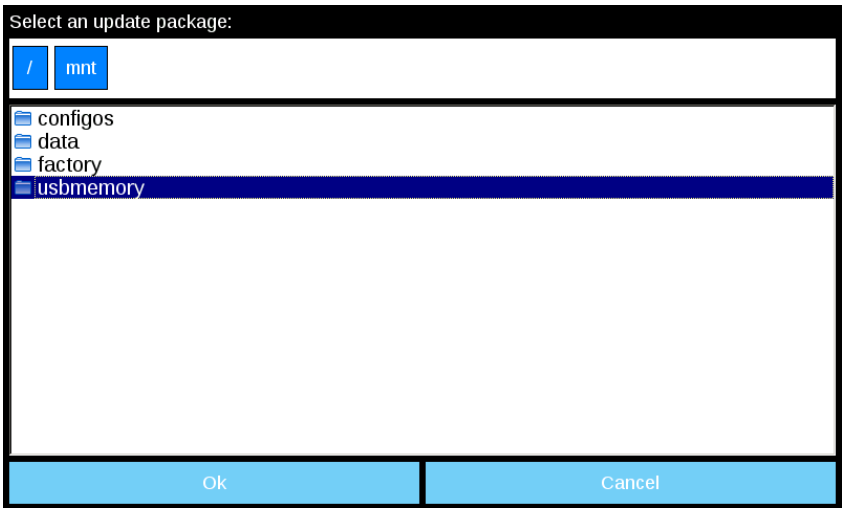
2. On HMI select [Startup sequence], then [Install]



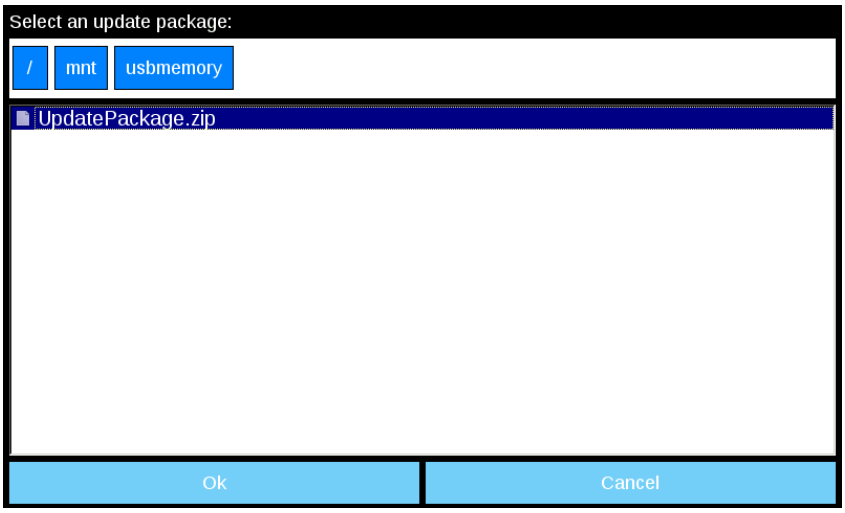
3. Double click on “mnt” to access this folder



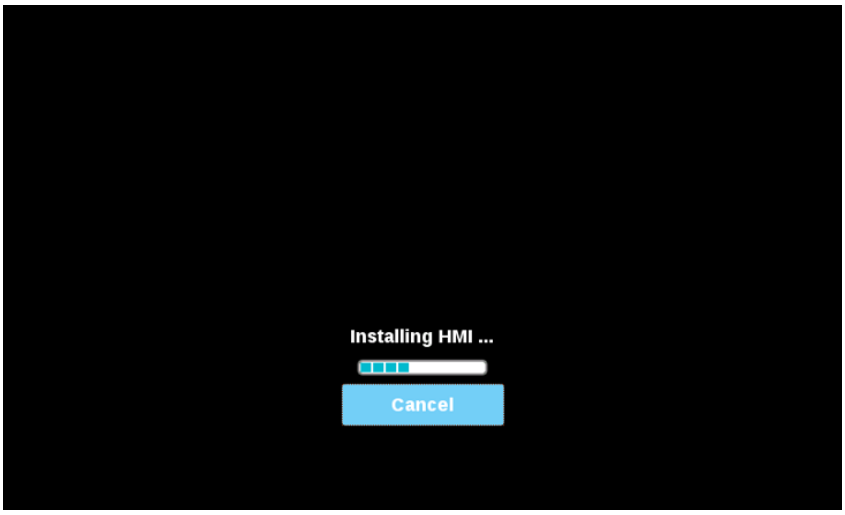
4. Then on “usbmemory”



5. Select “UpdatePackage.zip” and confirm with [Ok]



6. The runtime installation begin



Note: File systems supported are FAT16/32 and Linux Ext2, Ext3 and Ext4.

System Settings

The user interface of System Settings is based on HTML pages and can be accessed both locally on the HMI device screen and remotely using a Web browser.

Administrator username with full access right is "admin" with default password "admin". Generic username is “user” with default password “user”

WARNING: For security reasons, change the default passwords for both usernames (passwords can be modified from the “System Settings -> Authentication” command)

Accessing at the system settings from the HMI device do not require to enter a password until the default "admin" password is not changed.

System Setting access from Web browser

To access System Settings using a Web browser, enter the IP address of the device, in the following format:

https:///IP/machine_config

Note: Remote access requires port 443.

Browse through the options available in the menu on the left: the active item is highlighted and related information is displayed on the right.

Default security protocols proposed by the HTTPS server in the Linux HMI device are:

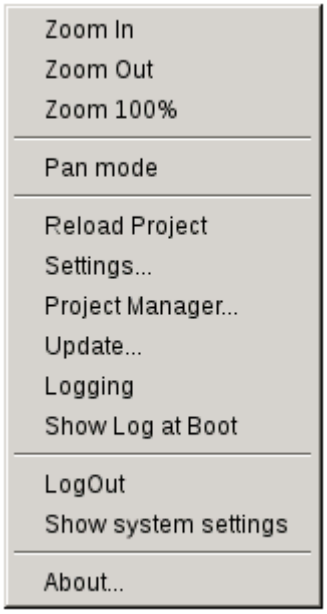
- SSLv3 256 bits ECDHE-RSA-AES256-SHA
- TLSv1 256 bits ECDHE-RSA-AES256-SHA

WARNING: We discourage usage of CBC cyber suites in the context of SSL3 or TLSv1.0 connections since potentially affected by some vulnerabilities.

System Setting access from HMI device

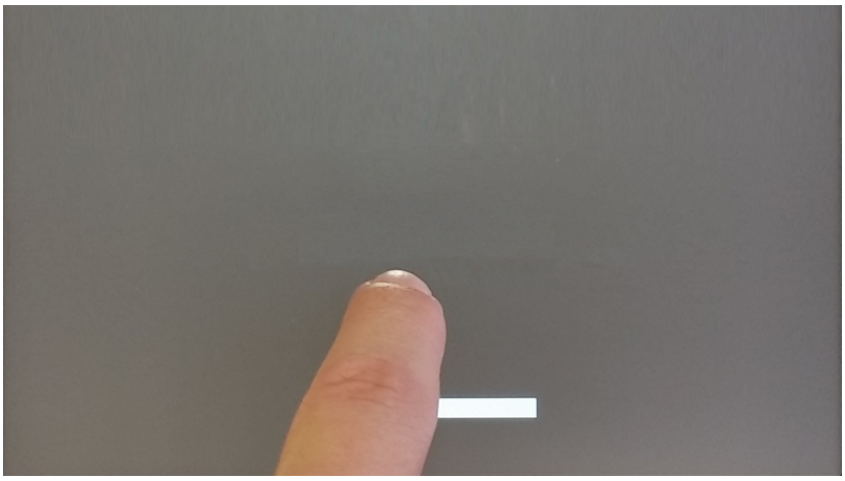
When Runtime is not installed, the System Settings is accessible from the Runtime Loader screen,

When Runtime is installed the System Settings is accessible selecting “Show System Settings” option of Context Menu,



Enter System Settings via tap-tap procedure

Tap-tap consists in a sequence of several touch activations by simple means of the finger tapping the touch screen performed during the power-up phase and started immediately after the HMI is powered on.



When “tap-tap detected” message appears on the top of the screen. Wait for 5 seconds (without touching the screen) to enter System Settings sub menu

Wait for 5 more seconds (without touching the screen) to enter Default Mode

Select "System Setting" from the HMI Default Mode screen

System Settings Sections

To change system settings values, enter in edit mode by click the edit button on the right top.

The edit button is available only inside the dialogs that contains modifiable parameters.

Languages

Select the language for the system settings interface

System

Parameter	Description
Info	Device information
Status	Device status (Free RAM, Up time, CPU Load)
Timers	Device timers (System on, Back light on)
PlugIn	Hardware plugins information

Logs

Device log files

Date & Time

Device date and time. Available parameters in edit mode:

Parameter	Description
Current Timezone	Timezone region
Current Date **Local Time**	Date and Time can set manually only when the Automatic Update is disabled.
Automatic Update (NTP)	Enable to keep date and time synchronized from a remote server

Parameter	Description
	• NTP Server Specify the Internet NTP Server address

Networks

Network parameters. Available parameter in edit mode:

Parameter	Description
General Settings	Device hostname
Network Interface	Network parameters of the available interfaces • DHCP • IP Address • Net Mask • Gateway
DNS	DNS Servers Generally provided from the DHCP servers, but can be modified in edit mode Search Domains Optional domains that will be used in concatenation with the provided urls

Services

Services are is available only when logged as admin.

Mouse click on the enable button to enable/disable the service. Click the service name to list the associate parameters.

