

SAP User Experience –  
Accessibility  
SAP INFORMATION



# Accessibility Guidelines for Web Dynpro for ABAP and Web Dynpro for Java

**Version 1.0**

October 17, 2007



# Copyright

© Copyright 2007 SAP AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft, Windows, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.

IBM, DB2, DB2 Universal Database, OS/2, Parallel Sysplex, MVS/ESA, AIX, S/390, AS/400, OS/390, OS/400, iSeries, pSeries, xSeries, zSeries, z/OS, AFP, Intelligent Miner, WebSphere, Netfinity, Tivoli, and Informix are trademarks or registered trademarks of IBM Corporation in the United States and/or other countries.

Oracle is a registered trademark of Oracle Corporation.

UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.

Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems, Inc.

HTML, XML, XHTML and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.

Java is a registered trademark of Sun Microsystems, Inc.

JavaScript is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

MaxDB is a trademark of MySQL AB, Sweden.

SAP, R/3, mySAP, mySAP.com, xApps, xApp, SAP NetWeaver, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

## Table of Content

<b>1</b>	<b>Introduction.....</b>	<b>4</b>
1.1	Disclaimer .....	4
1.2	Document History .....	4
<b>2</b>	<b>General .....</b>	<b>5</b>
2.1	Prerequisites.....	5
2.2	Terminology .....	5
2.3	What is Web Dynpro? .....	5
2.4	General Accessibility Rules .....	6
2.5	Examples of the General Rules .....	9
2.6	Accessibility Mode .....	12
2.7	Text Elements.....	13
2.8	Action Elements.....	18
2.9	Selection Elements .....	20
2.10	Complex Elements.....	26
2.11	Layout Elements .....	31
2.12	Graphical Elements .....	36
2.13	Integration Elements.....	38
2.14	Other Elements.....	39
2.15	Elements in the Pattern Library .....	42
<b>3</b>	<b>Testing Web Dynpro for ABAP and Web Dynpro for JAVA .....</b>	<b>47</b>
3.1	Prerequisites for the Test Environment .....	47
3.2	Fundamental Concepts.....	47
3.3	Automated Tests at Design Time .....	48
3.4	Manual Tests at Runtime.....	51
<b>4</b>	<b>How to Integrate Third Party Products .....</b>	<b>53</b>
4.1	Embedded Components .....	53
4.2	Launching External Programs .....	53
<b>5</b>	<b>Useful Links .....</b>	<b>54</b>

# 1 Introduction

## 1.1 Disclaimer

These guidelines do not represent any promise or obligation on SAP's part to make any aspect of the software accessible, nor any application that is evaluated against the Accessibility check-list accessible for ABAP Developers.

This document is for informational purposes only. Its content is subject to change without notice, and SAP does not warrant that it is error-free. SAP MAKES NO WARRANTIES, EXPRESS OR IMPLIED, OR OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.

The information contained in this document represents SAP's current view of accessibility criteria as of the date of publication; it is in no way intended to be a binding guideline on how to ensure accessibility of software products. SAP specifically disclaims any liability with respect to this document and no contractual obligations or commitments are formed either directly or indirectly by this document. This document is for internal use only and may not be circulated or distributed outside your organization without SAP's prior written authorization. © 2006 SAP AG

## 1.2 Document History

<b>Version (starting with 1.0)</b>	<b>Status (Draft/Review copy/ Released)</b>	<b>Date (DD.MM.YY)</b>	<b>Author</b>
[1.0]	[Released]	[17.10.07]	SAP

## 2 General

### 2.1 Prerequisites

For information about the prerequisites, see the newest version of the document "[Front-End Requirements and Infrastructure for Accessibility](http://www.sapdesignguild.org/resources/acc_technical_requirements_V3_1_external_EN.pdf)".  
[http://www.sapdesignguild.org/resources/acc\\_technical\\_requirements\\_V3\\_1\\_external\\_EN.pdf](http://www.sapdesignguild.org/resources/acc_technical_requirements_V3_1_external_EN.pdf)

### 2.2 Terminology

Term	Description
Caption	Textual description on or beside a user interface element. Static part of the dialog. Language-dependent.
Context	Situation inside an application.
Label	Textual description on or beside a user interface element. Static part of the dialog. Language-dependent. Typically aligned left of a succeeding user interface element.
Title	A visually attractive title text above a subordinate screen area. Language-dependent.
Tooltip	Additional information about a user interface element and its range of use. Example: A label "To" can have a tooltip "Recipient the mail is being sent to".
Screen Reader	Assistive software for blind users. The software extracts the screen content, formats it, and sends it to an output device. The output device can be a speech synthesizer or a refreshable braille display. Both devices can also be used together. The output of the screen reader can also be displayed on the screen.
Style Guide	Style guides are collections of rules and guidelines for designing the user interface of certain operating system platforms, such as Windows or Mac OS, or for certain application types such as ABAP applications.
Value	Textual data (of any format) inside data input fields.

You can find more terms and their descriptions in the [Accessibility Glossary](#) of the SAP Design Guild.

### 2.3 What is Web Dynpro?

Web Dynpro is a client-independent programming model on the *SAP NetWeaver* technology platform and is used to develop user interfaces for professional business applications. It can be used to design user interfaces for both Java and ABAP applications. It is based on the *Model-View-Controller* paradigm, which ensures that the business logic is separated from the presentation logic. This architecture is visible, for example in the Web Dynpro perspective of the SAP NetWeaver Developer Studio (NWDS).

#### Note:

Note that all statements about Web Dynpro refer to Release **SAP NetWeaver 7.0**.

*Web Dynpro ABAP* or *Web Dynpro for ABAP* (WD4A, WDA) is SAP's standard UI technology for developing Web applications in the ABAP environment. It consists of a runtime environment and a graphical development environment. It is equipped with special Web Dynpro tools that are integrated in ABAP Workbench.

Many of the things that application developers require to develop accessible applications are already integrated into the Web Dynpro framework or can be provided by *SAP NetWeaver Portal* for certain users or user groups. Application developers do not need to worry about any of the following:

- Predefined speech synthesis for screen readers (name, state, and value of the UI elements, or instructions for using the keyboard)
- Keyboard access to the UI elements
- Tab chain of the UI elements (defined by the layout container)
- Skipping UI elements and groups for quicker navigation within a view element
- Special style templates with user-defined color schemes
- Support for large fonts and scaling
- Semantic colors (colors that have a symbolic meaning, such as green or red for statuses)
- Keyboard commands

All these features provide a basis for supporting assistive technologies. However, application developers are still responsible for certain tasks, the most important of which are described in the chapters 2.4 and 2.6.

The remaining chapters follow the structure of the *Web Dynpro ABAP* element documentation and its development environments. They focus more explicitly on the individual UI elements of the Web Dynpro interfaces for ABAP and Java.

## 2.4 General Accessibility Rules

This section contains the most important Dos and Don'ts when programming in Web Dynpro ABAP or Web Dynpro Java.

### 2.4.1 Important Dos

The following are the most important **Dos**:

- Use the **labelFor** property for labels.  
Always use the **labelFor** property to associate **labels** with other UI elements (also read the notes in the chapter 2.7.5 about the **Label** UI element). Screen readers require this association so that they can read the correct label text when the element has the focus.
- Use the **tooltip** property for description purposes.  
Use the **tooltip** property of a UI element to describe how and where it is used and its context. If you use the **TextView** element to display inspection texts, for example, you must specify this in the **tooltip** property (refer also to *Example 4* in Chapter 2.5).
- Use the **accessibilityDescription** property as a replacement for titles or, alternatively, for short descriptions.

Use the **accessibilityDescription** property of a UI elements as a short description of its purpose or as a replacement title for the element. Do this if the element does not have a visible title for design reasons. (This can occur in group elements that have no title area,

such as **TransparentContainers**.) Also do this if the element does not have a title property. (This is recommended for elements such as **RoadMap** or **PhaseIndicator** without their own title lines.) More information is available in 2.11.

- Create group headers and column headers  
Always create group headers for group elements with visible title lines. Also always create column headers in table elements (refer also to *Example 5* and *Example 6* in Chapter 2.5).
- Note the following point for the **TimedTrigger** element.  
If you want to use the **TimedTrigger** UI element to display time-dependent or periodic activities in your application, (such as the current state of a server), read the recommendations in Chapter 2.8.3 about **TimedTrigger**.
- Note the following point for graphics and graphical displays.  
If you want to use graphics, graphical tools, overviews, maps, or charts in your application, read the recommendations in Chapter 2.12.
- Use semantic colors to represent states.  
Use semantic colors to represent different states. You can, for example, specify `BADVALUE_MEDIUM` as the value of the **cellDesign** property. This indicates critical values in a table column (or table cell) by background color.
- Use clear and simple texts, without abbreviations.  
Use clear and simple texts in labels, titles, headers, and other texts. If possible, do not use abbreviations or special characters. Use the text "Material Number" and not "Mat. No.", for example, if there is enough space in your layout. If there is not enough space, you must create the full text in the **tooltip** property of the element associated with the label. The same applies to acronyms, except that well known acronyms (like SAP) do not need to be explained in the tooltip.
- Always focus on the user as your target audience.  
Think of the requirements of the user when you design your interfaces. The simplest or most obvious approach from the development team's perspective is not always the best for the user. Always refer to the style guidelines that apply to interfaces. *All* users benefit from a clear arrangement of elements that observes recognized design conventions.
- Use the most up-to-date runtime environments.  
Whenever possible, use the latest version of the Web Dynpro runtime environments. The most current Web Dynpro framework always contains many enhancements related to accessibility.
- Run automated tests.  
Use the automated accessibility tests implemented in your development environments. An overview of the properties checked by these tests is available in Chapter 3.3.
- Run manual tests.  
Also run manual tests. These tests are particularly important if you combine elements in non-standard ways. More information is available in Chapter 3.4.
- Also read the framework documentation.  
The relevant chapters in the framework documentation contain additional information about accessibility in Web Dynpro Java and Web Dynpro ABAP.

## 2.4.2 Important Don'ts

The following are the most important **Don'ts**:

- Only use elements for their intended purposes.
 

Do not use, combine, or modify elements for purposes other than their intended use, just to achieve a certain type of layout. A user without visual impairments may not notice any difference in the usability of the application. However, a blind user will find the new combination of elements confusing at best. This is because the elements provide no new information and are still used in the old semantic context. The situation is particularly confusing if you display elements in a table-like structure in a View element. Blind users will not be able to immediately identify their position in a row or column. Also you must not modify any library elements at a deeper level (such as modifications to HTML code, scripting, or similar) since any changes can hamper or remove accessibility (refer also to *Example 1* and *Example 2* in Chapter 2.5).
- Do not use standalone label elements to display free text.
 

Never use standalone label elements to display free text. This is particularly important if you add elements to a view element dynamically at runtime. Always use the **TextView** element or the **Explanation** element (Web Dynpro ABAP only) for this purpose, or associate the label with another element.
- Do not use radio button elements to show or hide screen areas above the focus.
 

Do not use UI elements to modify the structure of the screen above the focused element. For example, do not place a group of radio buttons in the lower half of an application window, and make elements or element groups in the upper half appear or disappear depending on which radio button is selected in the lower half. Screen readers often cannot reliably tell the user which parts of the screen have changed, which means it is not immediately obvious to blind users that elements or element groups above the focus have been hidden. Redesign the screen instead, for example by placing the element groups on different tab pages of a tabstrip element.
- Do not use the **Table** element for layout purposes.
 

Do not use the **Table** UI element purely for layout purposes. Use layout containers instead. Screen readers are able to identify the existence of a layout table, but this confuses blind users. This is because table elements in Web Dynpro are intended for data display purposes only.
- Do not nest **TabStrip**, **Group**, or **Tray** elements within the same type of element.
 

Whenever possible, do not nest tabstrip, group, or tray elements within the same element type (for example, do not nest a tray element within another tray element). This kind of deep nesting can be identified visually, but is not immediately obvious to blind users, and is difficult to navigate to. If you want to implement nesting within a tabstrip or tray element, use **TransparentContainer** elements (refer also to *Example 3* in Chapter 2.5).
- Do not use the same text for the properties **tooltip** and **accessibilityDescription**.
 

Do not use the same text for the properties **tooltip** and **accessibilityDescription**. The **accessibilityDescription** property is intended for short descriptions to be used instead of titles, whereas tooltips are semantic descriptions that can be longer.
- Do not use a different element for each source text when you compose a longer text.
 

If you want to construct a longer text from different source texts, do not use a different UI element for each source, even when this is the simplest method. Instead, combine the individual texts as a complete text in the program code. The text can then be displayed in a TextView or InputField element using the **readonly** property (refer also to *Example 7* in Chapter 2.5).



## 2.5 Examples of the General Rules

### 2.5.1 Example 1: Using Elements for Their Intended Purposes

Always use the Web Dynpro elements that are currently available instead of constructing an element yourself or trying to anticipate a new element in a future release.

Figure 2.1 shows a group of CheckBox elements created by a Table element with five rows and two columns. This is not a recommended implementation of Checkbox elements. Firstly, the CheckBoxGroup element is available for this purpose. Secondly, the assignment of text to checkbox is visual only, and not specified in the program.



**Figure 2.1: Table Element for CheckBoxGroup Layout**

Always check whether Web Dynpro already provides an element that satisfies your layout requirements. If the functions you require are available, arrange the elements in accordance with the recognized guidelines. Whenever possible, do not use non-standard combinations of elements and groups (such as trays in Group elements within tabstrips).

Never modify Web Dynpro elements at the rendering level. This could remove accessibility support and also make rendering impossible for certain browsers.

### 2.5.2 Example 2: Table Elements for Table-like Text Layouts

Use Table Elements for Table-like Text Layouts.

Figure 2.2 shows a combination of TextView, Label, and Link elements combined to create a table. Users of screen readers cannot identify the current row and column in this type of arrangement. Use the **Table** element for this purpose instead.

