

Lycia Development Suite



Lycia WindowBuilder designer Guide

Version 1.1 – August 2014

Revision number 1.00

Lycia WindowBuilder designer

Reference Guide

Nov 2013

Part Number: 007-101-001-014

Svitlana Ostnek / Elena Krivtsova / Olga Gusarenko

Technical Writers

Last Updated Aug 2014

Lycia WindowBuilder designer Reference Guide

Copyright 2006-2014 Querix (UK) Limited. All rights reserved.

Part Number: 007-101-001-014

Published by:

Querix (UK) Limited. 50 The Avenue, Southampton, Hampshire,
SO17 1XQ, UK

Publication history:

August 2012: Beta edition

November 2013: Updated for Lycia 2.2

Last Updated:

August 2014

Documentation written by:

Svitlana Ostnek / Elena Krivtsova / Olga Gusarenko

Notices:

The information contained within this document is subject to change without notice. If you find any problems in the documentation please submit your comments to documentation@querix.com. No part of this document may be reproduced or transmitted, in any form or by any means, electronic or mechanical, for any purpose without the express permission of Querix (UK) Ltd.

Other products or company names used within this document are for identification purposes only, and may be trademarks of their respective owners.

Table of Contents

ABOUT THIS GUIDE	10
WHO SHOULD READ THIS REFERENCE GUIDE?	10
ORGANISATION OF THE GUIDE	11
FORMS CONVERSION	12
CONVERTING OLD LYCIA FORMS.....	12
CONVERTING THIRD-PARTY FORMS	12
WINDOWBUILDER DESIGNER LAYOUT	14
EDITOR AREA.....	16
PALETTE	17
FORM STRUCTURE	20
PROPERTIES	21
ADDING PROPERTY ELEMENTS AND LISTS.....	22
GENERAL PROPERTIES	24
<i>Size and Location Group</i>	24
<i>Text and Image Group</i>	27
<i>Appearance Group</i>	28
<i>Events Group</i>	31
CONTAINER PROPERTIES	36
<i>Ungrouped properties</i>	36
WIDGET PROPERTIES.....	36
<i>Text and Image group</i>	36
<i>Size and Location group</i>	38
<i>Appearance group</i>	38
<i>Ungrouped properties</i>	39
FORM ELEMENTS	41
FORM OBJECT.....	41
ROOT CONTAINER	41
MAIN TOOLBAR.....	42
FORM MENU	46
SCREEN RECORDS	48
<i>Default Screen Records</i>	49
<i>Adding Screen Records</i>	49



BORDER PANEL CONTAINER.....	51
<i>Padding</i>	52
COORD PANEL CONTAINER.....	53
GRID PANEL CONTAINER.....	53
<i>GridItemLocation Property.</i>	55
<i>GridLength</i>	55
<i>Padding</i>	56
GROUPBOX CONTAINER.....	57
<i>Font</i>	57
<i>ForeColor</i>	57
<i>Padding</i>	57
<i>Title</i>	57
<i>TitleJustification</i>	58
SCROLLVIEWER CONTAINER.....	58
<i>Padding</i>	59
<i>HorizontalScrollBarVisibility/ VerticalScrollBarVisibility</i>	59
STACK PANEL CONTAINER.....	59
<i>Orientation</i>	60
<i>Padding</i>	61
TAB PANEL CONTAINER.....	61
<i>Enable Multiline</i>	62
<i>Padding</i>	62
<i>Tab Page Placement</i>	62
<i>OnSelectedTabPageChanged Event</i>	62
TABPAGE CONTAINER.....	63
<i>Font</i>	63
<i>ForeColor</i>	63
<i>Image</i>	64
<i>OnSelectTabPage Event</i>	64
TABLE CONTAINER.....	64
<i>CellPadding</i>	64
<i>GridColor</i>	64



<i>HorizontalScrollbarVisibility</i>	64
<i>Identifier</i>	64
<i>Is Multiselect</i>	65
<i>Is Protected</i>	65
<i>Is Virtual</i>	65
<i>Margin</i>	65
<i>Padding</i>	65
<i>RowCount</i>	65
<i>RowHeight</i>	65
<i>RowResizable</i>	65
<i>TextAlignment</i>	65
<i>VerticalScrollbarVisibility</i>	65
<i>Visible</i>	65
<i>VisibleToolbarGroups</i>	66
TABLE COLUMN	66
<i>Font</i>	69
<i>ForeColor</i>	70
<i>ColumnLength</i>	70
<i>Text</i>	70
<i>Aggregate properties</i>	70
<i>TotalAggregate properties</i>	71
<i>IsGroupable</i>	72
<i>IsFilterable</i>	74
<i>IsSortable</i>	74
<i>Movable</i>	74
TREE TABLE CONTAINER	75
<i>ColumnId and ColumnParentId properties</i>	76
<i>ColumnEdit</i>	79
<i>ColumnExpanded</i>	80
<i>ColumnImage</i>	81
<i>ColumnIsNode</i>	83
<i>ImageCollapsed</i>	84



<i>ImageExpanded</i>	84
<i>ImageLeaf</i>	84
FORM WIDGETS	85
BUTTON WIDGET.....	85
<i>Accelerators</i>	85
<i>IsToggleButton</i>	85
BROWSER WIDGET	85
<i>Text</i>	86
CANVAS WIDGET	86
CHECKBOX WIDGET	86
<i>Checked Value</i>	87
<i>Unchecked Value</i>	87
COMBOBOX WIDGET	88
<i>Initializer</i>	88
<i>Combo Box Items</i>	88
<i>Editable</i>	89
<i>MaxDropDownheight</i>	89
<i>MaxDropDownWidth</i>	89
<i>OnDropdown Event</i>	90
<i>OnDropDownClose Event</i>	90
<i>OnSelectedItemChange Event</i>	90
CALENDAR WIDGET	90
<i>Century</i>	90
FUNCTION FIELD WIDGET	90
<i>Enable</i>	91
LABEL WIDGET	91
<i>Is Dynamic</i>	91
LISTBOX WIDGET	91
<i>List Box Items</i>	92
<i>EnableMultiselection</i>	92
RADIOBUTTON WIDGET	93
<i>GroupIdentifier</i>	93



RADIOBUTTONLIST WIDGET	93
<i>Orientation</i>	94
PROGRESS BAR WIDGET	94
<i>Step property</i>	95
SLIDER WIDGET	95
<i>MajorTick</i>	95
<i>MinorTick</i>	95
<i>ShowTicks</i>	95
SCROLLBAR WIDGET	95
<i>LargeStep</i>	95
<i>SmallStep</i>	96
<i>OnScroll Event</i>	96
SPINNER WIDGET	96
<i>Step</i>	96
TEXT FIELD WIDGET	96
<i>Editor</i>	96
TEXT AREA WIDGET	96
<i>UseTabs</i>	97
TIMEEDITFIELD WIDGET	97
WEBCOMPONENT WIDGET	97
<i>ComponentType property</i>	98
<i>ComponentProperties property</i>	98



About This Guide

Who should read this Reference Guide?

This reference guide is intended for use by anybody who is involved in 4GL development work, for applications that will include working with graphical forms.

This reference guide should provide the detail needed to create, convert and configure graphical forms, set up the properties and behaviour of their elements.

Any comments about how this reference guide could be changed in order to improve any aspects of the usability of the documentation should be addressed to the [technical writer](#).

This Guide is a work-in-progress and does not reflect all of the latest changes to Lycia WindowBuilder designer. .



Organisation of the Guide

The Guide consists of six chapters. The first four chapters give the general idea of the WindowBuilder designer and its tools, the last two chapters are devoted to detailed description of the form objects and properties.

Chapter 1. Forms Conversion

The chapter describes the ways to convert earlier Lycia and third-party forms into Lycia forms of new format.

Chapter 2. WindowBuilder designer Layout

The chapter describes the main views and tools available in the WindowBuilder designer.

Chapter 3. Editor View

The chapter gives a detailed description of the Editor View.

Chapter 4. Form Structure view

The chapter gives a detailed description of the Form Structure view.

Chapter 5. Properties

The chapter gives the idea of properties that can be applied to form objects, their organization in the Properties view, the idea and structure of property groups, and the description of the most common properties that can be applied to several or most of form objects.

Chapter 6. Form Objects

The chapter describes the objects that can be added to the form, as well as object-specific properties.

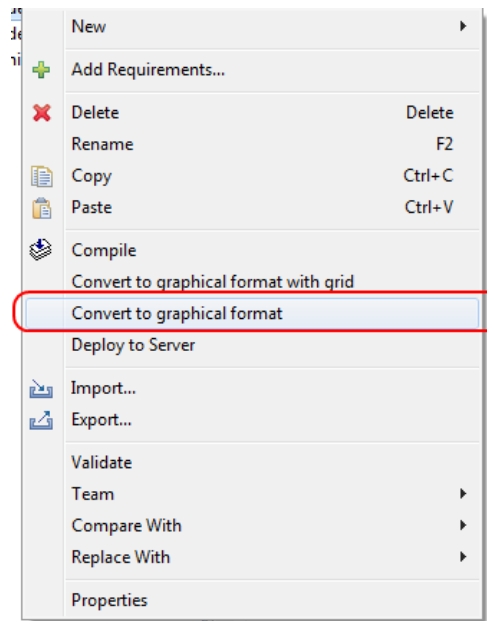
Forms Conversion

Lycia II can work with forms in three formats - the new graphical .fm2 forms, text forms in .per format, and graphical forms in .4fm format which were available in previous Lycia versions.

The forms in previous formats keep their properties and abilities, but they can be significantly enhanced by the facilities of the new WindowBuilder designer after the conversion into the .fm2 format.

Converting old Lycia forms

To convert a form from previous Lycia formats (.per and .4fm), right click on the target form file in the 4GI Project View and select "Convert to graphical format" option of the context menu.



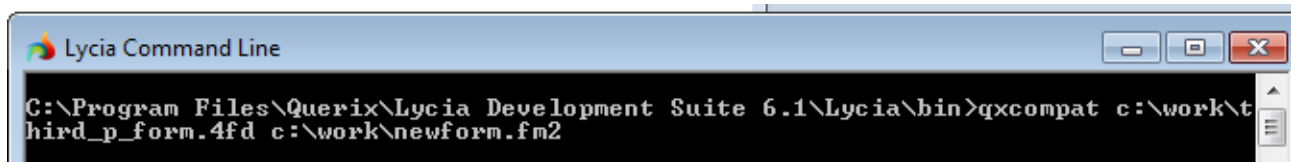
This option converts the source form to a form with a CoordPanel source container, which is a more common one.

The "Convert to graphical format with grid" option converts the source form into a form with a GridPanel as a root container. The detailed information about the types of containers is given in the Form Elements chapter.

When a form file is converted, its source version is automatically excluded from the application files, and the newly converted form replaces it.

Converting third-party forms

The forms generated and compiled by third-party tools can be converted to .fm2 format using the **qxcompat** command followed by the path, the name, and the extension of the source form and the path, the name and the extension of the target file:



The command on the screenshot will convert the *third_p_form.4fd* file into a Lycia form file *newform.fm2*.

You can convert the following files into .fm2 format:

- .4fm - is converted into a form containing all the source form objects
- .4fd - is converted into a form containing all the source form objects
- .4sm - is converted into an empty form with a menu
- .4tm - is converted into an empty form with a menu
- .4tb - is converted into an empty form with a toolbar.

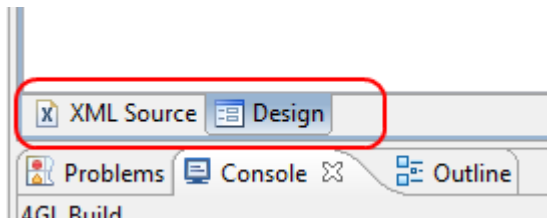
The **qxcompat** command returns a form with a CoordPanel root container. You can also use the **scrcompat** command to get forms with a GridPanel as root container which makes the form resizing at runtime more convenient.



WindowBuilder designer Layout

The WindowBuilder designer has two views: XML source view, in which the xml source code of the form is displayed and can be manually changed, and the design view, in which the form is displayed and manipulated graphically.

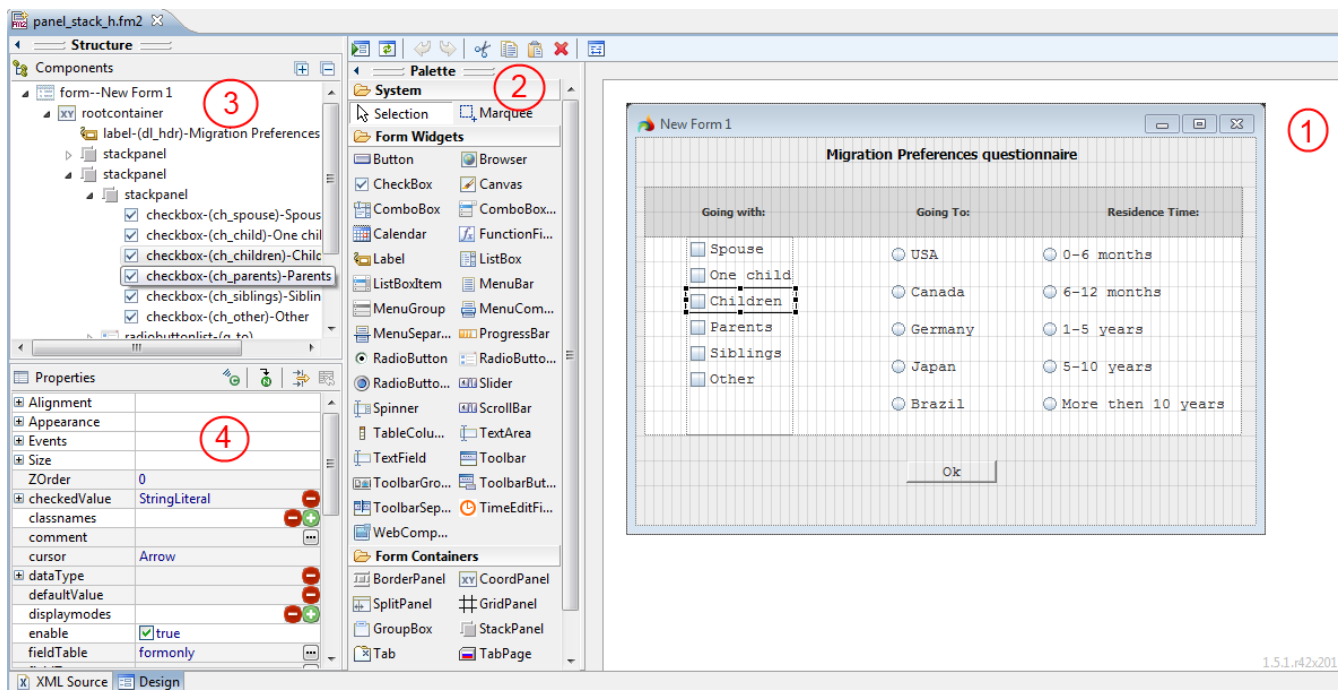
To switch between the views, use the tabs at the bottom part of the designer window:



It is not advised to change the form XML source code, because this may lead to mistakes due to which the form may not be processed correctly.

The WindowBuilder graphical design view, in its turn, consists of four views which give the user the full control over the form creation process.

The views are marked in the screenshot below:





1. **Form Editor Area.** The Editor Area displays the graphical form itself, including all its visible elements and widgets, the way they will look when the application is run.

Here, the objects are dropped from the palette. In the Form Layout view, one can select, move, resize and delete any graphical object present in the form.

2. **Palette.** The Palette contains the set of the graphical elements that can be added to the form. The Palette view contains two groups of widgets:

- Form Widgets - graphical form elements to which the data display or input can be performed.
- Form Containers - the areas of the user-specified size to which the graphical form elements should be added. Different containers locate and treat the form widgets according to their purpose.

3. **Structure.** The Structure view represents the hierarchical structure of the form. The parent items can be unfolded or folded in order to facilitate the access to the objects of lower levels and sibling objects.

The Structure view displays the names of the elements, placed in the form tree, the elements ID's and the text that is to be displayed to the element, if any.

The Structure view also displays the objects that cannot be graphically added to the Editor view.

The view can be used to select, add and delete different form objects.

4. **Properties.** The Properties view gives the user the full control over the graphical and behavioural properties of the form objects. The view is represented by two columns - Property column and Value column. The Property column is organized as a tree, where the top-level elements are the identifiers of the property groups. The property groups are organized according to the properties purpose. The groups, as well as the properties within them, are listed in the alphabetical order.

The views are described in details below in the document.



Editor Area

The Editor area is the place within the WindowBuilder designer, where the layout of the form is specified.

Here, one adds the containers of different types and the widgets within these containers. The graphical settings applied to the form objects are also reflected in the Editor area, so that you can see how the form will look like when the program is run (provided that no theme properties are applied to this form).

Note, that the Editor area does not display the following:

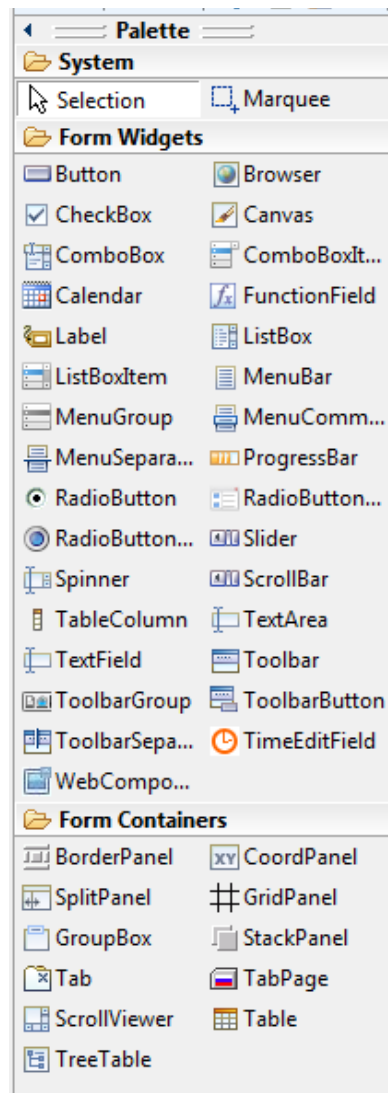
- The edges of some containers (e.g., coordpanel and border panel); therefore, you can see a container location and size only when it is currently selected from the Structure view or by a click on the container space within the form.
- Screen records. You can see the fields placed on the form, but the screen records are not marked or identified in any way within the Editor area.

In the Editor area, one can place the form objects, select them, change their location by drag and drop actions, change objects size by dragging their edges. When an object is selected in the Editor area, its properties are displayed and can be changed in the Properties view.



Palette

The Palette is a tool containing the list of the form elements that can be added to the graphical form. By default, the Palette is placed to the left from the Editor window. You can hide it by clicking the arrow icon at the right part of the palette header. The Palette contains the folder with system tools (Select and Marquee), and the Form Widgets and form Containers folders:

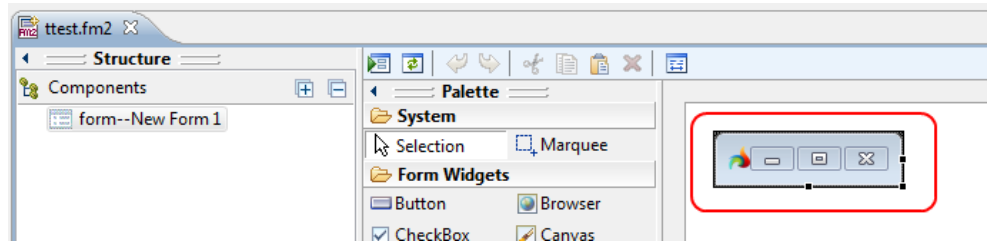


You can hide the content of a folder by pressing the folder icon at the folder title line.

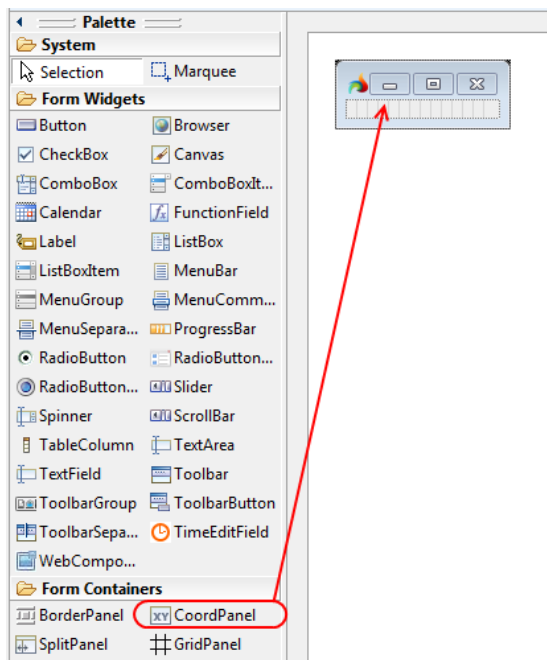
The Containers folder lists a set of containers - form areas, to which the form widgets are placed. The type of container influences the child widgets arrangement. It is possible to place one container within another one, if necessary.



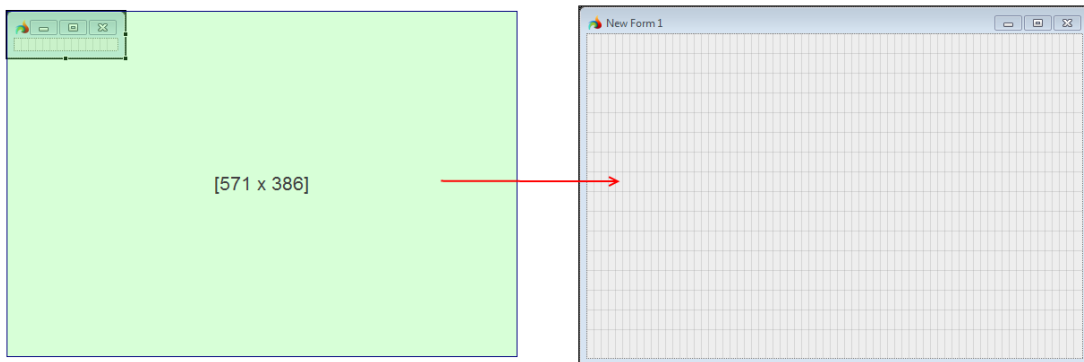
A new form is created without any container inside.



To add a root container, you should select one from the Containers folder and drop it to the form area. When the container is added, the form extends automatically:



To change the form size, select the form in the Editor view and drag its edges till the form gets the desired size and shape:

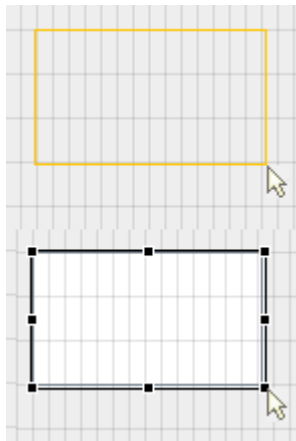




All the other form elements should be placed within the root container.

The Widgets folder contains all the widgets that can be added to the form and used for data input or display.

To add a widget or a container to a form, select it in the Palette and then click on the place on the form in the Editor area, at which the top left corner of the object should be placed. You can specify a non-default size and shape for the new object. To do it, click in the Editor area, hold the left mouse button clicked and drag the cursor so that the widget edges are displayed. Then drag the mouse cursor with the left mouse button pressed until you get the desired object shape and size. On the screenshot, you can see the edges of a text field during its placement on the form and the field itself after the left mouse button is relieved:

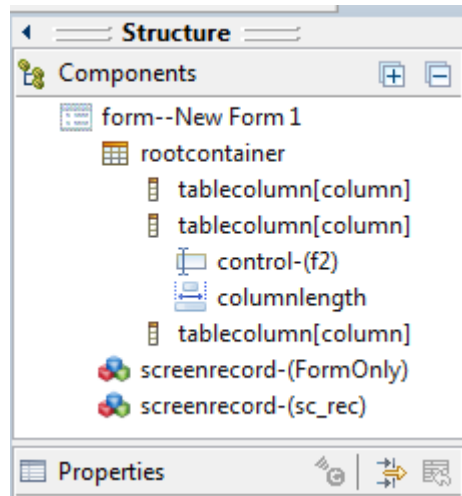


The form widgets can be placed only within one of the containers.



Form Structure

The WindowBuilder designer includes the Structure view which displays the form structure in a tree format.



The Form Structure displays all the elements present in the form and their parent-child relations. It is important that the Structure view displays all the elements and structures present in the form, including the objects invisible in the Editor area, such as screen records. You can select these objects in the Structure view to set up their properties

In the Structure view, you can not only see the structure of the form and select the objects to manipulate their properties, but also copy, paste and delete elements, manipulate screen records and perform some other actions. These actions are performed using the context menu.