Parameter	Description
Avahi Daemon	Avahi is a system which enables programs to publish and discover services and hosts running on a local network.
Cloud Service	Allow to manage remote HMI devices connected to a centralized server through gateways. • Server Type • Server • Username
Router Service	Enable routing between Ethernet adapters
SNMP Server	Enable the SNMP server
SSH Server	Enable the SSH server

Parameter	Description
System Logger	Enable system logger servcie
VNC Service	Enable VNC service • Port • Multiple clients • View only • Encryption • Authentication

Management

Management is available only when logged as admin.

From the management area is possible "Update System Components" below of the HMI device.

CAUTION: Working in the Management area is a critical operation and, when not performed correctly, may cause product damages requiring service of the product. Contact technical support for assistance.

Use the “Clear” command inside the “Data” section to remove HMI Runtime from the device (Factory Restore)

Display

Parameter	Description
Brightness	Brightness level of the display
Back light timeout	Backlight inactivity timeout
Orientation	Display orientation

Restart

HMI device restart command

Authentication

Enter in edit mode to change the authentication passwords.

EXIT

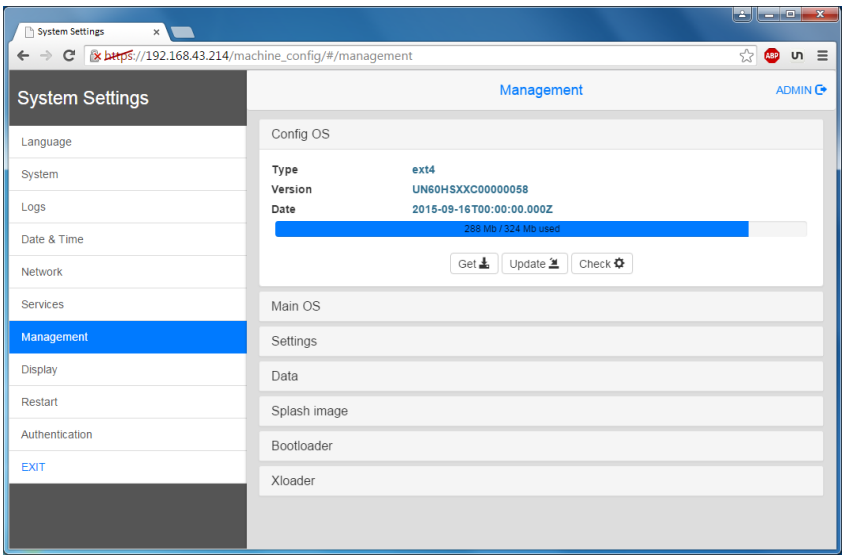
Exit from the System Setting tool.

Update System Components

CAUTION: Working in the Management area is a critical operation and, when not performed correctly, may cause product damages requiring service of the product. Contact technical support for assistance.

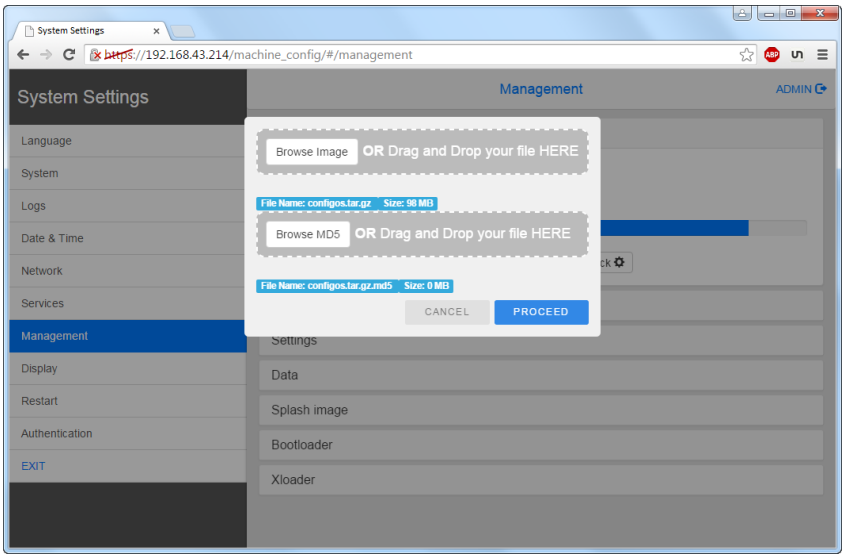
The system components of the Linux device can update locally using an USB memory key or remotely via web browser.

To update system components enter System Settings in Config OS mode via tap-tap procedure on HMI or open web browser to https://<HMI-IP-address> and select the “Management” section.



Expand the component to update and select [Update]

On the opened dialog, click [Browse Image], then select the “xxx-mainos-xxx.tar.gz” file. Click then on [Browse MD5] and select the “xxx-mainos-xxx.tar.gz.md5” file.



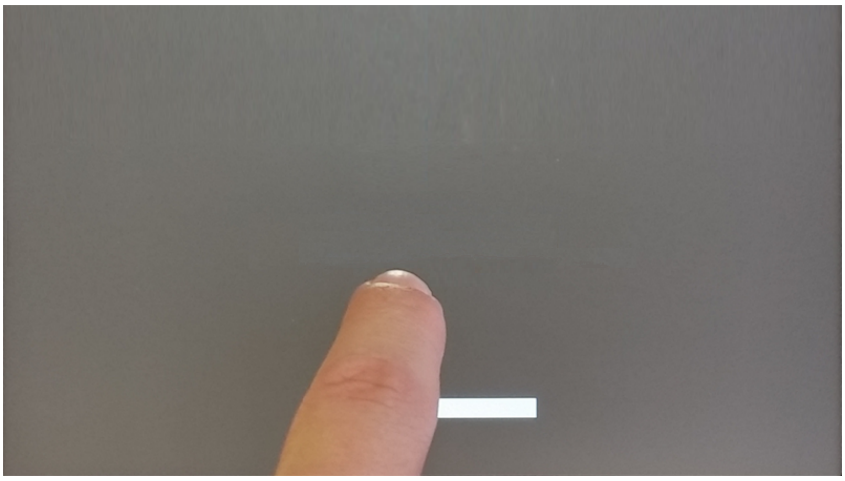
Important: Do not turn off the device while a system component is being upgraded.

At the end of the component update, restart HMI and leave it starting normally.

Enter System Settings in Config OS mode via tap-tap procedure

System Setting in Config OS mode is available via tap-tap sequence, this mode can be accessed also when HMI is facing a software failure.

Tap-tap consist in a sequence of several touch activations by simple means of the finger tapping the touch screen performed during the power-up phase and started immediately after the HMI is powered on.



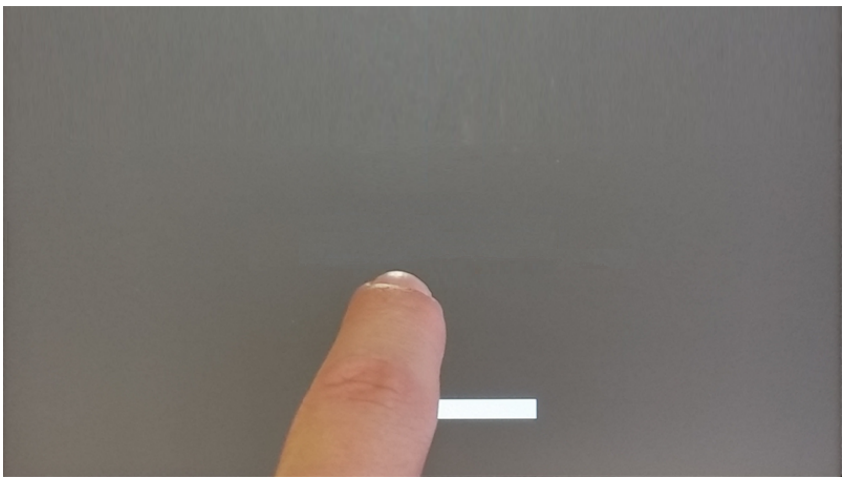
When “tap-tap detected” message appears on the top of the screen, press and hold the finger on touchscreen, to select “Restart: Config OS”

HMI will restart into System Settings in Config OS mode:

Touchscreen calibration

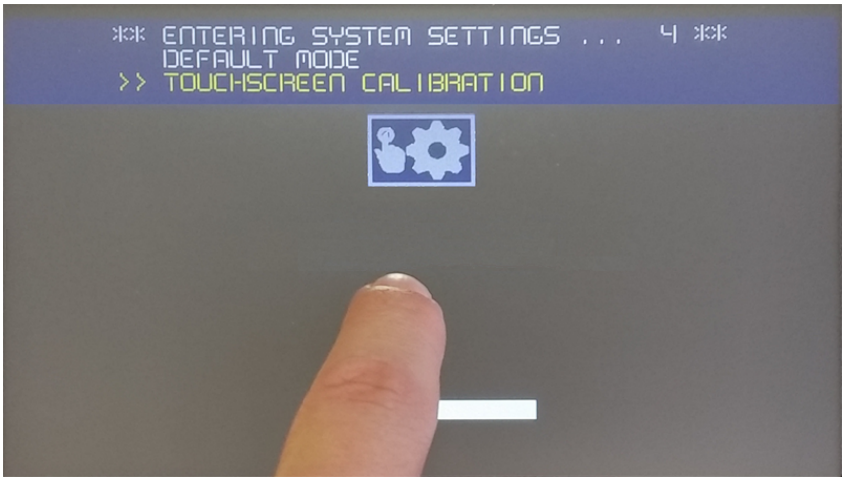
System Setting Calibration allows to calibrate Touchscreen device, can be accessed by tap-tap procedure.

