Adobe

Best Practices for Creating Adobe Form Designs

Adobe® LiveCycle™ Designer

Version 7.1 and 8.0

March 27, 2007

Contents

Overview

This document outlines the best practices to follow when creating forms using the Adobe® LiveCycle™ forms technology.

Software components

The LiveCycle forms technology includes the following software components.

Name	Description
Adobe LiveCycle Designer	A graphical user interface (GUI) application for creating form designs in Adobe XML Data Package (XDP) format.
Application or database	An application or database that exposes data in XML format.
Adobe document services	An intelligent data-capturing and data-merging solution that combines a form design, XML data, and configuration settings. It then renders the form as an Adobe PDF, an Adobe PostScript®, a Printer Control Language (PCL), or a Zebra Programming Language (ZPL) file.

Steps to render forms

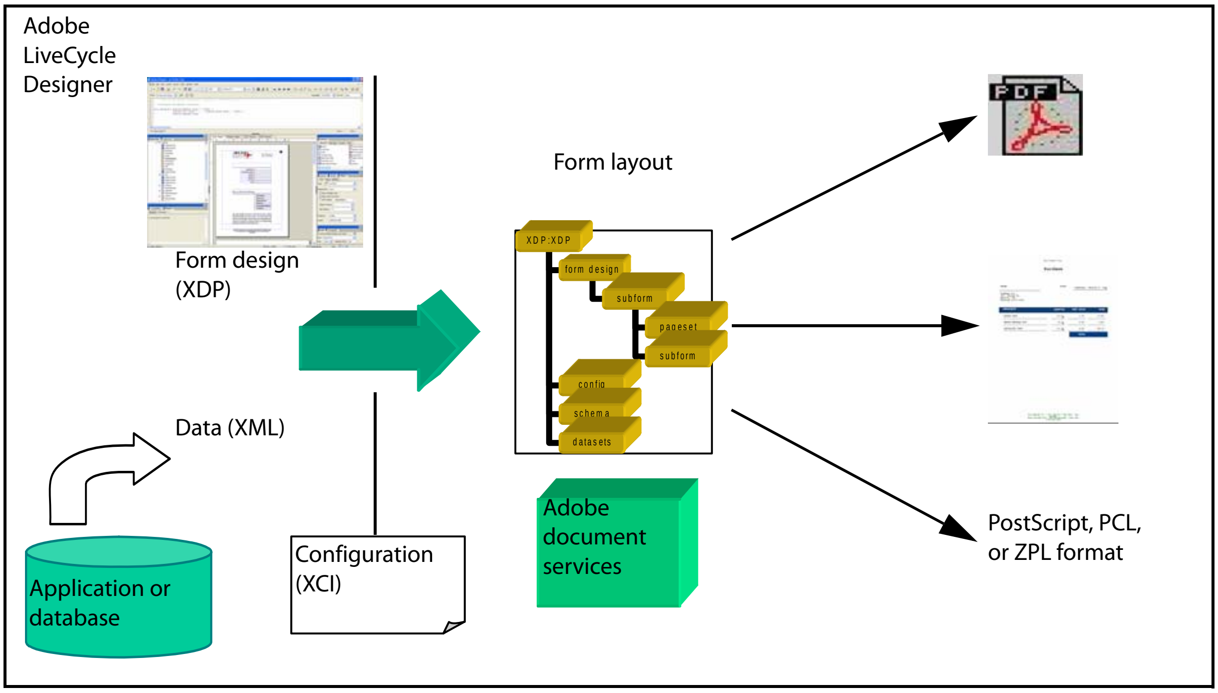
To render forms with the LiveCycle forms technology, follow these steps:

1. **Define the XML data:** Define the data structure that the application or database will expose and that the form design will use.

2. **Create the form design:** In LiveCycle Designer, drag fields to create the form design. The form design is stored on a server and used in the rendering process.

3. **Render the form:** Submit the form design (XDP), data (XML), and configuration settings (XCI) to Adobe document services. Adobe document services build an output independent version of the form, and then convert this form layout to PDF, PostScript, PCL, or ZPL format, as required.

As illustrated below, a form consists of a set of XML files that are combined by Adobe document services. It is not necessary to view or edit XML files to use the Adobe LiveCycle suite of products. However, it is essential to understand the operations that occur behind the scenes if you want to develop complex forms.



Attachments

This document includes attachments that provide detailed examples for some of the tasks described in this document. You can access the attachments on the Attachments tab. The following table summarizes where the attachments are used.

Task	**Related attachments**
"Create Accessible Multiline Tables" on page 27	multiline_output.pdf multiline_table.xdp multiline_table.xml
"Create Accessible Tables with Out-of-Place Rows" on page 44	out_of_place.pdf out_of_place.xdp out_of_place.xml
Create Accessible, Multiline, Heterogeneous Header Tablespage 39	mhh_table.xdp mhh_table.xml mhh_table.pdf.
Create Accessible Horizontal Tables	horz_tables.xdp horz_tables.xml horz_tables.pdf

Task	Related attachments
Preserving decimal digits when calculating	preserve_decimal_places.xdp preserve_decimal_places.pdf
Fitting a Large Graphic	fit_large_graphic.xdp fit_large_graphic.xml fit_large_graphic.pdf
Creating Multiple Levels of Repeating Headers	mlrh_data.xml mlrh_output.pdf mlrh_template.xdp
Creating a URL based read-only graphic	URL_readonly_graphic.xdp URL_readonly_graphic.pdf

You can open and save attachments in Adobe Acrobat® Professional, Acrobat Standard, or Adobe Reader®. However, you must have an application installed that can handle the file format of the attachment. Any changes you make to an attachment are not applied to the attachment. Instead, save a copy of the attachment, make the required changes to the file, and then reattach it to the PDF document.

‰ **To open an attachment in this PDF document:**

1. In the Attachments tab, select the attachment.
2. Click Open.

‰ **To save a copy of one or more attachments:**

1. In the Attachments tab, select one or more attachments.
2. Click Save.
3. Save the attachments.:
   - To save a single attachment, name the file, specify a location, and then click Save.
   - To save multiple attachments, specify a location, and then click OK.

Related Documentation

For additional information about LiveCycle Designer, refer to the following online resources:

Adobe website: The website contains documents that you can download, such as the *Adobe XML Forms Architecture (XFA) Specification*:

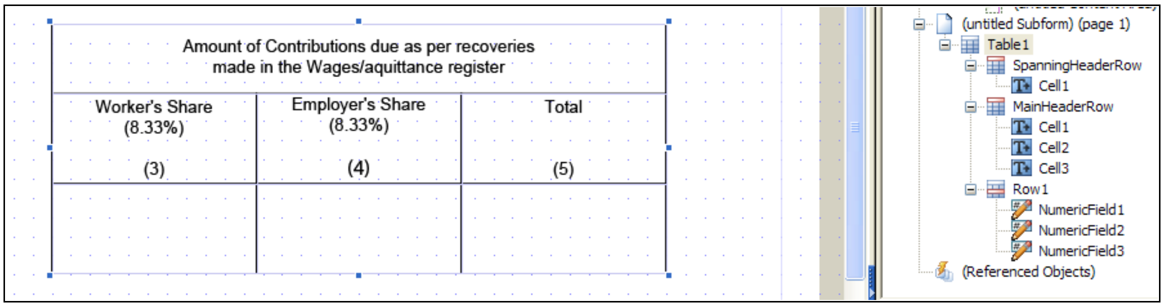
http://partners.adobe.com/public/developer/xml/index_arch.html

***LiveCycle Designer Help*:** *LiveCycle Designer Help* contains reference information and recommendations to follow when creating form designs.

Select Form Content in the Hierarchy Palette

Use the Hierarchy palette to select objects in a form design. Some objects in a form design are difficult to see without inspecting the Hierarchy palette. In form designs that use tables or nested subforms, adjacent objects have little discernible space between them, which makes it difficult to select an object. Using the Hierarchy palette makes it easier to select, resize, or copy objects when there is little space between them.

For example, in the table illustrated below, use the Hierarchy palette instead of selecting the table row in the form design. To add another header row, select the SpanningHeaderRow object in the Hierarchy palette, and then right-click to insert the row.



Amount of Contributions due as per recoveries made in the Wages/aquittance register		
Worker's Share (8.33%) (3)	Employer's Share (8.33%) (4)	Total (5)

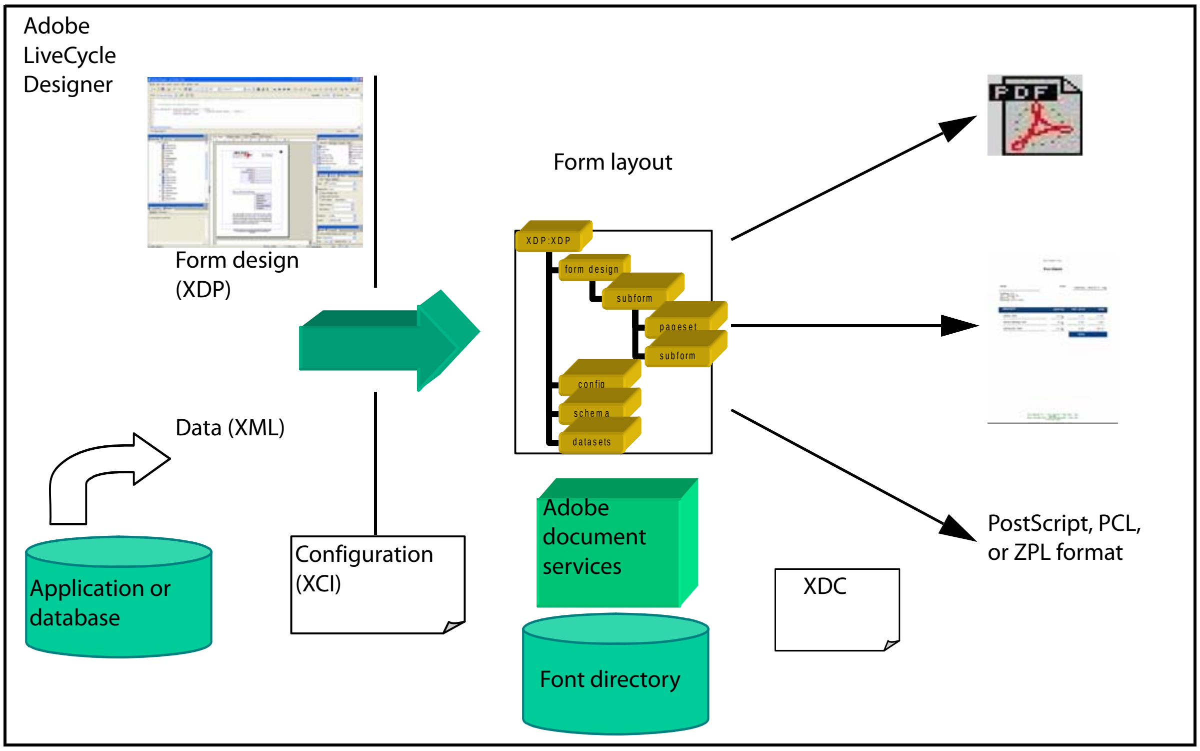
Choose Fonts That Ensure Accurate Output and Minimal Output Size

Font selection

Adobe document services follow these steps when a font is used in a form design and rendered in a form:

1. **The form developer chooses the fonts:** LiveCycle Designer displays the list of fonts that are installed on the form developer's computer. All of the fonts look the same on-screen and when printed.

2. **Adobe document services render the form design:** Adobe document services use an XML device control (XDC) file to determine which fonts are supported for a particular output device, such as Acrobat, PostScript, or a PCL device. The font-mapping table, Designer.xci, lists the font-substitution rules.

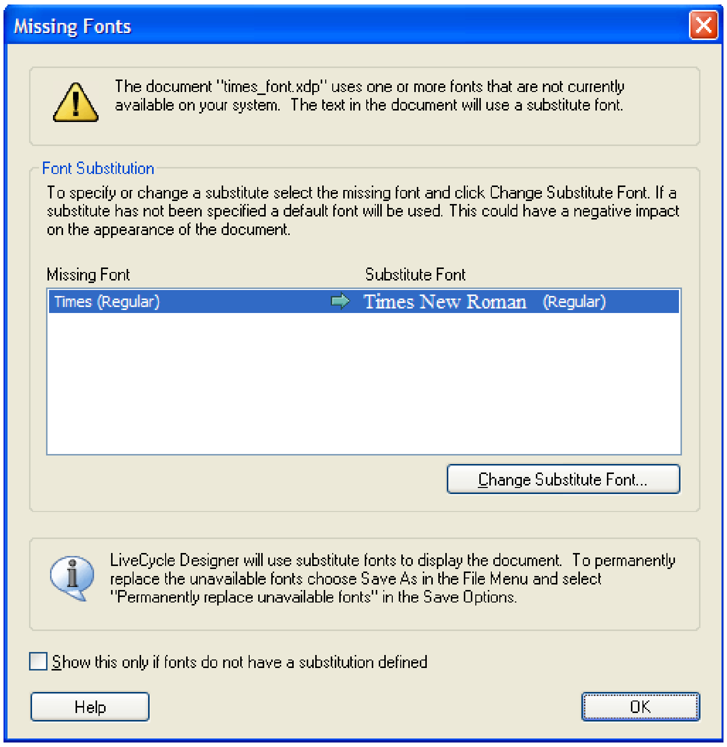


3. **Adobe document services embed fonts in the output stream, as required:** If an output device does not support a particular font, Adobe document services can embed all or part of the font as part of the output stream. A setting in the font-mapping file (XCI) controls the ability to embed a font.

Font substitution

Using the Missing Fonts dialog box, you can select substitute fonts for form designs that use fonts that are not available on your computer. Use the Missing Fonts dialog box to select a substitute font for the unavailable fonts. After you select the substitute fonts or accept the default fonts for a document, the document is displayed using those fonts.

In the following example, the form developer used the Times font to render a form on a PostScript printer that contains the Times font (Times is a printer-resident font). In some cases, a form must use a particular font for legal reasons. Because LiveCycle Designer cannot accurately display the Times font, it displays this dialog box and prompts the user to select a substitute font. The form design is not modified in this case.


Missing Fonts

The document "times_font.xdp" uses one or more fonts that are not currently available on your system. The text in the document will use a substitute font.

Font Substitution

To specify or change a substitute select the missing font and click Change Substitute Font. If a substitute has not been specified a default font will be used. This could have a negative impact on the appearance of the document.

Missing Font	Substitute Font
Times (Regular)	Times New Roman (Regular)

LiveCycle Designer will use substitute fonts to display the document. To permanently replace the unavailable fonts choose Save As in the File Menu and select "Permanently replace unavailable fonts" in the Save Options.

[ ] Show this only if fonts do not have a substitution defined

Fonts that ensure output device support

For print forms, use fonts that all the output devices support to ensure that the output device can render a form design as intended. The following fonts are supported by Adobe document services for PDF and most PCL, and PostScript output devices:

- Courier
- Arial
- Times New Roman

Helvetica and Times fonts are not part of the recommended list but will render on a generic PDF or older PostScript printers. However these fonts are not installed on a desktop PC and therefore cannot be previewed in Designer. If a legal form requires the use of the Times font that is resident on the printer, then the Times font can be specified by manually entering the name on the font field in Designer. However, the preview in Designer will not be accurate.

For interactive forms it is best to use fonts that are included with Adobe Reader. The best font choices are:

- Myriad Pro
- Minion Pro

For form designs that contain non-Latin-1 characters, use the Kazuko , Adobe Ming, Adobe Song, Adobe Heiti and MyungJo fonts for Asian characters or the Adobe Arabic, Adobe Thai, and Adobe Hebrew fonts. Notice that these fonts will perform well in Acrobat but will be sent as a embedded fonts to the printer.

If an output device does not support a particular font, Adobe document services can embed the font as part of the output stream. However, embedding fonts enlarges the size of the output stream. To minimize the size of the output stream, Adobe document services embed only the subset of the font that the rendered form uses. These settings are controlled within the font-mapping (XCI) file.

Font mapping on the server and client

It is possible to map one font to another on both the client machine, where for the form is being developed and on the server, when the form is rendered for use by the end user

On the client machine, in Designer, it is possible to map fonts using the dialog shown below. This dialog is available by clicking on Tools/Options, selecting “Document Handling” and clicking on the “Modify Font Substitutions...” button

If you map fonts, the mapping of these fonts only affects the output on the client machine. The contents of the template are not affected in any way.

If you wish to make font substitution on the server, then it is necessary to modify a text file on the server called CUSTOM_XFA.XCI. This file is located in the server in the following directory:

C:\usr\sap\<sapsid>\SYS\global\AdobeDocumentServices\lib (Windows)

/usr/sap/<sapsid>/SYS/global/AdobeDocumentServices/lib (UNIX)

where: <sapsid> is a three letter ID associated with your installation of the SAP server software.

Use Explicit Data Binding

If you want data to appear in a form design, you need to create a data binding. A *data binding* is an association between the objects in a form design and a node in a data source. It lets you build a form that captures data for enterprise infrastructures and use an external data source that populates a form at run time.

