TM611

Working with mapp View

PERFECTION IN AUTOMATION
www.br-automation.com

Prerequisites and requirements

Training modules	TM210 - Working with Automation Studio
Software	Automation Studio 4.2.5 Automation Runtime 4.25 Technology package – mapp View 1.0
Hardware	ARsim

Table of contents

1 Introduction

Smart devices such as tablets, smartphones, etc. are considered perfect examples of powerful technology with ultimate usability. Unsurprisingly, operators of industrial machines and systems – and therefore also manufacturers of such equipment – desire nothing less when interacting with the machinery they use every day.

With mapp View, B&R now offers access to web technology that can be used to develop HMI systems for B&R automation applications. Application engineers can use mapp View to create powerful and intuitive HMI solutions. The web technology used here is encapsulated with mapp View. Learning a broad field of technology is not necessary. HMI developers can focus entirely on creating a solution for the task at hand.

Figure 1: The mapp View visualization object

mapp View is fully integrated in B&R's Automation Studio engineering environment. mapp View is the first web-based HMI solution in the world that doesn't require knowledge of web programming languages.

1.1 Learning objectives

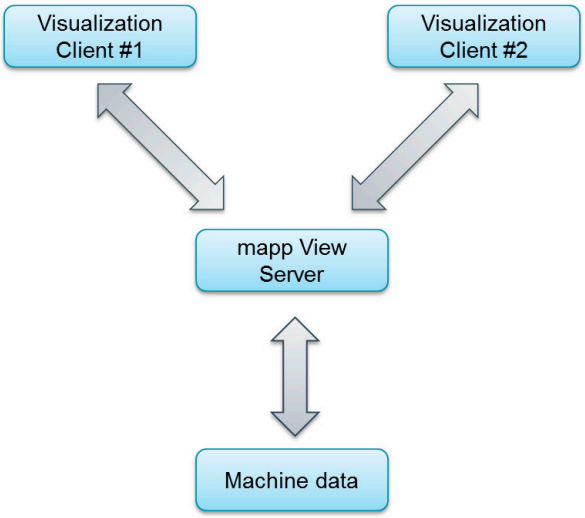
This training module uses selected exercise examples illustrating typical HMI tasks to help you learn how required functions are structured and used in Automation Studio.

- Participants will learn the basic concepts of mapp View and gain a better understanding of the characteristics of these concepts.
- Participants will be able to add a new mapp View visualization object to an Automation Studio project and configure it.
- Participants will be able to develop the content for a mapp View HMI application and use layout features to add the content to HMI pages.
- Participants will be able to use mapp View widgets to manage mapp View HMI application content.
- Participants will be familiar with the different ways to configure navigation between various pages of a mapp View HMI application.
- Participants will be able to include process data from the automation application in a mapp View HMI application.
- Participants will be able to implement images to organize the visual design of mapp View HMI application content.
- Participants will learn how the user role system is used to configure the properties of a mapp View HMI application depending on the logged in user.
- Participants will learn how the text system functions and will be able to configure language switching.
- Participants will become familiar with the unit system and learn how to configure unit conversion.
- Participants will learn how to configure triggering specific actions when events occur.

1.2 Symbols and safety notices

Unless otherwise specified, the descriptions of symbols and safety notices listed in "TM210 – Working with Automation Studio" apply.

2 mapp View concept



mapp View is an integrated system for HMI applications. This system, which is integrated in the existing automation software, is used to design HMI interfaces for machines.

mapp View has a modular structure. It allows monitoring and operation of technical processes on machinery and equipment. mapp View architecture is devised as a client-server system that consists of an HMI server connected to one or more HMI clients. The HMI server is decoupled from the automation application (in the machine or system logic).

Figure 2: mapp View architecture

The modularity of mapp View is also highlighted by the fact that the content and layout can be edited separately. This increases reusability of the HMI elements and reduces development time.

mapp View is devised as a multi-client and multi-user system. Various users can view customized content from different visualization clients. mapp View allows you to view role-based content without having to program this in the automation application. Customized HMI content can displayed simultaneously on different HMI clients independently of each other.

mapp View is based on web technology, but the developer of a mapp View HMI application does not have to deal directly with this technology. mapp View encapsulates the complexity of web technology and gives HMI developers the freedom to focus on designing the human-machine interface. HMI elements (widgets) are used via drag-and-drop to easily design HMI content.

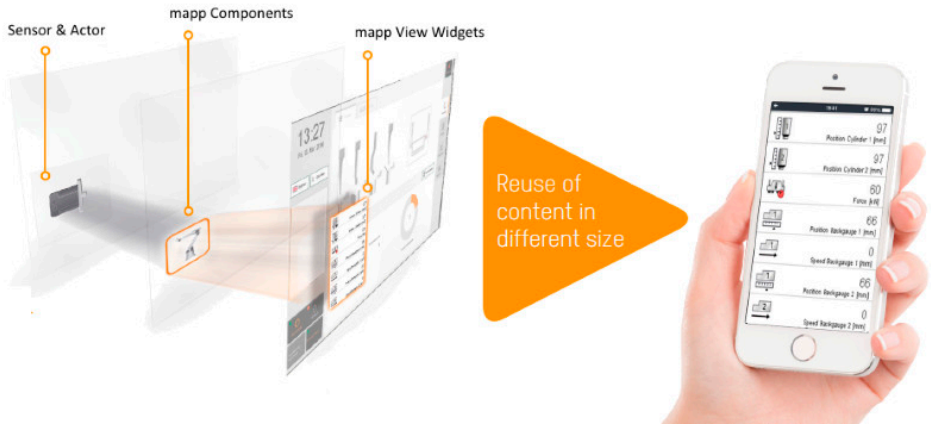


Figure 3: mapp View - Easy configuration through reusable elements

2.1 Installation: mapp View technology package

Automation Studio 4.2.5 is the minimum version required to install the "mapp View technology package".

The mapp View technology package can be downloaded from the B&R website and then installed using the upgrade dialog box.

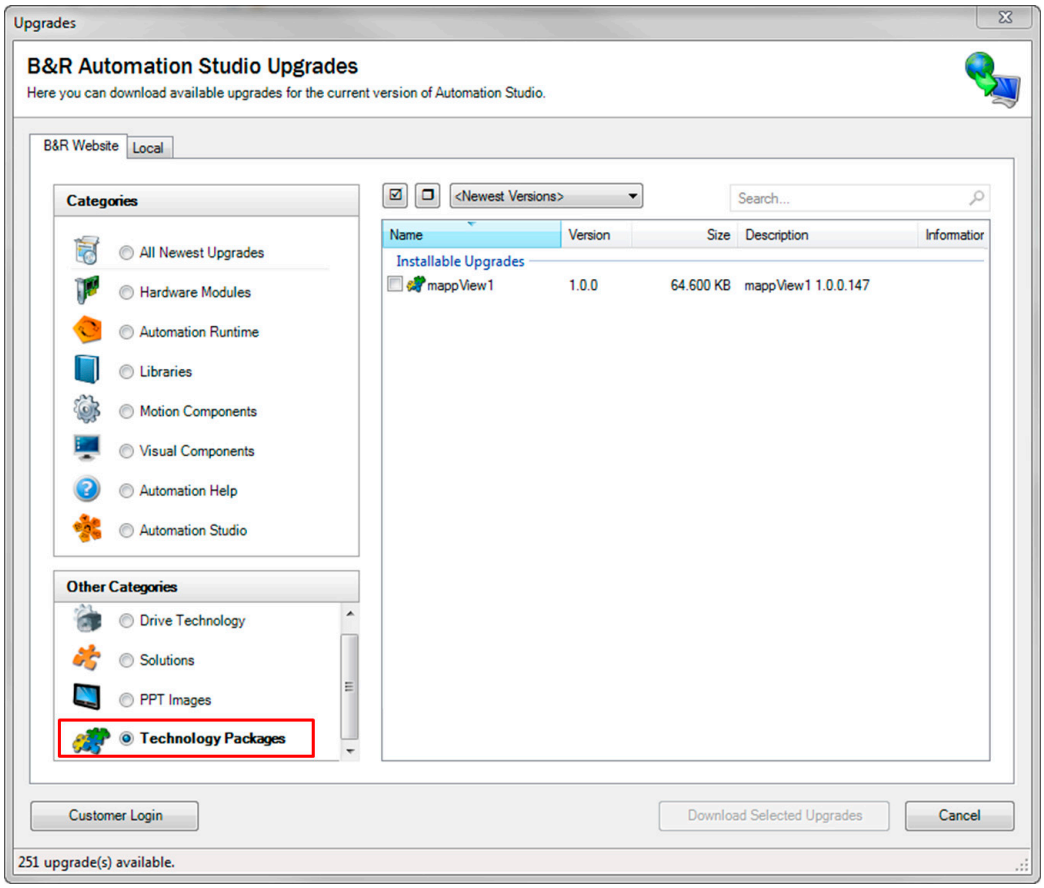


Figure 4: Automation Studio upgrade dialog box – Technology packages

During installation of the technology packages, Automation Help specific to mapp View is also installed.

2.2 Licensing

A mapp View HMI application can be used in ARsim without a license or restrictions.

To operate mapp View in ARemb and ARwin, a base license (model number 1TGMPVIEW.00-01) is required. This base license makes it possible to use one instance of mapp View and connect to a client. Additional client licenses (model number 1TGMPCLIENT.10-01) are necessary when connecting multiple clients to a mapp View server.

The number of necessary client licenses is determined by the maximum number of allowed clients in the mapp View configuration (not by the number of clients actually connected).

If a Technology Guard is not connected, then the HMI application is not shown on the client and a logger entry is created.

"No valid license for mapp View found. The mapp View server configuration allows more connections (10) than the number of available client licenses (4)!

2.3 Structure of an HMI application

Like the automation application, the elements of a mapp View HMI application are configured and managed in Automation Studio.

HMI pages, texts and image files are managed in the Logical View. In the Configuration View, one or more HMI applications are added to the active configuration from the elements in the Logical View. In this way, there is a clear separation between the source files for HMI elements and the necessary HMI for a machine configuration.

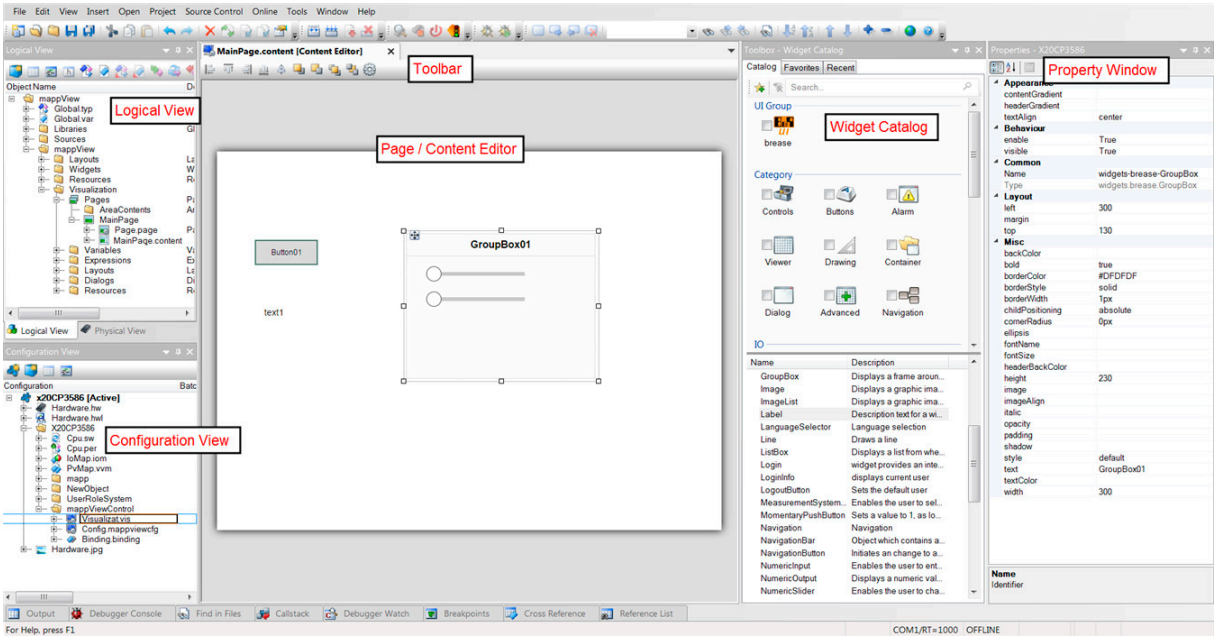


Figure 5: Organization of the HMI application in Automation Studio

Logical View

In the Logical View, elements of an HMI application (pages, pieces of content, texts, media files, etc.) are managed in a mapp View package.

Configuration View

The Configuration View is where one or more HMI applications are configured and managed.

Content Editor

The Content Editor is a visual editor used to create HMI content (pieces of content). HMI content can also be processed in a text editor (XML editor).

Toolbar

The toolbar in the visual editor provides tools for working with widgets during the design phase.

Widget Catalog

When a piece of content is opened (graphically or in XML form), widgets can be added from the catalog and configured.

Properties Window

Elements of an HMI application are configured in the properties window. Depending on the property (of a widget, for example), different dialog boxes are available for editing.

3 Page creation and navigation

A mapp View HMI application usually consists of several HMI pages. On the HMI client, the machine operator sees the content of one HMI page at a time. For the content of other pages to be seen, it is necessary to switch between the HMI application pages.

The HMI application navigation determines how the machine operator switches between HMI application pages.

This section describes how pages are created and the possible ways to navigate between the pages.

3.1 Overview – Creating a page

For design purposes, a mapp View HMI application page is divided into areas. The content of these areas can be designed independently of each other.

A layout defines how the areas that make up an HMI application page are structured. A layout is an independent construct and consists of the areas defined on an HMI application page.

To design an HMI application page, it is necessary to first define how the areas on the page should be structured. This is done by selecting a layout that includes the required number of areas in the required size. For a specific HMI application page, defined content can be assigned to each area.

Figure Structure of an HMI application page shows the elements used to design an HMI application page in the form of a practical example.

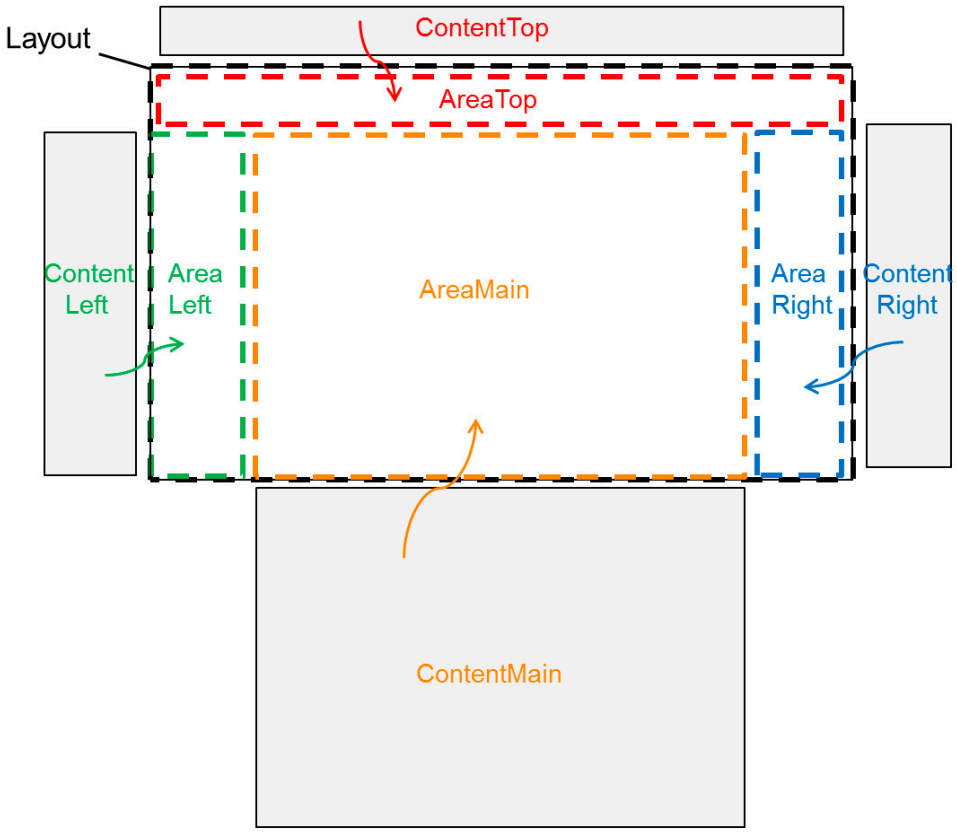


Figure 6: Structure of an HMI application page

3.1.1 Layout

A layout defines the space in which multiple areas are structured. The space structured by the layout is defined using the parameters "width" and "height". For clear identification, each layout has a unique "ID". Figure Layout overview"Layout overview" shows an overview of a layout.

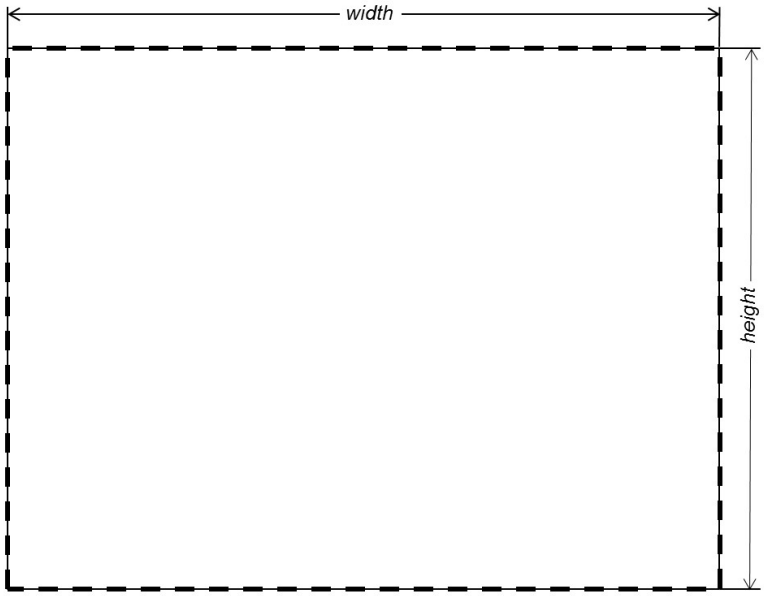
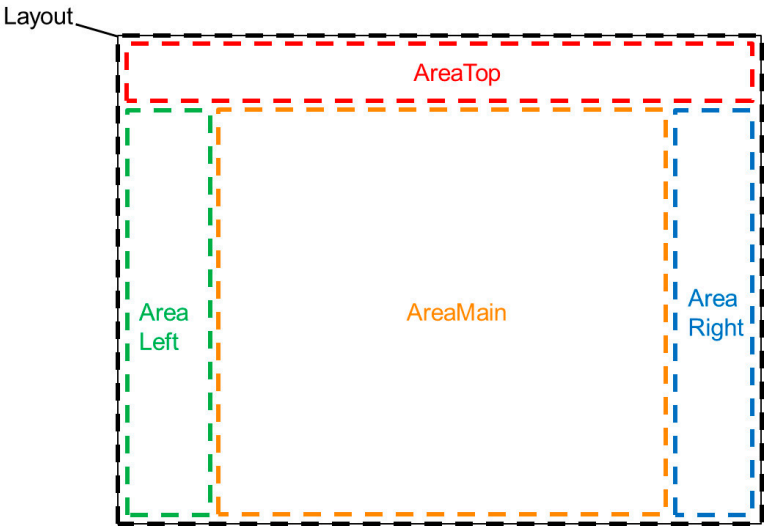


Figure 7: Layout overview

3.1.2 Area



Areas are defined portions of a layout.

Each area within a layout has a unique "ID". The "ID" of an area is unique within the layout in which that area is defined. An area is defined according to its size ("width" and "height"), which is specified in pixels and its position ("top" and "left") within the layout, which is also specified in pixels.

The reference point for an area's position within a layout is the upper left corner.

Figure 8: Layout with 4 areas

3.1.3 Content

Content refers to a piece of content that can be displayed in an HMI application. A piece of content is identified using a globally unique "ID" and its size ("width" and "height").

Widgets can be positioned within a piece of content.

Figure "Content with widgets" shows a piece of content with the following widgets:

- Button
- Label
- Numeric output
- Numeric input
- Image
- MeasurementSystemSelector
- LanguageSelector

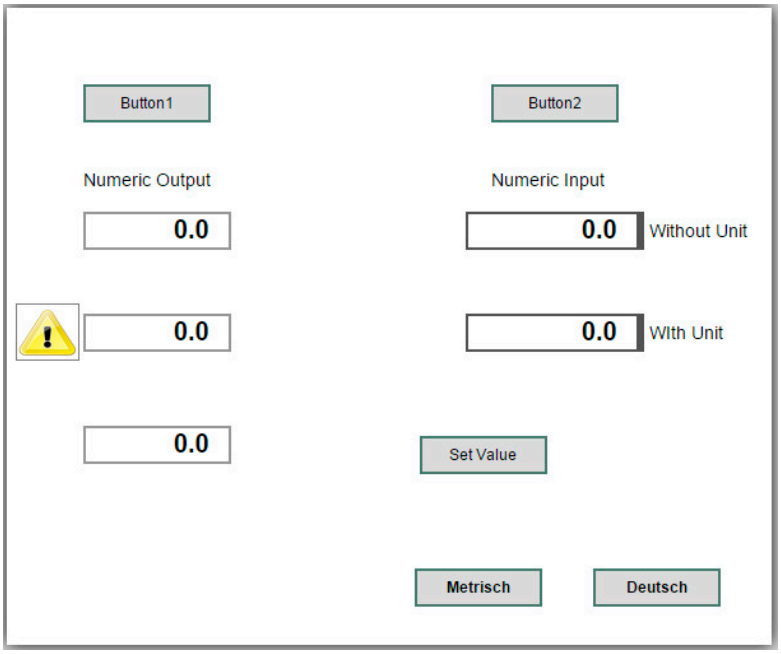


Figure 9: Content with widgets

3.1.4 Page

A page defines the content that can be displayed in the visible space of an HMI application client.

The page is identified using a globally unique "ID". The page defines which layout will be used to structure the areas and which content will be assigned to the individual areas.

Figure "Page with referenced areas, content and widgets" shows an example of a page. The four areas have different background colors to make them more distinctive.

Figure 10: Page with referenced areas, content and widgets

3.2 Step-by-step page creation

The objective of this first exercise is to create an HMI application with one HMI application page based on the Automation Studio project "mappViewGettingStarted". This project already includes the automation application and the required images.

The page will use a layout with four areas and each area will be assigned specific content. Different background colors will be used to better identify the four pieces of content. Figure MainPage with referenced layout, areas and content with referenced layout, areas and content" shows an overview of the structure of the page being designed and the "IDs" of the elements used (page, layout, areas, pieces of content).

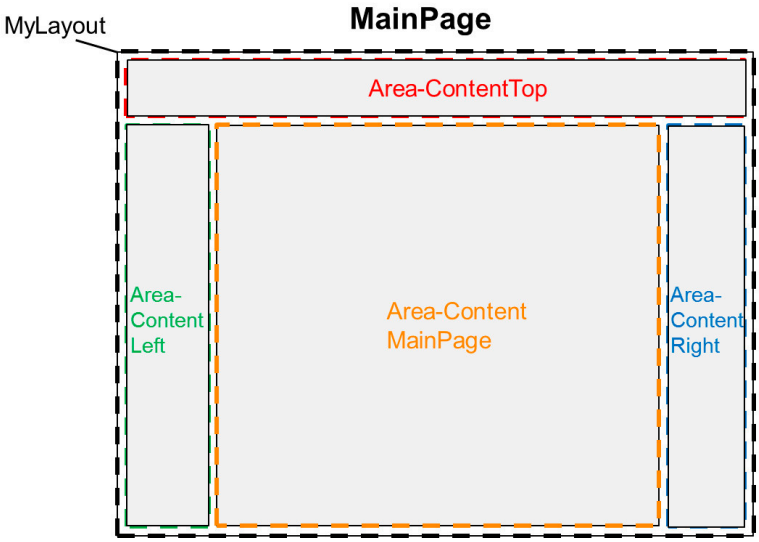


Figure 11: MainPage with referenced layout, areas and content

Exercise: Create your first HMI application page

In the following chapter, a mapp View HMI application with one page will be created.

The following steps must be carried out:

1) Open the mappViewGettingStarted project
2) Add packages in the Logical View
3) Create a layout
4) Create content
5) Create a page
6) Add files in the Configuration View
7) Configure the mapp View server (.mappviewcfg)
8) Customize the visualization object (.vis file)
9) Build and transfer the project
10) Display the HMI application in a browser

3.2.1 mappViewGettingStarted project

The mappViewGettingStarted project is available for the training as a .zip file and serves as the basis for creating the mapp View HMI application.

Exercise: Open the mappViewGettingStarted project

Open the mappViewGettingStarted project.

3.2.2 Add packages in the Logical View

Add packages

After the mappViewGettingStarted project has been opened, the mapp View package and the visualization package will be added from the Object Catalog in the Logical View. The visualization package will be added under the mapp View package.