Tap-tap consists in a sequence of several touch activations by simple means of the finger tapping the touch screen performed during the power-up phase and started immediately after the HMI is powered on.



When “tap-tap detected” message appears on the top of the screen, wait for 5 seconds (without touching the screen) to enter System Settings sub menu

Press on touch screen, “Touchscreen calibration” voice will be highlighted in yellow, hold pressed for few seconds until touchscreen calibration procedure starts

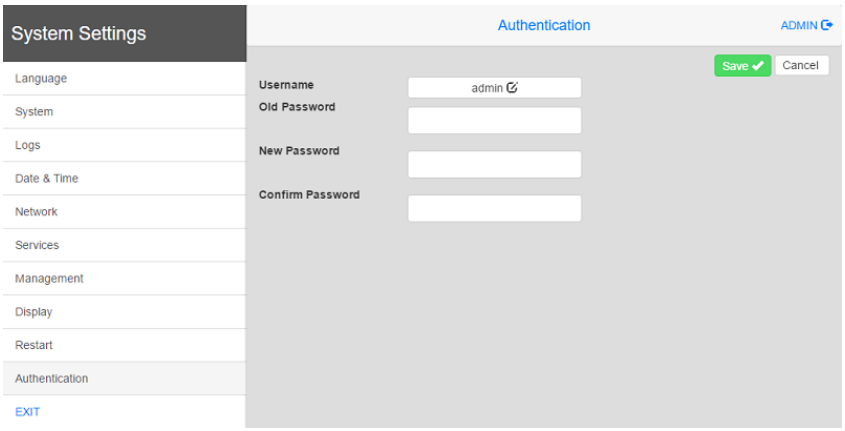


Follow the instructions on screen to complete the calibration procedure, system will prompt to touch specific points to calibrate the touchscreen device.

Password protection

Internal password of the HMI device.

From the Authentication tab, inside the "System Settings" on page 384, activate the edit mode and select the username to change the associated password.

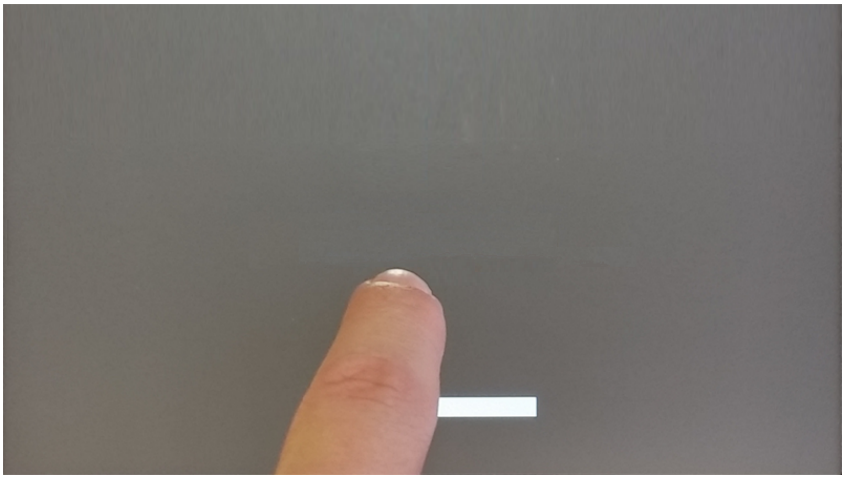


Password for admin user can modified even from the context menu of theHMI Runtime (see "Context menu options" on page 6 for details).

Factory restore

System Settings in Default mode allows to uninstall HMI Runtime or change Startup sequence, this mode is available via tap-tap sequence and can be accessed also when HMI is facing a software failure.

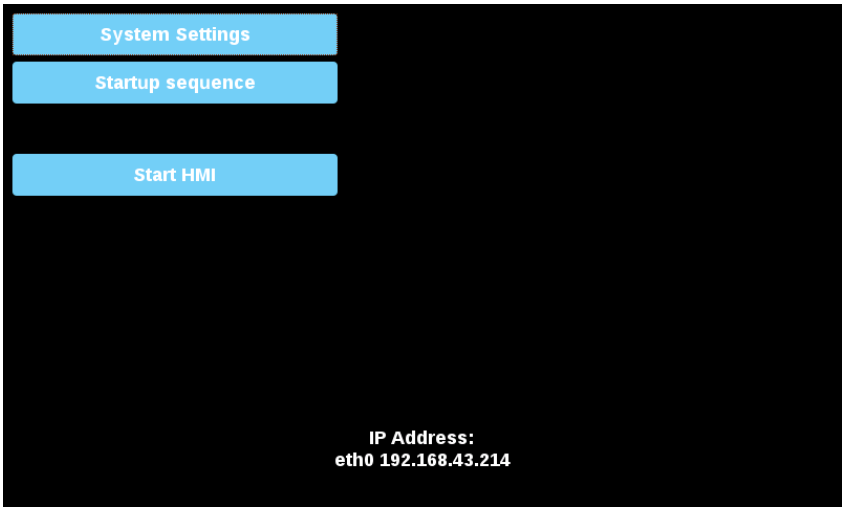
Tap-tap consists in a sequence of several touch activations by simple means of the finger tapping the touch screen performed during the power-up phase and started immediately after the HMI is powered on.



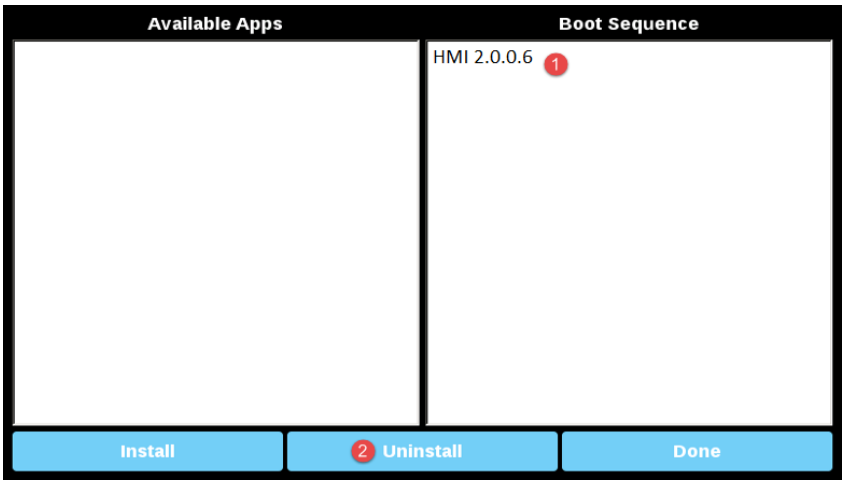
When “tap-tap detected” message appears on the top of the screen. Wait for 5 seconds (without touching the screen) to enter System Settings sub menu

Wait for 5 more seconds (without touching the screen) to enter Default Mode

To uninstall the Runtime from HMI in Default Mode screen select [Startup Sequence]:



Select the Runtime you want to remove (1) and click [Uninstall] button (2):



Runtime uninstall process will be performed:

39 Web access

SH4Web allows users to access HMI projects from a remote web browser running on a computer or on a mobile device such as a tablet or a phone. With SH4Web, users can create a web project to display at a remote location the same graphical display shown on the HMI device. SH4Web projects are based on HTML5 technology which means that no plug-ins or external software is needed for displaying the information.

This document assumes that you have a basic understanding of how to operate the web browser on your mobile devices as well as how to set up a connection to the HMI device where the server is running. For example, you must know how to set-up Wi-Fi access if you are working with tablet or phone devices to access the SH4Web pages on the HMI device.

Supported platforms and browsers

SH4Web supports 3 platforms:

- web, for desktop browsers,
- phone, for smart phone devices
- tablet, for tablet devices

You can therefore create pages of different content and size for the different platforms. For example, you may want to create a set of smaller pages in your project for phones whereas you will use full size pages for desktop web browsers and tablets.

Working with a computer

SH4Web works with all modern web browsers. The following browsers have been tested for compatibility with SH4Web:

- Mozilla Firefox 40+
- Microsoft Internet Explorer 11+
- Apple Safari 7.1+
- Google Chrome 36+

Working with tablets or phones

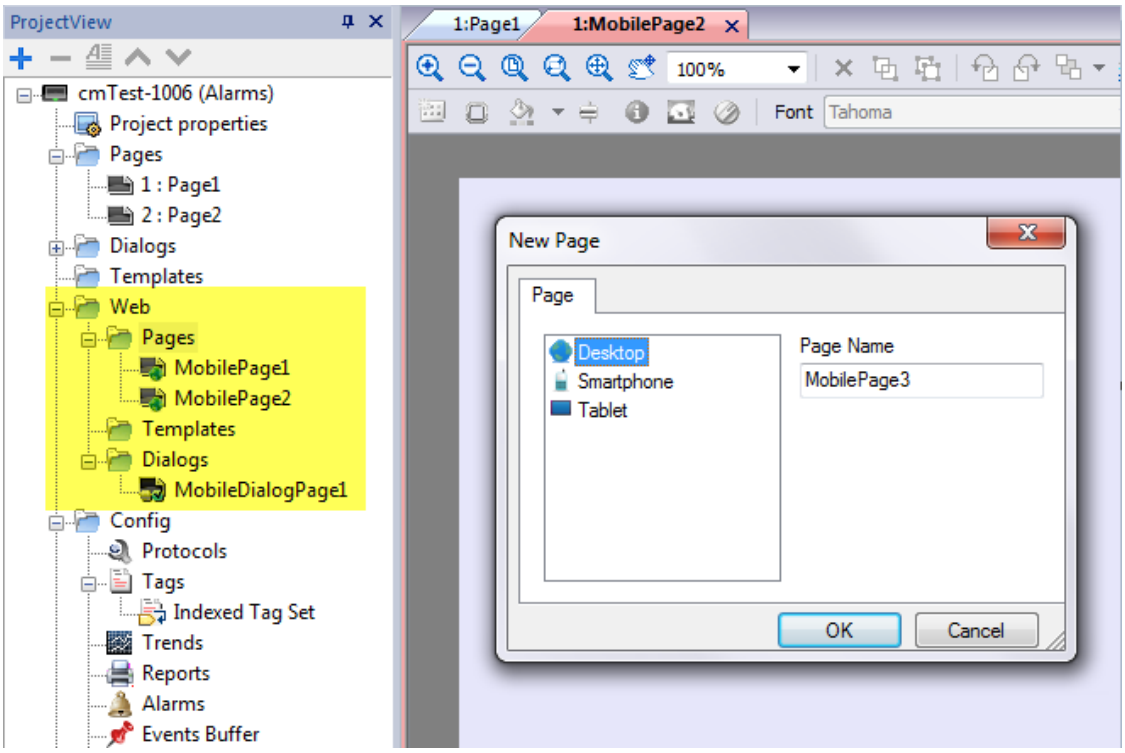
SH4Web works with most tablet and phone devices. The following tablets have been tested for compatibility with SH4Web:

- iOS 4+ - Mobile Safari
- Android 7+ - Android Webkit

Generating page for Web access

Path: ***ProjectView> Web> Pages***

Right-click the **Pages** node and select **Insert Page** to add a web page.



