

SAP User Experience –  
Accessibility  
SAP INFORMATION



# Accessibility Guidelines for SAP Interactive Forms by Adobe

**Version 1.0**

December 6, 2007

Note: These guidelines are part of the book “Developing Accessible Software with SAP Net-Weaver” (ISBN: 978-1-59229-112-0; ISBN of the German edition: 978-3-89842-862-0).



## 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>Disclaimer</b> .....	<b>4</b>
<b>2</b>	<b>General</b> .....	<b>5</b>
2.1	Adobe Components .....	5
2.2	Prerequisites .....	7
2.3	Print Forms or Interactive Forms .....	7
2.4	The Development Environment for PDF Forms .....	7
2.5	Tagged PDF Forms .....	8
2.6	General Accessibility Rules .....	8
2.7	<i>Library</i> Palette .....	9
2.8	<i>Accessibility</i> Palette .....	9
2.9	<i>Layout</i> Palette .....	11
2.10	Order of the Elements .....	12
2.11	Keyboards and Focus .....	15
2.12	Using Colors .....	15
<b>3</b>	<b>UI Elements and ABAP Language Elements</b> .....	<b>17</b>
3.1	Static Text .....	17
3.2	Text Field/Numeric Field/Date and Time Field .....	17
3.3	Floating Field .....	18
3.4	Static Image .....	18
3.5	Image Field .....	19
3.6	Barcode .....	20
3.7	Check Box .....	20
3.8	Radio Button .....	21
3.9	Drop-Down List .....	22
3.10	List Box .....	23
3.11	Button .....	23
3.12	Table .....	24
3.13	Circle, Line, and Rectangle Objects .....	25
<b>4</b>	<b>Useful Links</b> .....	<b>26</b>

# 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 checklist 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

## 2 General

### 2.1 Adobe Components

SAP and Adobe Systems Inc. have worked together to develop a solution for the creation of forms in SAP systems, known as *SAP Interactive Forms by Adobe*. This solution is based on the PDF format. PDF stands for *Portable Document Format* and was developed by Adobe in 1993. PDF is a standard format for forms in the internet. PDF forms can be implemented both for the simple output of documents for printing and for interactive business processes (PDFs in which you can enter data).

The Accessibility Guidelines for SAP Interactive Forms tell you how to create accessible PDF forms. A PDF form is any document in PDF format that can be opened by users in *Adobe Acrobat* and *Adobe Reader*. Some forms can also be edited. You can download *Adobe Reader* for free from [www.adobe.com](http://www.adobe.com). *Adobe Acrobat* is not available for free.

As part of a joint project, SAP and Adobe have integrated some technical components of Adobe into *SAP NetWeaver*, SAP's integration and application platform. As a result of this integration, all applications that run on *SAP NetWeaver* now include an environment for creating form-based business processes.

Adobe contributes two components to the joint form solution:

- **Adobe LiveCycle Designer**

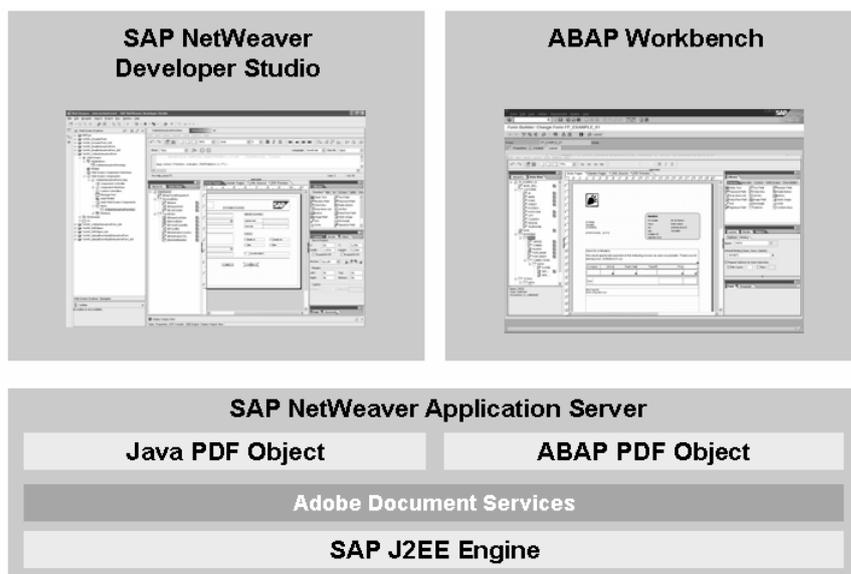
Adobe LiveCycle Designer is a user-friendly tool for designing form templates for use in development projects in the SAP system. Adobe LiveCycle Designer tool is integrated into both ABAP Workbench and SAP NetWeaver Developer Studio, as depicted in Figure 2.1. The version of *Adobe LiveCycle Designer* delivered by SAP is largely comparable to the version marketed by Adobe for its own server products.

- **Adobe Document Services (ADS)**

*Adobe Document Services* is the runtime component of the solution and performs two tasks:

- It generates the output format in the SAP system. The output format can be either the PDF form or the printer languages for a print form.
- In interactive scenarios, it extracts the user's input from the PDF form.

ADS is a server component that runs as a web service on SAP NetWeaver Application Server Java. It was developed by Adobe especially for SAP.



**Figure 2.1: Development Environment for SAP Interactive forms by Adobe**

*SAP Interactive Forms by Adobe* was shipped for the first time in SAP NetWeaver 2004.