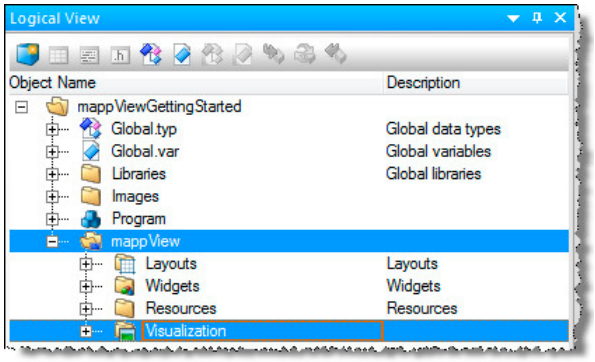


Figure 12: mapp View package and visualization package have been added

Exercise: Add packages

1) Add the mapp View package
2) Add the visualization package

HMI application/mapp View/HMI organization/LogicalView/mappView package

HMI application/mappView/HMI organization/LogicalView/mappView HMI package

3.2.3 Create a layout

Adding a layout

A layout is needed to divide everything up into areas. The layout is added to the Logical View.

If the Layout package is selected in the Logical View , a new layout can be added from the Object Catalog. The layout file added is then renamed to "MyLayout".

Double-clicking on the "MyLayout.layout" file opens the XML editor in the Automation Studio workspace.

The parameters from the table must be entered in the XML file:

Property name	Value
id	"MyLayout"
height	"600"
width	"800"

Table 1: Properties in the layout file

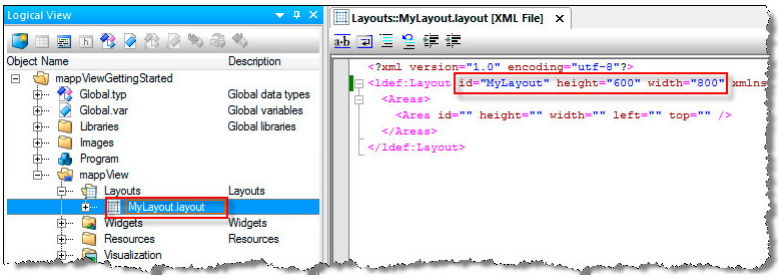


Figure 13: Add the MyLayout file in Logical View and edit the XML file

HMI application/mapp View/mapp View development environment/Layout and areas/Adding a layout

3.2.4 Creating areas

The page is divided into different regions by defining the areas in the layout.

Exercise: Define areas (XML)

Four areas will be created in "MyLayout.layout" with parameters that must be set to the values specified in the table.

ID	height	width	left	top
AreaMain	500	600	100	100
AreaTop	100	800	0	0
AreaLeft	500	100	0	100
AreaRight	500	100	700	100

Table 2: Properties for the areas used in the layout

Final result

The figure shown represents the definition for the four areas in "MyLayout".

```
<?xml version="1.0" encoding="utf-8"?>
<ldef:Layout id="MyLayout" height="600" width="800"
xmlns:ldef="http://www.br-automation.com/iat2015/layoutDefinition/v2">
  <Areas>
    <Area id="AreaMain" height="500" width="600" left="100" top="100" />
    <Area id="AreaTop" height="100" width="800" left="0" top="0" />
    <Area id="AreaLeft" height="500" width="100" left="0" top="100" />
```

```
    <Area id="AreaRight" height="500" width="100" left="700" top="100" />
  </Areas>
</ldef:Layout>
```

3.2.5 Create content

Adding content

Next, three pieces of content will be added to the "AreaContents" folder. The "AreaContents" folder is used to store the various piece of content, which can be used on multiple pages.

Select the "AreaContents" folder and add three content files.

After adding the content files in Logical View, they must be renames to "ContentTop", "ContentLeft" and "ContentRight".

Figure Content files added shows the three content files that have been added (with their distinctive file names).

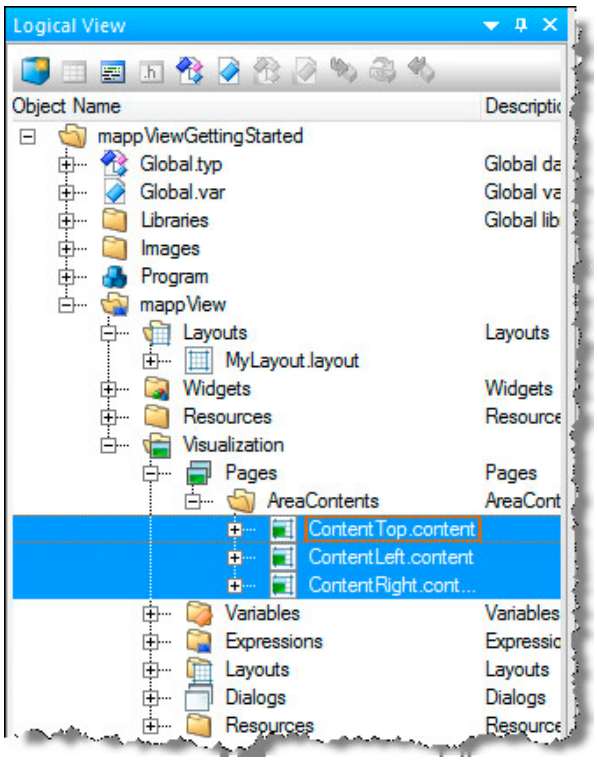


Figure 14: Content files added

Edit contents

A piece of content can be opened and edited with the visual editor or text editor.

The following section describes the process when using the visual editor.

The procedure for editing content using the XML editor can be found in the help system.

Visual editor

Double-clicking the content file opens the visual editor and the Properties window. The Properties window can be used to edit content attributes ("Name", "width" and "height").

In the Properties window, a unique "Name" will be entered as well as the "width" and "height" of the content.

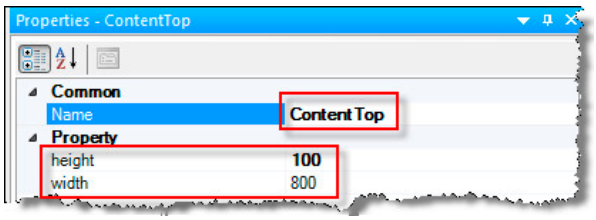


Figure 15: ContentTop Properties – Visual editor

We recommend selecting the name entered in the Logical View for the content file as the unique "Name".

Exercise: Edit all pieces of content

The parameter name, height and width for pieces of content must be defined as follows:

Name	height (px)	width (px)
ContentTop	100	800
ContentLeft	500	100
ContentRight	500	100

Table 3: Properties for Top, Left and Right content

HMI application/mapp View/mapp View development environment/HMI page content/Adding content

3.2.6 Create a page

A page object is added to an existing "Pages" package from the Object Catalog using drag-and-drop or by double-clicking on it. The "Pages" package is part of a mapp View visualization package.

The next step is to create a page with the name "MainPage" where the areas and the pieces of content can then be referenced.

Selecting the "Pages" package allows a page from the Object Catalog to be added to the Logical View and renamed.

In addition, a piece of content with the name "ContentMainPage" will be added under the MainPage package, opened by double-clicking on it and configured as follows:

Property name	Value
Name	"ContentMainPage"
height	"500"
width	"600"

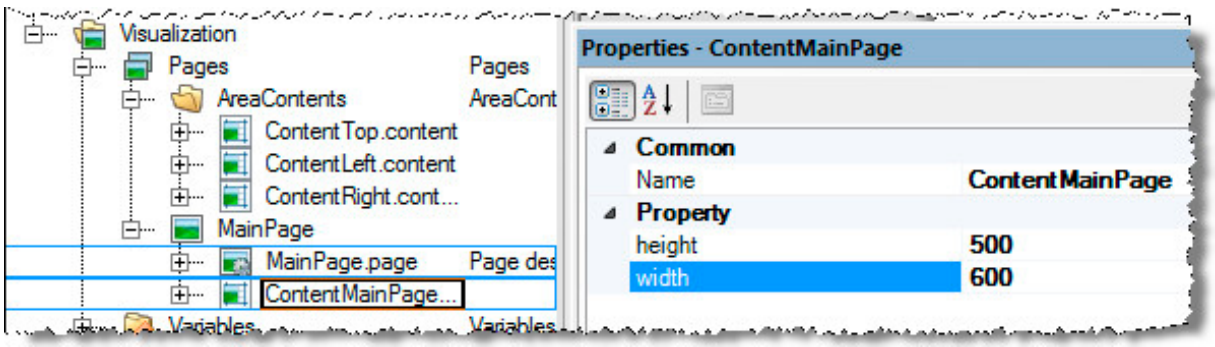


Figure 16: ContentMainPage in the Properties window

Double-clicking on the "MainPage.page" file opens the XML editor.

In order to uniquely identify the page, a globally unique "ID" must be specified.

To use the previously defined layout on the "MainPage", the "ID" of the layout ("MyLayout") must be entered.

In the <Assignments> element, the page is structured using the "type", "area" and "content".

The following areas and pieces of content are used to define the "MainPage":

type	baseContentRefId	areaRefId
Content	ContentMainPage	AreaMain
Content	ContentTop	AreaTop
Content	ContentLeft	AreaLeft
Content	ContentRight	AreaRight

Table 4: Main Page

```
<?xml version="1.0" encoding="utf-8"?>
<pdef:Page id="MainPage" layoutRefId="MyLayout"
xmlns:pdef="http://www.br-automation.com/iat2015/pageDefinition/v2">
  <Assignments>
    <Assignment type="Content" baseContentRefId="ContentMainPage"
        areaRefId="AreaMain" />
    <Assignment type="Content" baseContentRefId="ContentTop"
        areaRefId="AreaTop" />
    <Assignment type="Content" baseContentRefId="ContentLeft"
        areaRefId="AreaLeft" />
    <Assignment type="Content" baseContentRefId="ContentRight"
        areaRefId="AreaRight" />
  </Assignments>
</pdef:Page>
```

In order for the respective pieces of content to be clearly recognizable at runtime, a unique background color (backColor) will be defined for each piece of content. A color code is specified as a hexadecimal code for the "backColor" attribute:

```
<?xml version="1.0" encoding="utf-8"?>
<pdef:Page id="MainPage" layoutRefId="MyLayout"
xmlns:pdef="http://www.br-automation.com/iat2015/pageDefinition/v2">
  <Assignments>
    <Assignment type="Content" baseContentRefId="ContentMainPage"
        areaRefId="AreaMain" backColor="#2CB9E8"/>
    <Assignment type="Content" baseContentRefId="ContentTop"
        areaRefId="AreaTop" backColor="#2CE890"/>
    <Assignment type="Content" baseContentRefId="ContentLeft"
        areaRefId="AreaLeft" backColor="#2F3030"/>
    <Assignment type="Content" baseContentRefId="ContentRight"
        areaRefId="AreaRight" backColor="#DFE82C"/>
  </Assignments>
</pdef:Page>
```

HMI application/mappView/mapp View development environment/HMI pages/Adding HMI pages

3.2.7 Add files in the Configuration View

After "MainPage" has been created and configured, the mapp View server and the HMI application must be configured. This is done in the Configuration View under the mapp View node by adding the files "mapp View Configuration" ("Config.mappviewcfg") and "Visualization" ("Visualizat.vis") from the Object Catalog.

mapp View visualization object (.vis)

The mapp View visualization object (.vis) defines which visualization components in the Logical View and Configuration View are involved in displaying pages on the client. Each configuration can contain 1-n visualization objects (.vis).

Information about all definitions and parameters that have to be entered in the .vis file can be found in Automation Help.

> HMI application/mappView/HMI organization/Configuration View/mapp View HMI application

mapp View server configuration (.mappviewcfg)

This static configuration makes it possible to configure various settings for the mapp View server.

The exact settings for the server can be found in Automation Help.

> HMI application/mappView/HMI organization/Configuration View/mapp View HMI application

Exercise: Configure the server and add a visualization object

The following files must be added to the Configuration View:

1) mapp View configuration ("Config.mappviewcfg")
2) Visualization object ("Visualizat.vis")

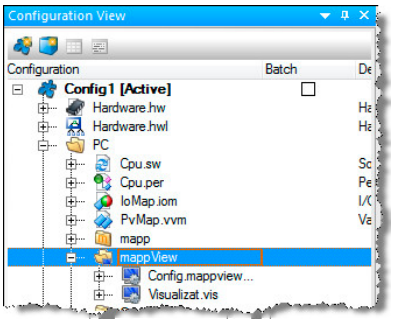


Figure 17: Configuration and visualization object added to Configuration View

Configuring Config.mappviewcfg

The default settings for file .mappviewcfg can be used without modifications.

Configuring Visualizat.vis

Element "Visualization ID" determines how the HMI application is launched from the client. Element <StartPage> defines which page is displayed when the HMI application is delivered to the client. The list of <Pages> elements contains all pages that are part of this HMI application.

The image shows the relevant content for practicing with this visualization object (.vis).

```
<?xml version="1.0" encoding="utf-8"?>
<vdef:Visualization id="Training"
xmlns:vdef="http://www.br-automation.com/iat2015/visualizationDefinition/v2">
  <StartPage pageRefId="MainPage" />
  <Pages>
    <Page refId="MainPage"/>
  </Pages>
```

3.2.8 Opening the HMI application in a browser

After the first page ("MainPage") has been created and referenced in the visualization object (.vis), the project can be compiled and transferred to ARsim.

The HMI application can be displayed using the following URL:

http://127.0.0.1:81/index.html?visuId=Training

Exercise: Compile the project and display it in a browser

1) Compile and transfer the project.
2) Open the browser and enter the URL

Expected result

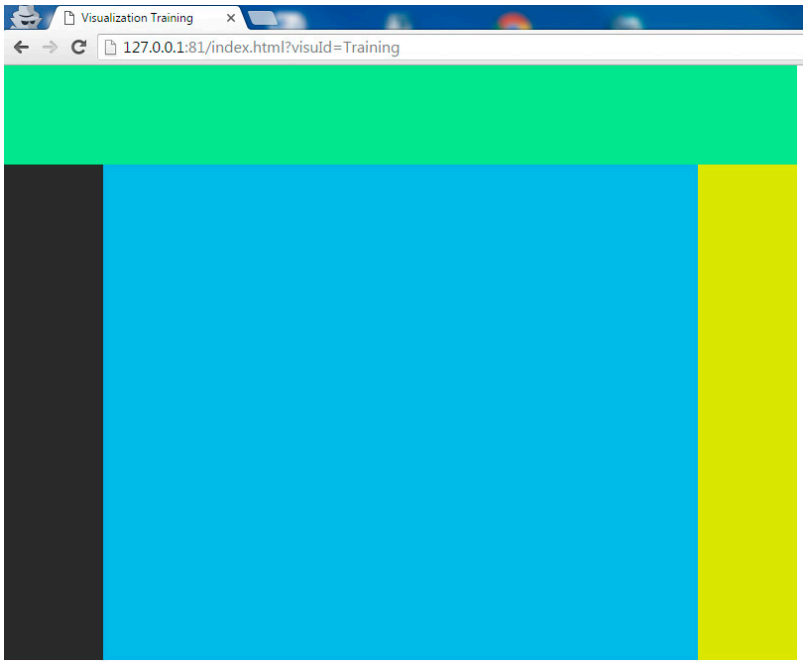


Figure 18: The HMI application containing a layout with 4 areas shown in browser

HMI application/mapp View/Accessing an HMI application from a client

3.3 Navigation overview

More often than not, an HMI application consists of more than one page. A system of navigation is needed in order to access these pages.

This system determines how these pages are navigated in the HMI application.

The example in figure Schema for manual navigation shows the three pages "MainPage", "ServicePage" and "AlarmPage" as well as the options for navigating between them.

In this example, it is possible to navigate to any page from any other page.

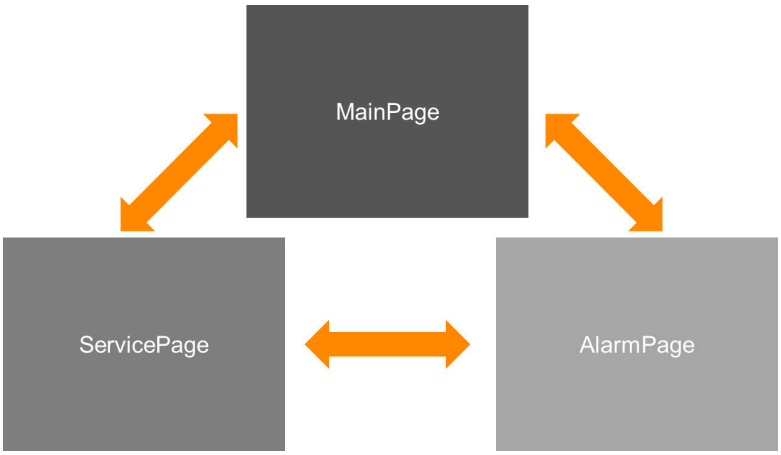


Figure 19: Schema for manual navigation

Two options are available for navigation in an HMI application, manual and automatic navigation. Both types of navigation can be combined.

3.3.1 Manual navigation

Manual navigation is implemented using an individual "NavigationButton" widget. In this case, a "NavigationButton" widget must be placed and configured for each page that can be navigated to from the current page.

Image Schema for manual navigation shows that three "NavigationButton" widgets that have been placed and configured on each page. Each individual "NavigationButton" widget is configured with the ID of the page that should be opened when the button is pressed.

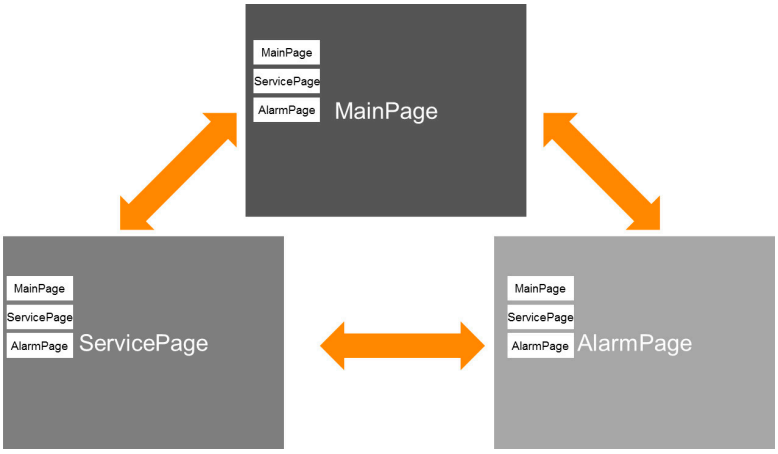


Figure 20: Schema for manual navigation

Manual navigation is well-suited to situations where the same "NavigationButton" widgets can be used for all pages. In this case, they have to be positioned in a piece of content and configured there. This navigation content can then be reused (referenced) on each page.

One drawback of manual navigation is that it is context-independent. The placement and configuration of the "NavigationButton" widget are the same on every page. If the placement and configuration of the "NavigationButton" widgets should vary depending on the page, you will have to create different pieces of content for navigation and reference them individually on the corresponding pages.

In HMI applications with a large number of pages, a great deal of work can be required here. Although there is a lot of work upfront developing the navigational structure of an extensive HMI application, the maintainability of manual navigation is very easy. The alternative to manual navigation in mapp View is automatic navigation.

3.3.2 Automatic navigation

Automation navigation relies on the definition of a navigation structure. This is used to define the different navigation paths, i.e. which pages can be navigated to from which other pages. mapp View takes care of placing the corresponding "NavigationButton" widgets on each page at runtime.

Each page only displays the target pages for which a navigation path from the current page has been defined. Displaying the "NavigationButton" widget is context-sensitive.

Image Automatic navigation paths shows automatic navigation for the three pages "MainPage", "ServicePage" and "AlarmPage". It is possible to navigate from any page to any other page, but each page shows only the context-sensitive navigation targets.

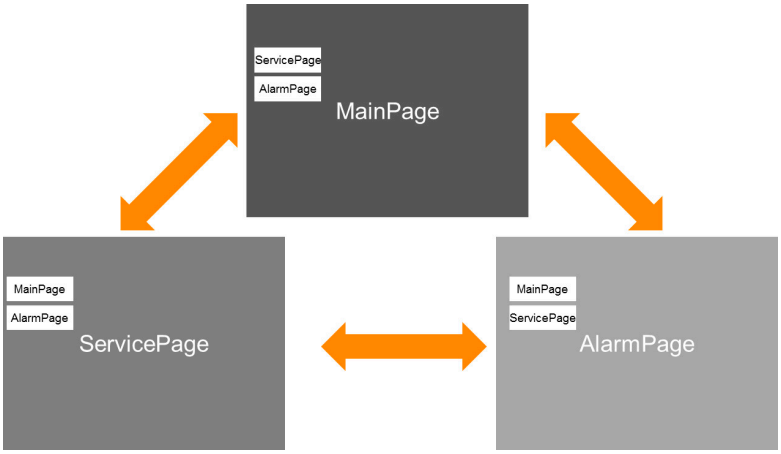


Figure 21: Automatic navigation paths

3.4 Using manual navigation

The goal of this exercise is to create a manual navigation system that can then be used to switch between different pages. To use manual navigation, 2 additional pages must be created ("ServicePage" and "AlarmPage"). The manual navigation system should be positioned and configured in content "ContentLeft" in the visual editor.

Exercise: Set up manual navigation

The following sections will be used to set up manual navigation.

The following steps must be carried out:

1) Create ServicePage and AlarmPage
2) Add and configure the NavigationBar widget
3) Add and configure the NavigationButton widget
4) Reference ServicePage and AlarmPage in the .vis file
5) Compile/Transfer the project and display it in a browser

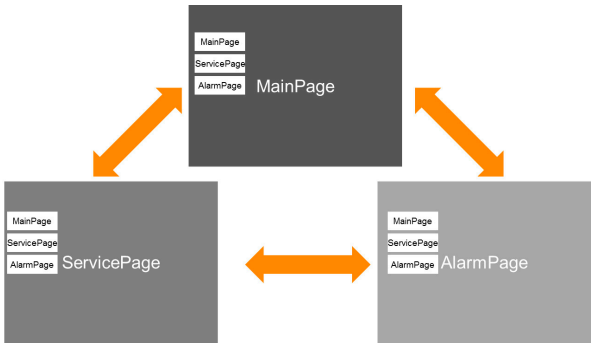


Figure 22: Manual navigation paths

3.4.1 Create ServicePage and AlarmPage

In order to be able to show manual navigation, 2 additional pages must be created with new content on each ("ContentServicePage" and "ContentAlarmPage"). The 2 pages should have the same layout ("MyLayout") as "MainPage" and reference pieces of content "ContentServicePage" and "ContentAlarm-Page".

Exercise: Create ServicePage and AlarmPage

For information about creating pages, see section "Step-by-step page creation".

1) Create ServicePage with ContentServicePage

2) Create AlarmPage with ContentAlarmPage

In order to give the pages a different appearance, attribute "backColor" will be configured individually when assigning to the respective main content.

Name	ServicePage	AlarmPage
ContentServicePage	x	
ContentAlarmPage		x
ContentTop/AreaTop	x	x
ContentLeft/AreaLeft	x	x
ContentRight/AreaRight	x	x

Table 5: Assigning areas and pieces of content

3.4.2 NavigationBar widget

The "NavigationBar" widget is used as a container for "NavigationButton" widgets. When "NavigationButton" widgets are added to the navigation bar, they can be automatically positioned or aligned with other "NavigationButton" widgets that have been added. This saves the user from having to manually align each individual "NavigationButton" widget.

NavigationBar widget

After "ServicePage" and "AlarmPage" have been created, a "NavigationBar" widget is added to "ContentLeft" and configured as follows:

Add the "NavigationBar" widget from the Widget Catalog to "ContentLeft" using drag-and-drop.

Set the size of the container. "Layout/Size" = 100; 300

In order for "NavigationButton" widgets to be relatively aligned with one another, change the "Behavior/child positioning" property to "relative".

Property name	Value
Layout/Size	100;300
Behavior / Child positioning	relative
Common/Name	NavigationBar1

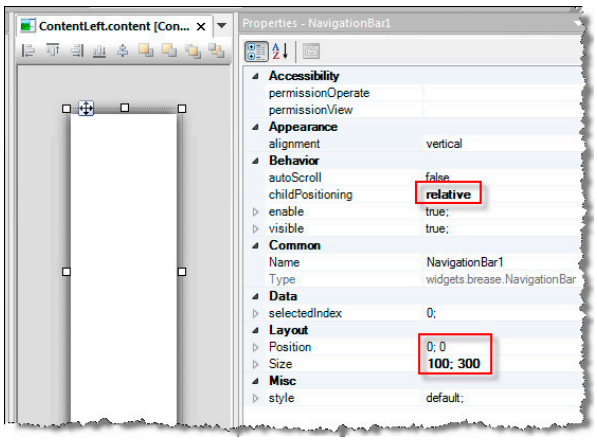


Figure 23: NavigationBar widget in the Properties window

3.4.3 NavigationButton widget

A "NavigationButton" widget makes it possible to navigate to a certain page.

Adding and configuring "NavigationButton" widgets

After the "NavigationBar" widget has been added and configured, a "NavigationButton" widget is added and configured for "MainPage".

"NavigationButton" widgets are added from the Widget Catalog to the "NavigationBar" widget using drag-and-drop. The text displayed on the "NavigationButton" is specified by the "Appearance/text" property. The unique name of the "NavigationButton" widget is defined by the "Common/Name" property, and the page that should be navigated to when clicking on the "NavigationButton" widget is specified with property "Data / pageId".