The context menu includes the following options:

- Cut, Copy, Paste and Delete - perform classic edition operations;
- Refresh - refreshes the information displayed by the Structure view;
- Layout - allows to change the root container;
- Add To Screen Record - add the selected element to a screen record;
- Delete Tags - deletes the selected type of tags from the form file (i.e, location, data type, displaymodes tags, etc.)

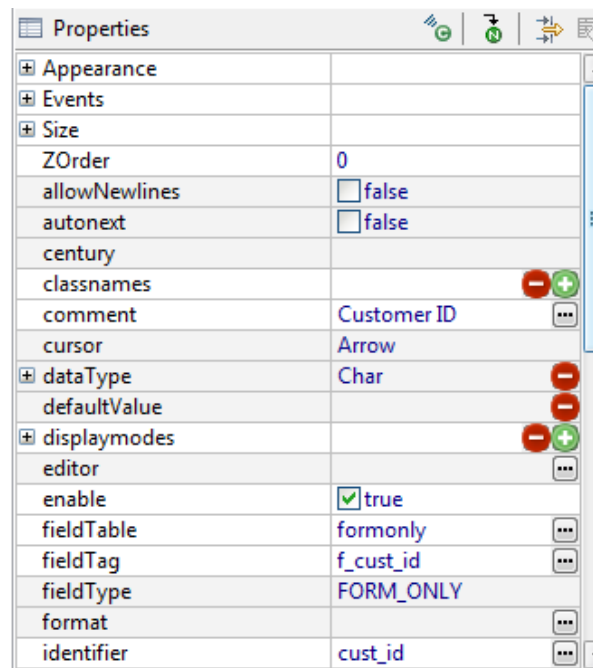
Some of the elements can have their own options available in the context menu. For example, the Radio Button widget has the Change Item Order option.



Properties

Lycia II WindowBuilder designer supports a set of form object properties that give the programmer better control of the form behaviour and layout.

To give the user the convenient control of the properties, Lycia II introduces the Properties view in which the properties of the currently selected element are displayed and can be modified. The Properties view is opened automatically when an .fm2 form is opened. Below is given a screenshot of the Properties view:



The properties view is displayed as a table containing two columns.

The first column is the Properties column displaying the names of the properties organized in a tree list. The top level of the tree displays the names of the property groups, based on the properties purpose. You can fold and unfold property group lists, if needed.

All the properties are sorted in groups according to their purpose. Each object has its own set of properties. There is a number of properties available for all the containers, properties, available for all the widgets, and properties, specific for one or several objects.

The Properties view is organized in such a way, that it is possible to display all the available properties or only the most frequently used ones. By default, the latter option is chosen. To see all the properties, press the **Advanced properties** button at the top right corner of the view:











The different elements of the form, depending on their purpose and behaviour, can contain different sets of property groups and properties.

The property groups, as well as the properties within the groups, are listed in the alphabetical order, which makes them easier to find.

The Value column displays the values, set up for the properties. You can see all the properties of the group or subgroup in the group name line, and the value of each property on the property level:

Size		
minSize	624,484	 
maxSize		 
preferredSize		 
ZOrder	0	








To see the properties of an object, click on the object within the form or select the element in the Form Structure view.

Adding property elements and lists

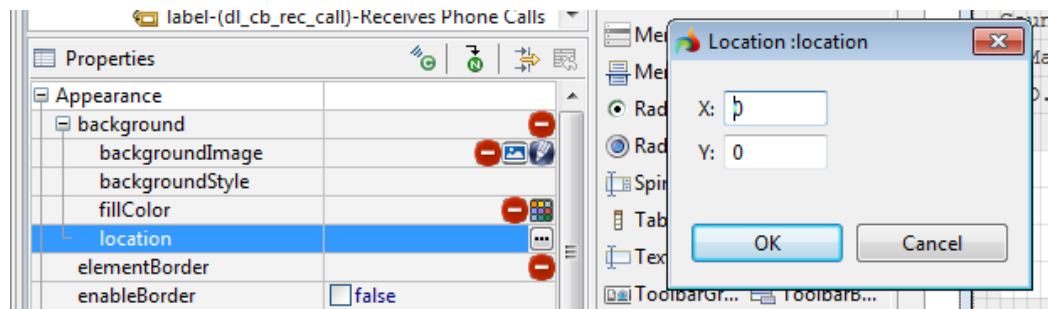
Some of the properties need one or several sub-properties or values to be specified. For example, you need to specify the colours and styles, add the list of the values acceptable within a field, etc.

There are two kinds of situations, where additional values and elements are needed:

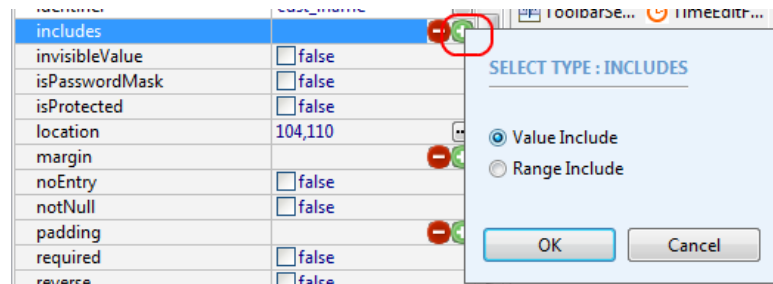
- The number and set of the sub-properties and elements are pre-defined. In this case, when the parent property is activated, all the possible sub-properties are added automatically and you have to set values for them. For example, the Background property includes a set of possible background details:

Appearance		
background		
backgroundImage		 
backgroundStyle		
fillColor		 
location		 

You can set up one or several of these properties. Sometimes, the properties have their own sub-properties, e.g., the location property needs X and Y coordinates to be specified:

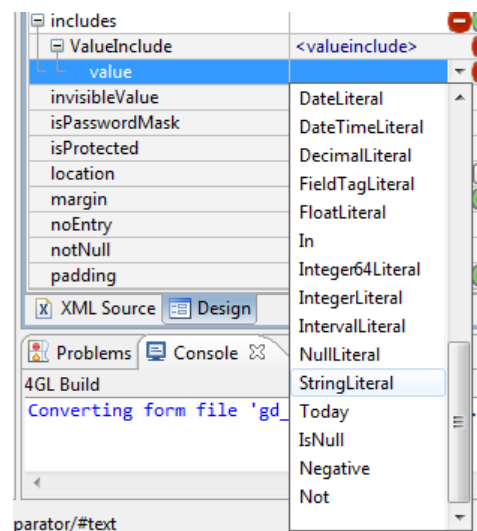


- The number of the elements that will be added is flexible. For example, during the specification of the values within the Includes property. To add a value to such property, click the green "+" button to the right from the property name. For the Includes property, after this click, a dialog window appears where the type of the included value is to be selected:



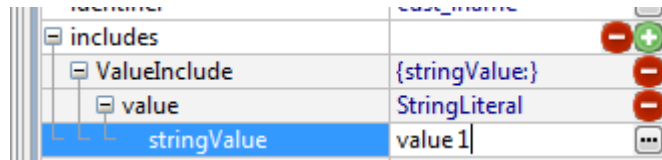
When a value type is defined, a "+" button appears to the left from the Includes property. Click the button to unfurl the property:

- When a property is unfolded, it is necessary to select the type of the data:





- When a data type is set, the value can be inputted:



To add a new value to the Includes list, click the green "+" button next to the Includes property name again. To delete a value, click the red "-" button.

Note that for different properties values and elements are specified in different ways and may have different sets of sub-properties.

Bulk Edition

It is possible to edit several fields simultaneously, if you want them to have the same values for one or several common properties.

To do it, you need to select several fields at once with the Marquee tool or clicking the necessary widgets with the Control key being pressed.

The Properties view will display only the properties available for ALL the selected items. Any value entered now will be automatically passed to all the selected widgets.

General Properties

This section describes a list of properties available for any container or widget.

Size and Location Group

BorderPanelItemLocation

The BorderPanelItemLocation property is used to specify the object location within a Border Panel (is available only when the object is a Border Panel Child). The property has five possible options:

- Center – the object will be placed in the centre of the Border Panel
- Left – the object will be placed along the left border of the Border Panel
- Right – the object will be placed along the right border of the Border Panel
- Top – the object will be placed along the top border of the Border Panel
- Bottom – the object will be placed along the bottom border of the Border Panel

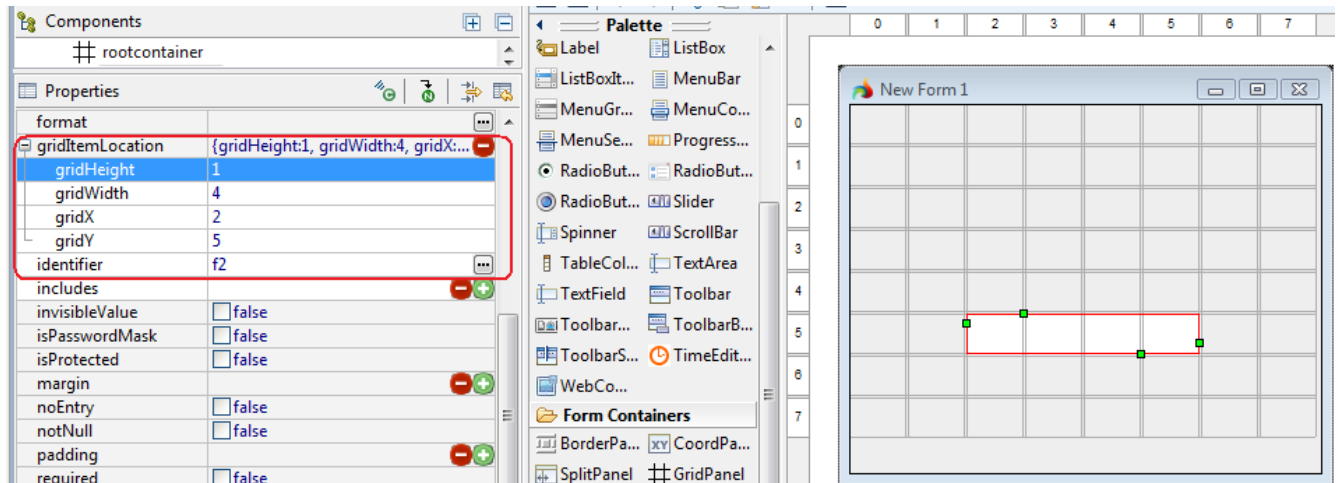
GridItemLocation

The GridItemLocation property is used to specify the location of the object within a GridPanel. The property includes four sub-properties:

- GridHeight – specifies the number of grid lines taken by the object



- GridWidth – specifies the number of grid columns taken by the object
- GridX – specifies the left-most column in which the object is placed
- GridY – specifies the upper row in which the object is placed



Horizontal Alignment

The Horizontal Alignment property specifies the horizontal location of the object relatively to its parent container.

The property can have one of the following values:

- Default - the object is located at its default place, pre-defined by the parent; for example, in the Stack panel, the default alignment is Left, i.e., the object is placed near the left border of the panel.
- Stretch - the object is horizontally stretched along the whole container, if it can have only one element in a row or along the whole involved cells in the GridContainer.
- Left - the object is moved to the left border of the container or the left-most container cell
- Right - the object is moved to the right border of the container or the right-most container cell
- Center - the object is placed in the centre of the line on which it is situated

The Horizontal alignment can be applied only when the object has MinSize or PreferredSize width specified. Otherwise, the object will be stretched along the container. The exception is for the CoordPanel child objects that have the size settings by default.

Location

The Location property identifies the object top left corner X and Y coordinates (in pixels) within a parent Coord Panel container (available only for containers nested within a Coord Panel)

The X value specifies the location on the horizontal axis, Y - on the vertical one.



MaxSize/MinSize/PreferredSize properties

MaxSize and MinSize properties are used to restrict the resize of the object and specify the maximal and the minimal size to which it can be resized at runtime. If these properties are not specified, the user can resize the object to any extent.

The Preferred Size property specifies the unchangeable size that the object should have at runtime. This property blocks the container resizing, i.e., the user cannot change the height and width parameters at runtime. It is possible to make an object partially resizable by specifying only one dimension in the PreferredSize property. This dimension remains invariable, whereas the other can be changed at runtime.

The size properties applied to container children (e.g., table or grid columns and rows) can influence the container resize of the container and restrict it.

The size properties have two sub-properties – Height and Width, which specify the container dimensions in pixels.

Margin

The Margin property specifies the distance from the edge of an object child elements to the edge of the object itself in pixels.

There are four options:

- *Left* – sets the distance from the left edge of a child element to the left edge of its parent element
- *Top* – sets the distance from the top edge of a child element to the top edge of its parent element
- *Right* – sets the distance from the right edge of a child element to the right edge of its parent element
- *Bottom* – sets the distance from the bottom edge of a child element to the bottom edge of its parent element

VerticalAlignment

The Vertical Alignment property specifies the vertical location of the object relatively to its parent container.

The property can have one of the following values:

- Default - the object is located at its default place, pre-defined by the parent; for example, in the Stack panel, the default vertical alignment is Top, i.e., the object is placed near the Top border of the panel.
- Stretch - the object is vertically stretched along the whole parent container, if it can have only one element in a row or along the whole involved cells in the GridContainer.
- Top - the object is moved to the top border of the parent container or the top-most container cell
- Bottom - the object is moved to the bottom border of the parent container or the bottom-most container cell
- Center - the object is placed in the centre of the vertical line on which it is situated



The Vertical alignment can be applied only when the object has MinSize or PreferredSize height specified. Otherwise, the object will be stretched along the container height. The exception is for the CoordPanel child objects that have the size settings by default

ZOrder

The objects are mostly non-transparent and overlap each other, when their coordinates intersect. The ZOrder property is used to specify, on which "layer" the object is situated, i.e., if it will overlap or be overlapped by the other objects in the area.

The ZOrder property value is an integer. The higher the value, the closer the object will be to the top of the stack.

Text and Image Group

Locale

The Locale property allows to specify the locale settings for the current object. This may be useful for applications designed for users from different countries. To set up the Locale property, click on the button to the right from the property value field. This will call a dialog window where the locale settings are available for choosing. When a country from the locales list is selected, the country name and the language fields are set automatically. However, you can change their values manually:



Click the **Ok** button when the language and the country are selected.

ToolTip

The Tooltip property is used to specify the text for the tooltip which is displayed when the mouse cursor is placed over the object.



Appearance Group

Background

The Background property includes a set of options that are used to change the appearance of the background of an object. A background can be of a specific colour and it can also include a background image. Colour and image can be used separately or both at once.

The default background is darker grey for containers or white or transparent for widgets. In the picture below, a part of a CoordPanel with a text field on it are shown:



To specify a new background for the selected object unfold the Background property and set up one or several of the available sub-properties:

- BackgroundImage
- BackgroundStyle
- FillColor
- location

BackgroundImage

To add a background image, you have to specify the Image URI path as the property value. Click on the button at the right part of the Value line to run the Image Selection dialog window. If there are no available images in the application folder, add one to the project and repeat the attempt. For more detailed information about the image specification, see the Image property description.

To delete the background image, click the red **Remove** button near the image path.

BackgroundStyle

The Background style specifies the way the background image will be displayed. There are several possible options:

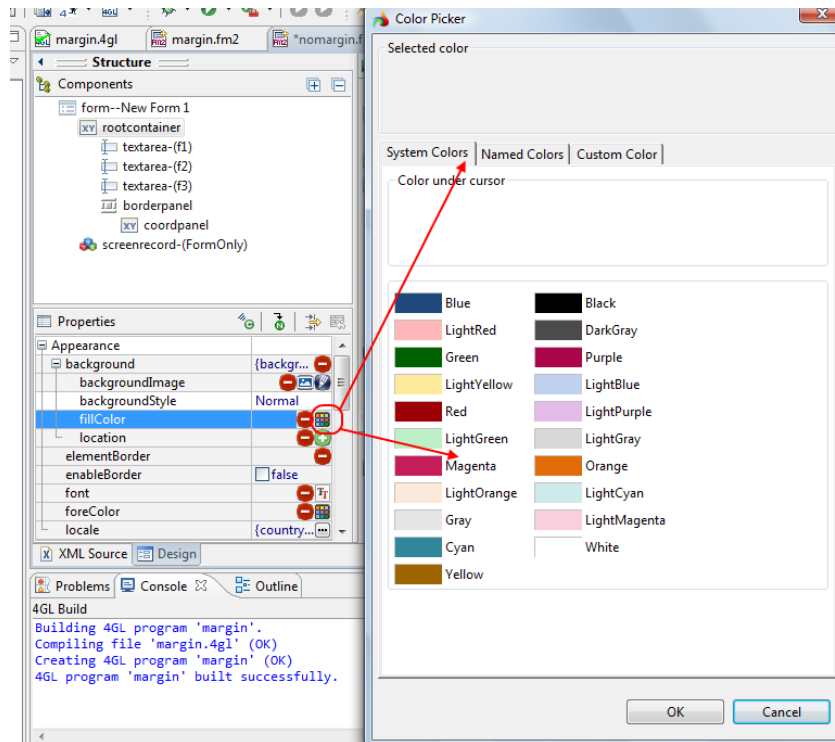
- *Normal* - the border image will be displayed as it is without adjustments.
- *Stretched* – the background image will be stretched along the border length. The image ratio may be distorted in this case.
- *Tiled* – the background image will be repeated along the border length.



- *Centered* – the background image will be located in the centre of the border. If the image is smaller than the background area, the area around it will only contain background color, if the image is larger than the background, its sides will be truncated.
- *Uniform* - the image will cover the whole background area without the image ratio being distorted. Some margin will be added to the image.
- *Uniform to fill* - the image will cover the whole background area without the image ratio being distorted. No margin will be added.

FillColor

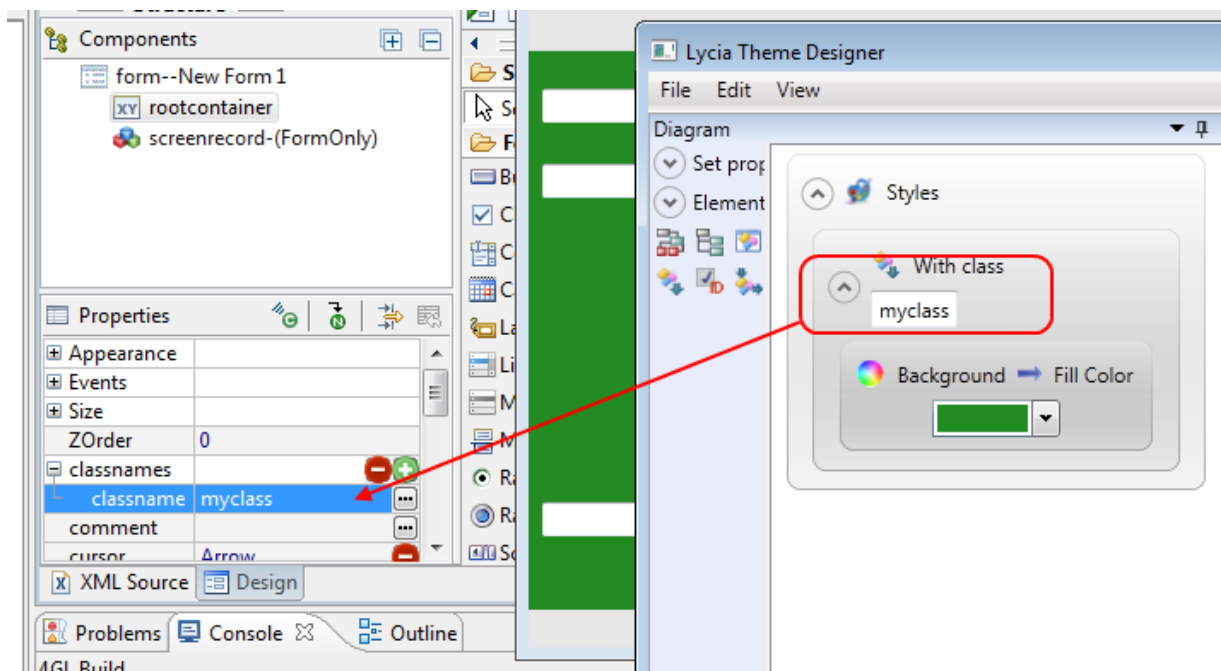
The FillColor property is used to specify the colour that will be used to fill the background of the selected object. There are two types of colours: system colour and customized colour. To activate the FillColor property and select the colour type click the **Choose Color** button in the FillColor property line:



In the dialog window, you can select one of the given system colours (**System Colors** tab), choose a colour from a palette (**Named Colors** tab) or specify a colour by setting its RGB parameters (**Custom Color** tab)

Classnames

The Classnames property is used to specify to bind an object to a Theme class or classes. The property should list the names of the class or classes the object will be associated with. The effect of the property can be seen only at runtime:



Cursor

The Cursor property specifies how the cursor will look when it is placed over the selected form object. The property accepts one of the 14 cursor names given in the Cursor property dropdown list:

- *Arrow*
- *Cross*
- *I beam*
- *Size All*
- *Size NESW*
- *Size NS*
- *Size NWSE*
- *Size WE*
- *Up Arrow*
- *Wait Cursor*
- *Help*
- *H Split*
- *V Split*
- *Hand*

ElementBorder

The ElementBorder is a set of properties that create a border around the object. The ElementBorder itself has three possible types of borders:

- *LineBorder*
- *EtchedBorder*



- BevelBorder

When a border type is specified, a list of border set-up properties appears. They are:

- Background - sets up the border background
- Border Brush - sets up the border colour
- Corner Radius - sets up the radius of the border corners curve
- Is Raised - the checkbox specifying whether an Etched or a Bevel border will be raised ("Is Set" value checked) or lowered ("Is Set" value is unchecked)
- Padding - specifies the border padding
- Thickness - specifies the border thickness. If set to 0, the border is not displayed.

Enable

The Enable property is a checkbox that indicates whether any user interaction can be performed within the container. The Enable property is set to TRUE by default

Visible

The Visible property is a checkbox that indicates whether the object should be visible (Checked) or hidden (Unchecked).

VisibleToolbarGroups

The VisibleToolbarGroups property is used to list the names of the toolbar groups that will be available when the application is focused on the object.

Events Group

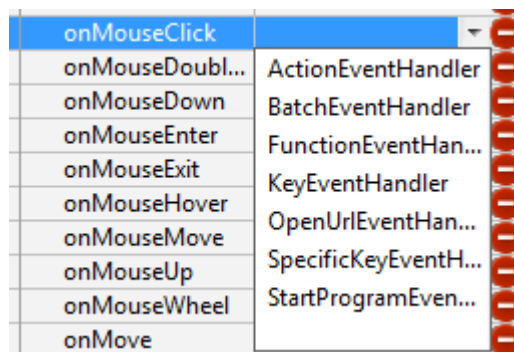
The Events property group specifies the way the program will react on the users' actions performed with the form object.

The properties of this group are used to specify one of the following events that will be triggered after the user performs some action:

- Action Event Handler
- Batch Event Handler
- Function Event Handler
- Key Event Handler
- Open URL Event Handler
- Specific Key Event Handler
- Start Program



To specify an Event, select on its row in the Properties view, and then press the details field:



Each event handler includes at least two properties – the handler specification (e.g., action or key names) and a SubDialogIdentifier. The SubDialogIdentifier is used to specify the ID of the field in which the current event can be activated. If this property is set, the Event will be triggered only when the cursor is placed in the specified field and ignored when in others.

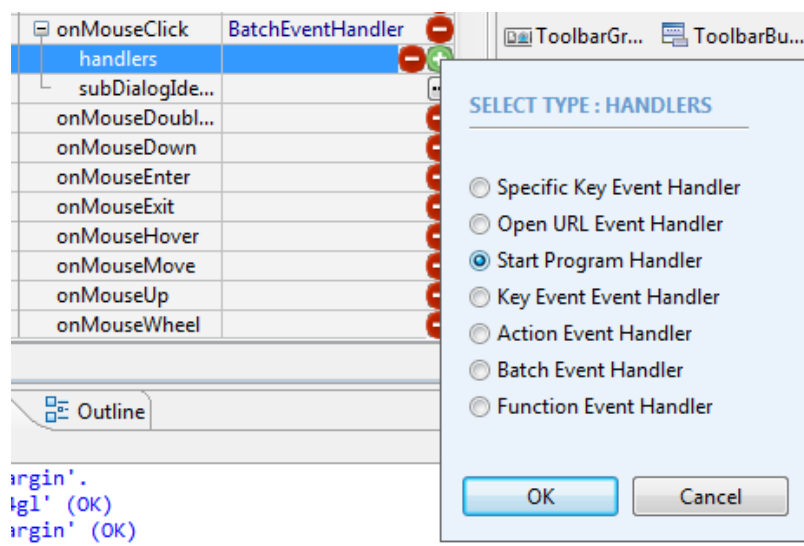
ActionEventHandler

The Action Event handler binds the event with some action name, which will be used in a corresponding **On Action** block of an input or display statement.

