Accessibility Best Practices
For Adobe LiveCycle Designer ES3 v10.0

CONTENTS

1.0 Introduction

2.0 Best practices for creating forms

3.0 Techniques for testing form accessibility

4.0 Mapping between guidelines and best practices

5.0 Useful links

1.0 Introduction

An accessible form is one that almost everyone can use, including those who may have disabilities that affect how they are able to interact with the form on the computer screen. Users with visual impairments or reduced mobility, for example, are still able to use accessible forms.

Adobe LiveCycle Designer includes a number of features and capabilities that enhance the usability of forms for users with various disabilities, and that assist form authors in creating PDF forms that are more accessible to people with disabilities.

Building accessibility into forms not only allows the widest possible audience for content, it is a requirement when supplying documents in regions where compliance with accessibility standards is mandated. In the United States, for example, accessibility standards such as Section 508 of the Rehabilitation Act exist to ensure that information technology is available to all users, including government employees with disabilities and members of the public with disabilities that consume government services.

LiveCycle Designer helps developers comply with the requirements mandated by accessibility standards. Its component-based approach enables form builders to take advantage of built-in accessibility features. LiveCycle Designer also provides support for creating accessible XHTML files (using LiveCycle Forms) and PDF forms. Accessible PDF forms include a complete logical structure plus additional information about a document's contents to increase accessibility.

For more information on Adobe's accessibility solutions, visit the Accessibility Resource Center at www.adobe.com/accessibility.

2.0 Best practices for creating forms

LiveCycle Designer enables you to build rich form content and comply with Section 508 guidelines. This guide contains an overview of the best practices for creating an accessible form, and guidelines for implementing these best practices using LiveCycle Designer. The following best practices are covered:

2.1 Keep forms simple and easy to use

2.2 Configure form properties to generate accessibility information

2.3 Choose the right controls

2.4 Provide text equivalents for images

2.5 Provide proper labels for form controls

2.6 Ensure the reading and tab order are correct

2.7 Ensure form controls are keyboard accessible

2.8 Use color responsibly

2.9 Provide heading cells for tables

2.10 Provide a navigable form structure

2.11 Avoid disruptive scripting

2.12 Ensure all audio and video content is accessible

2.13 Identify natural language and any changes in language

2.1 Keep forms simple and easy to use

A form is not accessible if it is not easy to use. You should try to design forms that are simple and usable. A simple layout of controls and fields with clear, meaningful captions and tool tips will make the form much easier for all users to use.

Designing forms that are uncluttered and logically arranged, and that provide clear and simple instructions, will help all users to fill forms as easily as possible. Navigation features, such as the tab order and keyboard shortcuts, should support the logical order of objects on the form.

2.1.1 Avoid moving, blinking, or flashing content

Some individuals with photosensitive epilepsy can have a seizure triggered by movement in frequencies greater than 2 Hz (1 Hz, or Hertz, equals one per second) and lower than 55 Hz (55 per second).

Movement at less than 2 Hz is considered slow enough as to be safe for individuals with photosensitive epilepsy. Movement at more than 55 Hz is considered to be unperceivable.

Developers should be aware of these parameters when using any movement in web content.

Other users may have cognitive disabilities that make it difficult to concentrate when there is animated or blinking content present in the form.

In general, try to avoid using optical effects inserted by scripts, such as flashing text or animation, in interactive forms. Such effects reduce the usability of the forms for certain users.

Related checkpoints

- Section 508 §11934.21

    - (h) When animation is displayed, the information shall be displayable in at least one non-animated presentation mode at the option of the user.
    - (k) Software shall not use flashing or blinking text, objects, or other elements having a flash or blink frequency greater than 2 Hz and lower than 55 Hz.
- Section 508 §11934.22
    - (j) Pages shall be designed to avoid causing the screen to flicker with a frequency greater than 2 Hz and lower than 55 Hz.
- WCAG 1.0
    - 7.1 Until user agents allow users to control flickering, avoid causing the screen to flicker. (P1)
    - 7.2 Until user agents allow users to control blinking, avoid causing content to blink (i.e., change presentation at a regular rate, such as turning on and off) (P2).
    - 7.3 Until user agents allow users to freeze moving content, avoid movement in pages.
    - 14.1 Use the clearest and simplest language appropriate for a site's content.
- WCAG 2.0
    - 2.2.2 Pause, Stop, Hide: For moving, blinking, scrolling, or auto-updating information, all of the following are true: (Level A)
    - 2.3.1 Three Flashes or Below Threshold: Web pages do not contain anything that flashes more than three times in any one second period, or the flash is below the general flash and red flash thresholds. (Level A)
    - 2.3.2 Three Flashes: Web pages do not contain anything that flashes more than three times in any one second period. (Level AAA)

2.2 Configure form properties to generate accessibility information

For a form to be accessible, it must be perceivable by assistive technology. For example, most screen readers will not consider the visual layout of your form, but rather the underlying structure.

To implement this underlying structure using LiveCycle Designer, you must create a PDF form with accessibility information (sometimes referred to as tags) included so that the screen reader or other assistive technology can read the form's text and components. In a form with accessibility information, each element contains information about its own structure, plus information about how it is related to or dependent on other elements. Only in PDF files with accessibility information included can screen readers identify and describe the content of a document accurately.

To create an accessible form, you must configure form properties to have LiveCycle Designer generate accessibility information when saving the form design as a PDF file:

1. Choose File > Form Properties.
2. Click the Save Options tab and, in the PDF area, ensure that Generate Accessibility Information (Tags) For Acrobat is selected.
3. Click OK.

In LiveCycle Designer, this option is selected by default.

Note: These options only apply when saving the form design as a PDF file. They do not apply to PDF files created with LiveCycle Forms which has configuration options that are independent of this option in LiveCycle Designer.

Related checkpoints

- Section 508 §1194.21
    - (d) Sufficient information about a user interface element including the identity, operation and state of the element shall be available to assistive technology. When an image represents a program element, the information conveyed by the image must also be available in text.
    - (l) When electronic forms are used, the form shall allow people using assistive technology to access the information, field elements, and functionality required for completion and submission of the form, including all directions and cues.
- Section 508 §1194.22
    - (n) When electronic forms are designed to be completed on-line, the form shall allow people using assistive technology to access the information, field elements, and functionality required for completion and submission of the form, including all directions and cues.

2.3 Choose the right controls

When you design your forms, use development objects from the tabs available in LiveCycle Designer's Object Library. You can display this panel by choosing Window > Object Library or by pressing Shift+F12 (see Figure 1).

Figure 1: Object Library Panel

If you use other objects, they may be ignored by assistive technology. Using only the standard objects saves you the additional effort of defining Accessibility properties for objects you have created yourself. If you do create and use your own custom objects, be sure to use the Accessibility palette to set accessibility properties such as Role, Tool Tip, Screen Reader Precedence, and Custom Screen Reader Text. To show the Accessibility palette, choose Window > Accessibility.

Related checkpoints

- Section 508 §1194.21
    - (c) A well-defined on-screen indication of the current focus shall be provided that moves among interactive interface elements as the input focus changes. The focus

shall be programmatically exposed so that assistive technology can track focus and focus changes.

- (d) Sufficient information about a user interface element including the identity, operation and state of the element shall be available to assistive technology. When an image represents a program element, the information conveyed by the image must also be available in text.
- (l) When electronic forms are used, the form shall allow people using assistive technology to access the information, field elements, and functionality required for completion and submission of the form, including all directions and cues.

- Section 508 §1194.22
  - (n) When electronic forms are designed to be completed on-line, the form shall allow people using assistive technology to access the information, field elements, and functionality required for completion and submission of the form, including all directions and cues.
- WCAG 2.0
  - 3.2.4 Consistent Identification: Components that have the same functionality within a set of Web pages are identified consistently. (Level AA).
  - 4.1.2 Name, Role, Value: For all user interface components (including but not limited to: form elements, links and components generated by scripts), the name and role can be programmatically determined; states, properties, and values that can be set by the user can be programmatically set; and notification of changes to these items is available to user agents, including assistive technologies. (Level A)

2.4 Provide text equivalents for images

Images can help improve comprehension for users with some types of disabilities. However, for screen reader users, images will decrease the accessibility of your form if you do not provide a textual alternative.

If you choose to use images, provide text descriptions for all image and image field objects. Ensure that the text describes the object and its purpose on the form. When you define a text alternative, the screen reader will read this alternative when it encounters the image. For this reason, an image containing information must always have a text alternative specified.

You provide text descriptions using the Tool Tip or Custom Screen Reader Text properties in the Accessibility palette or via text fields, captions, and object names, as specified in the Name option of the Binding tab. For example, Figure 2 shows an example of an image that contains the text "Get Adobe Reader". Since a screen reader is not able to read text that is part of an image, you should include a text alternative in the Custom Screen Reader Text field in the Accessibility palette for this object. In most cases, the alternative text should be the same as the text that is visible in the image (see Figure 2).

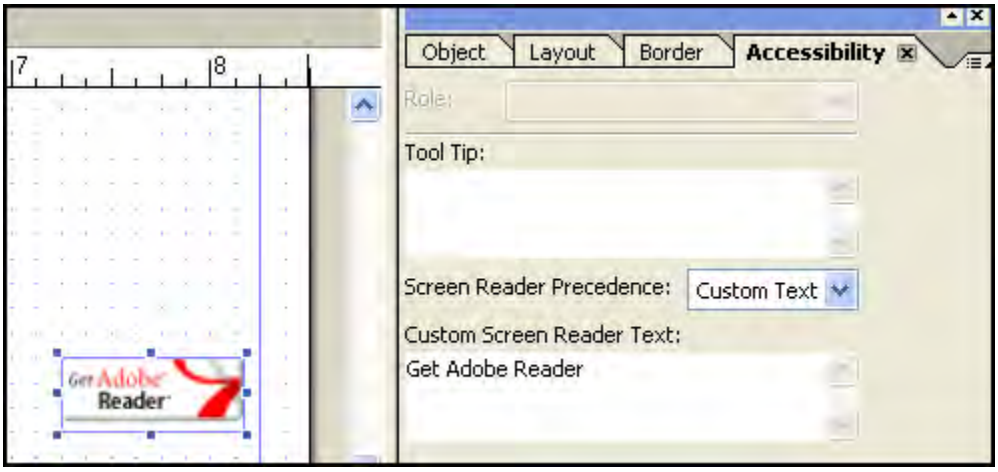

Figure 2: Specifying alternative text for an image using the Accessibility palette

When specifying the alternative text, consider the following:

- If the image object or scanned image includes important information for the form, create text for the image in the Accessibility palette that describes 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. 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 complex graphics, such as bar charts, provide the information in an alternative accessible version, such as a table or longer textual description.
- Do not create text descriptions for static images that are only used for decoration.
- Do not use scanned data as background information. This can happen when a designer scans a print form and uses Adobe LiveCycle Designer to add new fields to the form. Screen readers cannot detect the scanned data in this state.

When you are including purely decorative graphical content into your forms, you want to make sure that screen readers do *not* announce the image's presence. For most screen readers, this can be achieved by setting the Screen Reader Text property to None in the Accessibility palette. If you don't do this, some screen readers may announce the presence of a graphic, without indicating what the graphic represents. For dynamic images, such as image field objects, ensure that text alternatives are properly updated when the image is changed. Do not create text descriptions for image field objects that are only used for decoration. 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. For example, consider a form with an image field named ImageField1 and associated text in the `imagetext` node of the runtime data. You can use scripting to pass this text in an appropriate event (such as `form:ready`)as follows:

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

Related checkpoints

- Section 508 §1194.22
    - (a) A text equivalent for every non-text element shall be provided (e.g., via "alt", "longdesc", or in element content).
- WCAG 1.0
    - 1.1 Provide a text equivalent for every non-text element (e.g., via "alt", "longdesc", or in element content). This includes: images, graphical representations of text (including symbols), image map regions, animations (e.g., animated GIFs), applets and programmatic objects, ascii art, frames, scripts, images used as list bullets, spacers, graphical buttons, sounds (played with or without user interaction), stand-alone audio files, audio tracks of video, and video (P1).
- WCAG 2.0
    - 1.1.1 Non-text Content: All non-text content that is presented to the user has a text alternative that serves the equivalent purpose, except for the situations listed below. (Level A)

2.5 Provide proper labels for form controls

A form control's label or caption identifies what the form control is supposed to represent. For example, the text "First name" tells users that they have to enter their first name in a text field. To be accessible by screen readers, the label must be programmatically associated with the form control or the form control must be configured with additional accessibility information using the Accessibility palette; it is not enough to just place a text object next to the control. For sighted or low vision users it is important that the label is properly positioned adjacent to the control. Both techniques will be discussed in the following sections.