Image NavigationButton in the Properties window shows the values for the "NavigationButton" widget for navigating to "MainPage".

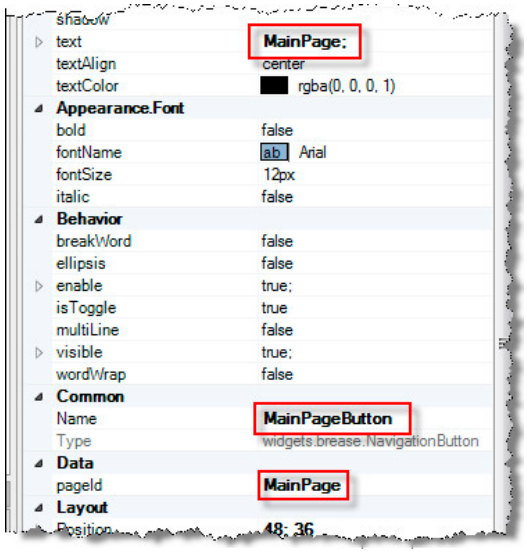


Figure 24: NavigationButton in the Properties window

Exercise: Add navigation buttons to ServicePage and AlarmPage

This procedure must be repeated with the following parameters for navigating to "ServicePage" and "AlarmPage".

NavigationButton name	Appearance / text	Common/Name	Data / pageId
MainPage	MainPage	MainPageButton	MainPage
ServicePage	ServicePage	ServicePageButton	ServicePage
AlarmPage	AlarmPage	AlarmPageButton	AlarmPage

Table 6: Properties of the navigation buttons

3.4.4 Entering ServicePage and AlarmPage in the .vis file

In order for the newly created pages to be taken into account in the HMI application, the pages must be referenced in the visualization object (.vis).

"ServicePage" and "AlarmPage" must be entered in element <Pages> as shown in the following image.

```
<StartPage pageRefId="MainPage" />
    <Pages>
        <Page refId="MainPage"/>
        <Page refId="ServicePage"/>
        <Page refId="AlarmPage"/>
    </Pages>
```

3.4.5 Compiling the project and opening it in a browser

After all "NavigationButton" widgets have been added and configured, the project can be compiled and transferred to ARsim.

The HMI application can be displayed using the following URL:

http://127.0.0.1:81/index.html?visuId=Training

Exercise: Compile the project and display it in a browser

- Compiling and transferring the project

1) Open the browser and enter the URL
2) Test the manual navigation.

Expected result

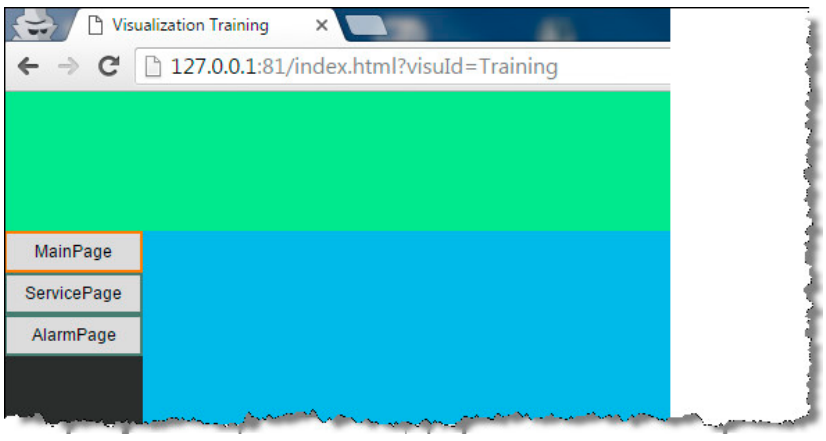


Figure 25: HMI application in the browser with manual navigation

3.5 Using automatic navigation

The goal of this exercise is to create an automatic navigation system that can then be used to navigate between pages.

The "Navigation" widget that is used to display the automatic navigation options will be added to "ContentRight" using the visual editor.

Unlike manual navigation, the "NavigationButton" widget should not show the current page. This is implemented by configuring automatic navigation in a "navigation object".

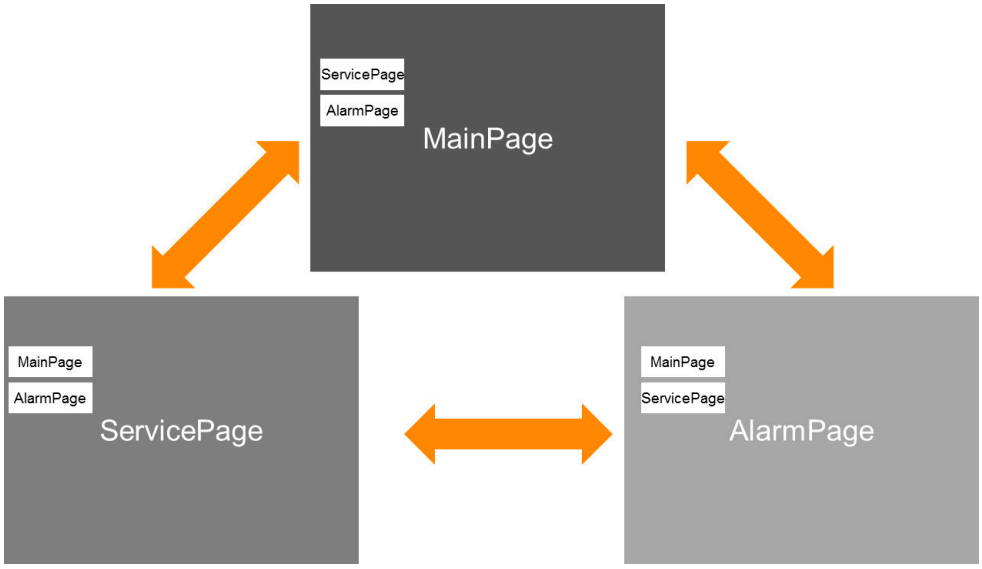


Figure 26: Automatic navigation paths

Exercise: Set up automatic navigation

Automatic navigation should be set up with the help of the following sections.

1) Add the navigation object to the Configuration View and configuring the navigation paths
2) Add the "Navigation" widget to "ContentRight" and configuring it
3) Enter the navigation ID in the .vis file
4) Compile/Transfer the project and display it in a browser

3.5.1 Navigation object in the Configuration View

With automatic configuration, the navigation paths are defined in a separate navigation object in the Configuration View. The "NavigationButton" widgets are displayed in the "Navigation" widget automatically at runtime.

A navigation object must be added to the "mapp View" package from the Object Catalog.

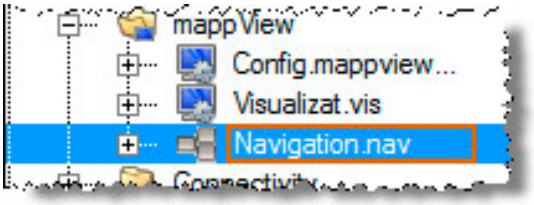


Figure 27: Added navigation object

Each navigation object must contain a unique "ID" as well as navigation paths that define from which source page (NavigationPath refId="") it is possible to navigate to which target page (Destination refId=""). Attribute "index" specifies the order in which the "NavigationButton" widgets are placed in the "Navigation" widget at runtime.

Configuring the navigation object

The first step is to assign unique <Navigation ID> "AutoNavigation".

The navigation paths are then defined as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<ndef:Navigation id="AutoNavigation" xmlns:ndef="http://www.br-
automation.com/iat2015/navigationDefinition/v2">
<NavigationPaths>
    <NavigationPath refId="MainPage">
        <Destination refId="ServicePage" index="0" />
        <Destination refId="AlarmPage" index="1" />
    </NavigationPath>

    <NavigationPath refId="ServicePage">
        <Destination refId="MainPage" index="0" />
        <Destination refId="AlarmPage" index="1" />
    </NavigationPath>

    <NavigationPath refId="AlarmPage">
        <Destination refId="MainPage" index="0" />
        <Destination refId="ServicePage" index="1" />
    </NavigationPath>

</NavigationPaths>
</ndef:Navigation>
```

3.5.2 Adding and configuring the Navigation widget

The "Navigation" widget automatically displays the navigation buttons for a page based on the defined navigation paths.

Adding the Navigation widget

Add a "Navigation" widget from the toolbox using drag-and-drop and configure it as follows:

First define the size of the "Navigation" widgets under "Layout / Size". In order for the "NavigationButton" widget to be shown correctly without being cut off, the default value of the "NavigationButton" widget must be changed under "Layout.Size / buttonWidth".

The unique name for the widget is entered under "Common / Name".

In order for the correct navigation paths to be used, the reference ID of the navigation object ("AutoNavigation") is specified under "Data / navRefId".

Property name	Value
Layout/Size	100;250
Layout.Size / buttonWidth	100
Common/Name	AutoNavigationWidget
Data/navRefId	AutoNavigation

Table 7: Properties of AutoNavigationWidget

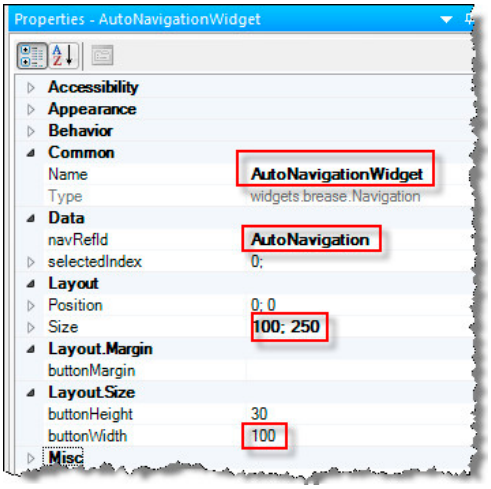


Figure 28: Properties window for Navigation widget

3.5.3 Referencing navigation in the .vis file

In order for the navigation object to be taken into account in the HMI application, its "Navigation ID" must be referenced in the .vis file.

Uncomment attribute <Navigations> in the XML and enter "AutoNavigation" for the "refId" attribute.

```
<Navigations>
  <Navigation refId="AutoNavigation" />
</Navigations>
```

3.5.4 Compile/Transfer the project and display it in a browser

After the "Navigation" widget has been positioned and all navigation paths added and configured, the project can be compiled and transferred to ARsim.

The HMI application can be displayed using the following URL:

http://127.0.0.1:81/index.html?visuId=Training

Exercise: Compile the project and display it in a browser

1) Compile and transfer the project
2) Open the browser and enter the URL
3) Test automatic navigation

Expected result

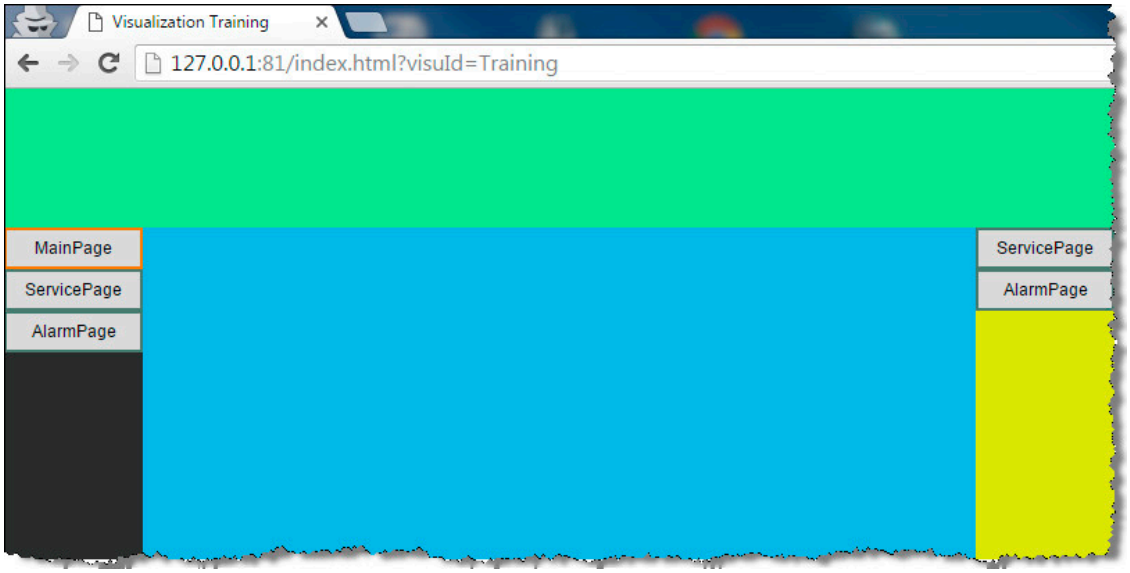


Figure 29: HMI application with manual and automatic navigation

In the previous navigation examples, the "ID" of the page was displayed as a label for the "NavigationButton" widget. How to make a customized label is explained in section "Localization".

4 Visual appearance - Styling

Styling refers to the ability to design the appearance of an HMI application without affecting how it actually functions.

4.1 Styling overview

The way an HMI application looks in its entirety as well as the appearance of its individual elements is determined using styleable properties, styles and themes.

4.1.1 Styleable property

A styleable property refers to a single property of an HMI element (e.g. widget or page) that influences its appearance. Examples of styleable properties include the background color of a widget or the width of its border.

4.1.2 Style

A style groups together all of an HMI element type's styleable properties. There can be several styles for each type of an HMI element. Let's look at the button widget as a type of HMI element. A style for the button widget type (button style) contains a concrete value for each styleable property of a button. Several styles are possible for the button widget type by setting each styleable property of a button to a certain value. For example, the background color for one button style can be green; for another, it can be set to blue. Styles are assigned to the different HMI element instances during project development. When a style is assigned to an instance of an HMI element (e.g. Button01), all of the instance's styleable properties are set to the values from the assigned style.

4.1.3 Theme

A theme is a grouping of styles for different types of HMI elements. A theme can therefore contain styles that set the background color to different shades of blue for all HMI element types. Another theme can contains styles that set the background color to different shades of green for all HMI element types. In this way, it is possible to design blue and green themes. The values of the styleable properties in a theme's styles make it so they fit very well together. Another use case for themes involves toggling

between day/night display in HMI applications where natural light plays a role. The night theme can show widgets with dark background colors and bright text fonts while daylight styles can include bright background colors and dark text fonts.

Figure 30: Displaying an HMI application with different themes

4.2 Using styles

The goals of this exercise are to change the visual appearance of the already existing "NavigationButton" widgets for manual navigation using styles, to create custom styles for the "Button" widgets and to use themes.

4.2.1 Styleable property

The subset of styleable properties is defined for each widget type. The properties that affect the appearance of a widget instance will vary depending on the type of widget.

HMI application/mapp View/Widgets/Navigation Button

Styling the NavigationButton widgets

The goal of the next exercise is to change the appearance of the "NavigationButton" widgets for manual navigation located on "ContentLeft".

In the Properties window, the background color should be changed to light red for "NavigationButton" widget "MainPage" and to light green for "NavigationButton" widgets "ServicePage" and "AlarmPage".

Exercise: Apply styling to the "MainPage" button

After selecting the "NavigationButton" widget called "MainPage", select the color light red (rgba(255, 0, 0, 1)) in the Properties window under "Appearance / backColor".

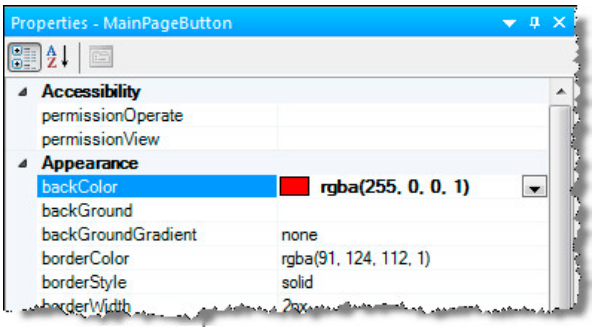


Figure 31: backColor property in the Properties window

Exercise: Apply styling to the "NavigationButtons" widgets for ServicePage and AlarmPage

Repeat the same procedure for the "NavigationButton" widgets for "ServicePage" and "AlarmPage".

Use the color value for light green (rgba(0, 255, 0, 1)).

Display the HMI application in a browser

After the background color of the "NavigationButton" widgets has been changed, the project can be compiled and transferred to ARsim.

The HMI application can be displayed using the following URL:

http://127.0.0.1:81/index.html?visuId=Training

Exercise: Compile the project and display it in a browser

1) Compile and transfer the project.
2) Open the browser and enter the URL

Expected result

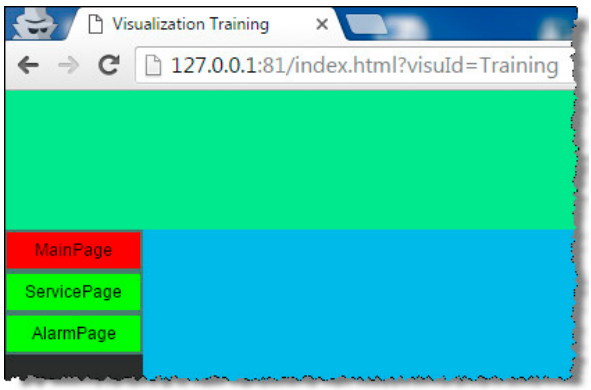


Figure 32: HMI application in the browser with changed "backColor" for the navigation buttons

New specifications for the look and feel of the HMI application

After the background color has been changed, the specifications for the background color have changed and the current color tones for the "NavigationButton" widgets are no longer correct.

The background color of the "NavigationButton" widgets should be changed to dark red (rgba(192, 0, 0, 1)) and dark green (rgba(0, 192, 0, 1)).

Exercise: Change the background color of the "NavigationButton" widgets again

Change the background color of the "NavigationButton" widgets again to the following:

1) "MainPage" button = "rgba(192, 0, 0, 1)"

2) "ServicePage" and "AlarmPage" buttons ="rgba(0, 192, 0, 1)"

In this HMI application, there were only 3 "NavigationButton" widgets whose appearance was changed. In real HMI applications, you have to think that there can be an enormous number of widgets whose appearance needs to be adjusted. If you were to do this using the styleable properties of each widget instance, the work needed to do so – as well as the probability of error – would be unreasonably high.

In order to change the appearance of a large number of widget instances, there is a better way using styles.

4.2.2 B&R theme and styles

B&R provides a theme that makes it possible to quickly and easily give an appealing appearance to an HMI application. This theme contains predefined custom styles for all widget types that can be used by specifying the style name.

The goal of this exercise is to use the provided BuRTheme packages and included styles for new "Button" and "NavigationButton" widgets for manual and automatic navigation.

The following section explains how to create styles and themes in more detail.

Adding the B&R theme

The first step is to add the B&R theme from the Object Catalog to the Logical View under package "Resources / Themes".

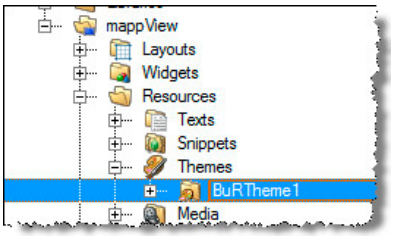


Figure 33: B&R theme package in the Logical View

? HMI application/mapp View/mapp View development environment/Themes and styles/Adding a theme

Adding the button widgets

Two "Button" widgets are added to content "ContentMainPage", one to the left and one to the right.

Predefined style "Command2" should be assigned to the left "Button", while predefined style "Operate2" should be assigned to the right "Button".

Exercise: Add two "Button" widgets to ContentMainPage

Add two "Button" widgets to "ContentMainPage", one to the left and one to the right.

Assign a style to the button widgets

The predefined styles in the B&R theme can be assigned to the "Button" widgets in the Properties window.

After selecting the respective "Button" widget, style "Command2" can be assigned to the left button and style "Operate2" to the right button.

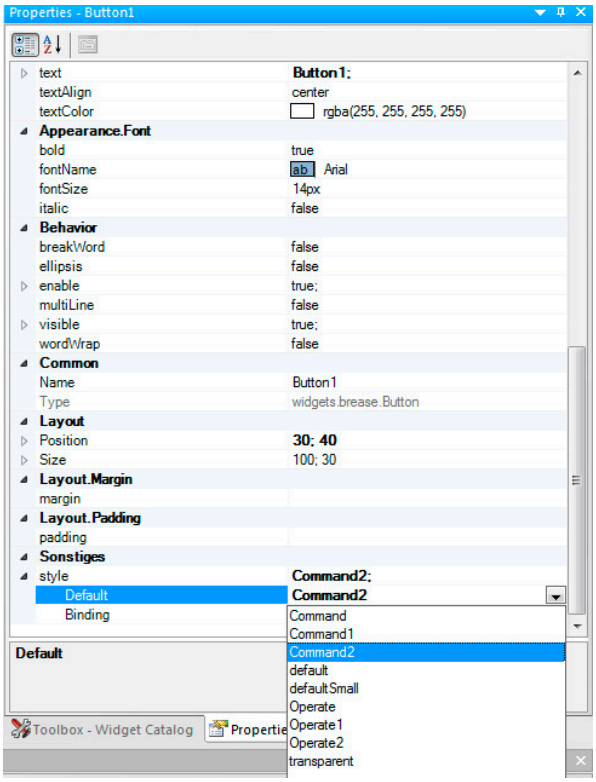


Figure 34: Selecting a style in the Properties window

Exercise: Assign a style to the button widgets

Assign style "Command2" to the left button and style "Operate2" to the right button.

Expected result:

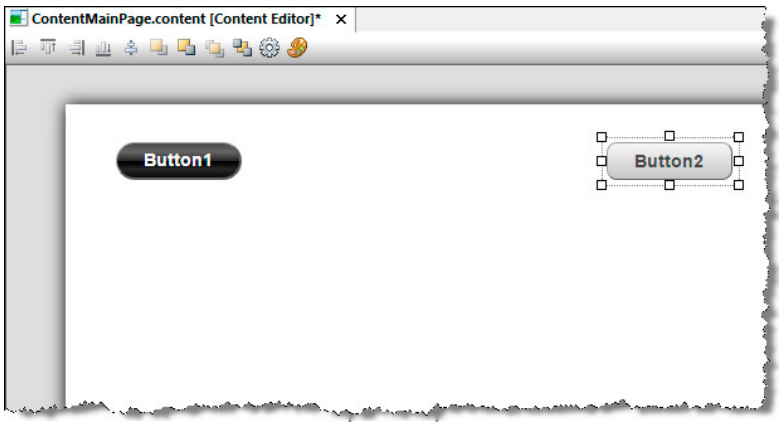


Figure 35: Assigned styles in the visual editor

Default style

Each widget has a default style that is used automatically whenever a style called "default" is included in a theme.

Since a default style is defined in the B&R theme package for each widget, it will be used automatically for each widget instance as long as the user does not change the style property. In this way, a style does not have to be assigned to the "NavigationButton" widgets for manual and automatic navigation. The appearance of the navigation buttons styled in section "Styleable property" changes automatically when the B&R theme is added.

Entering the B&R theme in the .vis file

In order for the B&R theme to be used in the HMI application, it must be referenced in the .vis file in the Configuration View.