All descriptions in this document refer to the creation of forms in *Adobe LiveCycle Designer* in *ABAP™ Workbench*. All our recommendations are based on using the *Form Builder* transaction, which is called using the transaction code **SFP**. A basic level of experience with *Adobe LiveCycle Designer* is required to understand this document, but if necessary you can refer to the *Adobe LiveCycle Designer* help documentation. Information about accessibility is available in the Adobe help documentation under *Accessibility* and is also referred to by this book.

The following topics are not in the scope of these guidelines:

- The system landscape in SAP NetWeaver and ADS, and technical details of the integration of Adobe form technologies into SAP NetWeaver (if not relevant to accessible forms)
- The creation of PDF forms with other tools, such as Adobe Acrobat Professional 7.0.
- The conversion of Microsoft Office files (such as Word or PowerPoint files) or other document formats to PDF formats
- General Adobe Reader functions, unless they are of special relevance
- The migration of Smart Forms to SAP Interactive Forms by Adobe

**Note:**

Note that the naming of development objects in this document is based on the terminology used by *Adobe LiveCycle Designer*, and can differ from SAP's terminology. The following table lists some of the differences:

Adobe LiveCycle Designer	SAP Terminology
Caption	Label
Check box	Checkbox
Button	Pushbutton

*Table 2.1: Differences in Terminology in Adobe LiveCycle Designer and SAP*

## 2.2 Prerequisites

For information about the prerequisites see the newest version of the document *Front-End Requirements and Infrastructure for Accessibility*, see [SAP Design Guild Edition Accessibility](#) under “Links & More” → “Related Topics” → “Frontend Requirements and Infrastructure for Accessibility”.

## 2.3 Print Forms or Interactive Forms

You can use *Adobe LiveCycle Designer* to create both interactive forms and print forms:

- **Interactive forms**

The fields in the form are ready for input and users can complete them directly on the computer.

- **Print forms**

The fields in the form are static and users must print the form and complete it by hand.

The descriptions in this document apply to both types of forms, if not otherwise stated.

## 2.4 The Development Environment for PDF Forms

You require the following development environment to create your own PDF forms in your SAP system:

- *Adobe LiveCycle Designer* to create templates for PDF forms
- *Adobe Document Services* to generate PDF forms
- *Adobe Reader* or *Adobe Acrobat* to display PDF forms

### Integration with Web Dynpro

When users use *Adobe Reader* to enter data in a PDF form in an online scenario, you must make sure that this data is passed to the Web Dynpro environment. On the user's computer, *SAP Interactive Forms by Adobe* requires *Adobe Reader* and Web Dynpro to communicate in one of the following two ways:

- **Active Components Framework (ACF)**

ACF integrates active components, such as ActiveX or Java Applets, into Web Dynpro. It also includes a special component for passing data between Web Dynpro and *Adobe Reader*. This integration is based on an ActiveX control, which restricts the use of the Web Dynpro application in question to the platform *Microsoft Internet Explorer* on *Microsoft Windows*.

- **Zero Client Installation (ZCI)**

From *SAP NetWeaver 7.0 SP 08*, the HTTP-based interface *Zero Client Installation* is available. Thus, no additional installation is required when you work with Interactive Forms based on the ZCI technology. It is supported by *Adobe Reader* Version 7.0.8 and higher. ZCI is used for communication between Web Dynpro and Adobe Reader.

**Note:**

The recommendations and examples in this guidelines are based on the following development environment:

- *Adobe LiveCycle Designer 7.1*
- *Adobe Document Services* as included in NetWeaver 2004s
- *Adobe Reader 7.0.8.*

Note that other versions may differ in certain respects.

## 2.5 Tagged PDF Forms

Tagged PDF forms are mandatory for accessibility. In a tagged form, each element contains all information about its own structure, plus information about how it is related to or dependent on other elements. Only in tagged PDFs can screen readers identify and describe the content of a document correctly.

The objects in a form can include tables, radio buttons, check boxes, input fields, or images. As well as the objects themselves, screen readers also announce additional information about the individual elements, such as the number of columns and rows in a table or the caption and status of a check box. The order in which the elements are read by the screen reader is also significant. If the order is not followed correctly, the information may be misleading.

The way that tagged forms are created depends on the development environment:

- **Web Dynpro for Java**

In this development environment, tagged PDF forms are created automatically. You do not need to make any further settings.

- **ABAP Dynpro in SAP GUI for Windows**

Before you can generate a tagged PDF form, you must enable the accessibility mode for *SAP GUI for Windows* in the Control Panel of your Windows operating system.

Another way of generating a PDF form with tags is to set the parameter SFOUTPUTPARAMS-PDFTAGGED of the function module **FP\_JOB\_OPEN** to "X" in the print program.

```
data ls_sfoutputparams type sfoutputparams.

ls_sfoutputparams-pdf tagged = 'X'
ls_sfoutputparams-(...) = (...) "other parameters

CALL FUNCTION 'FP_JOB_OPEN'
  CHANGING
    IE_OUTPUTPARAMS = ls_sfoutputparams
  EXCEPTIONS
    CANCEL           = 1
    USAGE_ERROR     = 2
    SYSTEM_ERROR    = 3
    INTERNAL_ERROR  = 4
    OTHERS          = 5
```

## 2.6 General Accessibility Rules

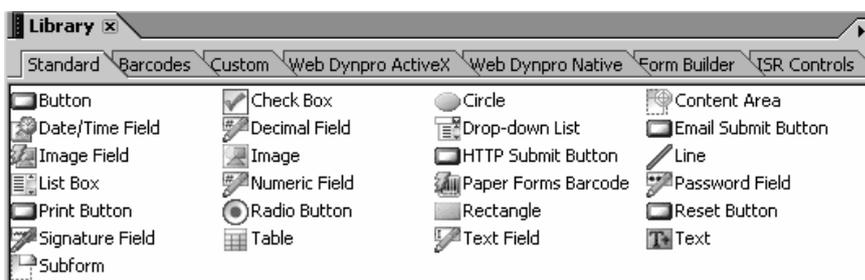
We recommend that you observe the following rules when creating SAP forms:

- Use *SAP Interactive Forms by Adobe* to create forms and not Smart Forms or SAPscript.
- When you migrate your forms to *SAP Interactive Forms by Adobe*, revise them to comply with accessibility guidelines.