2.5.1 Specifying accessible label text using the Accessibility palette

The label that is perceived by screen reader users does not necessarily have to be the same as the visual caption. In some cases you may want to be more specific about the control's purpose.

For each field object in a form, the Accessibility palette (see Figure 3) can be used to specify what the screen reader will announce to identify the specific form field.

To use the Accessibility palette, follow these steps:

1. Display the Accessibility palette by choosing Window > Accessibility or by pressing Shift+F6.
2. Select an object in your form. The palette will show the object's accessibility properties.

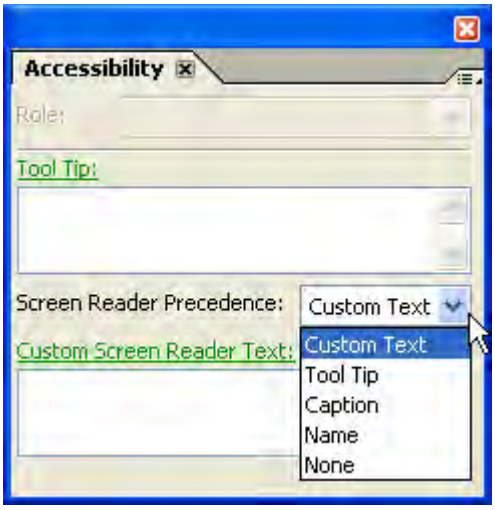

Figure 3: The Accessibility palette

When the form is saved as a PDF, LiveCycle Designer searches the form for Custom Text, Tool Tip, Caption, and Name properties, in that order, to find text to be read by screen readers. You can override this default order by using the Screen Reader Precedence option in the Accessibility palette:

1. Select the object on the form design.
2. Click the Accessibility palette.
3. Select any Screen Reader Precedence option other than None.

The following options are available:

- **Custom Text**, which you set in the Accessibility palette's Custom Screen Reader Text field. This option lets you specify any text you want assistive technology, such as screen readers, to use. Using the Caption setting is best for most situations – creating Custom Screen Reader Text should be considered an option only when using the Caption or a toolTip is not possible.
- **Tool Tip**, which you set in the Accessibility palette's Tool Tip field. For most objects, tool tips appear at run time when the user hovers the pointer over the object. Tool tips appear for some read-only objects, such as a paper form's barcode object, only when a screen reader is in use.
- **Caption**, which will cause LiveCycle Designer to use the form field's associated (visual) label as screen reader text.
- **Name**, which you set in the Binding tab's Name field. Note that this name cannot contain any spaces.
- **None**, which will cause the object to not have a name. This is never recommended for form controls.

Consider the following when using the Accessibility palette for form control labeling:

- If your form control's caption properly describes the control, then it is accessible for screen readers. In this case, either leave both Custom Text and Tool Tip fields empty in the Accessibility palette, or change the Screen Reader Precedence to Caption.
- When targeting screen readers, there is no point in specifying different text descriptions for the same form control, as only one will be used: The first non-empty field in the Screen Reader Precedence order. For example, there is no reason to specify both Custom Text and Tool Tip text for a screen reader.
- By default, the screen reader reads the caption if nothing is specified in the Tool Tip box or the Custom Screen Reader Text box.

- Do not use the Accessibility palette to create descriptions for any invisible fields or areas.
- If you have to create a description using the Tool Tip or Custom Screen Reader Text options, 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.
- For checkboxes and drop-down list controls in table cells, the screen reader will announce whatever caption, tool tip, or custom screen reader text you specify for the object. If you want to use the column header for the alternative text for these objects when placed in a table, then do not provide a caption, tool tip, or custom screen reader text.
- If the control requires additional instructions, make sure these are included in the text alternative as well. Include enough spoken information for users to know what input is expected and how to complete the field correctly, but do not overwhelm users with redundant information.
- Don't provide unneeded information describing how to operate controls – let the user's assistive technologies handle this for the user. Users can configure the verbosity to suit their comfort levels.

Figure 4 shows an example of a text field with a visual caption that may be unclear for some screen reader users. In this example, Custom Screen Reader Text is set to "Number of Pages", and the Screen Reader Precedence is set to Custom Text. As a result the actual (visual) caption text ("# of pages") will not be used by the screen reader. Alternatively, a Tool Tip could have been specified.

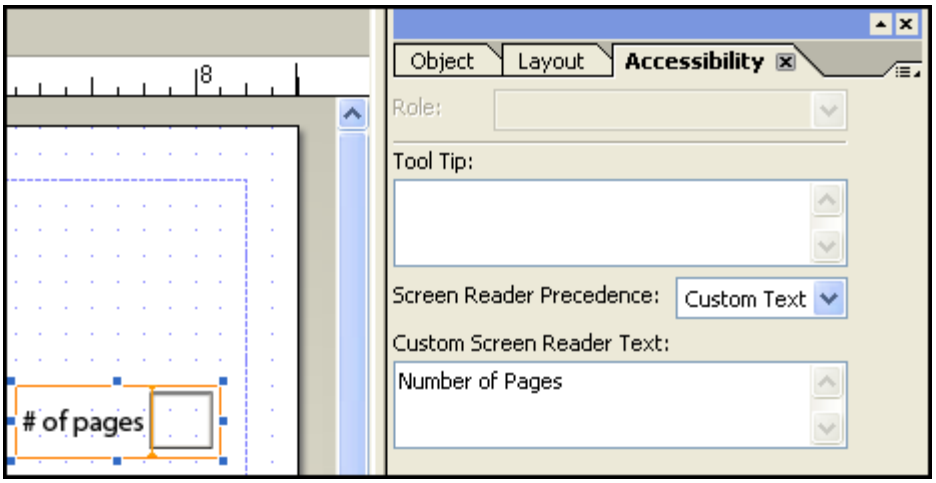

Figure 4: Specifying Custom Screen Reader Text when the visible label is inadequate

2.5.2 Labeling radio buttons

When a user with vision impairment tabs into a radio button, the screen reader needs to read two things:

- An indication of the purpose of the group of radio buttons
- A meaningful label for each radio button

To make radio buttons accessible using the button captions:

1. In the Hierarchy palette, select the exclusion group.
2. Click the Accessibility palette and, in the Custom Screen Reader Text box, type the text to be read for the group. For example for an exclusion group indicating options for payment by various credit cards, type **Select a method of payment**.

3. If the captions for each radio button provide text that will be meaningful when spoken by a screen reader, then in the Object palette, select the Binding tab and deselect Specify Item Value.

To make radio buttons accessible using a specified item value:

1. In the Hierarchy palette, select the exclusion group.
2. Click the Accessibility palette and, in the Custom Screen Reader Text box, type the text to be read for the group. For example for an exclusion group indicating options for payment by various credit cards, type **Select a method of payment**.
3. In the Hierarchy palette, select the first radio button in the group.
4. In the Object palette, click the Field tab. In the Item area, double-click the item and type a meaningful value for the selected radio button. For example for the first button in a group of payment methods, you might type **Cash**.
5. Repeat steps 3 and 4 for each radio button in the exclusion group.

2.5.3 Labeling custom controls

It is strongly recommended to use standard components rather than custom components, as they will provide the assistive technology with the correct cues and information by default. However, if custom controls are used, consider the following:

- Announce the state of checkboxes and radio buttons.
- In list boxes and drop-down lists, announce the default item selected in the list. Be sure that the user knows to use the Up Arrow and Down Arrow keys to move through the list items. Note that pressing the Tab key or the Enter or Return key will select the item in the list. Using scripting, you can set the object's Change event to announce which item is selected from the list.
- Announce to users any special keystrokes they need to perform a function, for example, pressing the spacebar to select a button or the Down Arrow key to select an item from a list box.

2.5.4 Correctly positioning a control's caption

The placement of a caption is important because users will expect them to be found adjacent to the control. For screen magnification users this is even more important, as they may not be able to view both the control and the caption at the same time if they are too far apart.

When you create an object, LiveCycle Designer automatically positions the caption as specified by the object type. The captions of radio buttons, for example, are placed on the right. This default placement is always the best location for an accessible caption. If you must change the position of the caption text, use these steps:

1. Select the object by moving the focus to it.
2. In the Layout palette, select the position of your object's caption from the Position option in the Caption section, at the bottom of the palette.

The example in Figure 5 shows a text box with a caption above it. The Position in the Layout palette is set to Top. The default location of the caption is to the left of the text box.

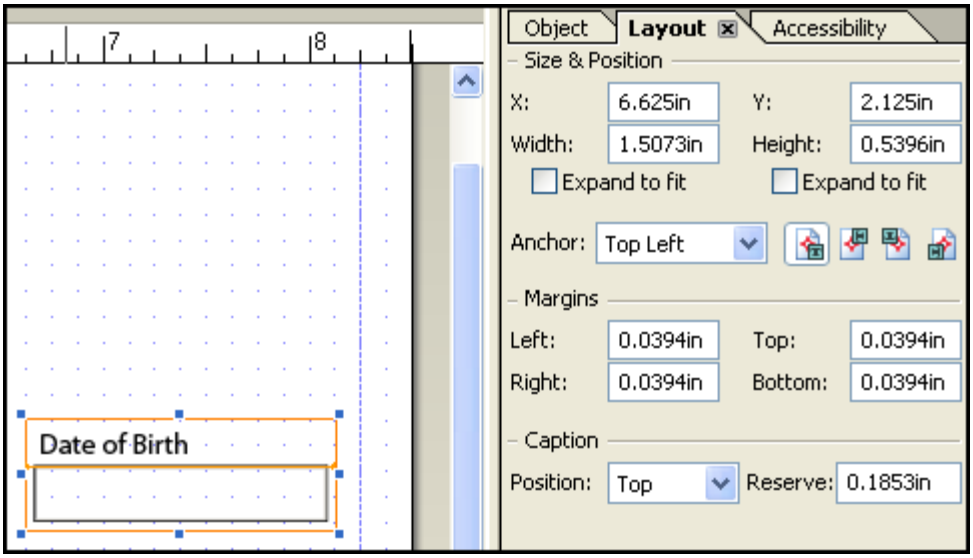

Figure 5: Changing caption positioning using the Layout palette

The following table provides overview of label placement rules for commonly used controls.

Control Type	Placement Rules
Text Input (including date, time and password fields)	Place the caption to the left of the control (default). If this is not possible, place it immediately above or below it. Labels should be positioned close to the control for users with increased magnification so that the label and control are more likely to be seen together in the magnified view.
Checkbox	Place the caption to the right of the checkbox (default). For checkbox controls in table cells, the screen reader will announce whatever caption, tool tip, or custom screen reader text you specify for the object. If you want to use the column header as the alternative text for a checkbox in a table, then do not provide a caption, tool tip, or custom screen reader text.
Radio Button Group	Create a visible title for the radio button group by creating a static text element and placing it in to the left of or above the group. For each individual radio button, place the label to the right (default).
Drop-Down List	Place the caption to the left of the object (default). If this is not possible, place it immediately above it. For drop-down list controls in table cells, the screen reader will announce whatever caption, tool tip, or custom screen reader text you specify for the object. If you want to use the column header as the alternative text for these objects in a table, then do not provide a caption, tool tip, or custom screen reader text.
List Box	The caption is positioned above the list box by default when you create it.
Button	The caption is automatically placed on the button and does not have to be positioned manually. Ensure that the button's purpose is properly described by the caption text.

2.5.5 Dynamically populating a Tool Tip or Custom Screen Reader Text.

You can also dynamically populate a form control's text alternative such as its Tool Tip, with a value from a data source. For example, you can display a custom Tool Tip for an object that is in French.