```
<StartTheme themeRefId="BuRTheme1" />
<Themes>
  <Theme refId="BuRTheme1"/>
</Themes>
```

Compile/Transfer the project and display it in a browser

After all "Button" widgets have been added and styles assigned, the project can be compiled and transferred to ARsim.

The HMI application can then be displayed using the following URL:

http://127.0.0.1:81/index.html?visuId=Training

Exercise: Compile the project and display it in a browser

1) Compile and transfer the project.
2) Open the browser and enter the URL

Expected result

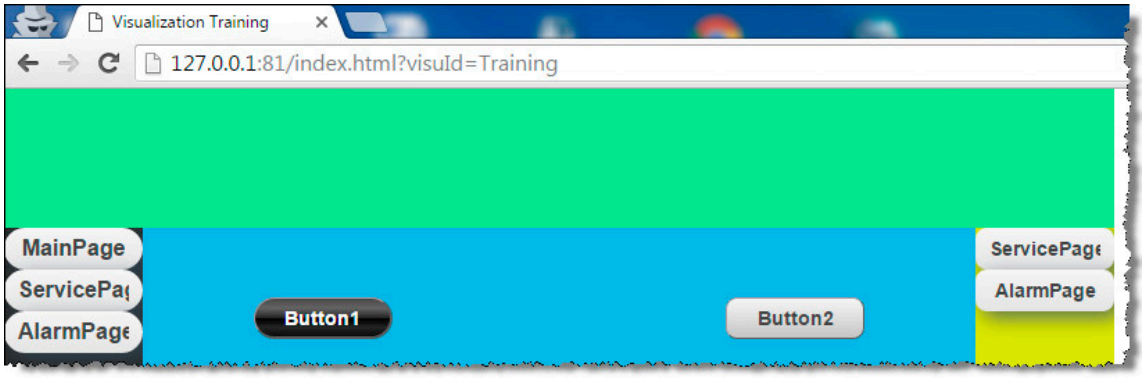


Figure 36: HMI application in the browser with B&R theme

4.2.3 Create a theme with styles

This section explains how to create a custom theme with styles to use on the "Button" widgets created earlier.

Exercise: Create a theme with styles

The goal of this exercise is to create a theme and 2 styles for the existing "Button" widgets.

The procedure looks like this:

1) Add a new theme package
2) Smart devices such as tablets, smartphones, etc. are considered perfect examples of powerful technology with ultimate usability.
3) Add a style file
4) Create and assign styles for button widgets
5) Enter the theme in the Visualization.vis file
6) Compile/Transfer the project and display the HMI application

Add a new theme package

HMI application/mapp View/mapp View development environment/Themes and styles/Creating a theme

The first step is to add an "empty theme package" from the Object Catalog. This package is used to structure the StyleSet files with the underlying style classes.

An empty theme package can be added from the Object Catalog after selecting the "Theme" package in the Logical View. The theme package contains a .theme file.

To keep a clear structure, the name of the theme package and the included .theme file is changed to "MyThemePackage" and "MyTheme", respectively.

The StyleSet files to be used are referenced in the .theme file. In addition, the unique ID used to reference the theme in the visualization object is specified in the .theme file.

Double-clicking on the .theme file opens it in the XML editor. You can then enter a unique <Theme ID="MyTheme"> as well as the <StyleSet refId="MyStyleSet"> described in the next step.

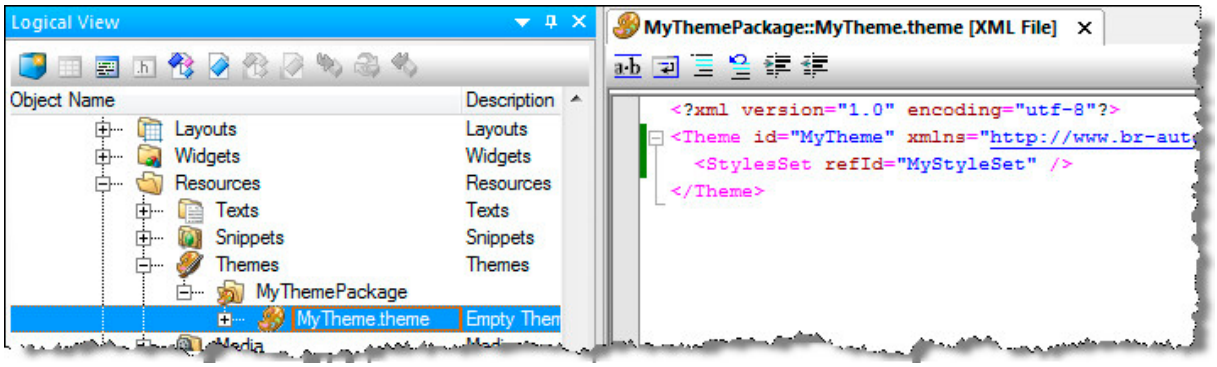


Figure 37: Added theme package and assigned ID

StyleSet package

HMI application/mapp View/mapp View development environment/Themes and styles/Creating a theme

A StyleSet package can be added under the "MyThemePackage" package from the Object Catalog. When a StyleSet package is added, a .StyleSet file is created as well. The StyleSet package is used to structure the style files.

After the StyleSet package including the .styleset file have been added and renamed to MyStyleSet in the Logical View, it can be opened in the XML editor by double-clicking on it.

The first thing to enter is the unique <StyleSet ID="MyStyleSet"> that has been assigned in the .theme file.

The <Styles refId> of the style referred to in the next step is then referenced.

After 2 styles have been created for the existing "Button" widgets, "MyButtonStyle" is assigned as the "refId".

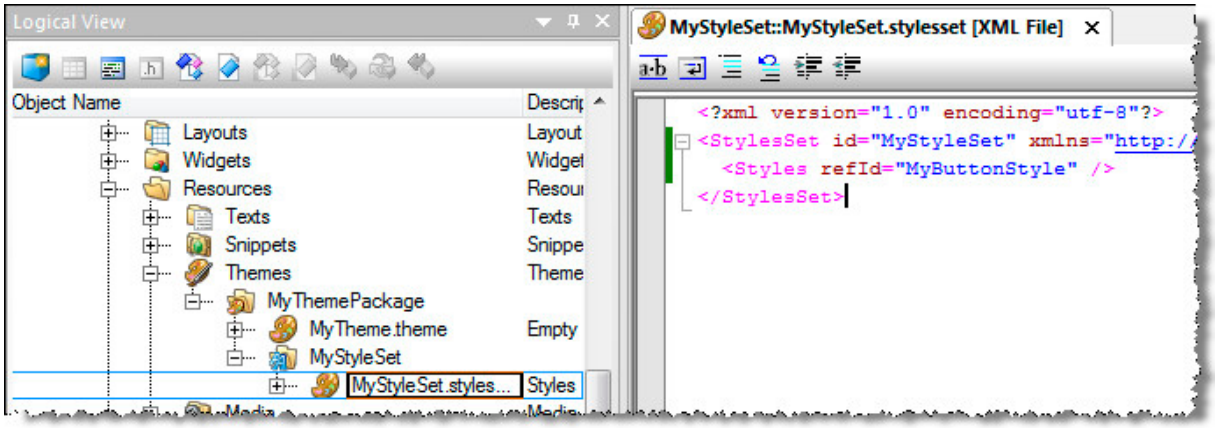


Figure 38: Added StyleSetPackage and assigned ID

Style file

HMI application/mapp View/mapp View development environment/Themes and styles/Creating a theme

After selecting the MyStyleSet package in the Logical View, a style file can be added from the Object Catalog. It can then be renamed to "MyButtonStyle" as specified earlier in the StyleSet file.

The unique ID is entered: <Styles ID="MyButtonStyle">.

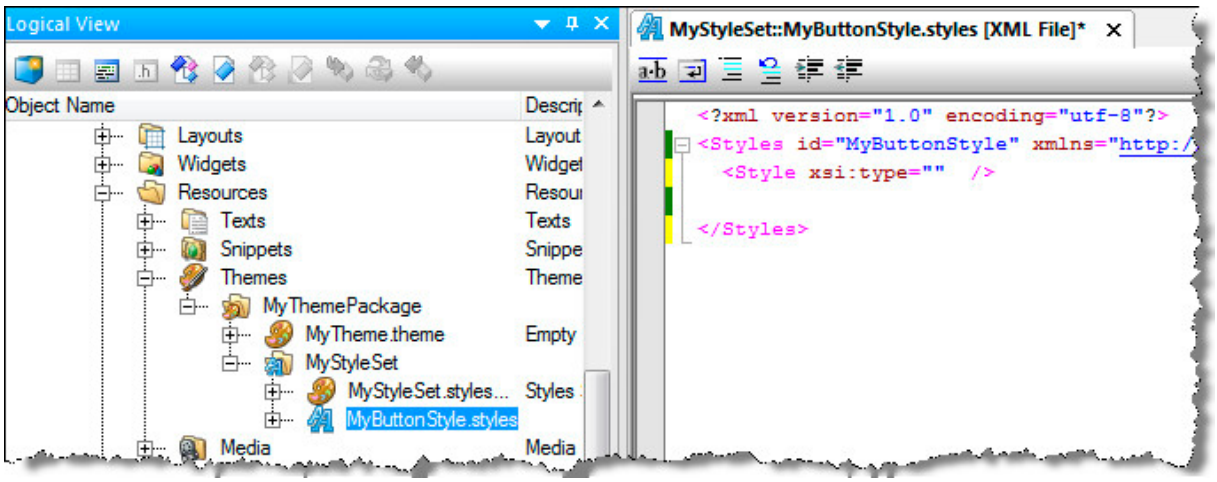


Figure 39: Added style file and assigned ID

Creating styles for a button widget and assigning them to a widget instance

HMI application/mapp View/mapp View development environment/Themes and styles/Creating a theme

Information about all styleable properties belonging to a widget is available in Automation Help.

The next step is to create 2 different styles in the style file for the "Button" widgets (style "Command2" and style "Operate2").

These styles should have the following styleable properties:

Style "ID"	xsi:type=	backColor=	textColor=	bor-derStyle=	border-Width=	cornerRa-dius=
"Com-mand2"	"widget-s.brease.Button"	"#FF0000"	"#000000"	"solid"	"4px"	"10px"
"Operate2"	"widget-s.brease.Button"	"#00FF00"	"#000000"	"solid"	"4px"	"10px"

Table 8: Properties for button styles "Command2" and "Operate2"

Expected result:

```
<?xml version="1.0" encoding="utf-8"?>
<Styles id="MyButtonStyle"
xmlns="http://www.br-automation.com/iat2015/styles/engineering/v1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <Style xsi:type="widgets.brease.Button" id="Command2"
         backColor="#FF0000"
         textColor="#000000"
         borderStyle="solid"
         borderWidth="4px"
         cornerRadius="10px"
      />

      <Style xsi:type="widgets.brease.Button" id="Operate2"
        backColor="#00FF00"
        textColor="#000000"
        borderStyle="solid"
        borderWidth="4px"
        cornerRadius="10px"
     />
</Styles>
```

Entering the theme in the .vis file

In order for the newly created theme to be used at runtime, it must be referenced in the .vis file in the Configuration View.