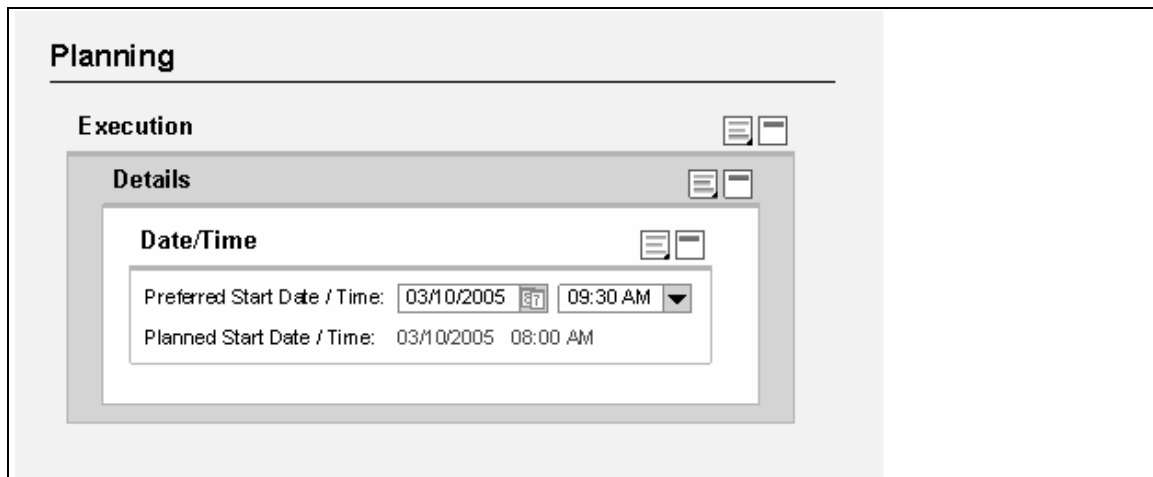
Application Service	Description	Req. SAPS	Req. Mem.(MB)
AC6(APP 04)	Acme Corp. Web AS Central Instance	600	1024
AC4(CE)	Acme Corp. Web AS Customer Site	600	1024
Test(DB2)	Prototype	600	1024

**Figure 2.2: Table-Like Layout with TextViews**

### 2.5.3 Example 3: Do not Nest Grouping Elements

In General, Do not Nest Grouping Elements with Visible Titles

Figure 2.3 shows nested Tray elements. This does not just look bad, it is also confusing, since the nesting depth is not apparent from the title. Instead, represent the hierarchy by using TransparentContainer and SectionHeader elements with different levels as titles.

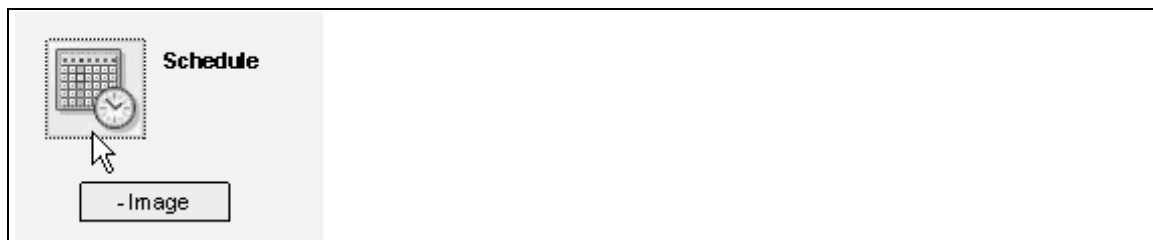


**Figure 2.3: Nested Trays**

## 2.5.4 Example 4: Tooltips for Images

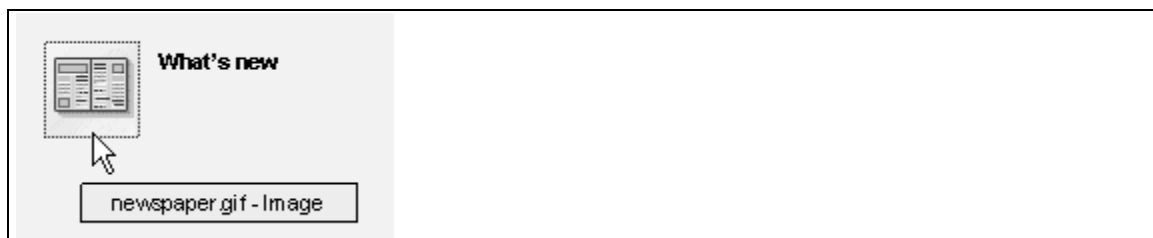
Always Create Tooltips to Describe Images.

No tooltip has been created for the image in Figure 2.4. If accessibility mode is activated, however, the image can be focused. The missing tooltip means that screen readers can only announce the word “Image” when the image has the focus.



**Figure 2.4: Image with No Semantic Information**

The technical name of the image has been entered as the tooltip in Figure 2.5. However, this is only a good solution if the technical name is meaningful and matches the object displayed in the image. Technical names can often be abstract, such as “123.gif”. This does not benefit the user and causes problems in translation. The file name extension also causes problems. We do not recommend this solution.



**Figure 2.5: Image with File Name as Semantic Information**

You can use the **isDecorative** property to remove images from the navigation sequence (the tab chain) in Web Dynpro. The keyboard focus will then never be moved to the image.

There are two possible scenarios:

- The image has a descriptive purpose or is a functional image.
  - Descriptive images may need to be given the focus. One example is when the items in a bill of material are illustrated by images of goods, and the layout does not indicate explicitly that an image is being displayed.
  - One example of a functional image is the visualization of a state (such as a red traffic light) that is linked to an event. The user could then activate the image of the red light to determine its cause. In this case, the image must be given the focus to enable it to be activated. Also, the tooltip text of the image must be associated with the semantics of the image (such as “Critical State” for a red light) and to navigation instructions (such as “Press the space bar for details.”). Refer also to the notes in Chapter 2.12.2 about the Image element). A better choice here would be to use the **LinkToAction element** with an image and no text (but a full tooltip). This is because the **LinkToAction element** already provides an appropriate text describing the activation information. This example demonstrates that you can use a different element to achieve the same function.
- The image is only for decorative purposes.
  - Set the **isDecorative** property for the Image element and place it next to a text used in the same semantic context. For assistive technologies, you must enter a short description of the image in the **tooltip** property. This is also important if the program cannot find the image at runtime.

## 2.5.5 Example 5: Column Headers in Table Elements

Always use column headers in your Table elements.

Figure 2.6 shows a table where the column header is missing from the second column. When navigating, blind users cannot identify which column they are in, or whether they have moved to a new column.

Sales Orders Overview		
Open		Link
Sales Orders		
US Region Northeast / East Coast	Wholesale	▲
US Region Southeast / South Coast	Wholesale	■
		▼

Figure 2.6: Missing Column Header

## 2.5.6 Example 6: Text in Visible Group Titles

If you use visible group titles, you must always create texts for them.

Figure 2.7 shows a Group element where the title is missing (from the top line). It would not be immediately obvious to blind users why the elements are grouped in this way.

File Name	File Size
Product HT-1001 Presentation Long.ppt	2,489 KB
Product HT-1001 Presentation Short.ppt	16,894 KB
Product HT-1000 Series.pdf	2,300 KB
Product HT-1001 Series.pdf	108 KB
Product HT-1002 Series.pdf	2,467 KB
Product HT-1003 Series.pdf	620 KB

**Figure 2.7: Group Element Without Group Title**

### 2.5.7 Example 7: Do not Construct a Text from Different TextView Elements

Do not construct a single text from different TextView elements. TextView elements contain tab stops in accessibility mode.

Figure 2.8 contains five TextView elements, which means it also has five tab stops (numbered 1-5). In this case, the simplest method for the developer was to create the display in a declarative fashion by using the context assignment of the content. This is because the program gets the information from various different sources. The style template also forced the developer to display the text with different TextViews. This is because only one style can be realized using the TextView at any one time.

Physical Landscape – Active Servers			Data captured at Wed Sep 29 11:57:08 CEST 2004	
1	2	3	4	5

**Figure 2.8: Too Many TextView Elements**

How could the example above be improved? At the very least, the number of tab stops could be reduced to two here, using one TextView element for 1-3 and one for 4-5. To make this possible, the developer must combine the texts for the relevant object in advance in the code. In Web Dynpro ABAP, a developer could also use a FormattedTextView element. This allows the developer to format the text in other ways.

## 2.6 Accessibility Mode

There are some check points for which the Web Dynpro framework cannot specify accessibility. In these cases, the application developer must intervene and implement a query about whether the correct appropriate mode of the framework is active at runtime.

The class **IWDFApplication** provides an appropriate method in Web Dynpro Java (Listing 2.1: Method for Querying Accessibility Mode). If the result is **true**, the program can react accordingly.

```
Boolean wdThis.wdGetAPI().getComponent().getApplication().isAccessibilityModeRequested()
```

*Listing 2.1: Method for Querying Accessibility Mode*

In Web Dynpro ABAP, the class **IF\_WD\_APPLICATION** provides the corresponding method **GET\_IS\_ACCESSIBLE**.

We recommend that you query these methods as rarely as possible and that you create a solution that looks the same whether in accessibility mode or not. Remember that accessibility mode is also used by users without visual impairments. Users with motor impairments, for example, enable accessibility mode for its additional keyboard.

An accessibility mode also requires extra programming work, since it needs to be implemented, tested, and maintained.

If you use the function module **GET\_ACCESSIBILITY\_MODE** to replace graphics with tables in accessibility mode, users with motor impairments then have no way of viewing these graphics. We can assume that this group of users would prefer to have the information displayed in a graphic and not a table.

We recommend that you design your solution so that it offers both the graphic and a text alternative, such as a table. You can also implement a Button element that allows the user to toggle between the graphic and the text alternative.

Another alternative is to let the user decide whether to display the information as a graphic or as table by making a personalized setting in the application. This requires you to integrate an appropriate dialog into the application.

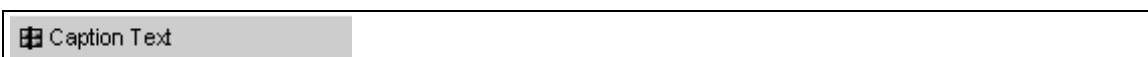
## 2.7 Text Elements

This chapter explains each of the UI elements in the standard library of **Text Elements**, and the programming rules. It discusses the following UI elements:

- Caption
- Explanation
- FormattedTextEdit
- FormattedTextView
- InputField
- Label
- TextEdit
- TextView

### 2.7.1 Caption

The **Caption** UI element (as shown in Figure 2.9) is used by some elements (such as Group, Tab, Table, TableColumn, and Tray) to display a header or a title.



**Figure 2.9: Caption Element**

#### Do's for Caption Elements

- Use the **tooltip** property of the element to add semantic information.

#### Don'ts for Caption Elements

- If any important information is not included in the **text** or **tooltip** properties of the Caption element, you must not convey it solely in the image of the Caption element. Instead, include

this information in the **text** or **tooltip** property. An image on a Caption element cannot be focused and is intended only as a visual aid.

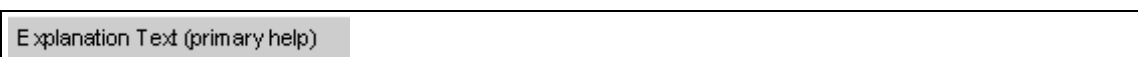
- Do not use the element property **imageAlt** since it will not be read by screen readers in accessibility mode.

## 2.7.2 Explanation

The **Explanation** UI element (shown in Figure 2.10) is a primary help element. It is visible only when help mode is activated.

### Note:

Note that in Release **NetWeaver 7.1** this element is available only in Web Dynpro ABAP.



### Figure 2.10: Explanation Element

The user can activate or deactivate the help texts. The relevant URL parameters are as follows:

- Primary help on: `?sap-explanation=x`
- Primary help off: `?sap-explanation=`

### Don'ts for Explanation Elements

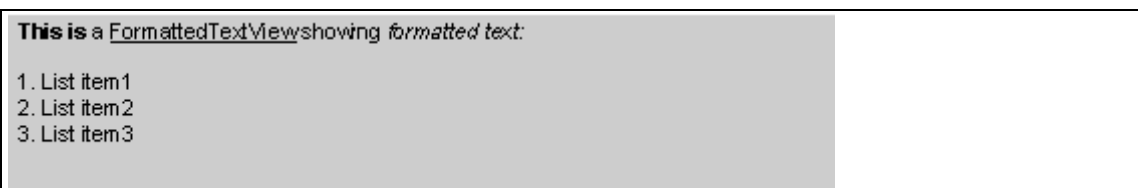
- Do not use the Explanation element to convey any important information about how to use the View element. This is because the **Explanation** element is used for primary help and can be deactivated.

## 2.7.3 FormattedTextView

The **FormattedTextView** UI element (as shown in Figure 2.11) is used to display formatted text

### Note:

Note that in Release **NetWeaver 2004s** this element is available only in Web Dynpro ABAP.



### Figure 2.11: FormattedTextView Element

No additional descriptions are generally required for the FormattedTextView element. Depending on the context, however, you may need to enter a text in the tooltip of this element. Too much text can be confusing, so use this option with care.

### Dos for FormattedTextView Elements

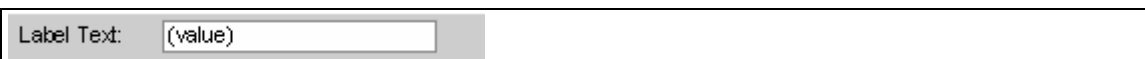
- Use the **tooltip** property of the element to add semantic information.
- An additional label for the element is optional.

## Don'ts for FormattedTextView Elements

- Do not place any important information in the tooltip of the label since it will not be read by screen readers in accessibility mode.

## 2.7.4 InputField

The **InputField** UI element (as shown in Figure 2.12) must always be used in combination with the Label element. This ensures that screen readers can identify the relationship between the two elements. It also ensures that they read the correct label text when the keyboard focus moves to the InputField element.



**Figure 2.12: InputField Element**

Labels are generally mandatory, but there are certain situations where the InputField element cannot have a Label element. In this case, the InputField element is specified more exactly by ABAP/Java Data Dictionary (DDIC), as long as the appropriate data type is specified there.

## Dos for InputField Elements

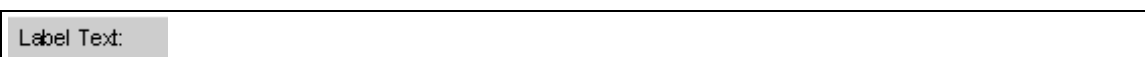
- Always create labels with meaningful texts for InputField elements.
- Use the **tooltip** property of the element to add semantic information.

## Don'ts for InputField Elements

- Do not use the **explanation** property of the element to convey any important information about how to use the View element. This is because the **explanation** element is used for primary help and can be deactivated.
- Do not place any important information in the tooltip of the label since it will not be read by screen readers in accessibility mode.