Any widgets and features can be used in Wizard; however, not all features are currently available in SH4Web. If the project includes a feature that is not available, SH4Web will still work correctly but the feature will not be available on the remote client device.

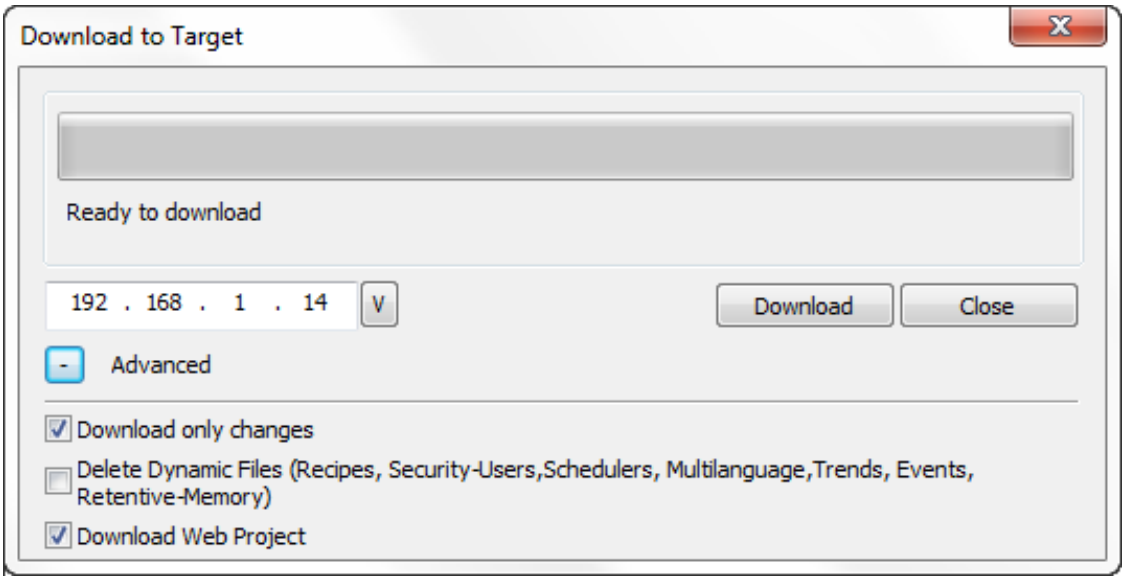
See "Web supported features" on page 403 for a list of the features supported in SH4Web and of the existing limitations.

Exporting pages

To select pages to export from the current project.

1. On the **Run** menu, click **Web Project Settings**: the **Web Project Settings** dialog is displayed.
2. Select the web pages you wish to export. By default all project and dialog pages are selected.
3. For each platform, select the home icon next to the page you want to define as the Home page. Only one Home page can be selected for each platform. All other home icons are grayed.

WARNING: When you download a project to the HMI device, make sure the Download Web Project option is selected.



Platform specific Home pages

The Home Page of the SH4Web project defines the first page that is shown in the browser of each platform type and defines the starting point for your web project. Pages that can be accessed from home page depend on the how other pages are linked in the project.

For example, if you have designed a set of pages for a phone platform, set as a Home Page a page appropriately sized for a mobile phone display. Then include in this page only links to other phone pages: the user will only access phone pages when browsing the SH4Web project from a phone.

Testing the Web project

You can test your SH4Web project using the online simulator opening a standalone web page directly from a browser.

Testing with the online simulator

Wizard includes an web server in the online simulator. You can start the simulator and access your SH4Web project from a web browser. The pages will be served from the simulator.

1. Create your project (see "Generating page for Web access" on page 398).
2. On the **Run** file, choose **Start Simulator**: the project will start running in a separate window.
3. Open a web browser (see "Supported platforms and browsers " on page 398 for a list of browser compatible with SH4Web).
4. Enter the following address: http://localhost:81: this tells the web browser to read the web pages from the local computer and use port 81, used by default by the online simulator in SH4Web.
5. Test your project in the browser.

Important: If you make any changes to the project pages in Wizard you must stop and restart the simulator.

Note: If you are using a device (for example, a smartphone) that is not the localhost where the simulator is running, you will be required to enter username and password.

Downloading the Web project

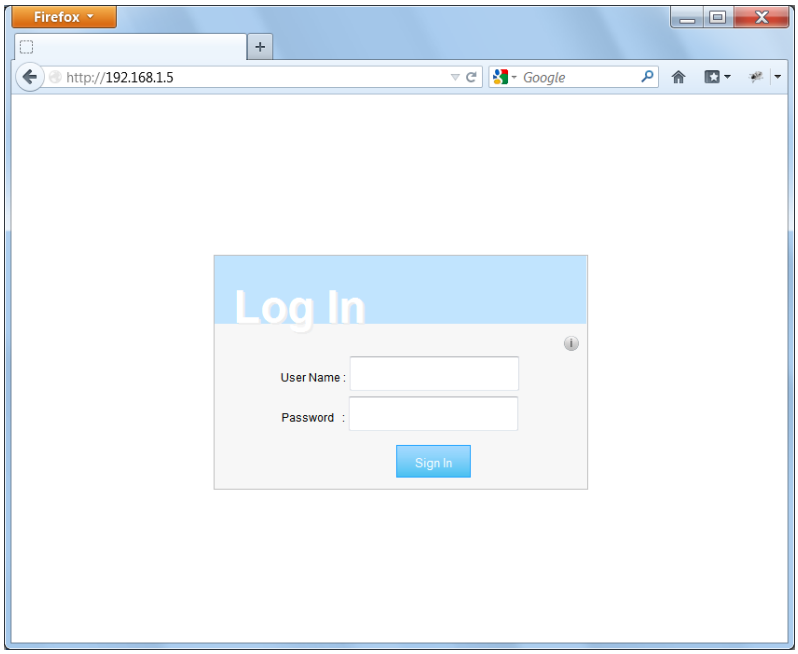
After testing the SH4Web pages, you can download the project to the desired HMI device.

The SH4Web project is downloaded together with the Wizard project, see "Download to HMI device" on page 70 for details.

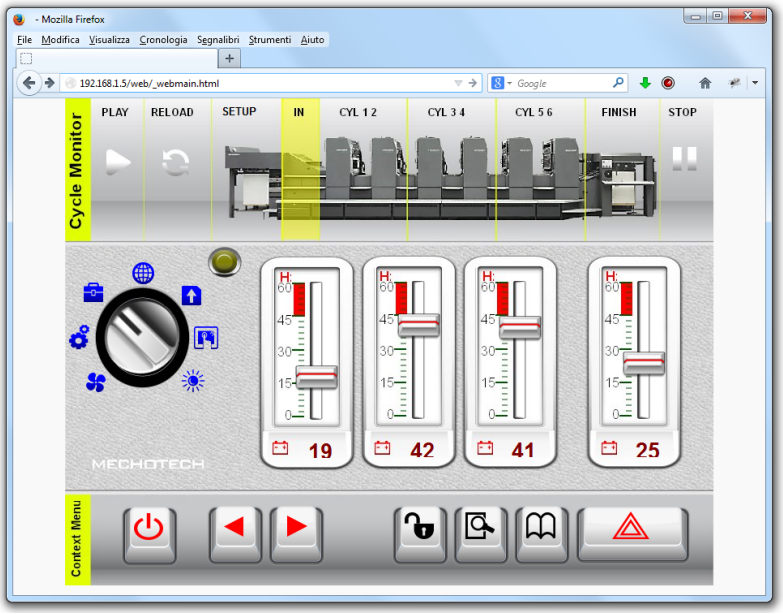
After the download process is completed, the HMI project automatically starts on the HMI device and the SH4Web project is ready to be used.

Running SH4Web from a browser

1. Open a web browser and enter the IP adress of your HMI device: the login page is displayed.



2. Enter **User Name** and **Password** and click **Sign In**: the Home page will be displayed.



See "User management and passwords" on page 227 for details on how to create credentials.