The schema you connect to could have the following defined for a Tool Tip:

```
<form>
   <tooltip dp_tt="tooltip1"/>
</form>
```

The data file you point to could have the following defined for a tool tip:

```
<form>
   <tooltip dp_tt="Quantité - Entrez un nombre inférieur ou égal à
100."/>
</form>
```

1. In the Object Library palette, click the Standard category and drag an object onto the form design. For example, insert a Text Field object.
2. (Optional) In the Object palette, click the Field tab and type a caption for the object in the Caption box. For example, type Quantité.
3. In the Accessibility palette, click the Tool Tip active label.
4. Select the data connection.
5. Click the triangle beside the Binding box and select a binding. For example, select tooltip > @dp_tt.
   The following string appears in the Binding box: `$record.tooltip.dp_tt` Tip:
   You could type this string into the Items box instead of selecting it.
6. Click OK.
7. View the form in the Preview PDF tab.

2.5.6 Providing link text

Users of assistive technology may have different methods of reading linked text. For example, screen reader users often use a links list such as the one shown in Figure 6 to quickly scan the available links on a page.

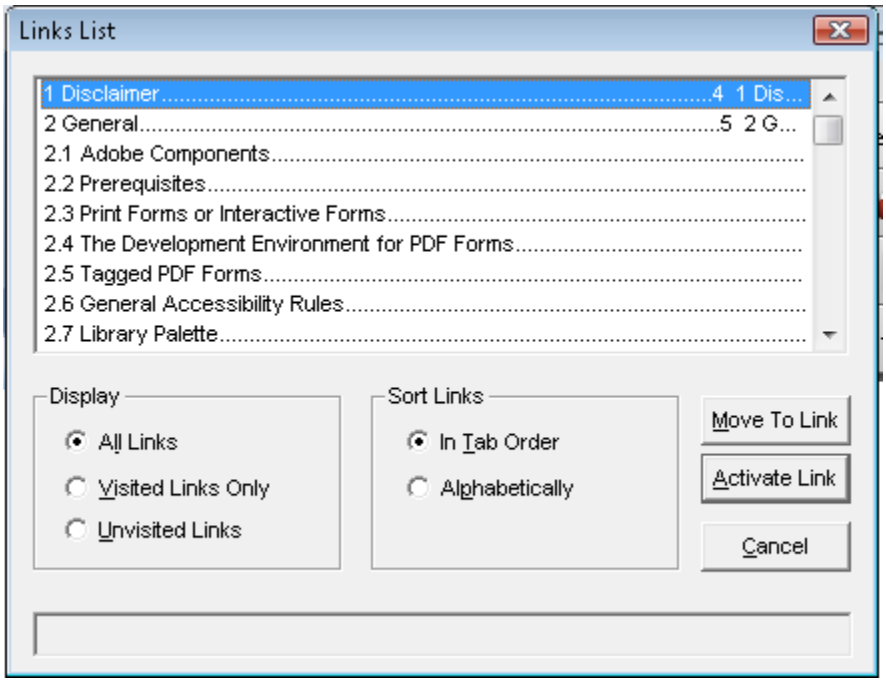

Figure 6: The JAWS Links List dialog box

For this reason links must be self describing; that is their meaning should not depend on their context (the surrounding text). For example, the words "click here" might form the actual link element in the phrase "click here to download our application form". Such a link would be difficult to understand when read through a links list, especially when there are multiple links containing the same text.

When using links in your form, make sure that each link properly describes its purpose, without depending on its surrounding text or position on the page. For example, instead of using a phrase such as "Click Here" as link text, use "Download application form" as the link text.

Related checkpoints

- Section 508 §1194.21
    - (d) Sufficient information about a user interface element including the identity, operation and state of the element shall be available to assistive technology. When an image represents a program element, the information conveyed by the image must also be available in text.
    - (l) When electronic forms are used, the form shall allow people using assistive technology to access the information, field elements, and functionality required for completion and submission of the form, including all directions and cues.
- Section 508 §1194.22
    - (n) When electronic forms are designed to be completed on-line, the form shall allow people using assistive technology to access the information, field elements, and functionality required for completion and submission of the form, including all directions and cues.
- WCAG 1.0
    - 12.4 Associate labels explicitly with their controls (P2).
    - 13.1 Clearly identify the target of each link (P2).
- WCAG 2.0
    - 1.1.1 Non-text Content: All non-text content that is presented to the user has a text alternative that serves the equivalent purpose, except for the situations listed below. (Level A)
    - 2.4.6 Headings and Labels: Headings and labels describe topic or purpose. (Level AA)
    - 3.2.4 Consistent Identification: Components that have the same functionality within a set of Web pages are identified consistently. (Level AA)
    - 3.3.2 Labels or Instructions: Labels or instructions are provided when content requires user input. (Level A)
    - 4.1.2 Name, Role, Value: For all user interface components (including but not limited to: form elements, links and components generated by scripts), the name and role can be programmatically determined; states, properties, and values that can be set by the user can be programmatically set; and notification of changes to these items is available to user agents, including assistive technologies. (Level A)

2.6 Ensure the reading and tab order are correct

Ensuring a meaningful reading order is very important when designing forms that are accessible to users with vision impairment or other disabilities. These users typically do not use a mouse to navigate through a form, so they depend on the keyboard. The reading order determines the sequence used by screen reader users as they read through your form. Additionally, the tab order allows users to quickly move from one interactive form control to the next using the Tab or Shift+Tab keys. A logical tab order ensures that they have access to all the fields on the form and that they can navigate the form in a way that is sensible and efficient.

The reading order of the form includes all static objects (such as text and images) and field objects, but only the interactive form controls are part of the tab order.

Note: In many cases, the tab order is closely related to the reading order. For simplicity, the term "tab order" will be used in place of "tab or reading order" in this guide.

2.6.1 The default tab order in LiveCycle Designer forms

The default tab order is automatically created when you save your form as a tagged PDF. Initially, the tab order in a form is determined from the local position of the objects using the following rules:

- All objects are ordered 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.

For simple forms (that is, forms with a left-to-right, top-to-bottom layout), the default tab order will usually be correct. To verify this, you should examine the default tab order before publishing your form. You can make the tab order visible with either of the following methods:

- Choose View > Show Tab Order.
- Click Show Order in the Tab Order palette.

All objects will be displayed with a number in the upper right corner, indicating the object's place in the default tab order. The interactive objects in this sequence form the tab order. Figure 7 shows the reading order visualization of a basic form.

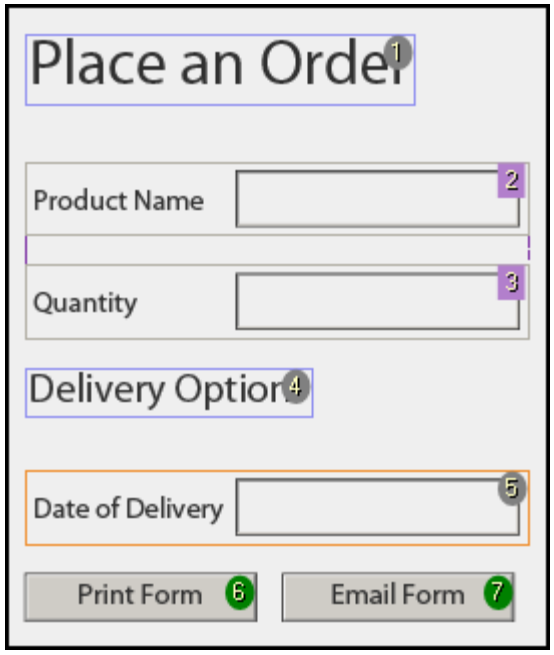

Figure 7: Visualization of the default reading order for a typical order form

Each tab order number is shown in a colored shape. The shapes have the following meaning:

- Gray circles (#1 and #4) are used for objects in the content area.
- Green circles (#6 and #7) are used for master page objects.
- Lavender squares (#2 and #3) are used for objects inside a fragment.

You can choose to only show interactive form controls (which make up the tab order), or all objects in the reading order (which also includes static objects such as text and images). To change this preference, choose Tools > Options > Tab Order and select Only Show Tab Order For Fields.

On a complex form, it may be difficult to see how the tabbing flows from one object to the next. You can use visual aids to help you see the tabbing flow on the form. With the visual aids turned on, when you hover the pointer over the object, blue arrows show the tabbing flow for the two preceding and two following objects in the tab order (see Figure 8).

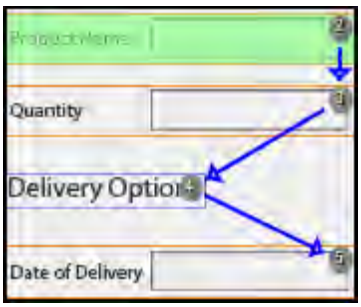

Figure 8: Visual aids highlight the tab order

To enable the visual aids, use of the following methods:

- Choose Tools > Options > Tab Order and, in the Tab Order panel, select Display Additional Visual Aids For Tab Order.
- In the Tab Order palette menu, select Show Visual Aids.

2.6.2 Using position to influence the default tab order

To influence the default tab order, you can change an object's coordinates by moving it to a different location. For example, in Figure 9, the Product Name field occurs in the tab order before the Quantity field. To change this order, you can move the Product Name field so that it is placed below or to the right of the Quantity field.

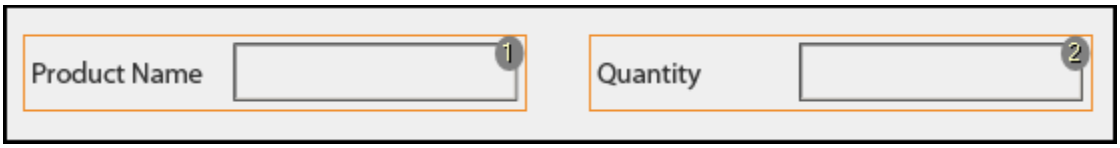

Figure 9: The default tab order is left to right

You can change an object's position by doing one of the following:

- Drag it using the mouse
- Select it, and move it using the keyboard arrow keys.

Note: It can be helpful to maintain object alignment by choosing View > Snap To Grid.

You can change an object's coordinates more precisely using the Layout palette (shown in Figure 10). This palette allows you to specify X and Y coordinates, as well as the object's width and height.

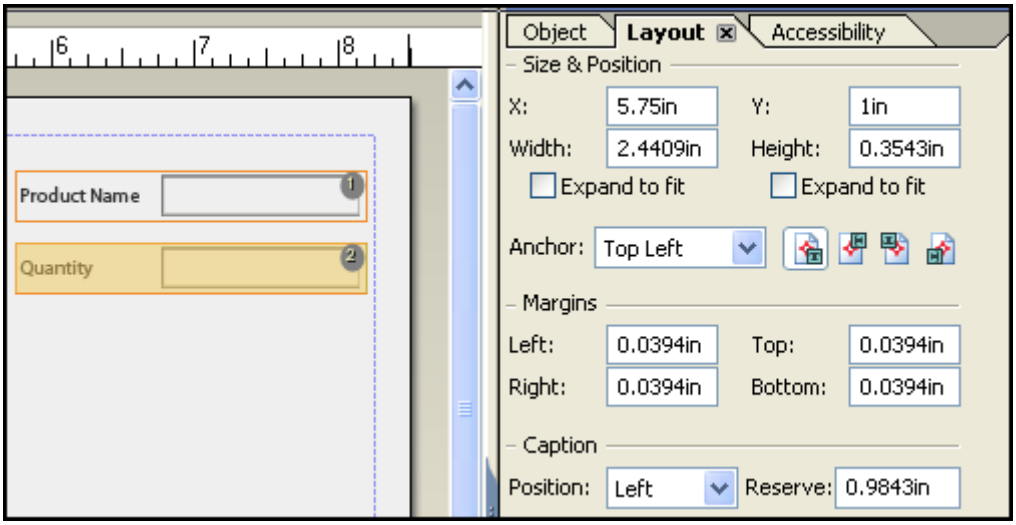

Figure 10: Using coordinates to precisely position an object with the Layout palette

Note: When the caption and the control are not merged, the position of a form control's caption is independent of the order in which screen readers read the object and its elements. For more information about captions, see the section 2.5 Provide proper labels for form controls in this guide.