You can use either explicit or implicit data binding in a form design:

Implicit binding Adobe document services creates implicit bindings by matching the names of the objects in the form design to the corresponding nodes in the data file. This automatic data matching is the default behavior for objects that are added from the Library palette. It corresponds to the Normal option in the Default Binding list in the Binding tab of the Object palette. For implicit binding to work, the nodes in the data file and the fields on the form design must have the same names.

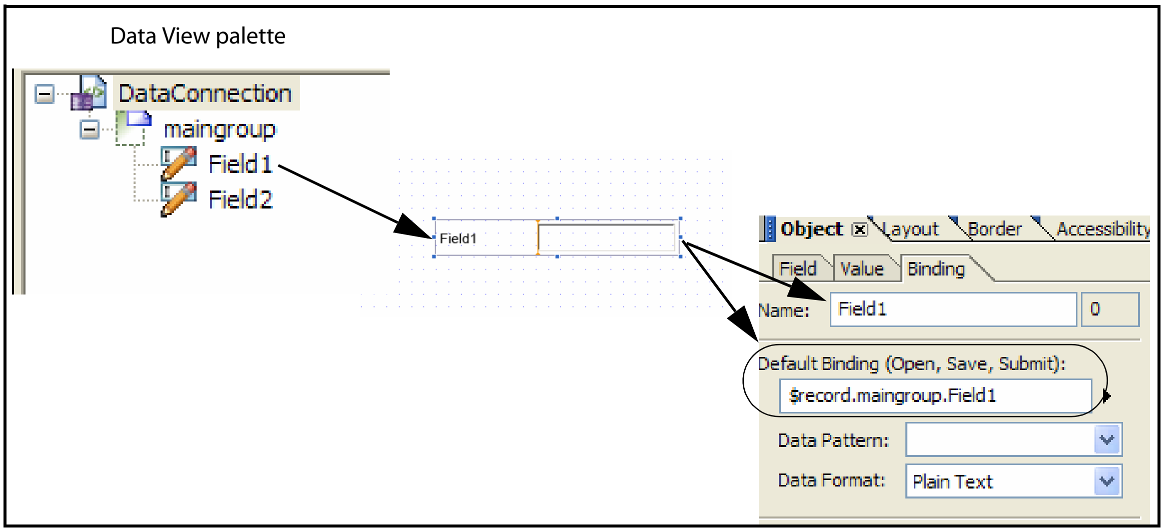
Explicit binding Adobe document services uses the specified data references, instead of object names, to map nodes in the data source to objects in the form design. Explicit binding overrides the automatic data matching and points directly to a data node.

Note: It is strongly recommended that you do not use both explicit and implicit binding in the same form design, particularly if the form design is interactive.

Absolute and relative binding expressions

You can create explicit bindings using absolute or relative binding expressions.

Absolute binding expressions An absolute binding expression is a fully qualified SOM expression describing a data node, which means that it starts from $record or $data. To create an explicit binding with an absolute SOM expression, drag an individual node from the Data View palette onto the form design. For example, if you drag the Field1 node from the Data View palette onto the page to create a new form object, the default binding value is $record.maingroup.Field1.



Name: Field1 0

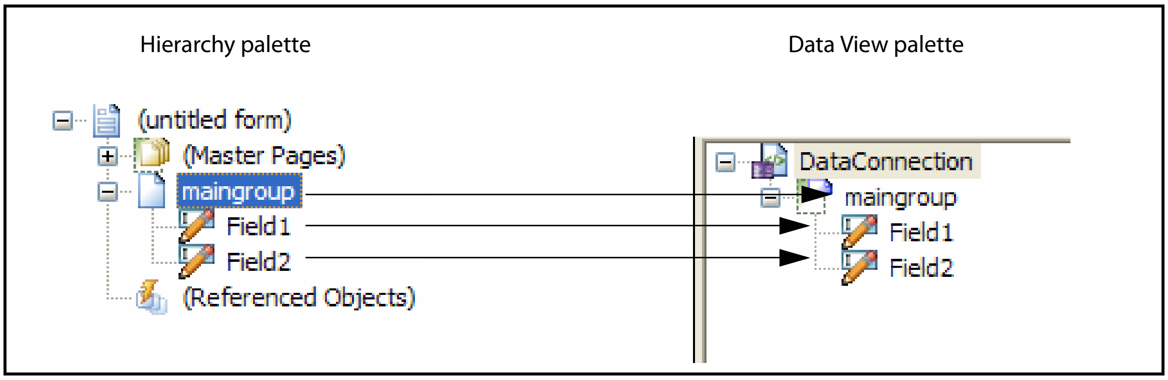
Default Binding (Open, Save, Submit): $record.maingroup.Field1

Data Pattern:

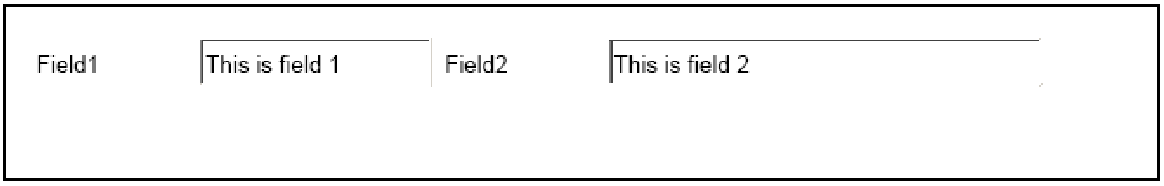
Data Format: Plain Text

Relative binding expressions A relative binding expression is a partial binding expression. It is evaluated relative to the data node that is bound to the containing object. To create an explicit binding with a relative binding expression, drag a parent node and its children from the Data View palette onto the form design. This creates an absolute binding expression for the parent object and relative binding expressions for its children.

As illustrated below, when you drag the maingroup node and its children onto the form design the binding value for the maingroup object is $record.maingroup. The children of the maingroup object automatically bind to the corresponding nodes in the data source.



The rendered form contains the values from the bound data source.



Field1	This is field 1	Field2	This is field 2

Use explicit binding and relative binding expressions

Use explicit binding instead of implicit binding whenever possible to ensure that the bound data in the form is accurate.

Use relative binding expressions, because they require less processing when Adobe document services merge the data with a form design. Absolute binding expressions specify each node in the path to the bound node, such as $record.group.field. For each absolute binding expression, Adobe document services must navigate the data hierarchy to locate the specified node, which is time consuming. Relative binding expressions specify only as much of the path as required to identify the node, such as field.

For information about how data is merged with form designs, see the *LiveCycle Designer Help* and the *Adobe XML Form Object Model Reference*.

Note: Avoid using a lot of subforms; they can degrade performance.

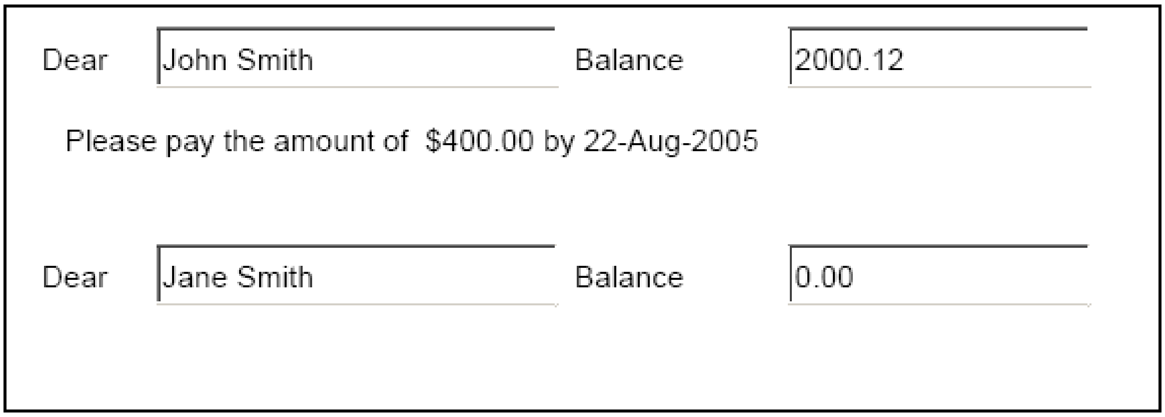
Use Repeating Subforms to Show or Hide Objects

You can use either repeating subforms or scripting to conditionally show or hide objects when a form is rendered. Use repeating subforms instead of scripting when you need to process a large number of records.

Scripting object presence

You can script with the `presence` property to show or hide an object based on some condition.

For example, on an invoice, if the customer does not have a balance owing, the rendered form should not display the statement about the amount to pay.



Dear	John Smith	Balance	2000.12

Please pay the amount of $400.00 by 22-Aug-2005

Dear	Jane Smith	Balance	0.00

In this example, you can use the following FormCalc script to check the value in the Balance field:

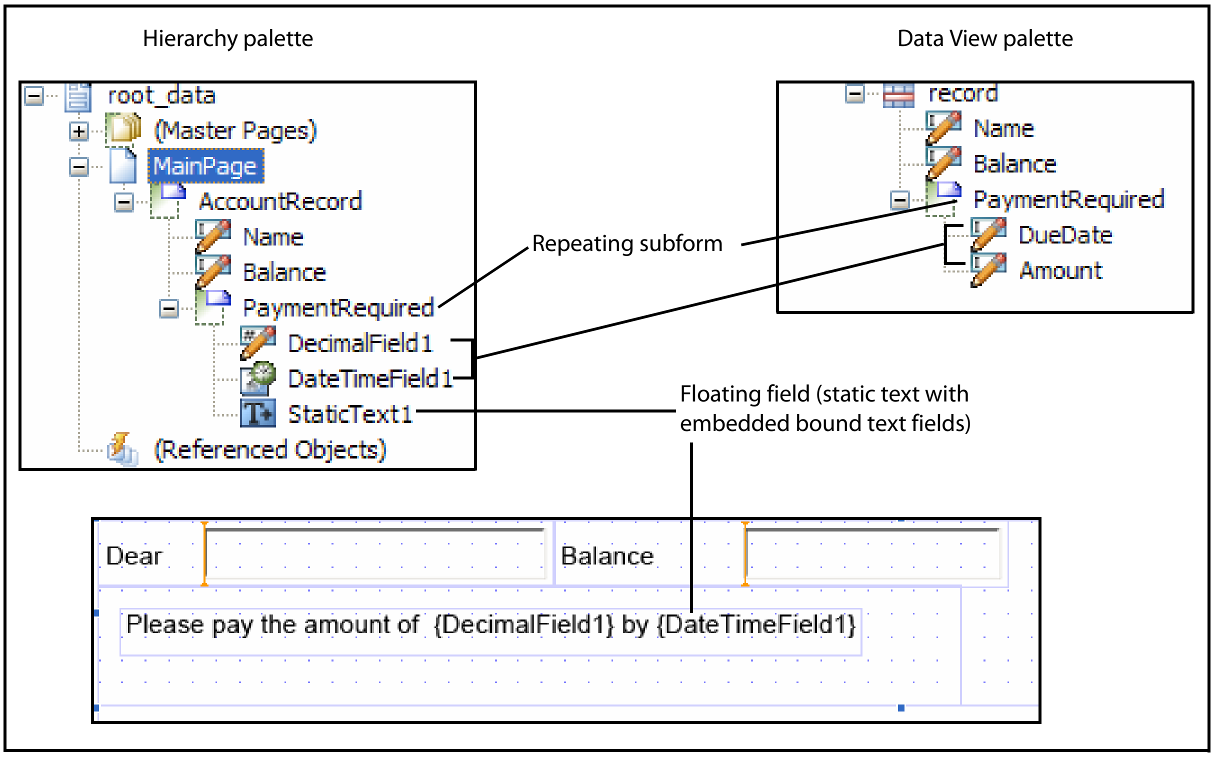
```
if (Balance.rawValue ne 0) then
   PaymentRequired.presence = "visible"
else
   PaymentRequired.presence = "invisible"
endif
```

Repeating subforms

If you want to show or hide objects on a form and you need to process a large quantity of records, it is more efficient to use repeating subforms than to use scripting. A subform can repeat the rendering of its objects as long as it is bound to a data source and nested inside a subform that flows content.

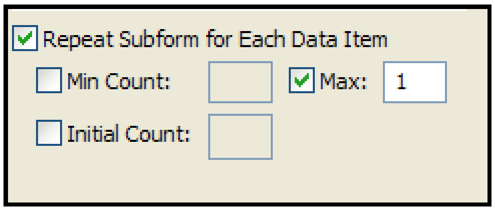
To use repeating subforms, you must set up the data structure to match the subform structure. In the data structure, nest the objects that you want to show or hide under a parent node and then bind the parent node to a repeating subform.

In the form design, the objects are bound to the corresponding data nodes. For example, in the form design for the invoice, the PaymentRequired object is a repeating subform that is bound to the PaymentRequired node in the data source. The DateTimeField object corresponds to the DueDate node and the DecimalField1 object corresponds to the Amount node. A static text object contains the statement about the balance owing.



Note: Floating fields are objects that support the merging of text, numeric values, run-time properties, and scripting within a text object when the form is rendered. You can insert floating fields into text objects only. You can bind floating fields to a data source to display specific text or numeric values.

In the Object palette, the Binding tab displays the Repeat Subform For Each Data Item option. Set the Min Count value to 0 or leave it blank, and set the Max value to 1 so that the subform repeats a minimum of zero times and a maximum of one time. If the Min Count value is set to 0 and no data is provided for the objects in the subform at data-merge time, the subform is hidden when the form is rendered.


[✓] Repeat Subform for Each Data Item
[ ] Min Count: [✓] Max: 1
[ ] Initial Count:

Use Dynamic Properties to Bind Data to a List

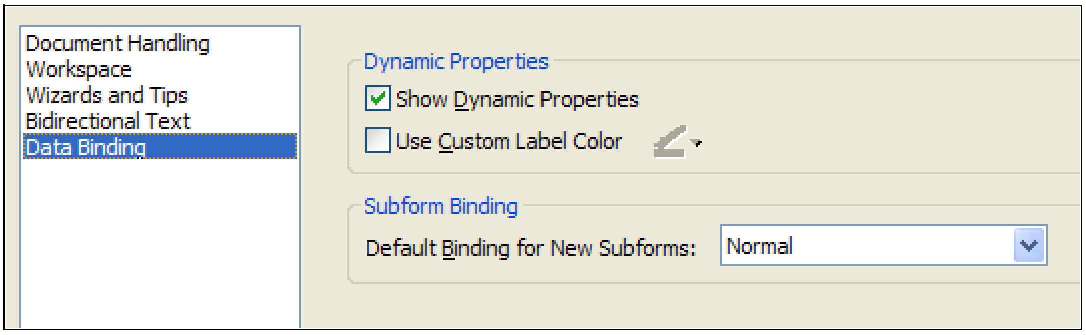
Use the Dynamic Properties feature to bind objects in an XML data stream to a list, such as a drop-down list. Using dynamic properties to populate drop-down lists is faster than using a script-based solution.