## 2.7.5 Label

The **Label** UI element (as shown in Figure 2.13) is used to describe the content of many other elements, such as InputField, Dropdown, ListBox, TextEdit, RadioButton, and CheckBox. A label is never used in isolation. It is always combined with other elements. The screen reader reads the label text when the focus moves to the element associated with the label.



**Figure 2.13: Label Element**

Associating a label with an element simplifies keyboard and mouse interaction, especially for smaller elements such as checkboxes. This is because:

- The user can identify the current focus more easily, since the label is enclosed in the focus rectangle (as is the case for checkboxes and radio buttons, for example).
- The user can click the label to activate the field or move the focus to the field.

Never use Label elements on their own, for example as a description for a View element. Always use the **TextView** element instead.

## Dos for Label Elements

Label association with elements:

- Always use the **labelFor** property to associate Label elements with other elements. This is particularly important if you use Label elements for typical field elements, such as `InputText` or `DropDownByIndex/DropDownByKey`.
- You can use Label elements with complex elements such as `RoadMaps`, but this is optional. The use of labels for elements that already have a descriptive **text** property (such as `Checkboxes` or `TextViews`) is also optional.
- Row headers and column headers replace the label in field elements contained in the cells of Table elements. This means that labels are not required in Table elements.

Alternatives for Label elements

- If you cannot use a label with a field element, for example because there is not enough space, you must create an appropriate label text in the **tooltip** property of the field element. The tooltip text must contain at least the label text.

Multiple Label elements with a single element

- In Web Dynpro, you cannot assign more than one Label element to a field. If you require multiple texts around an `InputField` element, then you can use a combination of **InputField** with **Label** and **TextView**.

Multiple elements with a single Label element

- In Web Dynpro, you cannot associate a single Label element with more than one other element. If multiple identical label texts exist, use multiple Label elements with the same label text.
- If multiple field elements require a common label, put the field elements in a **TransparentContainer** and associate it with a **Label** element. Handle the individual elements in the **TransparentContainer** element as described in the relevant sections about this element below.

## Don'ts for Label Elements

- Do not place any important information in the tooltip of the Label element since it will not be read by screen readers in accessibility mode.
- Never use Label elements in isolation. Always associate them with other elements.

## 2.7.6 Special Cases for Label Elements

This chapter discusses some special cases for the use of Label elements.

- **Preselection field** (also known as a label shuffler)

In Figure 2.14, the first Dropdown element is used as a label for the second Dropdown element. The content of the second element changes depending on which item is selected in the first Dropdown element. This type of assignment is also possible for text fields. The first Dropdown element is known as a *preselection field*. More information on assigning a preselection field is available in the sections on the elements **DropDownByIndex** and **ByKey**.





**Figure 2.14: Preselection Field**

- **Field elements as a label**

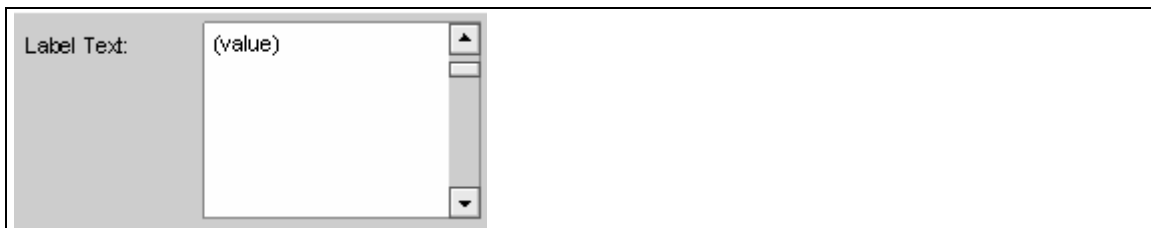
If you use a field element (or a TextView element) to display additional information about preceding or succeeding fields, the field element does not require a label. Specify its purpose in the tooltip of the field.

- **InputFields followed by buttons**

If you combine field elements with Button elements, always assign a Label element to the field element. If the purpose of the Button element is not immediately obvious from the text, then it requires a tooltip that explains its relationship with the preceding field. A good example of this is a “Search” field followed by a “Start” button. In this case, the tooltip of the Button element could contain the text “Start searching”.

## 2.7.7 TextEdit

The **TextEdit** UI element (as shown in Figure 2.15) is used to enter and display multi-line text.



**Figure 2.15: TextEdit Element**

### Dos for TextEdit Elements

- Always create a label with a meaningful text for a TextEdit field.
- Use the **tooltip** property of the element to add semantic information.

### Don'ts for TextEdit Elements

- Do not place any important information in the tooltip of the label since it will not be read by screen readers in accessibility mode.

## 2.7.8 TextView

The **TextView** UI element (as shown in Figure 2.16) is used to display text only.



**Figure 2.16: TextView Element**

You do not generally need to provide additional descriptions for `TextView` content. However, it can sometimes be necessary to place additional information in a tooltip, depending on the context in which a `TextView` element is used. Take care not to confuse the user by creating texts that are too long.

### Dos for `TextView` Elements

- Always create a label if you use the `TextView` element as a visual replacement for `InputField` elements with the `readOnly` property.
- A label is optional if you use the `TextView` element as an additional explanation text.
- Use the `tooltip` property to provide information about the semantic context. This is particularly important if there is no content at runtime. If you know that there will never be content at runtime, query the fill status, and hide the `TextView` element using the property `visible=none`.

### Don'ts for `TextView` Elements

- Do not place any important information in the tooltip of the label since it will not be read by screen readers in accessibility mode.

### Further comments

A `TextView` element with an empty `text` property has a tab stop in accessibility mode when the user moves the keyboard focus to the element. If you associate a label with the `TextView` element, the screen reader can read the label text and announces the text as being “empty”.

## 2.8 Action Elements

This chapter explains each of the UI elements in the standard library of **Action Elements**, and the programming rules. It discusses the following UI elements:

- `Button`
- `ButtonChoice`
- `LinkToAction/LinkToURL`
- `TimedTrigger`

### 2.8.1 Button

The `Button` UI element (as shown in Figure 2.17) represents a pushbutton on the screen. The user can execute statements and actions by activating the button element.

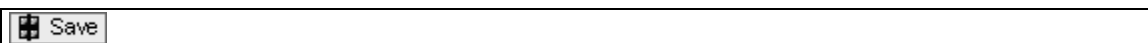


Figure 2.17: Button Element

### Dos for Button Elements

- Make sure that the meaning of a button element is obvious to the user. When you create labels for button elements, use verbs such as **Display** or **Print**.

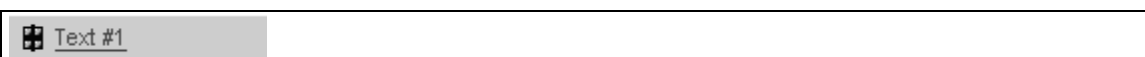
- If the Button element makes significant changes to the screen content or the state of the application, you must indicate this in the text of the element. A Button element that opens a new window should have the text “New Window”, for example.
- If the meaning of a Button element is not immediately obvious to the user, you must create a tooltip. This could be the case if the button text is an abbreviation or a single word.
- Always create a tooltip if the Button element only has a graphical symbol and no text.
- If you have placed both a graphical symbol and a text on your Button element, and the graphical symbol contains additional information not included in the text, provide this information in a tooltip for the Button element.
- If you have specified a Button element as the default Button element, a user can choose [Enter] to press this button even when it does not have the focus. Include this information in the tooltips of any buttons you classify as default buttons.
- To improve usability, important Button elements can be given a graphical symbol as well as a text. Make these important Button elements large enough to include both the symbol and the text. This makes these Button elements easier to see and access with the mouse pointer for users without visual impairments as well.
- Arrange multiple Button elements so that the most important Button element is farthest left (in western countries). You can use the ButtonRow element to do this. The Web Dynpro framework reverses the order of the buttons automatically for countries where the reading direction is right to left.

#### Don'ts for Button Elements

- Do not use the **text** or **tooltip** properties of the element to convey any important information about how to use the View element. This is because the **Explanation** element is used for primary help and can be deactivated.
- Do not use the **imageAlt** property since it will not be read by screen readers in accessibility mode.

## 2.8.2 LinkToAction/LinkToURL

The **LinkToAction/LinkToURL** UI element (as shown in Figure 2.18) is a hypertext link. When the user activates the link, either a Web Dynpro action is executed (LinkToAction) or a new page is displayed (LinkToURL).



**Figure 2.18: LinkToAction/LinkToURL Element**

#### Dos for LinkToAction/LinkToURL Elements

- Use the **tooltip** property of the element to add semantic information.

#### Don'ts for LinkToAction/LinkToURL Elements

- Do not convey important information solely in the image of a link element. Always place this information in the text or tooltip of the link. Link images must only be used as an additional illustration of the link's function.
- Never use links that have no text or icon. Screen readers interpret links without text as inactive links. In these cases, make the link invisible or create a text that indicates the inactive state (such as “Not assigned” or “No information available”).

- Do not use the **imageAlt** property since it will not be read by screen readers in accessibility mode.

### 2.8.3 TimedTrigger

The **TimedTrigger** UI element automatically and periodically triggers an event with a specified delay. TimedTrigger elements are not displayed on the user interface and therefore ignore the tooltip and the **visibility** property. However, they do take up space in certain layouts, such as matrix layouts.

#### Dos for TimedTrigger Elements

If you use a **TimedTrigger** element (for example, to query the server load at periodic intervals), you must implement one of the following solutions:

- Enable the user to define the length of the periodic interval.
- Provide an option for deactivating the automatic query and switching to manual queries.

In these situations, it may be helpful to implement a query about whether the user is in accessibility mode, and then proceed accordingly (refer also to the information about querying accessibility mode at runtime in Chapter 2.6).

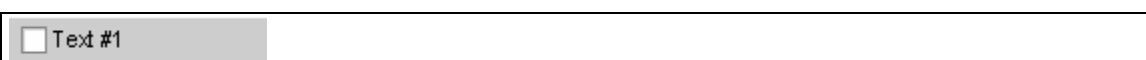
## 2.9 Selection Elements

This chapter explains each of the UI elements in the standard library of **Selection Elements**, and the programming rules. It discusses the following UI elements:

- CheckBox
- CheckBoxGroup
- DropDownByIndex/DropDownByKey
- ItemListBox
- RadioButton
- RadioButtonGroupByIndex/RadioButtonGroupByKey
- ToggleButton
- ToggleLink
- TriStateCheckBox

### 2.9.1 CheckBox

You use the **CheckBox** UI element (as shown in Figure 2.19) to implement a single on/off switch as a checkbox. A checkbox enables the user to select a Boolean value (TRUE/FALSE). The UI element consists of a graphic with text. The checkmark in the box indicates that the option is selected and the value is set to TRUE.



**Figure 2.19: CheckBox Element**

### Dos for CheckBox Elements

- Additional labels are optional. Either use the checkbox label intended for explanatory text or the **text** property of the checkbox. If you use both options, a screen reader will only read the label text.
- Use the **tooltip** property of the element to add semantic information. Information on creating individual tooltips within a group is available under *CheckBoxGroup* below.

### Don'ts for CheckBox Elements

- Do not use the **text** or **tooltip** properties of the element to convey any important information about how to use the View element. The **Explanation** element is used for primary help and can be deactivated.
- Do not place any important information in the tooltip of the optional label since it will not be read by screen readers in accessibility mode.

## 2.9.2 CheckBoxGroup

The **CheckBoxGroup** UI element (as shown in Figure 2.20) enables the user to select one element out of multiple options by marking it with a checkmark. **CheckBoxGroup** arranges the individual **CheckBox** elements in single-column or two-column tables.



**Figure 2.20: CheckBoxGroup Element**

### Dos for CheckBoxGroup Elements

- Use the **accessibilityDescription** property to create a short description (as a replacement for a title). A screen reader will then read this text when the focus moves to the element.
- To create a different tooltip for a given **CheckBox** element, associate the **tooltip** property with an attribute of the context node with which the **text** property of this **CheckBox** element is associated. Otherwise, the same tooltip is used for all **CheckBox** elements in the group.
- You have the option of using an additional label.

### Don'ts for CheckBoxGroup Elements

- Do not use the **text** or **tooltip** properties of the element to convey any important information about how to use the View element. The **Explanation** element is used for primary help and can be deactivated.
- Do not place any important information in the tooltip of the label since it will not be read by screen readers in accessibility mode.

### 2.9.3 DropDownByIndex/DropDownByKey

The **DropDownByIndex/DropDownByKey** UI element (as shown in Figure 2.21) provides the user with a dropdown list box. The user cannot select more than one item from the selection list. The UI element consists of a text field, a button, and a selection list. Any list item already selected is displayed in the text field. When the user presses the pushbutton, the element displays a list of all possible values.



**Figure 2.21: DropDownByIndex/DropDownByKey Element**

#### Dos for DropDownByIndex/DropDownByKey Elements

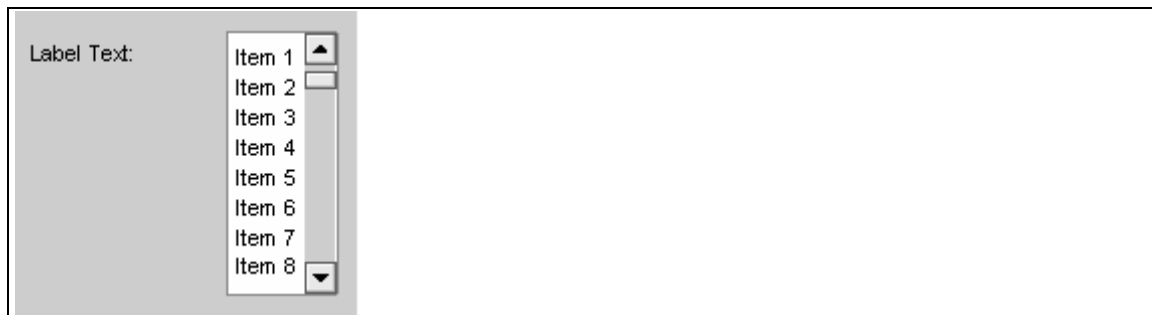
- Always create a label with a filled **text** property for the DropDownByIndex/DropDownByKey element. In situations where you cannot assign a label, a replacement label text is taken from ABAP/Java Data Dictionary (DDIC), as long as the text is specified there.
- The DropDownByIndex/DropDownByKey element can also be used as a label for another field (refer also to *Preselection Field* in the special cases for the Label UI element). Associate another element with the dropdown element by using the **labelFor** property of the dropdown element. The item currently selected by the user in the dropdown element is then used as the label text for the associated element.
- Use the **tooltip** property of the element to add semantic information.