2.6.3 Using subforms to influence the default tab order

As mentioned above, subforms allow you to insert groups of objects that have their own tab order. You can create a subform by doing one of the following:

- Choose Insert > Standard > Subform.
- Select your objects in the Hierarchy palette, and group them in a subform by choosing Insert > Wrap In Subform.
- Select your objects in the actual form, right-click the selection, and choose Wrap In Subform

When two subforms containing field objects are positioned side-by-side, the tabbing sequence will go through the fields in the first subform before moving on to the next. This is illustrated in Figure 11, where two subforms are used to create a column-based default tab order.

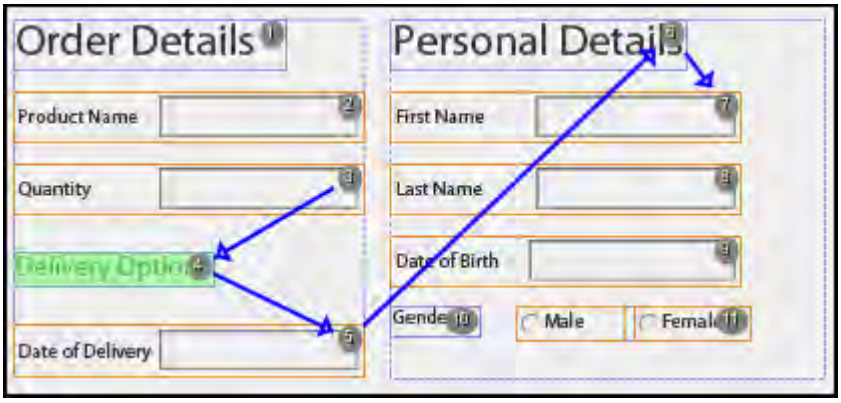

Figure 11: Default tab order using subforms

Subforms, radio buttons, and content areas, along with the vertical position of objects on a page and its master page, all affect the tab order.

2.6.4 Creating a custom tab order using the Tab Order palette

You can change the default tab order when you require a different sequence in your form and the change cannot be achieved with positioning or grouping in subforms. To change the default tab order you can create a custom tab order using the Tab Order palette.

The Tab Order palette (see Figure 12) allows you to inspect and modify the order in which objects in your form are read by assistive technology and navigated by the user's Tab key.

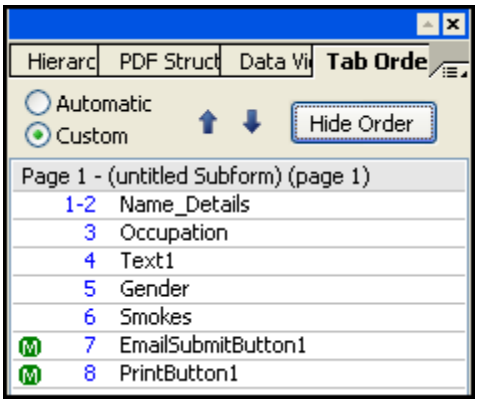

Figure 12: The Tab Order palette

The Tab Order palette provides an alternative view of the tab order on the form. It shows all the objects on the form as a numbered list, where each number represents the position of the object in the tabbing flow.

To open the Tab Order palette, choose Window > Tab Order.

The Tab Order palette provides the following visual markers:

- A gray bar marks each page of the form. The tab order on each page starts with the number 1.
- The letter M inside a green circle indicates master page objects (visible only when viewing the form on the Design View tab).
- A range of numbers indicates objects within a fragment.
- A yellow background indicates the currently selected item.
- A lock icon beside the first object on the page indicates that the object cannot be moved within the tab order (visible only when viewing the form on the Master Pages tab).

The list shows the same tab order numbers as the numbers displayed on the form itself when you choose View > Show Tab Order. You change the position of an object in the tab order by moving the object up or down in the Tab Order palette list. You can move a single object or a group of objects. This can be achieved through one of the following methods:

- Drag the selected object up or down the list and place it at the desired location. A black handle marks your current position within the list before you place the object.
- In the Tab Order palette, click the up or down arrow buttons until the selected object is placed in the correct position. Alternatively, press Ctrl+Up Arrow or Ctrl+Down Arrow.
- In the Tab Order palette menu, select Move Up or Move Down.
- In the Tab Order palette list, click the selected object (or select it and press F2) to make the number listed beside the object name editable. Then, type the number indicating the new position of the object in the tab order and press Enter.

- Select Copy from the Tab Order palette menu and, in the list, select the object above which to place the object you are moving, and then select Paste from the menu.

When you move the object to a new place in the order, LiveCycle Designer reassigns the tab order numbers.

Although the tab order for the objects that are located on a master page is displayed on the Design View tab, you can change the order for these objects only on the Master Pages tab. If you use fragment references in your form, the tab order inside a fragment is visible when viewing the order for the form. To change the tab order inside a fragment, you must open the fragment source file for editing, make the change, and save the file. Any forms that use this fragment are affected by this change.

If you decide that you do not want the customized tab order on your form, you can quickly return to the automatic (default) tab order using the following steps (you will lose any changes made to the tab order):

1. On the Tab Order palette, select Automatic.
2. In the message box that appears, click Yes to confirm the removal of the custom tab order.

Related checkpoints

- Section 508 §1194.21
    - (a) When software is designed to run on a system that has a keyboard, product functions shall be executable from a keyboard where the function itself or the result of performing a function can be discerned textually.
- WCAG 1.0
    - 9.2 Ensure that any element that has its own interface can be operated in a device-independent manner.
- WCAG 2.0
    - 1.3.2 Meaningful Sequence: When the sequence in which content is presented affects its meaning, a correct reading sequence can be programmatically determined. (Level A)
    - 2.1.1 Keyboard: All functionality of the content is operable through a keyboard interface without requiring specific timings for individual keystrokes, except where the underlying function requires input that depends on the path of the user's movement and not just the endpoints. (Level A)
    - 2.1.3 Keyboard (No Exception): All functionality of the content is operable through a keyboard interface without requiring specific timings for individual keystrokes. (Level AAA)
    - 2.4.3 Focus Order: If a Web page can be navigated sequentially and the navigation sequences affect meaning or operation, focusable components receive focus in an order that preserves meaning and operability. (Level A)

2.7 Ensure form controls are keyboard accessible

Users must be able to fill the form completely using only the keyboard or an equivalent alternative input device. Users with reduced mobility or impaired vision may have no choice but to use the keyboard, and many users who can use a mouse simply prefer keyboard input. By allowing various input methods, you not only create accessible forms, you also create forms that are better suited to the preferences of all users.

In LiveCycle Designer, the easiest way to ensure that your controls are keyboard accessible is by using the controls listed under the Common tab in the Object Library palette. These controls respond to both mouse and keyboard input by default. For more information, see the section 2.3 Choose the right controls in this guide.

Another important aspect of keyboard accessibility is ensuring that each interactive element is part of the form's tab order. This allows the user to move the cursor forward and backward through the form with the Tab and Shift+Tab keys. Be sure to set a logical tab order that includes all fields and buttons. For more information, see the section 2.6 Ensure the reading and tab order are correct in this guide.

Finally, it is important to ensure that scripted behavior is keyboard accessible as well, and is not dependent upon device-specific events. The mouse event `MouseEnter`, for instance, cannot be executed using the keyboard. Also, such event handlers must not interfere with keyboard accessibility. For example, make sure that `Change` events used in drop-down lists or list boxes do not trigger unexpected actions.

Related checkpoints

- Section 508 §1194.21
    - (a) When software is designed to run on a system that has a keyboard, product functions shall be executable from a keyboard where the function itself or the result of performing a function can be discerned textually.
- WCAG 1.0
    - 6.4 For scripts and applets, ensure that event handlers are input device-independent (P2).
    - 9.2 Ensure that any element that has its own interface can be operated in a device-independent manner (P2).
    - 9.3 For scripts, specify logical event handlers rather than device-dependent event handlers (P2).
- WCAG 2.0
    - 2.1.1 Keyboard: All functionality of the content is operable through a keyboard interface without requiring specific timings for individual keystrokes, except where the underlying function requires input that depends on the path of the user's movement and not just the endpoints. (Level A)
    - 2.1.2 No Keyboard Trap: If keyboard focus can be moved to a component of the page using a keyboard interface, then focus can be moved away from that component using only a keyboard interface, and, if it requires more than unmodified arrow or tab keys or other standard exit methods, the user is advised of the method for moving focus away. (Level A)
    - 2.1.3 Keyboard (No Exception): All functionality of the content is operable through a keyboard interface without requiring specific timings for individual keystrokes. (Level AAA)

2.8 Use color responsibly

Designing forms for accessibility involves considering some additional guidelines for using color. Designers use colors to improve the appearance of forms by highlighting various form components. Improper use of color, however, may make information in your form difficult or impossible to read by people with disabilities.

2.8.1 Do not convey information using color alone

Colors can emphasize and enhance certain parts of your form, but you should not convey information by color alone.

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.

For example, Figure 13 shows a form field that has a red caption (specified using the Font palette) to indicate the form field is required. In this example, the color is the only signifier of the difference between required and optional input fields, which makes it impossible for blind users or users with certain types of color blindness to tell them apart.

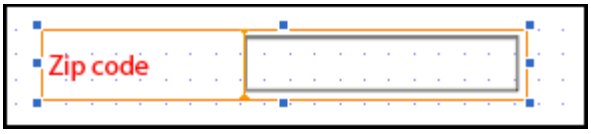

Figure 13: Using color alone to convey information

To solve this problem, also indicate the form's required status in the form control's alternative text (as described in the section 2.5 Provide proper labels for form controls). For example, you could set the screen reader text to "Zip code (required)". For users who have difficulties seeing color in certain combinations, it is recommended to set the text field type to User Entered – Required in the Object palette in addition to alternate text that indicates that the field is required. Alternatively, you can use indications other than color, such as visual text, text styles, and border styles. However, for screen reader users you will still have to convey the required information using the Accessibility palette.

Also, when providing descriptions or instructions to the form user, keep in mind that statements based on color alone are insufficient for users with a visual impairment. For example, instead of a statement such as, "Click the green button to continue," use a text description for actions, such as "Click the Next button to continue."

Note: This best practice does not prohibit the use of color. It prohibits the use of color *as the sole means of conveying important information*. If a visual indication is still desired for this sort of information, the designer could use an asterisk or similar visual indicator to mark required fields.

2.8.2 Provide sufficient color contrast

Many users with vision impairment rely on high contrast between text and the background to read forms. When the contrast between background and foreground colors is not sufficient, a form can become difficult if not impossible to read for some users. Figure 14 shows an example of a form with insufficient contrast.

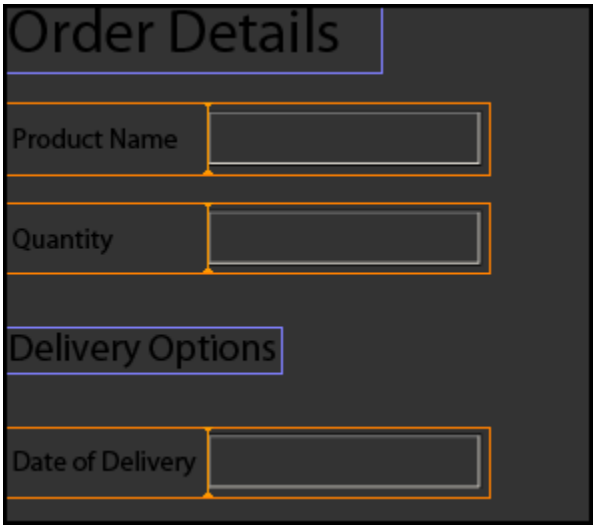

Figure 14: A form with insufficient color contrast

It is strongly recommended that you use the default font and background colors: black on a white background. If you must change these default colors, be sure to choose an appropriate combination of high-contrast colors; use either a dark foreground color on a light background color, or vice versa. To be certain, use a tool (such as the WAT-C Color Contrast Analyzer) to verify that the contrast is sufficient.

Adobe Reader and Adobe Acrobat allow users to specify whether colors have to be replaced to meet their visual needs. Users may specify their own contrast scheme, or they may choose to use a scheme provided by the operating system. Additionally, Adobe Reader and Adobe Acrobat have their own high contrast scheme that may be enabled. For these options to be successful, the best approach is always to use default colors.

While designing your form, test it frequently using a color scheme setting similar to what many users with vision impairment will be using to complete your form. This practice helps you discover and correct issues early in the design process.