- Create forms that are simple and easy to use.
- Always create *tagged* PDF forms (as discussed in Chapter 2.5).
- Check whether the elements in the form are arranged in a sensible and consistent order.
- Do not include any important information in data that is scanned in. If you cannot avoid this, create a meaningful text description for this data. Screen readers cannot detect any scanned data that does not have a text description.
- Create meaningful captions for the fields in your forms. If you cannot include enough information, create additional texts in the *Accessibility* palette.
- When creating interactive forms, create forms that are fully accessible from the keyboard.
- Do not use optical effects inserted by scripts, such as flashing text, in interactive forms. This reduces the legibility of the forms for certain categories of users.
- When you import forms, check the accessibility of the entire imported document, and make any revisions that are required.

## 2.7 Library Palette

When you design your forms, always use development objects from the *Standard* tab page of the *Library* palette. You can display this palette from the *Adobe LiveCycle Designer* menu by choosing *Palettes* → *Library*. Choose the *Standard* tab page and only use the objects listed there (as shown in Figure 2.2). If you use other objects, they may be ignored by screen readers. A full list of the objects is available in the *Adobe LiveCycle Designer* help documentation under *Working with Objects* → *Choosing Objects*.

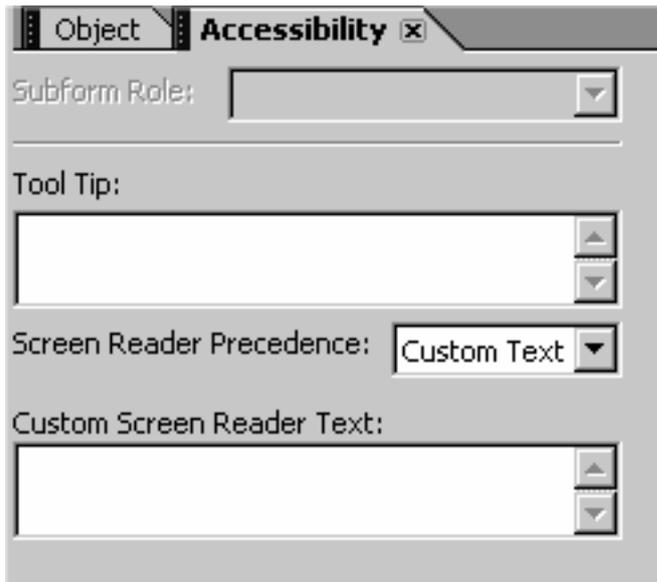


**Figure 2.2: Library Palette – Standard Tab Page**

Use only the standard objects depicted and do not use any objects you have created yourself. Instead of using the *Check Box* object, for example, a form developer might draw a box around the letter *x* to show that the check box is *On*. This may look like a check box, but cannot be recognized as such by screen readers, which would probably read just the letter *x* from this position. If you use the *Check Box* object instead, the screen reader can detect it and announce whether it is *On* or *Off*.

## 2.8 Accessibility Palette

One of the most useful palettes is the *Accessibility* palette, as shown in Figure 2.3. You can display this palette from the *Adobe LiveCycle Designer* menu by choosing *Palette* → *Accessibility*. Screen readers can detect and announce any texts entered in this palette.



**Figure 2.3: Accessibility Palette**

You have the following options in the *Accessibility* palette:

- **Tool tip**

Any text you enter under *Tool Tip* is read by screen readers. In print forms, the alternative of tooltip is the *Custom Screen Reader Text*. Interactive forms display the tool tip when you move the cursor over the associated object.

- **Screen Reader Precedence**

The *Screen Reader Precedence* (as shown in Figure 2.4) defines which text a screen reader will try to read.

- **Custom Text**

This is the text entered under *Custom Screen Reader Text*.

- **Tool Tip**

This is the text entered under *Tool Tip*.

- **Caption**

This is the caption text of an object. If no text exists under *Tool Tip* or *Custom Screen Reader Text*, the screen reader reads this text by default.

- **Name**

In this setting, the screen reader reads the technical name of the associated object, as displayed in the *Hierarchy* palette.

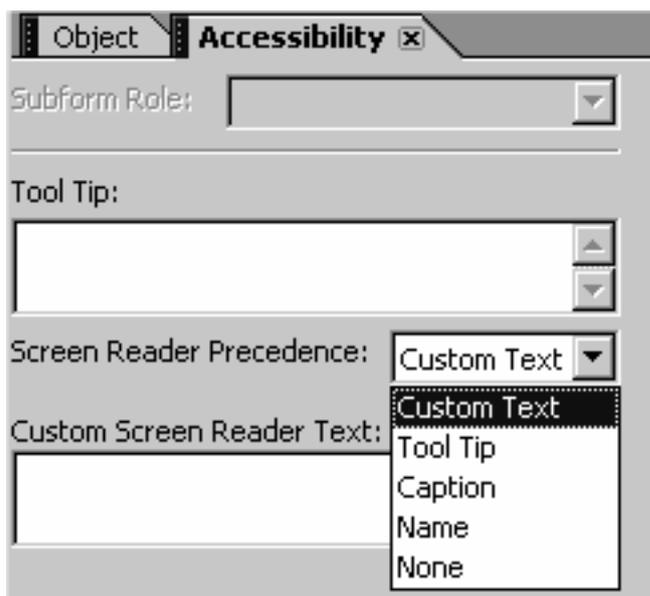
The technical names are also shown in the *Name* field on the *Binding* tab page in the *Object* palette.