BatchEventHandler

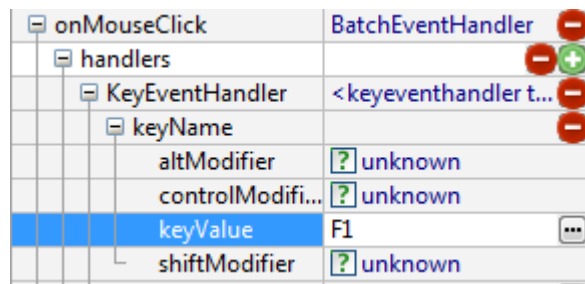
The Batch event handler is used to specify a set of event handlers that will be processed one by one when the event takes place.

To add a new event handler, click the **Add** button in the right part of the **Handlers** line. This will call a context window where the handler type is to be selected:





After this, you can set up the event handler, as usually:



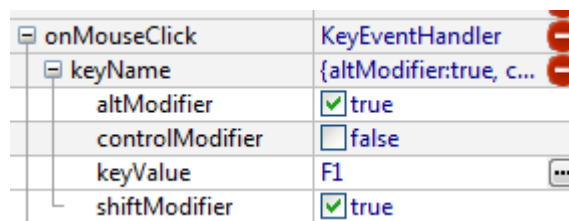
When the event handler is set up, click the Add button to add a new one to the batch.

FunctionEventHandler

The Function Event handler binds the event with a 4GL function present in the application. When the event takes place, the program calls the function specified in the event handler properties. The function name should go without brackets (*my_function*, not *my_function()*) and cannot pass any arguments.

KeyEventHandler

The Key Event handler binds the event with some key name that will be used in a corresponding **On Key** block of an input or display statement. You can set up the Key event handler so that it will be triggered when the given key is pressed together with Alt, Control, or Shift modifier keys. To select the modifier, check the corresponding checkbox in the key name properties list:



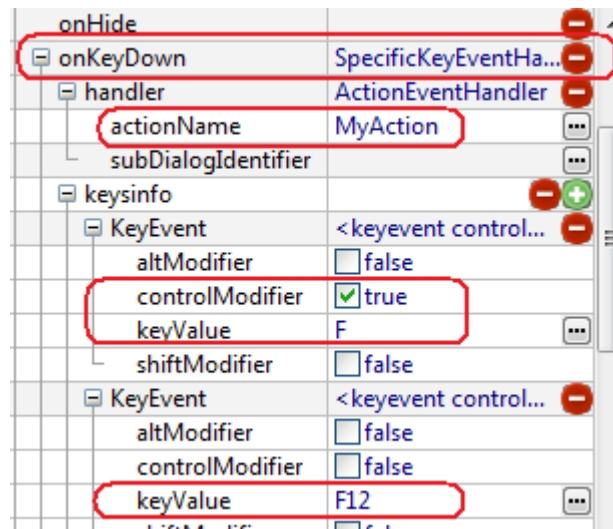
OpenUrlEventHandler

When an OpenUrl event handler is triggered, the application starts the system default web browser and passes the URL specified in the handler to it.

SpecificKeyEventHandler

The SpecificKey event handler is used to specify some events that will be triggered by a specific keypress. The handler has effect only when used with the OnKeyDown and OnKeyUp events that are triggered when the user, respectively, presses and releases a keyboard button.

The SpecificKey event handler includes the **Handler** property, which specifies the handler to be used, and the **KeysInfo** property that indicates one or several buttons that will trigger the event:



According to the settings on the screenshot above, pressing the F12 key or the Control-F combination will trigger the MyAction action execution.

StartProgramEventHandler

The StartProgram event handler is used to run a 4GL application. There is a list of application details that are to be given for the handler to launch the program:

- Program name
- Program Port
- Program Server
- User ID

OnDragEnter

The OnDragEnter event is triggered in the target dialog when the user moves the cursor from row to row after entering it or changes the drop operation (from copy to move or vice versa).

OnDragFinished

The OnDragFinished event is triggered in the source dialog when the rows are dropped to the target dialog.

OnDragOver

The OnDragOver event is triggered in the target dialog when the mouse button is released over it.

OnDragStart

The OnDragStart event is triggered in the source dialog when the mouse cursor enters the target dialog to which the dropping will be performed.



OnDrop

The OnDrop event is triggered in the target dialog when the user releases the mouse button over it.

OnFocusIn

The OnFocusIn event is triggered when the program focus goes to the object with this event being specified. This typically means that the cursor passes to this object.

OnFocusOut

The OnFocusOut event is triggered when the cursor leaves the object for which this event is set up.

OnHide

The OnHide event is triggered when the object visibility option is changed to hidden.

OnInvoke

The OnInvoke event is triggered when the user clicks a button widget, a button of the Function Field widget, a menu or a toolbar button.

OnKeyDown

The OnKeyDown event is triggered when a keyboard key is pressed down.

OnKeyUp

The OnKeyUp event is triggered when a pressed keyboard key is released.

OnMenuDetect

The OnMenuDetect event is triggered when the user releases the right mouse button.

OnMouseClicked

The OnMouseClicked event is triggered when the user clicks the left mouse button.

OnMouseDoubleClick

The OnMouseDoubleClick event is triggered when the user double clicks the left mouse button.

OnMouseDown

The OnMouseDown event is triggered when the user presses a mouse button down.

OnMouseEnter

The OnMouseEnter event is triggered when the user moves the cursor into the object for which this event is specified

OnMouseExit

The OnMouseExit event is triggered when the mouse cursor leaves the object.

OnMouseHover

The OnMouseHover event is triggered when the mouse cursor stays over the object and doesn't move for one second.



OnMouseMove

The OnMouseMove event is triggered when the mouse cursor moves over the object.

OnMouseUp

The OnMouseUp event is triggered when the pressed mouse button is released.

OnMouseWheel

The OnMouseWheel event is triggered when the user rolls the mouse wheel.

OnMove

The OnMove event is triggered when the object location is changed.

OnResize

The OnResize event is triggered when the object is resized at runtime.

OnSelection

The OnSelection event is triggered when the object is selected.

OnShow

The OnShow event is triggered when the object is displayed or redisplayed to the screen.

OnValueChanged

The OnValueChanged event is triggered when the value in a field is changed at runtime

Container Properties

This section describes a list of properties available for any container that can be added to a form. The container-specific properties are given in the corresponding container description.

Ungrouped properties

Identifier

The Identifier property is used to specify the container identifier. The rules of the container identifiers creation are similar to those for all the other form objects.

Widget Properties

This section describes a list of properties available for any widget that can be added to a form and not available for the containers. The properties specific for definite widgets are given in the corresponding widget description.

Text and Image group

AllowNewLines

AllowNewLines property accepts a Boolean value which indicates whether a text within the object will be automatically split into several lines in case the string is too long to fit the object width.



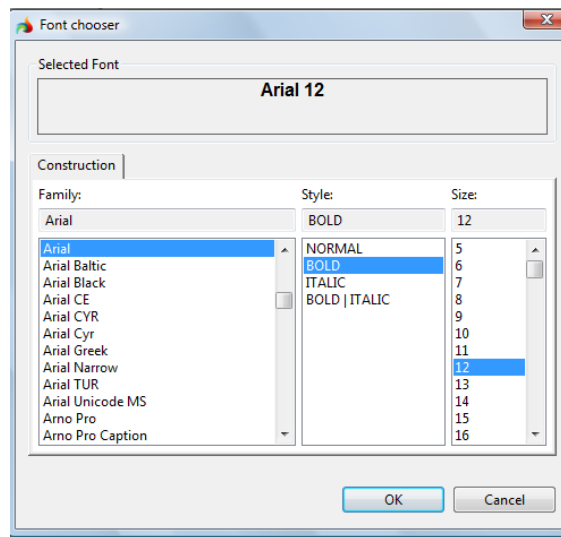
The property is most typically used in Label widgets in order to allow multiline labels, but may also be applied for Buttons, TextFields and textAreas.

If the value displayed to a widget includes the New line character, the AllowNewLines property is not needed to split the string in the specified place.

Font

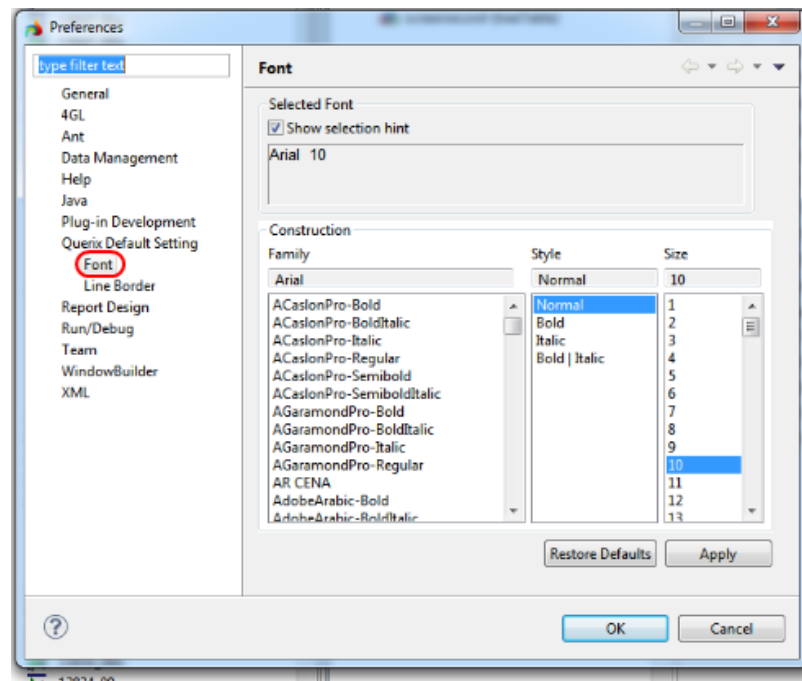
The Font property is used to specify font setting for the widget text. It is available for most widgets and some of the containers.

To specify a new font, click on the property line in the Properties view. This will open a dialog window with the font settings to be selected. All the properties – Font family, Style and Size should be selected, otherwise, the new font settings won't be submitted:



Default View Font

The default ViewFont property specify the TextFont property to any text element to which no TextFont property was set. These settings can be changed from the main menu: **Window -> Preferences** main menu option:



ForeColor

The ForeColor property is used to specify the widget forecolour, i.e., the colour of the widget text. The colour selection process is the same as for the background colour described above.

Size and Location group

Padding

The Padding property specifies the distance between the widget edge and content

Appearance group

IsProtected

The IsProtected property indicates that the widget will be on top of the stack if it is located in a place where several properties overlap each other.

DisplayModes

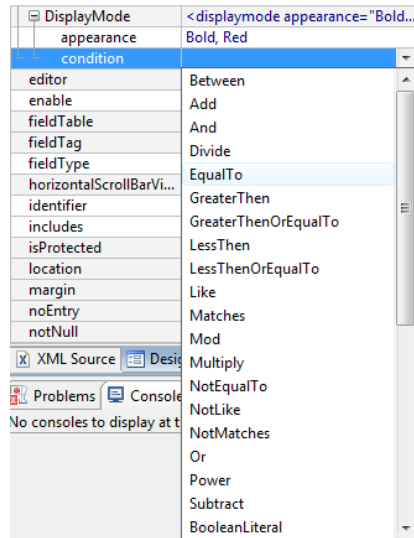
The DisplayModes property is available not for all, but for most widgets. It allows to specify conditional display settings for a widget. It means that different display settings can be used for one widget in different contexts and situations. Unlike the Forecolor, the DisplayModes can not only change the text colour, but also its intensity and type.

To add a display mode, click on the **Add** button to the right from the property value field. Each DisplayMode has two properties:

- Appearance – stores the appearance settings, e.g., "Bold, Red"



- Condition – specifies the condition under which the appearance settings are applied. The Condition dropdown list gives a vast number of possible options for different sets and types of possible values and contexts:



When the condition is chosen, you can fill the condition details.

An object can have several display modes, each specifying its own appearance and conditions setting, so that the display to the object will depend on the current situation.

If no condition is specified for a display mode, the appearance settings will be applied in any context.

If both Forecolor and DisplayModes apply colour settings to a widget, the Forecolor property has the higher priority. The DisplayModes settings that do not influence colour, will have effect.

Ungrouped properties

Identifier

The Identifier property is used to specify the widget identifier.

FieldTable

The FieldTable property is used to specify a database table to which the field will be linked.

FieldType

The FieldType property is used to specify the type of the field. There default option is FORM_ONLY. The other options are used to link the field to a database.



FieldTag

The FieldTag property is a legacy property which stores the field tag. The tag is not displayed either in Design view or Runtime, but may be useful after the form conversion from a .per file, especially if a .per form tag and fieldname are different:

```
tag=table.fieldname;
```

The **Tag** value is saved to the FieldTag property and it may be useful in finding the necessary fields in a newly converted form.

TabIndex

The TabIndex property is used to indicate the place of the field when the FIELD ORDER option of the OPTIONS statement is set to FORM. The FIELD ORDER FORM option makes the cursor move from field to field according to the TabIndex form fields attribute. If the tab index of a field is set to zero, the field is excluded from the tabbing, but you can still focus on it using the mouse.



Form Elements

The graphical form contains a set of objects, used for the form layout manipulation, data input and display and program workflow control.

In this chapter, you can see the description of different form objects, their purpose, and object-specific properties.

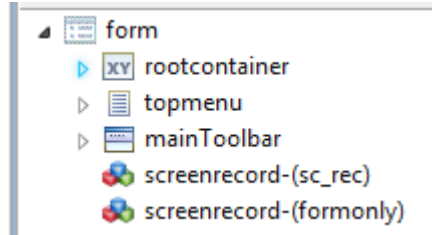
The objects in the chapter are ordered by levels - from top-level objects like Form and containers to lower-level objects like fields. Some of the containers can include only one child. To add several objects to such containers, you can add another container as a child.

Form Object

The Form object is the abstract container where all the contents of the form are stored. It cannot be deleted. The Form object has a set of unique properties that do not apply to individual widgets or containers.

In this section you can find the information about the properties that can be applied for the root form element. These properties that apply to the form include the following properties crucial for the application development.

Thus a form object can have any one of the elements described above or all of them. This is reflected in the Structure view, e.g.:

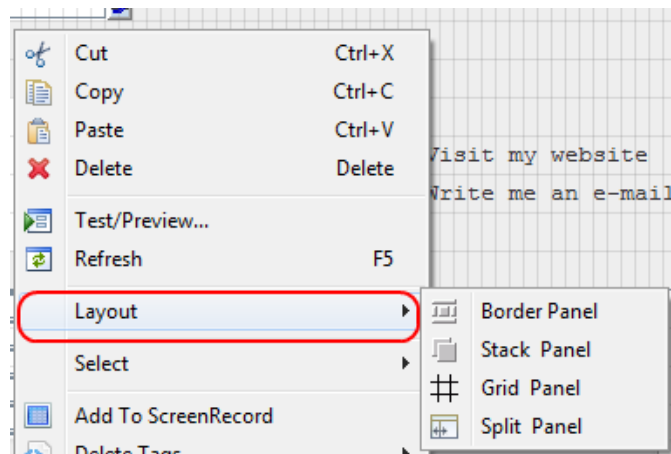


The Form object also allows you to specify the Database to be used for the form.

Root Container

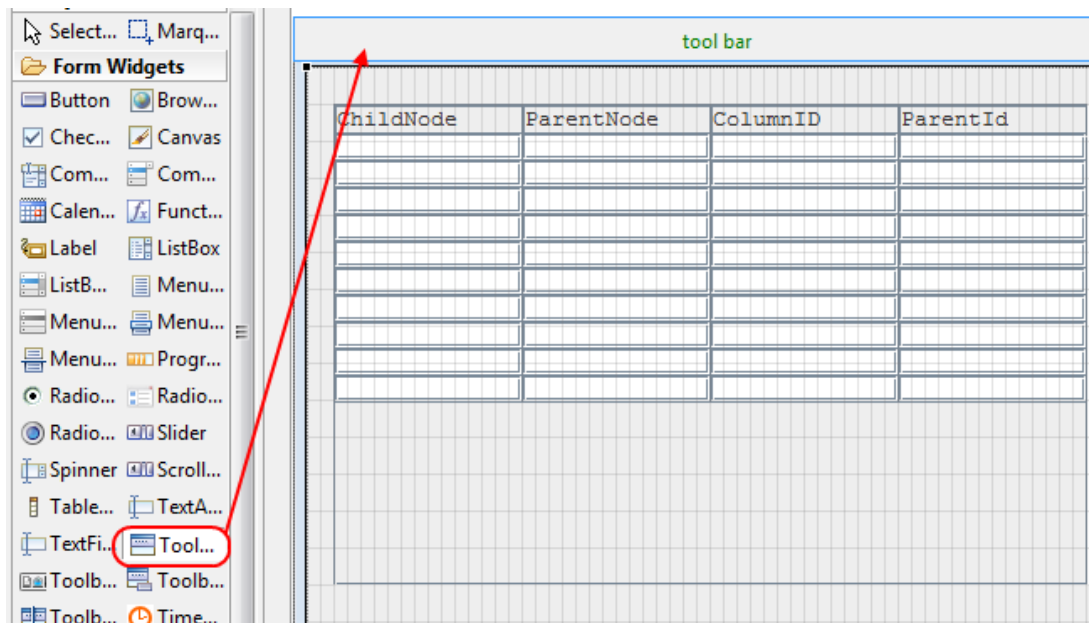
When the form is created, you need to select the root container it will be using. All the other containers and widgets can only be placed inside the root container. The choice of the root container influences the resizing and layout possibilities for the form at runtime. The Coord Panel is the default root panel of a form, but with it as the root container the form cannot be resized properly. Other containers pose their own limitations, for example if the Stack panel is used as the root container, the form can be resized only in one direction which is the opposite direction of the panel orientation.

The root container can be changed by a Layout option of a context menu called by a right click on the container in the structure view or design area:

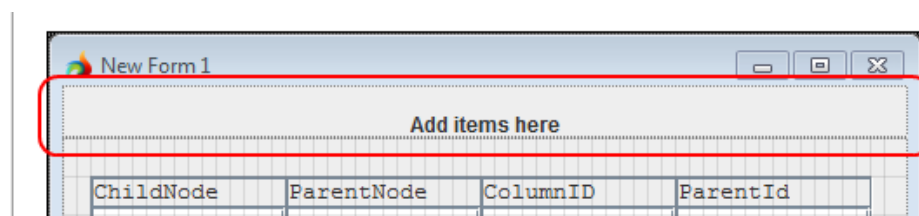


Main Toolbar

The form toolbar is can be created by adding the Toolbar object to the form. The toolbar is placed at the top of the form, above all the containers and widgets. To add a toolbar, select it in the Form Widgets palette folder and put the mouse cursor to the form title bar, so that the toolbar area becomes highlighted:

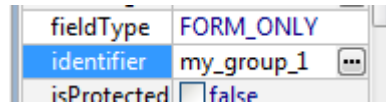


An empty toolbar is added when you click o the toolbar space, and is placed between the form title bar and content:

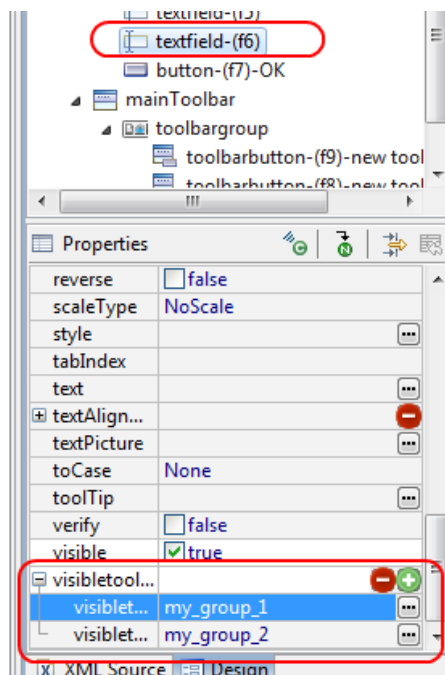




The main item of a toolbar is the **ToolbarGroup**. This item can include one or several buttons - the elements which are displayed to the toolbar and actually trigger events at runtime. The toolbar group itself does not trigger any options. It is used to group the buttons by their purpose and create dynamic toolbars with the content depending on the situation. Each toolbar group has a unique identifier specified by the **Identifier** property:

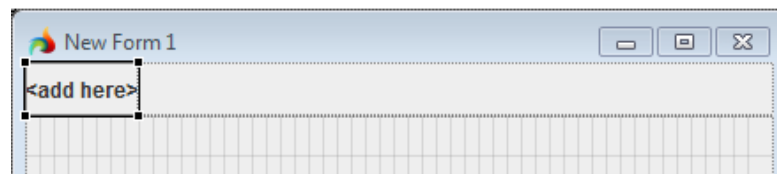


Form elements have the **VisibleToolbarGroups** property, which lists the identifiers of the toolbar groups that should be available when the program focus is in this or that element:



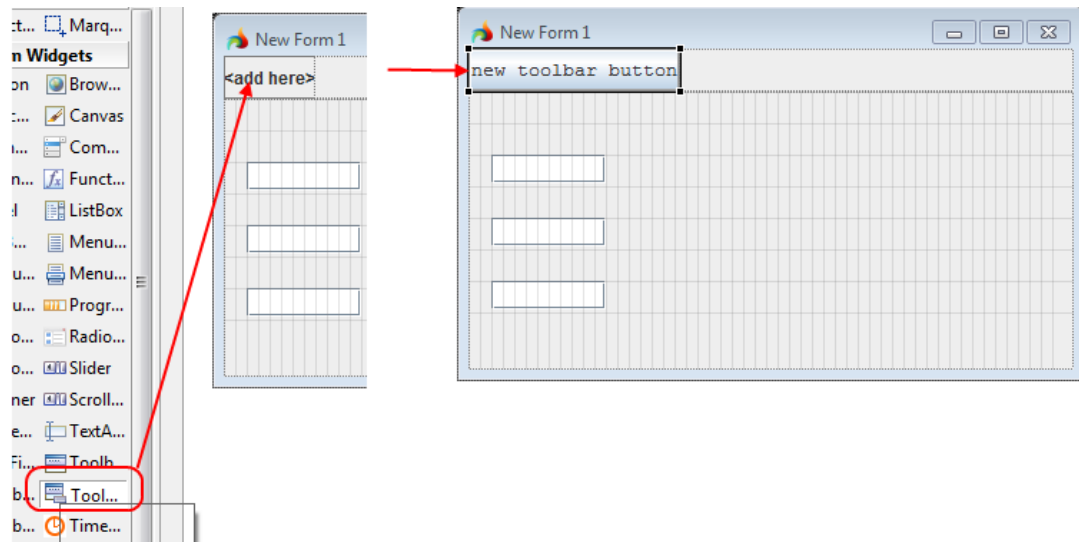
If this property is empty for an element, the toolbar group specified the first in the toolbar, will be used.