When creating form designs that use drop-down lists, structure the XML data so that a list of child nodes is displayed below a parent node. This data structure makes it easier to map the XML data to the drop-down list, as shown in this example:

```
<currencies>
   <Currency>USD</Currency>
   <Currency>GBP</Currency>
```

‰ **To bind data to a drop-down list:**

1. To enable Dynamic Properties in LiveCycle Designer, select Tools > Options > Data Binding.

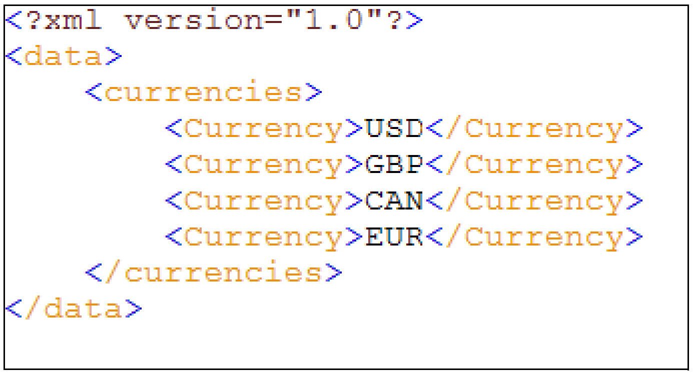


Document Handling
Workspace
Wizards and Tips
Bidirectional Text
Data Binding

Dynamic Properties
[✓] Show Dynamic Properties
[ ] Use Custom Label Color

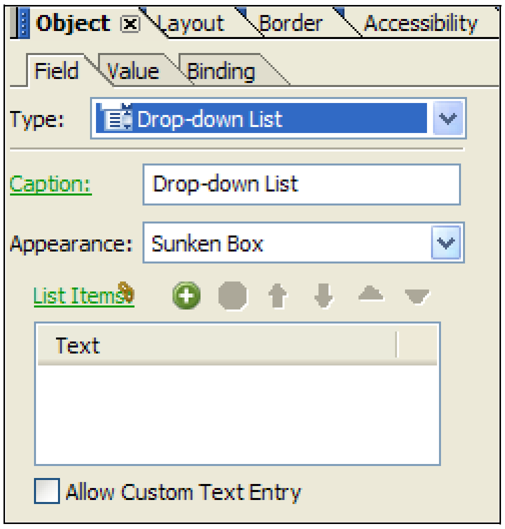
Subform Binding
Default Binding for New Subforms: Normal

2. To bind the following XML data to a drop-down list, select the drop-down list in the form design.

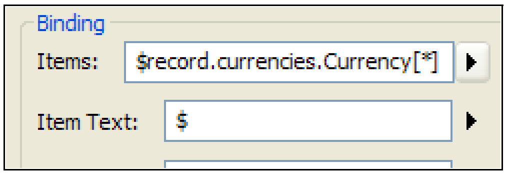


```
<?xml version="1.0"?>
<data>
    <currencies>
        <Currency>USD</Currency>
        <Currency>GBP</Currency>
        <Currency>CAN</Currency>
        <Currency>EUR</Currency>
    </currencies>
</data>
```

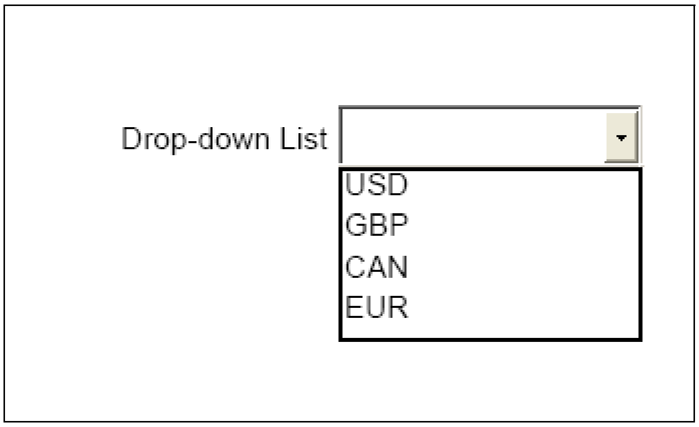
3. In the Object palette, click the Field tab and select List Items.



4. In the Dynamic Properties dialog box, select the following binding.


Binding
Items: $record.currencies.Currency[*]
Item Text: $

The resulting interactive form contains the list of items in the drop-down list.



Use Field Patterns for Universal Formatting

Use field patterns to ensure that field values can be displayed in all the required language locales.

Field patterns

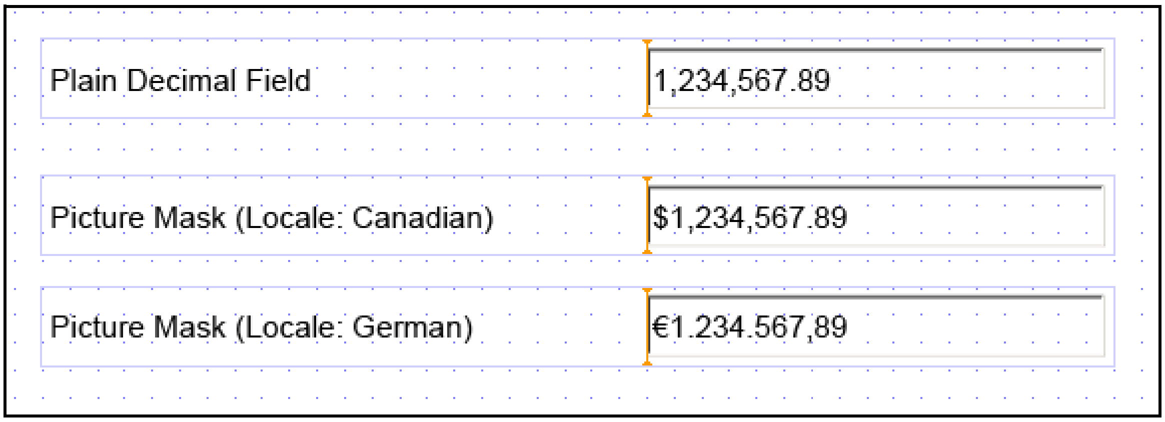
Field patterns are strings of characters that control how field values, such as text fields, numeric fields, and date/time fields, are formatted at run time:

- A display pattern describes how data will be displayed in the form. If you define an initial default value, it is formatted according to the display pattern. The display pattern is also responsible for formatting user input and any bound values retrieved at run time.
- An edit pattern describes the syntax for entering data into a date/time field, numeric field, text field, or password field at run time.
- A validation pattern validates user input at run time.
- A data pattern describes the syntax of bound or saved data.

Use field patterns only if the results of the pattern are correct for all the affected locales.

The field pattern syntax is locale independent and the rendered value is locale dependent. For example, if the decimal separator character (`.`) is used in a display pattern, such as `99.99`, the character that is rendered depends on the locale of the form. If the locale is German, the decimal separator character is the comma (`,`).

In the following example, all three fields contain the value `1234567.89`. The second and third fields in the example use the display pattern `$zzz,zzz,zz9.99`. The dollar sign (`$`) specifies the currency symbol, and the value is rendered using the locale of the object. When the locale of the object is German, the dollar sign (`$`) is mapped to the Euro symbol.



Plain Decimal Field	1,234,567.89
Picture Mask (Locale: Canadian)	$1,234,567.89
Picture Mask (Locale: German)	€1.234.567,89

For more information about field patterns, see the *LiveCycle Designer Help*.

Determine the locale of a form

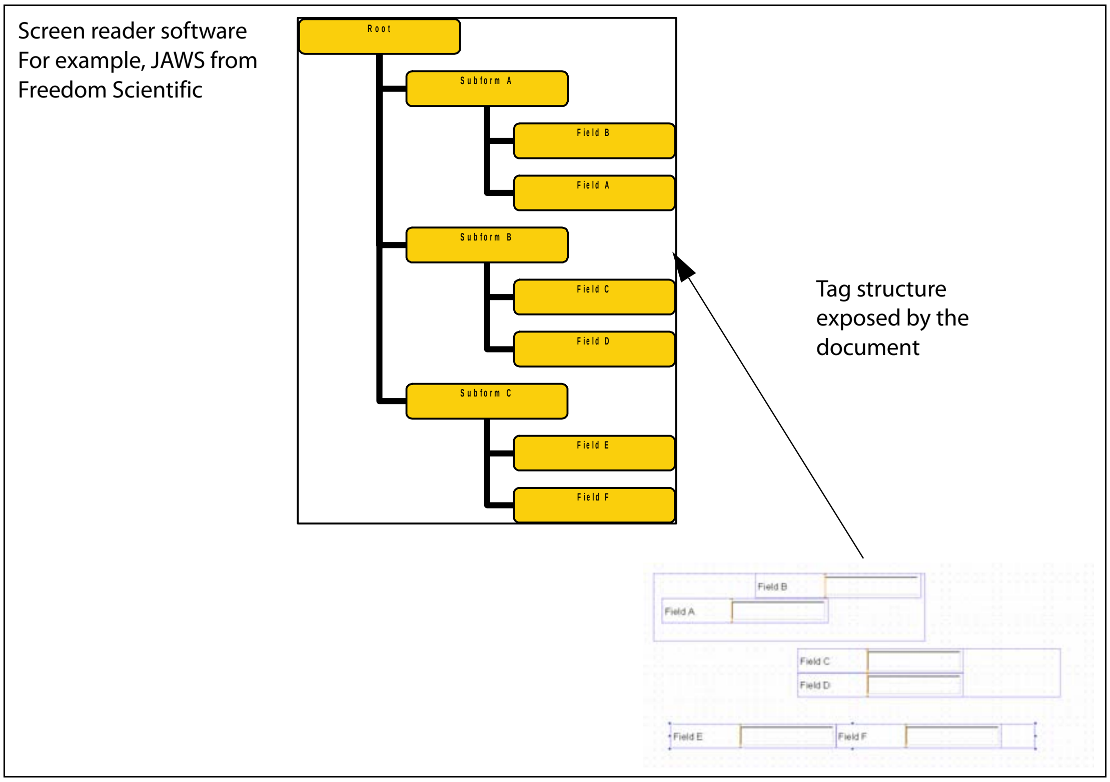
The locale of a form is determined by the following properties:

- The Locale property of the object if it is set.
- The Locale property of the form if it is set.
- The ambient locale. If a form is designed using the ambient locale, the form uses the locale of the computer on which it is rendered. This is true for all forms, including print forms and read-only fields on interactive forms.

  Note: The ambient locale is also known as the viewer locale.

Ensure Forms Are Accessible

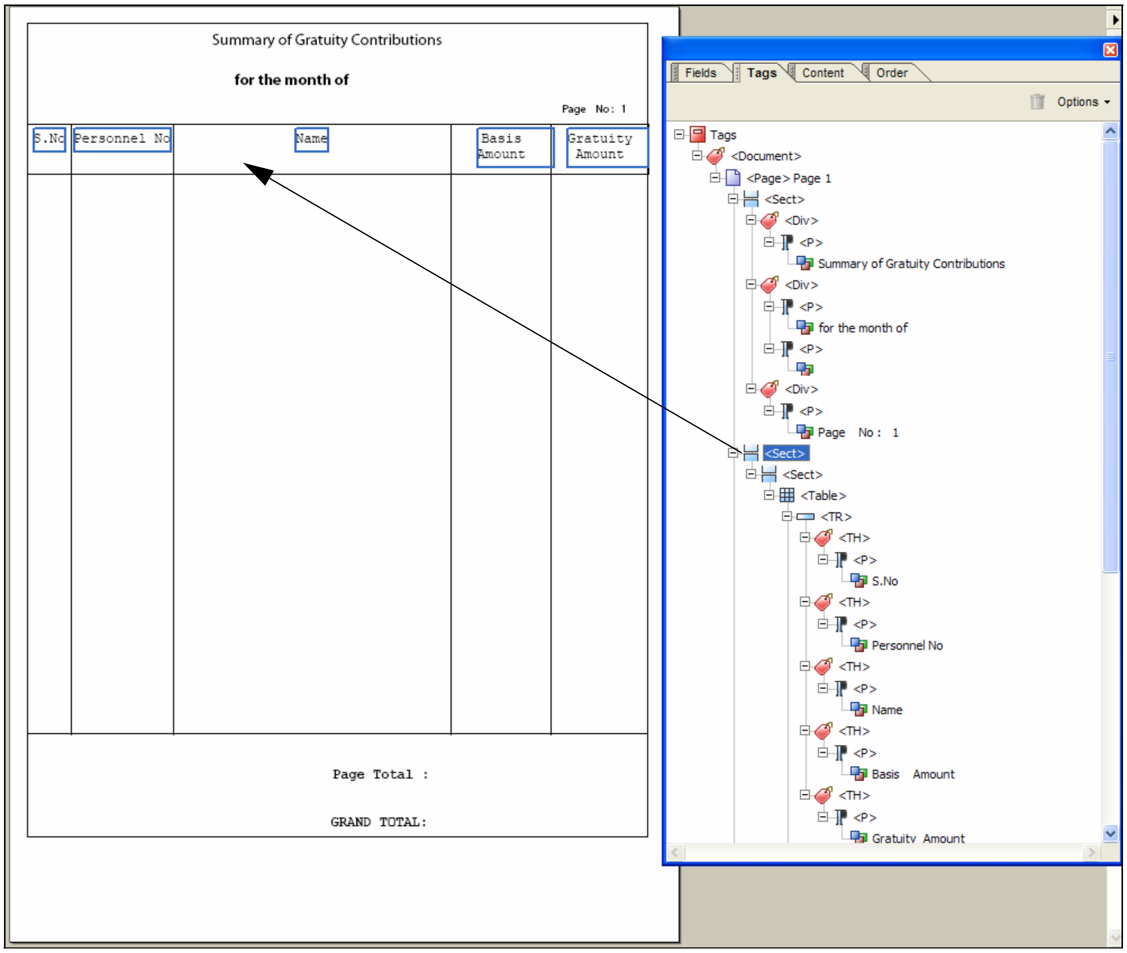
Many government agencies require electronic forms to be accessible to users with vision impairment. To make PDF forms accessible when rendered, specify a tagging structure that is included with the rendered PDF form. The tagging structure uses elements that resemble HTML tags and is read aloud by third-party screen reader software.



You can use Acrobat Professional to view the accessibility tags for a particular PDF form:

- To see the tags in Acrobat 7.0 Professional, select View > Navigation Tabs > Tags.
- To view the corresponding fields for a particular tag, in the Tags window, select Options > Highlight Content.

Accessibility tags visible in Adobe Acrobat Professional



Summary of Gratuity Contributions

for the month of

Page No: 1

S.No	Personnel No	Name	Basis Amount	Gratuity Amount

Page Total :

GRAND TOTAL:

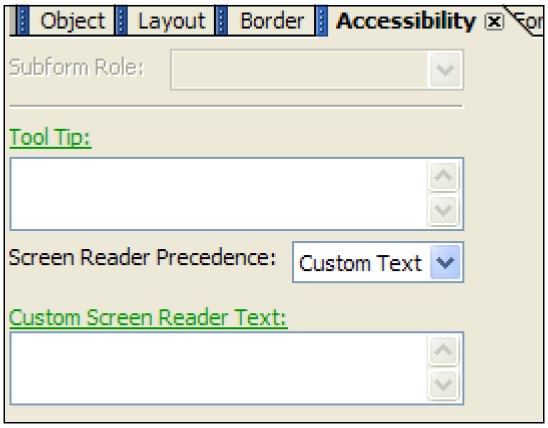
```
Tags
  <Document>
    <Page> Page 1
      <Sect>
        <Div>
          <P>
            Summary of Gratuity Contributions
        <Div>
          <P>
            for the month of
          <P>
        <Div>
          <P>
            Page   No :  1
      <Sect>
        <Sect>
          <Table>
            <TR>
              <TH>
                <P>
                  S.No
              <TH>
                <P>
                  Personnel No
              <TH>
                <P>
                  Name
              <TH>
                <P>
                  Basis   Amount
              <TH>
                <P>
                  Gratuity  Amount
```

Create accessible forms

To ensure that forms are accessible, each object should include screen reader text. For each field object in a form, you can specify one of several settings for screen reader text:

- Custom screen reader text, which you set in the Accessibility palette.
- Tool tips for objects, which you set in the Accessibility palette.
- Captions for fields.
- Names of objects, as specified in the Name option of the Binding tab. Set the screen reader text in the Accessibility palette.

Note: For objects with captions, such as text fields, radio buttons, and check boxes, use the captions and do not set custom text.



Screen Reader Precedence: Custom Text

To design accessible forms, follow these recommendations:

Use tables wherever possible: Tables help structure the content and enable the screen reader to identify the current location within a table. The screen reader uses the accessibility tags to read the column heading and row location.

Use subforms that flow content: Use subforms that flow content instead of subforms that position content to ensure that the accessibility tags are generated in the appropriate order. In subforms that position content, accessibility tags are generated left to right and top to bottom, which may not be the expected reading order.

Use Subforms That Flow Content

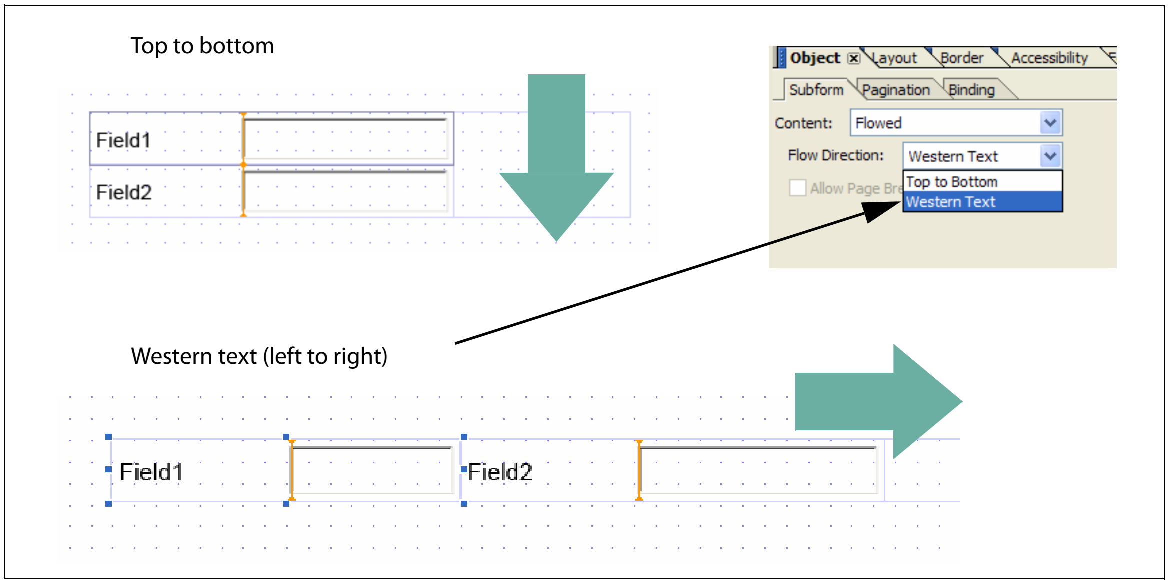
Subforms are container objects that you can use to group form design objects, including fields, boilerplate objects, and other subforms. You can configure subforms either to flow content or to position content.