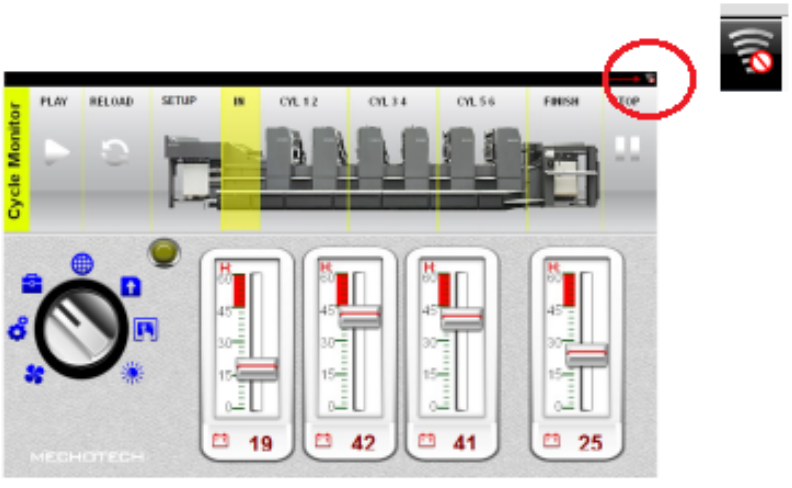
You can interact with the project using the browser in the same way you interact with a device when touching the screen: click buttons to change pages, view indicators and gauges, drag slider handles to change values, and so on. The SH4Web project will manage all communications with the web server while you are interacting with the HMI device remotely.

Web connectivity issues

Here are described the most common issues you might encounter when connecting remotely to your HMI device.

Server disconnection

Since SH4Web runs remotely from the HMI device, the server might disconnect from the browser (for example if the server is stopped or the network cable is unplugged). If this happens, a ‘disconnect’ icon will appear in a toolbar on top of the SH4Web as in this example.



Once the server is back online, the red circle-bar icon will disappear indicating normal communications with the device.

Note: If you make changes in the SH4Web pages while the server is disconnected, these changes will be visible on the client but will not be transferred to the server until the connection is restored.

Inactivity timeout

SH4Web will require you to re-enter your login credentials if the browser has been inactive for several minutes. If no activity is detected for 10 minutes, the login screen will reappear and you need to enter your login credentials to continue operation. A timeout feature guarantees that no unauthorized access is possible.The web inactivity timeout can be modified from the **Project Properties** table.

User session termination

A user session can be terminated either from the server or from the user.

In specific conditions the server might send a request to the client (browser) to perform the login process. In this case the user is redirected to the login page and then back to the page where he was working. This will happen for example if the user clears the browser cache or browser cookies.

Note: If the user is working in a dialog when redirected to the login page, he will be then redirected to the page from which the dialog was opened.

Non-Active SH4Web Project

The SH4Web page displayed in your browser might come from a project that is no longer active in the device. In this case a confirmation box is displayed and you can return to the active project.

Note: This redirection assumes that the current active project has SH4Web pages in it.

If you choose to stay in the non-active project all the actions you perform in the browser may not be executed properly as the SH4Web cannot perform any server-bound communication.

Web supported features

Currently not all Wizard features are supported in SH4Web. Here a list of features supported and limitations, classified by category.

When you copy and paste objects from standard pages to a web pages, make sure that all objects are supported in web pages. Eventually remove unsupported objects from the web page after paste.

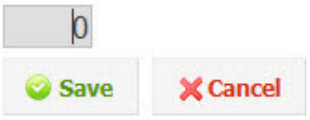
Category	Supported features	Limitations
Widgets	• Basic (Text/Numeric, Images, Shapes, Trends/Graphs, Recipes, Controls, Alarms, Texture) • Buttons • Meters • Switches • Lights • Media (IP Camera) • Icons • Factory Automation	• AttachToTag of system variables is not supported • Font files without web download permissions flag enabled are not loaded from the SH4Web • Widget properties with Attach to… dynamic behavior may not work for all properties supported by Wizard. • Multistate Image Multi-Layer is not supported. • Alarm Color based on trigger condition is not supported in Web • Can not edit the Alarm widgets in runtime
Alarms	• Alarms limits in SH4Web is the same of HMI device (500..2000 based on target)	• On Smartphone/Tablet (in general embedded devices) based on HW a user could expect performance problems with > 500 alarms.
Actions	• Widgets (Javascript) • Page (HomePage, LoadPage, NextPage, PrevPage, LastVisitedPage, ShowDialog, CloseDialog, ShowMessage, LaunchBrowser) • Multilanguage (SetLanguage) • Tag (WriteTag, StepTag, SetBit, ResetBit, ToggleBit)	• JavaScript is supported (see "JavaScript " on page 329 JavaScript chapter for a list of supported features)

Category	Supported features	Limitations
	• Trend/Graph (RefreshTrend, ScrollLeftTrend, ScrollRightTRend, PageLeftTrend, PageRightTrend, ScrollUpTrend, ScrollDownTrend, PageUpTrend, PageDownTrend, PageDurationTrend, ZoomInTrend, ZoomOutTrend, ZoomResetTrend, ZoomInAxisTrend, ZoomOutAxisTrend, ZoomResetXAxisTrend, PauseTrend, ResumeTrend, ShowTrendCursor, ScrollTrendCursor, ScrollTrendToTime) • Alarm (ResetAlarm, AckAlarm, SelectAllAlarms, EnableAlarms) • System (DumpTrend, DeleteTrend, DeleteEventArchive) • Recipes (DownLoadRecipe, UpLoadRecipe, WriteCurrentRecipeSet, DownLoadCurRecipe, UpLoadCurRecipe, ResetRecipe, DumpRecipeData, RestoreRecipeData, AddRecipeDataSet, DelRecipeDataSet)	• Page actions are not supported in alarm trigger condition
XForms	• Scaling • Offset • ColorPalette • BitIndex	• Some parameters do not support the ColorPalette functionality.
Keypads	• Only numeric keypads widgets are supported.	• Custom keypads are not supported. The numeric keyboard will be displayed as numeric widgets with a read/write or write mode.
Dialog Page	• Supported, you can show them and close them based on the ShowDialog and CloseDialog actions.	• Dialog pages support only modal dialogs.
User Management	• The login mechanism verifies user credentials on the server. The user name and password are based on the user credentials defined in User Management.	• Individual security settings applied to widgets or pages are not supported.
Concurrent User Connections	• The web server in the HMI device supports three concurrent connections at a time.	• If more than 3 connections are attempted from remote browsers, only the first 3 connections will be permitted.

Working with keypads in SH4Web

The user can click on the Numeric widget and a text box will be displayed in which the new value can be inserted.

After inserting the value the user can either press **Enter**, or equivalent in touch devices, or click **Save** to make the newly inserted value permanent. Only meaningful numbers will be accepted during the save process. Anything else will be ignored and will not result in a value change.



Troubleshooting and FAQ

Enable JavaScript

SH4Web requires JavaScript to provide interactivity with the server and the user. SH4Web will not work if JavaScript is disabled in your browser.

By default most browsers come with JavaScript enabled. But if you have disabled JavaScript in the past, please re-enable JavaScript before accessing SH4Web pages.

Browser cache

SH4Web includes resources that change infrequently such as CSS files, image files and JavaScript files. These resources take time to download over the network which increases the time required to load the SH4Web page in your browser. Browser caching allows these resources to be saved by a browser and used without requesting them each time from the server. This results in faster loading of SH4Web pages.

Caching is normally enabled by default, for optimal SH4Web performance make sure it has not been disabled.

Note: SH4Web pages will still work properly with disabled browser caching, however resource loading time will be slower compared with normal cached operations.

Using a proxy

Some users may be accessing the SH4Web project through a proxy. The proxies may control the number of parallel connection for the browser.

Make sure that the maximum parallel connections allowed (max connections) is not more than 10 and not less than 5.

Why I'm not able to see changes in the web pages?

Every time a new web page is added edited into the project, you need to download the project to the device. However, when you connect the device IP address, the web browser might display cached pages instead of the latest downloaded pages. To avoid this behavior you can:

- disable cache of your web browser
- force web page refresh
- by-pass browser cache

40 License activation of HMI device software modules

You perform license related tasks for the HMI device in the Manage Target dialog of Wizard.

Activate the device

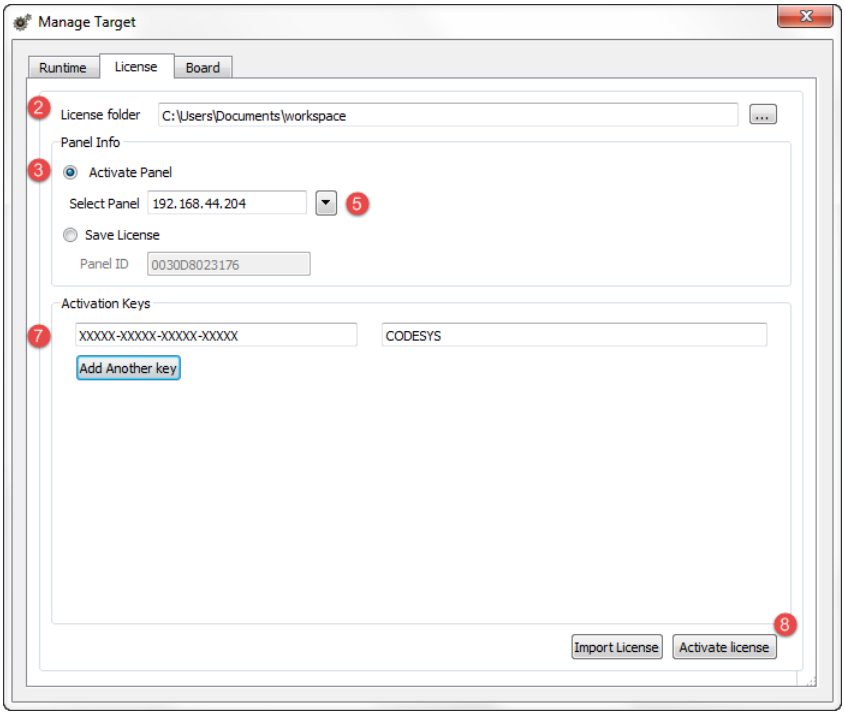
On each HMI device you need to activate the corresponding license. To do this you need an Internet connection.