When an empty toolbar group is added to the toolbar, it has an **<Add here>** label:

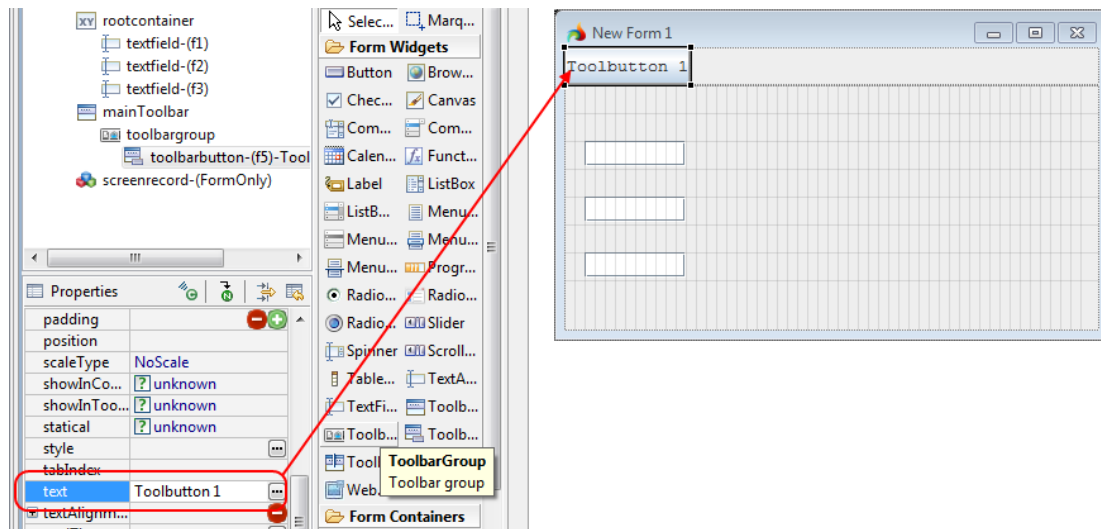




The toolbar buttons are to be placed within groups:



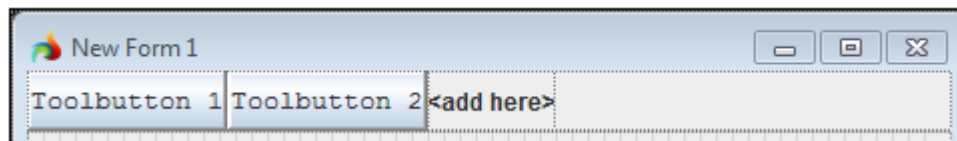
The toolbar button label is set up by the button Text property:



The toolbar buttons are displayed next to each other:

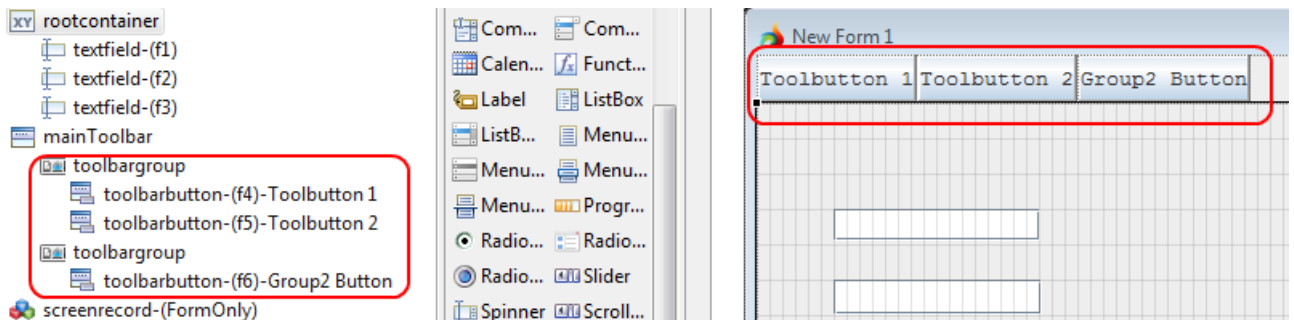


A new toolbar group cannot be placed between the buttons belonging to one agroup:

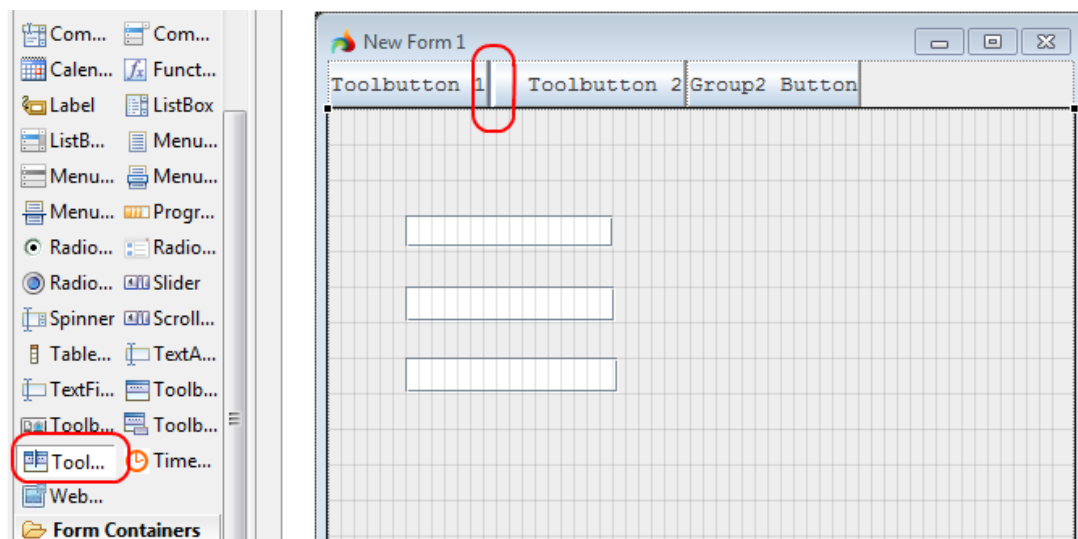


On Properties view, it is specified in the MainToolbar property group.

The toolbar groups are not graphically separated in the design view, but it is possible to see the toolbar structure in the Structure view:

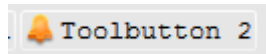


However, you can use the ToolbarSeparator widget to separate the buttons visually. It is a graphical object that has no effect on runtime. The ToolbarSeparator can be found in the Palette, together with the other form widgets:



To make a toolbar button work, you have to specify at least two properties:

1. **Text** or **Image** property, which specify the text or the image displayed to the toolbar button. If both these property values are empty, the toolbar button will be hidden. If both are specified, the image will be displayed to the left of the label text:

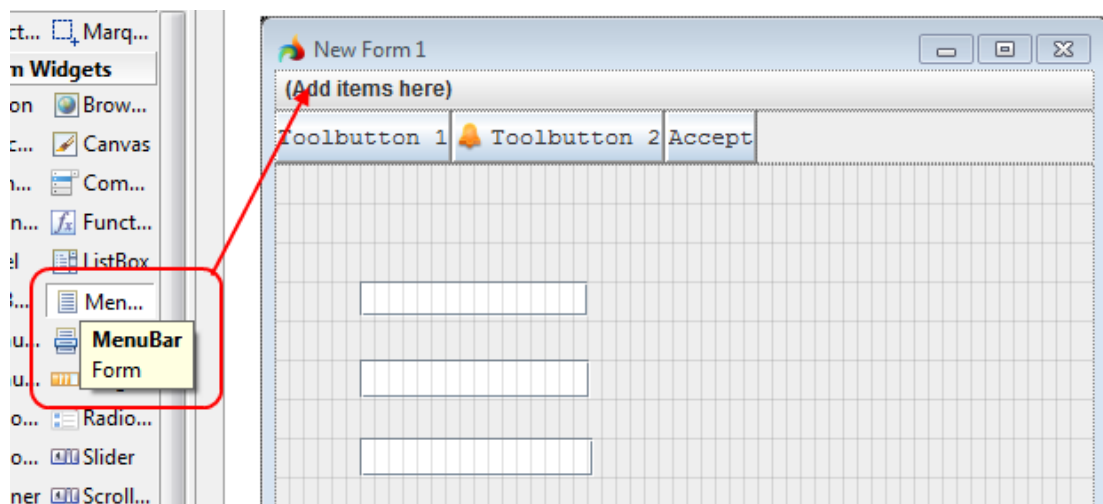


2. An **Event** property is needed to activate the toolbar button. You need to select one or more possible events and event handlers for them. The most typical event here is OnInvoke which is triggered when the user clicks on the corresponding button. However, you can specify other events from the available list.

Form Menu

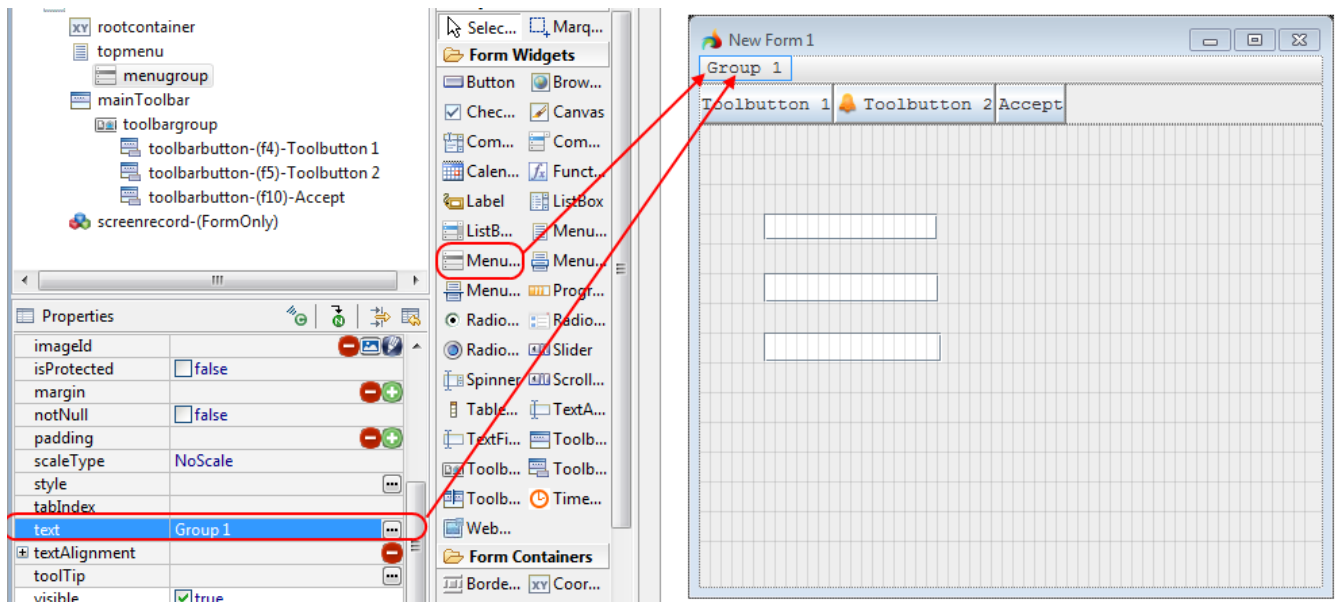
Lycia .fm2 forms can have their own menus, which are available during the input or display to the form. The form menus can have more complicated structure than 4GL-specified ring menus, e.g., it is possible to create sub-menus and add menu separators, as well as manage the menu display properties.

The main parent object of the form menu is the Menu Bar. It can be selected in the widgets folder of the Palette. The Menu Bar is placed above all the form content and the toolbar:

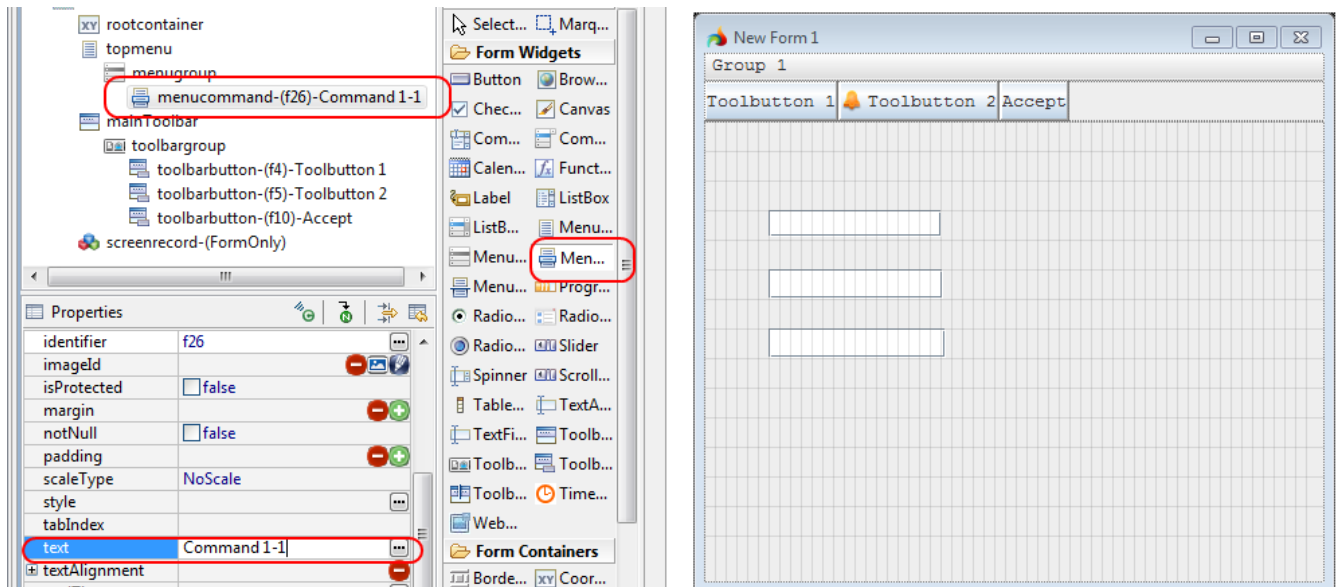


The main object of a form menu is a menu group - a container for a drop-down list with one or more menu commands and inbuilt menu groups.

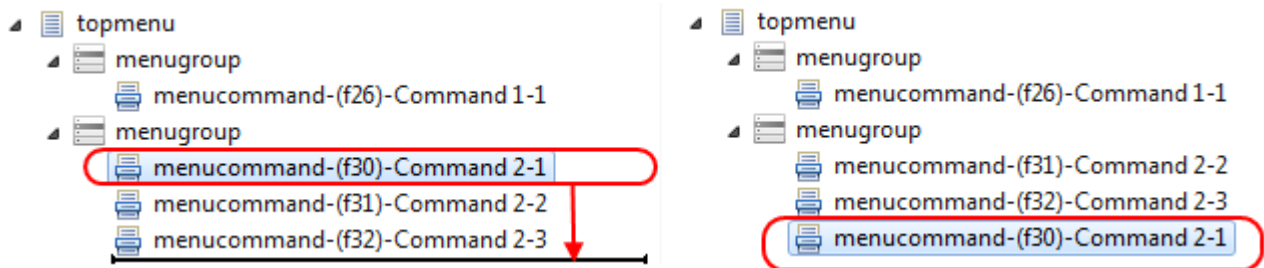
To add a menu group, select it in the Palette view and click on the Menu Bar. The text of the label is set up by the **Text** property:



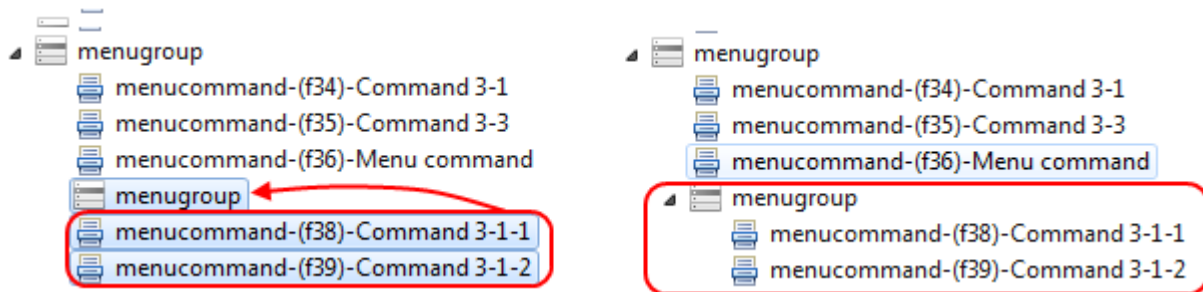
To add a command to a toolbar group, select the Menu Command in the Palette and click on the target menu group. Unlike the Toolbar, the menu commands are not displayed in the form layout, and can be found only in the Structure view:



You can change the order of the commands within a group or put them into another group by dragging and dropping them to the necessary place in the Structure view:



As it has been mentioned before, it is possible to create multi-level menu groups. To do this, add a menu group object to an existing menu group, the same way menu commands are added. The new group will become a child one. Then, add the necessary menu commands. By default, they will be the first-level children of the root group, but you can select them and drag to the in-built one:



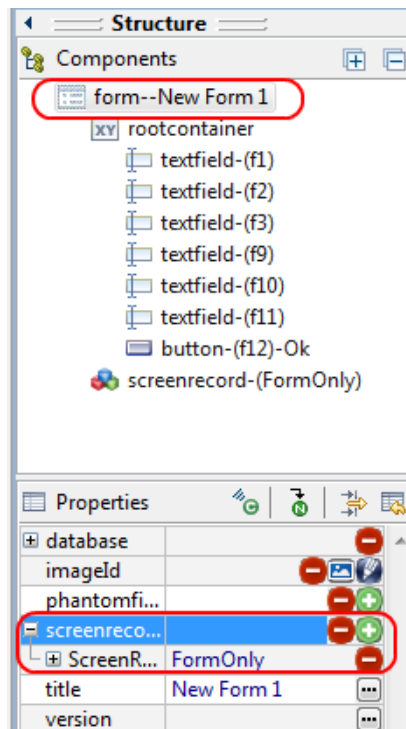
There are two properties, necessary to make the menu commands work.

1. **Text** or **Image** property is necessary for both Menu Group and Menu Command in order to make them visible. The string specified as the Text value or the specified image will be displayed to the corresponding Menu Group or Menu Command button.
2. An **Event** property is needed to activate the Menu Command button. The most typical event here is On Invoke which is triggered when the user clicks on the corresponding command. However, you can specify other events from the available list.

It is also necessary that the key, action or function specified in a Menu Command properties were referenced by the dialog, active in the form. Otherwise, they will be inactive.

Screen Records

Screen Records are listed as the first-level children of a form object and are specified with its ScreenRecords property:



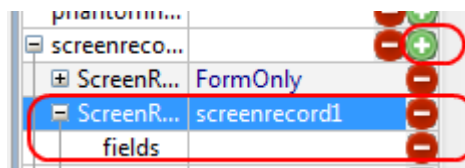
Default Screen Records

There are screen records that are created automatically and change when the form content is changed. The main default screen record is the Form Only screen record. Any form widget, to which the input is possible, is added to the FormOnly screen record. This means, that widgets designed only for data display or user manipulation (label, button, progressbar) are not included to the FormOnly screen record.

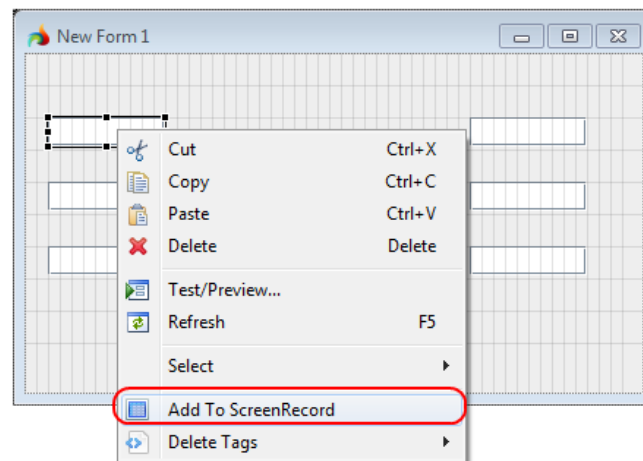
There are also screen records automatically created for form tables and tree tables. They include the fields that fill the table columns. The changes in the table structure are reflected in these screen records content. If a column is added or deleted from the table or a tree table, the corresponding field is added or deleted from the table screen record. The identifiers of tables and their screen records are identical and bound. The changes in one lead to the changes in the other.

Adding Screen Records

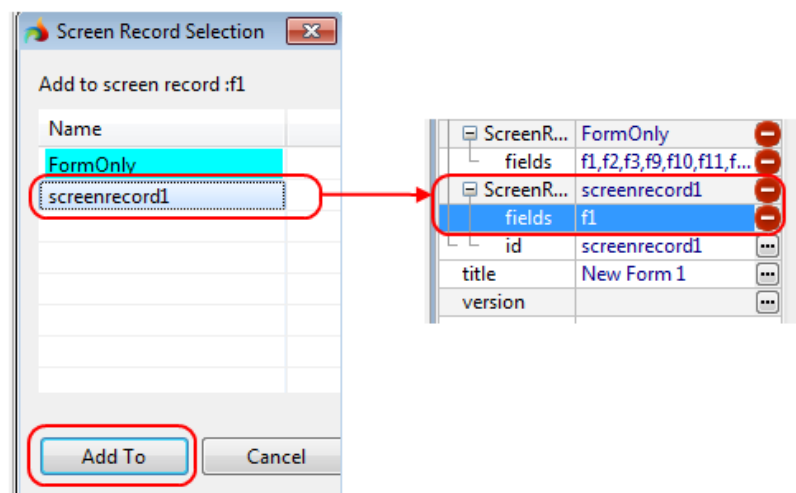
To add a new screen record, press the **Add** button to the right from the property. This will create an empty screen record with a default identifier:



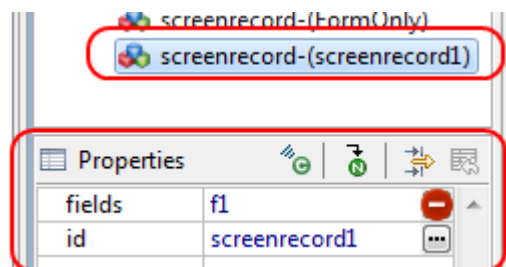
To add a field to the screen record, right click on this field in the editor or Structure view and select **Add to ScreenRecord** option:



This will call a dialog window where you should choose the screen record to which the field should be added, and then press the **Add To** button, and the field will appear in the selected Screen Record list:



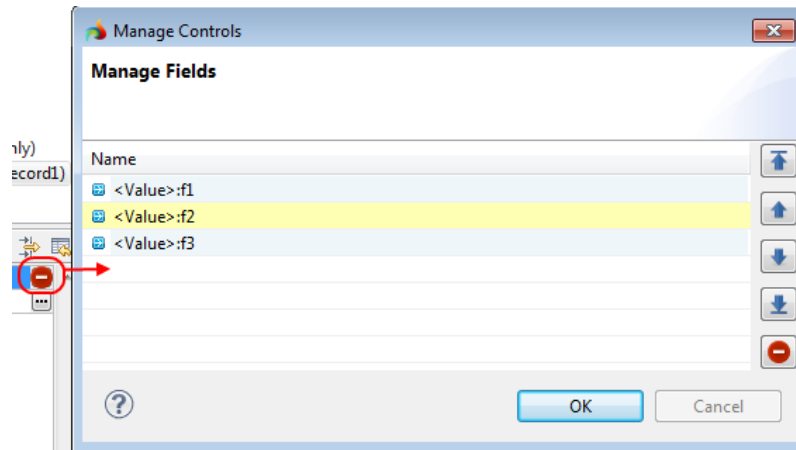
The screen records can be manipulated from both Form container properties, as is shown above, and from the Screen Record properties - each Screen Record is displayed to the Structure view and can be selected for further manipulations:





To add serd, select them at once, click the **Add to ScreenRecord** context menu option, and in the appearing dialog windows, select the screen record each of them should be added to.

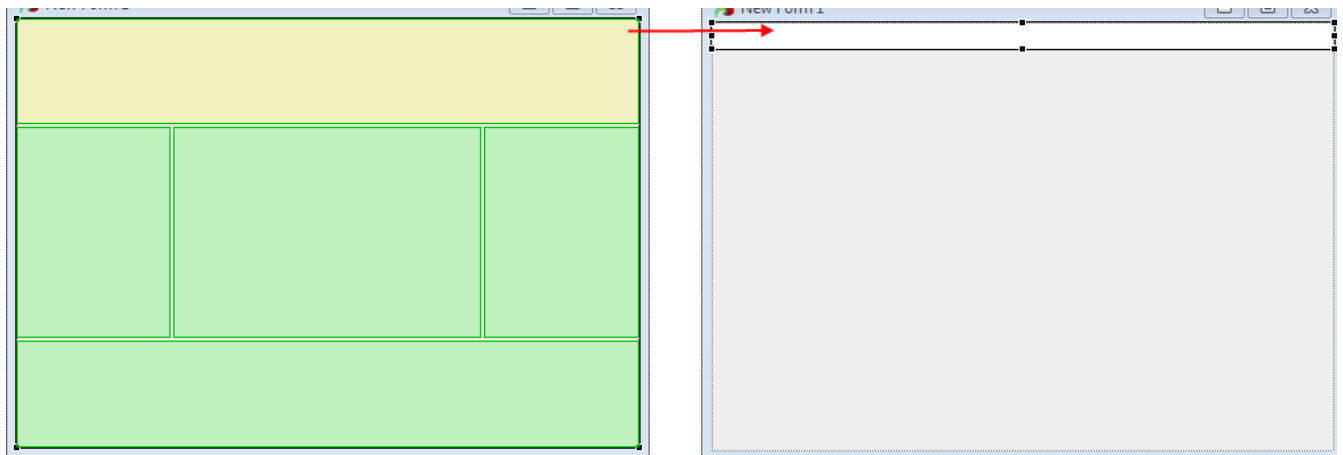
To change the order of the fields in a screen record or delete one or several fields, press the "-" button next to the fields list. This button opens the fields manipulation window:



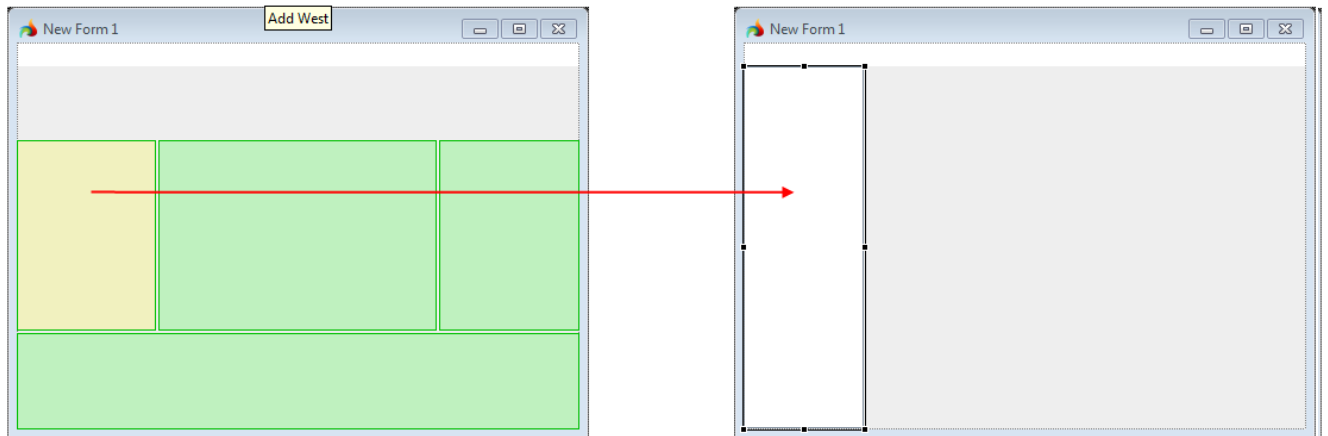
When a field is selected, the field manipulation buttons become activated, and it can be moved up, down, to the top or to the bottom of the list, as well as deleted from the screen record.