Subforms that flow content

Subforms that flow content allow the layout engine in the Adobe document services to position the objects within the subform. Subforms can hold varying amounts of data and flow content according to the Flow Direction setting of its associated content area. The objects within the subform move together during the data-merging process so that none of its objects interfere with each other.

By default, the root subform is defined to flow content. Consequently, all subforms nested under the root subform can flow from one form page or content area into the next when merged with data.

In subforms that flow content, the content can flow either from top to bottom or as Western text (left to right).

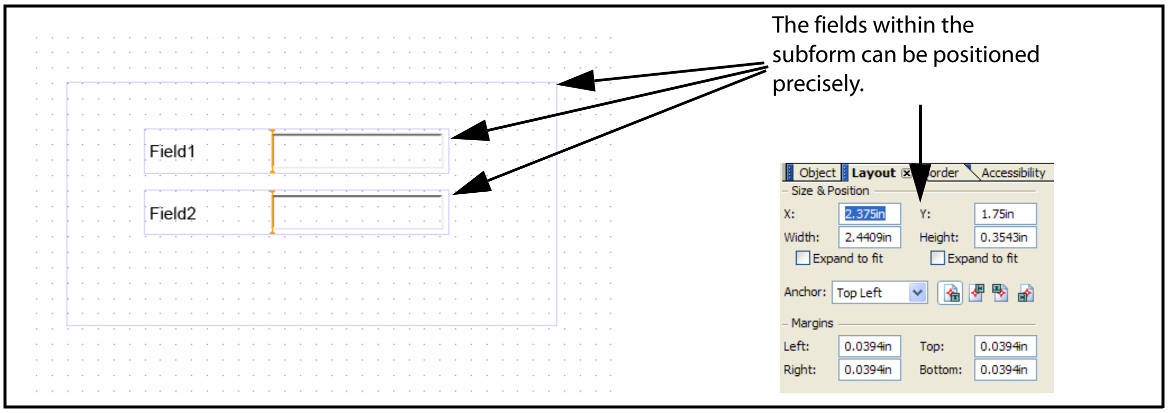


Content: Flowed

Flow Direction: Western Text

Subforms that position content

Subforms that position data can expand to fit any amount of merged data, but the objects within the subform cannot move from their anchor points. By default, all subforms except the root subform are defined to position content.



Size & Position
X: 2.375in Y: 1.75in
Width: 2.4409in Height: 0.3543in
Expand to fit Expand to fit
Anchor: Top Left
Margins
Left: 0.0394in Top: 0.0394in
Right: 0.0394in Bottom: 0.0394in

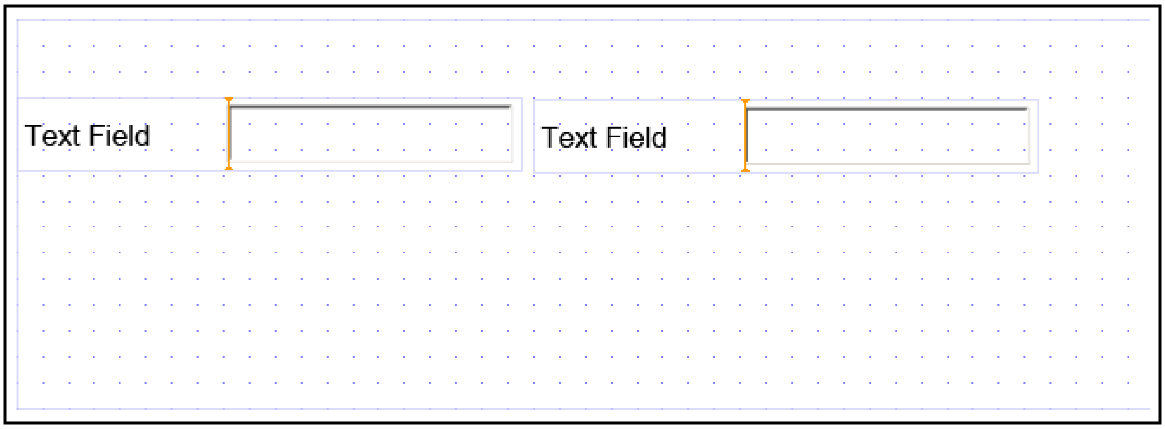
If a subform contains objects that merge with variable sizes of data, you should verify that those objects do not expand to the extent that they overrun the area occupied by another object. It is possible that expandable objects, such as text fields, might be rendered on top of other objects. To avoid this design concern, set the subform to flow and expand to fit the content.

When designing a dynamic form, complete the form design first and then set the default body page subform to flow content. Set the default body page subform to flow content last so that the subform remains visible and the objects that you place within the subform remain in the intended position on the body page.

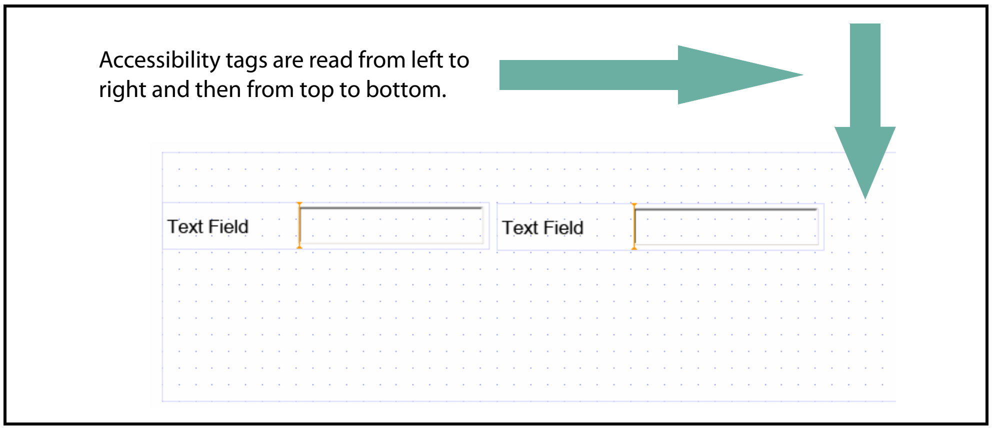
Flowed and positioned content

When designing forms, use the subforms that flow content property to avoid accessibility or navigation problems. Accessibility tags are generated from left to right and then from top to bottom. Accessibility problems can occur in subforms that position content when the objects are not positioned precisely; for example, the accessibility tags are not generated and read in the correct order.

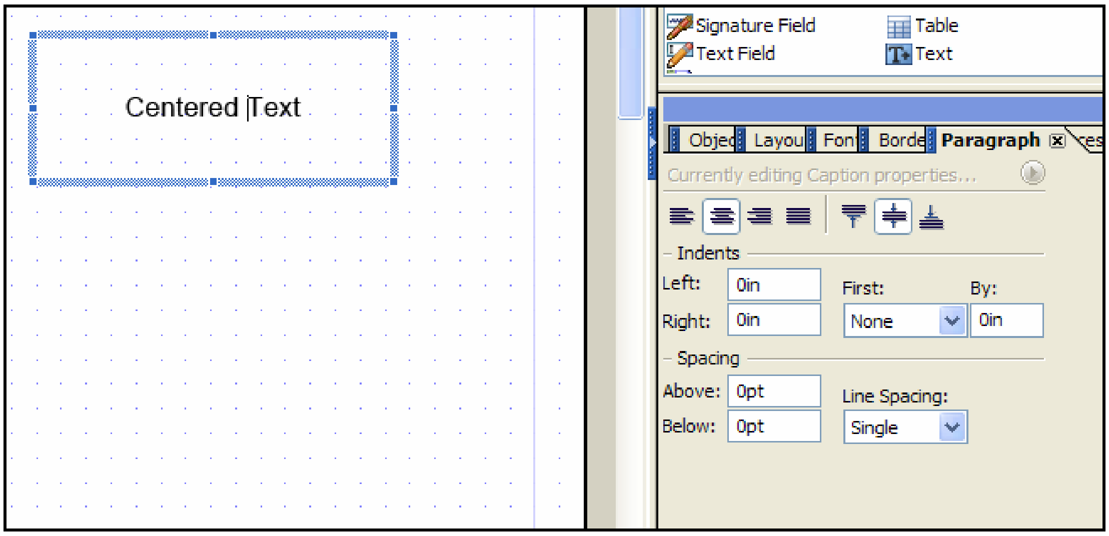
The example below illustrates the accessibility problems that can occur in a subform that positions content. The objects in the subform appear to be aligned vertically; however, the vertical heights actually differ by 1/10 inch.



In this example, the screen reader will read the text field on the right before the one on the left, which is not the intended reading order.


Accessibility tags are read from left to right and then from top to bottom.
Text Field
Text Field

To avoid accessibility problems, enclose the objects in a subform that flows content. To create space between the objects, use the margin values in the Layout palette. To position the caption and text for an object, use the Paragraph palette.



Left: 0in
Right: 0in
First: None
By: 0in
Above: 0pt
Below: 0pt
Line Spacing: Single

Use Nested Subforms to Structure Form Content

Use nested subforms to structure form content and to ensure that screen readers read the form content in the correct order. You can use nested subforms to create a form that looks and functions like a table. Use nested subforms instead of tables because the screen reader software reads the table structure. For example, the screen reader software reads “table with 4 columns” before it reads the contents of the table.

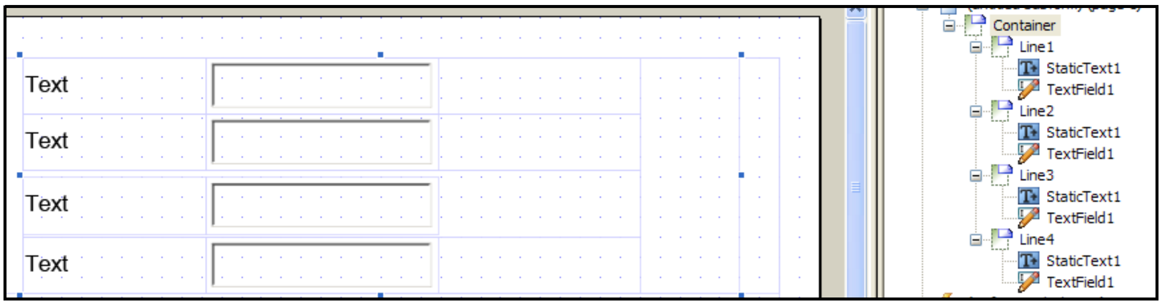
Use subforms to structure content because they provide anchoring, layout, and geometry management for objects. You can arrange the objects in a subform in rows, columns, or some other kind of balanced arrangement.

Nested subforms function like tables. *Tables* are flow content subforms that contain a repeatable subform with data, such as a body row. They ensure that the heights and widths of adjacent subforms remain consistent and export accessibility information (see “Ensure Forms Are Accessible” on page 20). Like tables, nested subforms are useful for structuring form content when the content originates from a relational database and contains multiple repeated record types or when nested data exists.

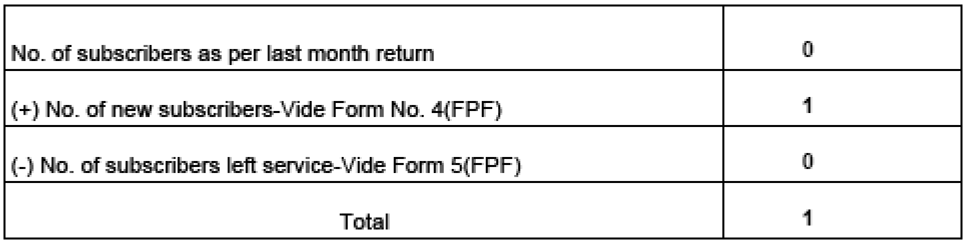
To structure the form content using nested subforms, do not set the Accessibility Role for the subforms. For more information about how to create accessible form designs using nested subforms, see the example in “Create Accessible Multiline Tables” on page 27.

A form design that uses nested subforms to structure the content nests the subforms in a hierarchy.

Form design with hierarchy of nested subforms



Rendered form



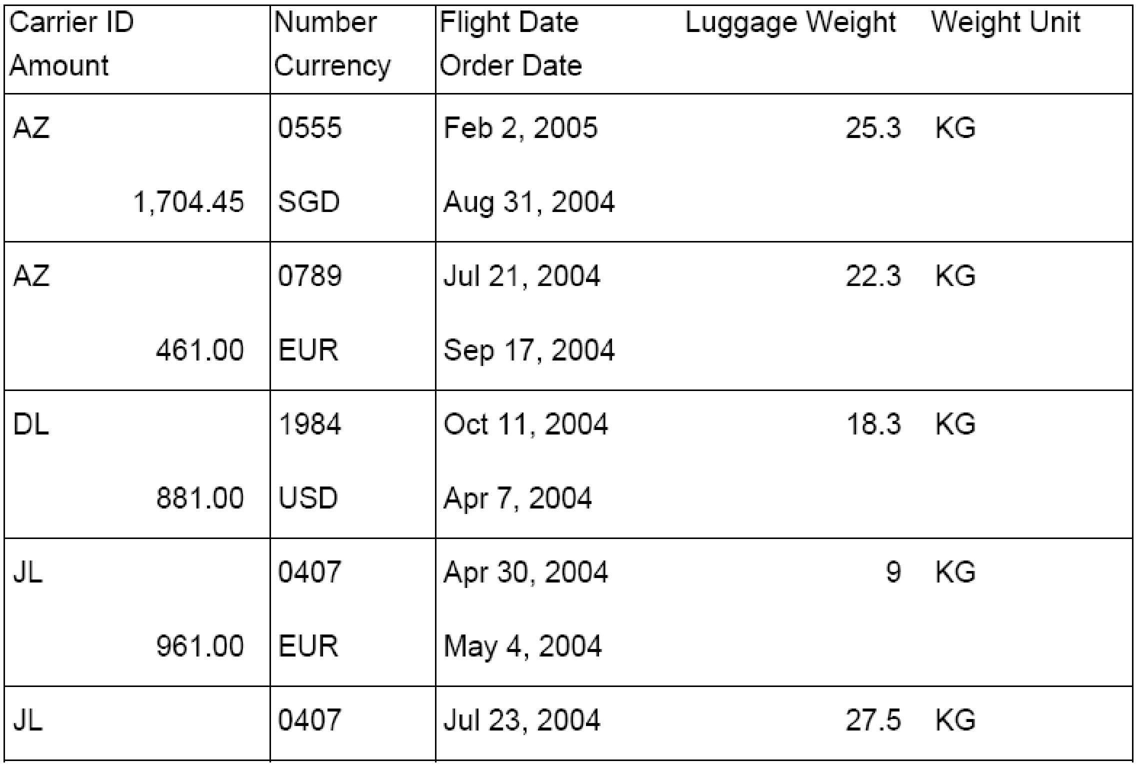
No. of subscribers as per last month return	0
(+) No. of new subscribers-Vide Form No. 4(FPF)	1
(-) No. of subscribers left service-Vide Form 5(FPF)	0
Total	1

Create Accessible Multiline Tables

These guidelines use an example to explain how to create an accessible, multiline table.

Note: See the attached files for the complete solution used in this example: multiline_output.pdf, multiline_table.xdp, and multiline_table.xml.

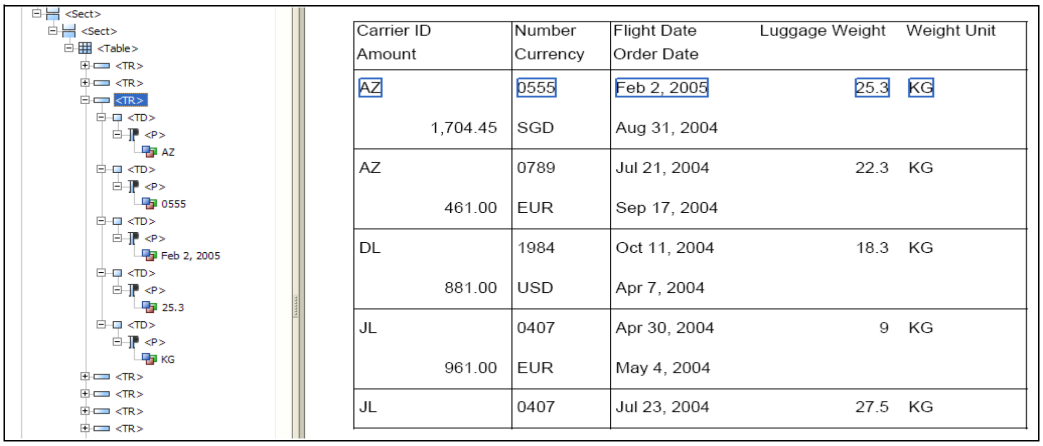
This example explains the steps required to create the following accessible multiline table.