Important: HMI Runtime must be installed in the device before you can activate your license.

Path: ***Run*** *>* ***Manage Target***

1. In the **Manage Target** dialog, click the **License** tab.
2. In **License folder** select the location of your backup license files.
3. Choose the **Activate Panel** option.
4. Select the panel.
5. Click **Select IP** button: all of the devices connected to the network are listed.
6. Select the device on which the license is to be activated.
7. Enter **Activation Key**.

   Once you have entered an activation key, the **Add Another Key** button is enabled, and you can add another activation key.
8. Click **Activate license**: if the activation key is valid, the license file is downloaded to the HMI device and the license is activated.



The enabled features or status of the activation keys is displayed in the Feature/status list box.

Note: You need to restart the HMI device to enable the activated license.

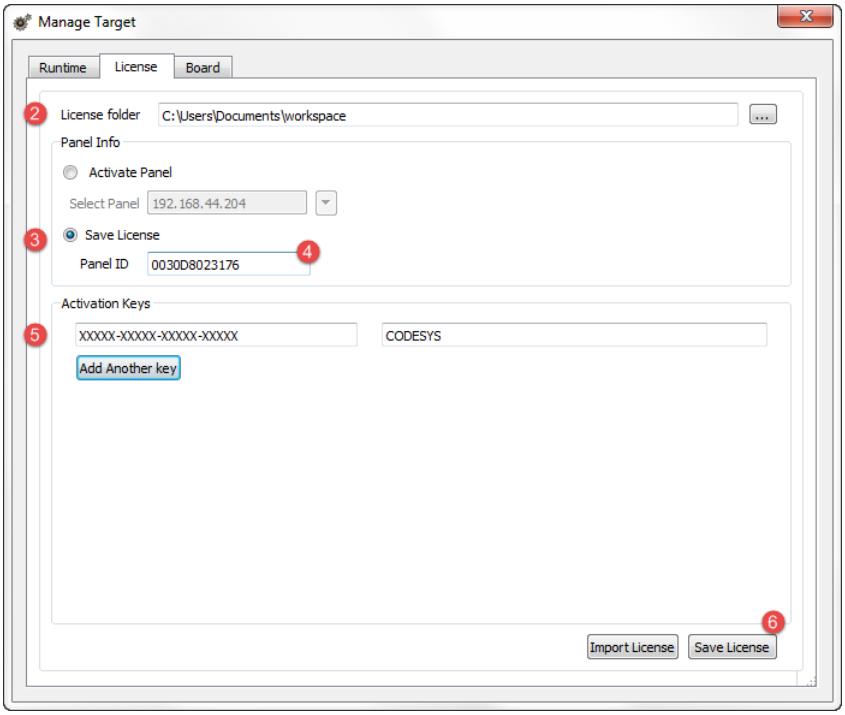
Save a license

You may need to create a license file for later activation oh the HMI device.

*Path: **Run**> **Manage Target***

1. In the **Manage Target** dialog, click the **License** tab.
2. In **License folder** select the location of your backup license file.
3. Choose the **Save License** option.
4. Enter **Panel ID**.
5. Enter **Activation Key**.

   Once you have entered an activation key, the **Add Another Key** button is enabled, and you can add another activation key.
6. Click **Save License**: if the activation key is valid, the license file is stored in the selected folder.



The enabled features or status of the activation keys is displayed in the Feature/status list box.

Import a license

If no working Internet connection is available on the computer running Wizard, you can neither save or activate a license. You must therefore import your license from another computer where you where able to download it.

This is a two-step procedure:

1. First you download and activate your license on a computer connected to the internet
2. Then you import the downloaded license to the HMI device.

Downloading and activating the license

1. On a computer with Internet access, open a browser and go to https://license.x-formation.com
2. Enter your activation key (for example, 5BDI0-FORLS-HR67G-5BI5T).
3. Enter your HMI MAC ID in the field **HostID Value** as **Custom=<MAC_ID>** (for example, Custom=0030D801DE27).
4. Click **Activate**: the license file is generated.
5. Click **Download**: the license file is saved to your computer.

Importing and activating the license

Path: ***Run*** *>* ***Manage Target***

1. In the **Manage Target** dialog, click the **License** tab.
2. Select the device if it is connected or enter the MAC ID if you are generating the license offline.
3. Click **Import license** and select the license file you created. Add more license files if more than one activation key was purchased.
4. Click **Activate license** to activate the licenses on the HMI device or **Save License** for offline license file generation.

Note: You need to restart the HMI device to enable the activated license.

41 Updating system components in HMI devices

Most of the system software components can be easily upgraded ensuring a high degree of flexibility in providing updates and fixes to existing and running systems.

New software modules can be updated

- Directly on HMI device using an USB flash drives (see "System Settings" on page 373 for details)
- From Wizard application (see "Update of system components from the application" on the next page for details)

Each HMI device is labeled with a product code including all factory settings (hardware, software and firmware components). Refer to this label for information on your HMI device. The HMI device update tool also provides detail on the components actually running on the device.

CAUTION: Make sure you use the correct upgrade files, since loading upgrade files unsuitable for your device will cause serious system malfunction. Always check your device product code.

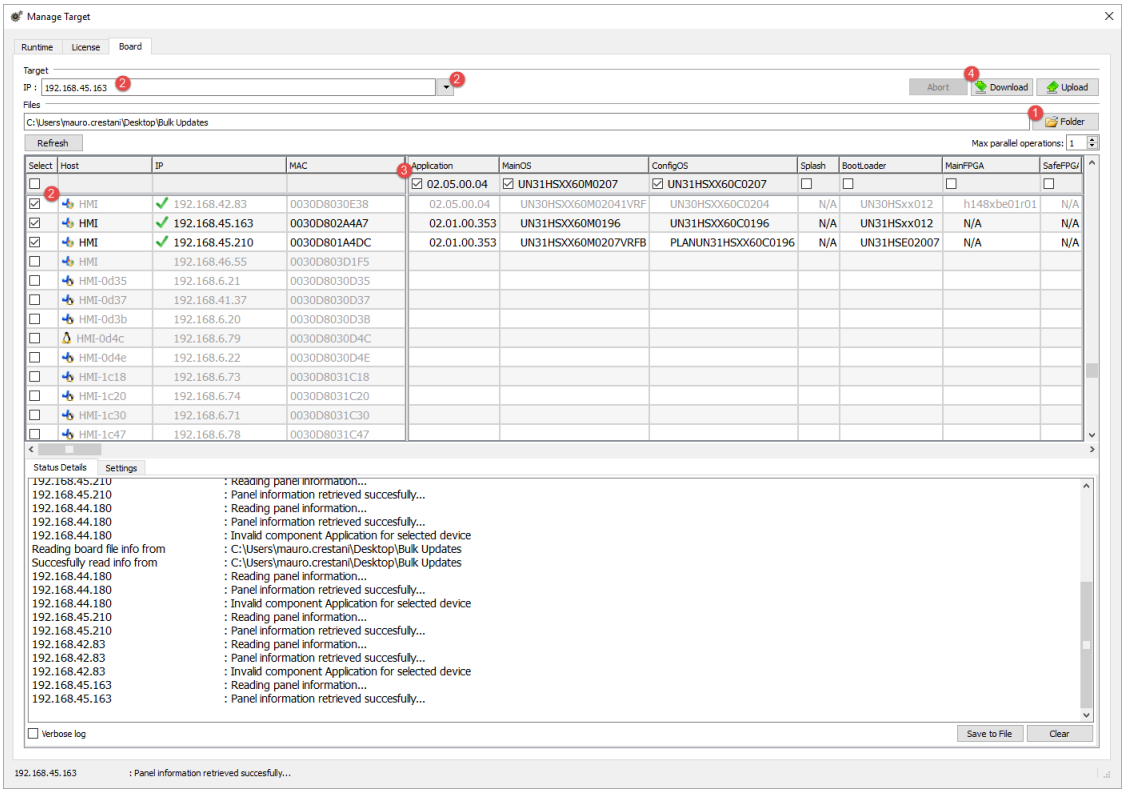
Note: Upgrade files are distributed upon request as a part of technical support activity.

Service call: Downgrade operations are complex tasks which might cause serious damage to your equipment if not performed correctly. These operations are reserved to technical support.

Update of system components from the application

You can download system components to a single HMI device or to a bulk of HMI devices of the same type using the Ethernet communication interface.

Path: ***Run> Manage Target> Board***



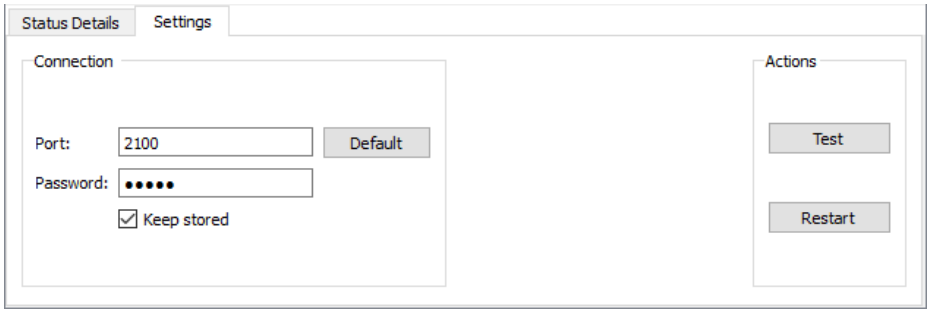
1. Select the folder that contains the files to download to the HMI device or where to upload files from the HMI device
2. Select one or more HMI device.
3. Select the components that you will download (or upload) to/from the devices
4. Start the Download to HMI or the Upload from the HMI operation

Note:

- The tool is designed to update multiple HMI devices of the same type. Please avoid putting files for different device type into the same folder
- If the desired target IP is not listed, type it directly into the box. The discovery service is a broadcast service. When a remote connection is done via VPN or from external networks, it will not work and you will have to enter the address manually.
- Download of the selected components will be performed only to the compatible devices
- Based on your network and hardware capabilities you can increase the number of devices to update in parallel
- You need to restart the HMI device to finalize the update.