Border Panel Container

The objects placed to the Border Panel, are automatically located along the panel borders. When you select a place where a widget is to be placed, the border areas are highlighted. When the widget is dropped, it takes all the available space along the selected border and gets its default height or width:

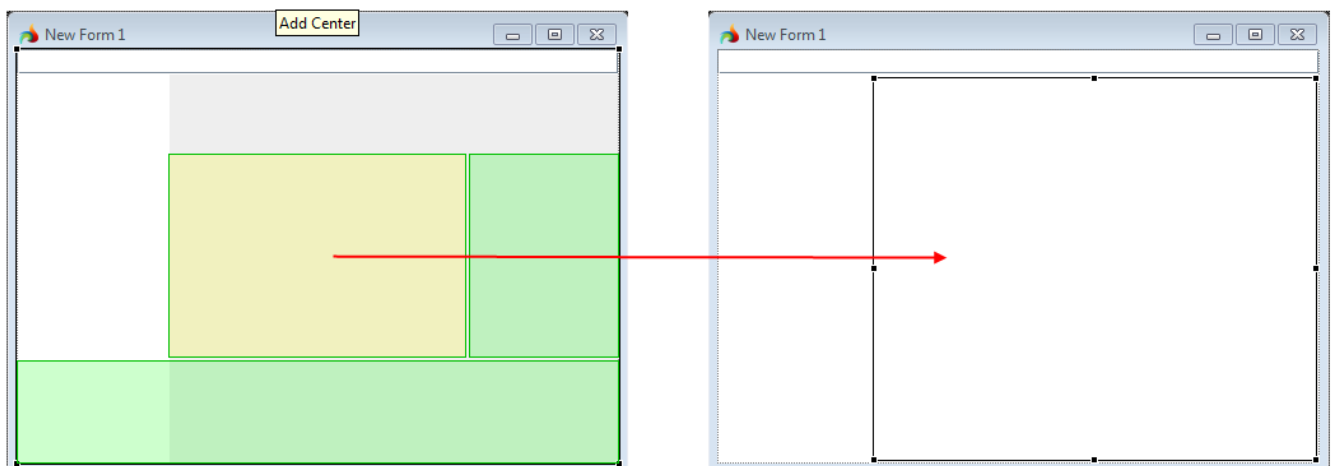


Therefore, if there is a widget already placed at one border, the edge of the widget placed to the neighbour border is defined by the edge of the previously placed one:



You can change the height and the width of the elements in the Border panel by dragging their edges.

When you put an element to a central part of the border panel, it automatically fills all the rest of the container:



There is a set of properties, available for the Border Panel only. They are described below.

Padding

The Padding property of the Border Panel container specifies the distance, in pixels, between the panel border and contents. The Padding property has four possible values to be specified:

- Top
- Bottom
- Right
- Left

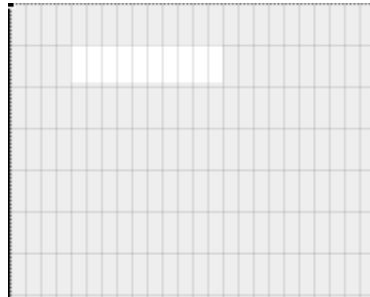


Coord Panel Container

The CoordPanel is the most common container used in forms design. The location of the objects within the Coord Panel is determined by x and y coordinates, which specify the location of the object top left corner on the horizontal and vertical axis, respectively.

The object with coordinates XCoord = 0, YCoord = 0 will be placed at the top left corner of the parent Coord Panel container.

The screenshot below demonstrates a part of a coordpanel with a text field on it:

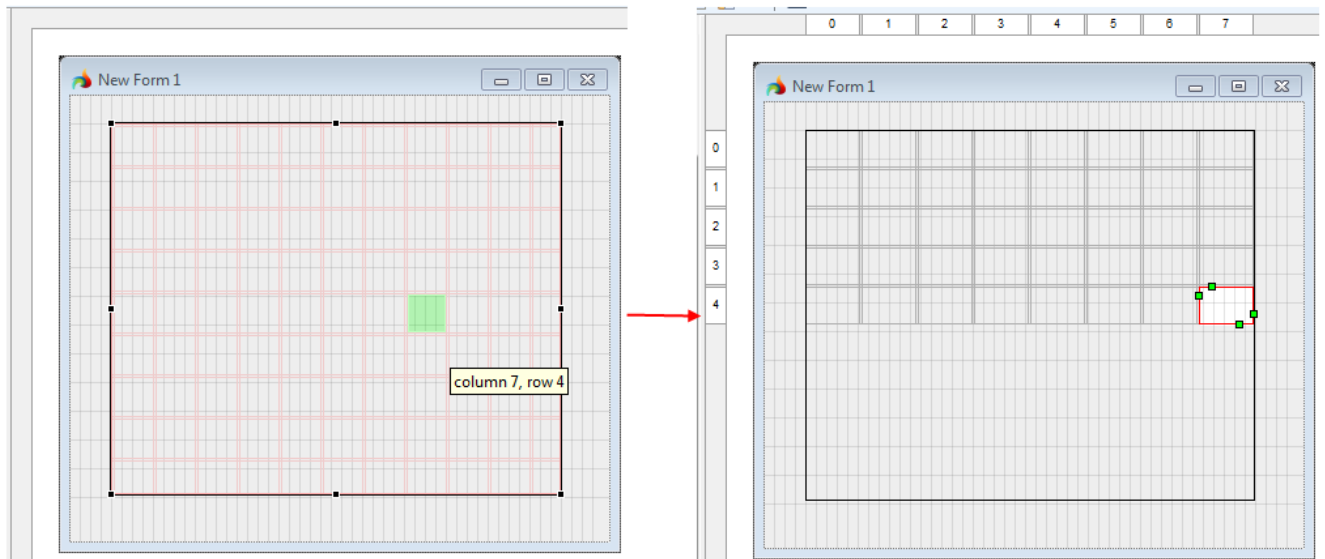


Note, that the Menu, the Toolbar and screen records are placed outside the default form container.

Grid Panel Container

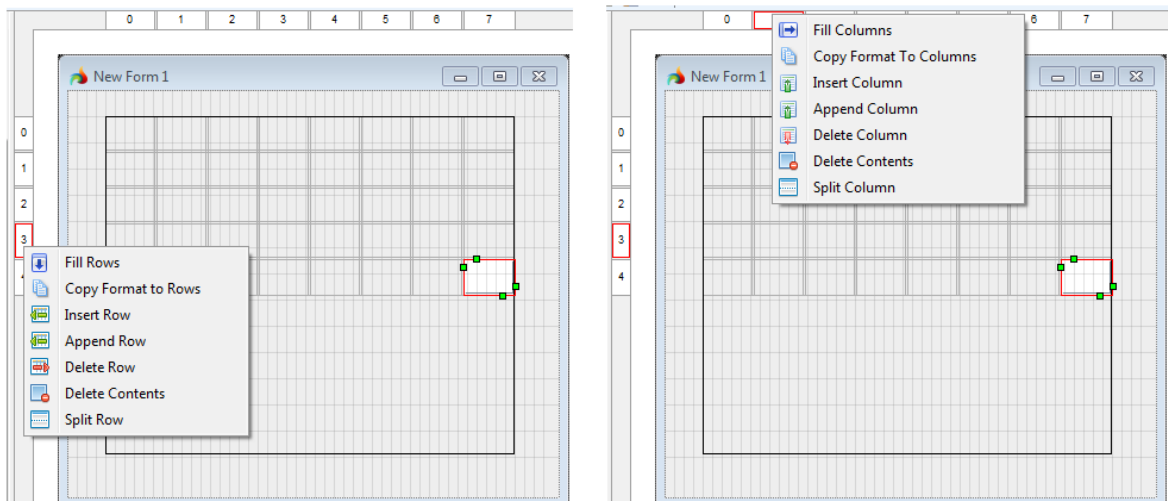
In Grid Panel Container, you can place the objects within a grid. One object can take several cells, and the grid structure allows adjusting the elements sizes and location easily. The grid lines are abstract, i.e., they are visible during the panel manipulations and not visible when the grid panel objects is inactive and in runtime.

Initially, the grid panel includes no rows and columns. When you add the first object, a conditional grid appears. A target location of a new element is highlighted in green. The first object, when dropped to the grid, is placed to the bottom right corner:



As it is seen from the screenshot, when the focus is within the Grid Panel, there are rows and column numbering lines displayed along the edge of the Form Editor area.

You can manipulate the rows and columns from the context menu called by right-clicking on the elements within the numbering line:

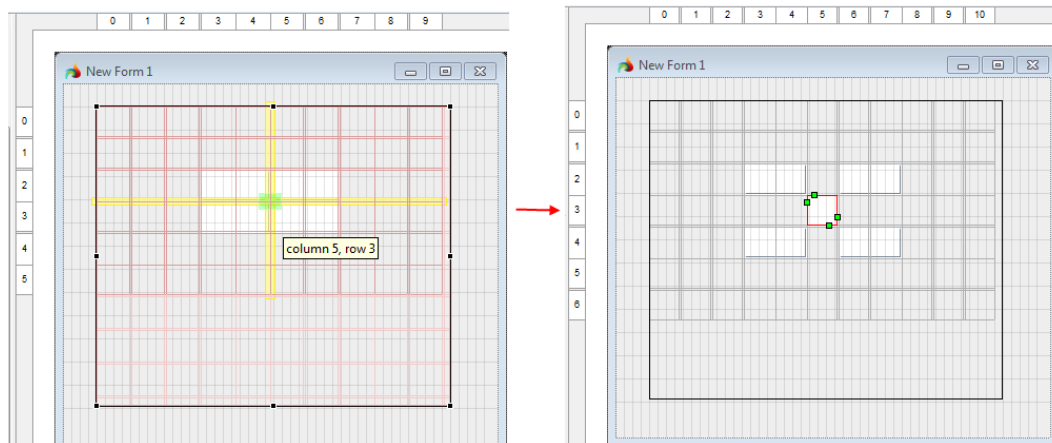


- Fill Rows/Columns: automatically fills the grid with rows or columns if there is an empty space in the container.
- Copy Format to Columns/Rows: copies the GridLength format to all the other columns/rows.
- Insert Column/Row: inserts a new column or row before the selected one.
- Append Column/Row: adds a column to the right edge of the grid or a row to the bottom of it.



- Delete Column/Row: deletes the currently selected column or row.
- Delete Contents: deletes the contents of the currently selected column or row.

You can also insert a new line or column when adding a new element between the existing columns and rows. The target location in this case is highlighted in yellow:



GridItemLocation Property.

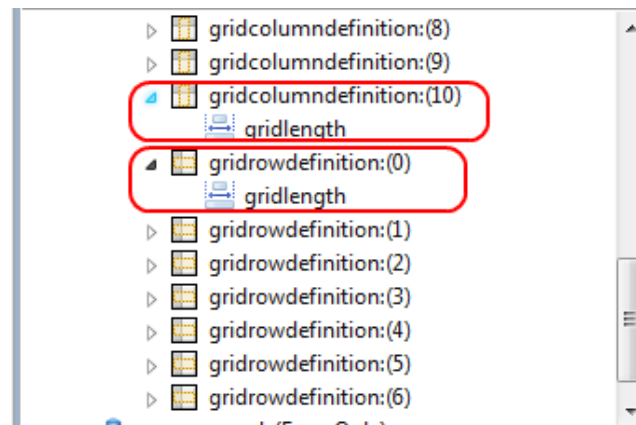
The GridItemLocation property indicates the place and the size of an object located within a Grid Panel container. The property is available for any element placed within a grid. It consists of four sub-properties:

- GridHeight: the number of the rows in grid that the object occupies.
- GridWidth: the number of grid columns that the object occupies.
- GridX: the index of the grid column in which the top left corner of the object is placed.
- GridY: the index of the grid row the object is placed in.

The coordinates of the top left grid cell are 0, 0.

GridLength

The GridLength property is applicable to grid rows and columns and specifies their width and height, respectively. The property can be found in the Structure view and is represented as a child of a row or a column:



The GridLength has a set of four properties:

- GridLengthType
- GridLengthValue
- GridMaxLength
- GridMinLength

Grid Length Type and Value

GridLengthValue property is used to specify the value for the column width or row height. GridLengthType property indicates the way of the length value interpretation. There are three possible GridLengthType values:

- Automatic: at runtime, the width and height of the object will be defined by its contents.
- Absolute: the length value indicates the grid length in pixels.
- Relative: the length value indicates the ratio of the grid. I.e., if all the column relative values are set to 1 they will be of the same length so that they could fill the whole grid area. If there are three columns with 1,2, and 5 relative length values, it means, that the column length will correlate as 1:2:5.

GridMaxLength and GridMinLength

GridMaxLength and GridMinLength properties specify the maximum and minimum length the grid element can have when the grid panel is resized.

Padding

The Padding property of the Grid Panel specifies the distance, in pixels, between the Grid Panel border and its contents. The Padding property has four possible values to be specified:

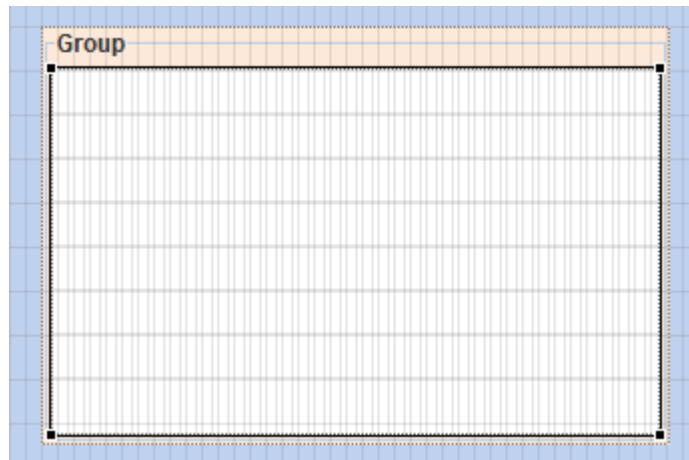
- Top
- Bottom
- Right
- Left



GroupBox Container

The GroupBox container is used to create a titled frame around a form widget. The Group Box can contain only one object, which can be a widget or any other panel with its own children.

On the screenshot below, there is a CoordPanel container placed within a GroupBox in another Coord. The background colours are used to identify the edges of the containers:



Font

The Font property is used to specify the font for the GroupBox title. For the detailed information, see the Font property description in the widget properties section.

ForeColor

The ForeColor property sets the font colour for the GroupBox title. For the detailed information, see the ForeColor property description in the widget properties section.

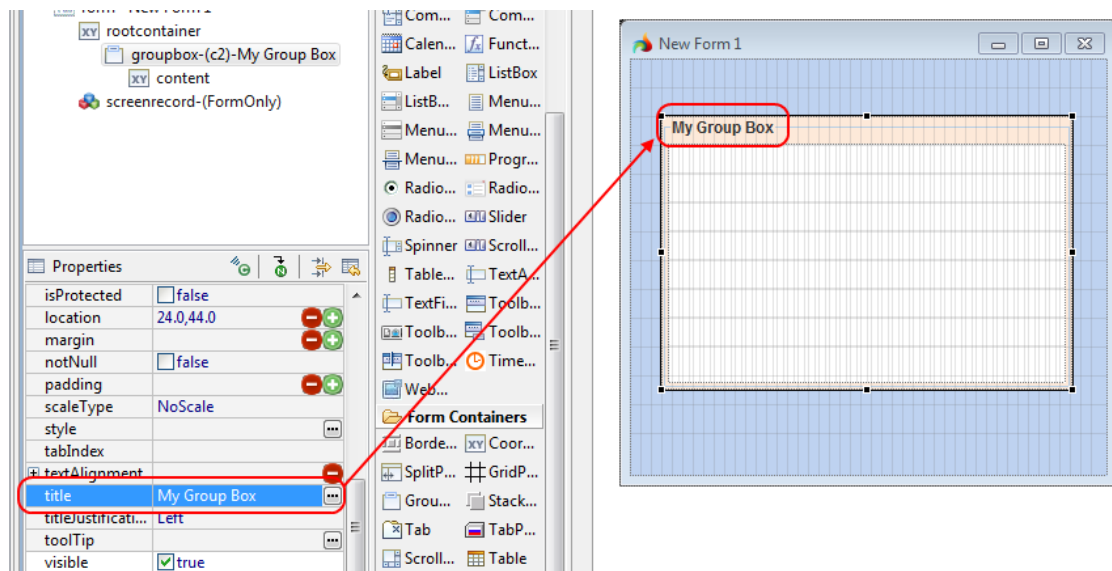
Padding

The Padding property of the GroupBox Panel specifies the distance, in pixels, between the GroupBox Panel border and its contents. The Padding property has four possible values to be specified:

- Top
- Bottom
- Right
- Left

Title

By default, a groupbox has a title displayed at the top of the container. The Title property allows to change the title text:



If the title property is left empty, no title will be displayed.

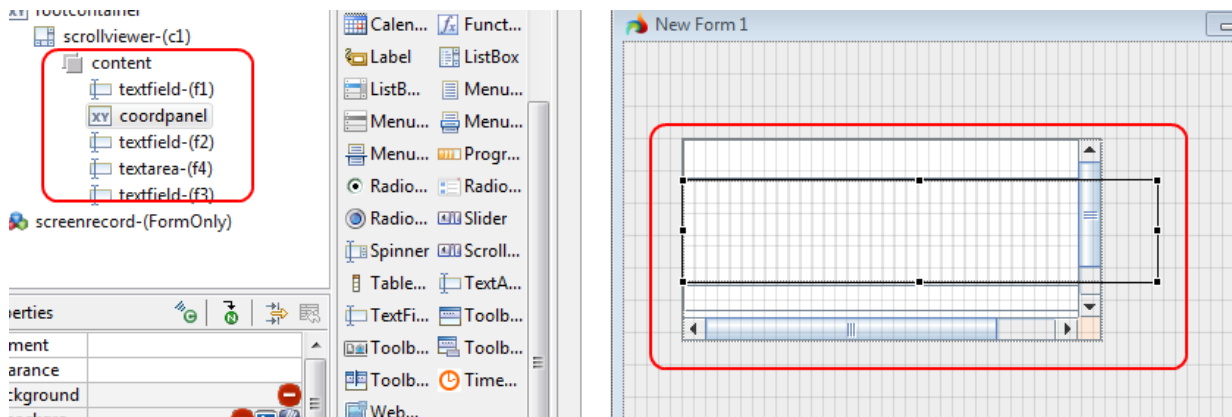
TitleJustification

The default justification of the Groupbox title is Left. The TitleJustification allows to switch between the Left, Center and Right justification for the groupbox:



ScrollView Container

The main feature of the ScrollView is that it can include more actual space than it is limited by the container edges. The container can have only one child, but this child may be represented by another container with its own set of fields. On a screenshot below, a Scroll Panel includes a stack panel with five children, and only three of them fit the scroll panel height. The width of the stack is also bigger than the width of the Scroll Panel, so, there are two scrollbars that will be used at runtime:



By default, the scrollbars appear automatically only in case the ScrollViewer contains elements that are beyond the visible part of the container content. However, the container has two properties that you can use to manipulate the conditions of the scrollbars appearance. They are `HorizontalScrollbarVisibility` and `VerticalScrollbarVisibility`.

Padding

The Padding property specifies the distance (in pixels) between the ScrollViewer edges and its content. The Padding has four sub-properties, specifying this distance for each side of the container:

- Top
- Bottom
- Left
- Right

HorizontalScrollbarVisibility/VerticalScrollbarVisibility

The `HorizontalScrollbarVisibility` and `VerticalScrollbarVisibility` properties of the Scroll Viewer container are used to identify in which situation the horizontal and vertical scrollbars will be visible. Each property has four possible values:

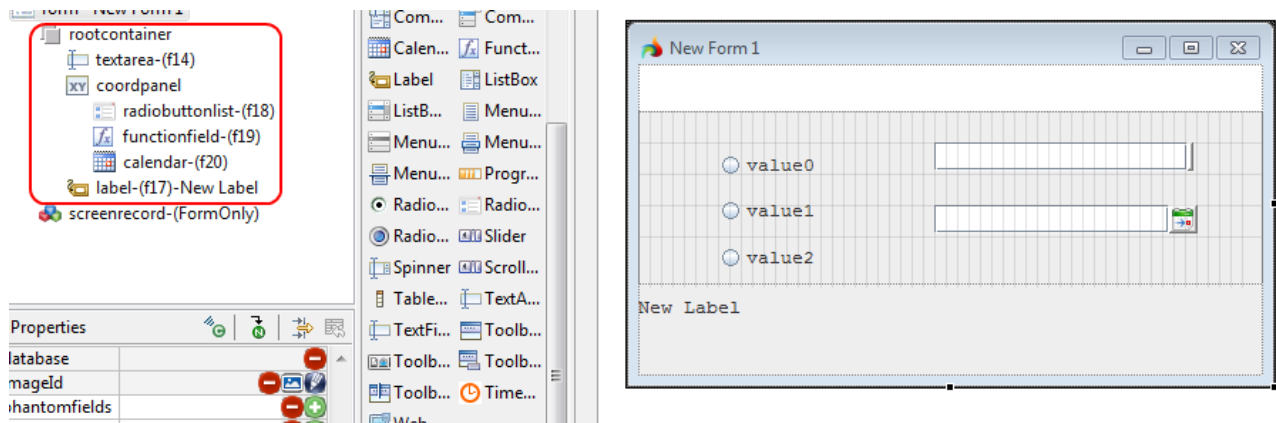
- Auto: the scrollbar appears when the content of the ScrollViewer exceeds its limits.
- Hidden: the scrollbar will not be displayed.
- Visible: the scrollbar will be displayed, even if the content of the ScrollViewer container fits its size.
- Disabled: the scrollbar will be visible but disabled.

Stack Panel Container

The Stack panel is used to create ordered stacks of objects. They are automatically aligned and placed under each other or next to each other, depending on the panel orientation.

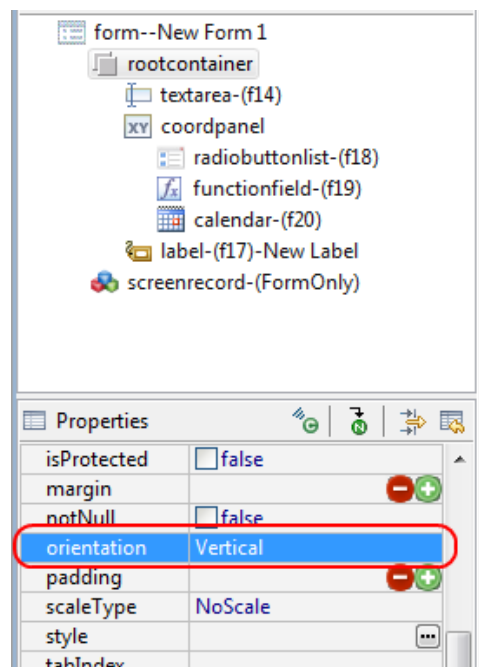
The default stack panel orientation is Vertical, and you can insert a new element at the top of the stack, at the bottom of the stack, or between the existing elements.