#### Don'ts for DropDownByIndex/DropDownByKey Elements

- Do not position a dropdown element used as a label (a preselection field) behind another associated element.
- Do not use the **text** or **tooltip** properties of the element to convey any important information about how to use the View element. The **Explanation** element is used for primary help and can be deactivated.
- Do not place any important information in the tooltip of the label since it will not be read by screen readers in accessibility mode.

### 2.9.4 ItemListBox

The **ItemListBox** UI element (as shown in Figure 2.22) is similar to the classical GUI concept of a selection list with simple and multiple selection (also known as a list box). The element displays a list of text entries in a box of a fixed size. The user can scroll through these entries. The element displays one or two columns of values. There may also be a column of symbols before the value columns.



**Figure 2.22: ItemListBox Element**

#### Dos for ItemListBox Elements

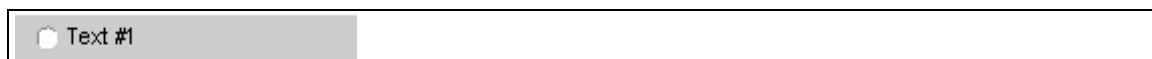
- Always create a label with a filled **text** property for this element
- Use the **tooltip** property of the element to add semantic information. To create a different tooltip for a given item element, associate the **tooltip** property with an attribute of the **dataSource** context node.

#### Don'ts for ItemListBox Elements

- Do not place any important information in the tooltip of the label since it will not be read by screen readers in accessibility mode.

### 2.9.5 RadioButton

Like the Checkbox element, the **RadioButton** UI element (as shown in Figure 2.23) represents a two-state on/off switch. However, it is only intended for use in a group of RadioButton elements since it enables the user to make an exclusive selection from various options. The selected option is assigned by associating the RadioButton elements in a group with the same context property. This property is then given the value of the selected RadioButton element.



**Figure 2.23: RadioButton Element**

#### Dos for RadioButton Elements

An additional label is optional. Either use the radio button label intended for explanatory text or the **text** property of the RadioButton element. If you use both options, a screen reader will only read the label text.

Use the **tooltip** property of the element to add semantic information. Information on creating individual tooltips within a group is available under *RadioButtonGroupByIndex/RadioButtonGroupByKey* below.

#### Don'ts for RadioButton Elements

- Do not place any important information in the tooltip of the label since it will not be read by screen readers in accessibility mode.

## 2.9.6 RadioButtonGroupByIndex/RadioButtonGroupByKey

The **RadioButtonGroupByIndex/RadioButtonGroupByKey** UI element (as shown in Figure 2.24) displays radio buttons grouped in columns and rows. Unlike the **CheckBoxGroup** UI element, this UI element allows the user to select no more than one element within the **RadioButtonGroup** element.



**Figure 2.24: RadioButtonGroupByIndex/RadioButtonGroupByKey Element**

### Dos for RadioButtonGroupByIndex/RadioButtonGroupByKey Elements

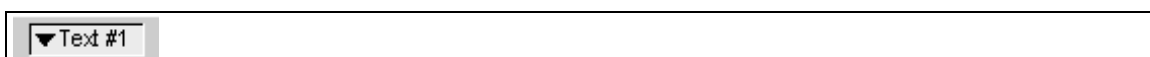
- Use the **accessibilityDescription** property to create a short description (as a replacement for a title). A screen reader will then read this text when the focus moves to the element.
- To create a different tooltip for a given RadioButton element, associate the **tooltip** property with an attribute of the context node with which the **text** property of this RadioButton element is associated. Otherwise, the tooltip of the RadioButtonGroup element is used for all RadioButton elements.

### Don'ts for RadioButtonGroupByIndex/RadioButtonGroupByKey Elements

- Never set the properties **enabled=false** and **readOnly=true** for the group. **readOnly** elements always have the property **enabled=true**.

## 2.9.7 ToggleButton

The **ToggleButton** UI element (as shown in Figure 2.25) is a pushbutton for toggling between states. The user can execute statements and actions by activating the ToggleButton element. A ToggleButton element is a visual variant of the CheckBox element. Its state can be TRUE or FALSE.



**Figure 2.25: ToggleButton Element**

### Dos for ToggleButton Elements

- Use the **tooltip** property of the element to add semantic information.
- If the element triggers an action that is not immediately obvious to a blind user, you must provide semantic information for this element. One example of this is when a new area is suddenly opened within the current view.

### Don'ts for ToggleButton Elements

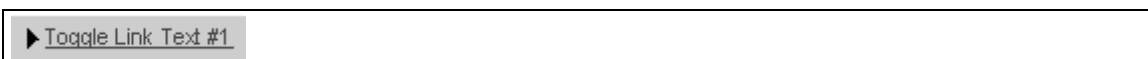
- Do not convey important information solely in the image of a ToggleButton element. Always place this information in the text or tooltip of the ToggleButton element as well. Images in a ToggleButton element must only be used as an additional illustration of the button's function.



- Do not add any status information, such as “pressed” or “opened”. The framework provides the necessary status information.
- Do not use the **imageAlt** property since it will not be read by screen readers in accessibility mode.

## 2.9.8 ToggleLink

The **ToggleLink** UI element (as shown in Figure 2.26) is used to display a hypertext link for an extended search. A ToggleLink element is a visual variant of the CheckBox element. Its state can be TRUE or FALSE.



**Figure 2.26: ToggleLink Element**

### Dos for ToggleLink Elements

- Always fill the **text** property.
- Use the **tooltip** property of the element to add semantic information.
- If the element triggers an action that is not immediately obvious to a blind user, you must provide semantic information for this element. One example of this is when a new area is suddenly opened within the current view.

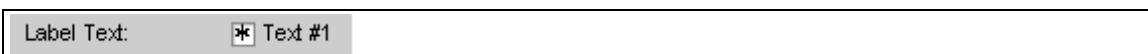
### Don'ts for ToggleLink Elements

- Do not add any status information, such as “pressed” or “opened”. The framework provides the necessary status information.
- Do not use the **imageAlt** property since it will not be read by screen readers in accessibility mode.

## 2.9.9 TriStateCheckBox

The **TriStateCheckBox** UI element (as shown in Figure 2.27) is similar to a CheckBox element, but it can have one of three states:

- An option can be activated and selected.
- An option can be deactivated and selected.
- An option is not defined.



**Figure 2.27: TriStateCheckBox Element with Associated Label**

### Dos for TriStateCheckBox Elements

- An additional label is optional. Either use the TriStateCheckBox label intended for explanatory text or the **text** property of the TriStateCheckBox element. If you use both options, a screen reader will only read the label text.
- Use the **tooltip** property of the element to add semantic information.

### Don'ts for TriStateCheckBox Elements

- Do not place any important information in the tooltip of the label since it will not be read by screen readers in accessibility mode.

## 2.10 Complex Elements

This chapter explains each of the UI elements in the standard library of **Complex Elements**, and the programming rules. It discusses the following UI elements:

- BreadCrumb
- DateNavigator
- Legend, LegendItem
- RoadMap
- PhaseIndicator
- Table
- Tree

### 2.10.1 BreadCrumb

The **BreadCrumb** UI element displays the current page in the context of a navigation path. You can use the BreadCrumb element to display a history of the pages last visited or to visualize the structure of information. A BreadCrumb element can consist of a single link or multiple links.

In **Single Link** mode (as shown in Figure 2.28), the tooltip of the BreadCrumb element is displayed for all steps (BreadCrumbSteps) of the element.

Start Order > Select products > Select payment method > Enter address information > Send order

**Figure 2.28: BreadCrumb Element in Single Link Mode**

In **Multiple Link** mode (as shown in Figure 2.29), the tooltip of the BreadCrumb element is displayed for the last (selected) step of the element.

Start Order > Select products > Select payment method > Enter address information > Send order

**Figure 2.29: BreadCrumb Element in Multiple Link Mode**

### Dos for BreadCrumb Elements

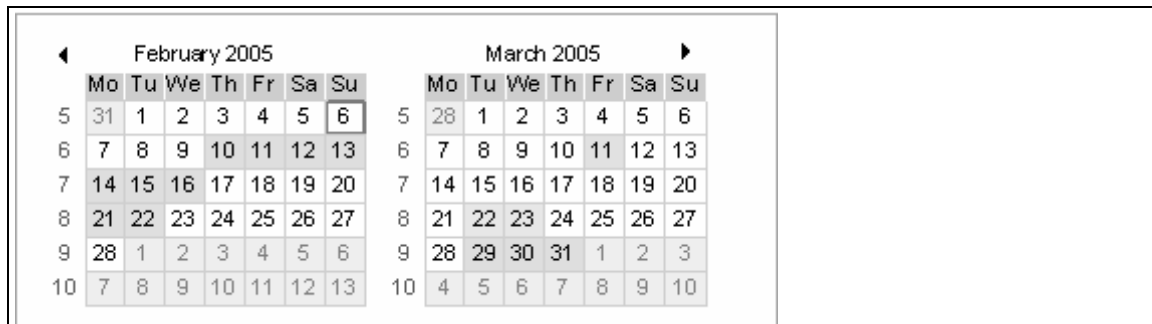
- In **Single Link** mode, use the tooltip of the BreadCrumb element to provide semantic information for the entire element.
- In **Multiple Link** mode, use the tooltip of a single BreadCrumbStep element to provide semantic information for the current step. (Single Link mode ignores the tooltips of the individual steps.)

### Don'ts for BreadCrumb Elements

- Do not add any important information to the **separatorText** property. Screen readers do not read this property in accessibility mode. **separatorText** is used to separate the individual links. The default separator character is ">".

## 2.10.2 DateNavigator

The **DateNavigator** UI element (as shown in Figure 2.30) enables users to display and enter dates. Its features include the ability to navigate in a calendar and select a day, month, year, or range of dates. However, its main role is to help users enter dates in the correct format.



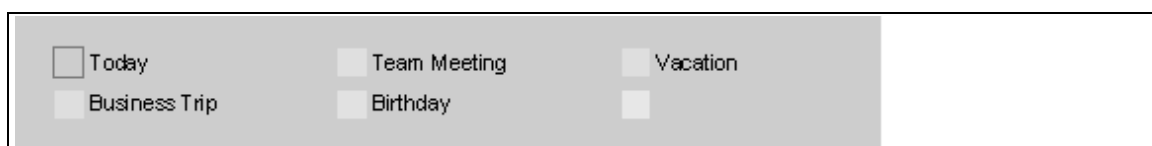
**Figure 2.30: DateNavigator Element**

### Dos for DateNavigator Elements

- Use the **accessibilityDescription** property to create a short description (as a replacement for a title). A screen reader will then read this text when the focus moves to the element.
- Use the tooltip of the DateNavigator element for all days that do not have the **dateNavigatorMarking** property assigned to them. You can use the **dateNavigatorMarking** property to create a separate tooltip for a specific date.
- Use the **Legend** UI element (as discussed in the section below) to assign semantic colors. Specify the ID of the **Legend** element in the **legendId** property of the DateNavigator element. The text of the **Legend** item is added to the calendar automatically.

## 2.10.3 Legend, LegendItem

The **Legend** UI element (as shown in Figure 2.31) allows you to display a descriptive text for the colors used in an assigned UI element.



**Figure 2.31: Legend Element with LegendItem Elements**

You can place the Legend element anywhere in the View object. You can then assign this element to a Table element or a DateNavigator element.

### • Assigning the Legend element to the DateNavigator element:

Insert a Legend element after the DateNavigator element in the View object. Then assign the Legend element to the DateNavigator element by setting the ID of the Legend element as the **legendId** property of the DateNavigator element.

### • Assigning the Legend element to the Table element:

Insert a Legend element after the Table element and use the **legendId** property to assign it to the Table element. To position the Legend element at the bottom of the table, you can use the **LegendPopin** aggregation. Insert a **LegendPopin** aggregation in the Table ele-

ment. Then create content in **LegendPopin**. You can then insert a Legend element into the content.

- **Color assignment of the LegendItem element:**

The **tableCellDesign** property is used to assign colors in the **LegendItem** UI element. The following properties are of this type:

- The **semantics** property for LegendItem elements
- The **daySemantics** property for the **dateNavigatorMarking** property.
- The **cellDesign** property of TableColumn for Table elements

#### Dos for Legend and LegendItem Elements

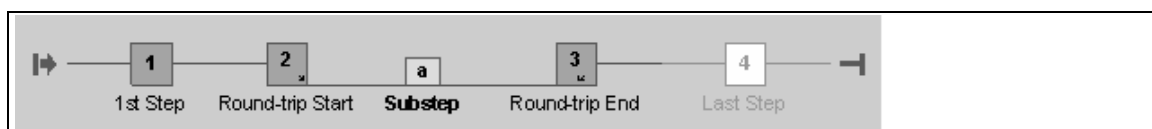
- Use the **Legend** UI element to assign semantic colors. Use the **FrameworkLegendItem** element to describe any special semantics (such as a special indicator for today's date).
- Use the tooltip of the **LegendItem** or **FrameworkLegendItem** element to add semantic information.

#### Don'ts for Legend and LegendItem Elements

- Do not use the **tooltip** property for the **Legend** and **LegendItem** elements. Screen readers will not read the tooltip of the **LegendItem** element from the associated UI element. Screen readers also never read tooltips created for the **Legend** element.

## 2.10.4 RoadMap

The **RoadMap** UI element (as shown in Figure 2.32) visualizes the steps of a wizard. Each step is represented by a separate **RoadMapStep** or **MultipleRoadMapStep** object. You can use various symbols to mark the starting points and end points of this UI element. You can assign the value **more** to the **startPointDesign** or **endPointDesign** property to indicate to the user that there are other steps before the first visible step, or after the last visible step.



**Figure 2.32: RoadMap Element**

The **RoadMap** UI element is well suited to displaying step-by-step workflows. A team developing an application can use it to visualize small steps in a clearly defined work process.

#### Dos for RoadMap Elements

- Use the **accessibilityDescription** property to create a short description (as a replacement for a title). A screen reader will then read this text when the focus moves to the element.
- Use the tooltip of the RoadMap element to add a longer semantic description of the element.
- Use the tooltips of the individual **RoadMapStep** elements to add semantic descriptions for these elements.