Recommendations for using colors:

- Make sure that no information is lost if the semantic color is not visible.
- 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 or Adobe Acrobat. Mac OSX only offers a simple grayscale filter for high contrast so this is not sufficient for testing.
- Do not convey information solely based on 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.

Related checkpoints

- Section 508 §1194.21
    - (i) Color coding shall not be used as the only means of conveying information, indicating an action, prompting a response, or distinguishing a visual element.
- WCAG 1.0

    - 2.1 Ensure that all information conveyed with color is also available without color, for example from context or markup.
    - 2.2 Ensure that foreground and background color combinations provide sufficient contrast when viewed by someone having color deficits or when viewed on a black and white screen. [Priority 2 for images, Priority 3 for text] (P2).
- WCAG 2.0
    - 1.4.1 Use of Color: Color is not used as the only visual means of conveying information, indicating an action, prompting a response, or distinguishing a visual element. (Level A)
    - 1.4.3 Contrast (Minimum): The visual presentation of text and images of text has a contrast ratio of at least 4.5:1, except for the following: (Level AA)
    - 1.4.6 Contrast (Enhanced): The visual presentation of text and images of text has a contrast ratio of at least 7:1, except for the following: (Level AAA)

2.9 Provide heading cells for tables

Tables are an effective way to organize and present content in accessible forms. When used appropriately, a table's rows and columns provide a predictable and consistent structure for form content.

For example, when a screen reader user navigates into a body row cell, the screen reader specifies the cell location and then reads the cell content. The screen reader specifies the cell location using a combination of row and column headers or row and column numbers. Because screen readers provide information that orients the user to the location of content in the table, its layout directly affects the table's accessibility.

You can specify the following roles for table elements as you construct tables. These roles allow screen readers to navigate the table structure using special shortcuts, and will convey to the user the relationship between table cells and corresponding header cells.

- **Table**
  Assigns the role of a table to the selected subform. When the user navigates to this subform, most screen readers identify it as a table and indicate the number of rows and columns.
- **Header Row**
  Assigns the role of a header row to the selected subform or table row. When speaking the contents of a body row cell, most screen readers first identify the content of the corresponding cell in the header row.
- **Body Row**
  Assigns the role of a body row to the selected subform or table row. If a cell contains a subform, screen readers typically speak the content of the corresponding cell in the header row, followed by the fields in the subform.
- **Footer Row**
  Assigns the role of a footer row to the selected subform or table row.
- **(None)**
  Specifies a row that conveys information about the table or its content. The row is not considered to be part of the table; however, the screen reader will read its contents.

When used properly, tables are an effective way to organize and present tabular information. Avoid overly complex tables, such as those with nested tables and sections.

2.9.1 Making simple tables accessible

Tables with simple layouts are recommended. Simple tables begin with a single header row followed by the body rows.

When designing simple tables for accessibility, keep these guidelines in mind:

- The tab order for a table is geographic order, which is the same as for the form itself. Ensure that the table content is organized such that it makes sense when read from left to right and top to bottom.
- Most screen readers interpret the first row in a table as the header row. When reading the content of a body row cell, these screen readers first read the content of the associated header row cell. Ensure that the content in each header row cell meaningfully describes the column content.
- Avoid cells that span two or more columns, nested tables, or table sections. Some screen readers have difficulty interpreting these features correctly or may not use them. For example, if a cell in a body row spans two columns, screen readers may not reference the correct cell content in the header row when reading the next cell in the row.

2.9.2 Making complex tables accessible

When designing tables for accessibility, strive to keep the table layout simple, with one header row followed by body rows. Of course some content may require a more complex table layout. For example, you may need to use cell spanning or more than one header to effectively convey the content.

You can create complex tables by using the table object or by combining subform objects. The table object lets you use features that are intended to help the design process, such as options for inserting and resizing columns and rows.

Using the Accessibility palette, you can specify table related roles to subforms to create an accessible complex table. Depending on your design experience and preferences, you may choose to create complex tables by combining subform objects. For example, you can create one subform that includes two rows and specify this subform as the header for the table and specify another subform for the table body rows.

When using subform objects instead of table objects to create tables, the following additional steps are required:

- On the Subform tab, set the type for each subform to Positioned.
- In the Accessibility palette, set the appropriate subform role for each subform that makes up the table. For example, assign the role of Header Row to the subform that is used as the table header.
- For rows that convey information about the table or its content but that are not considered to be part of the table, assign the subform role of None. The screen reader will read the row content.

The features supported by the screen reader determine the information read for a complex table. For example, consider a table that includes a header row and a section with a header row. When the user navigates into a body row cell in the table section, screen readers will typically read the following content, in order:

- Content from the appropriate cell in the header row for the table
- Content from the appropriate cell in the header row for the section
- Content from the selected cell

Some screen readers, however, may not read content from both header rows.

Create meaningful *visible* names or titles for your tables. You can create a table name as static text in Adobe LiveCycle Designer 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.

For controls in table cells, the screen reader will announce whatever caption, tool tip, or custom screen reader text you specify for the object. If you want to use the column header as the alternative text for a control in a table, then do not provide a caption, tool tip, or custom screen reader text. Be advised, however, that this strategy is not always as clear for screen reader users since screen readers may only associate the column heading with the control when the user is not in the form interaction mode of the screen reader.

Related checkpoints

- Section 508 §1194.22
    - (g) Row and column headers shall be identified for data tables.
    - (h) Markup shall be used to associate data cells and header cells for data tables that have two or more logical levels of row or column headers.
- WCAG 1.0
    - 5.1 For data tables, identify row and column headers (P1).
    - 5.2 For data tables that have two or more logical levels of row or column headers, use markup to associate data cells and header cells (P1)
- WCAG 2.0
    - 1.3.1 Info and Relationships: Information, structure, and relationships conveyed through presentation can be programmatically determined or are available in text. (Level A)

2.10 Provide a navigable form structure

When a form becomes long and complex, its ease of use will be greatly influenced by the way it is structured. Just as a book becomes easier to understand when it is divided into chapters and sections, a form becomes easier to use when it is divided into headings and subheadings. This partitioning is especially useful for screen reader users, for the following reasons:

- Each heading tells the screen reader user what can be expected in the section following the heading.
- Screen readers provide shortcuts to quickly jump back and forth between the different headings in the form, and also allow the user to access a list of headings, which provides an overview of the document structure and allows for quick navigation.

Providing mechanisms that enable users to skip to other areas of the form can make the form more convenient. You can add a heading structure to your form using the Accessibility palette in LiveCycle Designer.

2.10.1 Provide skipping mechanisms

Sighted users can scan a page in any order. They may start by looking at the lower right corner of the page, and scan backwards through the content. The screen reader user does not have this option because the screen reader will start reading the page in the upper left (as presented in the source code) and move through in a linear order. In addition, the sighted user can scan the page looking for interesting links and activate them with the mouse. A screen reader user must move through the page sequentially.

The easiest and most effective way to provide a navigable form structure is to use structural headings and properly defined lists in your form.

You can also provide mechanisms that allow the user to skip to other areas of the form, for example by adding navigational buttons to the top and bottom of the form. At the top of a form, you could include buttons such as Open Data File, Previous Page, and Next Page. At the bottom of the form you could include buttons such as Save Data, Email Data, Go to Top of Page, and Print.

Smart fields can be an effective way to make some forms easier to fill. For example, a travel request form may have several rows and columns of fields. If a particular row is empty, pressing the Tab key on the last item in that row could jump to the next section of the form rather than continuing to tab through a number of fields that will remain empty.

2.10.2 Adding headings using the Accessibility palette

You can use the Accessibility palette to assign roles to objects based on what the object is used for. These roles can be applied to create headings at different levels.

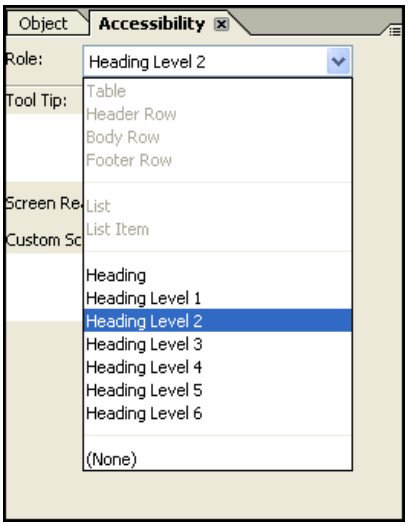

Figure 15: Specifying a heading role in the Accessibility palette

Follow these steps to create a heading in your form:

1. Identify the start of each logical segment of your form using static text labels,
2. For each label and select one of the heading options as the Role in the Accessibility palette. The different heading levels (1 to 6) enable you to create a heading structure in your form. Start with level 1, and then use level 2 and so on for nested subsections.

Most screen readers allow users to quickly navigate between heading elements based on their level.

Figure 16 shows a form that is divided into smaller segments using headings. In this example, the following heading structure is used:

- Heading Level 1: Product Request
    - Heading Level 2: Order Details
        - Heading Level 3: Delivery Options
    - Heading Level 2: Additional Information
        - Heading Level 3: Personal Details
        - Heading Level 3: Address

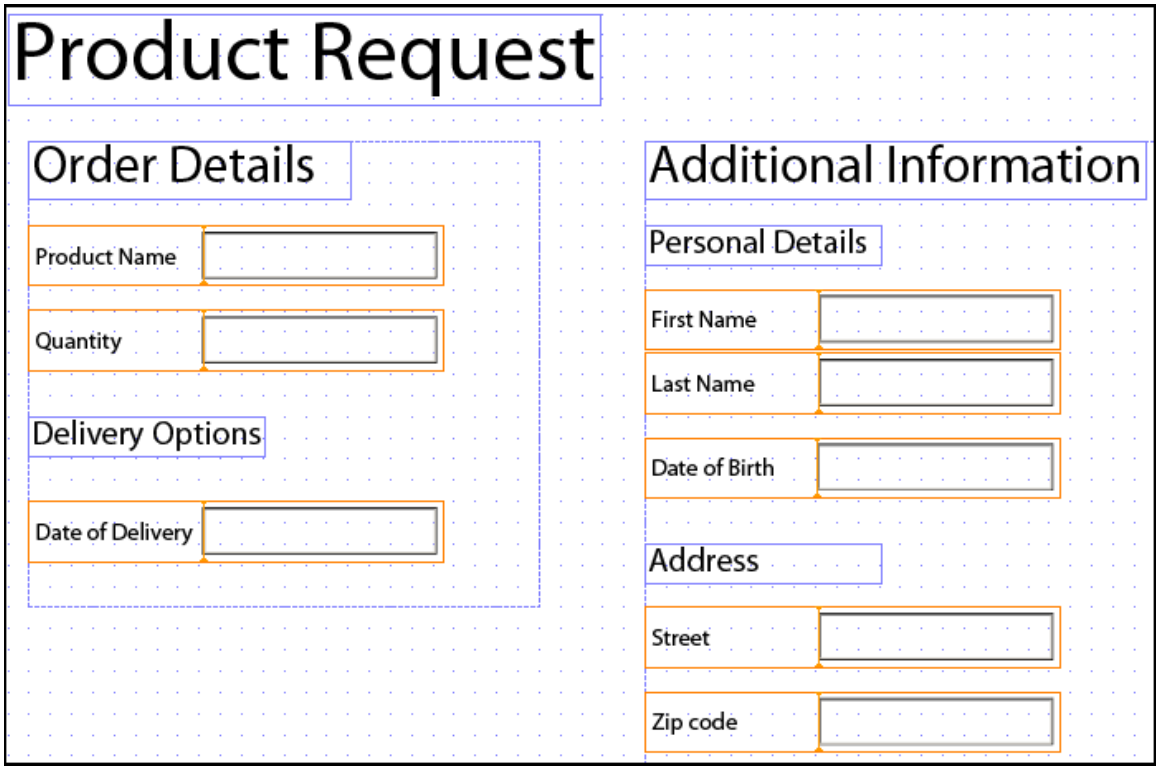

Figure 16: Structuring a form using headings

These headings are just static text elements that were given a specific font size and a heading role with the appropriate level.

Note: Simply changing a text label's visual appearance to look like a heading will not make screen readers recognize it as a heading. You must to apply a heading role.