As other containers, Stack panel can nest both containers and widgets. The screenshot below shows a stack panel with a test field, a Coord panel with widgets and a label:

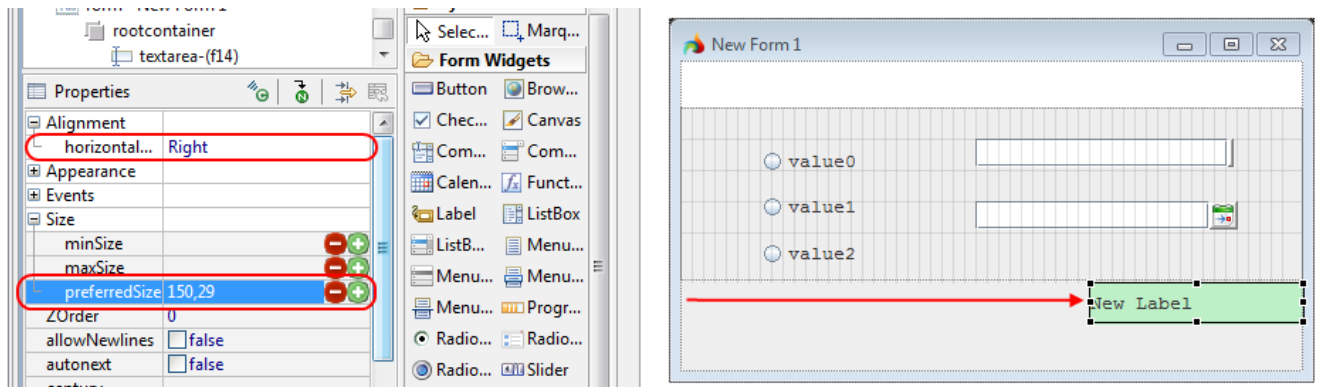


Orientation

The objects in the Stack Panel are placed one under another (vertical orientation) or side by side, in a row (horizontal orientation). The Orientation property is used to change the orientation of the Stack Panel:



By default, the objects in the vertical stack panel are stretched horizontally, and in the horizontal one - vertically. However, you can set up the size with the Preferred Size property. When the needed height/width is set, you can use the Alignment property to specify the object horizontal (for vertical stacks) or vertical (for vertical stacks) position:



Padding

The Padding property specifies the distance, in pixels, between the Stack Panel border and its contents. The Padding property has four possible values to be specified:

- Top
- Bottom
- Right
- Left

Tab Panel Container

The form can contain several tabs that can be easily switched. The tabs are created with the help of the Tab and Tab Page containers.

The Tab container identifies the place on the form on which the tabs will be placed. Note, it does not create any tabs, but allocates place for them. You can see the borders of the Tab space when you add it:



The tab container can have children of only one type - Tab Page containers. They are described further.



Enable Multiline

The Enable Multiline property when set to TRUE allows placing the labels of the tab pages in several lines, if they do not fit one. If this property is set to FALSE (default value), the tab page labels are displayed in one line and can be scrolled.

Padding

The Padding property specifies the distance, in pixels, between the Tab Page Panel border and its contents. The Padding property has four possible values to be specified:

- Top
- Bottom
- Right
- Left

Tab Page Placement

By default, the tabs with tab pages labels are placed one by one along the Top border of the tab container. You can change their location using the TabPagePlacement property. It accepts four values:

- Top (default): placed tabs along the top edge of the tab container.
- Left: placed tabs along the left edge of the tab container.
- Right: placed tabs along the right edge of the tab container.
- Bottom: placed tabs along the bottom edge of the tab container.

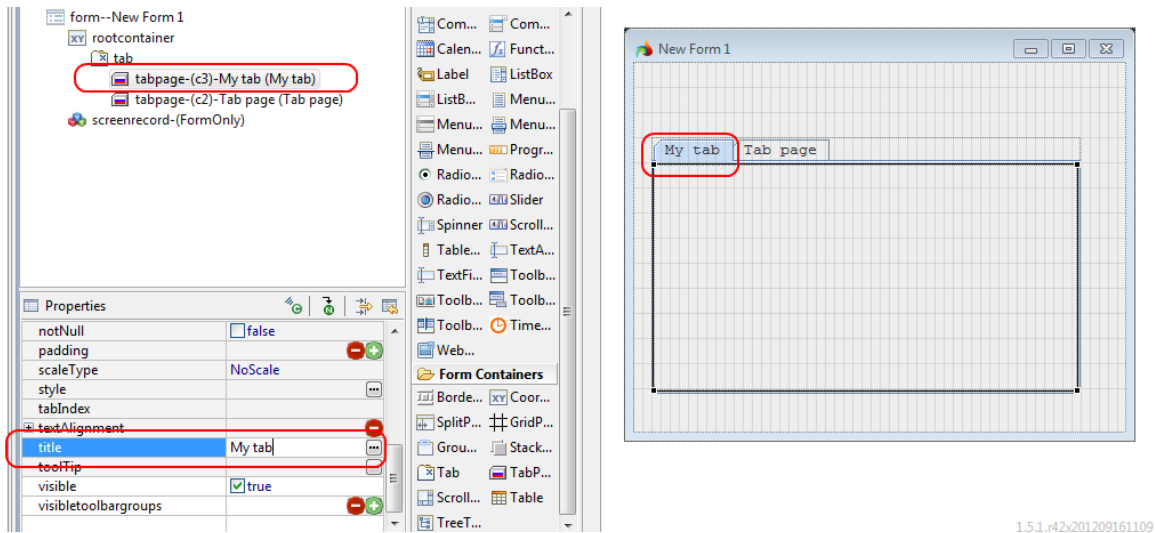
OnSelectedTabPageChanged Event

The OnSelectedTabPageChanged event is an event available for Tab container object. It specifies an action, triggered each time the user switches between tab pages.

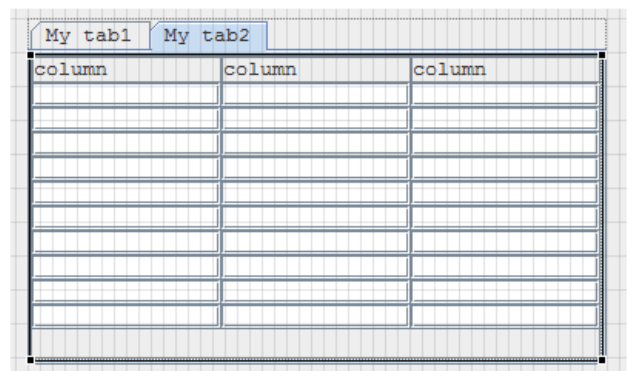
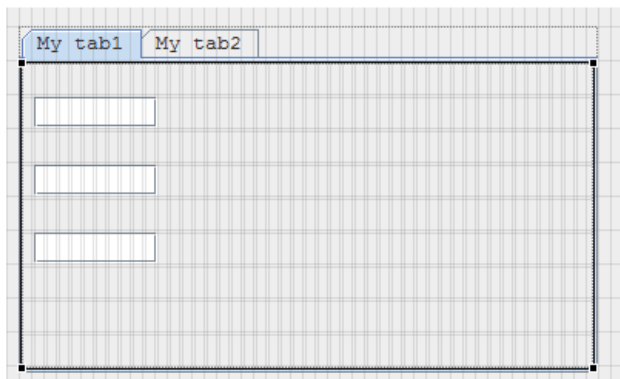


TabPage Container

The TabPages are to be placed within the Tab container. Each TabPage has a label and a body. The label is used to switch to the tab. The label is set by the Title property of the tab page:



The tab page body can include one child, which is typically represented by another container. The content of a tab page does not depend on the content of other tab pages.



Font

The Font property specifies the font settings for the tab page label. For the details, see the Font property description in the widget properties description section.

ForeColor

The ForeColor property specifies the font colour for the tab page label. For the details, see the ForeColor property description in the widget properties description section.



Image

The Image property specifies the path to an image that will be displayed to the tab label. The image specification is the same as in any other Image property (See the Image property description in the widget properties section).

OnSelectTabPage Event

The OnSelectTabPage event is an event available for Tab Page container object. It specifies an action, triggered each time the user selects the tab page with this event being specified.

Table Container

The Table container is designed to provide you with the ability to create tables for data input and display.

A form table has a hierarchical structure, in which the Table container is the main Parent object.

The Table container is filled with table columns, each of them having a form widget inside.

Table identifier is linked to the identifier of the screen record nesting the table column fields. Therefore, editing one of these IDs automatically changes the other.

CellPadding

The CellPadding property specifies the distance between the table cells.

GridColor

The GridColor property defines the color of the lines of the table grid. You can specify either System or Customized color as the property value.

HorizontalScrollbarVisibility

The HorizontalScrollbarVisibility property specifies the display mode for the horizontal toolbar. The following options are available:

- Auto - the toolbar is displayed in case the actual width of the table content exceeds the form width
- Hidden - the toolbar is hidden and not displayed, even if the form content width is larger than the table width
- Visible - the toolbar is visible, even if the content fits the table
- Disabled - the toolbar is visible, but disabled

Identifier

The Identifier property specifies the table identifier. It is important to make difference between this value and the name of the program array, bound to the table. It is the latter which is referenced by INPUT/DISPLAY ARRAY statements.



Is Multiselect

IsMultiselect property, when set to TRUE, allows the user to select several rows in time during the data display. To select a continuous set of rows, the user should select a row and then another one, keeping the Shift button pressed. To select separate rows, one should keep the Ctrl button pressed.

Is Protected

The Is Protected property, when set to TRUE, indicates, that the table and its content will overlap any other object that was or will be displayed within the table area.

Is Virtual

Is Virtual property, when its checkbox is ticked, makes a grid virtualized.

Margin

The Margin property specifies the margins within the table.

Padding

The Padding property specifies the distance between the table edges and its contents.

RowCount

The property is used to specify the number of the screen records comprising one page of the screen array.

RowHeight

The property is used to specify the height of the table rows, in pixels.

RowResizable

The property allows to resize the rows at runtime.

TextAlignment

Specifies the horizontal and vertical text alignment for the whole text within the table

VerticalScrollbarVisibility

The VerticalScrollbarVisibility property specifies the display mode for the vertical toolbar. The following options are available:

- Auto - the toolbar is displayed in case the actual height of the table content exceeds the form width
- Hidden - the toolbar is hidden and not displayed, even if the form content height is larger than the table height
- Visible - the toolbar is visible, even if the content fits the table
- Disabled - the toolbar is visible, but disabled

Visible

The Visible property, when unticked, makes the table be hidden.

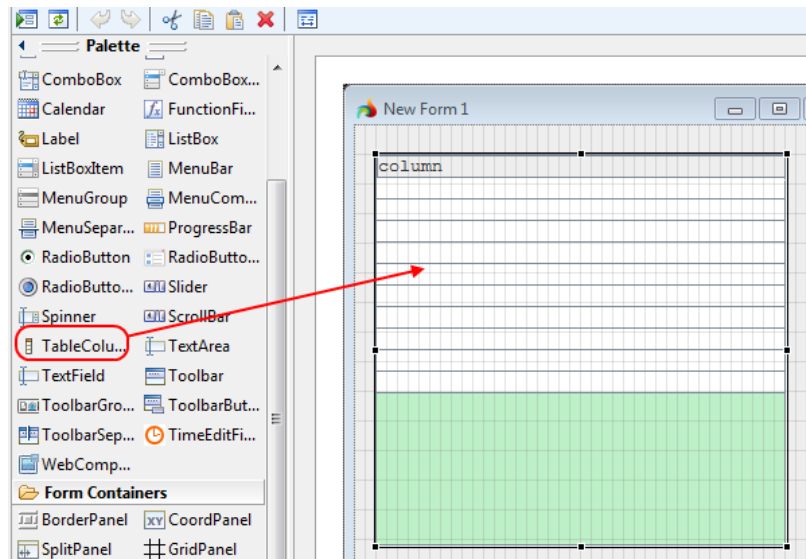


VisibleToolbarGroups

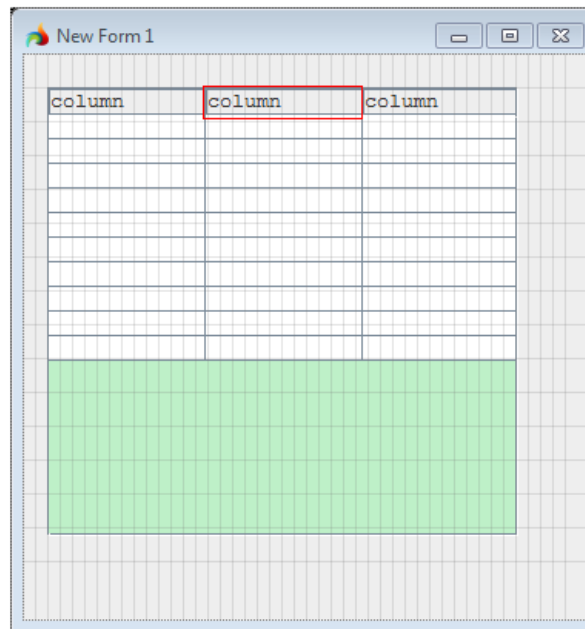
Allows to specify the toolbar groups that will be visible when the cursor is placed within the table.

Table Column

The Table Column objects can be used only within the table container. To add a column to the Table, select the Column object in the list of the widgets and drop it to the table. The column will fill the whole width of the table.



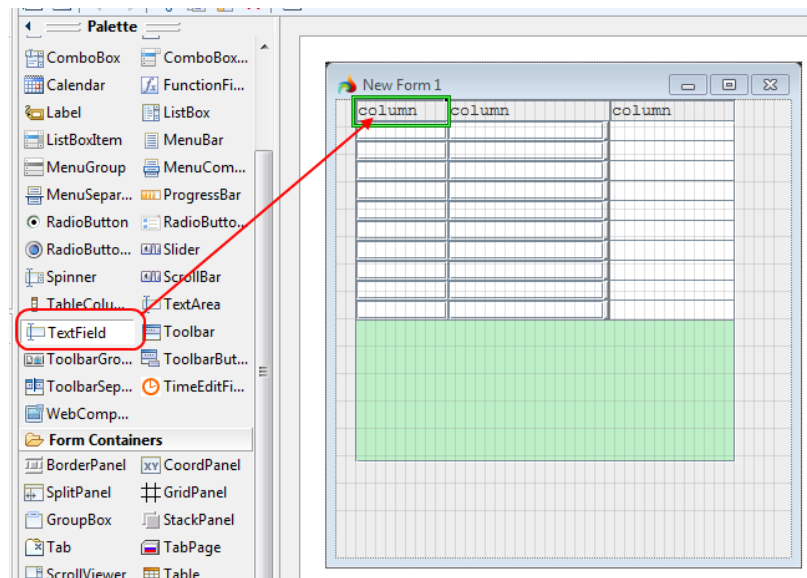
When you add other columns, they automatically fill the table width.



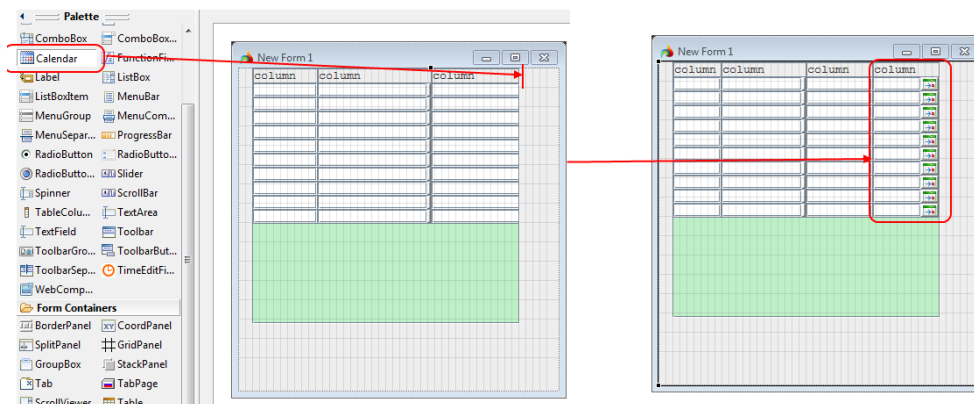


When the necessary number of columns is added, you can manually change their width by dragging the right and left edges of their headers.

Each column should contain only one field. To add a field to the column, select the necessary item from the palette and drop it to the target column header.

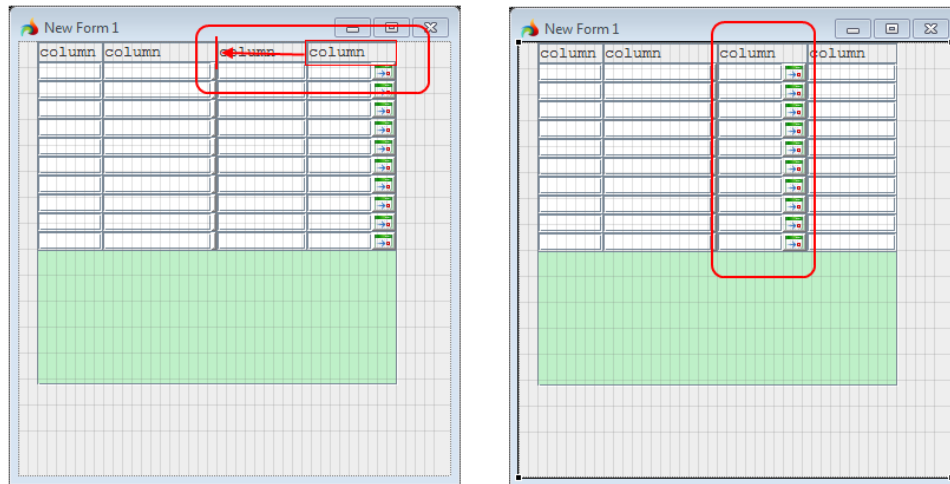


It is also possible to add a column together with a field in it. To do it, select a field of the necessary type and put in the headers area, to the place, where the field should be added to:

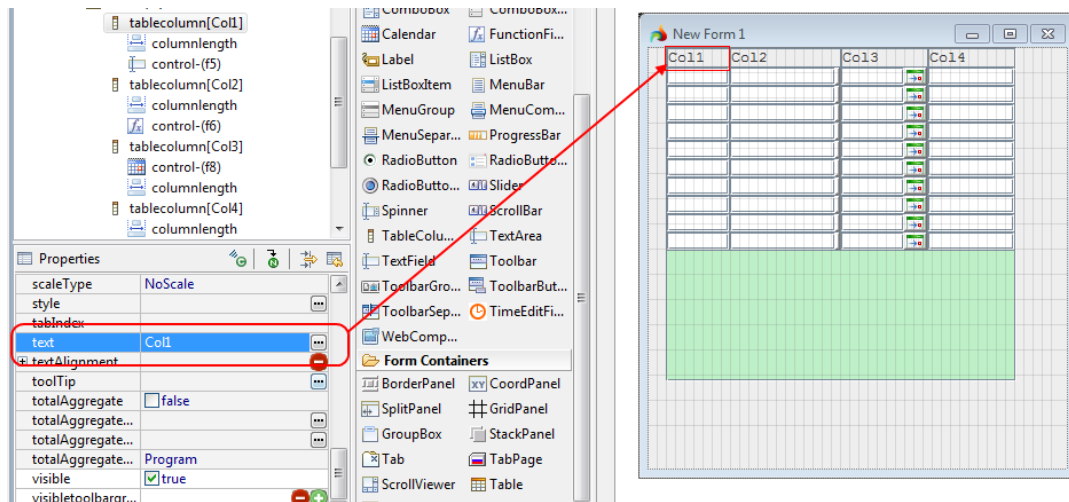




You can rearrange the columns order by dragging the columns headers and dropping them in the necessary places:

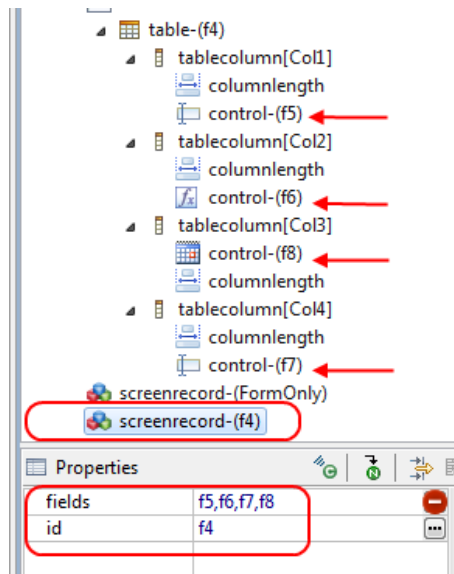


You can specify a column header by setting the Text property of the column:



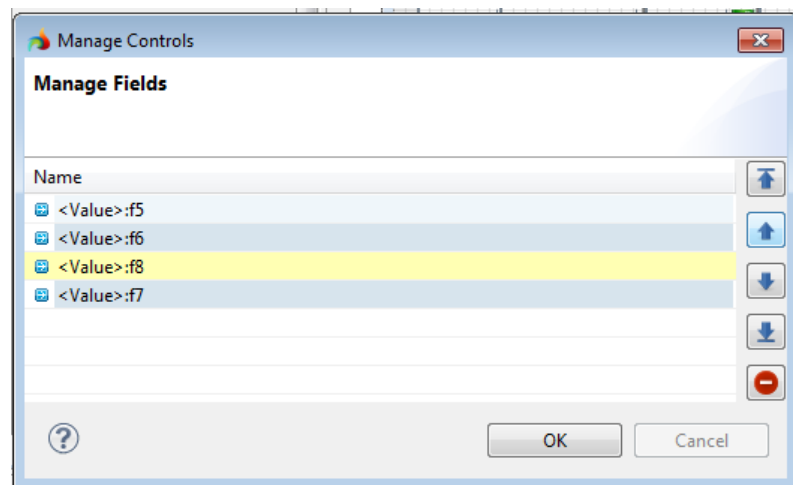


Lycia window Builder automatically creates a screen record bound to the table, and adds there all the fields added to this table:



It is the screen record name that is to be referenced in 4GL to display or input arrays.

When you change the order of the table columns, the order of the column field names is not changed in the corresponding screen record. Therefore, after the columns are finally arranged, it may be necessary to re-arrange the screen record list. To do it, click on the "-" button to the right of the fields list. This will call a dialog window, in which the record fields can be moved or deleted.



Font

The Font property is used to specify the font settings for the column header.

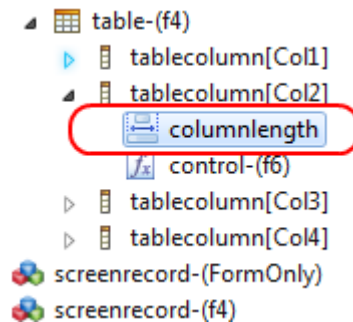


ForeColor

The ForeColor property is used to specify the forecolor for the column header.

ColumnLength

The ColumnLength property is applicable to table columns and specifies their width. The property can be found in the Structure view and is represented as a child of a column:



The GridLength has a set of four properties:

- GridLengthType
- GridLengthValue
- GridMaxLength
- GridMinLength

Grid Length Type and Value

GridLengthValue property is used to specify the width of the column. GridLengthType property indicates the way of the Grid length value interpretation. There are three possible GridLengthType values:

- Automatic - at runtime, the width of the column will be defined by its contents
- Absolute - the length value indicates the column length in pixels.
- Relative - the length value indicates the ratio of the grid. I.e, if all the columns relative values are set to 1, they will be of the same length so that they could fill the whole grid area. If there are three columns with 1,2, and 5 relative length values, it means, that the column length will correlate as 1:2:5

Text

The Text specifies the column header.

Aggregate properties

The table Aggregate properties, allow to apply an aggregate function on group-level. When the data in the table are grouped, the Aggregate property displays the aggregate function result to the group name line.

To make the program apply aggregate function to a column, do the following steps:



- Set the Aggregate property to True
- Set the AggregateType property. It specifies which of the aggregate functions will be applied to the column. The possible are:
 - Sum (Display the sum of the group values)
 - Avg (Display the average of the group values)
 - Min (Display the minimum of the group values)
 - Max (Display the maximum of the group values)
 - Count (Display the number of the group entries)
 - Grouped (Make the table values be automatically grouped by the column)
- Set the AggregateText property. It defines which string will be displayed before the result. For example, it can be "The average salary: " for the Average aggregate function

It is necessary that the data type of the column corresponds to the data type of the variable displayed to it. Otherwise, it is highly possible that the Aggregate property works incorrectly and displays no values.

Here is a table grouped by the Age column with the MaxAggregate property specified for the Salary column:

Grouped by: Age			
	Name	Age	Salary
▼ 32	Max Salary: 3200		
▼ 34	Max Salary: 3200		
▼ 45	Max Salary: 1400		
▼ 56	Max Salary: 5000		

It should be mentioned, that these aggregates should be used only when the table is not marked as Virtual.

TotalAggregate properties

The TotalAggregate properties are used to specify an aggregate function which will be applied to the column. The result of the function execution will be displayed to the automatically added row at the bottom of the table.