Carrier ID	Number	Flight Date	Luggage Weight	Weight Unit
Amount	Currency	Order Date		
AZ	0555	Feb 2, 2005	25.3	KG
1,704.45	SGD	Aug 31, 2004		
AZ	0789	Jul 21, 2004	22.3	KG
461.00	EUR	Sep 17, 2004		
DL	1984	Oct 11, 2004	18.3	KG
881.00	USD	Apr 7, 2004		
JL	0407	Apr 30, 2004	9	KG
961.00	EUR	May 4, 2004		
JL	0407	Jul 23, 2004	27.5	KG

Using subforms to emulate a table

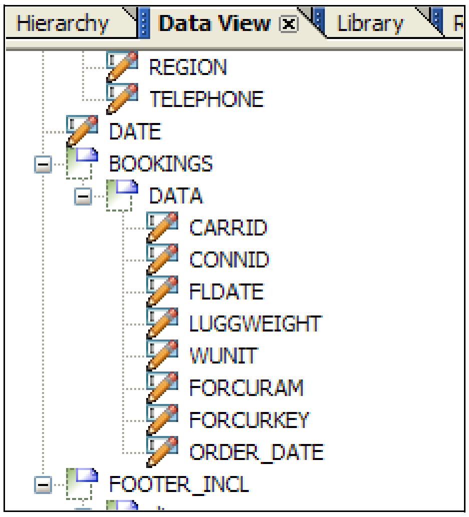
Instead of using Table Designer, build this form design with a set of nested subforms and then set the accessibility role of the subforms to emulate a table. Use nested subforms instead of Table Designer because a table with multiple header and body rows is not accessible. As illustrated below, when a table has multiple rows, the form generates multiple lines of accessibility tags. The form design generates these accessibility tags even if the body rows are grouped in a table section.



```
<Sect>
  <Sect>
    <Table>
      <TR>
      <TR>
      <TR>
        <TD>
          <P>
            AZ
        <TD>
          <P>
            0555
        <TD>
          <P>
            Feb 2, 2005
        <TD>
          <P>
            25.3
        <TD>
          <P>
            KG
      <TR>
      <TR>
      <TR>
      <TR>
```

Carrier ID	Number	Flight Date	Luggage Weight	Weight Unit
Amount	Currency	Order Date		
AZ	0555	Feb 2, 2005	25.3	KG
1,704.45	SGD	Aug 31, 2004		
AZ	0789	Jul 21, 2004	22.3	KG
461.00	EUR	Sep 17, 2004		
DL	1984	Oct 11, 2004	18.3	KG
881.00	USD	Apr 7, 2004		
JL	0407	Apr 30, 2004	9	KG
961.00	EUR	May 4, 2004		
JL	0407	Jul 23, 2004	27.5	KG

This example assumes that a data schema is set up for the form design. If you develop forms in the SAP environment, the data schema is generated from the context when the form is opened in LiveCycle Designer. As illustrated the Data View palette contains a set of fields that you want to put in the table.

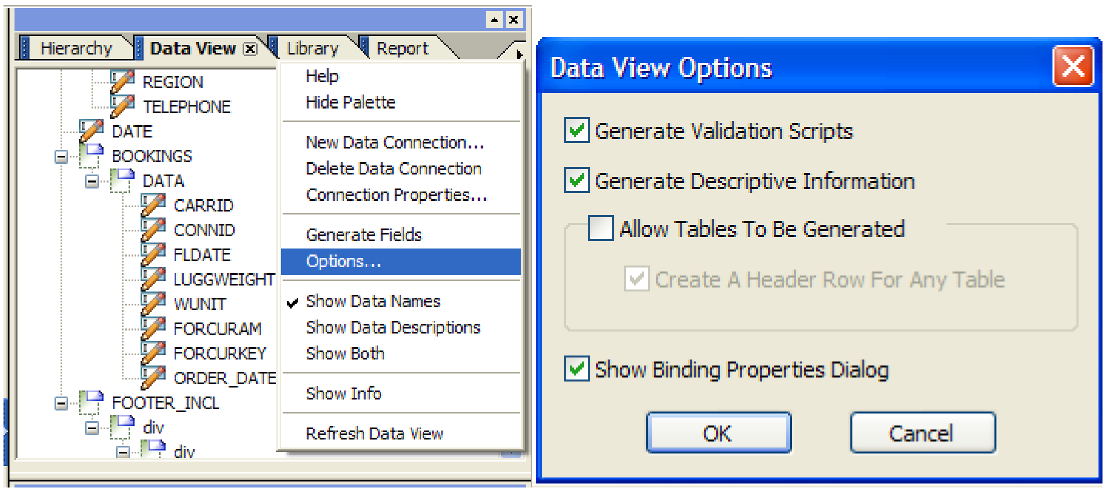


```
    REGION
    TELEPHONE
  DATE
⊟ BOOKINGS
  ⊟ DATA
      CARRID
      CONNID
      FLDATE
      LUGGWEIGHT
      WUNIT
      FORCURAM
      FORCURKEY
      ORDER_DATE
⊟ FOOTER_INCL
```

‰ To turn off the table generation option:

1. Drag data fields onto the form without creating a table. Tables do not support wrapped header and body rows, which are necessary to make this form design accessible.

2. Click the Data View palette menu and deselect Allow Tables To Be Generated.

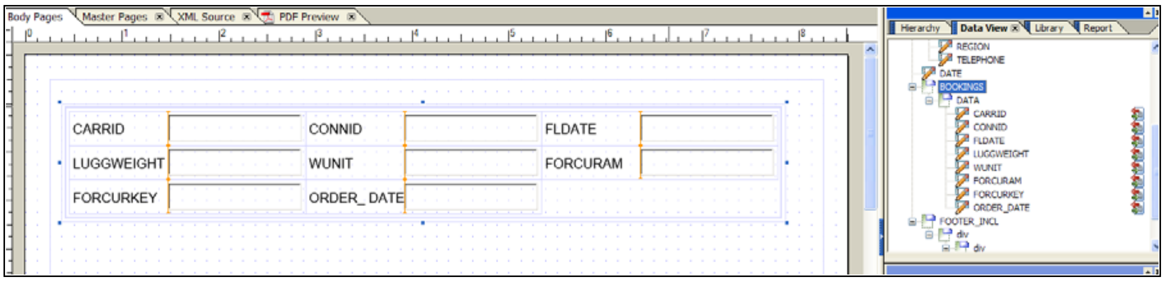


Data View Options

- [✓] Generate Validation Scripts
- [✓] Generate Descriptive Information
- [ ] Allow Tables To Be Generated
  - [✓] Create A Header Row For Any Table
- [✓] Show Binding Properties Dialog

‰ To create the data bindings:

l Drag the BOOKINGS data node from the Data View palette to the far left side of the form.



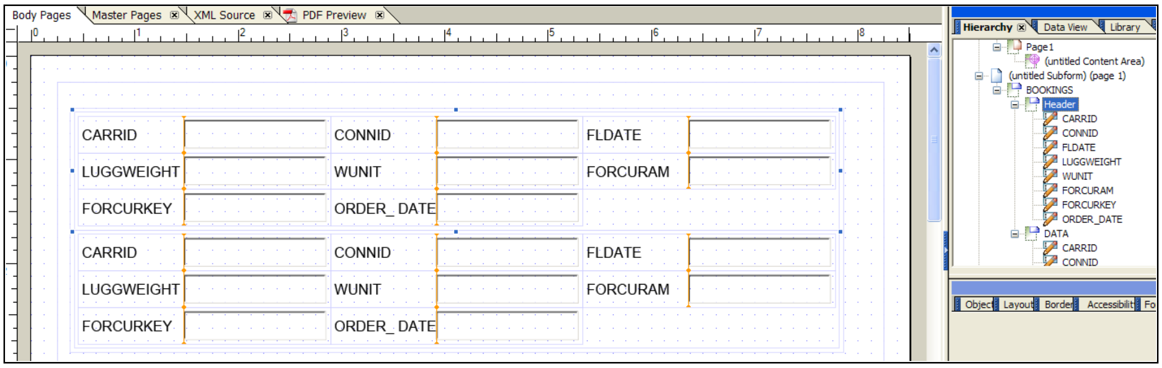
‰ To create the Header subform:

1. In the Hierarchy palette, click the DATA subform.

2. To create the header subform, select Edit > Duplicate. Duplicating the subform makes it easier to create the headings from the DATA subform text field captions.

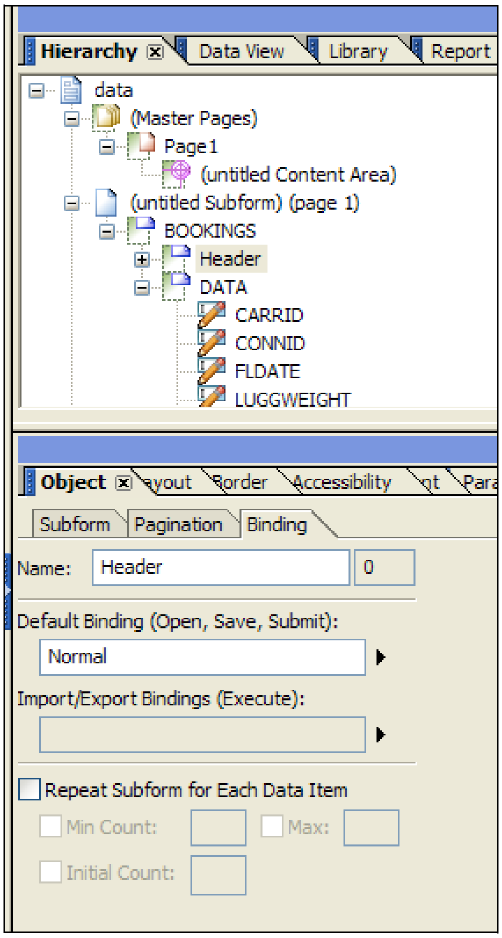
   There are now two subforms named DATA (DATA[0] and DATA[1]).

3. Rename the first DATA subform to Header. The form design should look like this illustration.



When the DATA subform was duplicated, the Header subform inherited two properties that are not required.

4. In the Object palette, click the Binding tab and deselect Repeat Subform For Each Data Item.



Name: Header 0

Default Binding (Open, Save, Submit): Normal

Import/Export Bindings (Execute):

☐ Repeat Subform for Each Data Item

☐ Min Count: ☐ Max:

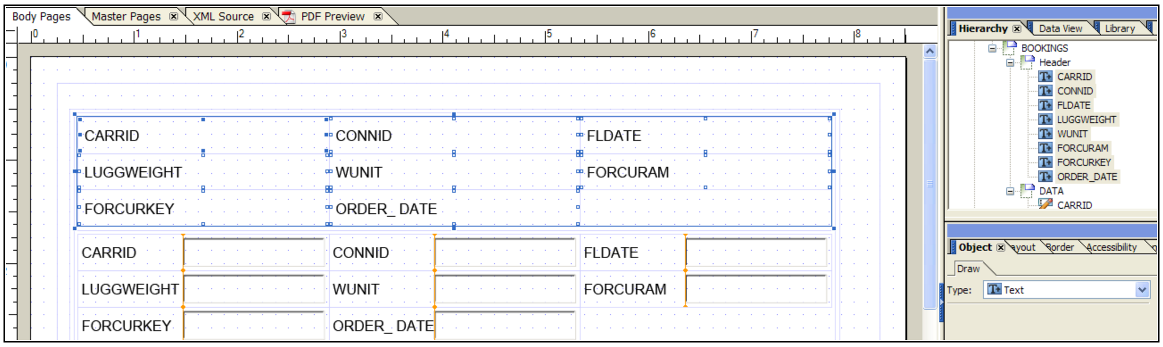
☐ Initial Count:

5. From the Default Binding list, select Normal.

‰ To create captions for the Header subform:

1. In the Hierarchy palette, select all of the text fields in the Header subform.

2. In the Object palette, click the Draw tab and select Text from the Type list. Setting the type to text converts all of the text fields to static text and copies the text field captions.

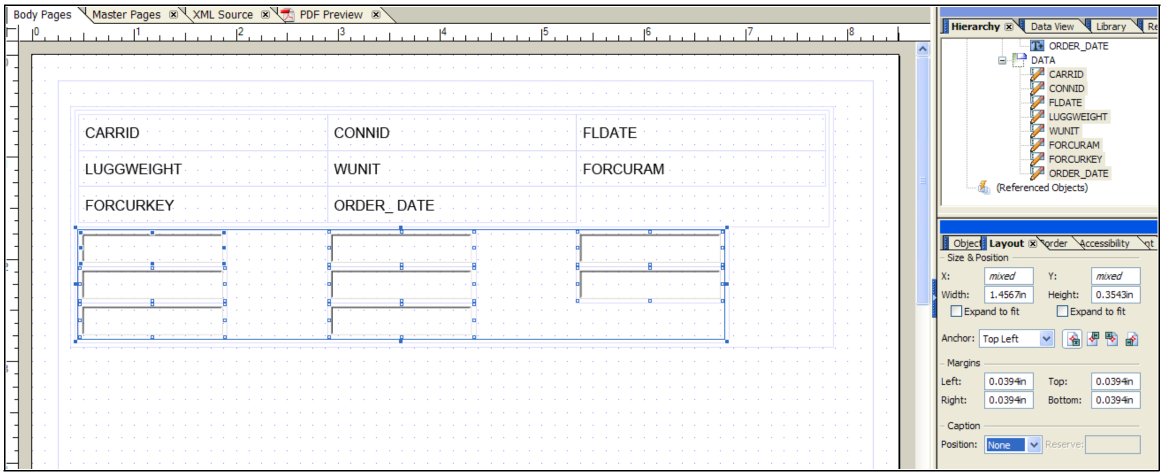


Type: Text

‰ To remove the captions from the DATA subform:

1. In the Hierarchy palette, select all of the text fields in the DATA subform.

2. In the Layout palette, under Caption, select None from the Position list.



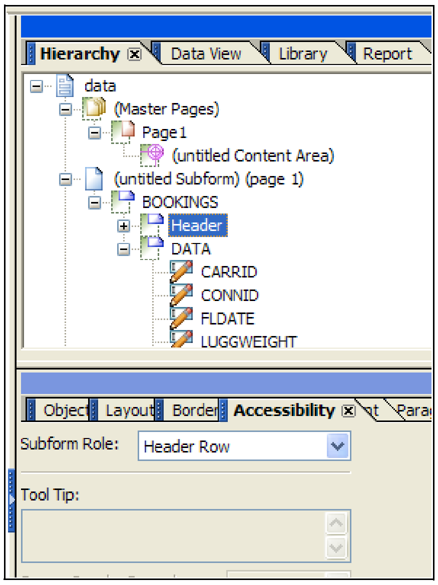
X: mixed Y: mixed
Width: 1.4567in Height: 0.3543in
Expand to fit Expand to fit
Anchor: Top Left
Left: 0.0394in Top: 0.0394in
Right: 0.0394in Bottom: 0.0394in
Position: None Reserve:

Making the table accessible

For the screen reader software to recognize the subforms as a table, the `<Table>`, `<TR>`, `<TH>`, and `<TD>` tags must be generated as part of the PDF file. To generate the accessibility tags, you must set the Subform Role field for the BOOKINGS, Header, and DATA subforms.

‰ **To set the accessibility role for each subform:**

1. In the Hierarchy palette, select a subform.



Subform Role: Header Row

2. In the Accessibility palette, select the Subform Role for each subform as follows:

Subform	Subform Role
BOOKINGS	Table
Header	Header Row
DATA	Body Row

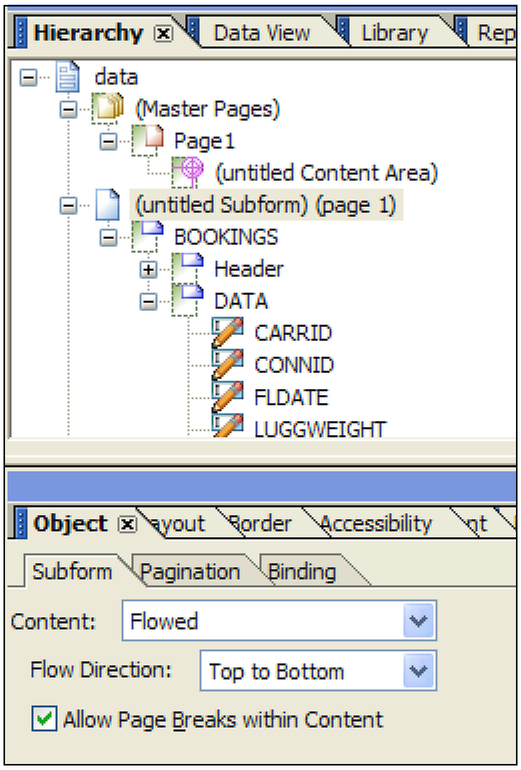
Enabling multipage tables

The parent subform on the body page should be set to flow content instead of position content. When the subform is set to flow content and the form is rendered with XML data, the table can span multiple pages.