Always make sure the order of the heading levels is logical. For example, a subsection of a level 2 heading must always be a level 3 heading; you should never skip levels when marking up subsections. Screen reader users use the different levels to get a better understanding of the form's structure. For example, after encountering a level 2 heading, the user may use a shortcut to look for level 3 headings and determine if there are any subsections. If you skip levels, the user will have difficulties identifying these subsections.

2.10.3 Marking up lists

Sometimes it may also be useful to add list content to your form. Lists are useful to group related items together, and they allow screen reader users to know how many items there are in a list and quickly navigate past it. Properly marking up lists makes your form's structure more clear to screen reader users.

In LiveCycle Designer you create lists using subforms with the following steps:

1. Select a subform that contains the content that will be marked as list items.
2. In the Accessibility palette, select List as the Role.
3. Select each nested subform within the List subform, and set its Role to List Item.

Note: A List Item role can only be assigned to a subform that is contained in a subform that has a List role specified. You cannot define a table or table row as a list or list item; however, a list item can contain a table.

Related checkpoints

- Section 508 §11934.22
    - (o) A method shall be provided that permits users to skip repetitive navigation links.
- WCAG 1.0
    - 3.5 Use header elements to convey document structure and use them according to specification (P2).
    - 3.6 Mark up lists and list items properly. (P2).
    - 12.3 Divide large blocks of information into more manageable groups where natural and appropriate. (P2).
    - 13.3 Provide information about the general layout of a site (e.g., a site map or table of contents).
    - 13.4 Use navigation mechanisms in a consistent manner (P2).
- WCAG 2.0
    - 1.3.2 Meaningful Sequence: When the sequence in which content is presented affects its meaning, a correct reading sequence can be programmatically determined. (Level A)
    - 2.4.1 Bypass Blocks: A mechanism is available to bypass blocks of content that are repeated on multiple Web pages. (Level A)
    - 2.4.5 Multiple Ways: More than one way is available to locate a Web page within a set of Web pages except where the Web Page is the result of, or a step in, a process. (Level AA)
    - 2.4.6 Headings and Labels: Headings and labels describe topic or purpose. (Level AA)
    - 2.4.10 Section Headings: Section headings are used to organize the content. (Level AAA)
    - 3.2.3 Consistent Navigation: Navigational mechanisms that are repeated on multiple Web pages within a set of Web pages occur in the same relative order each time they are repeated, unless a change is initiated by the user. (Level AA)

2.11 Avoid disruptive scripting

As part of the form design process, form developers can use scripts to provide a richer user experience. You can add scripts to most form fields and objects. For example, you can create simple scripts to dynamically update values on an interactive form in response to user input.

When designing scripts for accessibility, consider these general guidelines:

- Keep the form content free of visual interruptions. For example, avoid features that cause content to flicker, blink, or move.
- Ensure that pop-up windows appear only as a result of user-initiated actions. Similarly, do not allow the current focus of the form (the user's current view) to change or content to redisplay unless initiated by the user. For example, if the user is completing fields in the lower half of the form, do not allow the focus to change to the upper-left corner of the form unless the user chooses to navigate to this location.
- Users with disabilities may require more time to provide input in fields. Do not specify time-based responses for input fields.

- Be aware that client-side scripts can interfere with screen readers and keyboards if the script changes the focus of the client application. For example, the `change` and `mouseEnter` events, when used with drop-down lists or list boxes, have the potential to cause unexpected actions. Verify that your client-side scripts do not introduce problems for screen reader users and keyboard-only users.
- Users of assistive technology will sometimes require additional time to complete tasks. In any case where a timed routine is about to expire, display an accessible message to allow for an extension. Alert boxes created via JavaScript are usable by assistive technology. A new window with a message alerting the user of an impending time out may also be deployed.

Related checkpoints:

- Section 508 §1194.22
    - (l) When pages utilize scripting languages to display content, or to create interface elements, the information provided by the script shall be identified with functional text that can be read by assistive technology.
    - (p) When a timed response is required, the user shall be alerted and given sufficient time to indicate more time is required.
- WCAG 1.0
    - 1.4 For any time-based multimedia presentation (e.g., a movie or animation), synchronize equivalent alternatives (e.g., captions or auditory descriptions of the visual track) with the presentation (P1).
    - 6.2 Ensure that equivalents for dynamic content are updated when the dynamic content changes.
    - 6.3 Ensure that pages are usable when scripts, applets, or other programmatic objects are turned off or not supported. If this is not possible, provide equivalent information on an alternative accessible page.
    - 6.5 Ensure that dynamic content is accessible or provide an alternative presentation or page (P2).
    - 8.1 Make programmatic elements such as scripts and applets directly accessible or compatible with assistive technologies [Priority 1 if functionality is important and not presented elsewhere, otherwise (P2)
    - 9.3 For scripts, specify logical event handlers rather than device-dependent event handlers (P2).
    - 10.1 Until user agents allow users to turn off spawned windows, do not cause pop-ups or other windows to appear and do not change the current window without informing the user.
- WCAG 2.0
    - 3.2.1 On Focus: When any component receives focus, it does not initiate a change of context. (Level A)
    - 3.2.2 On Input: Changing the setting of any user interface component does not automatically cause a change of context unless the user has been advised of the behavior before using the component. (Level A)
    - 3.2.5 Change on Request: Changes of context are initiated only by user request or a mechanism is available to turn off such changes. (Level AAA)

2.12 Ensure all audio and video content is accessible

If your forms incorporate audio or video content, including audio and video clips, you must ensure that this content is accessible. Specifically, make sure that video clips incorporated into forms contain captions (sometimes called subtitles) for deaf and hard of hearing users and video descriptions for blind users. For audio files that are not synchronized with video content, a simple transcript is sufficient.

For Flash based media, consult [link] for information on providing captions.

Related checkpoints:

- Section 508 §1194.22
    - (b) Equivalent alternatives for any multimedia presentation shall be synchronized with the presentation.
- WCAG 1.0
    - 1.1 Provide a text equivalent for every non-text element (e.g., via "alt", "longdesc", or in element content). *This includes*: images, graphical representations of text (including symbols), image map regions, animations (e.g., animated GIFs), applets and programmatic objects, ascii art, frames, scripts, images used as list bullets, spacers, graphical buttons, sounds (played with or without user interaction), stand-alone audio files, audio tracks of video, and video (P1).
    - 1.3 Until user agents can automatically read aloud the text equivalent of a visual track, provide an auditory description of the important information of the visual track of a multimedia presentation (P1).
    - 1.4 For any time-based multimedia presentation (e.g., a movie or animation), synchronize equivalent alternatives (e.g., captions or auditory descriptions of the visual track) with the presentation (P1).
- WCAG 2.0
    - 1.2.1 Audio-only and Video-only (Prerecorded): For prerecorded audio-only and prerecorded video-only media, the following are true, except when the audio or video is a media alternative for text and is clearly labeled as such: (Level A)
    - 1.2.2 Captions (Prerecorded): Captions are provided for all prerecorded audio content in synchronized media, except when the media is a media alternative for text and is clearly labeled as such. (Level A)
    - 1.2.3 Audio Description or Media Alternative (Prerecorded): An alternative for time-based media or audio description of the prerecorded video content is provided for synchronized media, except when the media is a media alternative for text and is clearly labeled as such. (Level A)
    - 1.2.4 Captions (Live): Captions are provided for all live audio content in synchronized media. (Level AA)
    - 1.2.5 Audio Description (Prerecorded): Audio description is provided for all prerecorded video content in synchronized media. (Level AA)
    - 1.2.6 Sign Language (Prerecorded): Sign language interpretation is provided for all prerecorded audio content in synchronized media. (Level AAA)
    - 1.2.7 Extended Audio Description (Prerecorded): Where pauses in foreground audio are insufficient to allow audio descriptions to convey the sense of the video, extended audio description is provided for all prerecorded video content in synchronized media. (Level AAA)

- 1.2.8 Media Alternative (Prerecorded): An alternative for time-based media is provided for all prerecorded synchronized media and for all prerecorded video-only media. (Level AAA)
- 1.2.9 Audio-only (Live): An alternative for time-based media that presents equivalent information for live audio-only content is provided. (Level AAA)

2.13 Identify natural language and any changes in language

Form content will be read by assistive technologies that use speech synthesizers that are language-specific, so it is important to correctly identify the primary language of the form to ensure that forms are read in the intended language.

If the text (or alternative text) in your forms is presented in more than one language, you must identify the areas of your form in which a switch is made from one language to another.

In LiveCycle Designer, setting the primary language is accomplished by setting the Locale property of the form and the Locale property for the top-level subform. To identify changes to the primary language, change the Locale property for any object that uses a language other than the form's language.

To set the Locale property of a form:

1. Choose File > Form Properties and select the Default tab
2. Select the appropriate language for the Form Locale (see Figure 17)
3. Click OK

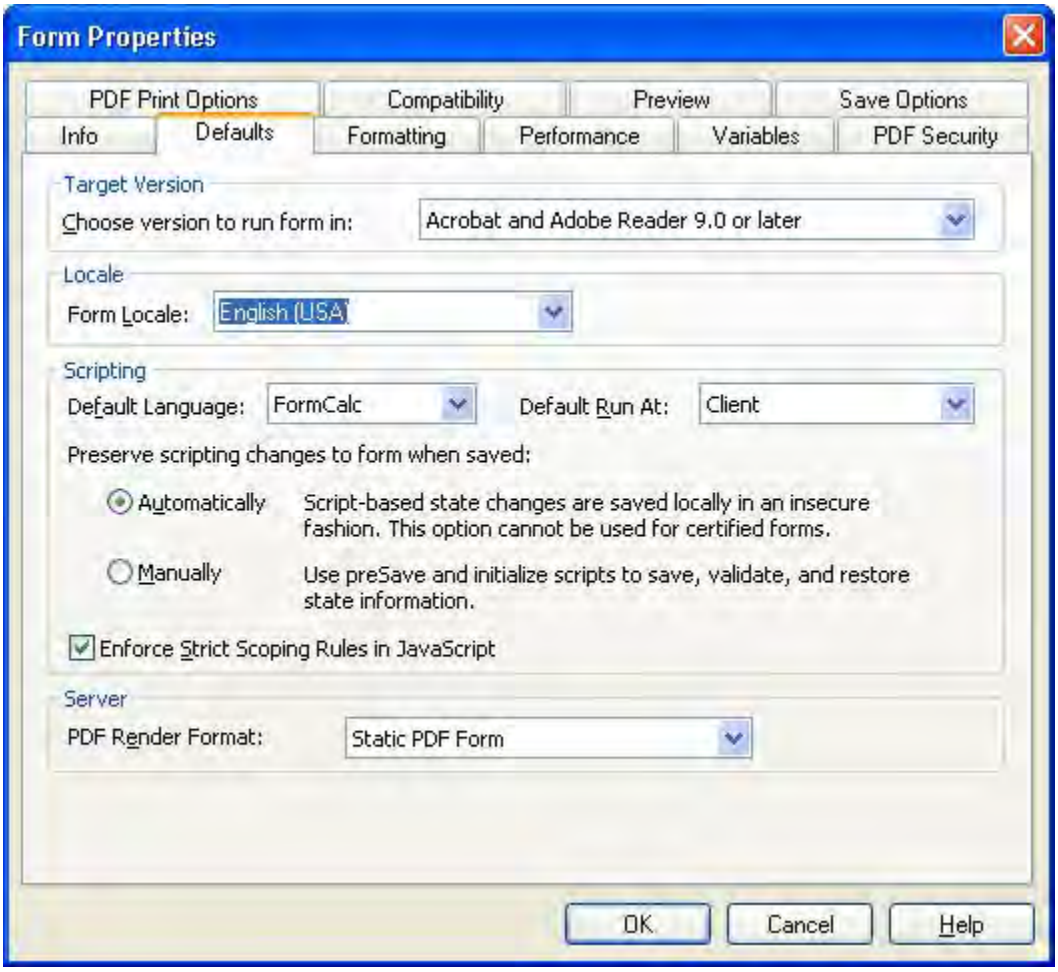

Figure 17: Changing the Form Locale on the Form Properties dialog box

To set Local property of the top-level subform or an object that requires a different language:

1. Select the top-level subform or object in design view
2. Display the Object palette by choosing Window > Object
3. In the Object palette, select the Field tab, and in the Locale list select the language to be used for the object (see Figure 18). When applying different locale options to individual objects, keep in mind that the objects that are within tables and subforms automatically receive the same locale setting as the table and subform object.