To make the program apply an aggregate function to a column, do the following steps:

- Set the TotalAggregate property to True
- Set the TotalAggregateText property. It defines which string will be displayed before the result. For example, it can be "Total: " for the Sum aggregate function.
- Set the TotalAggregateType property. It specifies which of the aggregate functions will be applied to the column. The possible are:
 - Sum (Display the sum of the group values)
 - Avg (Display the average of the group values)
 - Min (Display the minimum of the group values)
 - Max (Display the maximum of the group values)



- Count (Display the number of the group entries)
- Program (Displays the value passed from the 4GL routine)
- Set the TotalAggregateName property in case the TotalAggregateType is set to "Program"

The screenshot below shows how the standard TotalAggregate function (Sum) result is displayed:

Name	Age	Salary
John White	34	1000
Jill Black	32	1400
Alice Brown	34	1400
Tallulah Gre	56	5000
Alex Red	34	3200
Jason Violet	32	3200
Tommy Magent	56	5000
Lucy Brown	45	1000
Timoty Silve	45	1400
Joahn Beige	32	3200
		Total: 25800

The Program total aggregate function is designed to allow the programmer pass their own values to the Total Aggregate line in case none of the default aggregate functions fit their requirements and needs.

To make the Program aggregate property work, specify the TotalAggregateName property. It defines the identifier by which the total aggregate field of the column will be referenced by 4GL.

Then, you can reference this name in your DISPLAY statements. For example, if you specified the TotalAggregateName as "f_coefficient", you can pass a value there using a command like this:

```
DISPLAY sal_cf TO f_coefficient
```

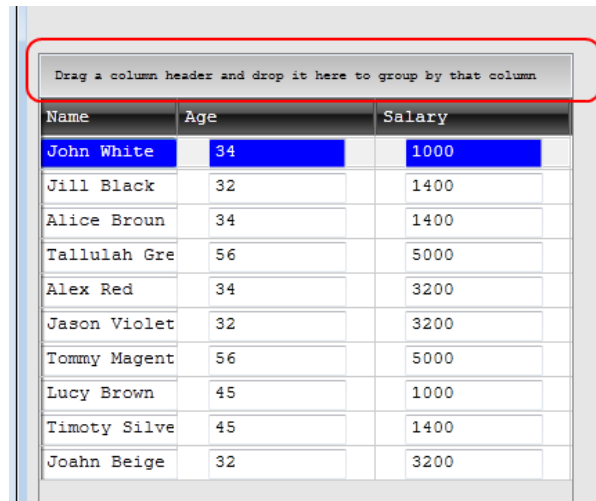
The line will display the value of the variable *sal_cf* to the column total aggregate field

IsGroupable

The IsGroupable property allows to group the data in the table according to the repeating values of the column in which the property is set to TRUE.

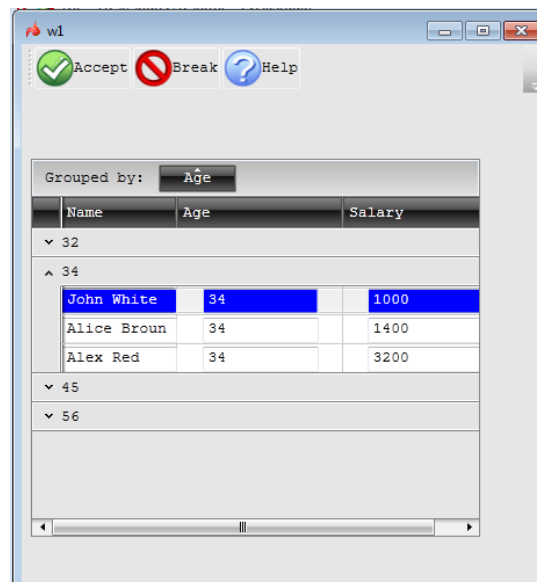


When the property is set to TRUE at least for one column, a group line is added to the whole table in the runtime.



Name	Age	Salary
John White	34	1000
Jill Black	32	1400
Alice Broun	34	1400
Tallulah Gre	56	5000
Alex Red	34	3200
Jason Violet	32	3200
Tommy Magent	56	5000
Lucy Brown	45	1000
Timoty Silve	45	1400
Joahn Beige	32	3200

To group the data in the table, drag and drop the header of the column with the Is Groupable property set to true. This will change the table layout. The table will display the group names only, each name identifying the value that became the basis for the group creation. Click on the group line to unfold it and see the entries containing the grouping value:



Grouped by: Age

Name	Age	Salary
32		
34		
John White	34	1000
Alice Broun	34	1400
Alex Red	34	3200
45		
56		

To cancel the grouping, drag the group name in the sorting line and drop it to the table.

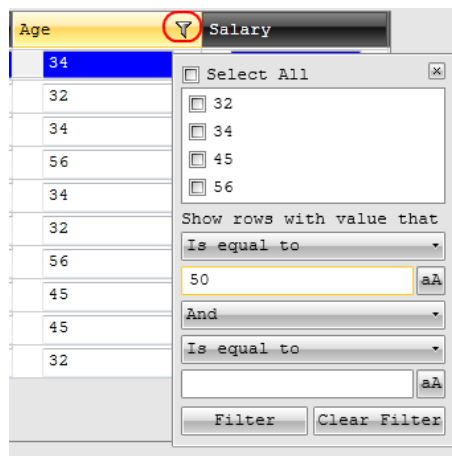
Note, that the property is applied only when the column headers are specified.



IsFilterable

The IsFilterable property, when set to True, allows to filter the values displayed to the table, according to some criterion. At runtime, there is a Filter icon in the header of the column with IsFilterable property specified to true.

When the user clicks the icon, the Filter dialog box opens. Here one can input the filter criteria:



When the filter is applied, the table display only the where the values in the selected column correspond to the filter criteria.

Note, that the property is applied only when the column headers are specified.

IsSortable

When the IsSortable property is set to True, the user can click the column header to sort the values in the table. The first click sorts the values in the ascending order, the second one - in descending. The third click resets the sorting and makes the lines be displayed in default order. Note, that the sorting changes only the visual display of the table and does not influence the order of records in the program array.

Movable

The Movable property, when set to true, allows the user to move the column within the table at runtime. To move a column, drag its header and drop to the header of the column which should be replaced. The replaced column should also have the Movable property set to true. The dropped column will take its new place, and the target column will move right.



Name	Age	Salary
John White	34	1000
Jill Black	32	1400
Alice Broun	34	1400
Tallulah Gre	56	5000
Alex Red	34	3200
Jason Violet	32	3200
Tommy Magent	56	5000
Lucy Brown	45	1000
Timoty Silve	45	1400
Joahn Beige	32	3200

Salary	Name	Age
1000	John White	34
1400	Jill Black	32
1400	Alice Broun	34
5000	Tallulah Gre	56
3200	Alex Red	34
3200	Jason Violet	32
5000	Tommy Magent	56
1000	Lucy Brown	45
1400	Timoty Silve	45
3200	Joahn Beige	32

Tree Table Container

The Tree Table container is a container that allows to create tree tables without having to specify additional theme properties and settings.

As well as a standard table, the Tree table consists of columns, each having a field in it. Tree Table has all the properties the standard table has and acquires some specific ones. Unlike the Table container, the Tree table includes the columns that are displayed, the columns that regulate the parent-child relations between the nodes, and the columns which specify additional settings - e.g., identify, whether the parent node is expanded or collapsed by default. The purpose of the columns is specified by means of special Tree Table properties, which are:

- Column Edit
- ColumnId
- ColumnIsNode
- ColumnParentId
- Column Expanded
- ColumnImage
- ImageCollapsed
- ImageExanded
- ImageLeaf

The values of these properties should be the names of the table columns, in which the corresponding node info is stored.



ColumnId and ColumnParentId properties

The basic tree form should include four columns (the column names in the example are used to illustrate the columns purpose and their interrelation with the program array):

ParentNode	ChildNode	ColumnID	ParentId

Parent and Child node columns are those that actually contain the information displayed to the tree. The parent node column contains the names of the brunches and the control buttons to expand and collapse them. The Child node column contains the brunches' details. The first column is by default treated as the Parent Node column.

The following program array will be displayed to the table given on the screenshot above. The given names for the record elements are used to make it easier to associate the program array with the tree table:

```
LET ar[1].nodename = "name-1"  
LET ar[1].nodedescr = "descr-1"  
LET ar[1].id = 1  
LET ar[1].parentid = NULL  
  
LET ar[2].nodename = "name-1.1"  
LET ar[2].nodedescr = "descr-1.1"  
LET ar[2].id = "1.1"  
LET ar[2].parentid = 1  
  
LET ar[3].nodename = "name-1.2"  
LET ar[3].nodedescr = "descr-1.2"  
LET ar[3].id = "1.2"  
LET ar[3].parentid = 1
```



```
LET ar[4].nodename = "name-2"
LET ar[4].nodedescr = "descr-2"
LET ar[4].id = 2
LET ar[4].parentid = null

LET ar[5].nodename = "name-2.1"
LET ar[5].nodedescr = "descr-2.1"
LET ar[5].id = "2.1"
LET ar[5].parentid = 2

LET ar[6].nodename = "name-2.2"
LET ar[6].nodedescr = "descr-2.2"
LET ar[6].id = "2.2"
LET ar[6].parentid = 2

LET ar[7].nodename = "name-2.2.1"
LET ar[7].nodedescr = "descr-2.2.1"
LET ar[7].id = "2.2.1"
LET ar[7].parentid = "2.2"

LET ar[8].nodename = "name-2.3"
LET ar[8].nodedescr = "descr-2.3"
LET ar[8].id = "2.3"
LET ar[8].parentid = 2
```

The values in the screen array directly influence the way the tree table will be organized at runtime.

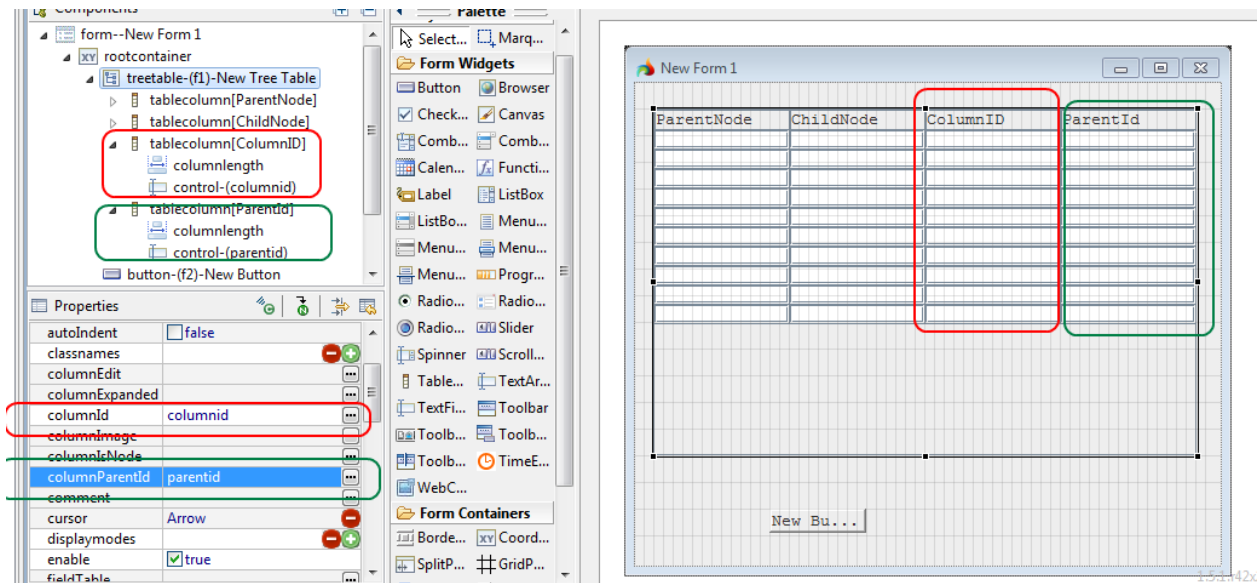
The ColumnId property specifies the name of the column, in which the IDs of the nodes are listed. Each node should have its unique ID, by which it can be referenced (arr.id element in the example).

The ColumnParentId tree table property is used to specify the ID of a node, parent for the current one. This should be one of the ID's of the other nodes, present in the table. If you want a node to have several children, specify it's identifier in the parent id element of the respective records (arr.parentid in the example).

If the value in the ParentID column is set to NULL , the node will become a root node.



Therefore, only two properties are needed to establish a tree table: they are ColumnID and ColumnParentId:



The screenshot below shows how the table is displayed when these properties are set. By default, all the parent nodes are collapsed:

ParentNode	ChildNode
+ name-1	descr-1
+ name-2	descr-2



You can expand them by clicking the "+" button to the left of the parent node name:

ParentNode	ChildNode
[-] name-1	descr-1
name-1.1	descr-1.1
name-1.2	descr-1.2
[-] name-2	descr-2
name-2.1	descr-2.1
[-] name-2.2	descr-2.2
name-2.2.1	descr-2.2.
name-2.3	descr-2.3

ColumnEdit

The property is used to specify a column to which the parent nodes are displayed. If this property is empty, it is the first column which is treated as the Edit column.

In the example below, we swapped the Child and Parent node columns in the form layout and run the application without adding any new properties:

New Form 1

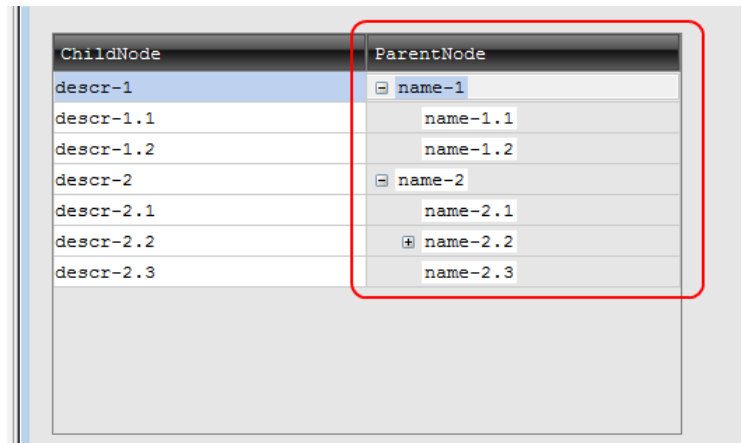
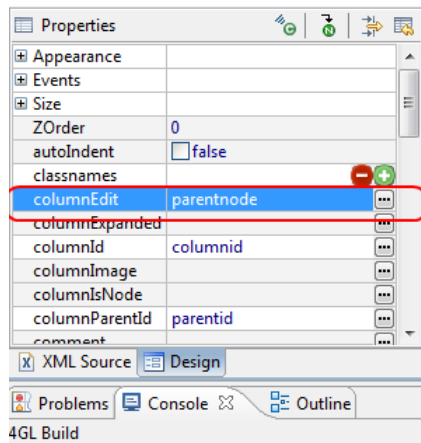
ChildNode	ParentNode	ColumnID	ParentId

ChildNode	ParentNode
[-] descr-1	name-1
descr-1.1	name-1.1
descr-1.2	name-1.2
[-] descr-2	name-2
descr-2.1	name-2.1
[-] descr-2.2	name-2.2
descr-2.2.	name-2.2.1
descr-2.3	name-2.3

As can be seen, the branches names and descriptions also swapped.



When the ColumnEdit property is set to "parentnode" (the ID of the column with parent nodes names), the right column becomes the parent nodes holder:



ColumnExpanded

As it has been mentioned above, by default, the tree table is displayed with all the parent nodes being collapsed.

ColumnExpanded property is used to set the ID of the column, which contains Boolean values describing whether the node should be initially collapsed (FALSE) or expanded (TRUE). This property needs a special column in the tree table and a special element of the program records:

```

MAIN
DEFINE ar: DYNAMIC ARRAY OF RECORD
  nodename, nodedescr: char(10),
  id, parentid: string,
  isexpanded: Boolean
END RECORD

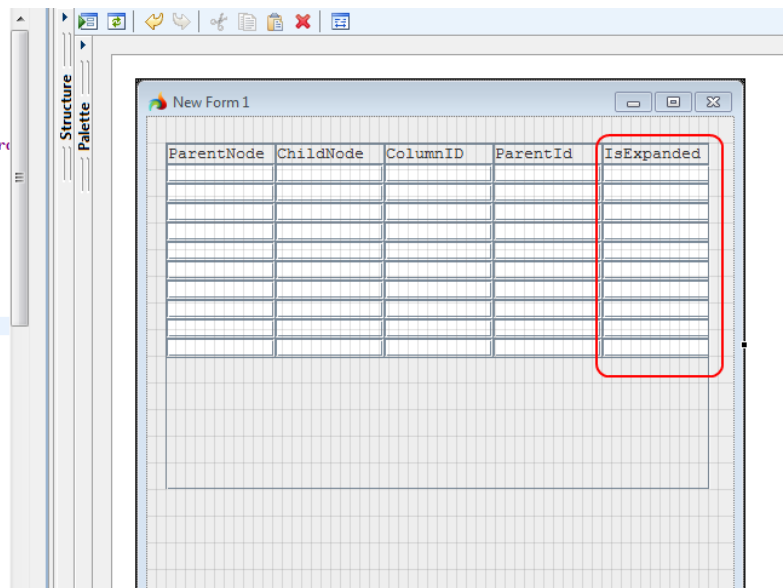
OPEN WINDOW w1 AT 2,2 WITH FORM "column_exp" ATTRIBUTE (border)

LET ar[1].nodename = "name-1"
LET ar[1].nodedescr = "descr-1"
LET ar[1].id = 1
LET ar[1].parentid = NULL
LET ar[1].isexpanded = 1 ←

LET ar[2].nodename = "name-1.1"
LET ar[2].nodedescr = "descr-1.1"
LET ar[2].id = "1.1"
LET ar[2].parentid = 1
LET ar[2].isexpanded = 0 ←

LET ar[3].nodename = "name-1.2"
LET ar[3].nodedescr = "descr-1.2"
LET ar[3].id = "1.2"
LET ar[3].parentid = 1
LET ar[3].isexpanded = 0 ←

LET ar[4].nodename = "name-2"
LET ar[4].nodedescr = "descr-2"
LET ar[4].id = 2
LET ar[4].parentid = null
LET ar[4].isexpanded = 1 ←
  
```





In the example above, we specified the first and the fourth node as expanded (1 or TRUE). These are the top-level nodes. Other nodes that have children will be initially collapsed:

```

LET ar[1].nodename = "name-1"
LET ar[1].nodedescr = "descr-1"
LET ar[1].id = 1
LET ar[1].parentid = NULL
LET ar[1].isexpanded = 1

LET ar[2].nodename = "name-1.1"
LET ar[2].nodedescr = "descr-1.1"
LET ar[2].id = "1.1"
LET ar[2].parentid = 1
LET ar[2].isexpanded = 0

LET ar[3].nodename = "name-1.2"
LET ar[3].nodedescr = "descr-1.2"
LET ar[3].id = "1.2"
LET ar[3].parentid = 1
LET ar[3].isexpanded = 0

LET ar[4].nodename = "name-2"
LET ar[4].nodedescr = "descr-2"
LET ar[4].id = 2
LET ar[4].parentid = null
LET ar[4].isexpanded = 1

LET ar[5].nodename = "name-2.1"

```

ParentNode	ChildNode
[-] name-1	descr-1
name-1.1	descr-1.1
name-1.2	descr-1.2
[-] name-2	descr-2
name-2.1	descr-2.1
name-2.2	descr-2.2
name-2.3	descr-2.3

ColumnImage

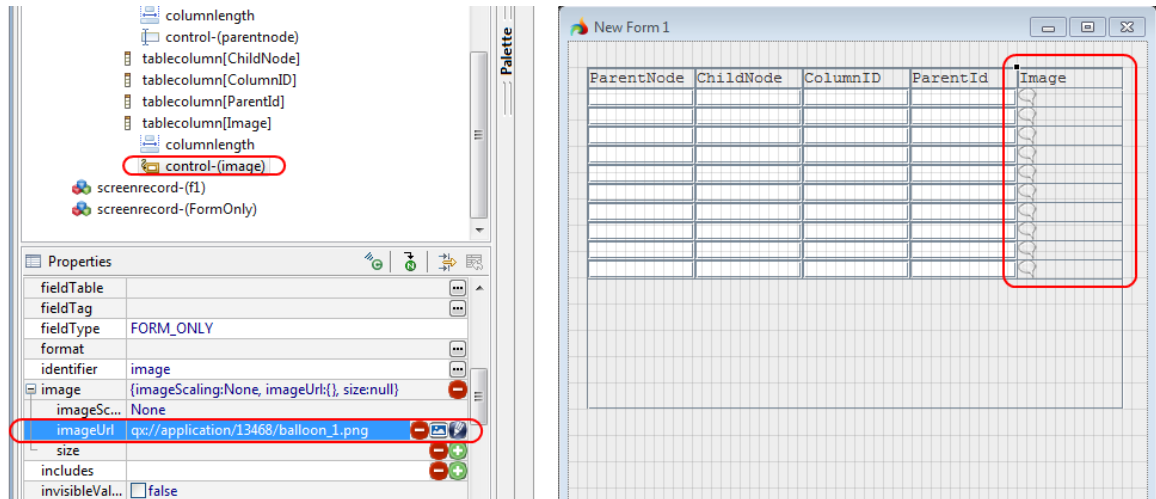
The property specifies a name for a column listing the paths to the pictures to be displayed near the node names:

The Properties panel for a table widget shows the following properties: ZOrder, autoIndent, classNames, columnEdit, columnExp..., columnId, and columnImage. The 'columnImage' property is highlighted with a red box and set to the value 'image'.

A screenshot of a table widget in a form designer. The table has five columns: ParentNode, ChildNode, ColumnID, ParentId, and Image. The 'Image' column is highlighted with a red box.

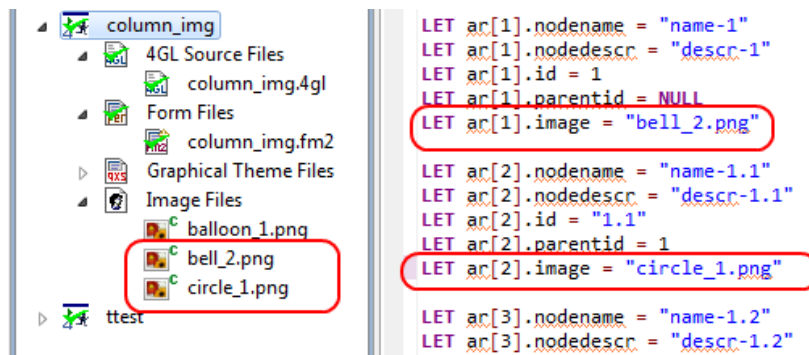


As with the ColumnExpanded, the ColumnImage property needs a special column to be added to the table. The column should consist of label widgets without text, but with some initial picture:



This picture won't be displayed at runtime and is needed only to indicate that the label should be treated as an image, not as a text.

The image that should be displayed to the left of the tree nodes, is specified in the program array as an image name and path, if needed. In the example, we used different pictures for parent and child nodes. The pictures should be added to the application requirements.:



ParentNode	ChildNode
[-] 📁 name-1	descr-1
🔍 name-1.1	descr-1.1
🔍 name-1.2	descr-1.2
[-] 📁 name-2	descr-2
🔍 name-2.1	descr-2.1
[-] 📁 name-2.2	descr-2.2
🔍 name-2.2.1	descr-2.2.
🔍 name-2.3	descr-2.3

The property specifies a name for a column, listing the Boolean values indicating whether the node can have children(TRUE) or not(FALSE), irrespectively of whether the parent-to-be element currently has children.

83



In the screenshot, we specified all the records as TRUE, and have the following display results:

ParentNode	ChildNode
[- name-1	descr-1
+ name-1.1	descr-1.1
+ name-1.2	descr-1.2
[- name-2	descr-2
+ name-2.1	descr-2.1
+ name-2.2	descr-2.2
+ name-2.3	descr-2.3

ImageCollapsed

ImageCollapsed property is used to specify the image that will indicate that the tree branch is collapsed. By default this is the "+" image.

ImageExpanded

ImageExpanded property is used to specify the image that will indicate that the tree branch is expanded. By default this is the "-" image.

ImageLeaf

ImageLeaf property is used to specify the image that will indicate that the node cannot be expanded or collapsed.



Form Widgets

The form widgets are the graphical elements located in the form and used for data input and display as well as for the program workflow manipulation. Each form widget has its purpose and specific properties. The only property necessary for all the form widgets is the widget identifier. Each form field has a default identifier set by the Window Builder designer during the field creation; however, it can be changed in the Properties view.

Each section describing the form widgets includes the list of the properties that can be applied to the widget. Most of these properties are common for several widgets and are described in the Properties section of this document. Widget-specific properties are described in the section dealing with the corresponding widgets.

Button Widget

The button widget creates an input field, which is a button that the user can click. The most typical way to process the user interaction with the Button widget is specifying the OnInvoke event for it.