- **None**

In this setting, the screen reader does not read any text for the field. Any *Tool Tips* or *Custom Screen Reader Texts* you have defined are deactivated.

- **Custom screen reader text**

This text is read by screen readers and is used as an alternative to *Tool Tip* in print forms. If a text is entered under *Custom Screen Reader Text* for a particular object, the screen reader reads this text instead of the tool tip, technical name, or associated caption.



**Figure 2.4: Accessibility Palette – Screen Reader Precedence**

#### **Dos for the *Accessibility* Palette**

- When you are designing accessible fields in forms, we recommend that you create a meaningful visible caption without any additional text descriptions. Always check whether the caption text is descriptive enough before resorting to alternative texts in the *Accessibility* palette.
- Create either a *Tool Tip* or a *Custom Screen Reader Text* for your object, but not both.
- When you create a *Tool Tip* or a *Custom Screen Reader Text*, always include the caption that is visible on the form.
- If you have to create a *Tool Tip* or a *Custom Screen Reader Text*, always include the caption that is visible on the form, except when the visible caption is not meaningful, for example when the caption itself is abbreviated. This helps screen reader users communicate effectively with other users about UI elements. These different groups of users have difficulty identifying the same UI element if its caption text differs from the *Tool Tip* or *Custom Screen Reader Text*.
- Do not use the *Accessibility* palette to create descriptions for any invisible fields or areas.

## **2.9 Layout Palette**

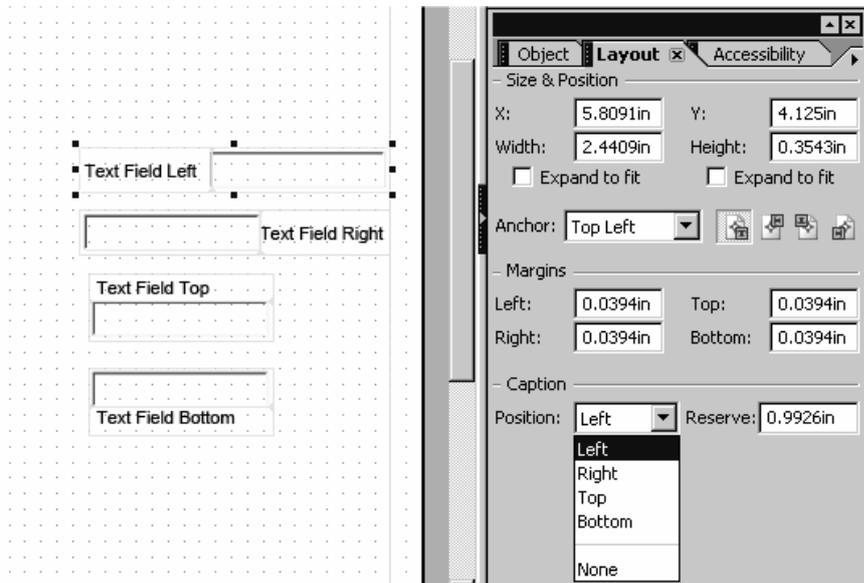
You can use the *Layout* palette to define the size, position, and margins of an object and its caption. You can display this palette in the *Adobe LiveCycle Designer* menu under *Palettes* → *Layout*.

The position of the caption is independent of the order in which screen readers read the object and its elements.

When you create an object, *Adobe LiveCycle Designer* automatically positions the caption as specified by the object type. The captions of radio buttons, for example, are placed on the right. If you want to change the position of the caption text, proceed as follows:

1. Select the object by moving the focus to it.
2. In the *Layout* palette, you can select the position of your object from the *Position* drop-down list.

As shown in Figure 2.5, you can position the object *Right*, *Left*, *Top*, or *Bottom*. You can also choose no caption by selecting *None*. You can find our recommendations for the positioning of captions under each object type.



**Figure 2.5: Layout Palette – Position of the Caption**

#### Dos for the *Layout* Palette

- Create meaningful captions for the fields in your forms.
- Observe our recommendations when you position the caption texts for your objects.

#### Don'ts for the *Layout* Palette

- Do not hide the captions on the form, for example by selecting the position *None* in the *Layout* palette.

## 2.10 Order of the Elements

The layout of a PDF form is determined by the order of its elements. Elements arranged in a random order can provide misleading information, especially when the form is read by a screen reader.

#### Tabbing order

Two concepts are important when you navigate through a form: the *tabbing order* (or tab chain) and the *reading order*. The tabbing order is only of relevance in interactive forms and not in print forms. This is because in interactive forms users can use the tab key to navigate from one interactive field to the next. Interactive forms also contain static objects that are ignored by the tab key (this is the case for all objects in a print form). All elements in a tagged form contain information about the order in which they are read by screen readers. This is known as the *reading order*. You specify the reading order in the tags when you create a PDF form. The logical structure of the form includes all static objects and field objects. In most cases, we mean the same thing when we discuss the tabbing order and the reading order. (Further information is available in the *Adobe LiveCycle Designer* help documentation.)

## Reading order

Initially, the reading order in a form is determined from the local position of the objects. You can find out the order of the objects using their coordinates. You can also change the order of the objects. The reading order begins at the object with the lowest vertical coordinate and finishes at the object with the highest vertical coordinate. The objects can be located on the text page or the master page. For technical reasons, you cannot specify that the reading order jump between the master page and body page.