```
  <StartTheme themeRefId="MyTheme" />
  <Themes>
    <Theme refId="BuRTheme1"/>
    <Theme refId="MyTheme"/>
  </Themes>
```

Compile/Transfer the project and display the HMI application

After all of the styles for the buttons have been created and assigned, the project can be compiled and transferred to ARsim.

The HMI application can be displayed using the following URL:

http://127.0.0.1:81/index.html?visuId=Training

Exercise: Compile the project and display it in a browser

1) Compile and transfer the project.
2) Open the browser and enter the URL

Expected result

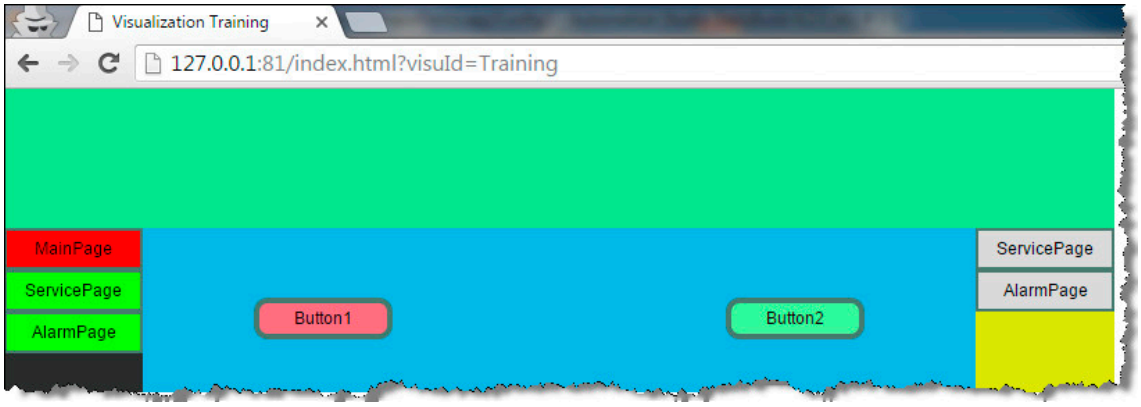


Figure 40: HMI application in the browser with theme and assigned styles for the buttons

5 Data binding

5.1 Data binding overview

5.1.1 OPC UA

OPC Unified Architecture (OPC UA) is a standard for interoperability that ensures secure and reliable data exchange in industrial automation and other industries. OPC UA is platform-independent and can be used to seamlessly exchange data between devices from different vendors.

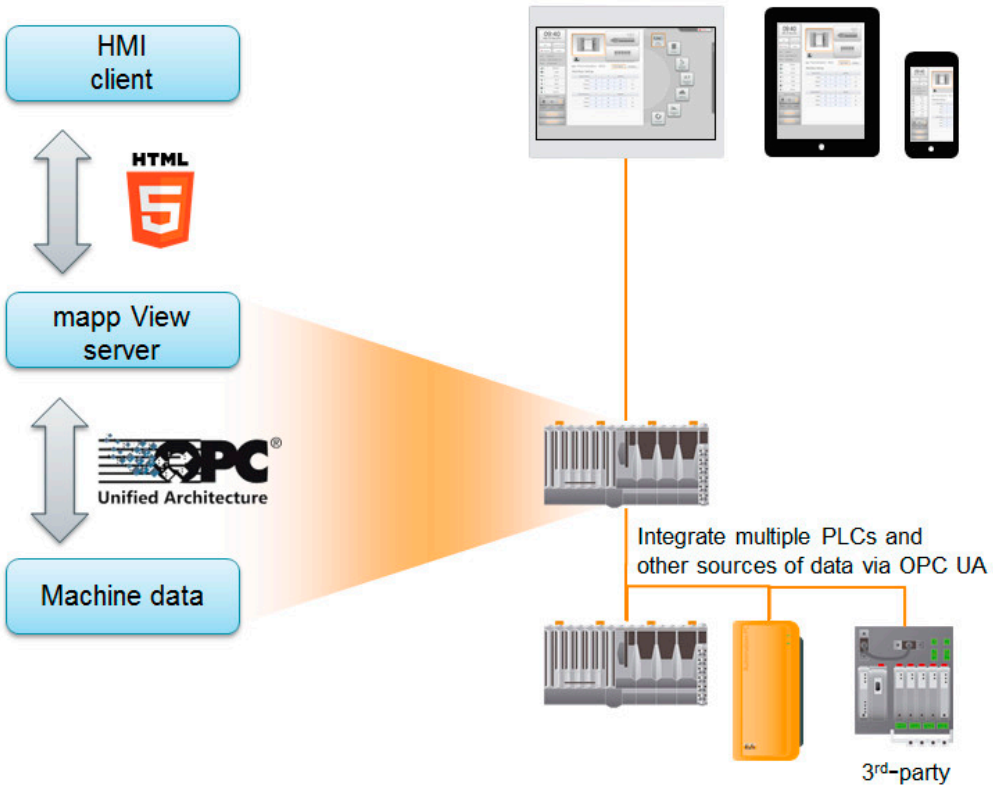


Figure 41: OPC UA architecture

OPC UA node

If a process variable from an automation application is made available via the OPC UA system, it can be read or written from OPC UA clients as an OPC UA node. Properties such as engineering unit or EU range can be added to an OPC UA node.

Engineering unit

The engineering unit for an OPC UA node specifies the physical unit to be used when interpreting the value. Automation Studio provides 1,400 engineering units for seven physical quantities: length, mass, time, current, temperature, material amount and luminosity as well as derived physical quantities (e.g. speed, force, pressure, acceleration).

EU range

The EU range (engineering unit range) determines the valid range of values for an OPC UA node. The EU range is defined by a lower and upper limit value.

OPC UA default view

The OPC UA default view contains all process variables in an automation application that are made available by the OPC UA server on the B&R controller to the OPC UA clients.

5.1.2 Binding

An HMI application facilitates interactions between people and a machine. The HMI application must be able to display data from the automation application (current sensor values, current machine states, etc.).

It must also be possible for machine operators to intervene in the automation application (to deploy a new setpoint, for example). For these reasons, elements of an HMI application (e.g. widget instances) must be connected with elements from the automation application in order for data to be exchanged between the HMI application and automation application.

In mapp View, this connection is implemented using what is known as binding (data binding). The binding in a mapp View HMI application defines which data sources (OPC UA nodes in an automation application) are connected to which HMI elements (e.g. widget instances), the type of binding as well as the binding mode (how the data is exchanged).

mapp View differentiates between the following types of bindings:

- "Value binding" for binding single values
- "Node binding" for binding OPC UA nodes with a unit and limits
- "Array binding" for binding arrays
- "List binding" for selecting variables from a list
- "Complex binding" for binding structures

Only binding types "value binding" and "node binding" will be included in this seminar. All other types are included in the mapp View advanced seminars.

Binding modes

Specifying the binding mode defines the direction in which data should flow.

Binding mode "oneWay" (Read only)

The "oneWay" binding mode is used for read access to the source.

Example: Binding between an OPC UA node and an output widget (e.g. NumericOutput widget)

Binding mode "twoWay" (Read / Write)

The "twoWay" binding mode is used for read and write access to the source.

Example: Binding between an OPC UA node and an input widget (e.g. NumericInput widget)

Binding mode "oneWayToSource" (Init Read / Write)

The "oneWayToSource" binding mode is used for write access to the source.

Example: Binding between an OPC UA node and a PushButton widget.

5.2 Using data binding

The goal of the next exercises is to display the values of process variables from the automation application in the HMI application using various types of binding and binding modes.

5.2.1 Displaying data in the HMI application

The goal of this exercise is to display the value of the "Current temperature" OPC UA node in the HMI application.

Exercise: Display the CurrentTemperature variable value using the NumericOutput widget

In order to achieve this goal, a "NumericOutput" widget and "Read only" binding mode are used to display the value.

The following steps must be carried out here:

1) Program and variables
2) Enable OPC UA server
3) Add OPC UA default view and enable the OPC UA node
4) Add binding file from the Object Catalog
5) Add NumericOutput widget and assign the OPC UA node
6) Define BindingSet ID and .enter it in the .vis file
7) Compile the project, transfer it and display it in the HMI application

Program and variables

The program and the required global variable (CurrentTemperature) for displaying the value in the "NumericOutput" widget already exist.

It is not necessary to create other programs or additional variables.

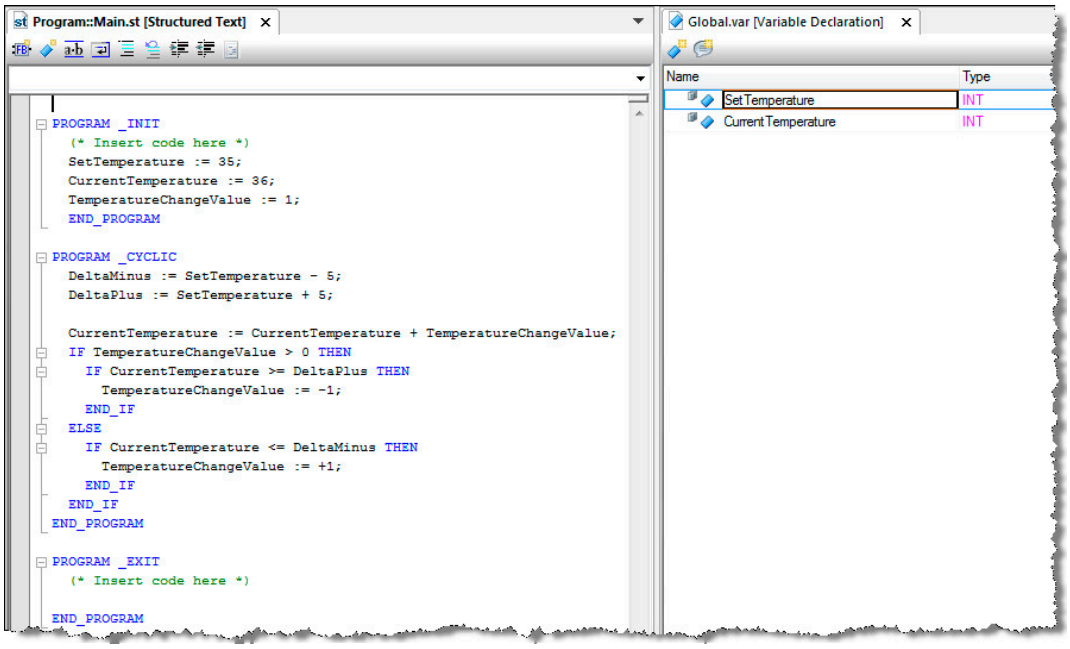


Figure 42: Global variable CurrentTemperature

Enable OPC UA server

For communication to take place between the automation application and the HMI application, the OPC UA server must be enabled.

The OPC UA server is enabled in the CPU configuration in the Physical View under the OPC UA system node.

Communication/OPC UA/Enabling the server

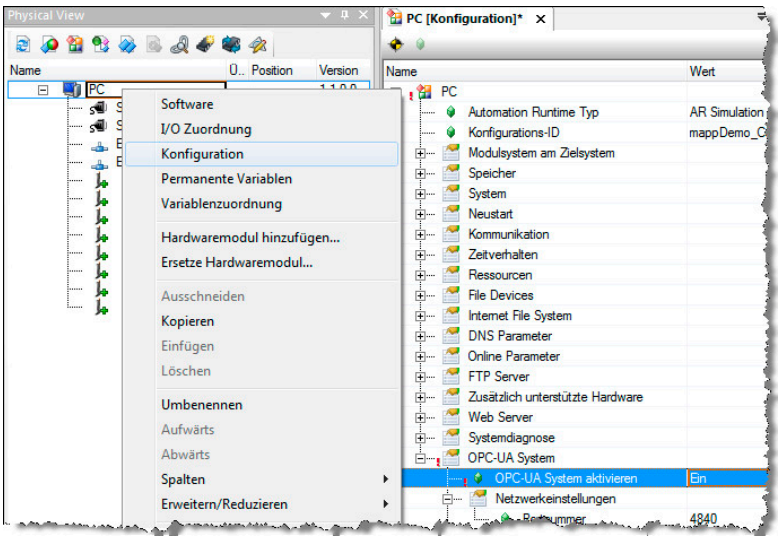


Figure 43: Enable OPC UA server

Add OPC UA default view and enable the OPC UA node

In order for process variables that have been created to be used, it must be declared as an OPC UA node. To do this, the OPC UA default view must be added in the Configuration View under the "Connectivity/OpcUA" node from the Object Catalog via drag-and-drop.

Communication/OPC UA/Default view editor

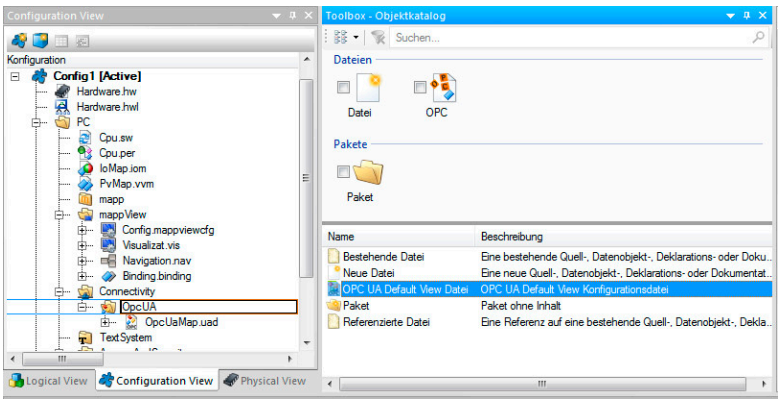


Figure 44: Add OPC UA default view from the Object Catalog

The global process variable can then be enabled as an OPC UA node.

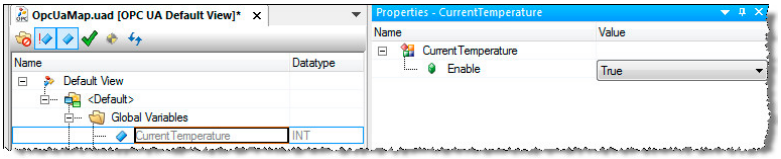


Figure 45: Enable the global process variable in the OPC UA default view

Add the binding file from the Object Catalog and specify the ID

All bindings created using the "Select Variable dialog box" are automatically entered in a binding file by the system.

By selecting the mapp View node, a binding file can be added from the Object Catalog.

For the system to know which binding file the binding has to be entered in, a unique "ID" must be defined for the binding file.

"MyBinding" should be entered as the unique "ID"

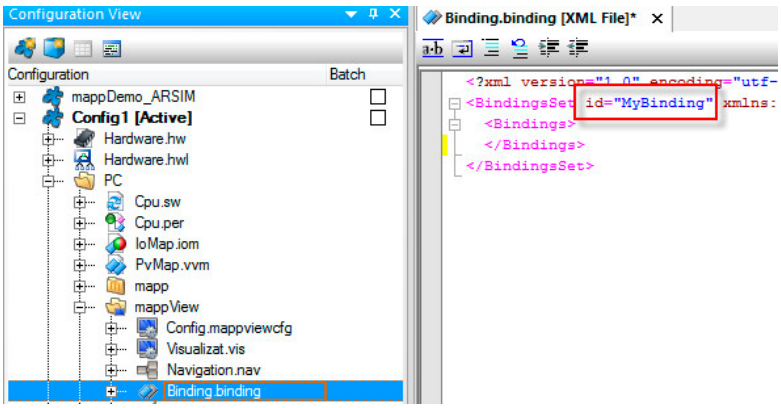


Figure 46: Added binding file and assigned ID

Add NumericOutput widget and assign the OPC UA node

In order for the value of the OPC UA node to be displayed in the HMI application, a "NumericOutput" widget must be added to the "MainPage" content.

Value binding is used if only the numeric value is needed without units or limits. Data forwarding only involves the value of the bound variable.

Double-clicking on "ContentMainPage" opens the visual editor and a "NumericOutput" widget from the Object Catalog can be placed.

Since this is a "NumericOutput" widget, "Read-only" binding mode is used. "MyBinding" is selected as binding set because the bindings that have been cretaed will be saved here.

By selecting the "NumericOutput" widget, the "Select Variable dialog box" can be opened in the Properties window under "Data/Value/Binding". The OPC UA node is then selected in the dialog box.

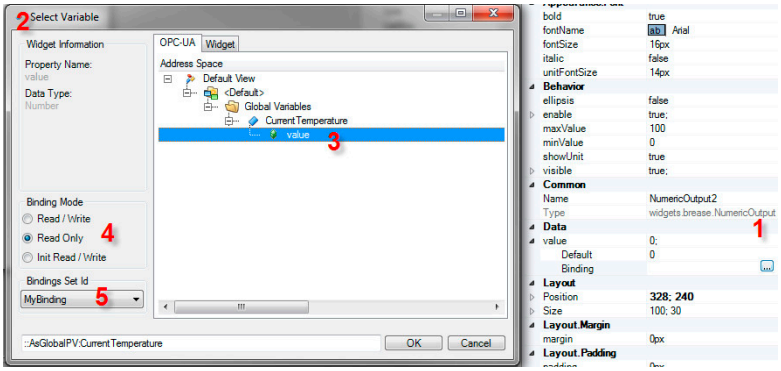


Figure 47: Open the binding dialog box and select the properties

Reference the binding file in the .vis file

For the binding file that has been created to also be taken into account, the unique "ID" for the file ("MyBinding") must now be entered in the .vis file.

Under <BindingSets>, the BindingSet section can be commented out and the unique "ID" ("MyBinding") can be entered.

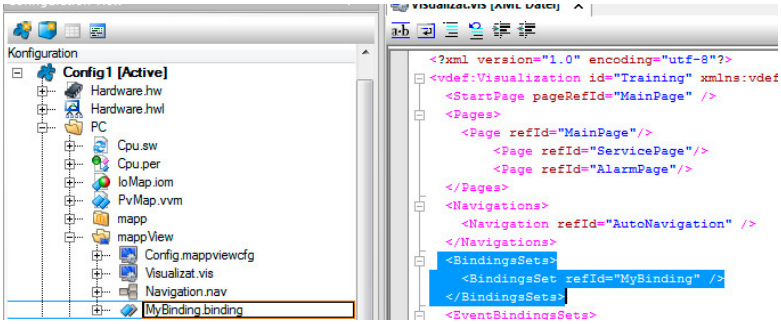


Figure 48: Reference the binding file in the .vis file.

Compile the project, transfer it and display it in the HMI application

After binding the of variable to the numeric output has been completed, the project can be compiled and transferred to ARsim.

The HMI application can be displayed using the following URL:

http://127.0.0.1:81/index.html?visuId=Training

Exercise: Compile the project and display it in a browser

1) Compile and transfer the project.
2) Open the browser and enter the URL

Expected result:

Figure 49: HMI application displayed in the browser with value shown the NumericOutput widget

5.2.2 Setpoint entry

The goal of this exercise is to change the setpoint of the OPC UA node (SetTemperature) when a user makes an entry in thge HMI application.

Exercise: Change setpoint using a NumericInput widget

To achieve this goal, a numeric input" widget will be placed in the "MainPage" content and connected to the corresponding OPC UA node in the automation application via binding. In order for the value changed in the HMI application to be written, TwoWay ("Read / Write") binding mode will be selected.

The following steps must be carried out here:

1) Program and process variable
2) Enable the variable as an OPC UA node
3) Add NumericInput widget and assign the OPC UA node
4) Compile the project, transfer it and display it in the HMI application

Program and variable

The program and the required global variables (CurrentTemperature and SetTemperature) for displaying the value in the "NumericOutput" widget already exist.

It is not necessary to create other programs or additional variables.

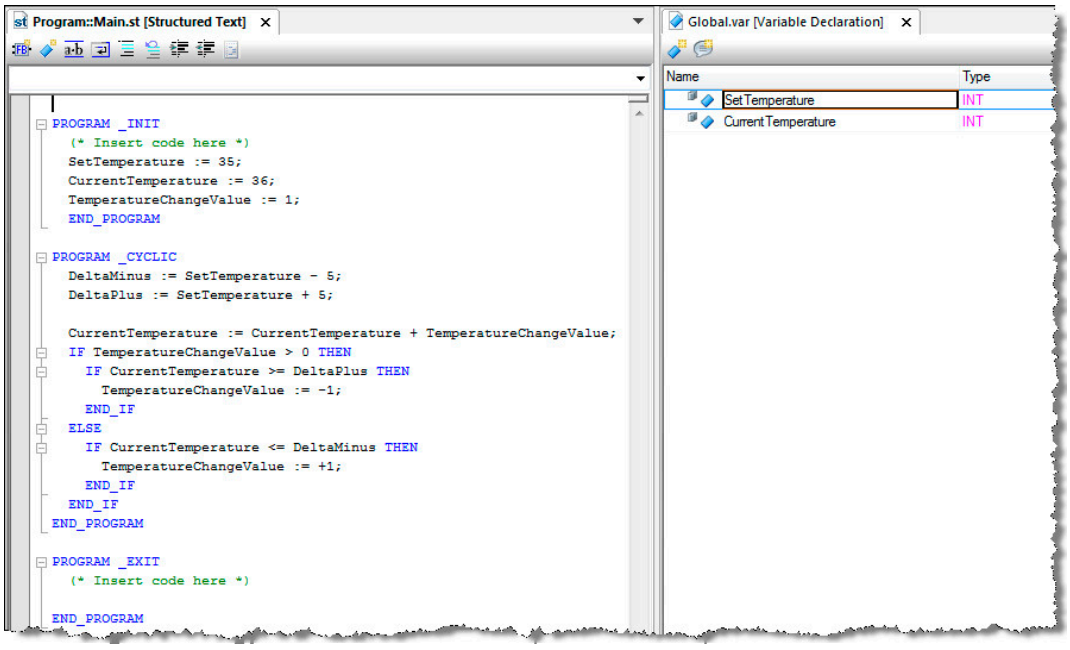


Figure 50: Global variables CurrentTemperature and SetTemperature)

Exercise: Enable the process variable as an OPC UA node

Since the OPC UA server has already been enabled and the OPC UA default view can only be added once, the variable (SetTemperature) can be enabled right away in the OPC UA default view.

1) Enable the variable as an OPC UA node

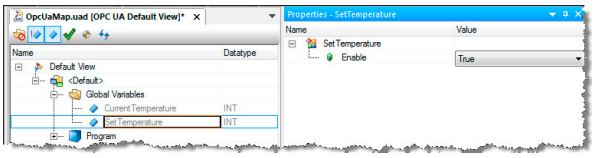


Figure 51: Enable the SetTemperature process variable in the OPC UA default view

Add a numeric input and assign the variable

Since all bindings are stored in the same file, another binding file does not have to be created.

Therefore a "NumericInput" widget can now be added to "ContentMainPage".

Since it is necessary to interact with the "NumericInput" widget and the value of the OPC UA node should be configurable by the user, "Read / Write" must be selected as binding mode. This makes it possible for the operator to change the value.

Exercise: Add and configure the NumericInput widget

1) Add the NumericInput widget to ContentMainPage
2) Assign NumericInput to the SetTemperature OPC UA node
3) Select "MyBinding" as the binding set ID
4) Use Read / Write as binding mode

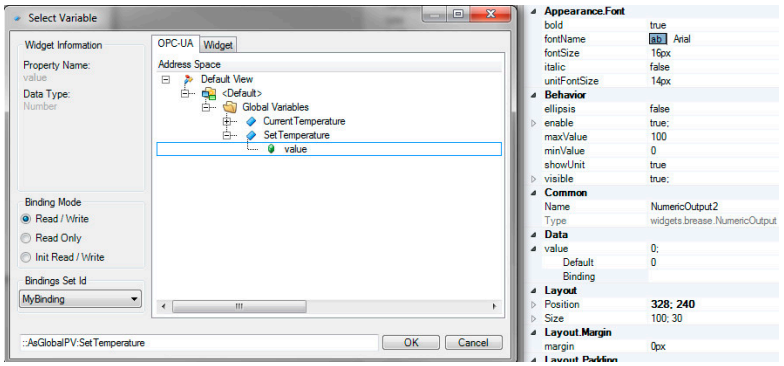


Figure 52: In the binding dialog box, connect the numeric input with the SetTemperature OPC UA node

Compile the project, transfer it and display it in the HMI application

After another variable has been created, enabled as OPC UA node and set to binding mode "Read/ Write", the project can be compiled and transferred to ARsim.

The HMI application can be displayed using the following URL:

http://127.0.0.1:81/index.html?visuId=Training

Exercise: Compile the project and display it in a browser

1) Compile and transfer the project.
2) Open the browser and enter the URL

Expected result

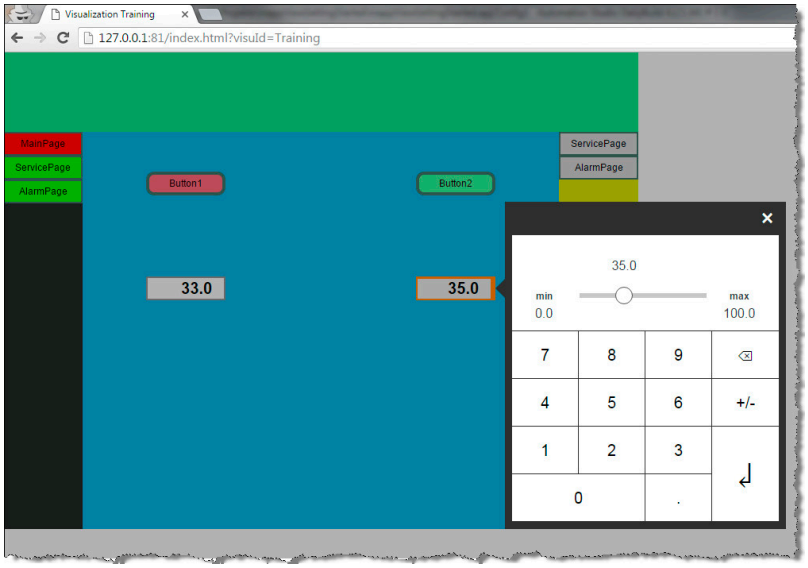


Figure 53: Display the HMI application in the browser with the selected NumericInput widget

5.2.3 Specify the value range

Up to now, the temperature setpoint (SetTemperature) can be set to any value. This is not always useful. To prevent the user from entering arbitrary or impermissible values, the EU range (low/high limit values) for an OPC UA node can be defined. Defining the EU range therefore makes it impossible for the user to enter impermissible values.

Exercise: Limit the value range of the numeric input

The goal of this exercise is to prevent the user form entering an impermissible value.

In order to achieve this goal, limits will be assigned to the OPC UA node.

The following steps must be carried out:

1) Define low/high values for the OPC UA node
2) Change binding to "NodeBinding"
3) Compile the project, transfer it and display it in the HMI application

Define low/high values for the OPC UA node

Double-click on the OPC UA default view and select the OPC UA node (SetTemperature) to update the Properties window.

Then the low and high value for the node can be set under the "EU range" node.

EU range	Value
Low	25
High	50

Table 9: EU range properties for the SetTemperature OPC UA node

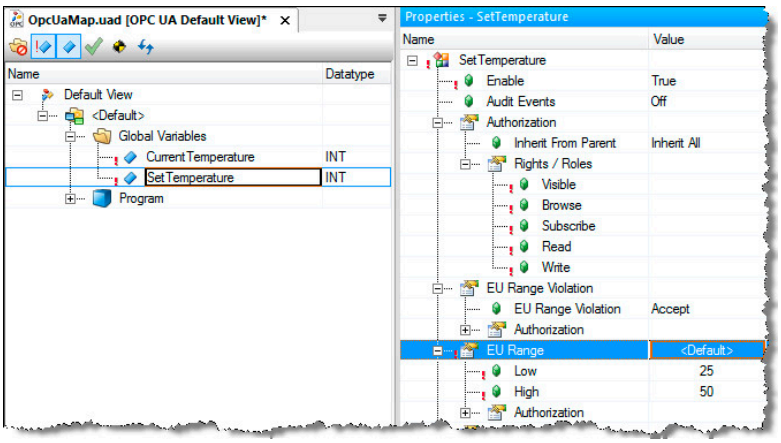


Figure 54: Define the EU range fro the SetTemperature OPC UA node

Communication/OPC UA/Default view configuration/Default view editor/Properties/Range of values/Role-based range of values

Change the binding to Node-Binding

To ensure that the EU range for the OPC UA nodes will be taken into account, binding must be changed from value binding to node binding.

Node binding is configured by selecting the entire OPC UA node ("SetTemperature").

For the operator, it is now no longer possible to enter a value larger or smaller than what is defined in the EU range.

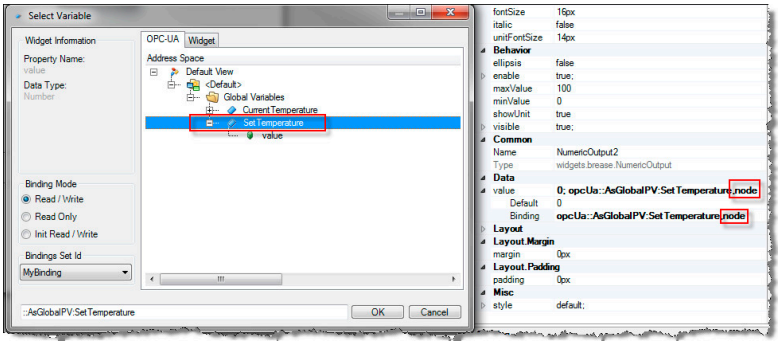


Figure 55: Assign NumericInput to the SetTemperature OPC UA node

Compile the project, transfer it and display it in the HMI application

After the limits have been defined for the OPC UA node, the project can be compiled and transferred to ARsim.

The HMI application can be displayed using the following URL:

http://127.0.0.1:81/index.html?visuId=Training

Exercise: Compile the project and display it in a browser

1) Compile and transfer the project.
2) Open the browser and enter the URL

Expected result

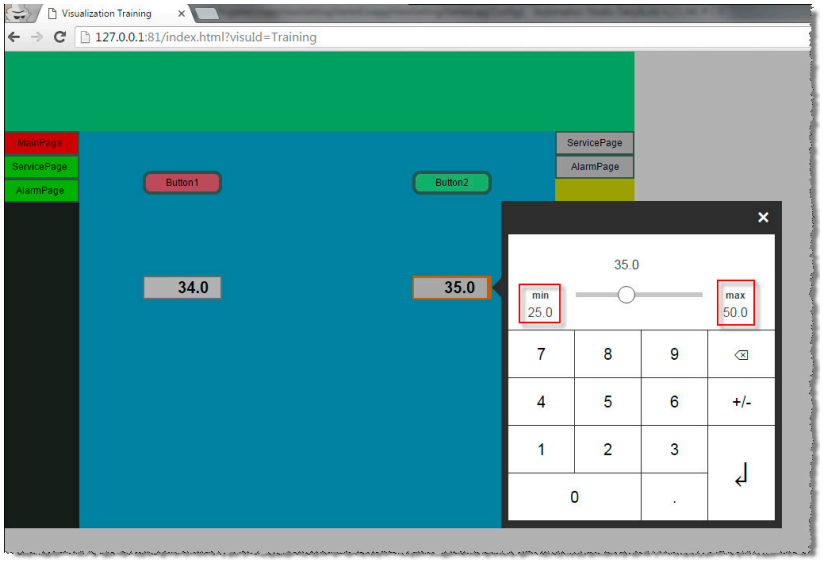


Figure 56: Display the HMI application in the browser with defined EU range for SetTemperature OPC UA node

6 Media files

6.1 Overview – Media files

Media files can be added to mapp View in order to improve the appearance of an HMI application.

6.2 Adding media files

6.2.1 Insert an image

The Media package is automatically made available in new mapp View projects and is used to logically manage additional files such as images used in the HMI application. It can be structured using packages.

All common image formats such as .png, .jpg, .svg, etc. can be used.

All files in this folder will be transferred to the target system.

The path to reference images always begins with the "Media" package followed by the file name of the image or the sub-package and then the file name of the image.

Example for the image "test.png", which is stored directly in the Media package:

URL= "Media/Test.png"

Example for the image "Test.png", which is stored in the "SmallPictures" sub-package in the Media folder:

URL= "Media/SmallPictures/Test.png"

HMI application/mapp mapp View/HMI organization/Logical View/mapp View Media package

Exercise: Insert an image

The goal of this exercise is to display a company logo on every page.

The company logo should be added in the lower right corner of "ContentRight" using the "Image" widget.

The following steps are necessary for this:

1) Save the image in the Media folder
2) Add the image widget and place it as needed
3) Configure the image in the widget
4) Compile the project, transfer it and display it in the HMI application

Save the image in the Media folder

Using drag-and-drop, images can be easily added to the Media package in the Logical View. In the prepared project, the B&R logo is already saved in the Images package outside of the mapp View package and therefore only needs to be moved to the mapp View Media package.

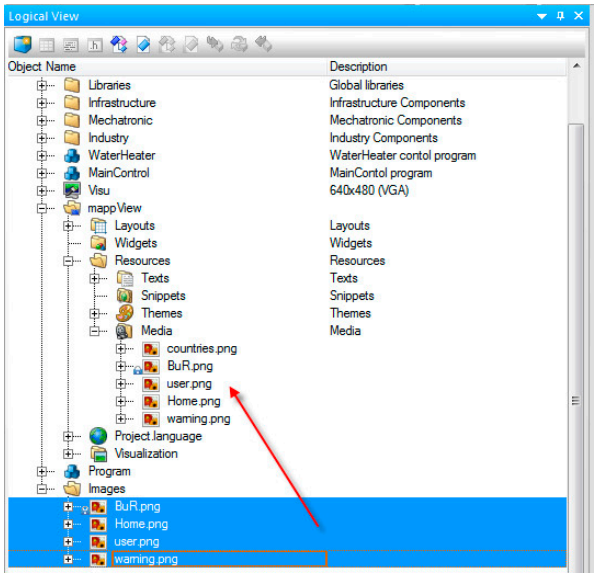


Figure 57: Save the images in Media package

Add, place and configure the Image widget

After selecting the content, an Image widget from the Object Catalog can be added.

Then the size and position of the widget is defined in the Properties window.

The following values can be set for the widget under "Layout/..":

Name	value (px)
Position "top"	444
Position "left"	0
Size "width"	100
Size "height"	60

Table 10: Image widget properties

Configuring the image in the widget

To reference the B&R logo that has already been added in the widget, the location and the name of the image must be entered under "Appearance/image".

Since the image is stored directly in the "Media" package and is named "BuR.png", the path is as follows:

"Media/BuR.png"

Since the size of the "Image" has already been adjusted and the default property setting is "sizeMode"="contain", the "Image" widget is completely filled by the image.

Information about other "sizeMode" properties can be found in the help system.

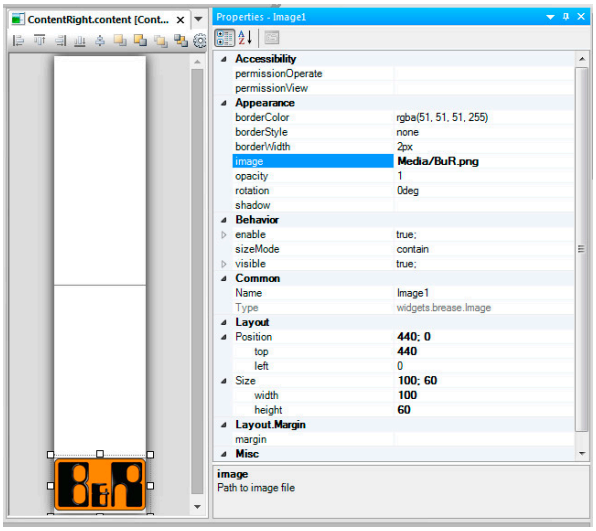


Figure 58: Image widget properties

HMI application/mapp View/Widgets

Compile the project, transfer it and display it in the HMI application

After all Image widget has been added and the B&R logo referenced, the project can be compiled and transferred to ARsim.

The HMI application can be displayed using the following URL:

http://127.0.0.1:81/index.html?visuId=Training

Exercise: Compile the project and display it in a browser

1) Compile and transfer the project.
2) Open the browser and enter the URL

Expected result:

Figure 59: Displaying the HMI application in the browser with a B&R logo added

6.2.2 Image in the NavigationButton widget

Images can also be displayed in a "Button" widget. This possibility is used in the following exercise to implement the "Home" icon for navigation to "MainPage" on a "NavigationButton".

Exercise: Configure the Home image on the NavigationButton

The goal of this exercise is to implement an icon on the "NavigationButton" instead of using the normal "NavigationButton" widgets for "Navigation" to "MainPage".

So that the "NavigationButton" does not have to be place on each page individually, the "NavigationButton" will be placed in ContentLeft, which is referenced on each page.

The following steps must be carried out:

1) Add and configure the NavigationButton widget
2) Configure the icon on the navigation button
3) Compile the project, transfer it and display it in the HMI application

Add and configure the NavigationButton widget

First, a navigation button from the Object Catalog will be added via drag-and-drop in the bottom portion of ContentLeft. Then the size of the "NavigationButton" widget will be changed and the "NavigationButton" will be placed at the middle bottom of the content.

Since an image should be placed on the "NavigationButton" widget, the default text can be deleted.