#### Don'ts for RoadMap Elements

- Do not add any status information to the tooltip, such as "Selected step". The framework provides the necessary status information.

## 2.10.5 PhaseIndicator

Like the **RoadMap** UI element, the **PhaseIndicator** element (as shown in Figure 2.33) visualizes the steps of a wizard. Each step is represented by a separate **Phase** object. An application development team can display larger steps using this element than with the RoadMap element. These are steps that may take the user more time to complete.



**Figure 2.33: PhaseIndicator Element**

### Dos for PhaseIndicator Elements

- Use the **accessibilityDescription** property to create a short description (as a replacement for a title). A screen reader will then read this text when the focus moves to the element.
- Use the tooltip of the PhaseIndicator element to add a longer semantic description of the element.
- Use the tooltips of the individual PhaseIndicator elements to add semantic descriptions for these elements.

### Don'ts for PhaseIndicator Elements

- Do not add any status information to the tooltip, such as the selected phase, phase type, or number of phases. The framework provides the necessary status information.

## 2.10.6 Table

In a Web Dynpro **Table** element, data is displayed two-dimensionally in table cells arranged in rows and columns. The Table UI element in Web Dynpro consists of a parent **Table** (as shown in Figure 2.34) and multiple TableColumn and TableRow elements.

Orders					
Edit					
Order Number	Order Type	Language	Created By	Date / Time	
8100001	Inbound Request	Shipping Instruction	GER	Catherine Kennedy	11/12/2005 11:22:33
9100002		Shipping Instruction	ENG	Bob Frazier	11/12/2005 11:11:11
A-0006	Core Burner	Customer Wish	CHN	April Woo	11/02/2005 05:11:11

**Figure 2.34: Table Element with Title Row and ToolBar Element**

The Table UI element contains properties that apply to the entire element. One example is the property specifying whether the table is read-only. (**readOnly=true** indicates that the Table element is read-only.)

The **TableColumn** UI element has further properties:

- Properties that apply only to the column header
- Properties that apply to the whole column
- Properties that apply to cells and whose values can vary from cell to cell

Column headers are implemented by an aggregation of Caption elements. The second aggregation of the TableColumn element, **TableCellEditor**, contains a UI element that is used to

display the cells in the column. In accessibility mode, paginators are used to control movements within data records.

### Dos for Table Elements

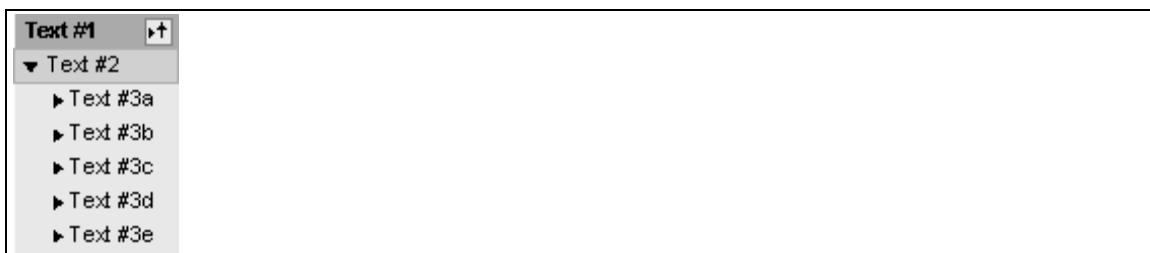
- Create a title for each Table element. If the element does not have a title for design reasons, use the **accessibilityDescription** property of the Table element. Screen readers read the **accessibilityDescription** property when the focus is moved to the Table element.
- Use the **tooltip** property of the Table element to add semantic information about the element.
- Use the **tooltip** property of the **TableCellEditor** element to add semantic information. If the column header is too small for its description, you can create a different tooltip for each cell in a column by assigning tooltips to the appropriate context nodes. If you do not use the **RowTablePopin** element to show or hide a **TablePopin** element, for example, and you use a Link element in a cell instead, you must include this information in the tooltip.
- Always create a text for the header cell. If you do not do this, blind users cannot identify which column currently has the focus.
- The above recommendations also apply to the **TableColumn**, **TableColumnGroup**, **TreeByKeyTableColumn**, and **TreeByNestingTableColumn** elements.
- Use semantic (named) colors for cell backgrounds (for example, set the property **cellDesign** to the value `BADVALUE_MEDIUM`).

### Don'ts for Table Elements

- Do not add any status information to the tooltip, such as the column number or row number. The framework provides the necessary status information. Any status information you include in the tooltip will be redundant.

## 2.10.7 Tree

You can use the **Tree** UI element (as shown in Figure 2.35) to visualize context-defined hierarchies in Web Dynpro.



**Figure 2.35: Tree Element**

Each tree node is given a text and an indicator that can be activated. This indicator is used to open and close the hierarchy levels. The indicator also shows the current status. If a node has no subnodes it is known as a **TreeItem** and does not have an indicator. Both nodes and items can be activated and the node text can be supplied by an optional additional image.

### Dos for Tree Elements

- You can give the Tree element a (visible) title. If you do not use a title for design reasons, put the information in the tooltip of the element.
- Make sure that each tree node has a text that explains its meaning. If the Tree element is a column-based element, create texts for the header lines (**TreeColumnHeader**) as well.

- Use the tooltips of the tree nodes and Treeltems to add semantic information. To create a different tooltip for a tree node or Treeltem element, assign the tooltip to the appropriate context node.

### Don'ts for Tree Elements

- Do not use the **iconAlt** property for tree nodes and Treeltems since it is ignored in accessibility mode.
- Do not convey important information solely in the image of a tree node or Treeltem element. Always place this information in the text or tooltip of the element. Images are used only as additional illustration for a function.
- Do not add any status information to the tooltip, such as the hierarchy levels. The framework provides the necessary status information. Any status information you include in the tooltip will be redundant.

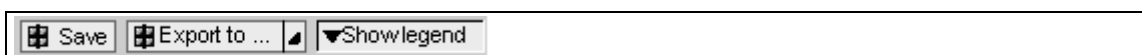
## 2.11 Layout Elements

This chapter explains each of the UI elements in the standard library **Layout** and the programming rules. It discusses the following UI elements:

- ButtonRow
- Group
- TabStrip
- TransparentContainer
- Tray

### 2.11.1 ButtonRow

The **ButtonRow** UI element (as shown in Figure 2.36) is used to create an ordered arrangement of multiple Button elements. You can insert both Button elements and ToggleButton elements in this element. The **ButtonRow** element includes the methods needed to create and manage the Button elements. It has no properties other than this.



**Figure 2.36: ButtonRow Element**

### Dos for ButtonRow Elements

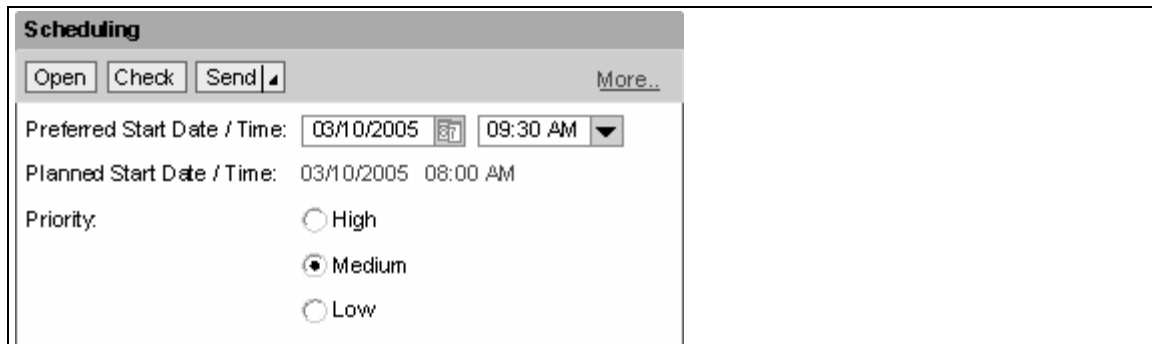
- Use the ButtonRow element for layout purposes, for example to arrange a group of Button elements in a row.
- Arrange multiple Button elements so that the most important Button element is farthest left (in western countries).

### Don'ts for ButtonRow Elements

- Do not place any important information in the tooltip of the **ButtonRow** element since it will not be read by screen readers in accessibility mode.

## 2.11.2 Group

You can use the **Group** UI element (as shown in Figure 2.37) to group a series of UI elements under the same title. Therefore, it acts as a container. This UI element resembles a display panel with a colored or white background.



**Figure 2.37: Group Element with Toolbar Element and Other Elements**

### Note:

In the **Group** UI element, the **enabled** property has no effect on the elements you insert in the **Container** UI element. If you set the **enabled** property to **false** in the Group UI element, for example, an input field in the group is not deactivated automatically. If you also want to deactivate the elements in this Group element, you must set the relevant property for each UI element *separately*.

### Dos for Group Elements

- Use the **title** property of the group header element if the group title is visible.
- Use the **accessibilityDescription** property to create a short description (as a replacement for a title). A screen reader will then read this text when the focus moves to the element.
- Use the **tooltip** property of the element to add semantic information.

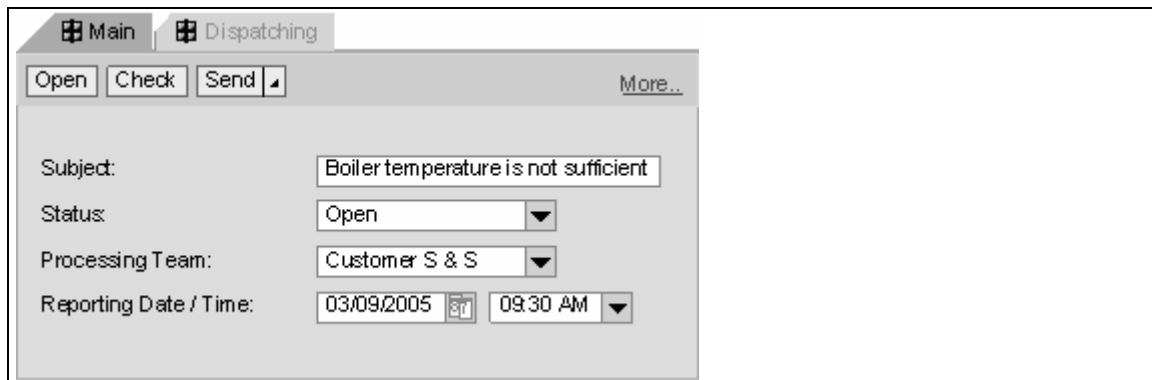
### Don'ts for Group Elements

- Do not nest Group elements within each other. Always use the **TransparentContainer** UI element for nested groups.
- Do not convey important information solely in the image of a group header element. Always place this information in the text or tooltip of the group header element. Images in group header elements are used only as additional illustration for a function.
- Do not use the **imageAlt** and **tooltip** properties for the group header element since they are ignored in accessibility mode.

## 2.11.3 TabStrip

The **TabStrip** UI element (as shown in Figure 2.38) displays a tabstrip. The user can toggle between tab pages by selecting different tab titles. The user can display the content of a tab by selecting its title. Each of the tab pages in turn uses the same area of the window to display its content.





**Figure 2.38: TabStrip Element with Toolbar Element and Other Elements**

If no **selectedTab** property is specified for the **TabStrip** element, or if the tab specified in the **selectedTab** property is not visible, Web Dynpro displays the first visible tab instead.

### Dos for TabStrip Elements

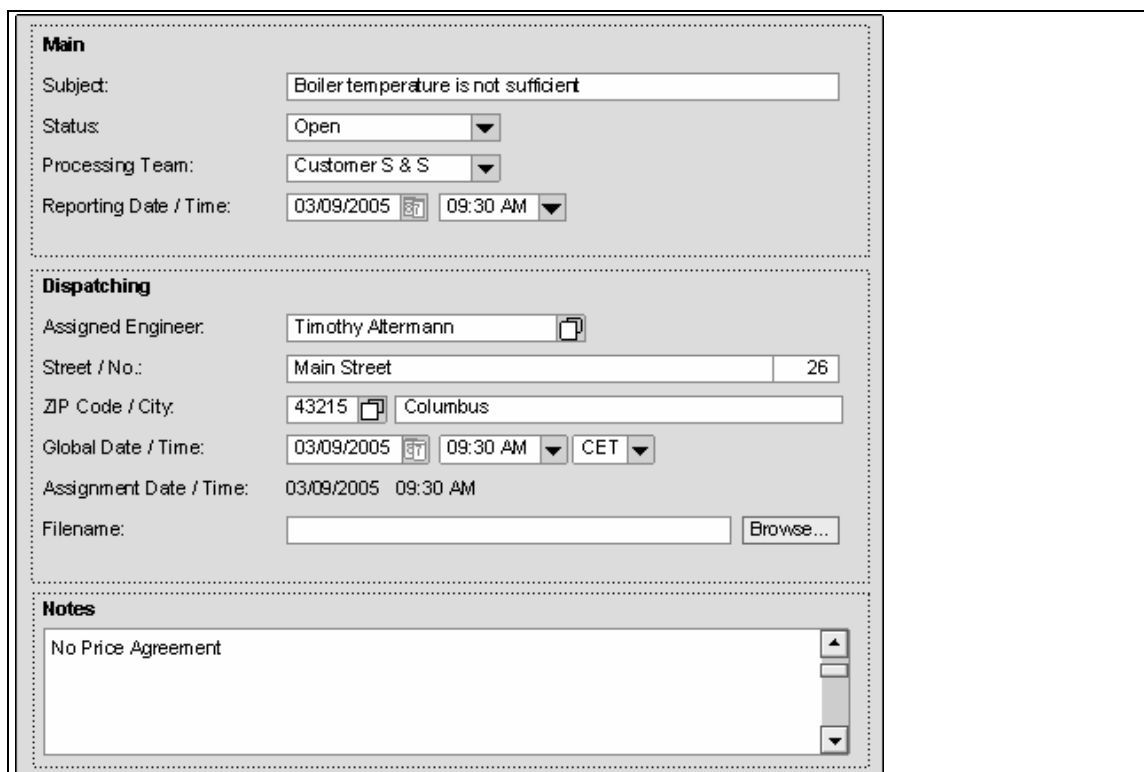
- Use the **accessibilityDescription** property to create a short description (as a replacement for a title). A screen reader then reads this text when the focus moves to the element.
- Use the **tooltip** property of the element or of the individual tabs to add semantic information.