You can display the order of your form objects on the screen. To do this, go to the *Adobe LiveCycle Designer* menu and choose *View → Tab Order*. *Adobe LiveCycle Designer* assigns a number to each object in your form automatically. This numbering then determines the order in which a screen reader reads the objects. You can also use this menu item to hide the tabbing order. Note that you change the tabbing order as soon as you click one of the displayed numbers.

## Default order

The default order on each form page is defined as follows:

- From left to right and from top to bottom (local order), starting from the top left corner of the form
- Any subforms you create are treated as self-contained units and are also navigated from left to right and from top to bottom. If two subforms are positioned next to each other, both of which contain objects, the reading order navigates through all objects in the first subform before moving to the next subform.

## Creating a subform

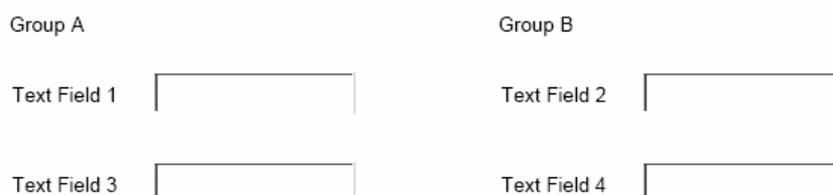
When you create a subform by combining sections of a form, you change the default order. You can create a subform in the *Adobe LiveCycle Designer* menu by choosing *Insert → Standard → Subform*. You can also select your objects in the *Hierarchy* palette and group them in a subform by choosing *Insert → Wrap in Subform*.

The following example explains how subforms work. Figure 2.6 shows a PDF form with six objects:

- Two static texts, *Group A* and *Group B*
- Four text fields, *Text Field 1*, *Text Field 2*, *Text Field 3*, *Text Field 4*

These six objects were positioned in the form as follows:

- Row one consists of *Group A* (left), *Group B* (right).
- Row two consists of *Text Field 1* (left), *Text Field 2* (right).
- Row three consists of *Text Field 3* (left), *Text Field 4* (right).



**Figure 2.6: Example of a Reading Order**

A screen reader would read these fields in the default reading order, from left to right, starting at the top left:

- Group A
- Group B
- Text field 1
- Text field 2
- Text field 3
- Text field 4

You now make a change to the form, grouping the six objects into two subforms:

- Subform A
- Subform B

Subforms A and B contain the following objects:

Subform A	Subform B
Group A	Group B
Text Field 1	Text Field 2
Text Field 3	Text Field 4

*Table 2.2: Grouping Objects in Subforms*

The new subforms have changed the reading order of the screen reader. It first reads all objects in subform A and then all objects in subform B:

- Group A
- Text field 1
- Text field 3
- Group B
- Text field 2
- Text field 4

### Dos for the Order of the Elements

- Make sure that all elements in a form are accessible by screen readers, except when the objects are used only for design purposes. Note that circle, line, and rectangle objects in *Adobe LiveCycle Designer* are not included in the reading order. They are purely graphical objects and have no informational content.
- In interactive forms, make sure that all interactive elements can be accessed using the tab key.
- Group logically related information together by combining objects in subforms.
- Take care when you define the order of the elements. Keep the form layout in mind: left to right, and top to bottom.

### Don'ts for the Order of the Elements

- Do not use the tabbing order display to define the order explicitly. Only use it to check the order (as shown in the *Adobe LiveCycle Designer* menu under *View → Tab Order*).

## 2.11 Keyboards and Focus

Interactive forms are ready for input and users can complete them directly on the computer. The user can use the mouse and the keyboard to move the focus to the form fields and complete them. Many users prefer to use the tab key to move between fields, rather than the mouse. Some groups of users, such as blind users or users with certain motor impairments, have to use the keyboard and cannot use the mouse. Forms are accessible to all groups of users only if users can enter data regardless of the input device they use. In addition to this, keyboard input offers certain universal benefits to all groups of users.

A focus rectangle must indicate visually to the user which interactive form field currently has the focus.

### Dos for Keyboards and Focus

- Create interactive forms that are fully accessible from the keyboard. You must be able to navigate to all interactive elements using the tab key.

### Don'ts for Keyboards and Focus

- Do not use device-specific events to trigger scripting functions in interactive forms. The mouse event *MouseEnter*, for example, cannot be executed using the keyboard.
- In interactive forms, do not use or create client-side scripts that can cause conflicts between the keyboard and screen reader. For example, *Change* events used in drop-down lists or list boxes can trigger unexpected actions.

## 2.12 Using Colors

Colors can optimize the appearance of your forms by highlighting certain parts of them. However, you must observe certain additional accessibility rules. Any information that is conveyed solely in color (colors with semantic meaning) is not accessible to blind users. The same applies to users with color vision deficiencies, or users who use different color schemes, such as a high contrast color screen with white text or foreground on a black background. You must also bear in mind that screen readers cannot detect color information automatically.

The following example illustrates this:

In your interactive form, you want to use captions in two different colors, green and blue, to indicate the difference between mandatory input fields and optional fields. You specify this color highlighting in the *Font* palette. In this example, the color is the only signifier of the difference between the two types of input fields, which makes it impossible for blind users or users with certain types of color blindness to tell them apart.

### Dos for Using Colors

- Make sure that no information is lost if the semantic color is not visible.
- If possible, use default colors for the text and background.
- If you cannot use default colors, make sure that your colors are high contrast, such as black on a light (white) background. Partially sighted users generally require a high contrast between the text and its background to be able to read it.

- Test the legibility of your forms by switching your screen to a high contrast display, both in Windows and in Adobe Reader.