MainPage will be selected under the "Data/pageId" property as the page that should be navigated to when clicking on the "NavigationButton" widget.

Property name	Value
Layout/size	48;48
Position/Top/left	452;26
Data/PageId	MainPage
Appearance/Text	

Table 11: NavigationButton widget properties

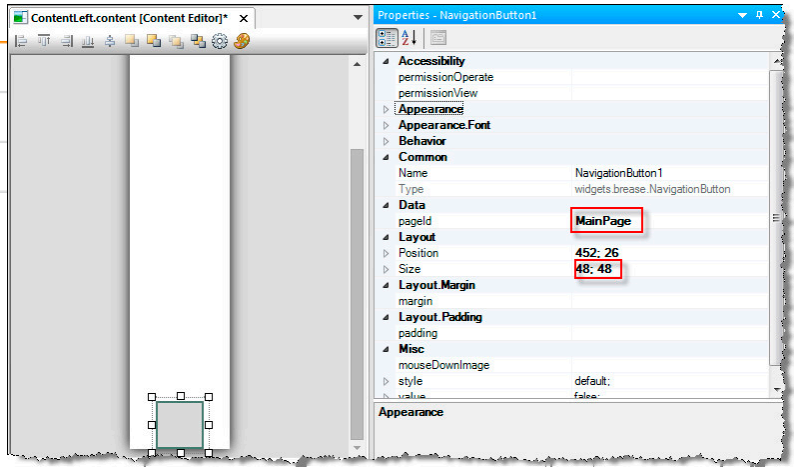


Figure 60: Navigation button that has been added and placed in position

Referencing the image on the NavigationButton widget

The image that should be used for the "NavigationButton" is already stored in the Media package. As with the "Image" widget, the "NavigationButton" has an "Image" property that is used to specify the path of the image.

The "ImageAlign" property is used to ensure that the image that has been added is positioned correctly.

Since the image looks better after clipping, i.e. when isolated so it has no background color and frame, the "backColor" and "borderStyle" properties will be changed.

The following properties should be defined:

Property name	Value
Appearance/image	Media/Home.png
imageAlign	bottom
backColor	Transparent
borderStyle	None

Table 12: NavigationButton widget properties

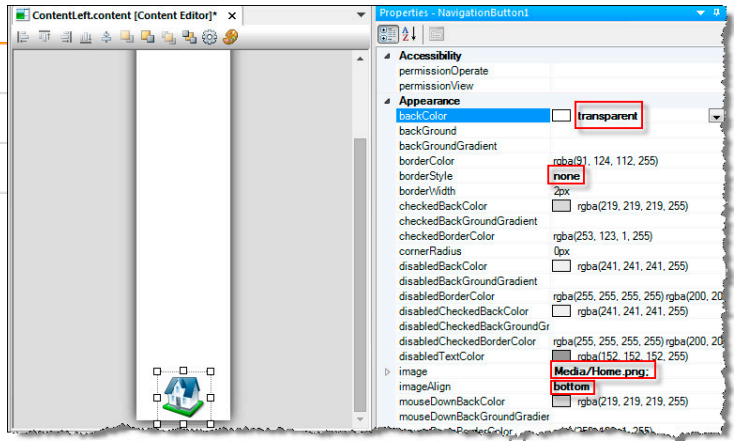


Figure 61: Reference the Home icon on the navigation button

Compile the project, transfer it and display it in the HMI application

After all the properties have been set correctly and the "NavigationButton" has been correctly positioned at the bottom, the project can be compiled and transferred to ARsim.

The HMI application can be displayed using the following URL:

http://127.0.0.1:81/index.html?visuId=Training

Exercise: Compile the project and display it in a browser

1) Compile and transfer the project.
2) Open the browser and enter the URL

Expected result

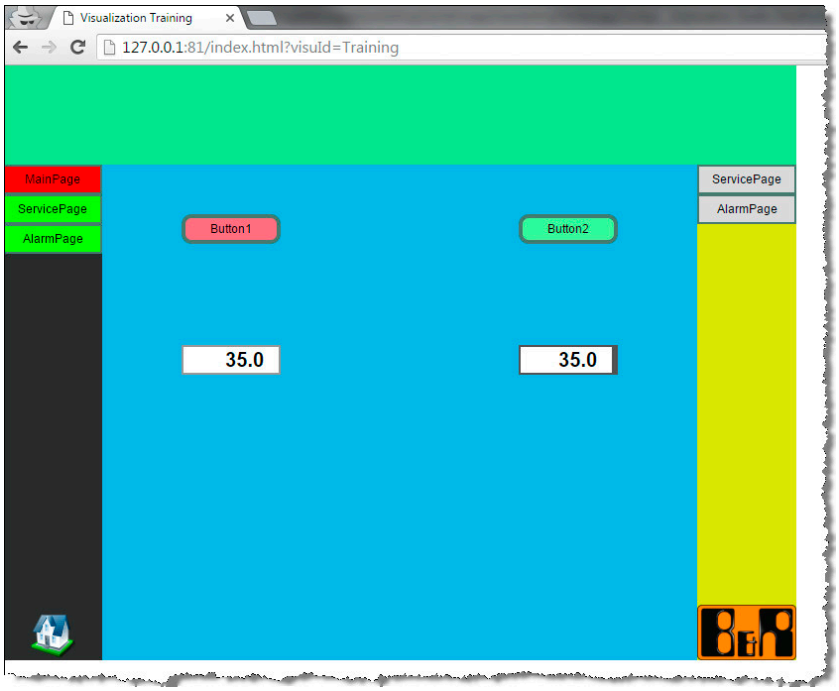


Figure 62: Displaying the HMI application in the browser with Home image as an additional NavigationButton for MainPage

7 User role system

The automation software provides a user roles system which is used by mapp View. This system implements role-based access control (RBAC) as defined in ANSI standard 359-2004.

7.1 Overview – User role system

Role-based access control deals with users, roles and rights. Rights are assigned to roles and roles are associated with users. Users can have several roles at once. Users are not assigned rights directly because this has proven to be complex and error-prone in practical situations.

In the system, a user is a "real" person, who is identified by first and last name, etc. A user also incorporates information used for authentication in the system. With this authentication, a "real" person can prove to the system that they are who they claim to be. The most commonly used authentication method consists of a unique identification of the user (User ID) and a secret password that only the "real" person and the system know.

A role describes how a user interacts with a system to carry out specific tasks. Examples of roles can be: Administrator, service engineer, machine operator. Different permissions are normally required to complete various tasks, and rights and role are assigned to handle this. If a "real" person's duties change, it is only necessary to change corresponding user to the new roles for this person to have the rights needed to complete the new tasks.

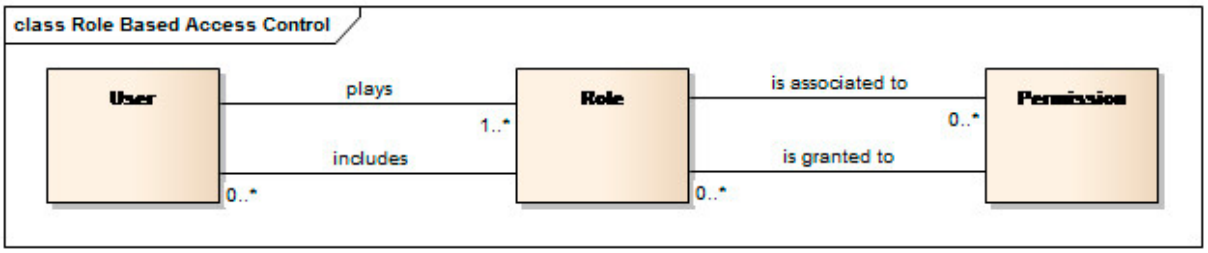


Figure 63: Correlations within the user role system

Access & Security/User role system/General

7.2 Using the user role system

7.2.1 Limiting write access to OPC UA nodes

One right that can be assigned to roles is the right to change the value of a process variable in the automation application (write access). In this case, the right to change the process variable must be assigned to a role. This is done in the OPC UA configuration.

For a user to be able to change the value of an OPC UA node, the user must be assigned to a role that has write access. Only users assigned to a role with write access are allowed to change values.

Exercise: Limit write access for users

The goal of this exercise is to prevent write access to setpoint SetTemperature for non-authorized users. The conditions for this will be created using the use role system. It is also possible to add authentication to the HMI application.

The following steps are necessary for this:

1) Add "Operator" role and "Service" role
2) Add a new user and assign a role
3) Define write access in the OPC UA default view
4) Place the Login, LogoutButton and LoginInfo widgets
5) Compile the project, transfer it and display it in the HMI application

Add a new role

Access & Security/User role system/Configuration in Automation Studio/Configuring roles

Add "Operator" role and "Service" role

In Automation Studio, the roles "Administrators" and "Everyone" already exist in each new project. Add "Operator" and "Service" roles to the existing roles. Double-clicking on the Role.role file in the Configuration View, opens the editor where the roles are managed.

Then add two new roles and rename them to "Operator" and "Service". The assigned "IDs" remain unchanged.

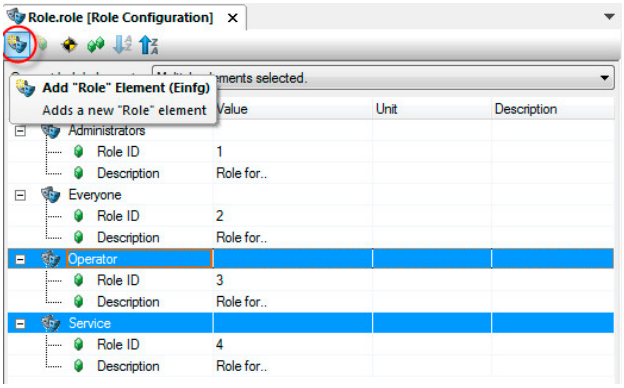


Figure 64: Add "Operator" and "Service" roles

Add a new user and assign a role

In Automation Studio, the user "Anonymous" already exist in each new project.

After the new roles have been created, two new users can be created and assigned roles.

Double-clicking on the User.user file opens the editor where the users are managed.

Then create users "John" and "Dave".

The user "John" is assigned the "Operator" role and the user "Dave" is assigned the "Service" role. The respective name is used as the password.

Username	Password	Role
John	John	Operator
Dave	Dave	Service

Table 13: Overview of users and assigned roles

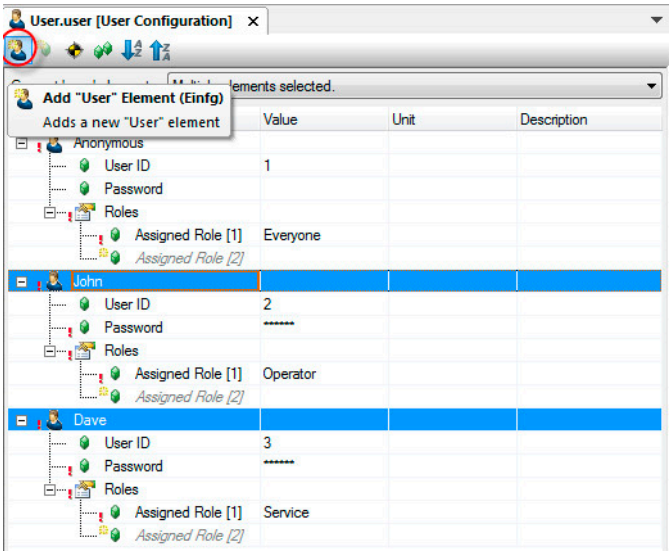


Figure 65: Add and configure users "John" and "Dave"

Access & Security/User role system/Configuration in Automation Studio/Configuring users

Define write access in the OPC UA default view

After the user has been created and roles have been assigned, the rights for the roles are defined in the OPC UA default view.

Selecting the "Default View" node sets the default permissions, from which all permissions for the underlying variables are derived.

The permissions for each role can be defined by adding the various roles. To ensure that write access to OPC UA nodes is only given to users assigned the "Service" role, "Write" is only set for the "Service" role.

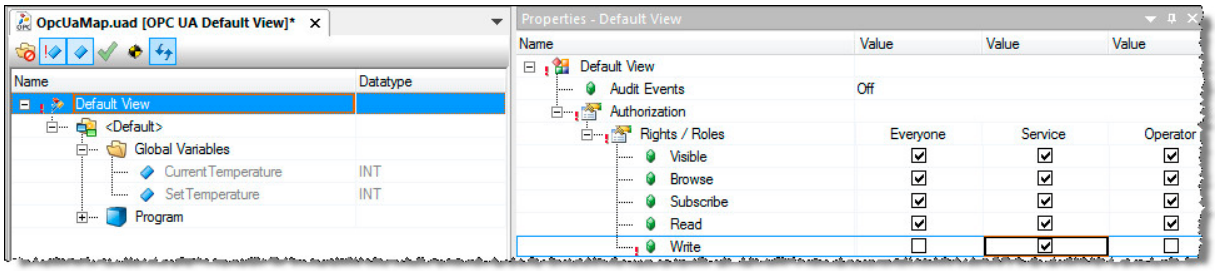


Figure 66: Define write access for roles in the OPC UA default view

Since all OPC UA nodes inherit the permissions set for the "Default View" by default, no further settings have to be made.

Place the Login, LogoutButton and LoginInfo widgets

For the permissions set for each user to take effect, the user needs be authenticated on the system. There are widgets available to log in and log out as well as to display detailed login information.

If a user is not authenticated, the system assumes that user "Anonymous" is interacting with it. The rights given to all roles assigned to user "Anonymous" are valid during non-authenticated use of the system. By default, user "Anonymous" is assigned role "Everyone".

- **Login widget**
  The "Login" widget can be placed on the content of the service page from the Widget Catalog using drag-and-drop.
- **LoginInfo widget**
  In order to see which user is currently logged in, the "LoginInfo" widget should be placed in "ContentTop".
- **LogoutButton widget**
  In order for the user to be able to log out again, a logout button should be placed in "ContentTop".
- **Label and Image widget**
  In order to make the HMI application clearer and better looking, an Image widget with a user icon and a Label widget with the text "Logged in as:" will be placed in "ContentTop". The image for the user icons is already in the Media folder.

If a logged in user is logged out using the LogoutButton, the rights assigned to user "Anonymous" are once again valid.

Set property "Layout.Padding/padding" to 0px.

Widget	Content	Top	Left	Width	Height
Login widget	Service page	320	296	300	177
Login info	Content Top	28	720	78	30
Logout Button	Content Top	66	700	100	30
Label	Content Top	2	690	104	30

Table 14: Widgets, content and properties for login information

Widget	Content	Top	Left	Width	Height
Image	Content Top	24	690	30	38

Table 14: Widgets, content and properties for login information

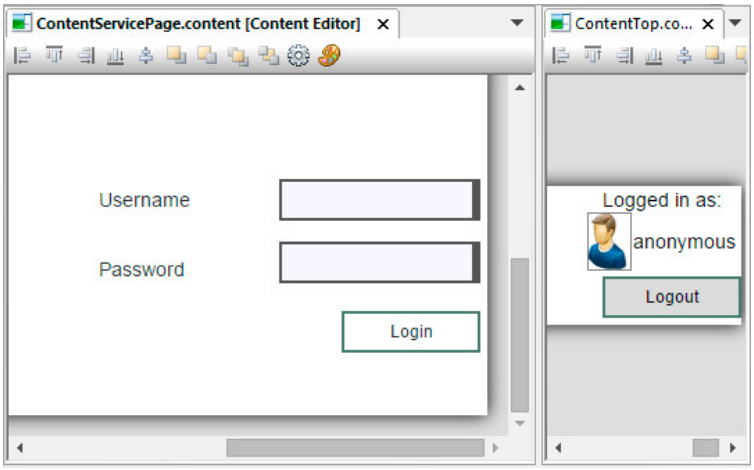


Figure 67: Login, Logout, Label and Image widgets placed on ContentServicePage and ContentTop

If property "Layout.padding/padding" is set to 0px, the distance between the text and the frame of the widget changes.

Compile the project, transfer it and display it in the HMI application

Once the roles and users are created and the widgets have been placed, the project can be compiled and transferred to ARsim.

The HMI application can be displayed using the following URL:

http://127.0.0.1:81/index.html?visuId=Training

Exercise: Compile the project and display it in a browser

1) Compile and transfer the project.
2) Open the browser and enter the URL

Expected result

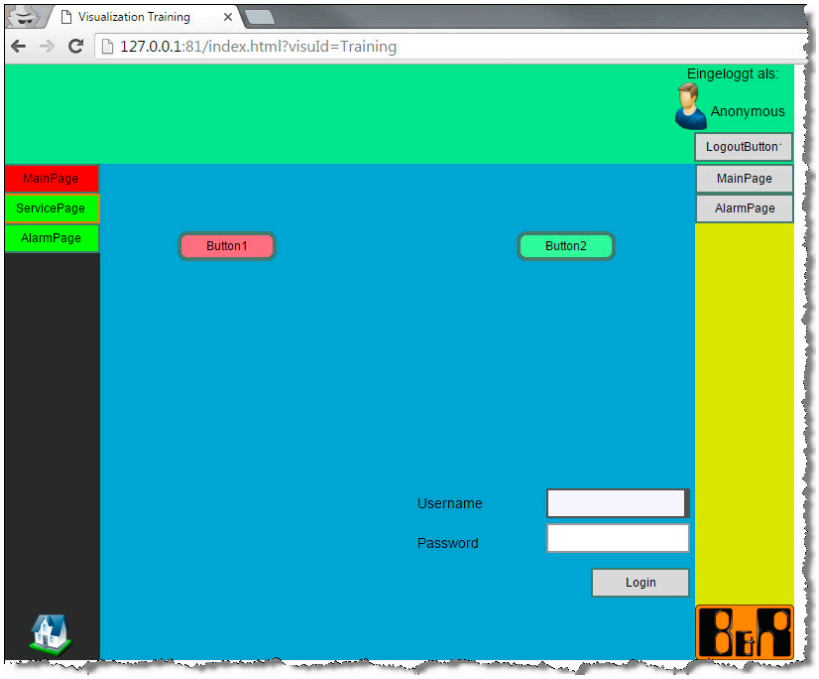


Figure 68: HMI application with Login widget

7.2.2 Limit the range of values

EU range OPC UA node

OPC UA nodes make it possible to limit the range of values (EU range) when making entries. Using the user role system, the EU range can be defined individually for the various roles.

Exercise: Extend range of values for the "Supervisor" role

The goal of this exercise is to extend the EU range of the OPC UA node for the new "Supervisor" role. To achieve this goal, a new role and a new user must be created and the EU range of the OPC UA nodes must be defined.

The following steps must be carried out:

1) Create a new role
2) Create new user, assign the role and define write access
3) Extend EU range for the "Supervisor" role
4) Compile the project, transfer it and display it in the HMI application

Exercise: Create a new role

Create new "Supervisor" role with "ID" = 5.

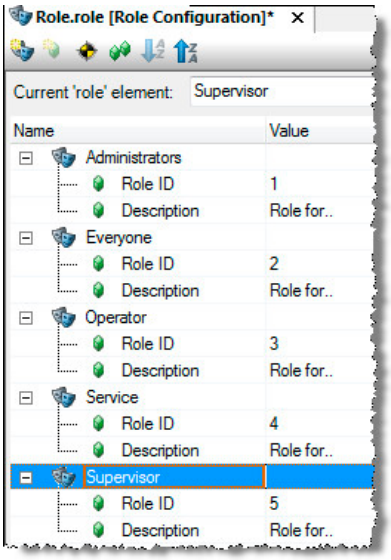


Figure 69: Supervisor role added

Exercise: Create new user, assign the role and define write access

After the new role has been created, another user can be created and assigned the "Supervisor" role. In addition, write access for the role will be assigned in the OPC UA default view.

1) Create user Mike and assign the Supervisor role

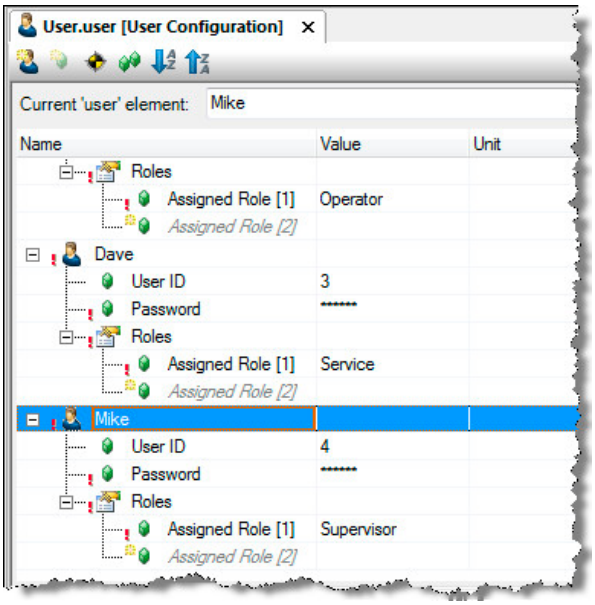


Figure 70: New user Mike created and assigned to the Supervisor role

Extend EU range for the "Supervisor" role

Through the configuration of roles, the range of values can be defined individually for each role in the OPC UA default view. Previously, the "Default" EU range for the OPC UA node was used. By adding a role in the EU range category, the range of values can be extended or limited.

The "Supervisor" role will be added under "EU range". The range of values is derived from the "Default" values based on "Low=20" and "High=60".

Role	EU range LOW	EU range HIGH
Supervi-sor	20	60

Table 15: Extended EU range for the Supervisor role

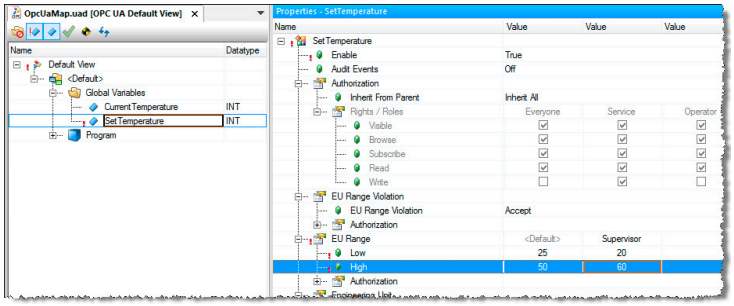


Figure 71: EU range extended for the Supervisor role

Since a customized EU range has not been defined for the other roles that have write access to the OPC UA node, the default values are valid for these roles.

The EU range violation response is the behavior that should occur when you enter a value outside of the EU range.

The following EU range violations responses exist:

EU range violation	Description
Accept	Entry of the value will be accepted even if the value is outside of the EU range.
Reject	The entered value will not be accepted.
Clamp	The entered value is clipped at the high/low limit.

Table 16: Overview – EU range violations

The project can then be compiled and the HMI application can be tested by logging in the various users and therefore setting the respective permissions (read/write access + default/extended value range).

Exercise: Compile the project, transfer it and display it in the HMI application

After the EU range for the new role has been created, the project can be compiled and transferred to ARsim.

The HMI application can be displayed using the following URL:

http://127.0.0.1:81/index.html?visuId=Training

Exercise: Compile the project and display it in a browser

1) Compile and transfer the project.
2) Open the browser and enter the URL

Expected result

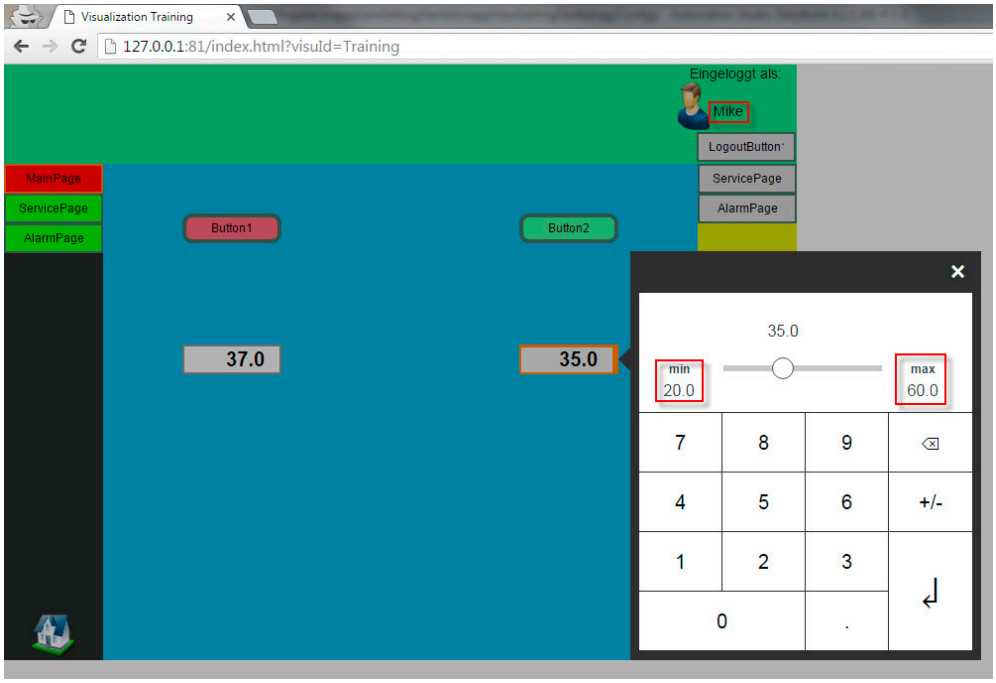


Figure 72: HMI application in the browser with the user Mike logged in and NumericInput selected

7.2.3 Changing and adding roles

Changes are continually made during a machine's lifecycle. Machine users change, new ones are added, others move on, etc. All permissions for the various users depend on the defined roles, so existing users only need t be assigned activities of the appropriate roles.

Exercise: Change users and rights

In this exercise, user "Mike" with the "Supervisor" role goes to a different company. The Supervisor role is taken over by user "John". A new employee named "Patrick" is hired and is assigned the "Operator" role. The user Dave retains the "Service" role.

The goal of the next exercise is to make changes for the users as described:

1) Remove user "Mike"
2) Assign user "John" the new "Supervisor" role
3) Create the new user "Patrick" with password "Patrick" and assign the "Operator" role

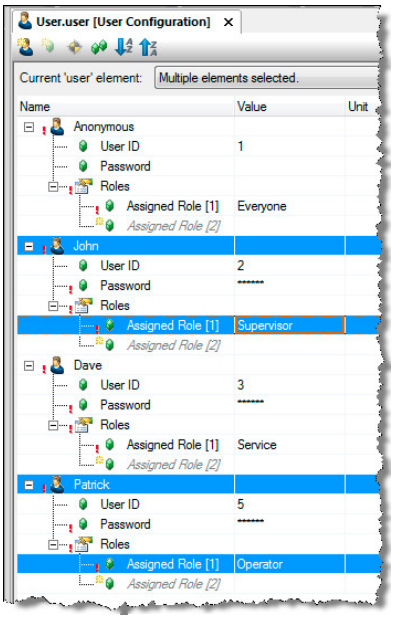


Figure 73: User editor with modified users

8 Localization

With mapp View, localization means adapting the content of an HMI application to the local language and local cultural norms. mapp View allows texts and units to be adapted at runtime for localization of the HMI application.

The text system is used for localization of texts and the unit system is used to adjust the units. The text system and the unit system are not integral parts of mapp View. The functionality of these systems is used by mapp View, so an introduction to use of the text system and unit system is part of this training manual.

8.1 Overview – Text system

Automation Studio provides the text system for localization of texts. This infrastructure functionality is not an integral part of mapp View. The text system makes texts available in multiple languages and can be thought of as a two-dimensional, tabular structure, as shown in the following table.

One dimension lists semantic meanings that are represented by a unique identifier, and the other dimension is formed by all languages with a text-based representation of the semantic information. Each line in this table corresponds to a semantic meaning with texts in the respective languages. Each column in this table represents the texts in a specific language with the various semantic meanings. Each table cell is thus a text for the semantic meaning (identifier in the first column) in the language that this table column represents.

Identifier	de	en	fr	es
Text_Cancel	Abbrechen	Cancel	Annuler	Cancelar
Text_Yes	Ja	Yes	Oui	Si
Text_No	Nein	No	Non	No

Table 17: Example schematic representation of localized texts

The example in the table shows four lines for the semantic meanings of "Cancel", "Abort", "Yes" (approval) and "No" (rejection). These semantic meanings are clearly designated using identifiers "Text_Cancel", "Text_Abort", "Text_Yes" and "Text_No".

For semantic meaning "cancel", there is text is German ("Abbrechen"), English ("Cancel"), French ("Annuler") and Spanish ("Cancelar"). For semantic meaning "abort" (ID: "Text_Abort"), there is a text in German and English, but not in French and Spanish.

If text statements are used in a system with localizable text, identifiers for the semantic meanings of these textual statements must be used. If the system is used in linguistic context "German", the system replaces the identifier for the semantic meanings with the German language texts. Here, the texts from table column "de" are used for the identifier. If the same system is used in linguistic context "English", the system replaces the identifier for the semantic meanings with the English language texts (table column "en").

8.1.1 Identifier

An identifier is a semantic statement that can be represented as text in different languages. Because a large quantity of semantic statements can be contained in an automation system, it is necessary to be able to structure them in order to prevent having the same identifier assigned to different semantic statements. This is particularly important because semantic statements can be defined by different peo-

ple independently of each other. For example, the developer of the automation application can define semantic statements for use within logger entries and, independent of this, the developer of the HMI application can define semantic statements that are used in HMI elements.

The text system provides the ability to structure identifiers for semantic statements in namespaces. An identifier for a semantic statement is composed of several parts:

Identifier = Namespace+NamespaceSeparator+Text_ID

A namespace can in turn also be part of a different namespace. Namespaces can therefore be structured hierarchically. For a namespace:

Namespace = NamespaceName [+ NamespaceSeparator+Namespace]

Identifiers are also referred to as fully qualified Text_IDs. For Text_ID and NamespaceName, there are rules described in detail in the help system. The NamespaceSeparator is a single character, which is also described in the help system.

A few identifier examples are shown below (fully qualified Text_IDs).

1) Texts/AppEvents/Internal/FatalError
2) Texts/Alarms/Alarm1
3) Texts/Program/Alarms/AlarmID1

For all examples, the NamespaceSeparator is the character '/'. The Text_IDs in the examples are "Fatal Error", "Alarm1" and "AlarmID1". The first example contains a namespace called "Texts", which includes another namespace called "AppEvents" and this namespace also includes another namespace called "Internal".

8.1.2 Configuring languages in the project

In an Automation Studio project, it is necessary to define which languages can have texts specified for the semantic statements. The number of project languages must also be specified.

In addition to specifying the project languages that should be provided, the "design language" must also be defined. The design language defines the language used when displaying the texts in Automation Studio during configuration.

8.1.3 Configuring the languages for the target system

From the languages configured in the Automation Studio project, it is possible for the languages that should be transferred from the Automation Studio project to the target system for this specific configuration to be defined for each project configuration.

The "system language" and "fallback language" are also defined. The "system language" determines the language used to display the texts on the target by default. The "fallback language" defines the language used to display the texts if no texts are available in the "system language".

8.1.4 Text system in mapp View