The most common properties for the Button widget are:

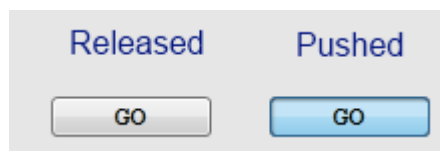
- IsToggleButton
- Text
- AllowNewLines
- Image
- Events(OnInvoke)

Accelerators

By default

IsToggleButton

By default, a button click is animated so that the button is displayed pushed and released quickly. The IsToggleButton property, when set to True, allows the button to have two positions - pushed and released, which change after each mouse click:



The change in the button position does not influence the application response to the events specified for the button.

Browser Widget

The file browser widget embeds a fully functional browser into the form. The file browser will attempt to render the source specified in its field value. You can use the browser to view web pages, files, images and folders on your computer. The file browser is also useful to merge 4GL applications with Web-based applications.



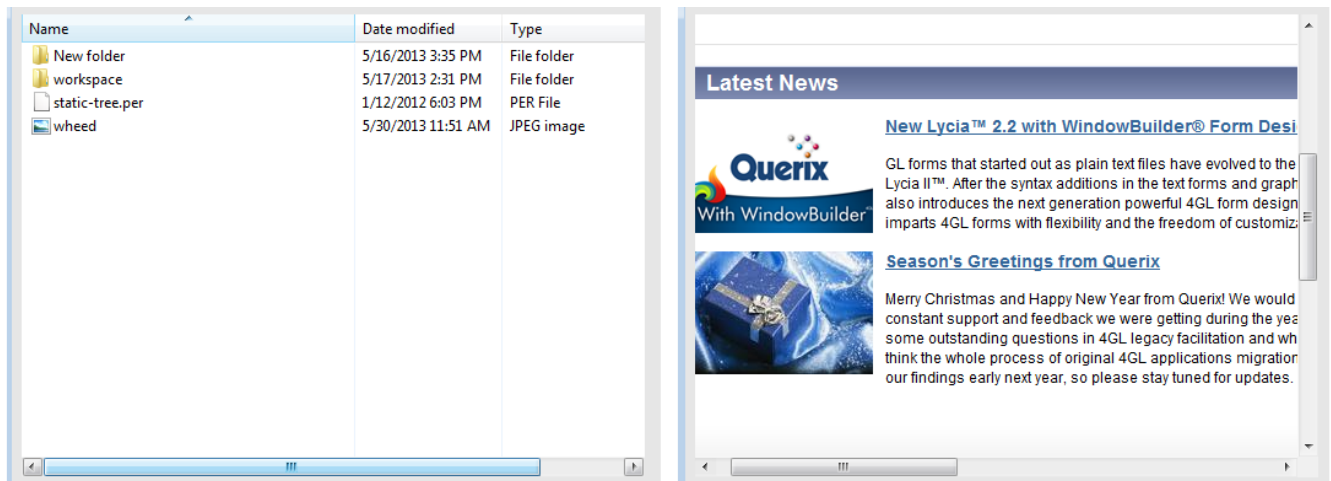
The properties typically used with the Browser widget are:

- Text
- Events

Text

The text property specifies the Url address of the web page, or a path to the folder or file you want to display to the browser. The Text can be specified as a field value, so you can change it dynamically at any time using 4GL statements such as DISPLAY TO <field>.

The screenshot below shows the Browser widget displaying a directory on a local disc and a web page:



Canvas Widget

The Canvas widget is used to display interactive SVG images. The most common properties for the Canvas widget are:

- Image
- Events

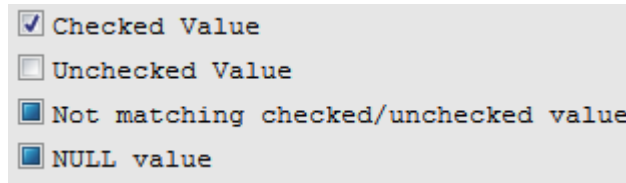
Checkbox Widget

Check boxes are used for making binary choices and each check box relates to a single variable. Unlike radio buttons, it is possible to select all or any of the options.

The programmer can specify the value that should be written to the underlying variable when checking and unchecking this box.



Besides the checked and unchecked states, the Checkbox widget has a Not Touched or NULL state which is indicated by the checkbox filled with blue. This mark is also used when a value displayed to the field does not match either checked or unchecked value:



During the input, you can use the Not Touched state to leave the initial value of the variable unchanged.

You can also use a check box to trigger events, instead of writing values to variables directly. This may be required if you want to execute some program logic straight after the user has clicked the check box.

The properties typically applied to the checkbox widgets are:

- CheckedValue
- UncheckedValue
- Events(onCheck, onUncheck)
- Title
- Image

Checked Value

The Checked Value property is used to specify the value that will be passed to the variable associated with the checkbox, when this checkbox is checked.

During the INPUT WITHOUT DEFAULTS and DISPLAY statement execution the checkbox will be initially displayed as marked if the value of the corresponding variable matches the one specified in the Checked Value property.

Remember, that the data type and format of the specified Checked Value must correspond to those of the variable bound to the field.

Unchecked Value

The Unchecked Value property is used to specify the value that will be passed to the variable associated with the checkbox, when this checkbox is unchecked.

During the INPUT WITHOUT DEFAULTS and DISPLAY statement execution the checkbox will be initially displayed as not marked if the value of the corresponding variable matches the one specified in the Unchecked Value property.

Remember, that the data type and format of the specified Unchecked Value must correspond to those of the variable bound to the field.



Combobox Widget

A combo box field displays itself as a field with a down arrow button at the right hand end. Clicking the down arrow displays a drop down list of values from which users can choose. The list can be populated statically within the form definition using the ComboBoxItem widget. There is also the option to allow the user to enter data into the field which do not exist in the list. At runtime, the combo list values can be changed (removed, inserted, searched, etc...) by using the relevant `fgl_list_xxx()` functions. The user can select list entries either by using the mouse or by navigating through the list using the keyboard (cursor keys or alpha numerical keys to search for entries).

The typically used combobox properties are as follows:

- Editable
- Initializer
- MaxDropDownHeight
- MaxDropDownWidth
- Events (onDropDown,OnDropDownClose,onSelectedImageChange)

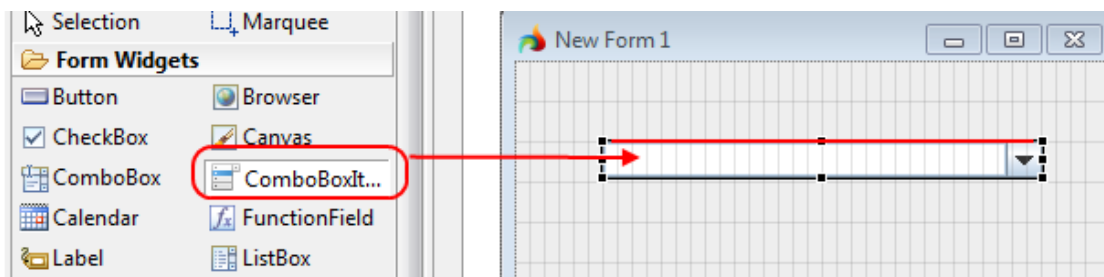
Initializer

The Initializer property is used to specify the function that will be executed when the form with this checkbox is displayed. The function name passed as the Initializer property value should be given without the brackets (unlike the usual `CALL function()` syntax), and no arguments can be passed to this function.

Combo Box Items

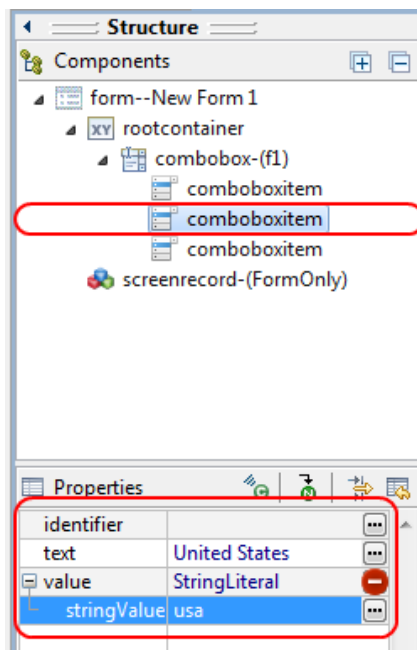
The ComboBox Item is available in the Palette view and is used to specify the set of the items that will be available in the combobox dropdown list.

To add a ComboBox item, select it in the Palette view and then click on the target ComboBox widget in the form:





The ComboBox Items can be selected in the Structure view for the further editing:



Each Combo Box item needs two values to be specified:

- Text - Identifies the text that will be displayed to the combobox dropdown list
- Value - Specifies the actual value that will be passed to the variable when the user selects the item. The Value property consists of the value type ("**StringLiteral**" on the screenshot) and the value itself ("**usa**" in the screenshot). The value type should correspond to the data type set for the ComboBox widget.

During INPUT WITHOUT DEFAULTS and DISPLAY statement execution the combobox displays the item the Value property of which matches the value of the related variable.

You cannot specify some range to be passed to the combo box list, only separate values. To let the user choose from a given range, use the Spinner and Slider widgets.

Editable

The Editable property, when set to TRUE, allows the user to enter their own value to the combobox field besides those already present in the dropdown list.

MaxDropDownHeight

The MaxDropDownHeight specifies the maximum height of a combo box dropdown list, in pixels.

MaxDropDownWidth

The MaxDropDownHeight specifies the maximum width of a combo box dropdown list, in pixels.



OnDropdown Event

The OnDropdown event specifies an event handler called when the user unfolds the combobox list.

OnDropdownClose Event

The OnDropdownClose event specifies an event handler called when the user selects a combobox item and closes the combobox list.

OnSelectedItemChange Event

The OnSelectedItemChange event is called when the user changes the item selected to the combobox.

Calendar Widget

The calendar widget is displayed as a field with a button at its right part. The button has a calendar icon. Clicking on the button opens a calendar window, allowing the user to select the required date graphically using the mouse.

The selected date is then written back to the field. The date can also be entered manually using the keyboard.

The format of the data displayed to the calendar field is defined by the locale settings of the client machine.

The properties that are most typically used with the Calendar widget are:

- Identifier
- Century
- Required
- Format
- Comment
- Autonext
- Tooltip


Century

The Century property applied to the fields to which the date is to be entered, specifies the way the inputted date will be expanded, in case the year value is represented by two digits only.

There are four possible values for the Century property:

- Closest - Expands the two digits of the year value to the year closest to the current date (either in present, past, or future)
- Current - Adds two first digits of the current year
- Past - Expands the year value to the nearest year in past
- Future - Expands the year value to the nearest year in future

Function Field Widget

The Function Field is a field which has a button displayed to its right. This button has a default image () that can be changed by the Image attribute. The button should be used to trigger an event. A classic example would be a field where the e-mail address of a contact is stored. The button could be used to start the e-mail client to



send an e-mail to this e-mail address. Another example would be a field to keep the website of a company. When the user clicks the button, the browser window opens with the corresponding URL.

The most typical properties used with the widget are:

- Autonext
- Identifier
- Comment
- Enable
- DataType
- Required
- TabIndex
- TextAlignment
- ToolTip
- Image (applied to the button)
- Event (OnInvoke)

Enable

The Enable property specifies whether the button can be pressed or not. If the Enable property is set to False, the button becomes gray and the user cannot press it.

Label Widget

The Label widget is a form widget typically used to display data that cannot be manually changed by the user on runtime. This can be some text or an image. The most typical properties for the Label widget are:

- IsDynamic
- Image
- Text
- Text Alignment
- AllowNewLines
- Forecolor
- Events (OnMouseClicked)

You can use the Events properties to allow the program workflow control to be performed via the Label widget. It is convenient to use the Label as a link to web-pages (use the OpenUrlEvent handler).

Is Dynamic

The IsDynamic property when set to TRUE identifies that the label text or image can be changed dynamically by the DISPLAY TO statement. If the property is set to FALSE, the label is static and cannot be changed.

Listbox Widget

The Listbox widget is used to display structured list of values the user can choose during the input. It is similar to the ComboBox widget, but the user cannot add their own values and the whole list is displayed to the form, not to the dropdown list.

The other feature of the Listbox item is that it is possible to select several items at once.

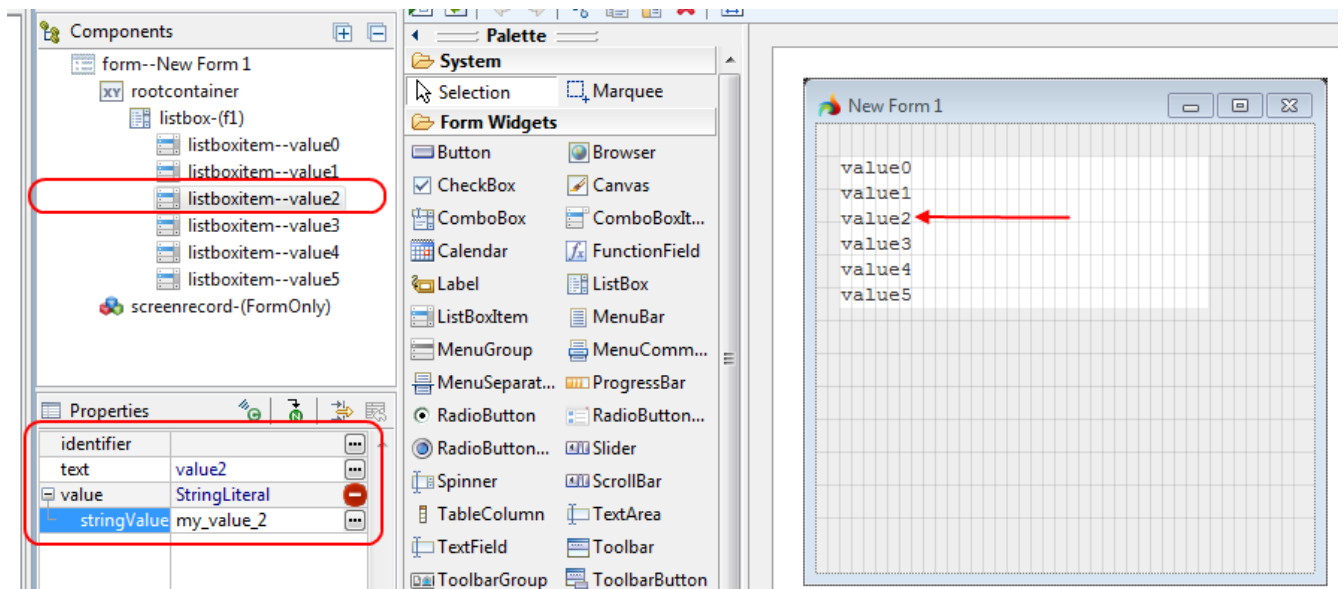


The properties most typically used with the ListBox widget are:

- EnableMultiSelection
- Events (OnSelectedItemChange)
- Text Alignment

List Box Items

The List Box Items are selected from the palette and added directly to the listbox item. The added items can be viewed in the editor area, and selected in the Structure view for further settings:



Each Combo Box item needs two values to be specified:

- Text - Identifies the text that will be displayed to the combobox dropdown list
- Value - Specifies the actual value that will be passed to the variable when the user selects the item. This property consists of the value data type specification and the value itself.

During INPUT WITHOUT DEFAULTS and DISPLAY statement execution the listbox displays the item the Value property of which matches the value of the related variable.

EnableMultiselection

The EnableMultiselection property when set to TRUE allows the user to select several listbox items during the input. To select several items, the user should keep the CTRL key pressed.

The multiselection result is passed to the receiving variable as a string value where the selected values are separated with commas. Therefore, the receiving variable should be of a data type that can accept a character string (CHAR, VARCHAR, STRING).



RadioButton Widget

Radio buttons allow you to create a group of items from which only one option can be chosen. You can have one single or several radio button items in one radio button group but all items in the group will write to the same variable.

Radio Button widget allows to add radio buttons to the custom location on the form, unlike the RadioButtonList widget, where all the items are grouped in one place and definite order.

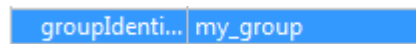
The main properties applied to the radiobutton widget are:

- Value
- GroupIdentifier
- Events (OnCheck/OnUncheck)
- Title
- Image
- TextAlignment

Note, that the Title value specifies which label will be displayed to the radio button object. The Value property identifies which value will be actually passed to the variable or may be received by the radio button.

GroupIdentifier

The GroupIdentifier property specifies the name of a RadioButton group to which the current RadioButton will be put. All the RadioButtons that are to be included to the same group should have one GroupIdentifier:



The GroupIdentifier unites the radio buttons into one object and it is the Identifier that should be referenced in INPUT or DISPLAY statements:

```
INPUT a FROM my_group
```

Each item of a Radio Button group, when selected can pass its own value to the related variable and/or invoke an event, which is typically an OnCkeck/OnUncheck one.

To write to more than one variable, you need to create separate radio button groups, each group having its own Identifier.

If the value returned by the radio button field is a null string or does not match any of the radio button values, all buttons are shown as 'not' selected.

Alternatively, you can use a radio button to trigger key events, instead of writing values to variables directly. This may be required if you want to execute some program logic straight after the user has selected an item from the radio button group.

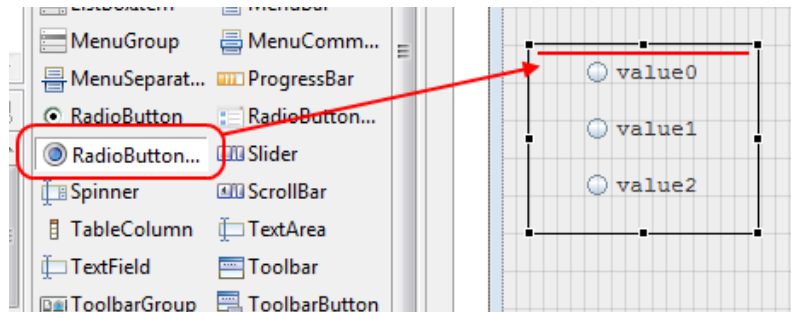
RadioButtonList Widget

RadioButtonList widget is a container that allows you to create a set of radio buttons grouped automatically under one identifier. The main properties used with the RadioButtonList widget are the following:



- Orientation
- Events (OnSelectedItemChange)
- Text alignment

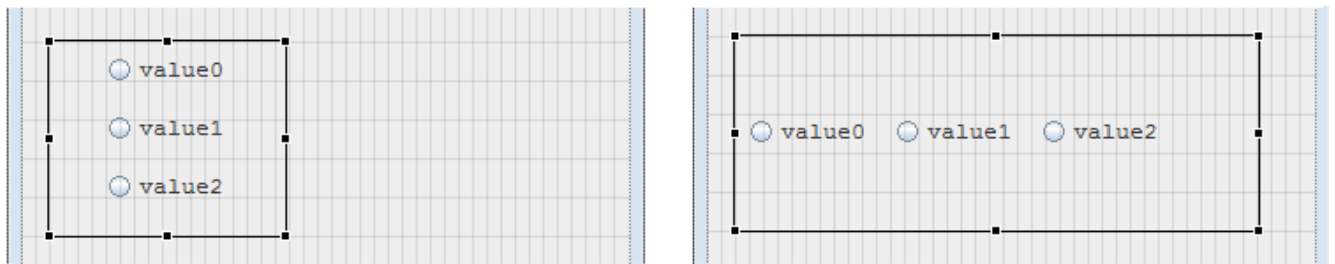
The RadioButtonList should be filled with radioButtonListItems, that are also present in the Palette:



He items in their turn also have a set of properties, the most essential of which are Title, Value, and OnCheck.

Orientation

The Orientation property has two possible options (Horizontal/Vertical) and specifies whether the items in the RadioButtonList are listed vertically (the default option) or horizontally:



Progress Bar Widget

The Progress Bar widget is used to display a numeric value within the specified range. It is typically used to display graphically the level of progression of some process. One can also use it to display some level of fulfilment. For example, during a form filling, you can use it to show the relative number of the fields filled and left to fill.

The typical properties used with the ProgressBar widget are:

- MaxValue
- MinValue
- Step
- Orientation

The MaxValue property specifies which value is needed to display the progress bar as totally filled. The MinValue identifies at which value the Progress bar widget is displayed empty.



Step property

The Step property specifies the difference between each the two values passed to the progress bar that leads to filling or clearing the new part of it.

Slider Widget

The Slider Bar widget is an graphical widget that allows to input or display a numeric value by changing the position of a slider on a scale.

The most typical slider properties are:

- MajorTick
- MinorTick
- MaxValue
- MinValue
- Orientation
- ShowTicks
- Events (OnValueChanged)

MajorTick

The MajorTick property specifies the slider step in case when the user clicks on the scale away from the current slider position.

MinorTick

The MinorTick property specifies the slider step in case when the user presses Right or Left keyboard arrow keys.

ShowTicks

The ShowTicks property indicates whether the ticks will be displayed along the scale. The property displays only the minor ticks.

ScrollBar Widget

The Scrollbar widget is used to manipulate numeric values within the specified range. It can be used to input values, but is more likely to make the application perform some actions when the user moves the scrollbar slider (OnScroll event).

The main ScrollBar properties are:

- MaxValue
- Minvalue
- LargeStep
- SmallStep
- Orientation
- Events (OnScroll)

LargeStep

The LargeStep property specifies the slider step in case when the user clicks on the ScrollBar away from the current slider position.



SmallStep

The SmallStep property specifies the slider step in case when the user presses Right or Left keyboard arrow keys or the arrow buttons at the ends of the scrollbar.

OnScroll Event

The OnScroll event is used to specify an event handler that will be invoked when the user changes the position of the ScrollBar slider.

Spinner Widget

The Spinner widget is a graphical widget that allows entering numeric values both manually or using the spinner arrow buttons.

The most used properties of the Spinner Widget are:

- Step
- MaxValue
- MinValue

Step

The Step property specifies the difference between the current and the next value resulting from the user click on the spinner buttons.

Text Field Widget

The Text Field widget is a widget used to input and display text data of different formats. The value inputted to the text field can include any printable characters. The most significant properties typically applied to Text Field widgets are as follows:

- Editor
- Font
- Text alignment
- ToCase
- Textpicture
- Displaymodes
- DataType

Most of these properties are described above in the Properties section.

Editor

The Editor property is used for text fields to which variables of large data types are displayed. The property value should indicate the extension of the file to work with. The passed extension indicates the program which will be used to manipulate the BLOB file.

Text Area Widget

As with the Text Field, Text Area is used to input and display text data of different formats. The value inputted to the text field can include any printable characters. The only feature that distinguishes Text Area from Text



Field is that Text Area allows the data string to be split into several lines. This option is activated by setting the AllowNewLines property value to True.

The most significant properties typically applied to the TextArea widget are given below:

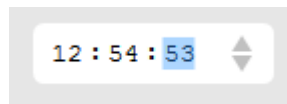
- Editor
- AllowNewLines
- UseTabs
- Horizontal/VerticalScrollBarVisibility
- Text alignment

UseTabs

The UseTabs property, when set to True, allows the user to insert the tab character to the field value. It means that if the property is set and the user presses the Tab key when the cursor is within the text area, the cursor does not move to the next field, but a tab space is added to the field.

TimeEditField Widget

The TimeEditField widget is displayed like a spinner field with three numeric buttons separated with a colon symbol:



The spinner is used to change the hours, minutes and seconds values, depending on the part of the time value in which the cursor is currently set. The up arrow button changes the value to one hour/minute/second later, the down arrow button changes the value to one hour/minute/second earlier.

The properties typically used with the TimeEditField widget are the same as for the TextField widget.

When creating a TimeEditField, you should keep in mind that the widget and the receiving variable data types should correspond to the field purpose, i.e., should be set to DATETIME HOUR TO SECOND. If the receiving variable has larger precision, the missing parts (day, month, year, fraction) will be automatically filled with default values

WebComponent Widget

A WebComponent widget is a form widget that can serve as a seat for external objects, or front-end plug-in mechanisms.

The WebComponent properties are used to specify the object to be used and to set it up. The settings allow to establish communication between the application and the web-component. The WebComponent performance differs from that of the other widgets. The data for this widget usually cannot be stored as is, so that it could be taken and applied directly from the database column. Therefore, the WebComponent widgets are typically FORM_ONLY fields.



ComponentType property

The ComponentType property is used to specify the type of the external object to be used. This is the name that will be passed and associated to the front-end script. The property value should also include a path to the object.

ComponentProperties property

The ComponentProperties property specifies a list of settings specific for the selected external object. For example, the properties should be listed one by one and separated with commas. For example, in case it is necessary to add an external chart widget, the property value may look as follows:

```
type="DEBUG", caption="Expenses summary", subcaption="By teams, first half-year, 2013",  
xaxisname="Team", yaxisname="Expenses", numberprefix="$",  
labels=("Developers", "Designers", "HR", "Documentation", "Support"),  
values=( 0,      0,      0,      0,      0);
```

The set and the order of the properties listed here depends on the external plug-in.