### **Don'ts for Using Colors**

- Do not convey information solely in color. For example, do not use only color to highlight important pieces of text. Use other highlighting methods and text descriptions as well.
- Do not use too many colors, since this can make the actual information in the content difficult to read. Always keep the legibility of the information as your top priority when you decide which colors to use.

## 3 UI Elements and ABAP Language Elements

### 3.1 Static Text

You define static texts in the form template. Static texts cannot be edited by the user in the displayed PDF form. A typical example of a static text is the title of a section of the form.



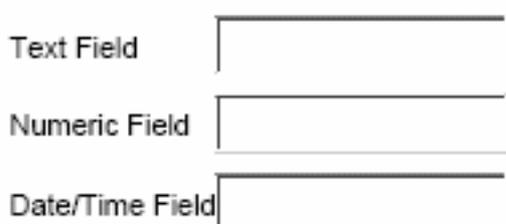
**Figure 3.1: Static Text – Highlighting**

#### Don'ts for Static Texts

- Avoid using abbreviations in static texts.
- Do not highlight texts by including spacing between individual letters. Screen readers cannot read words highlighted in this way. Instead, they read each letter out individually.
- In Figure 3.1, the static text *Hiring* is displayed with spacing on the left, and without spacing on the right.

### 3.2 Text Field/Numeric Field/Date and Time Field

Text fields, numeric fields, and date and time fields are all types of field objects (as shown in Figure 3.2). You can use these fields either as input fields or output fields.



**Figure 3.2: Text Field, Numeric Field, Date and Time Field**

#### Dos for Text Fields/Numeric Fields/Date and Time Fields

- Create meaningful caption texts for single field objects such as text fields, numeric fields, and date and time fields.
- Position the caption on the left of the object. The caption is positioned on the left of the field object automatically when you create it. If it is not for some reason, you can change its position on the *Layout* palette (as shown in Chapter 2.9).
- For field objects in table cells, set the *Screen Reader Precedence* to *None* in the *Accessibility* palette. Otherwise, the screen reader will read the technical name of the object.

- If the caption of a field object contains abbreviations, you can create the texts in full in the *Accessibility* palette; however, remember that the screen reader will no longer read the visible caption text. Include this text in the *Accessibility* palette.

### Don'ts for Text Fields/Numeric Fields/Date and Time Fields

- Do not create captions for field objects inserted in table cells. The column header of the table is used as the caption for these objects.
- If a field object does not contain an abbreviation, do not create an additional text description for it in the *Accessibility* palette.

## 3.3 Floating Field

You can use the *Floating Field* element to insert text variables into a static text. A typical example of this is when the name of a customer is added to a form. By default, a floating field is a text field object (as shown in Chapter 3.2).

To insert a floating field, proceed as follows:

1. Choose the text object in your form where you want to place the floating field.
2. Place the cursor on the exact position where you want to insert the floating field.
3. In the *Adobe LiveCycle Designer* menu, choose *Insert* → *Floating Field*.
4. To set properties for a floating field, move the focus to the field and then choose the appropriate options.

### Dos for Floating Field

- For floating fields, set the *Screen Reader Precedence* to *None* in the *Accessibility* palette. Otherwise, the screen reader will read the technical name of the object.

### Don'ts for Floating Field

- Do not create a caption or a description for floating fields in the *Accessibility* palette. Generally, floating texts are inserted into static texts and do not need captions or descriptions.

## 3.4 Static Image

You can also insert images in forms. *Adobe LiveCycle Designer* supports the formats BMP, JPG, GIF, PNG, TIF, and EXIF. A typical example of a static image is a company logo (as shown in Figure 3.3).



**Figure 3.3: Static Image**

### Dos for Static Images

- If the image object or scanned image includes important information for the form, create a text for the image in the *Accessibility* palette (as shown in Chapter 2.8), describing the object and its purpose.

- The text for a company logo, for example, could consist of the words “company logo” and the name of the company.
- If the image object contains semantic color information, include this in the description as well. The screen reader can only read the text created for the image in the *Accessibility* palette. A description of a green traffic light, for example, could be “Transmission successful” and the description of a red light could be “Transmission failed”.
- If you use larger graphics, such as bar charts, provide the information in an alternative accessible version, such as a table.

### Don'ts for Static Images

- Do not create text descriptions for static images that are only used for decoration.
- Do not use scanned data as background information, such as when scanning a print form and using *Adobe LiveCycle Designer* to add new fields to the form. Screen readers cannot detect the scanned data in this state.

## 3.5 Image Field

The image field object is used as a placeholder for an image loaded dynamically at runtime. This means that either the image content or a reference to the content must be included in the runtime data used to create the PDF form or print format in Adobe Document Services (ADS).

### Dos for Image Fields

- If the image field object shows information, create a text description for it in the *Accessibility* palette (as shown in Chapter 2.8).
- Use tooltips or custom screen reader texts for images that do not have a caption, even if these images are merely decorative. Otherwise, the screen reader will read the technical name of the object.
- You can use the *FormCalc* scripting language to assign text descriptions to an image field object dynamically. *FormCalc* is the standard scripting language of *Adobe LiveCycle Designer*; JavaScript is an alternative. More information is available in the *Adobe LiveCycle Designer* help documentation.
- The following example illustrates how you can use *FormCalc* scripting to make dynamic assignments:
- If the image field is called `ImageField1` and the associated text is included in the `imagetext` node in the runtime data, then you can use scripting to pass this text in an appropriate event (such as `form:ready`). The following is a scripting example:

```
ImageField1.assist.tooltip = $record.imagetext.value
```