In mapp View, localized texts from the text system can be used. Text files are used in order to be able to make your own texts in mapp View. If your own texts are entered in the system using text files, they will be made available for use by the text system.

Your own localized texts and the text system are the basis used to allow mapp View HMI applications to be displayed in different languages. mapp View provides the capability to switch languages for an HMI application.

8.2 Using the text system

Localized texts can be used in multiple locations in a mapp View HMI application. The following exercises demonstrate this in the form of a few examples.

8.2.1 Texts for manual navigation

The goal of this exercise is to configure the "NavigationButton" widget texts for manual navigation in German and English and then switch languages at runtime.

Exercise: Localize texts for manual navigation

To achieve this goal, the following steps must be carried out:

1) Set up project languages "de" and "en"
2) Create texts in "de" and "en"
3) Configure the localized texts on the widget
4) Configure the textconfig file
5) Add the Language Selector widget
6) Compile and transfer the project and display it in the browser

Set up project languages "de" and "en"

To define the languages to be used in a project, the language configuration file Project.language must be added from the Object Catalog in any position in the Logical View.

Text system / Project languages

First, a Project.language file must be added in the Logical View from the Object Catalog via drag-and-drop.

Double-clicking on the file opens a table editor. Languages English ("en"), German ("de") and French ("fr") already exist after inserting the .language file. The French language is not needed in our example and can therefore be removed.

The "design language" remains "English" as set by default.

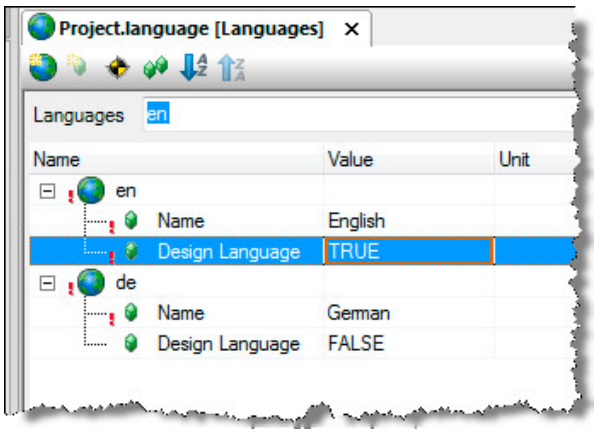


Figure 74: The project languages editor with languages DE and EN defined

Create texts in "de" and "en"

After the languages "de" and "en" have been specified, a localizable Texts.tmx file can be added from the "mappView/Resources/Texts" node in the Object Catalog and renamed as "VisualizationTexts".

The namespace "IAT" must be entered for the text file in order to use it in a mapp View HMI application. The namespace is entered in the "Namespace" tab using the "Property" context menu for the text file.

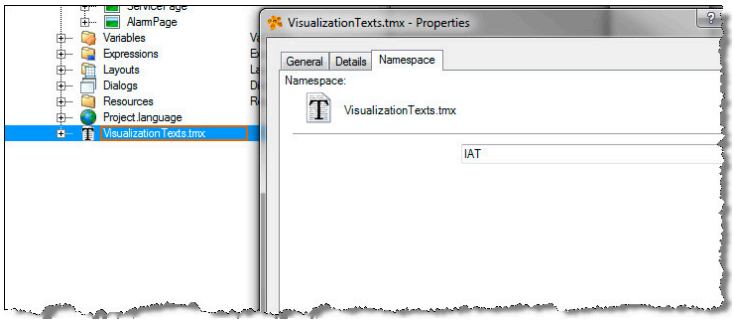


Figure 75: Defined NameSpace IAT in the localizable text file

Then the texts for the manual "NavigationButton" widget will be created in the table editor in English and German with a unique "ID".

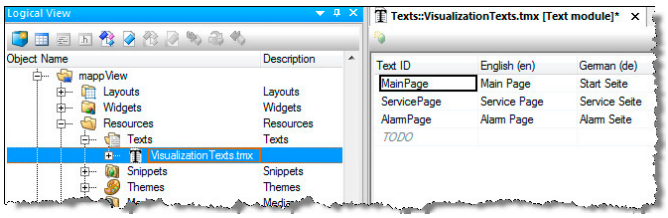


Figure 76: Localizable text file with defined texts

Configuring the localized texts

Once all texts have been created in German and English, they can be referenced in the "NavigationButton" widget for manual navigation.

This requires "ContentLeft" to be opened in the visual editor and a "NavigationButton" to be selected.

Then the text can be assigned to the widget in the Property window using 2 variants:

1.) The text ID can be entered directly with a "$" character as a prefix.

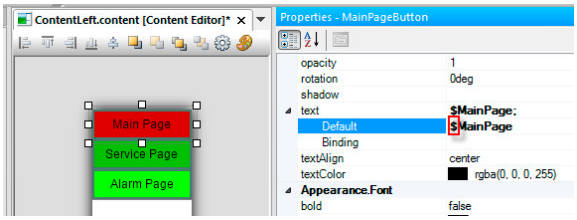


Figure 77: Assigning the manual navigation button widget localized texts with a "$" as prefix

2.) The text ID can be found using the "Assign Text" dialog box.

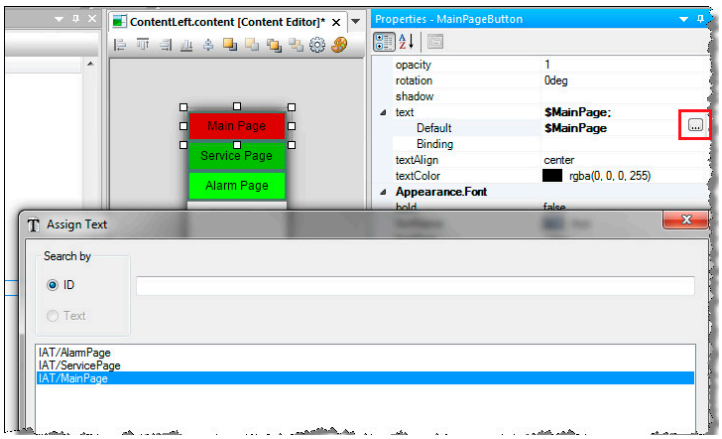


Figure 78: Assigning the manual navigation button localized texts using the "Assign Text" dialog box

Configure the textconfig file

For the texts that have been created to be transferred to the target system, a "TC.textconfig" file from the Object Catalog must be added to the "Text System" package in the Configuration View.

Then the "system language" and the "fallback language" are defined in the "TC.textconfig" file. It is also necessary to define which languages will be transferred to the target system (Target Language 1, etc.).

The TMX file with the texts that have been created must be selected under "TMX files for target" for text files to also be available on the target system.

Text system / Text system configuration

A complete definition looks like this:

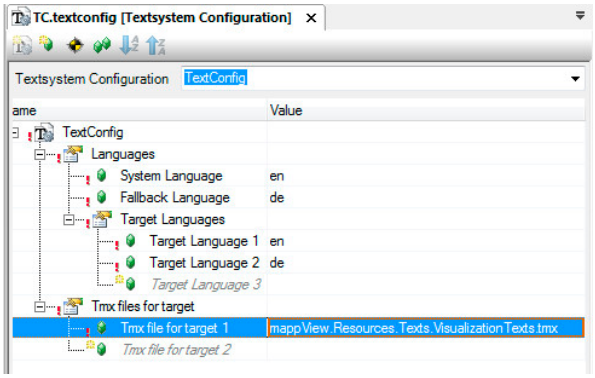


Figure 79: Configured TextConfig file

Adding the LanguageSelector widget

To be able to change the language in the HMI application, a LanguageSelector widget from the Object Catalog must also be added to the "MainPage" content. The LanguageSelector that has been added makes languages available in a drop-down list.

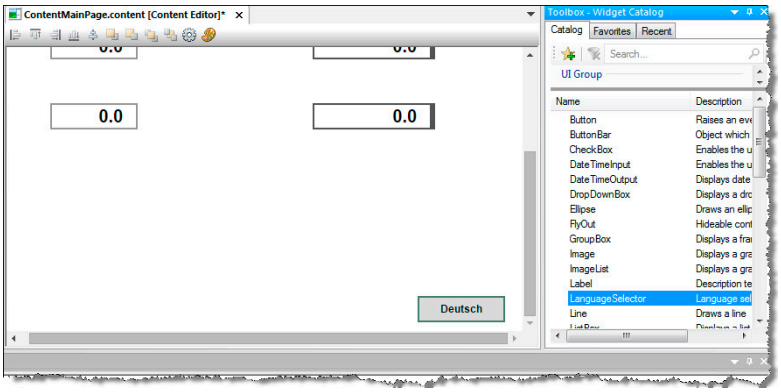


Figure 80: Language selector added to MainPage

Exercise: Compile/Transfer the project and display it in a browser

After the project languages and texts have been configured and the LanguageSelector added, the project can be compiled and transferred to ARsim.

The HMI application can be displayed using the following URL:

http://127.0.0.1:81/index.html?visuId=Training

Exercise: Compile the project and display it in a browser

1) Compile and transfer the project.
2) Open the browser and enter the URL

Expected result:

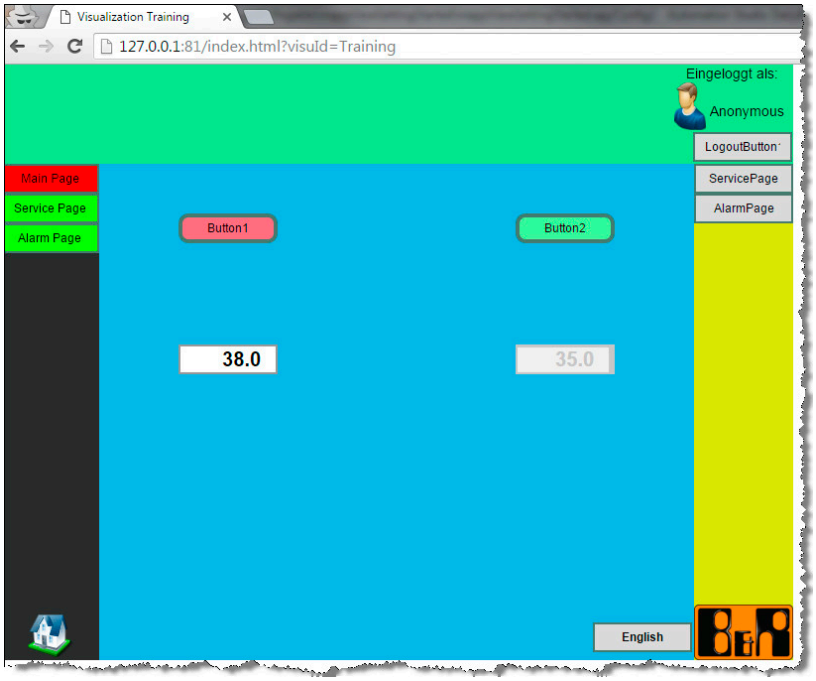


Figure 81: HMI application displayed in the browser with localized texts for the manual navigation button and a language selector

8.2.2 Texts for automatic navigation

The goal of this exercise is to configure texts for the automatic "NavigationButton" widget in German and English and then switch languages at runtime. The "NavigationButton" widget for automatic navigation uses texts that are configured on the pages to which they link.

Exercise: Localize the texts for automatic navigation

To achieve this goal, the following steps must be carried out:

1) Configure the TextID on the respective page
2) Compile and transfer the project and display it in the browser

Configure the text ID on the respective page

Since the project languages "de" and "en" and the texts with the unique text IDs in "de" and "en" have already been created and configured for the manual navigation buttons, these steps are no longer necessary in this exercise.

To localize the texts for the automatic navigation buttons and thus display them in the HMI application, the attribute "displayName" must be entered in the .page file for the respective page.

Double-clicking on the "MainPage.page" file opens the XML editor for the page.

Then the attribute "displayName=" must be set on the page. Specifying the relevant text "ID" starting with "$" in the "displayName" attribute allows the text for the "NavigationButton" widget to be referenced for automatic navigation.

The following attribute must be entered in the XML file for "MainPage.page:

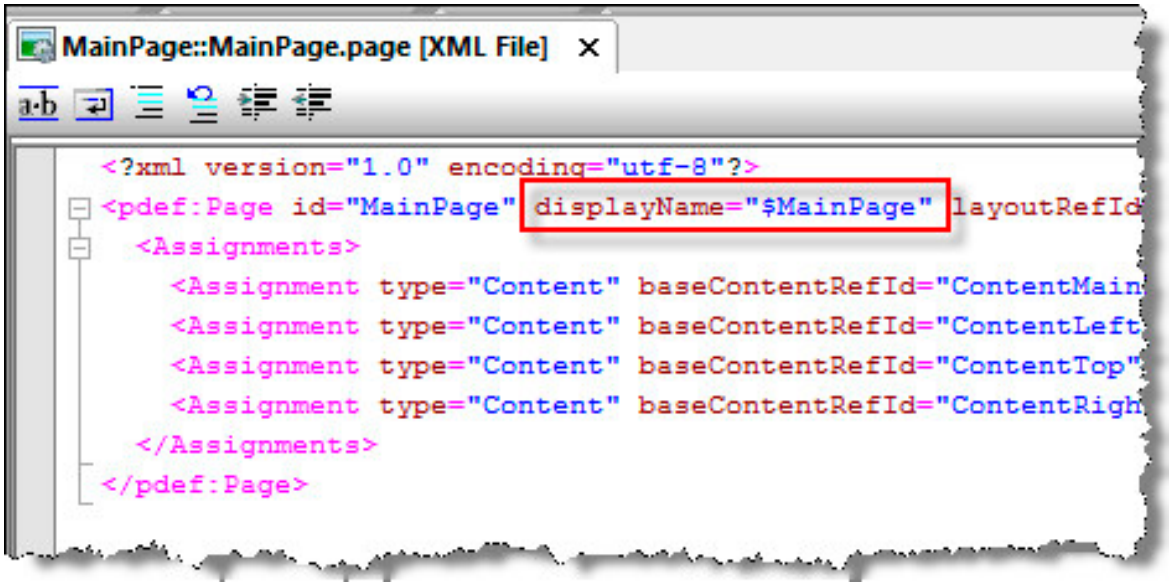


Figure 82: Configuring localized texts for automatic navigation buttons

Exercise: Enter text IDs for ServicePage and AlarmPage

Repeat the described procedure for the "ServicePage" and "AlarmPage".

Page	displayName=
ServicePage	"$ServicePage"
AlarmPage	"$AlarmPage"

Table 18: Assign the text IDs for automatic navigation

Exercise: Compile/Transfer the project and display it in a browser

After the TextIDs have been defined on the pages, the project can be compiled and transferred to ARsim.

The HMI application can be displayed using the following URL:

http://127.0.0.1:81/index.html?visuId=Training

Exercise: Compile the project and display it in a browser

1) Compile and transfer the project.
2) Open the browser and enter the URL

Expected result

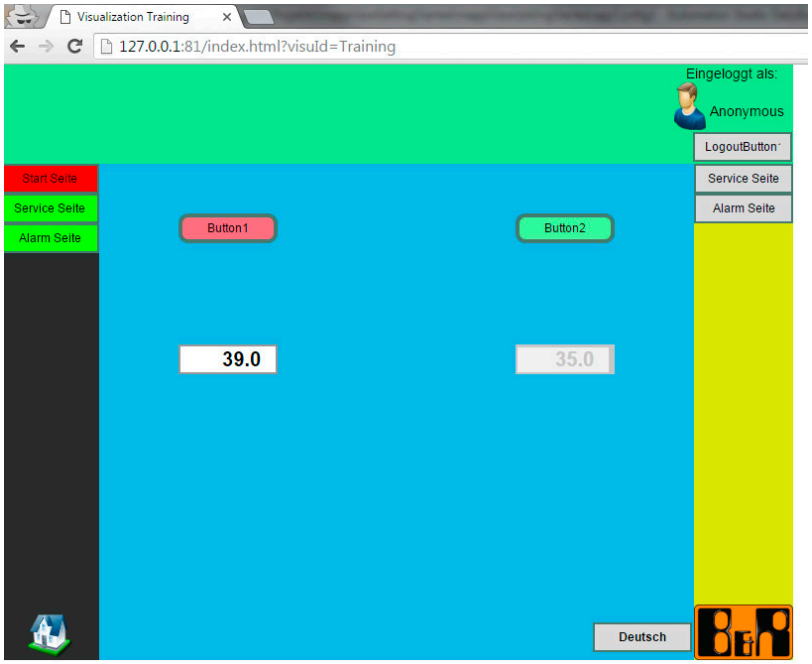


Figure 83: HMI application displayed in the browser with language switched to German

8.2.3 Texts for LogoutButton and LabelLoginInfo widget

The goal of this exercise is to localize the texts for the "Label" widget with information about logged in users and the text for the "LogoutButton".

Exercise: Localize widget texts

The texts for the "Label" widget with the information about logged in users and the text for the logout button should be localized and then switched between German and English using the "Language Selector".

Expected result:

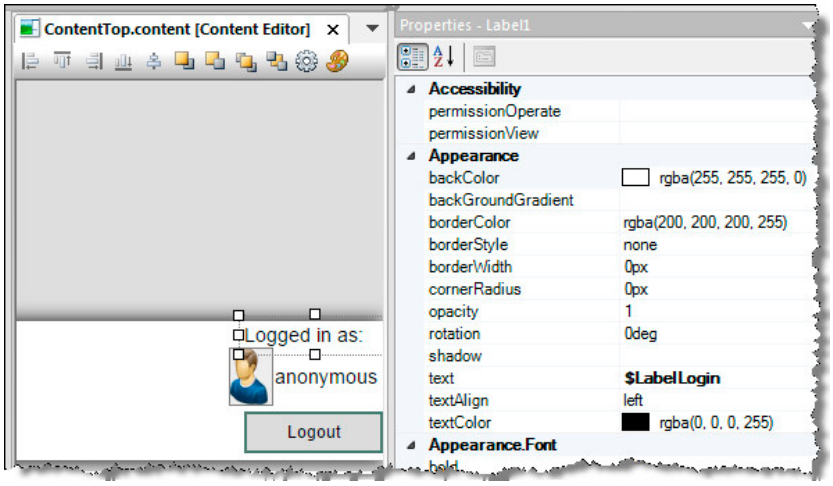


Figure 84: Localized texts for the label and the logout button

8.3 Overview – Unit system

Automation Studio provides users an integrated unit system with automatic unit conversion. For localization of units, users have the "metric", "imperial" and "imperial-us" systems of measurement and more than 1400 units available to them.

With an OPC UA node, the engineering unit can be configured in addition to the EU range. The engineering unit for an OPC UA node specifies the physical unit to be used when interpreting its value.

With a widget, the unit used to display the value for a bound OPC UA node can be defined for the selected system of measurement.

If the unit used for the value prepared by the automation application is known as well as the unit that should be used to display the value, then the system automatically converts the value to the desired unit.

All units as well as the common code (e.g. degrees Celsius = CEL) to be entered in the "unit" property for each unit are described in Automation Help.

Unit system – Available standard units

8.4 Using the unit system

8.4.1 Displaying a value with a unit and switching the system of measurement

To display the unit of a value from the automation application in the HMI application, it is necessary to bind the unit in addition to the numeric value.

Exercise: Display the unit "degrees Celsius" in the NumericOutput widget.

The goal of this exercise is to define the unit degrees Celsius (°C) for OPC UA nodes "CurrentTemperature" and "SetTemperture" and then display them in another NumericOutput/NumericInput widget and to be able to use the "MeasurementSystemSelector" widget to switch between °C and °F.

The following steps must be carried out here:

1) Configure OPC UA node Engineering units as "degrees Celsius"
2) Add NumericOutput and select node binding
3) Configure unit switching
4) Add "MeasurementSystemSelector" widget
5) Repeat the process for SetTemperature and NumericInput
6) Compile the project, transfer it and display it in the HMI application

Append "degrees Celsius" unit to OPC UA node Engineering

In the OPC UA default view, a unit can be configured for an OPC UA node after it has been enabled.

Add unit "degree Celsius" from the Object Catalog via drag-and-drop in the Properties window for the OPC UA default view after selecting the "Engineering Unit" node.

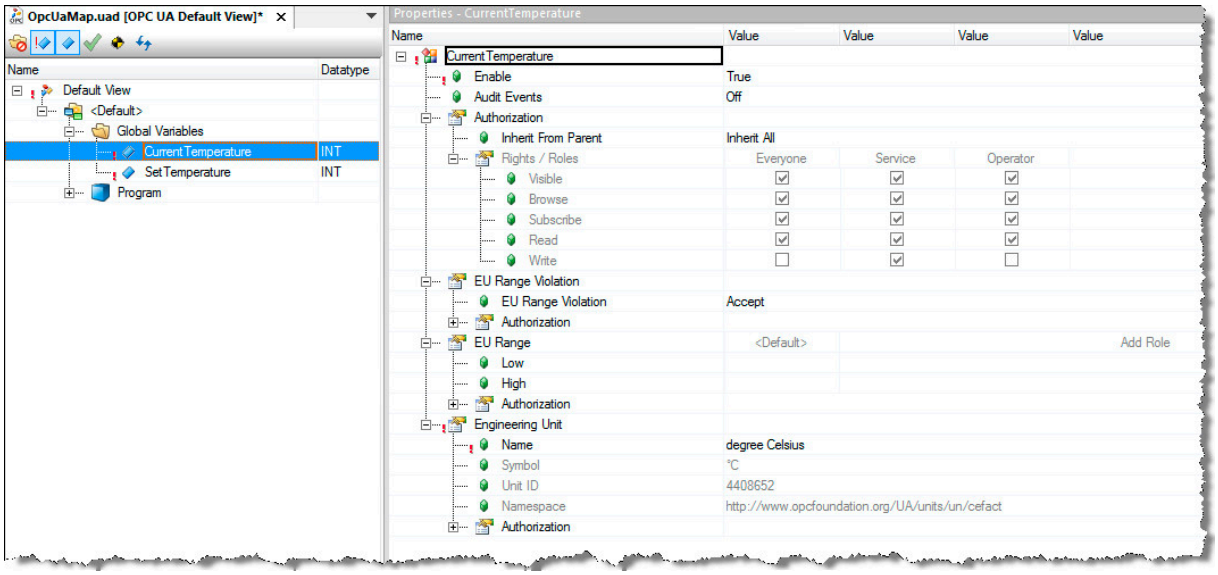


Figure 85: Appending "degrees Celsius" unit to the OPC UA node CurrentTemperature

HMI application/mapp View/Creating an HMI application/Connecting widgets to data/Displaying values and units

Adding the NumericOutput widget

After another "NumericOutput" widget has been added to the "MainPage" content and increased in size, node binding must be selected instead of value binding so the appropriate unit can also be displayed in the "NumericOutput" widget.

Procedure:

- Add a numeric output, change the size and select it
- Open the Select Variable dialog box

Selecting the OPC UA node (and not its values) sets up node binding.

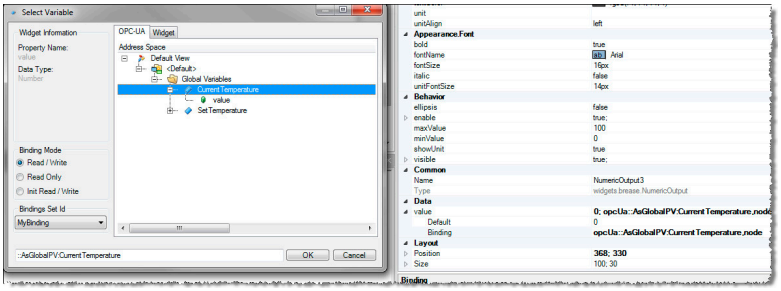


Figure 86: Select node binding in the Binding dialog box for the ActualTemperature node

The binding that has been set up successfully is shown in the Properties window with the extension ".node". Clicking on "OK" applies the new binding.

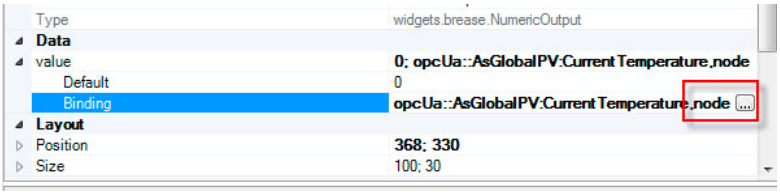


Figure 87: Node binding for the CurrentTemperature OPC UA node

Configuring unit switching

In order for the unit to also be displayed in the NumericOutput widget, the corresponding "unit" property must also be configured. The unit that should be displayed must be specified for each system of measurement ("metric", "imperial", "imperial-us"). The unit is specified using its common code. Information

about the common code for supported units is provided in Automation Help. Conversion of the value from the unit provided by the automation application to the unit in which the value is to be displayed is performed automatically.

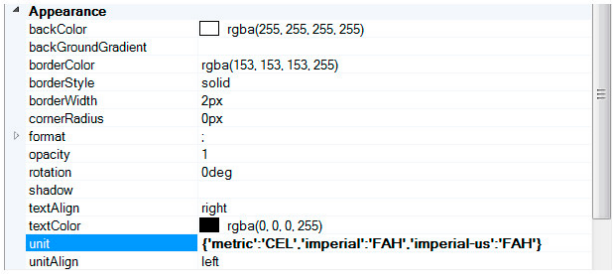


Figure 88: Defining the unit properties

The unit properties must always be specified as follows:

{'metric':'<CommonCode>','imperial':'<CommonCode>','imperial-us':'<CommonCode>'}

For our example, the unit string is as follows:

{'metric':'CEL','imperial':'FAH','imperial-us':'FAH'}

Adding the MeasurementSystemSelector widget

After the unit for the OPC UA node has been configured and the units that should be displayed have been defined for the system of measurement, the possibility to switch the system of measurement must be configured. To switch the units, the "MeasurementSystemSelector" widget from the Object Catalog will be placed on the "MainPage" contents.

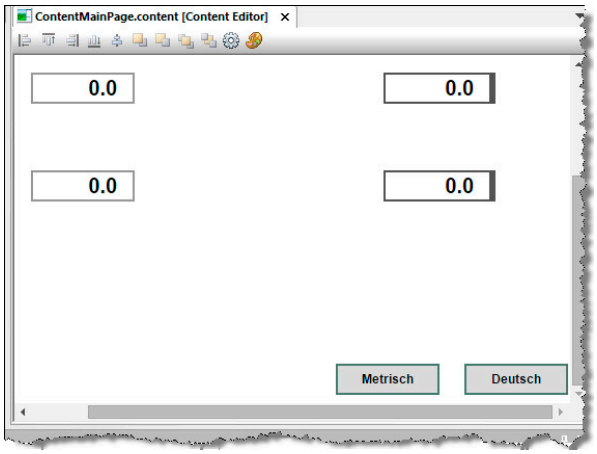


Figure 89: Placing MeasurementSelector on MainPage

Exercise: Display the SetTemperature node with units in the NumericInput widget

Before the project is compiled, OPC UA node "SetTemperature" should also be displayed with the unit °C in another "NumericInput" widget that can also be switched to °F.

The following tasks must be completed:

1) Configure OPC UA node (SetTemperature) units as "degrees Celsius"
2) Add NumericInput widget and select node binding
3) Configure unit switching
4) Compile the project and display it in a browser

Compile the project, transfer it and display it in the HMI application

After the unit of measure has been assigned to the OPC UA node, the project can be compiled and transferred to ARsim.

The HMI application can be displayed using the following URL:

http://127.0.0.1:81/index.html?visuId=Training

Exercise: Compile the project and display it in a browser

1) Compile and transfer the project.
2) Open the browser and enter the URL

Expected result

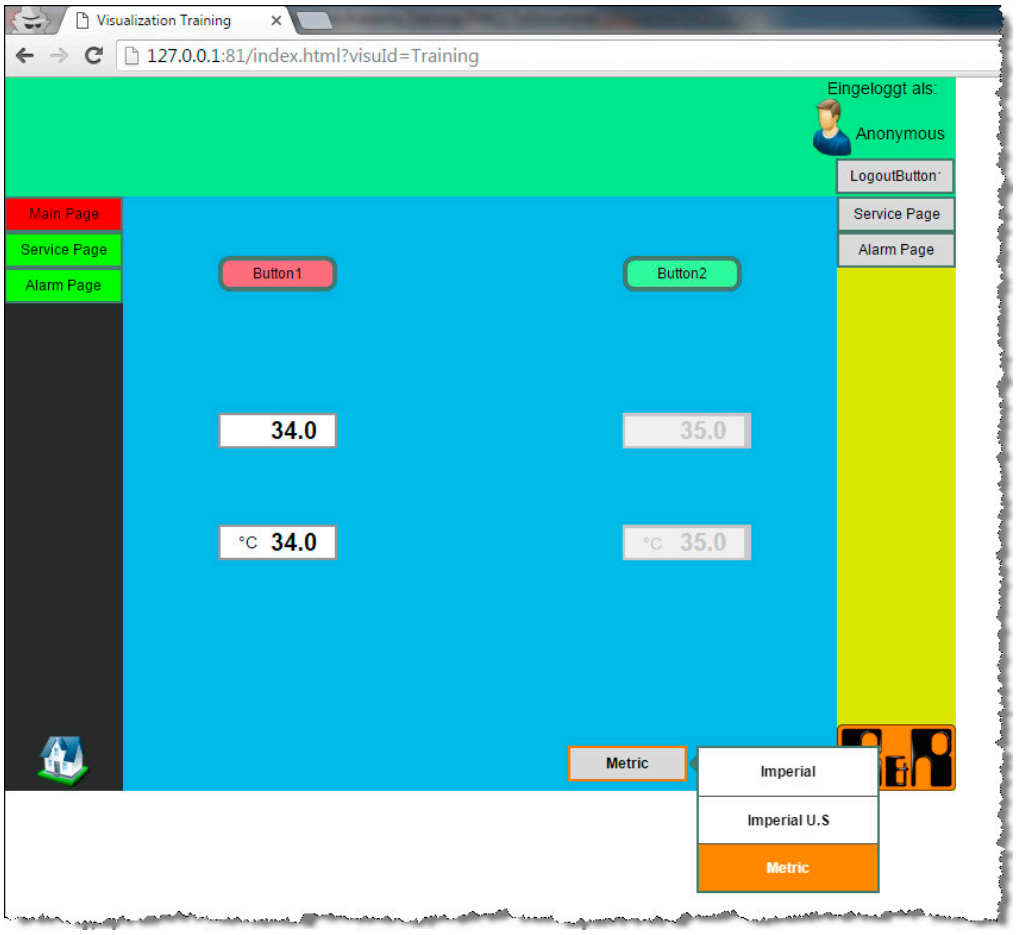


Figure 90: HMI application displayed in a browser with the corresponding unit

9 Events and actions

For configuring the behavior of the HMI application, we have already become familiar with the possibilities for data binding and navigation. Navigation determines how users can move between the various pages in the HMI application. Data binding determines which data is exchanged between the automation application and the HMI application and in which direction.

For some HMI applications, these options may not be sufficient. mapp View provides events and actions to individually configure the behavior of an HMI application.

Events are occurrences that can be used to trigger responses. An action is a response triggered by an event. The combination of events and subsequent actions is referred to as event handling.

9.1 Overview – Events

An event is an occurrence that can be used to trigger a response. mapp View provides HMI application developers several types of events.

OPC UA events

mapp View defines events to provide information about changes to values in the automation application. For this purpose, mapp View provides event type "opcUa.Event" with the name "ValueChanged".

Widget events

Widget events provide information about occurrences in specific widget instances. Different types of widget can provide information about different occurrences in the form of events. An example of this is the "Click" event for the "Button" widget.

Session events

mapp View defines events in order to detect occurrences in a client session. Session events are covered the mapp View Advanced training module.

9.2 Overview – Actions

An action is a response triggered by an event.

Actions are grouped together by provider. The following action providers exist in mapp View:

- OPC UA
- Session
- Client
- Widgets

Each provider defines one or more groups of actions.

Provider "OPC UA" provides group "opcUa.NodeAction" and provider "Widgets" provides group "widgets.brease.<WidgetType>".

Providers "Session" and "Client" are topics of the mapp View Advanced training course. In this training course, we will only cover providers "OPC UA" and "Widgets".

Each group defines one or more actions.

HMI application/mapp View/mapp View development environment/Events and action/Actions

9.2.1 OPC UA actions

For OPC UA, actions are available that have an effect on OPC UA nodes.

HMI application/mapp View/mapp View development environment/Events and action/Actions

9.2.2 Widget actions

For widgets, actions are available that have an effect on instances of widgets.

There are specific actions for each widget type. To find out more about actions available for a widget type, see the documentation for that widget.

HMI application/mapp View/mapp View development environment/Events and action/Actions

9.3 Using event and actions

9.3.1 Setting a value

The goal of this exercise is to reset the setpoint for the "SetTemperature" OPC UA node displayed in the "NumericInput" widget to the default value (35) by clicking on a button.

Exercise: Reset the setpoint for the OPC UA node by clicking a button.

To achieve this goal, a suitable event will be used and an appropriate action will be configured as a response to this event.

The following steps must be carried out:

1) Add and configure a new Button widget
2) Add EventBinding file and define action and events
3) Reference the event binding file in the .vis file
4) Compile the project, transfer it and display it in the HMI application

HMI application/mapp View/mapp View development environment/Events and action/Events

Add and configure a new button

First, a new "Button" widget will be added from the Object Catalog to "MainPage" via drag-and-drop and the CommonName for the button widget will be changed to "SetToDefault" for easier identification. Then the button text will be localized so it can be displayed in German and English.

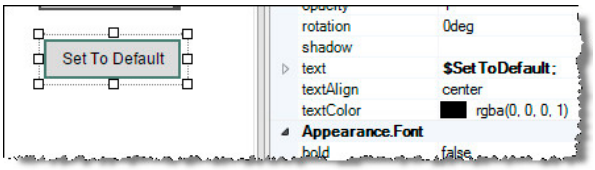


Figure 91: New button added and configured

Add EventBinding file and define action and events

The events and the subsequent actions will be defined in the event binding file. In Automation Help, all possible parameters used for the various strings and the respective widgets (e.g. event type "Click") are defined.

HMI application/mapp View/Widgets

After selecting the mapp View node in the Configuration View, an "event binding file" can be added from the Object Catalog. Double-clicking on the file opens the XML editor and then the unique "ID" (="Event-Binding") can be entered as the first step.

In the next step, the event that will be triggered when you click on the "Button" widget will be defined as the source.

First, the event type that can be seen in the properties for each widget is defined, which is composed as follows:

widgets."Namespace"."WidgetTyp".Event

Then you need to specify which widget instance (CommonName) is being referenced to and the content in which it is located.

The widget instance will be specified with the widgetRefId="" attribute.

The content will be specified with the contentRefId="" attribute.

Lastly, the event that can trigger a response will be specified with the event="" attribute.

The finished entry for the example when clicking on a "Button" widget looks as follows:

