ROWDYNAVIS and COLDYNAVIS

The ROWDYNAVIS or COLDYNAVIS container is used to add dynamic reaction to your layout.

The container is not visible - similar to the TABLE0 container. What is the difference? You control the appearance of the container by an adapter property. Have a look at the following example.

If you enter "United States" as a country, the input line for the state will appear under the input line for the country:

The XML code looks as follows:

```
<rowarea name="Address Input">
    <itr>
        <label name="Country" width="100">
        </label>
        <field valueprop="country" flush="true" length="30">
        </field>
    </itr>
    <rowdynavis valueprop="visible">
        <itr>
            <label name="State" width="100">
            </label>
            <field valueprop="state" length="30">
            </field>
        </itr>
    </rowdynavis>
</rowarea>
```

A ROWDYNAVIS container is placed inside the ROWAREA container. The ROWDYNAVIS container is bound to the adapter class property `visible`.

The adapter class looks as follows:

```
import com.softwareag.cis.server.Adapter;

// This class is a generated one.

public class RowTable0Adapter
    extends Adapter
{
    // property >factor1<
    String m_factor1;
    public String getFactor1() { return m_factor1; }
    public void setFactor1(String value) { m_factor1 = value; }

    // property >factor2<
```

```
    String m_factor2;
    public String getFactor2() { return m_factor2; }
    public void setFactor2(String value) { m_factor2 = value; }

    // property >country<
    String m_country;
    public String getCountry() { return m_country; }
    public void setCountry(String value) { m_country = value; }

    // property >state<
    String m_state;
    public String getState() { return m_state; }
    public void setState(String value) { m_state = value; }

    // property >visible<
    boolean m_visible = true;
    public boolean getVisible()
    {
        if (m_country != null && m_country.equalsIgnoreCase("United States"))
            return true;
        else
            return false;
    }
    public void setVisible(boolean value) { m_visible = value;}
}
```

The property visible depends on the user input of the country property. It returns a boolean value.
Since a ROWDYNAVIS area is a container, it can contain rows - with input lines - and containers.
Therefore, you can add dynamic reaction to your layout definitions in a very flexible way.

The following topics are covered below:

- ROWDYNAVIS Properties

- COLDYNAVIS Properties

- Some Comments on Controlling the Visibility of Controls

ROWDYNAVIS Properties

Basic			
valueprop	Name of adapter property that defines if the area is visible ("true") or invisible ("false"). Must be of type "boolean" / "Boolean".	Obligatory	

height	Height of the control.	Optional	100
	There are three possibilities to define the height:		150
	(A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content.		200
			250
			300
	(B) Pixel sizing: just input a number value (e.g. "20").		250
	(C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		400
			50%
			100%
style	CSS style definition that is directly passed into this control.	Optional	background-color: #FF0000
	With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:		color: #0000FF
	border: 1px solid #FF0000		font-weight: bold
	background-color: #808080		
	You can combine expressions by appending and separating them with a semicolon.		
	Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.		
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

| fixlayout | The fixlayout property is important for saving rendering performance inside your browser. To become effective it requires to have specified the height and the width (if available as property) of the control.

If setting fixlayout to "true" then the control's area is defined as area which is not sized dependent on its content (as normally done with table rendering). Instead the size is predefined from outside without letting the browser "look" into the content of the area. If the content is not fitting into the area then it is cut.

You typically use this control if the content of the control's area is flexibly sizable. E.g. if the content (e.g. a TEXTGRID control) is following the size of the container.

When using vertical percentage based sizing you should pay attention to set the fixlayout-property to "true" as often as possible. - The browser as consequence will be much faster in doing its rendering because a screen consists out of "building blocks" with simple to calculate sizes. | Optional | true

false |

COLDYNAVIS Properties

The properties of COLDYNAVIS are very similar to those of ROWDYNAVIS.

Basic			
valueprop	Name of adapter property that defines if the area is visible ("true") or invisible ("false"). Must be of type "boolean" / "Boolean".	Obligatory	

| width | Width of the control.

There are three possibilities to define the width:

(A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content.

(B) Pixel sizing: just input a number value (e.g. "100").

(C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect. | Optional | 100

120

140

160

180

200

50%

100% |
| takefullheight | Indicates if the content of the control's area gets the full available height.

If you use percentage sizing inside the control's area then this property must be switched to 'true'. If you use no explicit vertical sizing at all - or you use vertical pixel sizing for your controls - the property must be switched to 'false'.

Background information: container control's internally open up a table in which you place rows (ITR/TR) which then hold controls (e.g. LABEL/FIELD). The table that is opened up normally has no explicit height and grows with its content as consequence. By specifying "takefullheight=true" the table itself is sized to fill the maximum height of the available area. | Optional | true

false |

style	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
fixlayout	The fixlayout property is important for saving rendering performance inside your browser. To become effective it requires to have specified the height and the width (if available as property) of the control. If setting fixlayout to "true" then the control's area is defined as area which is not sized dependent on its content (as normally done with table rendering). Instead the size is predefined from outside without letting the browser "look" into the content of the area. If the content is not fitting into the area then it is cut. You typically use this control if the content of the control's area is flexibly sizable. E.g. if the content (e.g. a TEXTGRID control) is following the size of the container. When using vertical percentage based sizing you should pay attention to set the fixlayout-property to "true" as often as possible. - The browser as consequence will be much faster in doing its rendering because a screen consists out of "building blocks" with simple to calculate sizes.	Optional	true false
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

Some Comments on Controlling the Visibility of Controls

ROWDYNAVIS and COLDYNAVIS are container controls that are explicitly defined to provide an area which can be explicitly switched on and off. In addition you will later on see that many controls can control their visiblity and their input status by themselves. For example, a FIELD control can specify if it is invisible, editable, holding an error input etc. in a dynamic way. You may also have noticed that an ITR row definition has an associated `visibleprop` property - linking to a data property that dynamically controls the visibility of the row at runtime.

Use ROWDYNAVIS and COLDYNAVIS for explicitly defining container areas to be switched on/off. Use the control's binding to properties to do the fine-granular control of visibility inside one container.

A bad example of usage would be if you place a COLDYNAVIS container around each FIELD that you want to control in means of visibility. Use the FIELD's `statusprop` property instead.