### Don'ts for TabStrip Elements

- Do not nest TabStrip elements within each other. Always use the **TransparentContainer** UI element for nested groups.
- Do not convey important information solely in the image of a tab header. Always place this information in the text or tooltip of the element. Images in a tab header element must only be used as an additional illustration of the tab header's function.
- Do not use the **imageAlt** property of the tab header since it will not be read by screen readers in accessibility mode.

## 2.11.4 TransparentContainer

The **TransparentContainer** UI element (as shown in Figure 2.39) is a container for UI elements that have no visual display. A TransparentContainer UI element is transparent and can be filled with any number of other UI elements. The TransparentContainer UI element also allows you to use the defined layout to arrange the UI elements inserted in it. Figure 2.39 shows some TransparentContainer elements. The frames in this figure (dotted lines) are normally invisible.



**Main**

Subject:

Status:

Processing Team:

Reporting Date / Time:

---

**Dispatching**

Assigned Engineer:

Street / No.:

ZIP Code / City:

Global Date / Time:

Assignment Date / Time: 03/09/2005 09:30 AM

Filename:

---

**Notes**

**Figure 2.39: Some TransparentContainer Elements**

You can use the TransparentContainer element in two different ways:

- **As a layout container (property isLayoutContainer=true)**  
In this case, you use the element to design the layout of other UI elements. The TransparentContainer element does not have a tab stop and is not displayed in accessibility mode.
- **As a grouping container (property isLayoutContainer=false)**  
In this case, you use the element to group other UI elements. The TransparentContainer element then has a semantic meaning, like a Group element. It has a tab stop and the **accessibilityDescription** property can be read by a screen reader.

The group title in Figure 2.39 is a TextView element with **design=GroupTitle**. Remember that, in accessibility mode, screen readers do not read the group title when the focus is on the TextView element. If you have filled the **accessibilityDescription** property of the TransparentContainer element, the screen reader reads this property instead.

If you use the TransparentContainer element as a grouping container for accessibility reasons, you can create a title for this group in one of two ways:

- Use the **accessibilityDescription** property of the TransparentContainer element as a group title. Do not use a TextView element as a group title element.  
We recommend this option if you do not want a visible group title.
- Use the SectionHeader element as a visible group title. This element then replaces the **accessibilityDescription** property.

Proceed as described below in these cases:

- **Grouping Without Title**
  1. Set the property **isLayoutContainer=false**.
  2. In the **accessibilityDescription** property, create a text to replace the group title.

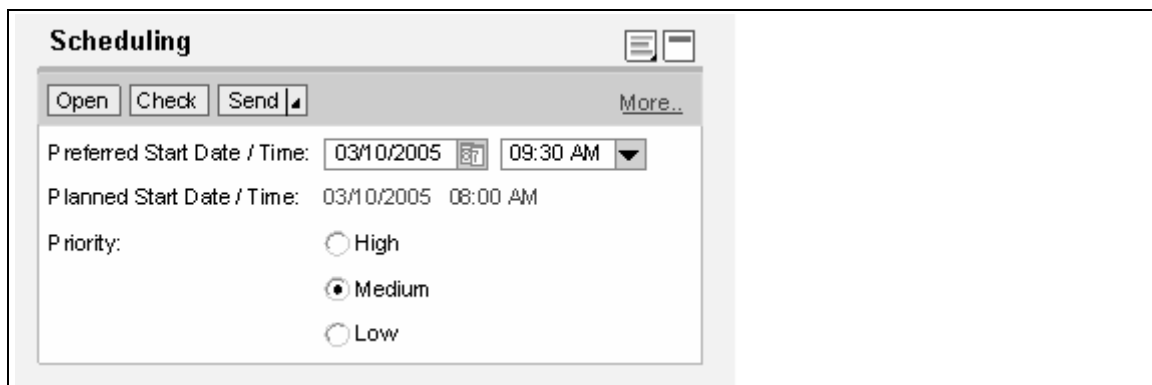
- **No Grouping, Layout Only**

1. If the `TransparentContainer` element is used for layout reasons only, and does not contain any relevant information, set the property `isLayoutContainer=true`. Screen readers do not read the `accessibilityDescription` property in this case.
2. If you use the elements as layout containers, arrange them so that no unexpected changes are made to the tab chain. Nest `TransparentContainer` elements in a way that avoids making the tab chain jump backwards and forwards from one group to another.

For example, if you want to display a inspection text in a `TextView` element after an `InputField` element, use a `TransparentContainer` element with **MatrixLayout** (not `FlowLayout`), to group the `InputField` and its following text. To make this construction accessible in this context, associate a label with the `TransparentContainer` element (using the `labelFor` property). Create an additional tooltip for the field and the inspection text. This tooltip is optional for the field, but mandatory for the `TextView` element.

## 2.11.5 Tray

You can use the **Tray** UI element (as shown in Figure 2.40) to arrange a set of UI elements under the same header. Like the `Group` element, this element is a container for other UI elements. However, unlike the `Group` element the `Tray` element offers you a range of other functions, such as the ability to open and close the element.



**Figure 2.40: Tray Element with ToolBar Element and Other Elements**

### Do's for Tray Elements

- Use the `tooltip` property of the element to add semantic information.
- Use the `accessibilityDescription` property to create a short description (as a replacement for a title). A screen reader then reads this text when the focus moves to the element.

### Don'ts for Tray Elements

- Do not nest `Tray` elements within each other. Always use the `TransparentContainer` UI element for nested groups.
- Do not convey important information solely in the image of a tray header. Always place this information in the text or tooltip of the tray header. Images in a tray header must only be used as an additional illustration of the tray header's function.
- Do not use the `imageAlt` and `tooltip` properties of the tray header since they are ignored in accessibility mode.

## 2.12 Graphical Elements

This chapter explains each of the UI elements in the standard library **Graphics** and the programming rules. It discusses the following UI elements:

- ValueComparison
- Image
- ProgressIndicator

### 2.12.1 ValueComparison

The ValueComparison UI element (as shown in Figure 2.41) is used for the graphical display of different values within a horizontal bar. Next to a 100% marker, for example, you can show values that go over 100%.



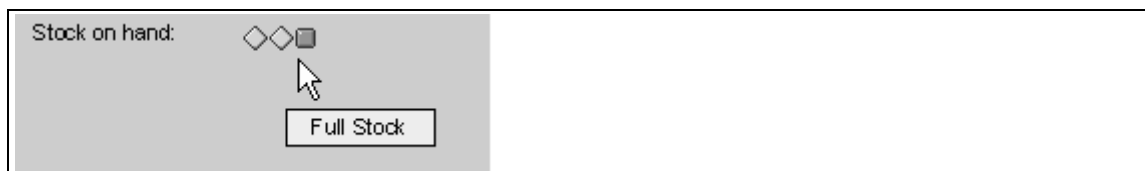
**Figure 2.41: ValueComparison Element**

#### Dos for ValueComparison Elements

- Specify all information about the values of the element in the **tooltip** property of the element. You must do this yourself because the framework does not specify this information.
- You must use the tooltip to explain the values of the element if you use the element in a table cell.
- If you use the element in a separate View element, you can also use other UI elements to explain the values, in addition to the tooltip.

### 2.12.2 Image

The **Image** UI element (as shown in Figure 2.42) enables you to integrate graphics and images into a web application in a format that a web browser can process (for example, GIFs, JPGs, and PNGs).



**Figure 2.42: Image Element with Meaningful Tooltip**

Accessibility regulations do not forbid the use of graphics and images even though blind users do not benefit directly from the information in them. However, you must always provide additional information about the content of the image, for example in its tooltip. Similarly, you must also make sure that any information conveyed in the colors of images is made accessible to users with color vision deficiencies.

Figure 2.43 shows the three states of a traffic light icon. The active light is not just indicated by its color (or the shade of gray used in a printout), but also by its shape and position ("critical" on

the left, "OK" on the right). This means that no information is lost when the image is displayed in gray.



**Figure 2.43: Traffic Light Example**

### Dos for Image Elements

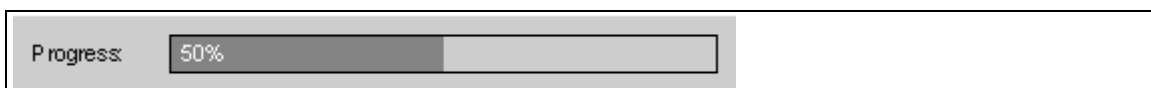
- A Label element is optional for an Image element
- If you do not create a label for the description of an image, you must use its tooltip instead. The tooltip is displayed directly when the image is not available.
- As shown in Figure 2.42, it may occasionally be a good idea to create a different text in the label than in the tooltip. In the example, the label indicates the semantics of the image ("Warehouse Inventory"), whereas the tooltip gives you information about the current status ("Warehouse is full").
- When you use semantic images (icons), also consider what is conveyed by their color and shape (as shown in Figure 2.43).
- If you use Image elements to control actions, make sure that images with the same meaning always appear in the same place in the application. Consistency is a major criterion for accessibility.
- If the content of the image is not relevant to the context (for example, because it is only used for decoration) and the information is available in text form elsewhere in the view object, you can remove the image from the tab chain. Also set the Image property **isDecorative=true**.

### Don'ts for Image Elements

- Do not use the **imageAlt** property since it will not be read by screen readers in accessibility mode.

## 2.12.3 ProgressIndicator

The **ProgressIndicator** UI element (as shown in Figure 2.44) displays the progress of an activity in a scale on a horizontal progress bar. The scale uses the value that you have assigned to the **percentValue** property.



**Figure 2.44: ProgressIndicator Element**

You can use the **displayValue** property to display a text, such as a percentage, in the ProgressIndicator in the UI element. This makes it possible to create descriptions for specific percentage values. You can use the **showValue** property to hide the value of **displayValue**.

You can use the **barColor** property to change the color of the ProgressIndicator element. You can assign the **Menu** property to a ProgressIndicator element.

### Dos for ProgressIndicator Elements

- Use the **tooltip** property of the element to add semantic information.
- You can create a label for the ProgressIndicator element.

### Don'ts for ProgressIndicator Elements

- Do not add any properties such as **barColor**, **percentValue**, or **displayValue** to the tooltip text. The framework provides this information.

## 2.13 Integration Elements

This chapter explains each of the UI elements in the standard library of **Integration Elements**, and the programming rules. It discusses the following UI elements:

- InteractiveForm
- FileDownload
- FileUpload

### 2.13.1 InteractiveForm

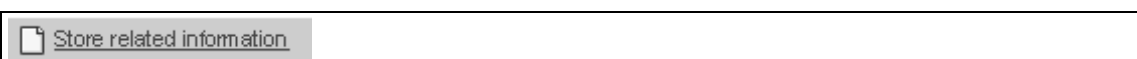
You can use the **InteractiveForm** UI element to insert an interactive or non-interactive *SAP Interactive Form by Adobe* into a view object. This also enables you to create and design a form from scratch.

#### Dos for InteractiveForm Elements

- Use the **tooltip** property of the element to add semantic information about its embedded content.
- Make sure that the embedded content is accessible. More information is available in the *Accessibility Guidelines for SAP Interactive Forms by Adobe*.

### 2.13.2 FileDownload

The **FileDownload** UI element (as shown in Figure 2.45) is visualized as a link and is used to load files from the server to the client. The data of a FileDownload element is accessed when the user requests a specific file by clicking a link.



**Figure 2.45: FileDownload Element**

#### Dos for FileDownload Elements

- Use the **tooltip** property of the element to add semantic information.
- Additional labels are optional.

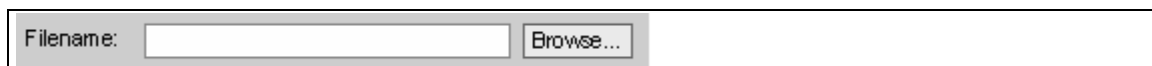
#### Don'ts for FileDownload Elements

- Do not convey important information solely in the image of a FileDownload element. Always place this information in the text or tooltip of the FileDownload element. Images are used only as additional illustration for the FileDownload element's function.

- Do not use the **imageAlt** property since it will not be read by screen readers in accessibility mode.
- Do not place any important information in the tooltip of the label since it will not be read by screen readers in accessibility mode.

### 2.13.3 FileUpload

The **FileUpload** UI element (as shown in Figure 2.46) is used to upload files from the client to the server. The FileUpload element is displayed with an InputField element, in which the directory path and the file name appear. It also has a Button element for searching for the file.



**Figure 2.46: FileUpload Element**

#### Dos for FileUpload Elements

- Use a Label element for this element.
- Use the **tooltip** property of the element to add semantic information.

#### Don'ts for FileUpload Elements

- Do not place any important information in the tooltip of the label since it will not be read by screen readers in accessibility mode.
- Do not use the **text** or **tooltip** properties of the element to convey any important information about how to use the View element. The **Explanation** element is used for primary help and can be deactivated.

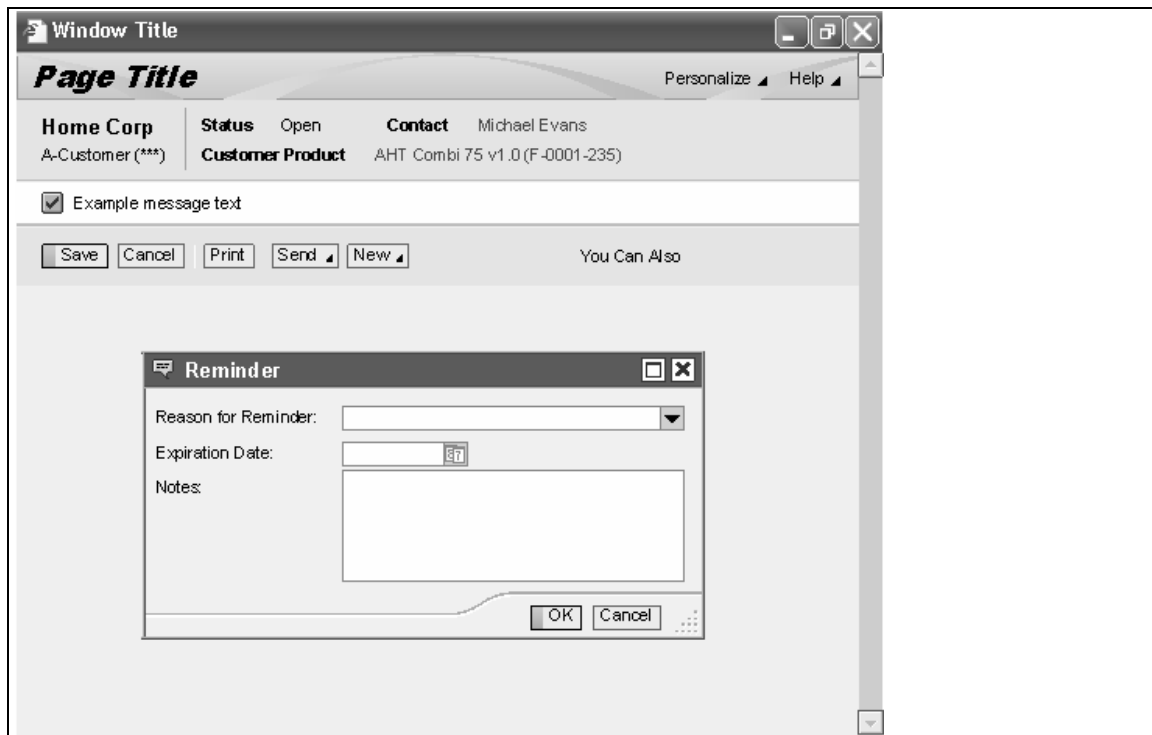
## 2.14 Other Elements

This chapter explains each of the UI elements in the standard library **Other Elements**, and the programming rules. It discusses the following UI elements:

- Main window and dialogs
- MenuBar
- Menu
- ToolBar
- SectionHeader
- RowRepeater

### 2.14.1 Main Windows and Dialogs

The following Dos and Don'ts refer to main windows and dialogs in Web Dynpro applications. (The **Main Window** is generally the window of the web browser.)