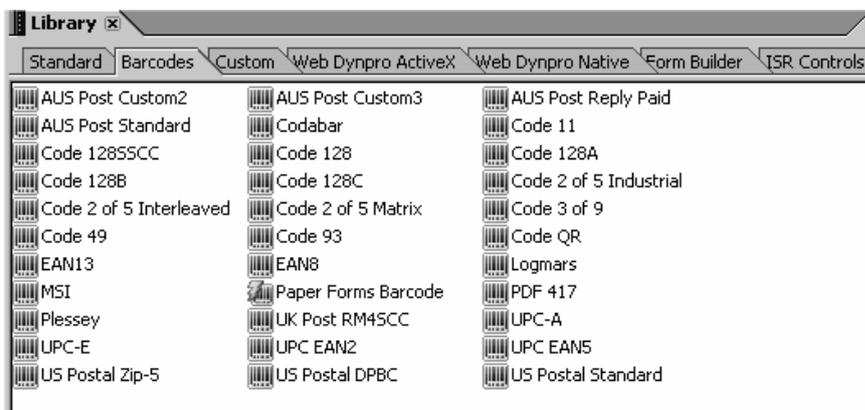
### Don'ts for Image Fields

- Do not create text descriptions for image field objects that are only used for decoration.

## 3.6 Barcode

Barcodes contain machine-readable information in parallel lines of variable widths and spacing. This encrypted information can be interpreted by a special scanner. A list of the supported barcode formats is available in the *Adobe LiveCycle Designer* help documentation.

You insert a barcode by choosing the *Barcodes* tab page in the *Library* palette and selecting a barcode type (as shown in Figure 3.4).



**Figure 3.4: Barcodes**

You can create a description text for a barcode in the *Accessibility* palette (as shown in Chapter 2.8). You can change the technical name (barcode type) in the *Hierarchy* palette. When you make these changes, the screen reader no longer reads the barcode type and only reads the barcode value. This is because, initially, the technical name is the same as the barcode type.

In interactive forms, you can also change the field value of the barcode by moving the focus to the object and editing the default value.

### Do's for Barcodes

- When you use the *Accessibility* palette to assign a description text, the barcode type is no longer read by the screen reader. If the barcode type contains important information, add the barcode type to the description (as shown in Chapter 2.8).
- When you change the technical name (or barcode type) in the *Hierarchy* palette, set the screen reader precedence to "name" so that the screen reader can read the new text you enter there.
- When you change the technical name (or barcode type) in the *Hierarchy* palette, set the screen reader precedence to "name" so that the screen reader can read the new text you enter there. If the barcode type contains important information, you can create the barcode type as a description in the *Accessibility* palette (as shown in Chapter 2.8).

### Don'ts for Barcodes

- Do not use scanned barcodes.

## 3.7 Check Box

You can use check boxes to depict an option in a form as selected (active) or not selected (not active). A check box in *Adobe LiveCycle Designer* consists of a circular or square element whose state can be toggled by the user, and an associated caption. The state of a check box

can be *On (active)* or *Off (not active)*. In interactive forms, the user can set the status (on or off) electronically on the computer.

Figure 3.5 shows an active check box:



**Figure 3.5: Check Box**

#### Do's for Check Boxes

- Create a meaningful caption for every check box by focusing on it and changing the automatic caption into appropriate text.
- Place the caption on the right of the check box. The caption is positioned on the right of the check box automatically when you create it. If it is not for some reason, you can change its position on the *Layout* palette (as shown in Chapter 2.9).
- For check boxes in table cells, set the *Screen Reader Precedence* to *None* in the *Accessibility* palette. Otherwise, the screen reader will read the technical name of the object.
- If you create alternative text information for a check box in the *Accessibility* palette, the screen reader no longer reads the visible caption text. This means you will have to include the caption in your alternative text. Always check whether the caption text is descriptive enough before resorting to alternative texts in the *Accessibility* palette.

#### Don'ts for Check Boxes

- Do not create captions for check boxes inserted in table cells. The column header of the table is used as the caption for these checkboxes.
- Do not place check boxes on master pages. They are not supported on master pages of static forms.

## 3.8 Radio Button

A radio button is a UI element that you can use to select a single item from a list of items. A radio button in *Adobe LiveCycle Designer* consists of a circular or square element, a caption, and a group title. In interactive forms, the user makes a selection electronically on the computer.

Figure 3.6 shows two radio buttons with a visible title. You can select only one of these two radio buttons at a time:



**Figure 3.6: Radio Button Group**

#### Do's for Radio Buttons

- Create a meaningful caption for every radio button by focusing it and changing the automatic caption to an appropriate text.

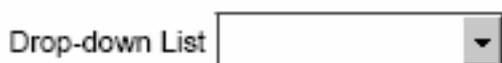
- Place the caption on the right of the radio button. The caption is positioned on the right of the radio button automatically when you create it. If it is not for some reason, you can change its position on the *Layout* palette (as shown in Chapter 2.9).
- If you create alternative text information for a radio button in the *Accessibility* palette, the screen reader no longer reads the visible caption text. Include the caption in your alternative text. Always check whether the caption text is descriptive enough before resorting to alternative texts in the *Accessibility* palette.
- Create a **visible** title for the radio button by creating a static text and placing it in front of or above the group.

### Don'ts for Radio Buttons

- The user must not trigger any unexpected changes to the screen when he or she selects a radio button in an interactive form.
- Do not place radio buttons on master pages. They are not supported on master pages for static forms.

## 3.9 Drop-Down List

A drop-down list shows you a list of options from which you can select one to be displayed. You can use it to select an item from a list of items, making it easier to enter data. Drop-down lists are used only in interactive forms, and not in print forms. Figure 3.7 shows a drop-down list.