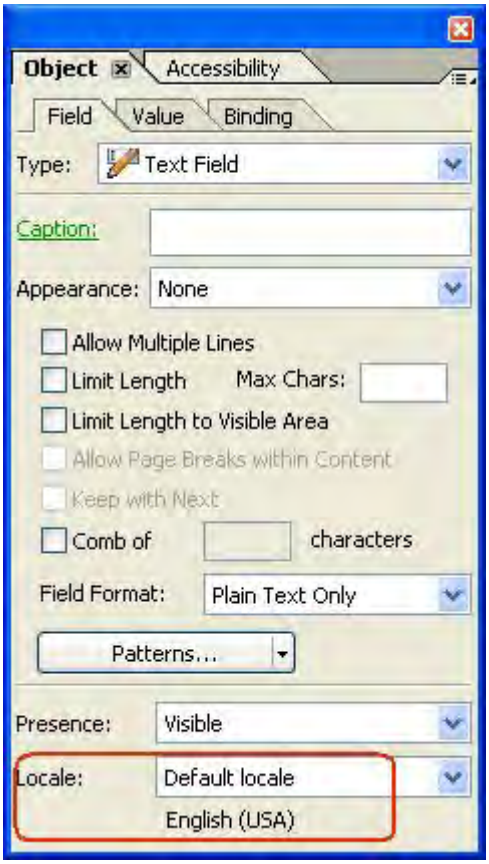

Figure 18: Changing an object's locale

Related checkpoints:

- WCAG 1.0
    - 4.1 Clearly identify changes in the natural language of a document's text and any text equivalents (e.g., captions).
- WCAG 2.0
    - 3.1.1 Language of Page: The default human language of each Web page can be programmatically determined. (Level A)
    - 3.1.2 Language of Parts: The human language of each passage or phrase in the content can be programmatically determined except for proper names, technical terms, words of indeterminate language, and words or phrases that have become part of the vernacular of the immediately surrounding text. (Level AA)

3.0 Techniques for testing form accessibility

To ensure that your forms are accessible to a wide variety of users, you should test them with a variety of assistive technologies. You can test your forms simply and inexpensively using the techniques described in this section.

Ensure that the form can be filled using only the keyboard. Be sure to fill the entire form and test all fields and buttons. As you complete the form, determine whether improvements are required based on your answers to the following questions:

- Are there any operations that cannot be performed?

- Are any operations awkward or difficult to perform?
- Are keyboard mechanisms well-documented?
- Do all controls and menu items have underlined access keys?

Demo versions of screen reader software can be downloaded free via the Internet. To test screen reader results, turn your monitor off and use only the screen reader to navigate and fill the form. If you are the form author, your familiarity with the form may make it difficult to determine if the information read by the screen reader is sufficient and makes sense. If possible, have someone else test your form in this way.

Demo versions of screen magnification software are also available for testing from the Internet.

Speech-to-text software, available at a nominal cost, can be used to test the form by using voice input only.

Many users with vision impairment rely on high contrast between the text and the background to read the form. Microsoft Windows has a high contrast color scheme that provides a display similar to what many users with vision impairment will be using to complete your form. To set your display to high contrast mode, enable the feature through Accessibility Options in the Windows Control Panel. As you complete the form in this mode, determine whether improvements are required based on your answers to the following questions:

- Do parts of the form become invisible, unrecognizable, or difficult to use?
- Do any areas continue to appear black on a white background?
- Are any elements improperly sized or truncated?

4.0 Mapping between guidelines and best practices

The following sections map Section 508 and WCAG guidelines to the best practices described in this guide.

4.1 Section 508 §1194.21: Software applications and operating systems.

§ 1194.21 guideline	Guideline Description	Required LiveCycle Designer Best Practices for Compliance	Notes
(a)	When software is designed to run on a system that has a keyboard, product functions shall be executable from a keyboard where the function itself or the result of performing a function can be discerned textually.	2.7 Ensure form controls are keyboard accessible 2.6 Ensure the reading and tab order are correct	

§ 1194.21 guideline	Guideline Description	Required LiveCycle Designer Best Practices for Compliance	Notes
(b)	Applications shall not disrupt or disable activated features of other products that are identified as accessibility features, where those features are developed and documented according to industry standards. Applications also shall not disrupt or disable activated features of any operating system that are identified as accessibility features where the application programming interface for those accessibility features has been documented by the manufacturer of the operating system and is available to the product developer.	No LiveCycle Designer specific techniques – this guideline is handled by Adobe Reader for PDF forms.	
(c)	A well-defined on-screen indication of the current focus shall be provided that moves among interactive interface elements as the input focus changes. The focus shall be programmatically exposed so that assistive technology can track focus and focus changes.	2.3 Choose the right controls	To ensure the focus is exposed programmatically as well as visually, always use the standard controls.
(d)	Sufficient information about a user interface element including the identity, operation and state of the element shall be available to assistive technology. When an image represents a program element, the information conveyed by the image must also be available in text.	2.1 Keep forms simple and easy to use 2.1.1 Avoid moving, blinking, or flashing content 2.2 Configure form properties to generate accessibility information 2.5 Provide proper labels for form controls	

§ 1194.21 guideline	Guideline Description	Required LiveCycle Designer Best Practices for Compliance	Notes
(e)	When bitmap images are used to identify controls, status indicators, or other programmatic elements, the meaning assigned to those images shall be consistent throughout an application's performance.	2.4 Provide text equivalents for images 2.5 Provide proper labels for form controls	This standard only applies if you use the same image in multiple places on a form. The use of image based custom controls is not recommended. Instead, use only standard controls provided by LiveCycle Designer, If you do use images in your controls, always make sure they are used consistently.
(f)	Textual information shall be provided through operating system functions for displaying text. The minimum information that shall be made available is text content, text input caret location, and text attributes.	2.3 Choose the right controls	Avoid using images to convey textual information. Rather than using custom input components which might not expose text properties properly to the operating system, always use the standard controls.
(g)	Applications shall not override user selected contrast and color selections and other individual display attributes.	No LiveCycle Designer specific techniques	Where possible, use the basic, default system colors.
(h)	When animation is displayed, the information shall be displayable in at least one non-animated presentation mode at the option of the user.	2.1 Keep forms simple and easy to use	Avoid using animations in your forms, or provide separate versions in which animations are replaced with static images.
(i)	Color coding shall not be used as the only means of conveying information, indicating an action, prompting a response, or distinguishing a visual element.	2.8 Use color responsibly	

§ 1194.21 guideline	Guideline Description	Required LiveCycle Designer Best Practices for Compliance	Notes
(j)	When a product permits a user to adjust color and contrast settings, a variety of color selections capable of producing a range of contrast levels shall be provided.	Not applicable	This functionality is generally provided by Adobe Reader, not by the form developer.
(k)	Software shall not use flashing or blinking text, objects, or other elements having a flash or blink frequency greater than 2 Hz and lower than 55 Hz.	2.1.1 Avoid moving, blinking, or flashing content	
(l)	When electronic forms are used, the form shall allow people using assistive technology to access the information, field elements, and functionality required for completion and submission of the form, including all directions and cues.	2.5 Provide proper labels for form controls	

4.2 Section 508 §1194.22: Web-based intranet and internet information and applications.

§11942 Guideline	Guideline Description	Required LiveCycle Designer Best Practices for Compliance	Notes
(a)	A text equivalent for every non-text element shall be provided (e.g., via "alt", "longdesc", or in element content).	2.4 Provide text equivalents for images	
(b)	Equivalent alternatives for any multimedia presentation shall be synchronized with the presentation.	2.12 Ensure all multimedia content is accessible	
(c)	Web pages shall be designed so that all information conveyed with color is also available without color, for example from context or markup.	2.8 Use color responsibly	
(d)	Documents shall be organized so they are readable without requiring an associated style sheet.	Not applicable	
(e)	Redundant text links shall be provided for each active region of a server-side image map.	Not applicable	
(f)	Client-side image maps shall be provided instead of server-side image maps except where the regions cannot be defined with an available geometric shape.	Not applicable	
(g)	Row and column headers shall be identified for data tables.	2.9 Provide heading cells for tables	
(h)	Markup shall be used to associate data cells and header cells for data tables that have two or more logical levels of row or column headers.	2.9 Provide heading cells for tables	
(i)	Frames shall be titled with text that facilitates frame identification and navigation.	Not applicable	
(j)	Pages shall be designed to avoid causing the screen to flicker with a frequency greater than 2 Hz and lower than 55 Hz.	2.1 Keep forms simple and easy to use	

§11942 Guideline	Guideline Description	Required LiveCycle Designer Best Practices for Compliance	Notes
(k)	A text-only page, with equivalent information or functionality, shall be provided to make a web site comply with the provisions of this part, when compliance cannot be accomplished in any other way. The content of the text-only page shall be updated whenever the primary page changes.	Not applicable	
(l)	When pages utilize scripting languages to display content, or to create interface elements, the information provided by the script shall be identified with functional text that can be read by assistive technology.	2.11 Avoid disruptive scripting	
(m)	When a web page requires that an applet, plug-in or other application be present on the client system to interpret page content, the page must provide a link to a plug-in or applet that complies with §1194.21(a) through (l).	Not applicable	Web pages linking to PDF forms should provide a link to Adobe Reader.
(n)	When electronic forms are designed to be completed on-line, the form shall allow people using assistive technology to access the information, field elements, and functionality required for completion and submission of the form, including all directions and cues.	2.5 Provide proper labels for form controls	
(o)	A method shall be provided that permits users to skip repetitive navigation links.	2.10 Provide a navigable form structure	
(p)	When a timed response is required, the user shall be alerted and given sufficient time to indicate more time is required.	2.11 Avoid disruptive scripting	

4.3 WCAG 1.0 Priority 1 checkpoints

Priority 1 Checkpoint	Checkpoint Description	Required LiveCycle Designer Best Practices for Compliance	Notes
1.1	Provide a text equivalent for every non-text element (e.g., via "alt", "longdesc", or in element content). *This includes*: images, graphical representations of text (including symbols), image map regions, animations (e.g., animated GIFs), applets and programmatic objects, ASCII art, frames, scripts, images used as list bullets, spacers, graphical buttons, sounds (played with or without user interaction), stand-alone audio files, audio tracks of video, and video.	2.4 Provide text equivalents for images 2.12 Ensure all multimedia content is accessible	
1.2	Provide redundant text links for each active region of a server-side image map.	Not applicable	
1.3	Until user agents can automatically read aloud the text equivalent of a visual track, provide an auditory description of the important information of the visual track of a multimedia presentation.	2.12 Ensure all multimedia content is accessible	
1.4	For any time-based multimedia presentation (e.g., a movie or animation), synchronize equivalent alternatives (e.g., captions or auditory descriptions of the visual track) with the presentation.	2.12 Ensure all multimedia content is accessible	

Priority 1 Checkpoint	Checkpoint Description	Required LiveCycle Designer Best Practices for Compliance	Notes
2.1	Ensure that all information conveyed with color is also available without color, for example from context or markup.	2.8 Use color responsibly	
4.1	Clearly identify changes in the natural language of a document's text and any text equivalents (e.g., captions).	2.13 Identify changes in language	
5.1	For data tables, identify row and column headers.	2.9 Provide heading cells for tables	
5.2	For data tables that have two or more logical levels of row or column headers, use markup to associate data cells and header cells.	2.9 Provide heading cells for tables	
6.1	Organize documents so they may be read without style sheets. For example, when an HTML document is rendered without associated style sheets, it must still be possible to read the document.	Not applicable	
6.2	Ensure that equivalents for dynamic content are updated when the dynamic content changes.	2.11 Avoid disruptive scripting	
6.3	Ensure that pages are usable when scripts, applets, or other programmatic objects are turned off or not supported. If this is not possible, provide equivalent information on an alternative accessible page.	2.11 Avoid disruptive scripting	