**Figure 2.47: Dialog in Main Window**

### Dos for Main Windows and Dialogs

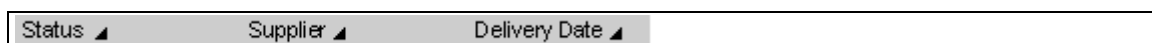
- Choose a meaningful title for the application. The Web Dynpro framework displays the application title in the title bar of the web browser window. Screen readers can read the application title (for example, when you switch to the application by choosing [Alt] + [Tab] on Microsoft Windows).

### Don'ts for Main Windows and Dialogs

- Keep the number of window elements you use to a minimum. Do not use web browser windows that require the user to consult a background window to complete or understand the window content.
- You can include images in the title bar of a dialog. Do not convey important information solely in the image of a dialog. Always place this information in the dialog title as well.

## 2.14.2 MenuBar

The **MenuBar** UI element (as shown in Figure 2.48) is used to display actions in a menu. The MenuBar element is a toolbar that can be organized into different blocks called the Menu elements. Under each block, you can create individual menu items or other Menu elements.



**Figure 2.48: MenuBar Element**

### Dos for MenuBar Elements

- Use the **tooltip** property of the element and the individual top-level menu items to add semantic information.



### 2.14.3 Menu

You can use the **Menu** UI element (as shown in Figure 2.49) and its related elements (**MenuItem**, **MenuRadioButton**, and **MenuCheckBox**) to create a menu for a UI element. This menu can then be used by the MenuItem elements of the MenuBar element.



**Figure 2.49: Menu Element**

#### Dos for Menu Elements

- Create meaningful texts for the menu items. You can add an image to illustrate the function.
- Use the **tooltip** property of the element to add semantic information.

### 2.14.4 ToolBar

The **ToolBar** UI element (as shown in Figure 2.50) is a collection of tools that can be accessed using other UI elements. Toolbars provide an additional method of grouping UI elements by function.



**Figure 2.50: ToolBar Element**

A **ToolBar** element is not a user interface element in its own right. It can only be used in the following elements:

- Group
- TabStrip
- Table
- Tray

#### Dos for ToolBar Elements

- Use the **tooltip** property of the element to add semantic information.
- Use the **accessibilityDescription** property to create a short description (as a replacement for a title). A screen reader then reads this text when the focus moves to the element.
- Information about how to use the following toolbar elements in an accessibility context is included in the corresponding section for each standard element. For example, information about the **ToolBarButton** element is found under the **Button** element.
  - ToolBarButton
  - ToolBarButtonChoice
  - ToolBarDropDownByIndex or ToolBarDropDownByKey
  - ToolBarInputField

- `ToolBarLinkToAction` or `ToolBarLinkToURL`
- `ToolBarToggleButton`

## 2.14.5 RowRepeater

The **RowRepeater** UI element is used to display UI elements in a list. The UI elements are specified in an aggregation (called `ROW_ELEMENTS`). Each row in a `RowRepeater` element looks the same because they all consist of the same UI elements. Only the content of each row differs.

### Note:

Note that in Release **NetWeaver 2004s** this element is available only in Web Dynpro ABAP.

You can use the `RowRepeater` element to display lists of search results, for example. Internally, the Web Dynpro framework handles this element like a table, with the same keyboard navigation and screen reader properties as the `Table` element.

### Dos for RowRepeater Elements

- Use the **tooltip** property of the element to add semantic information.
- Use the **accessibilityDescription** property to create a short description (as a replacement for a title).

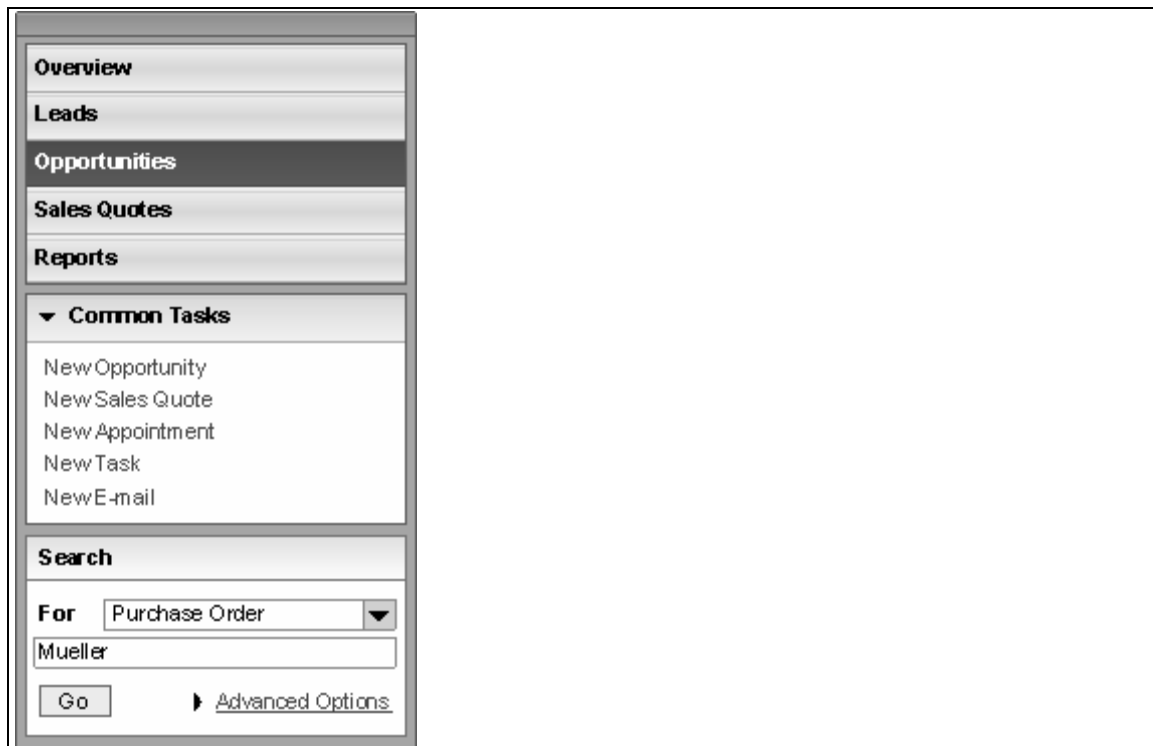
## 2.15 Elements in the Pattern Library

This chapter explains each of the UI elements in the **Pattern** library, and the programming rules. It discusses the following UI elements:

- `ContextualPanel`
- `FreeContextualArea`
- `HorizontalContextualPanel`
- `MessageArea`
- `NavigationList`
- `PageHeader`
- `PatternTabStrip`, `PatternTitle`, `PatternExpandFunction`
- `Shuttle`

### 2.15.1 ContextualPanel

The **ContextualPanel** UI element (as shown in Figure 2.51) can contain the elements **ViewSwitcher**, **NavigationList**, and **FreeContextualArea**. This element was originally developed for launching applications.



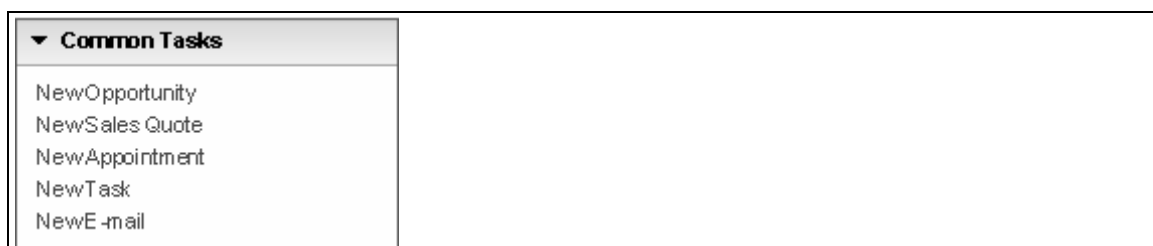
**Figure 2.51: ContextualPanel Element with ViewSwitcher, NavigationList, and FreeContextualArea**

#### Dos for ContextualPanel Elements

- Use the tooltip of the element to add semantic information.

### 2.15.2 NavigationList

The **NavigationList** UI element (as shown in Figure 2.52) displays a navigation area and is used in the ContextualPanel element.



**Figure 2.52: NavigationList Element**

#### Dos for NavigationList Elements

- Use the header text as the title of the element.

#### Don'ts for NavigationList Elements

- Do not place any important information in the tooltip since it will not be read by screen readers in accessibility mode.

### 2.15.3 FreeContextualArea

The **FreeContextualArea** UI element (as shown in Figure 2.53) in the ContextualPanel element is announced as a group in accessibility mode. This element is always a part of the Contextual-Panel element and can contain any Web Dynpro element (the example shows a search function).



**Figure 2.53: FreeContextualArea Element**

#### Dos for FreeContextualArea Elements

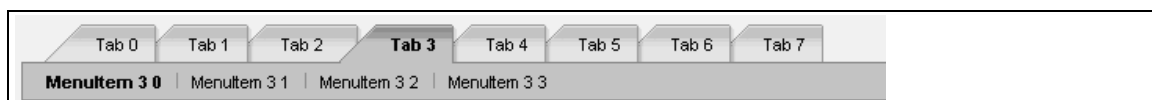
- Use the header text as the group title of the FreeContextualArea element.

### 2.15.4 HorizontalContextualNavigationPanel

The **HorizontalContextualNavigationPanel** UI element (as shown in Figure 2.54) displays navigation options in a similar way to the ContextualPanel element, but with the hierarchy restricted to two levels.

#### Note:

Note that in Release **NetWeaver 2004s** this element is available only in Web Dynpro ABAP.



**Figure 2.54: HorizontalContextualPanel Element**

The selected entry is specified by the **lead selection** property.

#### Dos for HorizontalContextualNavigationPanel Elements

- Use the tooltip of the element to add semantic information.
- Use the tooltip to provide additional information for each tab and subtab. Associate the tooltip with the appropriate dataSource node of the tab or subtab.

### 2.15.5 MessageArea

The **MessageArea** UI element (as shown in Figure 2.55) is a free placeholder on the application interface. This placeholder specifies the position of messages (such as errors or warnings) in the View element. Only one MessageArea element is allowed in each View element.



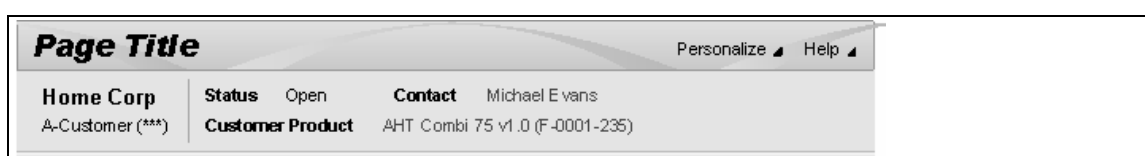
**Figure 2.55: MessageArea Element**

#### Don'ts for MessageArea

- Do not place any important information in the tooltip since it will not be read by screen readers in accessibility mode.

### 2.15.6 PageHeader

You use the **PageHeader** UI element (as shown in Figure 2.56) to display a header on a page. You can display any number of UI elements in the area under the header.



**Figure 2.56: PageHeader Element**

#### Dos for PageHeader Elements

- Use the tooltip for any additional semantic information.

### 2.15.7 PatternTabStrip, PatternTray

The **PatternTabStrip** and **PatternTray** UI elements (not pictured) are used in pattern layouts.

#### Dos for PatternTabStrip and PatternTray Elements

- Use the **accessibilityDescription** property of the elements to create a short description (as a replacement for a title).
- Use the tooltip for additional semantic information about the elements **PatternTabStrip** or **PatternTray**.

### 2.15.8 Elements Not Relevant to Accessibility

The following UI elements are utilities that cannot be given the screen focus. They are used to group other (accessible) elements visually. They are either not relevant in accessibility contexts or do not support accessibility:

- ButtonRow
- HorizontalGutter
- InvisibleElement
- ToolBarSeparator
- ScrollContainer

- ViewContainerElement
- MatrixLayout
- RowLayout
- FlowLayout
- GridLayout

#### **Don'ts for Elements Not Relevant to Accessibility**

- Do not use the tooltip or any other properties of these elements for accessibility purposes. The supported assistive technologies ignore these properties.

## 3 Testing Web Dynpro for ABAP and Web Dynpro for JAVA

### 3.1 Prerequisites for the Test Environment

For information about the prerequisites for the test environment (settings for Web Dynpro applications and the portal, information about screen reader), see the newest version of the document "[Front-End Requirements and Infrastructure for Accessibility](http://www.sapdesignguild.org/resources/acc_technical_requirements_V3_1_external_EN.pdf)".  
[http://www.sapdesignguild.org/resources/acc\\_technical\\_requirements\\_V3\\_1\\_external\\_EN.pdf](http://www.sapdesignguild.org/resources/acc_technical_requirements_V3_1_external_EN.pdf)

### 3.2 Fundamental Concepts

In Web Dynpro, keyboard navigation within and between UI elements is defined entirely by the framework. Application developers do not need to worry about most aspects of keyboard navigation, with the exception of layouts based on groups (as discussed in Chapter 3.4).