```
<Source xsi:type="widgets.brease.Button.Event"
    contentRefId="ContentMainPage" widgetRefId="SetToDefault"
event="Click"  />
```

After the event has been defined, the subsequent action must be defined. In Automation Help, possible parameters that can be selected are defined.

HMI application/mapp View/mapp View development environment/Events and action/Actions/OPC UA actions

The entry for the action consists of the "opcUaNodeAction" attribute and the global variable (SetTemperature) that should be changed.

The definition then looks like this:

```
<Action>
<Target xsi:type="opcUa.NodeAction" refId="::AsGlobalPV:SetTemperature">
```

Information about configuring the refId for local and global variables can be found in the help system in section "Addressing OPC UA variable".

In the last step, the action that should be executed is defined in the element <Method>.

The string consists of the possible action (NodeAction) and the value that should be set, and looks like this:

```
<Method xsi:type="opcUa.NodeAction.SetValueNumber" value="35" />
```

The entire event binding looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<EventBindingSet id="EventBinding"
 xmlns="http://www.br-automation.com/iat2014/eventbinding/v2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Bindings>
    <EventBinding>
      <Source xsi:type="widgets.brease.Button.Event"
        contentRefId="ContentMainPage" widgetRefId="SetToDefault"
        event="Click" />
    <EventHandler>
       <Action>
         <Target xsi:type="opcUa.NodeAction"
        refId="::AsGlobalPV:SetTemperature ">
```

```
          <Method xsi:type="opcUa.NodeAction.SetValueNumber" value="35" />
           </Target>
           </Action>
    </EventHandler>
    </EventBinding>
  </Bindings>
</EventBindingSet>
```

Reference event binding in the .vis file

After the event and the action have been defined, the entered EventBindingSet Id (=EventBinding) must be referenced in the .vis file so that event binding also is taken into account.

Double-clicking on the .vis file allows the event binding element to be commented out and the "ID" entered.

Exercise: Compile the project, transfer it and display it in the HMI application

After the event and action have been defined, the project can be compiled and transferred to ARsim.

The HMI application can be displayed using the following URL:

http://127.0.0.1:81/index.html?visuId=Training

Test the HMI application with various roles

The value of the numeric input widget will only be set if all defined parameters (OPC UA default view) and permissions for the logged-in role/user are met.

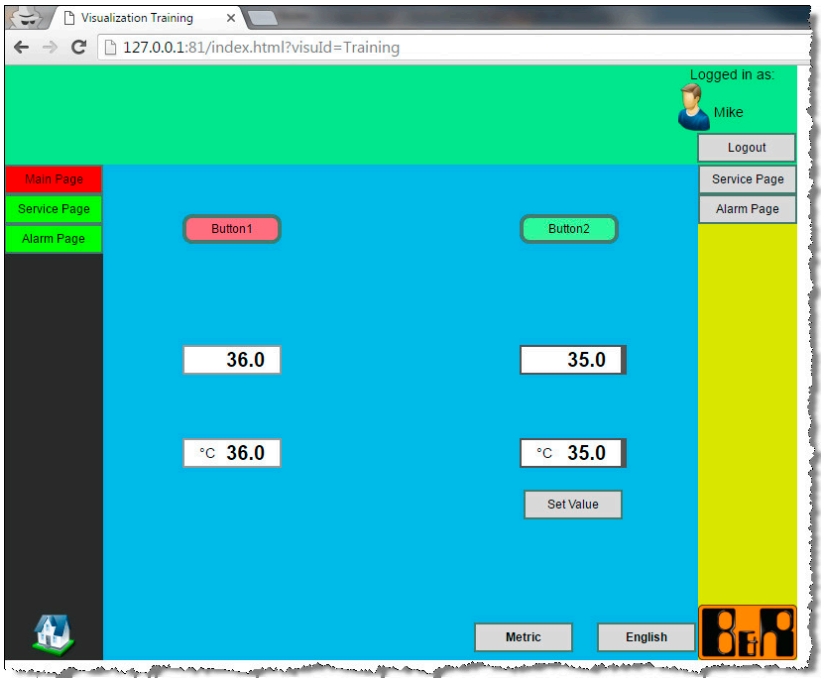


Figure 92: HMI application in the browser with the user "Mike" logged in

9.3.2 Displaying and hiding a widget

The goal of this exercise is to show an image in the "MainPage" content (action) when a certain value has been reached on the "CurrentTemperature" OPC UA node (event).

Exercise: Display and hide an image when a value is reached.

To achieve this goal, an image will be added on the MainPage content and its visibility set to "false". Then, the event when reaching the value and the action that defined displaying the image will be defined in EventBinding.

The following steps must be carried out here:

1) Add and configure the image
2) Define the event and action for displaying the image in event binding
3) Define the event and action for hiding the image in event binding
4) Compile the project, transfer it and display it in the HMI application

Add and configure the image

First, an "Image" widget will be added from the Object Catalog and placed on the "MainPage" content via drag-and-drop. Then the "warning" image, which has already been stored, will be referenced in the Image widget. Afterwards, the "visible" property to must be set to "false" in order for the image to only become visible when triggered by the event. The CommonName for the widget must be changed to "ImageWarning" for easier identification.

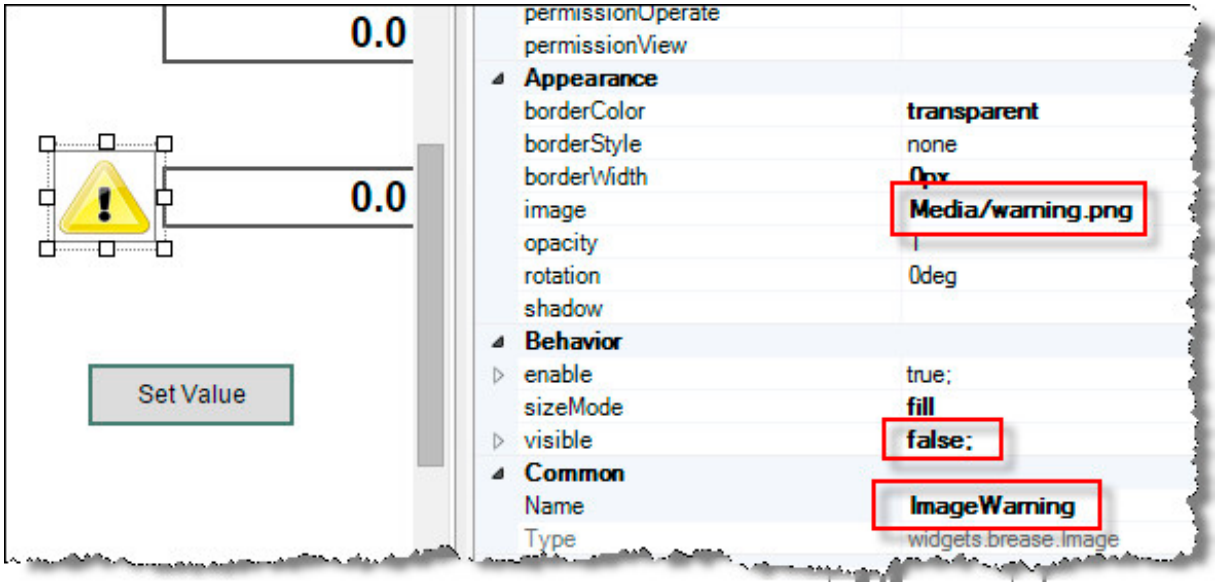


Figure 93: Image widget properties

Defining the event and action for displaying the image

The events and the subsequent actions will be defined in the event binding file.

The EventBinding file has already been added in the "Setting a value" exercise and referenced in the .vis file.

First, the event is defined for when the value "45" has been reached on the "CurrentTemperature" OPC UA node. This event is for "OPC UA". The event type is therefore "opcUA.Event". The entry for the EventBinding element <Source> looks as like this:

```
<Source xsi:type="opcUa.Event"
refId="::AsGlobalPV:CurrentTemperature" event="ValueChanged"/>
```

In the String element <Source>, the "refId" of the corresponding OPC UA node and the "ValueChanged event" will be specified in addition to the xsi:type.

EventHandler describes the response to the event. For EventHandler, a condition can be defined that must be fulfilled in order for the configured actions to be executed.

```
<EventHandler condition="newValue &gt; 44">
```

Now the action will be defined as a response to the event.

The element defines the object to which the action is applied. In our example, the action is applied to a widget that has been placed in content.

```
<Target xsi:type="widgets.brease.Image.Action"
widgetRefId="ImageWarning" contentRefId="ContentMainPage">
```

For the "visible" property of the image to change from "false" to "true", the "value" attribute will be set to "true in the "Method" element in addition to the xsi:type.

```
<Method xsi:type="widgets.brease.Image.Action.SetVisible"
value="true" />
```

The finished EventBinding looks like this:

```
<EventBinding>
    <Source xsi:type="opcUa.Event"
    refId="::AsGlobalPV:CurrentTemperature" event="ValueChanged"/>
    <EventHandler condition="newValue &gt;= 45">
        <Action>
        <Target xsi:type="widgets.brease.Image.Action"
        contentRefId="ContentMainPage" widgetRefId="ImageWarning">
            <Method xsi:type="widgets.brease.Image.Action.SetVisible"
        value="true" />
            </Target>
        </Action>
    </EventHandler>
</EventBinding>
```

Defining the event and action for hiding the image

For the "warning" image to be hidden again when the value drops below "45", another event and subsequent action must be defined.

To hide the image, the entire EventBinding for displaying the image will be copied and the condition parameters changed from "&GT" to "&LT"; "45" and the "value" for the "SetVisible" action changed to "false".

Final result:

```
<EventBinding>
    <Source xsi:type="opcUa.Event"
    refId="::AsGlobalPV:CurrentTemperature" event="ValueChanged"/>
```

```
    <EventHandler condition="newValue &lt; 45">
    <Action>
       <Target xsi:type="widgets.brease.Image.Action"
         contentRefId="ContentMainPage" widgetRefId="ImageWarning">
    <Method xsi:type="widgets.brease.Image.Action.SetVisible"
          value="false" />
        </Target>
    </Action>
    </EventHandler>
</EventBinding>
```

Exercise: Compile the project, transfer it and display it in the HMI application

After the events and actions have been defined, the project can be compiled and transferred to ARsim.

The HMI application can be displayed using the following URL:

http://127.0.0.1:81/index.html?visuId=Training

Expected result:

Figure 94: HMI application in the browser with image shown

10 Summary

After completing this training module, a mapp View HMI application with multiple HMI pages can be created. Navigation between the HMI pages can be configured for both manual and automatic navigation. Displaying values from the automation application can also be configured as well as entering data. When doing so, value range limits defined in the automation application can be automatically checked in the HMI application. Value range limits defined differently for specific roles can also be checked. To allow role-specific behavior of the HMI application, the user role system in Automation Studio can be used for HMI user authentication. Widgets and media can be modified using themes and styles in order to adapt the optical appearance of the HMI application to individual needs. Texts and units can be adapted to linguistic and cultural conditions via localization. As a conclusion to this training module, the definition of actions as responses to events has been covered.

Figure 95: mapp View glasses

Seminar participants can therefore now create simple HMI applications using mapp View.

Creating more complex HMI applications is covered int he "mapp View Advanced" training module.

Seminars and Training Modules

At the Automation Academy, you'll develop the skills you need in no time!
Our seminars make it possible for you to improve your knowledge in the field of automation engineering.

Automation Studio seminars and training modules

Programming and configuration	Diagnostics and service
SEM210 – Basics SEM246 – IEC 61131-3 programming language ST* SEM250 – Memory management and data storage SEM410 – Integrated motion control* SEM441 – Motion control: Electronic gears and cams** SEM480 – Hydraulics** SEM1110 – Axis groups and path-controlled movements** SEM510 – Integrated safety technology* SEM540 – Safe motion control*** SEM610 – Integrated visualization*	SEM920 – Diagnostics and service for end users SEM920 – Diagnostics and service with Automation Studio SEM950 – POWERLINK configuration and diagnostics* If you do not happen to find a seminar on our website that suits your needs, keep in mind that we also offer customized seminars that we can set up in coordination with your sales representatives: SEM099 – Individual training day Please visit our website for more information ****: **www.br-automation.com/academy**

Overview of training modules	
TM210 – Working with Automation Studio TM213 – Automation Runtime TM223 – Automation Studio Diagnostics TM230 – Structured Software Generation TM240 – Ladder Diagram (LD) TM241 – Function Block Diagram (FBD) TM242 – Sequential Function Chart (SFC) TM246 – Structured Text (ST) TM250 – Memory Management and Data Storage TM400 – Introduction to Motion Control TM410 – Working with Integrated Motion Control TM440 – Motion Control: Basic Functions TM441 – Motion control: Electronic gears and cams TM1110 – Integrated Motion Control (Axis Groups) TM1111 – Integrated Motion Control (Path Controlled Movements) TM450 – Motion Control Concept and Configuration TM460 – Initial Commissioning of Motors TM500 – Introduction to Integrated Safety TM510 – Working with SafeDESIGNER TM540 – Integrated Safe Motion Control	TM600 – Introduction to Visualization TM610 – Working with Integrated Visualization TM611 – Working with mapp View TM630 – Visualization Programming Guide TM640 – Alarm System, Trends and Diagnostics TM670 – Advanced Visual Components TM920 – Diagnostics and service TM923 – Diagnostics and Service with Automation Studio TM950 – POWERLINK Configuration and Diagnostics TM280 – Condition Monitoring for Vibration Measurement TM480 – The Basics of Hydraulics TM481 – Valve-based Hydraulic Drives TM482 – Hydraulic Servo Pump Drives TM490 – Printing Machine Technology In addition to a printed version, our training modules are also available on our website for download as electronic documents (login required): Visit our website for more information: **www.br-automation.com/academy**

Process control seminars and training modules

Process control standard seminars	Process control training modules
SEM841 – Process Control Training: Basic 1 SEM842 – Process Control Training: Basic 2 SEM890 – Advanced Process Control Solutions	TM800 – APROL System Concept TM810 – APROL Setup, Configuration and Recovery TM811 – APROL Runtime System TM812 – APROL Operator Management TM813 – APROL Web Portal TM820 – APROL Solutions TM830 – APROL Project Engineering TM835 – APROL ST-SFC Configuration TM840 – APROL Parameter Management and Recipes TM850 – APROL Controller Configuration and INA TM860 – APROL Library Engineering TM865 – APROL Library Guide Book TM870 – APROL Python Programming TM880 – APROL Reporting TM890 – The Basics of LINUX

* SEM210 - Basics is a prerequisite for this seminar.
** SEM410 - Integrated motion control is a prerequisite for this seminar.
*** SEM410 - Integrated motion control and SEM510 - Integrated safety technology are prerequisites for this seminar.
**** Our seminars are listed in the Academy\Seminars area of the website. Seminar titles may vary by country. Not all seminars are available in every country.