‰ **To enable the table to span multiple pages:**

1. Select the parent subform.

2. In the Object palette, click the Subform tab and select Flowed from the Content list.

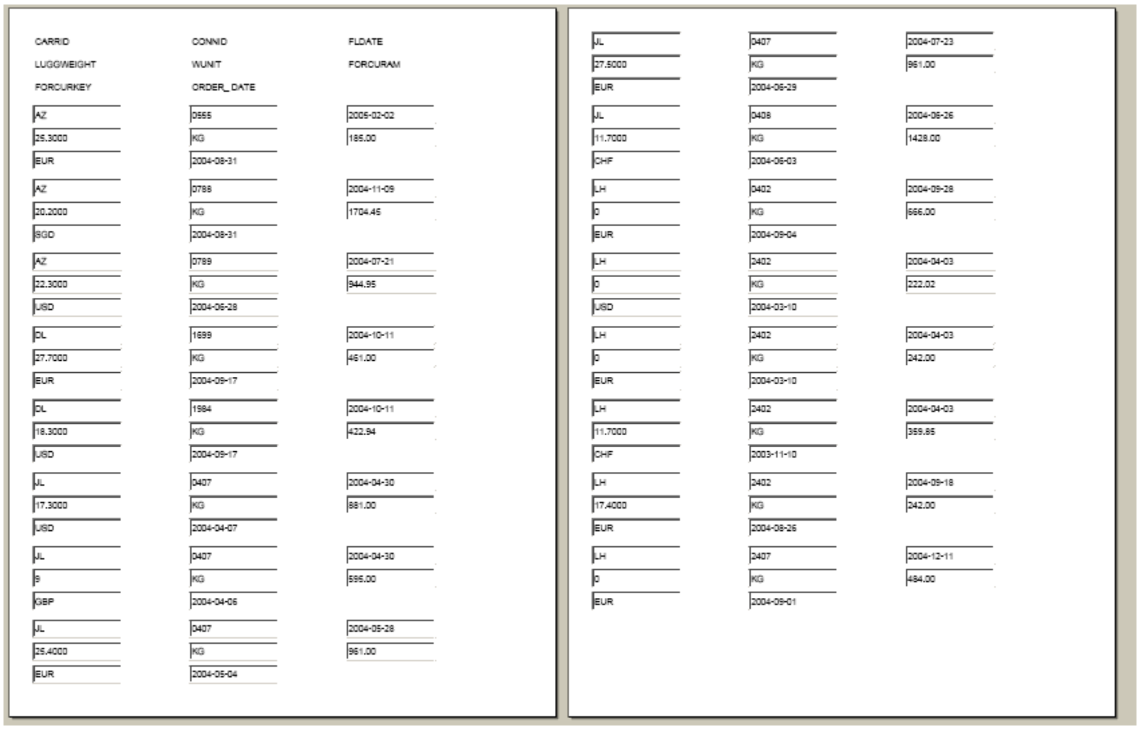


Content: Flowed

Flow Direction: Top to Bottom

[✓] Allow Page Breaks within Content

3. From the Flow Direction list, select Top to Bottom so that the form can render data on multiple pages. However, the Header appears only on the first page.

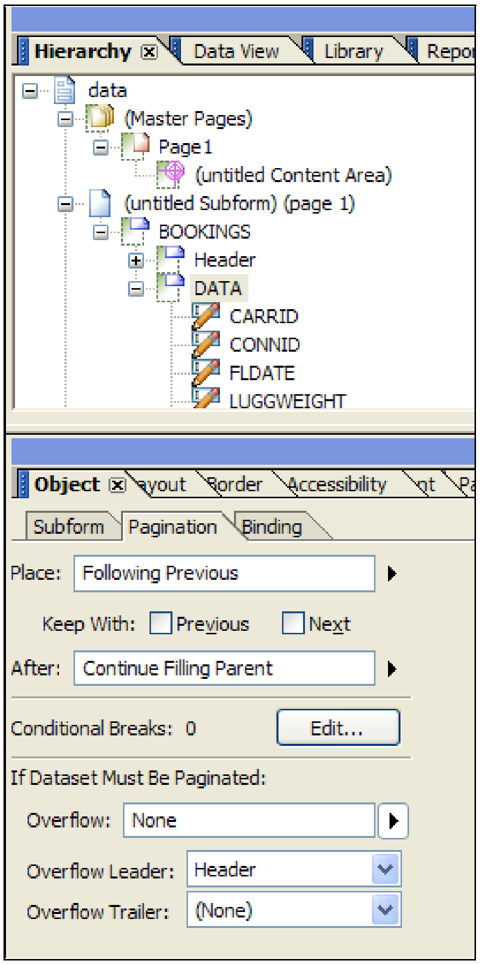


CARRID	CONNID	FLDATE
LUGGWEIGHT	WUNIT	FORCURAM
FORCURKEY	ORDER_ DATE	
AZ	0555	2005-02-02
25.3000	KG	185.00
EUR	2004-08-31	
AZ	0788	2004-11-09
20.2000	KG	1704.45
SGD	2004-08-31	
AZ	0789	2004-07-21
22.3000	KG	944.95
USD	2004-06-28	
DL	1699	2004-10-11
27.7000	KG	461.00
EUR	2004-09-17	
DL	1984	2004-10-11
18.3000	KG	422.94
USD	2004-09-17	
JL	0407	2004-04-30
17.3000	KG	881.00
USD	2004-04-07	
JL	0407	2004-04-30
9	KG	595.00
GBP	2004-04-06	
JL	0407	2004-05-28
25.4000	KG	961.00
EUR	2004-05-04	

JL	0407	2004-07-23
27.5000	KG	961.00
EUR	2004-06-29	
JL	0408	2004-06-26
11.7000	KG	1428.00
CHF	2004-06-03	
LH	0402	2004-09-28
0	KG	666.00
EUR	2004-09-04	
LH	2402	2004-04-03
0	KG	222.02
USD	2004-03-10	
LH	2402	2004-04-03
0	KG	242.00
EUR	2004-03-10	
LH	2402	2004-04-03
11.7000	KG	359.85
CHF	2003-11-10	
LH	2402	2004-09-18
17.4000	KG	242.00
EUR	2004-08-26	
LH	2407	2004-12-11
0	KG	484.00
EUR	2004-09-01	

‰ **To set the Header subform to appear at the top of each page:**

1. In the Hierarchy palette, select the DATA subform.
2. In the Object palette, click the Pagination tab. Under If Dataset Must Be Paginated, select Header from the Overflow Leader list.



Place: Following Previous

Keep With: ☐ Previous ☐ Next

After: Continue Filling Parent

Conditional Breaks: 0

If Dataset Must Be Paginated:

Overflow: None

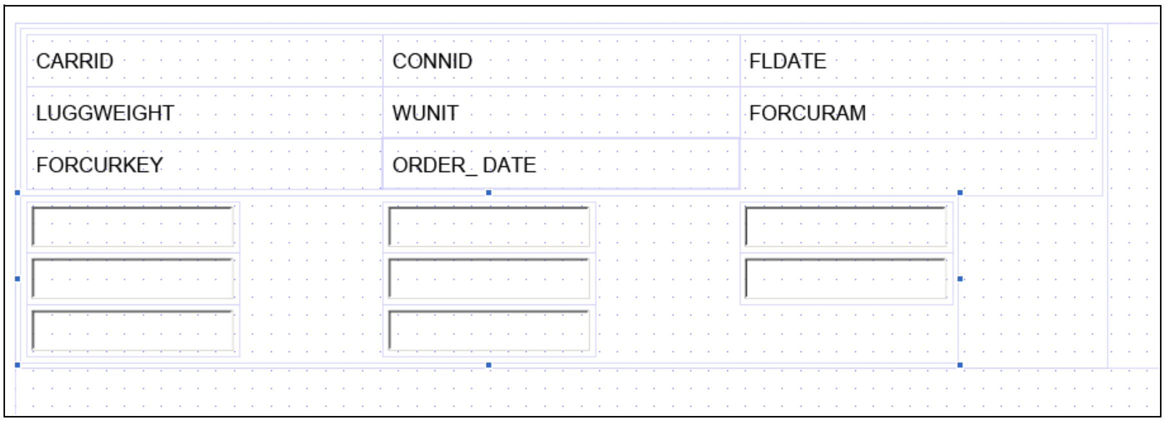
Overflow Leader: Header

Overflow Trailer: (None)

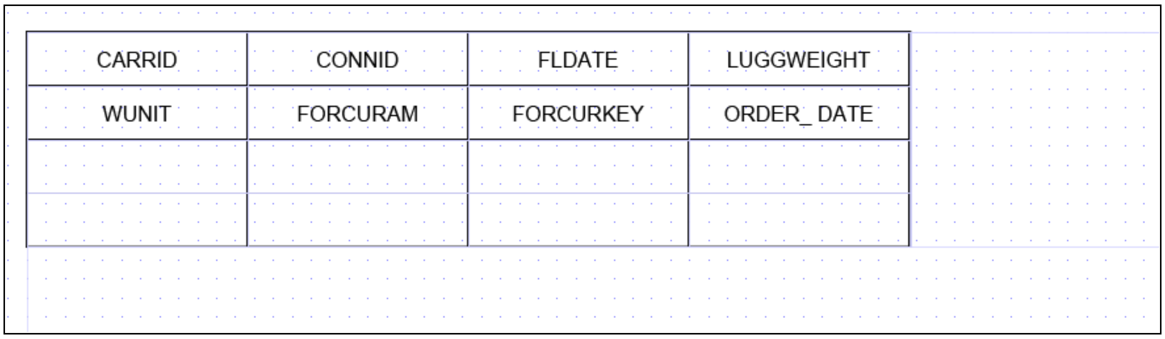
3. (Optional) To use the subform as an overflow leader, in the Subform tab, select Positioned from the Content list.

Arranging and resizing the fields

Configure the subforms so that the rendered output resembles a table. Before you arrange and resize the fields, the form design looks like this illustration.



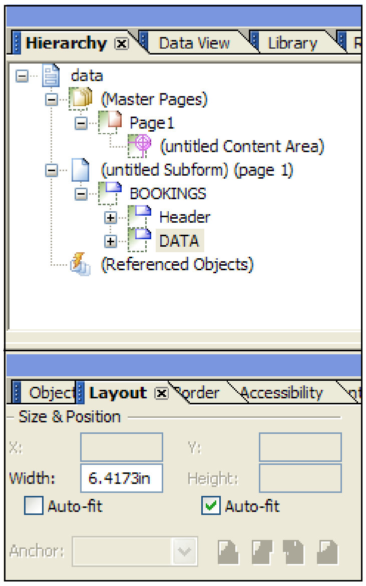
After you arrange and resize the fields, the form design should look like this illustration.



To arrange the fields as a body row in a table, you must set the flow content of the subform to Western Text, which flows the fields from left to right and top to bottom within the subform. Changing the flow content of the subform initially sets the direction to top to bottom; therefore, you must ensure that the width of the subform is fixed and will not automatically adjust. Otherwise, when you set the flow content of the subform, it will not change the width of the subform to a size that exceeds the content area.

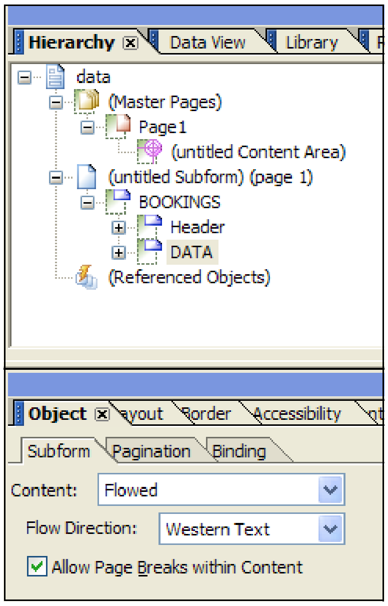
‰ **To arrange the DATA and Header subform fields:**

1. In the Hierarchy palette, select the DATA subform.
2. In the Layout palette, deselect Auto-Fit.



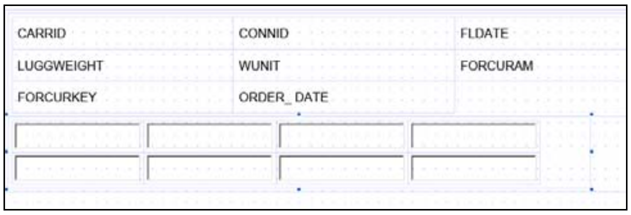
Width: 6.4173in
☐ Auto-fit
☑ Auto-fit

3. In the Object palette, click the Subform tab and select Flowed from the Content list. This resizes the subform so that all fields are vertically arranged.



Content: Flowed
Flow Direction: Western Text
☑ Allow Page Breaks within Content

4. From the Flow Direction list, select Western Text. The subform now arranges the fields in two rows, each with four fields. The fields are arranged next to each other without requiring manual resizing.



5. In the Hierarchy palette, select all of the fields in the DATA subform.

6. In the Object palette, click the Field tab and select Solid from the Appearance list.

7. In the Layout palette, set the margins to 0.

8. In the Hierarchy palette, select the Header subform and then repeat steps 1 to 4.

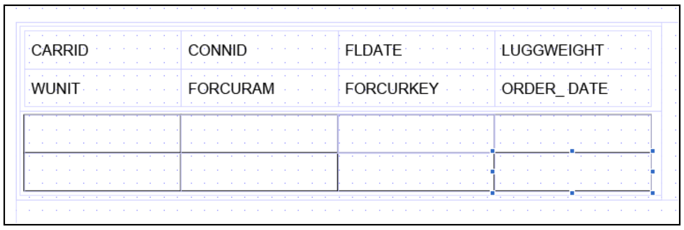
‰ **To resize the Header and DATA subform fields:**

1. In the Hierarchy palette, select the first field in the DATA subform.

2. Ctrl+click the first field in the Header subform and then select Layout > Make Same Size > Width. This resizes the Header field to match the DATA field.

3. Resize the Header and DATA subforms so that they are the same width as the four fields in the first row in the DATA subform.

4. Resize all of the Header fields so that they are the same size as the first field. The form design should now look like this illustration.



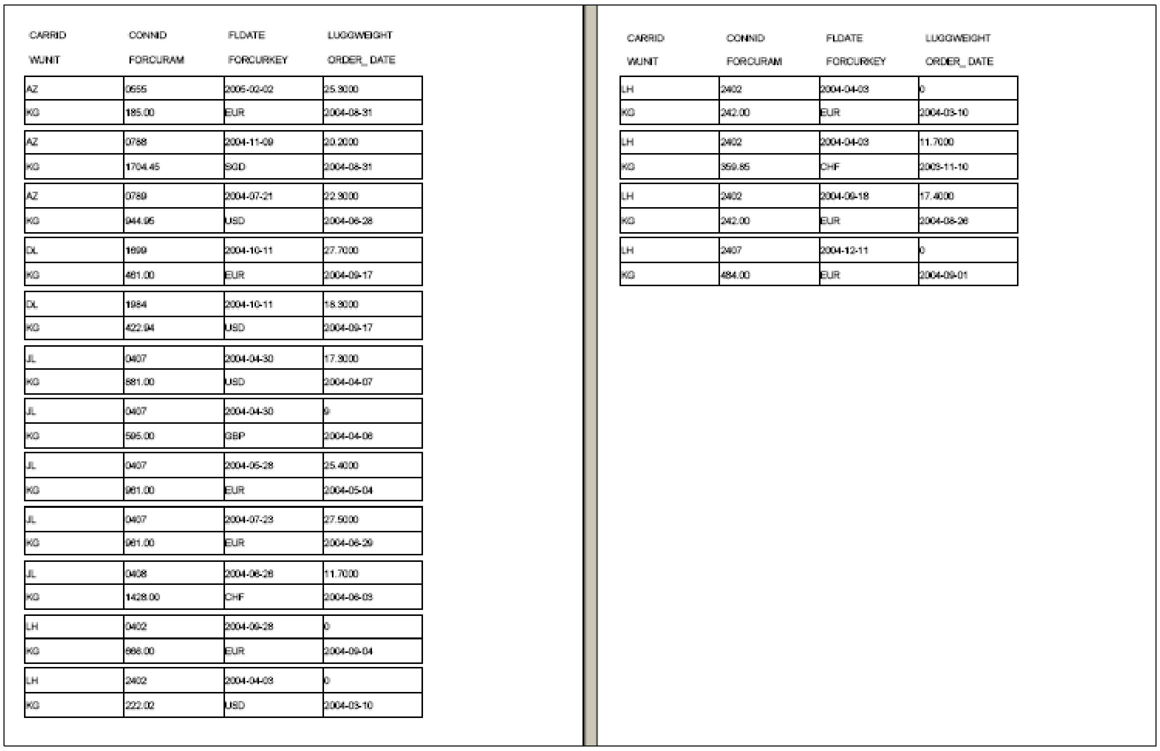
5. To use the Header subform as an overflow leader that appears on each page, in the Object palette, click the Subform tab and select Positioned from the Content list.