**Figure 3.7: Drop-Down List**

### Dos for Drop-Down Lists

- Create a meaningful caption for every drop-down list.
- Position the caption on the left of the object. The caption is positioned on the left of the drop-down list automatically when you create it. If it is not for some reason, you can change its position on the *Layout* palette (as shown in Chapter 2.9).
- For drop-down lists in table cells, set the *Screen Reader Precedence* to *None* in the *Accessibility* palette. Otherwise, the screen reader reads the technical name of the object.
- You can create alternative text descriptions in the *Accessibility* palette (as shown in chapter 2.8). If you create alternative text information for a drop-down list, the screen reader no longer reads the visible caption text. Include the caption in your alternative text. Always check whether the caption text is descriptive enough before resorting to alternative texts in the *Accessibility* palette.

### Don'ts for Drop-Down Lists

- Do not create captions for drop-down lists inserted in table cells. The column header of the table is used as the caption for these drop-down lists.
- Do not use drop-down lists in print forms.

- Avoid causing conflicts with the keyboard and screen reader when you create client-side scripts for interactive forms. For example, *Change* events used in drop-down lists can trigger unexpected actions.

### 3.10 List Box

A list box also shows you a list of options from which you can select only one. Unlike drop-down lists, however, you can display more than one option at the same time. List boxes are used only in interactive forms, and not in print forms. Figure 3.8 shows a list box.



**Figure 3.8: List Box**

#### Do's for List Boxes

- Create a meaningful caption for every list box.
- Position the caption above the object. The caption is positioned above the list box automatically when you create it. If it is not for some reason, you can change its position on the *Layout* palette (as shown in Chapter 2.9).
- You can create alternative text descriptions in the *Accessibility* palette (as shown in chapter 2.8). If you create alternative text information for a drop-down list, the screen reader no longer reads the visible caption text. Include the caption in your alternative text. Always check whether the caption text is descriptive enough before resorting to alternative texts in the *Accessibility* palette.

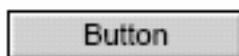
#### Don'ts for List Boxes

- Do not use list boxes in print forms.
- Avoid causing conflicts with the keyboard and screen reader when you create client-side scripts for interactive forms. For example, *Change* events used in list boxes can trigger unexpected actions.

### 3.11 Button

A button, also known as a pushbutton, is an object used to execute commands or client requests. Buttons are used only in interactive forms, and not in print forms.

Figure 3.9 shows a button:



**Figure 3.9: Button**

### Dos for Buttons

- Create a meaningful caption for every button by focusing on it and changing the automatic caption to an appropriate text.
- You can create alternative text descriptions in the *Accessibility* palette (as shown in chapter 2.8).

### Don'ts for Buttons

- Do not use buttons in print forms.

## 3.12 Table

You can use tables as a more effective and structured way of displaying information. A table generally has the following parts:

- Table name (also known as a table title or header)
- Header row (also known as a column header row)
- Body row

The first row of a table is usually the header row. The rows below the header row are the body rows. It is important that you use the correct elements in *Adobe LiveCycle Designer* to define header rows and body rows. Figure 3.10 shows a simple table with three columns, a header row, and three body rows.

You can use *complex tables* to depict complex information. Examples of complex tables are tables with nested columns, multiple header rows, or table cells that cross two or more columns. It is generally more difficult for users to navigate around complex tables. Investigate other simpler ways of depicting your data before resorting to a complex table.

This chapter only discusses simple tables in depth. More information about tables is available in the *Adobe LiveCycle Designer* help documentation.

Header	Header	Header

**Figure 3.10: Table**

### Dos for Tables

- Create simple tables instead of complex tables. Simple tables display content in the most readable way.
- Before you use complex tables, investigate whether you can redesign your data and display it in a simpler manner.
- To create tables, use the dedicated table object in *Adobe LiveCycle Designer*.

- Create meaningful **visible** names or titles for your tables. You can create a table name as a static text in *Adobe LiveCycle Designer* (as shown in chapter 3.1) and place it in front of the table. You can group a table and its name together in a subform. Subforms are particularly useful when you want to combine associated objects in a layout.
- Define the first row of the table as a **header row**.
- Create a meaningful name for every cell in the header row. This name is then used as the column header.
- Define the rows below the header row as **body rows**.
- For objects in table cells, set the *Screen Reader Precedence* to *None* in the *Accessibility* palette. Otherwise, the screen reader reads the technical name of the object.

#### Don'ts for Tables

- Do not use subforms to create simple tables.
- Do not create captions for objects inserted in table cells (such as check boxes, radio buttons, or drop-down lists). The header row (column header) is used as the caption instead.

### 3.13 Circle, Line, and Rectangle Objects

You can use circle, line, and rectangle objects in your PDF form for graphical purposes. These objects do not represent any informational content, which is why they do not have tags (as discussed in Chapter 2.5). This also means that these objects are ignored by screen readers.

#### Dos for Circle, Line, and Rectangle Objects

- Use the line object to draw lines in forms instead of using text characters such as hyphens or underscores. Screen readers read any characters used to draw lines as actual repeated characters, for example, *Hyphen, Hyphen, Hyphen...*

#### Don'ts for Circle, Line, and Rectangle Objects

- Do not use circle, line, or rectangle objects to draw other objects such as check boxes or tables. Screen readers will ignore these objects because they do not have tags.
- You do not need to create alternative text descriptions for circles, lines, or rectangles since they are only used as decoration and have no informational content.

## 4 Useful Links

The following list contains the links used in this document.

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)

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>

HTML Techniques for Web Content Accessibility Guidelines 1.0

<http://www.w3.org/TR/WCAG10-HTML-TECHS/>