Priority 1 Checkpoint	Checkpoint Description	Required LiveCycle Designer Best Practices for Compliance	Notes
7.1	Until user agents allow users to control flickering, avoid causing the screen to flicker.	2.1 Keep forms simple and easy to use	
9.1	Provide client-side image maps instead of server-side image maps except where the regions cannot be defined with an available geometric shape.	Not applicable	
11.4	If, after best efforts, you cannot create an accessible page, provide a link to an alternative page that uses W3C technologies, is accessible, has equivalent information (or functionality), and is updated as often as the inaccessible (original) page.	Not applicable	
12.1	Title each frame to facilitate frame identification and navigation.	Not applicable	
14.1	Use the clearest and simplest language appropriate for a site's content.	2.1 Keep forms simple and easy to use	

4.4 WCAG 1.0 Priority 2 checkpoints

Priority 2 Checkpoint	Checkpoint Description	Required LiveCycle Best Practices for Compliance	Notes

Priority 2 Checkpoint	Checkpoint Description	Required LiveCycle Best Practices for Compliance	Notes
2.2	Ensure that foreground and background color combinations provide sufficient contrast when viewed by someone having color deficits or when viewed on a black and white screen. [Priority 2 for images, Priority 3 for text].	2.8 Use color responsibly	
3.1	When an appropriate markup language exists, use markup rather than images to convey information.	2.1 Keep forms simple and easy to use 2.1.1 Avoid moving, blinking, or flashing content 2.2 Configure form properties to generate accessibility information	Always use actual text rather than images of text.
3.2	Create documents that validate to published formal grammars.		PDF forms must match the published PDF specification in order to render in Adobe Reader.
3.3	Use style sheets to control layout and presentation.	Not applicable	
3.4	Use relative rather than absolute units in markup language attribute values and style sheet property values.	Not applicable	
3.5	Use header elements to convey document structure and use them according to specification.	2.10 Provide a navigable form structure	
3.6	Mark up lists and list items properly.	2.10.3 Marking up lists	Mark up list-based content as lists using the List and List Item roles.

Priority 2 Checkpoint	Checkpoint Description	Required LiveCycle Best Practices for Compliance	Notes
3.7	Mark up quotations. Do not use quotation markup for formatting effects such as indentation.	Not applicable	
5.3	Do not use tables for layout unless the table makes sense when linearized. Otherwise, if the table does not make sense, provide an alternative equivalent (which may be a linearized version).	No specific LiveCycle techniques	There is no reason to use tables for layout in LiveCycle forms. Instead, use the Layout palette to position the form fields in a grid pattern. Only use a table when utilizing table specific features such as table headers.
5.4	If a table is used for layout, do not use any structural markup for the purpose of visual formatting.	No specific LiveCycle techniques	
6.4	For scripts and applets, ensure that event handlers are input device-independent.	2.7 Ensure form controls are keyboard accessible	
6.5	Ensure that dynamic content is accessible or provide an alternative presentation or page.	2.11 Avoid disruptive scripting	
7.2	Until user agents allow users to control blinking, avoid causing content to blink (i.e., change presentation at a regular rate, such as turning on and off).	2.1 Keep forms simple and easy to use	
7.3	Until user agents allow users to freeze moving content, avoid movement in pages.	2.1 Keep forms simple and easy to use	

Priority 2 Checkpoint	Checkpoint Description	Required LiveCycle Best Practices for Compliance	Notes
7.4	Until user agents provide the ability to stop the refresh, do not create periodically auto-refreshing pages.	Not applicable	
7.5	Until user agents provide the ability to stop auto-redirect, do not use markup to redirect pages automatically. Instead, configure the server to perform redirects.	Not applicable	
8.1	Make programmatic elements such as scripts and applets directly accessible or compatible with assistive technologies [Priority 1 if functionality is important and not presented elsewhere, otherwise Priority 2.]	2.11 Avoid disruptive scripting	
9.2	Ensure that any element that has its own interface can be operated in a device-independent manner.	2.7 Ensure form controls are keyboard accessible	
9.3	For scripts, specify logical event handlers rather than device-dependent event handlers.	2.7 Ensure form controls are keyboard accessible	
10.1	Until user agents allow users to turn off spawned windows, do not cause pop-ups or other windows to appear and do not change the current window without informing the user.	2.11 Avoid disruptive scripting	

Priority 2 Checkpoint	Checkpoint Description	Required LiveCycle Best Practices for Compliance	Notes
10.2	Until user agents support explicit associations between labels and form controls, for all form controls with implicitly associated labels, ensure that the label is properly positioned.	2.5 Provide proper labels for form controls	
11.1	Use W3C technologies when they are available and appropriate for a task and use the latest versions when supported.	Not applicable	
11.2	Avoid deprecated features of W3C technologies.	Not applicable	
12.2	Describe the purpose of frames and how frames relate to each other if it is not obvious by frame titles alone.	Not applicable	
12.3	Divide large blocks of information into more manageable groups where natural and appropriate.	2.10 Provide a navigable form structure	
12.4	Associate labels explicitly with their controls.	2.5 Provide proper labels for form controls	
13.1	Clearly identify the target of each link.	2.5 Provide proper labels for form controls 2.5.6 Providing link text	
13.2	Provide metadata to add semantic information to pages and sites.	Not applicable	

Priority 2 Checkpoint	Checkpoint Description	Required LiveCycle Best Practices for Compliance	Notes
13.3	Provide information about the general layout of a site (e.g., a site map or table of contents).	2.10 Provide a navigable form structure	
13.4	Use navigation mechanisms in a consistent manner.	2.10 Provide a navigable form structure	Use master pages to create consistent navigation content.

4.4 WCAG 2.0 Success Criteria

Priority 1 & 2 Checkpoints	Required LiveCycle Best Practices for Compliance	Notes
1.1 [Text Alternatives]		
1.1.1 [Non-text Content]	2.4 Provide text equivalents for images 2.5 Provide proper labels for form controls	
1.2 [Time-based Media]		
1.2.1 [Audio-only and Video-only (Prerecorded)]	2.12 Ensure all audio and video content is accessible	
1.2.2 [Captions (Prerecorded)]	2.12 Ensure all audio and video content is accessible	
1.2.3 [Audio Description or Media Alternative (Prerecorded)]	2.12 Ensure all audio and video content is accessible	
1.2.4 [Captions (Live)]	2.12 Ensure all audio and video content is accessible	
1.2.5 [Audio Description (Prerecorded)]	2.12 Ensure all audio and video content is accessible	

Priority 1 & 2 Checkpoints	Required LiveCycle Best Practices for Compliance	Notes
1.2.6 [Sign Language (Prerecorded)]	2.12 Ensure all audio and video content is accessible	
1.2.7 [Extended Audio Description (Prerecorded)]	2.12 Ensure all audio and video content is accessible	
1.2.8 [Media Alternative (Prerecorded)]	2.12 Ensure all audio and video content is accessible	
1.2.9 [Audio-only (Live)]	2.12 Ensure all audio and video content is accessible	
1.3 [Adaptable]		
1.3.1 [Info and Relationships]	2.9 Provide heading cells for tables	
1.3.2 [Meaningful Sequence]	2.6 Ensure the reading and tab order are correct 2.10 Provide a navigable form structure	
1.3.3 [Sensory Characteristics]	2.8 Use color responsibly	
1.4 [Distinguishable]		
1.4.1 [Use of Color]	2.8 Use color responsibly	
1.4.2 [Audio Control]	No specific LiveCycle techniques	
1.4.3 [Contrast (Minimum)]	2.8 Use color responsibly	
1.4.4 [Resize text]	No specific LiveCycle techniques	
1.4.5 [Images of Text]	No specific LiveCycle techniques	
1.4.6 [Contrast (Enhanced)]	2.8 Use color responsibly	

Priority 1 & 2 Checkpoints	Required LiveCycle Best Practices for Compliance	Notes
1.4.7 [Low or No Background Audio]	No specific LiveCycle techniques	
1.4.9 [Images of Text (No Exception)]	No specific LiveCycle techniques	
2.1 [Keyboard Accessible]		
2.1.1 [Keyboard]	2.6 Ensure the reading and tab order are correct 2.7 Ensure form controls are keyboard accessible	
2.1.2 [No Keyboard Trap]	2.7 Ensure form controls are keyboard accessible	
2.1.3 [Keyboard (No Exception)]	2.6 Ensure the reading and tab order are correct 2.7 Ensure form controls are keyboard accessible	
2.2 [Enough Time]		
2.2.1 [Timing Adjustable]	No specific LiveCycle techniques	
2.2.2 [Pause, Stop, Hide]	2.1 Keep forms simple and easy to use	
2.2.3 [No Timing]	No specific LiveCycle techniques	
2.2.4 [Interruptions]	No specific LiveCycle techniques	
2.2.5 [Re-authenticating]	No specific LiveCycle techniques	
2.3 [Seizures]		
2.3.1 [Three Flashes or Below Threshold]	2.1 Keep forms simple and easy to use	
2.3.2 [Three Flashes]	2.1 Keep forms simple and easy to use	

Priority 1 & 2 Checkpoints	Required LiveCycle Best Practices for Compliance	Notes
2.4 [Navigable]		
2.4.1 [Bypass Blocks]	2.10 Provide a navigable form structure	
2.4.2 [Page Titled]	No specific LiveCycle techniques	
2.4.3 [Focus Order]	2.6 Ensure the reading and tab order are correct	
2.4.4 [Link Purpose (In Context)]	No specific LiveCycle techniques	Link purpose is dependent on authors choosing meaningful text for linked elements.
2.4.5 [Multiple Ways]	2.10 Provide a navigable form structure	
2.4.6 [Headings and Labels]	2.5 Provide proper labels for form controls 2.10 Provide a navigable form structure	
2.4.7 [Focus Visible]	No specific LiveCycle techniques	The default focus in LiveCycle forms is visible.
2.4.8 [Location]	No specific LiveCycle techniques	Not applicable: LiveCycle forms do not require navigation systems.
2.4.9 [Link Purpose (Link Only)]	No specific LiveCycle techniques	Link purpose is dependent on authors choosing meaningful text for linked elements.
2.4.10 [Section Headings]	2.10 Provide a navigable form structure	
3.1 [Readable]		
3.1.1 [Language of Page]	2.13 Identify natural language and any changes in language	
3.1.2 [Language of Parts]	2.13 Identify natural language and any changes in language	

Priority 1 & 2 Checkpoints	Required LiveCycle Best Practices for Compliance	Notes
3.1.3 [Unusual Words]	No specific LiveCycle techniques	
3.1.4 [Abbreviations]	No specific LiveCycle techniques	
3.1.5 [Reading Level]	No specific LiveCycle techniques	
3.1.6 [Pronunciation]	No specific LiveCycle techniques	
3.2 [Predictable]		
3.2.1 [On Focus]	2.11 Avoid disruptive scripting	
3.2.2 [On Input]	2.11 Avoid disruptive scripting	
3.2.3 [Consistent Navigation]	2.10 Provide a navigable form structure	
3.2.4 [Consistent Identification]	2.3 Choose the right controls 2.5 Provide proper labels for form controls	
3.2.5 [Change on Request]	2.11 Avoid disruptive scripting	
3.3 [Input Assistance]		
3.3.1 [Error Identification]		LiveCycle Designer provides tools to mark form fields as required and to perform form input validation.
3.3.2 [Labels or Instructions]	2.5 Provide proper labels for form controls	
3.3.3 [Error Suggestion]		LiveCycle Designer provides tools to mark form fields as required and to perform form input validation.

Priority 1 & 2 Checkpoints	Required LiveCycle Best Practices for Compliance	Notes
3.3.4 [Error Prevention (Legal, Financial, Data)]	No specific LiveCycle techniques	
3.3.5 [Help]	No specific LiveCycle techniques	
3.3.6 [Error Prevention (All)]	No specific LiveCycle techniques	
4.1 [Compatible]		
4.1.1 [Parsing]	No specific LiveCycle techniques	
4.1.2 [Name, Role, Value]	2.3 Choose the right controls 2.5 Provide proper labels for form controls	

5.0 Useful links

- Adobe Accessibility Resource Center: www.adobe.com/accessibility
- 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/

Adobe Systems Incorporated

345 Park Avenue, San Jose, CA 95110-2704 USA

http://www.adobe.com

24 March 2012