6. To set different column widths, do these tasks:
   - Select the DATA subform and select Positioned from the Content list.
   - Adjust the width of a field in the Header subform and then use the Layout > Make Same Size > Width command to set width of the corresponding DATA field.

Note: Resetting the DATA subform to flow content helps arrange the fields but is not mandatory.

7. Select the DATA subform, click the Layout palette, and set the margins to 0.
8. (Optional) You can adjust the alignment of columns to suit the type of data that is merged with the form design.

   The rendered form looks like this illustration.

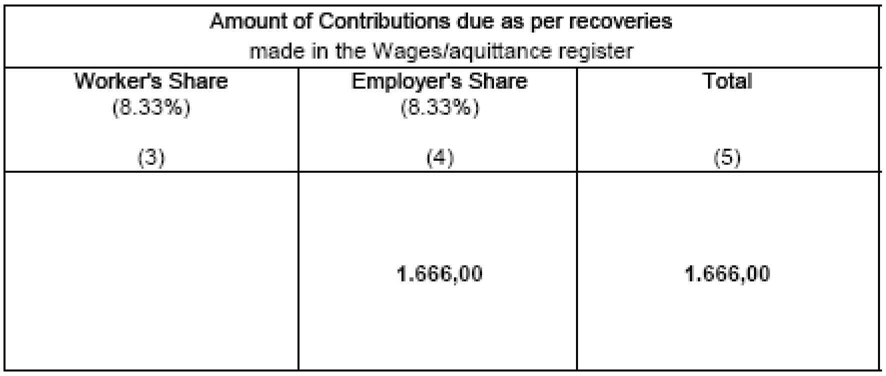


CARRID	CONNID	FLDATE	LUGGWEIGHT
WUNIT	FORCURAM	FORCURKEY	ORDER_ DATE
AZ	0555	2005-02-02	25.3000
KG	185.00	EUR	2004-08-31
AZ	0788	2004-11-09	20.2000
KG	1704.45	SGD	2004-08-31
AZ	0789	2004-07-21	22.3000
KG	944.95	USD	2004-06-28
DL	1699	2004-10-11	27.7000
KG	481.00	EUR	2004-09-17
DL	1984	2004-10-11	18.3000
KG	422.94	USD	2004-09-17
JL	0407	2004-04-30	17.3000
KG	881.00	USD	2004-04-07
JL	0407	2004-04-30	9
KG	595.00	GBP	2004-04-08
JL	0407	2004-05-28	25.4000
KG	981.00	EUR	2004-05-04
JL	0407	2004-07-23	27.5000
KG	981.00	EUR	2004-06-29
JL	0408	2004-06-28	11.7000
KG	1428.00	CHF	2004-06-03
LH	0402	2004-09-28	0
KG	666.00	EUR	2004-09-04
LH	2402	2004-04-03	0
KG	222.02	USD	2004-03-10

CARRID	CONNID	FLDATE	LUGGWEIGHT
WUNIT	FORCURAM	FORCURKEY	ORDER_ DATE
LH	2402	2004-04-03	0
KG	242.00	EUR	2004-03-10
LH	2402	2004-04-03	11.7000
KG	359.85	CHF	2003-11-10
LH	2402	2004-09-18	17.4000
KG	242.00	EUR	2004-08-26
LH	2407	2004-12-11	0
KG	484.00	EUR	2004-09-01

Create Accessible, Multiline, Heterogeneous Header Tables

A multiline, heterogeneous header table has a multiline header row that spans several columns.



Amount of Contributions due as per recoveries made in the Wages/aquittance register		
Worker's Share (8.33%) (3)	Employer's Share (8.33%) (4)	Total (5)
	1.666,00	1.666,00

To create this table, use the Table Designer and then use column spanning on the first header row. The sample files for this document are mhh_table.xdp, mhh_table.xml and mhh_table.pdf.

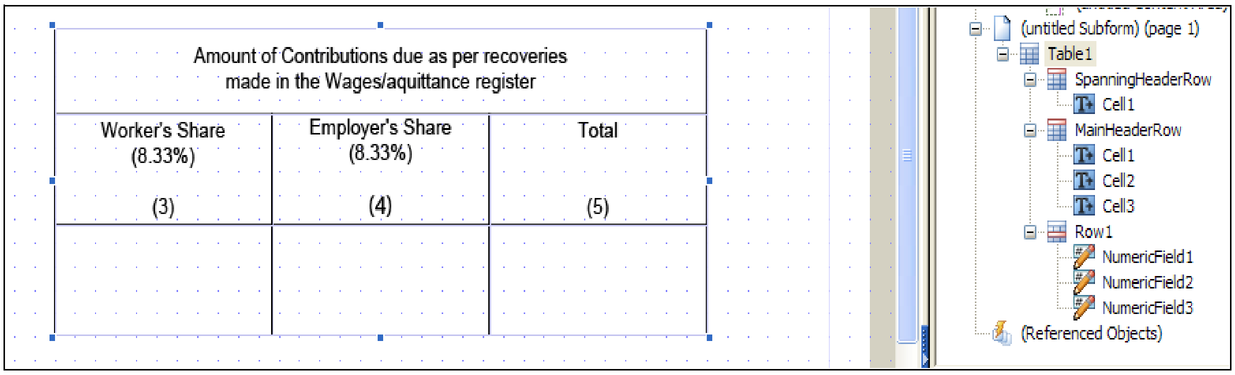
‰ **To create a multiline, heterogeneous header table:**

1. Select Table > Insert Table. Create a simple table with three columns, one body row, and one header row.

2. In the Hierarchy palette, right-click the first header row and then select Insert > Row Below.

3. Select all three cells in the first header row, and then right-click and select Merge Cells.

   The form design looks like this illustration.



Amount of Contributions due as per recoveries made in the Wages/aquittance register		
Worker's Share (8.33%) (3)	Employer's Share (8.33%) (4)	Total (5)

```
(untitled Subform) (page 1)
  Table1
    SpanningHeaderRow
      Cell1
    MainHeaderRow
      Cell1
      Cell2
      Cell3
    Row1
      NumericField1
      NumericField2
      NumericField3
  (Referenced Objects)
```

The screen reader software, such as JAWS, should read the main header along with individual cells from the second header for each body row cell in the table.

For example, for the following table:

H 1		
H 2 1	H 2 2	H 2 3
Data 1	Data 2	Data 3

JAWS would read the table positions as follows:

Cell Data 1	H 1 H 2 1
Cell Data 2	H 1 H 2 2
Cell Data 3	H 1 H 2 3

For the following, more complex, example:

H 1 1		H 1 2
H 2 1	H 2 2	H2 3
Data 1	Data 2	Data 3

JAWS would read the table positions as follows:

Cell Data 1	H 1 1 H 2 1
Cell Data 2	H 1 1 H 2 2
Cell Data 3	H 1 2 H 2 3

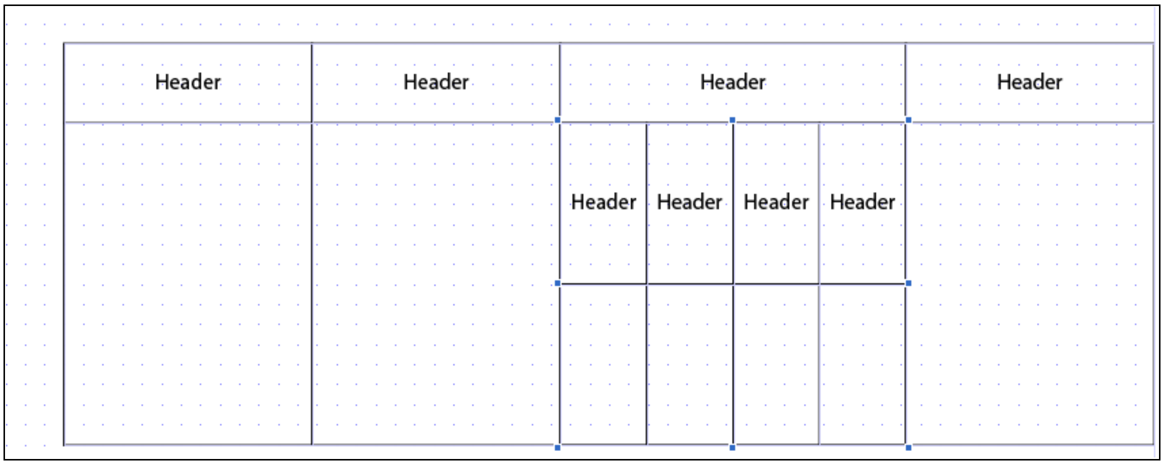
Note: At this time (March 27, 2007), only the following supported software solution is accessible:

- JAWS 7.0
- Adobe Reader 7.07 (available in early 2006)

It is recommended that forms requiring multiline, heterogeneous header tables use this kind of design. For this solution to comply with accessibility requirements, the JAWS software may require fixes.

Create Accessible Nested Tables

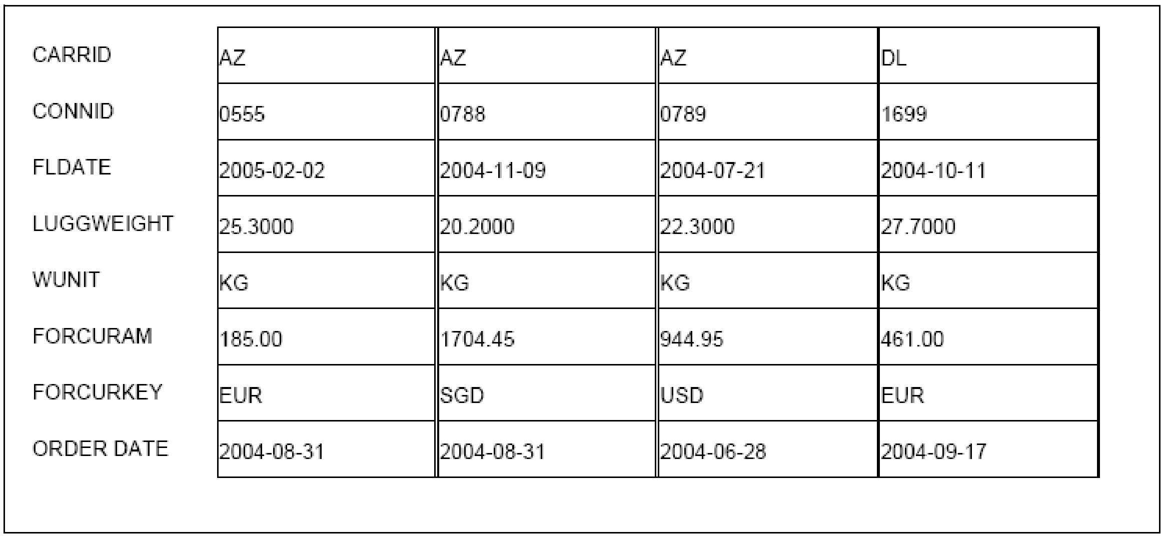
It is possible to design nested tables by using the LiveCycle Designer table features. As illustrated here, you can create a nested table by selecting a cell on the outer table and then inserting a second table by using the Insert Table menu command.



JAWS 7.0 reads nested tables as layout tables. Layout tables are used to arrange objects on the screen but do not imply any hierarchical relationship. Although Adobe Reader provides a hierarchy of table structures to JAWS, JAWS cannot read the inner table.

Create Accessible Horizontal Tables

If you need to flow data horizontally across each page, use a set of nested subforms instead of using Table Designer. The sample files are this best practice are horz_tables.xdp, horz_tables.xml and horz_tables.pdf



CARRID	AZ	AZ	AZ	DL
CONNID	0555	0788	0789	1699
FLDATE	2005-02-02	2004-11-09	2004-07-21	2004-10-11
LUGGWEIGHT	25.3000	20.2000	22.3000	27.7000
WUNIT	KG	KG	KG	KG
FORCURAM	185.00	1704.45	944.95	461.00
FORCURKEY	EUR	SGD	USD	EUR
ORDER DATE	2004-08-31	2004-08-31	2004-06-28	2004-09-17

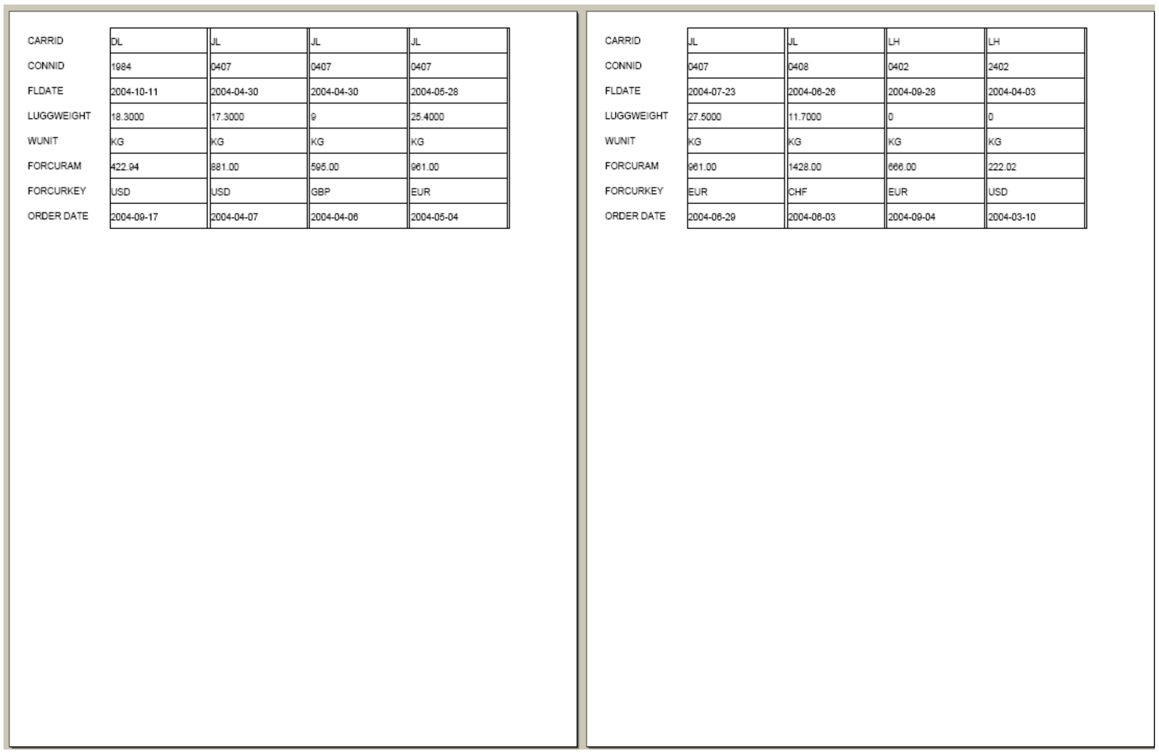
For more information about creating an accessible multiline table, see "Create Accessible Multiline Tables" on page 27.

To create this form, you must make the following changes:

- For the Header subform, set the Content option to Positioned.
- For the DATA subform, set the Content option to Positioned.
- For the DATA subform, turn off the Allow Page Breaks within Content option.

Caution: At this time, this is not an officially recommended solution for creating accessible horizontal tables. JAWS 7.0 is not reading the accessibility tags for this solution as intended. JAWS 7.0 uses the physical location of the cells on the page to analyze the table structure. This document will be updated when this problem is resolved.

The form renders as follows.