Settings

From the **Settings** tab you can specify the Port and the Password parameters to use to communicate with the HMI devices. Leave Password empty if no password is set on the HMI device side.



WARNING: Bulk mode is working only with the HMI devices that have the same connection parameters

Uploading a splash screen picture

You can replace the default splash screen image shown by the devices during the power up phase.

The image used as splash screen must comply with the following requirements:

Filename	splash.bmp
Format	Bitmap, RGB 565 format
Size	< 500 KB
Bitmap width	Even number (for example 430x239)

To upload the splash screen image:

1. Rename the new image splash.bmp and copy it in the source folder.
2. Select HMI devices
3. Click **Download**.

To ensure the best visual results, splash screen images must have a black background.

42 Protecting access to HMI devices

The following operations are password protected on the HMI device:

- HMI Runtime management: install HMI Runtime and update HMI Runtime
- Board management: replace main BSP components such as Main OS, Configuration OS, Bootloader, and so on
- Download and upload of project files
- Optional services on Linux devices (e.g. SSH Protocol, VNC Server)

WARNING: Unauthorized access to the device can cause damage or malfunctions. When connecting the device to a network protect the network against unauthorized access.

Measures for protecting the network include:

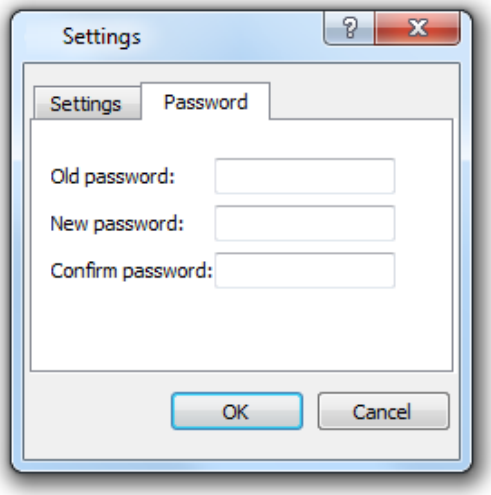
- Firewall
- Intrusion Prevention System (IPS)
- Network segmentation
- Virtual LAN (VLAN)
- Virtual Private Network (VPN)
- Security at physical access level (Port Security).

Further information, guidelines and standards regarding security in information technology: IEC 62443, ISO/IEC 27001.

Changing password on HMI device

To change the password on the HMI device, use one of the following methods:

- From the HMI Runtime context menu: **Settings**> **Password** tab.



- Use the **Set Target Password** function in update package: the password is updated by HMI Runtime just after the update process is completed.
- Using HMI device "System Settings" on page 373 Tool

Leave “Old password” empty as default if target password is not set.

For Win32 HMI Runtime, password is saved into *Users\[username]\AppData\Roaming\CGC S.p.A.\buildNumber\server\config\RemoteUpdateConfig.xml*.

Ports and firewalls

Here a list of all the ports used by Wizard components.

Port	Usage	Remote Access	Board Management	Runtime/Project Management
80/tcp	HTTP port	Yes	-	Yes
21/tcp	FTP cmd port	-	-	Yes
2100/tcp	Board port	-	Yes	-
16384-17407/tcp	FTP data port (passive mode)	-	Yes	Yes
990/udp	UDP broadcast (Device discovery)	-	Optional	Optional

Port	Usage	Remote Access	Board Management	Runtime/Project Management
991/udp	UDP broadcast (Device discovery)	-	Optional	Optional
998/udp	UDP broadcast (Device discovery)	-	Optional	Optional
999/udp	UDP broadcast (Device discovery)	-	Optional	Optional
5900/tcp	VNC Server	VNC only	-	-
5100/tcp	JS Remote Debugger	-	-	Optional

Remote access

Remote access is required to connect to HMI Runtime using:

- HMI Client
- Web access SH4Web

Runtime and project management ports

You use these ports to connect to HMI Runtime for operations such as update, installation and project download.

Board management ports

You use these ports to connect to the HMI device for Board operations such as BSP update, splash image download and so on.

Note: When broadcast service is not available, for example in VPN networks, type in the exact IP address to connect to the HMI device from Wizard.

43 Tips and tricks to improve performance

Wizard allows great flexibility for a project designers.

Follow these guidelines to create projects that perform better in terms of boot time, page change and animations.

Static Optimization

Static optimization is a technique used in Wizard to improve run-time performance.

Using a lot of images and pictures in a project might degrade performances, static optimization merges several images into a single background image thus reducing rendering and loading times. Using this method only one raster image needs to be loaded and rendered instead of many single raster and/or vector images.

When you create a project in Wizard, the pages might contain widgets such as texts, images, background images, background colors and so on which can be classified as:

- Static: values or properties do not change at run time.
- Dynamic: values or properties change at run time.

Note: Based on security settings, static parts of widgets could be not merged to background. This happens when a widget is configured as “hide” in security settings.

Important: When you change the properties of widgets with JavaScript set the widget Static Optimization to Dynamic, otherwise changes to properties will be ignored.

When downloading or validating a project, Wizard identifies static components and renders them as background images to .png files. These background images are saved as a part of the project under the folder */opt*.

Background images can be created as follows:

- full page background images, containing all widgets merged to page background
- group background images, containing a group of static widgets merged together to form a group background. For example, the Gauge group is normally composed by a background, a scale, a label and a needle, where background scale and label can all be merged to a single background image.

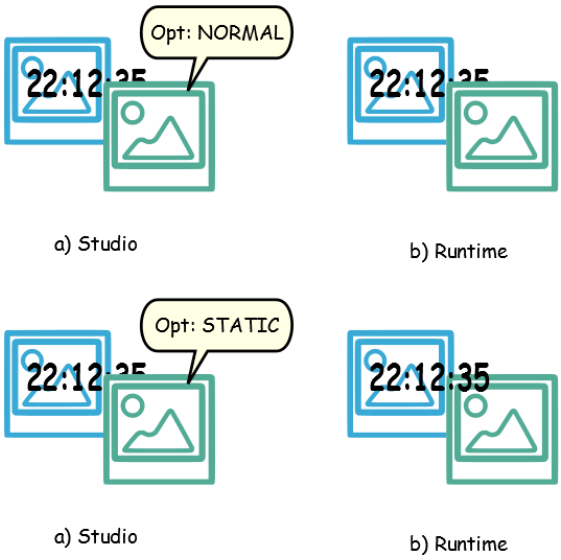
The **Static Optimization** page attribute enables and disables static optimization of the whole page. If it is set to **false** the optimization is totally disabled.

Finer control can be achieved setting the **Static Optimization** attribute of each single widget as follows:

- **Normal**: Wizard automatically detects if the widget can be merged with the background. This can be used if the widget is not a dynamic widget and does not overlap, that is it is not stacked above, a dynamic widget.
- **Static**: The image is forced to be merged with the background. This can be used when the static widget overlaps a dynamic transparent widget.

Note: In this case the automatic optimization will fail because it does not make any assumption on invisible areas which might be rendered at run time.

- **Dynamic**: The widget is not optimized at all. Use this flag when a static widget needs to be changed by Javascript.



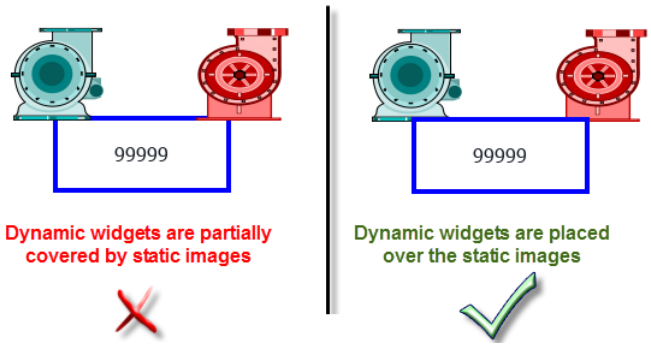
Tips for best performance

1. First of all: avoid placing static widgets over a dynamic widget. The overlapping area is computed considering the bounding rectangles of the widgets, that is the rectangles delimited by editing handles.
2. Don’t use static optimization if your pages contain almost only dynamic objects. Static optimization would save many almost identical full size images for each page using up a lot of memory space that could be more effectively used to improve project performance with other techniques (such as, for example, page caching).
3. Bounding rectangles can include transparent areas, minimize transparent areas (for example splitting the image in multiple images) since they can be a waste of resources even when optimized.
4. Optimize image size. The image will be rendered at the size of the image widget containing the image. For best performances the widget needs to be the same size of the image.
5. Avoid using **Scale to fit** for image widgets, since this forces a rescaling at run time for dynamic images and “hides” the actual image size during editing.
6. Use **Size to fit** to make the widget to the real size of his contents.
7. If overlapping cannot be avoided make sure to place the static widgets in the back, that is behind the dynamic widget.
8. Choose the image file format based on the HMI device you are connecting to.