The framework is also responsible for the description of elements for screen readers. Application developers only need to check that the properties for which they are responsible convey the elements and combinations of elements as intended in the given context.

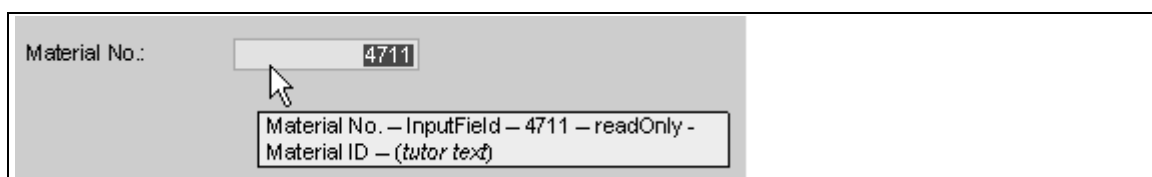
The general reading order for screen readers when focused on the **InputField** UI element and associated label is implemented as follows:

```
<labeltext> <type> <value> <state> <tooltip> <tutor>
```

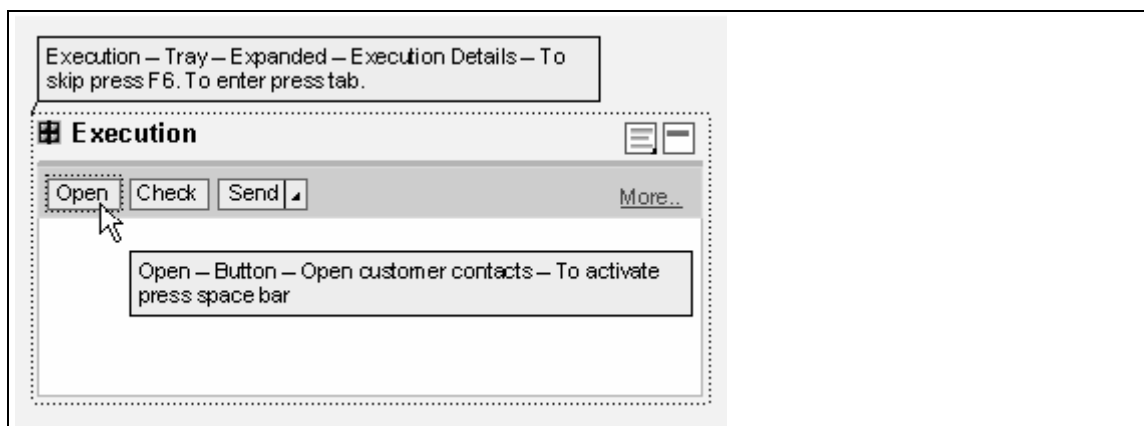
where

- *labeltext* is the text of the label
- *type* is the type of the element (an input field in this case)
- *value* is the value of the field
- *state* is the state of the current field (such as *not available*, *modifiable*, *read-only*, *inactive*)
- *tooltip* is the tooltip of the field
- *tutor* is additional information about how to use the element.

Figure 3.1 shows the texts provided for a screen reader by an InputField element. Figure 3.2 shows the texts for a Tray element.



**Figure 3.1: Properties of an InputField Element for Screen Reader Output**



**Figure 3.2: Properties of a Tray Element for Screen Reader Output**

The Web Dynpro framework always provides the screen reader with the properties *type*, *value*, *state*, and *tutor*. Application developers are only responsible for the properties **text**, **tooltip**, and **accessibilityDescription**. The texts they create here are added to the full description of the element by the framework, where the screen reader can access them.

There are currently two ways of testing these properties:

- Automated tests in the development environment (design time)
- Manual tests at runtime

The next two chapters describe these two options.

### 3.3 Automated Tests at Design Time

The following standard automatic accessibility checks run at design time in the development environments for Web Dynpro Java (*SAP Developer Studio*) and Web Dynpro ABAP (*Object Navigator*, transaction SE80). These checks are implemented as part of the syntax checks. Table 6.2 gives you an overview of these checks. We recommend that you keep these checks enabled during your development work. Any properties that you need to define yourself are indicated as *Required* in the development environments and listed in Table 6.3.

Note the following information about automated tests at design time:

- In *Object Navigator*, the properties of each component include the *Accessibility Checks Active* option. (Double-click a component to display its properties.) If you deactivate this option, the development environment does not carry out any automatic design time accessibility checks for the given component and its views.
- These automatic checks are not possible if the elements are added dynamically at runtime (dynamic UI element programming). In this case, you must run manual accessibility tests.
- Automatic checks only test whether the **text** property contains text or, if it does not, whether the associated properties have been defined. No semantic checks are made. The automatic checks cannot tell whether the texts contain complete and correct information.
- The automatic accessibility checks do not run for **test** applications in Web Dynpro ABAP. These include temporary objects in the package \$TMP and applications in the package SWDP\_TEST.



Elements	Type of Check
Button, ButtonChoice, LinkToAction, LinkToURL, ToggleButton, ToggleLink, ToolBarButton, ToolBarButtonChoice, ToolBarLinkToAction, ToolBarLinkToURL, ToolBarToggleButton, BreadCrumbStep, MultipleBreadCrumbStep, LegendItem, MultipleLegendItem, TextView, TreeNodeType, TreeItem, MenuItem	If the <b>text</b> property was not set, the system checks for the <b>tooltip</b> property.  An error message is displayed if, for example, a Button element or a Link element with an icon has neither the <b>text</b> property nor the <b>tooltip</b> property.
CheckBox, RadioButton, FileDownload, TriStateCheckBox	If the properties <b>text</b> and <b>label</b> were not set, the system checks for the <b>tooltip</b> property.  An error message is displayed if, for example, a CheckBox element has neither the <b>text</b> property nor the <b>label</b> property nor the <b>tooltip</b> property.
Group, Tray	If the <b>caption</b> property has not been set, the system checks for the <b>accessibilityDescription</b> property.  An error message is displayed if, for example, a Group element has neither the <b>caption</b> property nor the <b>accessibilityDescription</b> property.
Image	If the <b>label</b> and <b>isDecorative</b> properties have not been set, the system checks for the <b>tooltip</b> property.  An error message is displayed if the <b>tooltip</b> property is empty for an Image element that can be navigated to.
InputField, DropDownByIndex, DropDownByKey, ItemListBox, TextEdit, ToolBarDropDownByIndex, ToolBarDropDownByKey, ToolBarInputField, FileUpload	The system checks whether the <b>label</b> property has been set.  If the <b>label</b> property has not been set and no description has been entered for the associated Context element in ABAP/Java Data Dictionary (DDIC), the system checks for the <b>tooltip</b> property.  An error message is displayed if, for example, a TextEdit or DropDownByKey element does not have an associated <b>label</b> property and there is no description entered in ABAP/Java Data Dictionary (DDIC).
Table	If the <b>caption</b> property has not been set, the system checks for the <b>accessibilityDescription</b> property.  Furthermore, the system checks whether the aggregation header is set for columns and whether the aggregation header is visible.  The <b>tooltip</b> property is not checked.  An error message is displayed if a table has neither a <b>caption</b> property nor an <b>accessibilityDescription</b> property, or if a column does not have a (visible) column header.

Elements	Type of Check
CheckBoxGroup, RadioButtonGroupByIndex, RadioButtonGroupByKey, RoadMap, DateNavigator, PhaseIndicator, TabStrip, ToolBar, GeoMap	If the <b>tooltip</b> property has not been set, the system checks for the <b>accessibilityDescription</b> property. An error message is displayed if, for example, a CheckBoxGroup element has neither the <b>tooltip</b> property nor the <b>accessibilityDescription</b> property.
TransparentContainer	If the <b>layoutContainer</b> property has not been set, the system checks for the <b>accessibilityDescription</b> property. An error message is displayed if a TransparentContainer element has the property <b>layoutContainer=false</b> and does not have an <b>accessibilityDescription</b> property.
Legend	The system checks whether the <b>legendId</b> property of the Legend element (Calendar, DateNavigator, or Table) has been set.
ProgressIndicator, ValueComparison, BusinessGraphics, GeoMap, Iframe	The system checks whether the <b>tooltip</b> property has been set.
MenuItem	If the <b>imageSource</b> property has not been set, the system checks for the <b>text</b> property.
Phase, <a href="#">MultiplePhase</a> , RoadMapStep	If the <b>description</b> property has not been set, the system checks for the <b>tooltip</b> property.
TabStripTab	The system checks whether the <b>caption.text</b> property has been set. If not, the system checks whether <b>caption.tooltip</b> property has been set.

Table 3.1: Automated UI Element Tests at Design Time

Required (mandatory) properties are indicated as *Required* in the development environments. These properties are listed in Table 6.3 below.

Elements	Required Property
BreadCrumbStep, MultipleBreadCrumbStep, LegendItem, MultipleLegendItem, TextView, SectionHeader, NavigationListItem, Label, CheckBox, RadioButton, MenuCheckBox, MenuRadioButton, FormattedTextView	text
TabStripTab	caption.text
HorizontalContextualPanel, NavigationList	itemText
MultiplePhase	description
Menu	title
ProgressIndicator, ValueComparison, BusinessGraphic, GeoMap, Iframe	tooltip

Table 3.2: Required Properties of UI Elements

## 3.4 Manual Tests at Runtime

The following describes special manual tests for the Web Dynpro environment:

- Elements added statically and, in particular, dynamically at runtime
  - Check that the specified element properties **text**, **tooltip**, and **accessibilityDescription** are conveyed correctly in the context of the element. You do this by moving the focus to the element with the tab key and checking the screen reader output.
- Elements associated with labels
  - Check that the screen reader announces the correct label text when the focus moves to the elements associated with labels (such as input fields).
- "Empty" elements
  - If an element can be empty at runtime (if it has no text), check its text output in the screen reader. This is particularly important for the elements **TextView** and **Caption** because missing information can be confusing in some contexts. If in doubt, always specify a placeholder text or create a text in the element's tooltip (such as "no data").
- Titles for complex elements
  - Check that visible titles have been created for complex elements or elements that group other elements (such as Table elements or TabStrip elements) or check that a title text has been entered in the **accessibilityDescription** property.
  - Titles are optional in some cases. If you use only one complex element in your application, such as a table, a title is optional, since the window title is also the table's title in this case. However, the window title must be meaningful.
- Element positioning
  - If elements with the same function (for navigation or in the menu) are used on more than one screen or page in the application, check that these elements are always positioned in the same place.
- Element grouping
  - Check that the tab chain is correct. If the reading direction is from left to right and from top to bottom, the tab chain within the group must also operate in this direction. The tab chain must not jump to the next group until it has reached the end of the current group. If nested element groups are used, check that the screen reader reads out the correct group headers (if visible) when the focus moves to the group box. If no visible group header exists, the screen reader must read out the **accessibilityDescription** property of the group.
- Tables
  - Check that tables have visible titles. If they do not, check that the **accessibilityDescription** property has been set. Also check that the table columns have visible header rows. Navigate to the columns. If a column header uses an abbreviation, check that the full text is in the tooltip.
- Embedded objects/external elements
  - Check the accessibility of embedded objects and, if necessary, whether an accessible alternative is available.
- Unexpected UI changes

- Check whether any UI changes, such as new control elements, take place above the keyboard focus. Any changes of this type must take place after the keyboard focus, in accordance with the reading direction.
- Periodic UI changes
  - If an application refreshes its content or appearance periodically and automatically (such as a ticker with share prices), check that the updates can be deactivated and enabled manually. Alternatively, check that the user can specify the auto-refresh interval.
- Timed responses
  - If an application requires the user to perform an action within a specified period of time, check that the requirement can be deactivated and enabled manually. Alternatively, check that the user can specify the response time.
- Abbreviations
  - If abbreviations have been used, check that the texts are written out in full in a suitable location (such as in the tooltip of the element).
- Terminology
  - Check that the elements use correct and appropriate terminology.

## 4 How to Integrate Third Party Products

### 4.1 Embedded Components

Embedded components always run as a part of a screen inside an SAP application window. Embedded components can be, for example, ActiveX Controls (OLE Controls).

The component must therefore conform to the following standards:

- Tab In  
The focus moves from the SAP application window to the component and the user can navigate through the objects/items inside the component.
- Tab Out  
The component loses the keyboard focus and should save the last focus position for when the user goes back to the component later. (This is highly recommended to avoid extra navigation overhead). The focus must return to the application window.
- Content must be navigable (for example, by tab keystroke or arrow keys).
- Focus must be clearly visualized.

Furthermore, the component must comply with the standards of US Section 508 and the standards of WCAG 1.0 (Priority 1 and 2).

### 4.2 Launching External Programs

Launched programs usually appear in their own application window, on top of the started application window. The keyboard focus should automatically move to the new program window.

After the external program is closed, the focus should return to the application window that started the external program.

Furthermore, the component must comply with the standards of US Section 508 and the standards of WCAG 1.0 (Priority 1 and 2).

## 5 Useful Links

The following list contains useful links.

SAP Design Guild Edition Accessibility

<http://www.sapdesignguild.org/editions/edition9/acc.asp>

Accessibility glossary of the SAP Design Guild

[http://www1.sapdesignguild.org/editions/edition9/acc\\_glossary.asp](http://www1.sapdesignguild.org/editions/edition9/acc_glossary.asp)

For information about the prerequisites, settings for Web Dynpro applications and the portal, information about screen reader, see the newest version of the document "[Front-End Requirements and Infrastructure for Accessibility](#)".

[http://www.sapdesignguild.org/resources/acc\\_technical\\_requirements\\_V3\\_1\\_external\\_EN.pdf](http://www.sapdesignguild.org/resources/acc_technical_requirements_V3_1_external_EN.pdf)

SAP Note [995251](#): "JAWS for Windows Settings for SAP Software".

[http://service.sap.com/~form/handler?\\_APP=01100107900000000342&\\_EVENT=REDIR&\\_NNUM=995251](http://service.sap.com/~form/handler?_APP=01100107900000000342&_EVENT=REDIR&_NNUM=995251)

Standards of WCAG 1.0

<http://www.w3.org/TR/WAI-WEBCONTENT/>

Standards of US Section 508

<http://www.section508.gov/index.cfm?FuseAction=Content&ID=12>