9. Avoid using too many widgets in a single page. Often widgets are placed outside the visible area or their transparency is controlled by a tag. Since widgets are loaded even if they are not visible, having too many widgets in a page can significantly slow down the page change time.
10. Split a page with many widgets into multiple pages with less widgets.
11. For popping up new graphic elements in a page, prefer dialog pages with controlled positioning to transparent widgets.
12. Check the *opt* folder to see if static optimization is working as expected, the widgets z-order might need to be adjusted.
13. Numeric fields are often used to run JavaScript code on OnDataUpdate event even if the widget doesn't need to be visible on the page. In this case place the widget outside the page visible area instead of making it invisible, altering font color or visibility property. In the latter case you might end up with many left over wedges.
14. Use a HotSpot button if you need a touch area to react to user inputs.
15. If you reuse a widget from the gallery or you create your own, remember to set the correct optimization properties. For example button widgets are dynamic widgets, if you use a button widget just for its frame it won't be optimized since the button widget is dynamic. If you just need the frame you should use the Up image.
16. With many pages having many dynamic widgets and using a common template:
    1. set template static optimization to **true,**
    2. set page static optimization to **false**, since the background is already provided by the template.

    In this scenario the background image can be reused by many different pages thus saving memory space.
17. Do not use dynamic widgets, such as buttons, only for graphic purposes, when the button function is not needed, use image widgets instead to obtain the same graphical effect.

Here is an example of a correct and an incorrect use of static optimization.



Supported image formats

Wizard supports several raster formats like BMP, PNG, JPEG, TIFF and the vector format SVG. Here a list of pros and cons:

Image format	Pros	Cons
RASTER	• Fast rendering • Well standardized	• Big file size • Fixed resolution
VECTOR (SVG)	• Small file size • Rescale without quality loss • Can handle dynamic properties	• Complex SVG images with many graphic items and layers can be slow to render. • Creating an optimized SVG is not simple. • Only Tiny 1.2 (http://www.w3.org/TR/SVGTiny12/) supported.

Note: Scour software is free tool that can be used to remove foreign code from file (http://www.codedread.com/scour/).

Static optimization of templates

Template pages can have large amounts of static content. However, static optimization cannot be applied to a template page, since where the template is used is based on the page design.

If a huge background image should be repeated in every page that uses the same template, this would increase the footprint of the device as the same static image would be created for each of the pages using the template page.

FAQ on Static Optimization

Q: In a page where there are a few identical widgets, in the *opt* folder I see a PNG for each one of them. If they are really identical, why should the software duplicate them instead of having just one PNG?

A: The software does not know if static images are actually the same since each widget could have different settings/properties altering the actual rendering at run time.

Q: Why are the static images stored in a separate folder called *opt* instead of storing them directly in the project folder?

A: This avoids name collisions and allows skipping the upload of optimization images

Q: Why are the static images stored as a PNG files instead of common JPEG files?

A: PNG format uses a lossless compression for images and supports transparencies. JPEG files would render fuzzier compared to the PNG files with a different result in Wizard(not using optimization) and HMI Runtime.

Q: What will happen when no optimization is done in the software?

A: Every single widget is rendered at run time. In particular SVG images may require a lot of time to render in an embedded platform.

Page caching

Once accessed all pages are kept in a RAM cache up to the maximum allowed cache size depending on the actual platform's available RAM. This allows a much faster access since cached pages, once reloaded, only need to re-paint their content without reloading all page resources.

Image DB

Image DB is a technique used to track the usage of image files and reduce the cost of image loading by caching most frequently used images (example, Push Button images, Gauge needles, Slider thumbs and so on). The same image used in many different places is therefore loaded just once.

The image DB function will preload the top most used images at startup until memory limit is reached. This would further improve the individual page loading times.

The file imagecachelist.xml is created in *project/opt* folder, containing relevant information:

- Fill color (in case of SVG images)
- Size of SVG image
- Number of times an image is used in the project
- Number of different sizes for the same image

Tips for using the Image DB function

1. Use uniform size of buttons, gauges and other widgets wherever possible.
2. Use same color themes among widgets of the same kind.

Precaching

The Precache attribute of pages can be used to notify HMI Runtime to preload some pages in RAM at boot time for quicker access. Precaching is useful for complex pages having many dynamic widgets.

When this function is enabled on a page, access to the page is faster, however it also slows down boot-time since the system is not ready until all pages to be precached are not saved into the RAM.

Tips to precaching

1. Enable the precache function just for few pages having many dynamic widgets or for pages frequently used by users.
2. Do not enable the precache function for all the pages in the project since you would hìruno out of memory and have no benefit at all.
3. Disable static optimization for pages where the precache function is enabled to reduce memory used.

FAQ on precaching

Page limit for precaching

Based on the size and complexity of a page, the space required for precaching can be from 1,5Mb to 3Mb.

When a project is loaded, HMI Runtime proceeds as follows:

1. Page images are preloaded until 76 MB of memory space is still available (imageDBLowMem)
2. Pages where precache is set to **true** are preloaded untill 64 MB of memory space is still available (pageCacheLowMemMax). The images of these pages are loaded in the RAM (into the Image DB).

When the project is ready:

1. Any new page visited is saved in the cache (RAM) with all related images until 40 MB of memory space is still available(pageCacheLowMemMin)
2. When a page change happens and space in RAM is critical (<40MB), the HMI Runtime starts emptying the cache (RAM) removing pages and related images until 64 MB of memory space is made available. HMI Runtime removes data stored in the cache in the following order:
    1. last visited pages and bigger and unused images (>320x240),
    2. if more memory is needed also the pages in precache and all images loaded in Image DB can be removed.

44 Functional specifications and compatibility

Here is an overview of the supported functions and related limitations. Limitations indicated here represent a safe limitation, beyond that proper operation and state-of-the-art performance of the system is not guaranteed.

Table of functions and limits

Function	Max limit
Number of pages	1.000
Number of basic widgets	2.000 x page
Number of tags	10.000
Number of dialog pages	50
Number of dialog pages that can be open at the same time	5
Number of Recipes	32
Number of parameter sets for a recipe	1.000
Number of elements per Recipe	1.000
Number of user groups	50
Number of users	50
Number of concurrent remote clients	4
Number of schedulers	30
Number of alarms	2.000 (See "HMI devices capabilities" on the facing page)
Number of data transfers	1000
Number of templates pages	50
Number of actions programmable per button state	32
Number of Trend Buffers	30
Number of curves per trend widget	5
Number of curves per scatter diagram widget	10
Number of samples per trend buffer	200.000
Number of tags per trend buffer	200
Number of trend buffer samples for a project	1.200.000 (See "HMI devices capabilities" on the facing page)
Number of messages in a message field	1024
Number of languages	12
Number of events per buffer	2.048
Number of event buffers	4

Function	Max limit
JavaScript file size per page	16 KB
Size of project on disk	60 MB (See "HMI devices capabilities" below)
Number of indexed instances	100
Number of indexed alias	100
Number of indexed tag sets	30
Number of physical protocols	4
Number of reports	32
Number of reports pages	32
Max number of variables in variables widget	255
User folder size (UpdatePackage.zip)	5 MB
FTP additional folders	5

HMI devices capabilities

See "Table of functions and limits" on the previous page for the standard capabilities.

HMI Devices (Windows CE)	Limits
BTM-T4-24, BTM-T7-24	Max_Alarm = 500 Max_ProjectSize = 30 MB

PC (Windows)	Limits
BTM-PCRUNTIME	Max_Alarm = 10.000 Max_PageWidth = 10.000 px Max_PageHeight = 10.000 px

Features not available in Linux devices:

- LaunchBrowser macro
- Media Player widget
- Printer devices are not supported. Reports can be printed only on PDF files. Print of text reports and alarm events are not supported.

Features not available in BTM-PCRUNTIME:

- VNC and PDF Readers plug-in
- Manage Target
- System Settings Tool
- Backup/Restore
- Serial protocols that requires special hardware

Compatibility

The following compatibility policy has been adopted:

- Wizard version must always be aligned with HMI Runtime on the device,
- the user is responsible for updating HMI Runtime components on the HMI device at any Wizard update,
- the HMI Runtime update can be done directly from Wizard using the Update Target command available in the Run\Manage Target dialog,
- projects created in a Wizard version no older than V1.00 (00) can be opened and handled by any newer version,
- projects created with older versions of Wizard, opened with later versions and deployed to compatible HMI Runtime, are ensured to maintain the performance and functionality,
- compatibility between newer versions of HMI Runtime and projects created and deployed with older versions of Wizard is not ensured.

Important: Do not edit projects with a version of Wizard older than the one used to create them. It can result in a damage of the project and to HMI Runtime instability.

Converting projects between different HMI devices

Project conversion from different HMI device models is supported, however, some manual operations may be required if the project uses features not supported in the destination device.

Guideline

Before converting a project have a look if some unsupported features are present (see "HMI devices capabilities" on the previous page), and adjust your project by removing the unsupported features before converting the project.

In particular:

- Verify limitations and features not supported by the new HMI device (see "Table of functions and limits" on page 428 for details).
- Remove unsupported widgets, actions, system variables, protocols, project properties.
- If the project uses external storage, verify if the same storage path is still available.
- Adjust OS-specific external applications or paths.
- If necessary, reduce project size according to the new HMI device type limitations (see "Limitations" for details).
- Since HMI devices are based on different hardware platforms with different CPU speed, RAM memory size, cache size, make sure to check project boot time and page loading time for each page in the project.
- Verify JavaScript code for OS-specific operations.

OS-specific features

Linux is case sensitive while Windows CE is not. Consequently, projects on Linux HMI devices might have different files named based on upper and lower case, e.g. 'dump1.csv' and 'Dump1.csv' are not possible on Windows CE HMI devices.