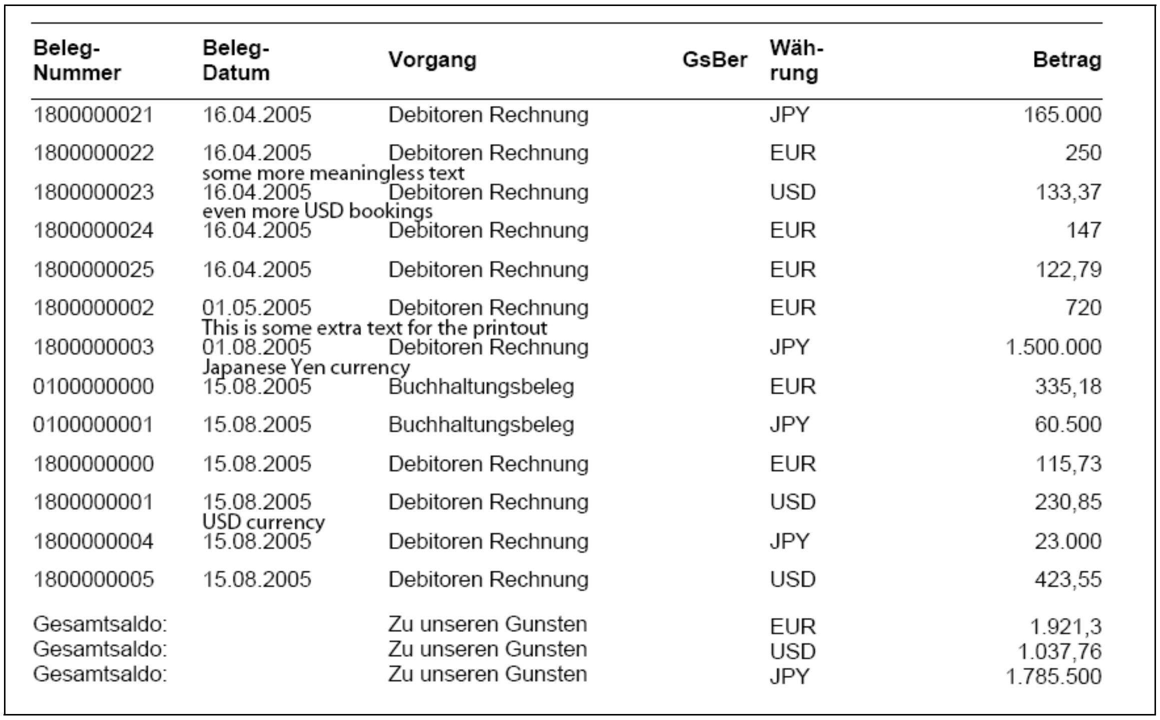


CARRID	DL	JL	JL	JL
CONNID	1984	0407	0407	0407
FLDATE	2004-10-11	2004-04-30	2004-04-30	2004-05-28
LUGGWEIGHT	18.3000	17.3000	9	25.4000
WUNIT	KG	KG	KG	KG
FORCURAM	422.94	881.00	595.00	961.00
FORCURKEY	USD	USD	GBP	EUR
ORDER DATE	2004-09-17	2004-04-07	2004-04-06	2004-05-04

CARRID	JL	JL	LH	LH
CONNID	0407	0408	0402	2402
FLDATE	2004-07-23	2004-06-26	2004-09-28	2004-04-03
LUGGWEIGHT	27.5000	11.7000	0	0
WUNIT	KG	KG	KG	KG
FORCURAM	961.00	1428.00	666.00	222.02
FORCURKEY	EUR	CHF	EUR	USD
ORDER DATE	2004-06-29	2004-06-03	2004-09-04	2004-03-10

Create Accessible Tables with Out-of-Place Rows

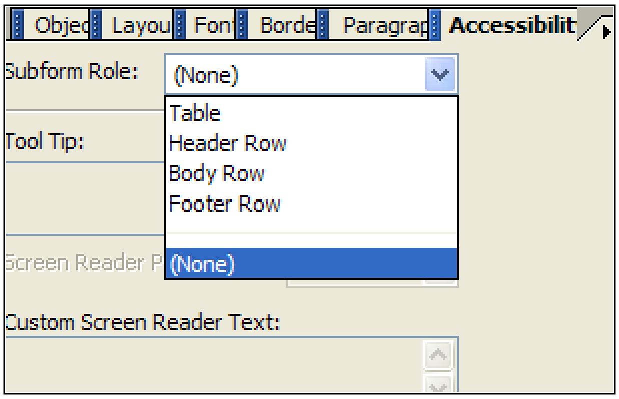
This example of a table with out-of-place rows has six columns. The seventh data field is comment text that appears in some of the body rows of the table.



Beleg-Nummer	Beleg-Datum	Vorgang	GsBer	Wäh-rung	Betrag
1800000021	16.04.2005	Debitoren Rechnung		JPY	165.000
1800000022	16.04.2005	Debitoren Rechnung		EUR	250
	some more meaningless text				
1800000023	16.04.2005	Debitoren Rechnung		USD	133,37
	even more USD bookings				
1800000024	16.04.2005	Debitoren Rechnung		EUR	147
1800000025	16.04.2005	Debitoren Rechnung		EUR	122,79
1800000002	01.05.2005	Debitoren Rechnung		EUR	720
	This is some extra text for the printout				
1800000003	01.08.2005	Debitoren Rechnung		JPY	1.500.000
	Japanese Yen currency				
0100000000	15.08.2005	Buchhaltungsbeleg		EUR	335,18
0100000001	15.08.2005	Buchhaltungsbeleg		JPY	60.500
1800000000	15.08.2005	Debitoren Rechnung		EUR	115,73
1800000001	15.08.2005	Debitoren Rechnung		USD	230,85
	USD currency				
1800000004	15.08.2005	Debitoren Rechnung		JPY	23.000
1800000005	15.08.2005	Debitoren Rechnung		USD	423,55
Gesamtsaldo:		Zu unseren Gunsten		EUR	1.921,3
Gesamtsaldo:		Zu unseren Gunsten		USD	1.037,76
Gesamtsaldo:		Zu unseren Gunsten		JPY	1.785.500

For more information about this example, see the attached form design, data, and PDF file (BP15.*).

To avoid the appearance of out-of-place rows, create the form design by using a set of nested subforms, and then add an extra subform for the comment row. In the Accessibility palette, set the accessibility role of the extra subform to None.

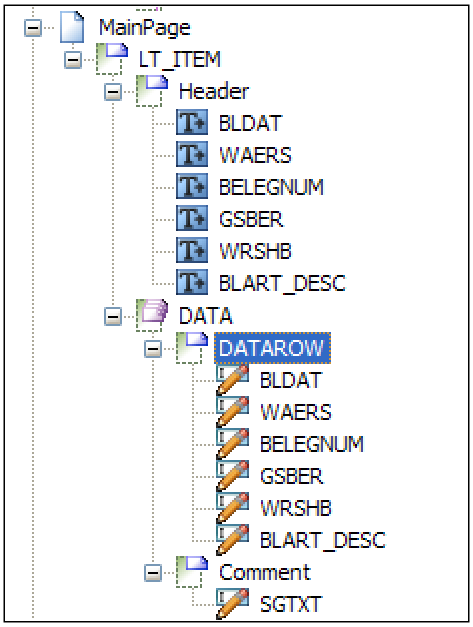


Subform Role: (None)

Using nested subforms is better than creating the form design as a wrapped multiline table because the screen reader software would read the wrapped table as a 7-column table, including the headers.

For an example of how to create a set of nested subforms, see the example in "Create Accessible Multiline Tables" on page 27.

The Hierarchy palette displays the subform structure.

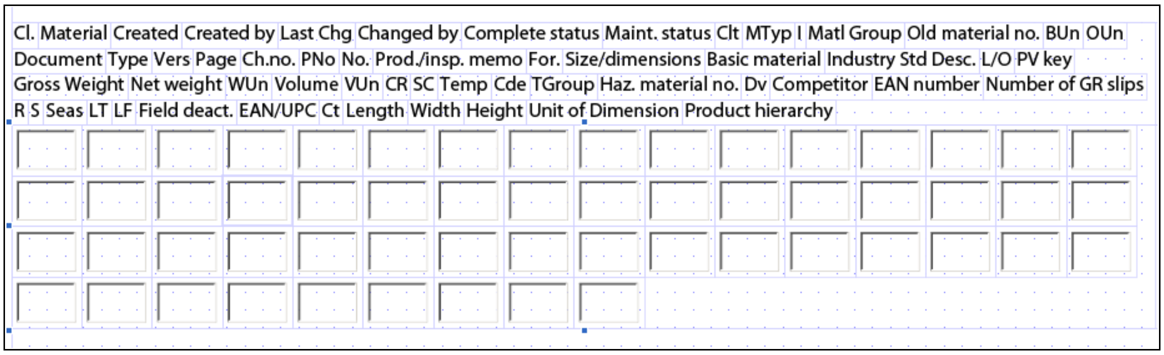


```
MainPage
  LT_ITEM
    Header
      BLDAT
      WAERS
      BELEGNUM
      GSBER
      WRSHB
      BLART_DESC
    DATA
      DATAROW
        BLDAT
        WAERS
        BELEGNUM
        GSBER
        WRSHB
        BLART_DESC
      Comment
        SGTXT
```

Create Accessible Tables with Multiple Data Fields

If you need to place more than 10 data fields in a table, see "Create Accessible Multiline Tables" on page 27. Because a large number of data fields are involved, the header and body subforms contain many rows.

Avoid creating such forms if the number of fields is large. The following example of a form that contains 57 data fields illustrates how such a form design may be very usable.



Cl. Material Created Created by Last Chg Changed by Complete status Maint. status Clt MTyp I Matl Group Old material no. BUn OUn
Document Type Vers Page Ch.no. PNo No. Prod./insp. memo For. Size/dimensions Basic material Industry Std Desc. L/O PV key
Gross Weight Net weight WUn Volume VUn CR SC Temp Cde TGroup Haz. material no. Dv Competitor EAN number Number of GR slips
R S Seas LT LF Field deact. EAN/UPC Ct Length Width Height Unit of Dimension Product hierarchy

Creating Multiple Levels of Repeating Headers

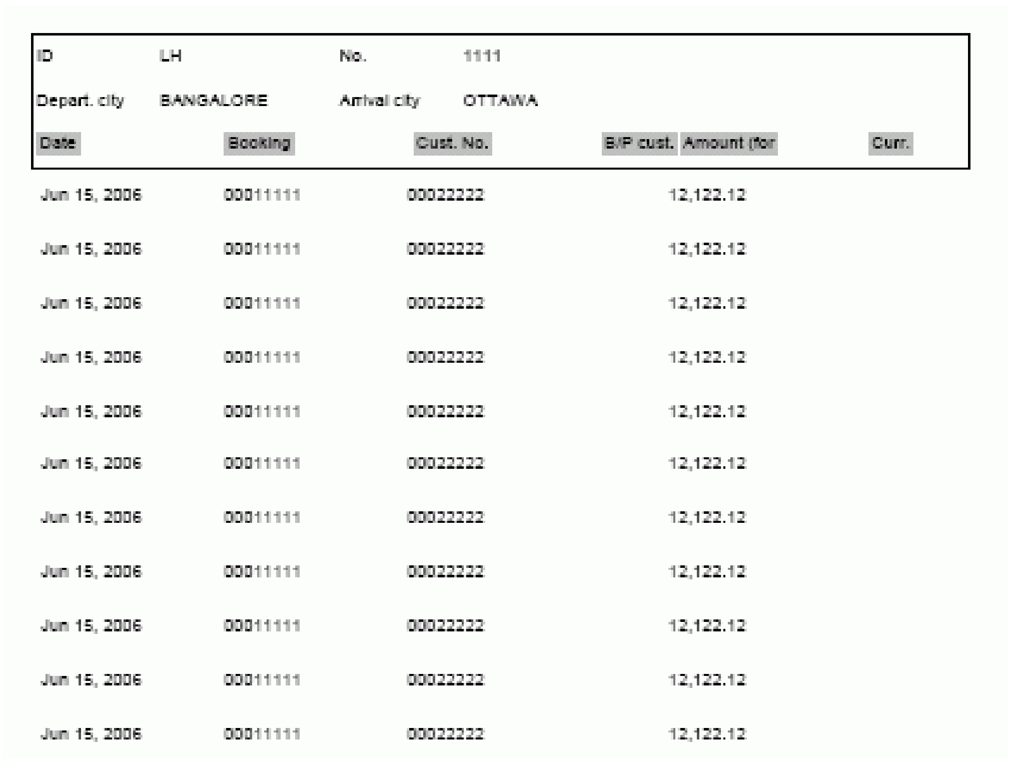
These guidelines use an example to explain how to create an accessible table that contains multiple levels of repeating headers for a nested set of tabular data.

Note: See the attached files for the complete solution used in this example:

mlrh_output.pdf, mlrh_template.xdp, and mlrh_data.xml.

This example explains the steps required to create the following accessible table. with multiple levels of repeating headers:

The objective is to produce an output as follows:

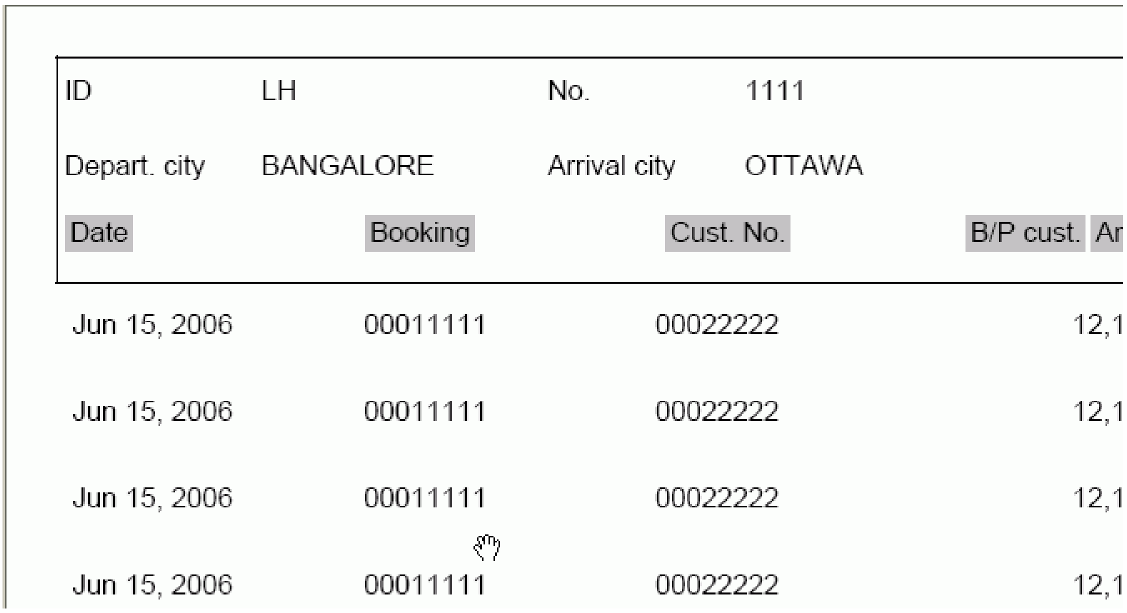


ID	LH	No.	1111
Depart. city	BANGALORE	Arrival city	OTTAWA

Date	Booking	Cust. No.	B/P cust.	Amount (for	Curr.
Jun 15, 2006	00011111	00022222		12,122.12	
Jun 15, 2006	00011111	00022222		12,122.12	
Jun 15, 2006	00011111	00022222		12,122.12	
Jun 15, 2006	00011111	00022222		12,122.12	
Jun 15, 2006	00011111	00022222		12,122.12	
Jun 15, 2006	00011111	00022222		12,122.12	
Jun 15, 2006	00011111	00022222		12,122.12	
Jun 15, 2006	00011111	00022222		12,122.12	
Jun 15, 2006	00011111	00022222		12,122.12	
Jun 15, 2006	00011111	00022222		12,122.12	
Jun 15, 2006	00011111	00022222		12,122.12	

Creating Multiple Levels of Repeating Headers

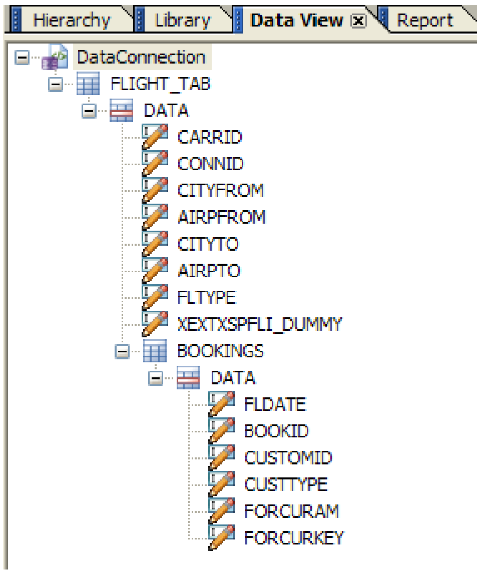
The objective is to show detailed information for each record of data. For example, in the report shown below, details about each flight appear beneath each record:



ID	LH	No.	1111
Depart. city	BANGALORE	Arrival city	OTTAWA

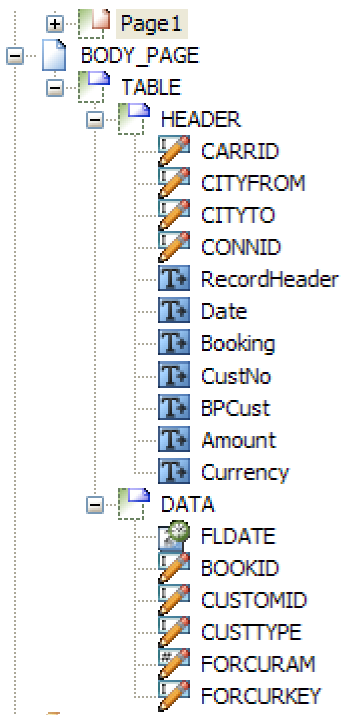
Date	Booking	Cust. No.	B/P cust.	Ar
Jun 15, 2006	00011111	00022222	12,1	
Jun 15, 2006	00011111	00022222	12,1	
Jun 15, 2006	00011111	00022222	12,1	
Jun 15, 2006	00011111	00022222	12,1	

This example assumes that a data schema is set up as follows:.



```
DataConnection
  FLIGHT_TAB
    DATA
      CARRID
      CONNID
      CITYFROM
      AIRPFROM
      CITYTO
      AIRPTO
      FLTYPE
      XEXTXSPFLI_DUMMY
      BOOKINGS
        DATA
          FLDATE
          BOOKID
          CUSTOMID
          CUSTTYPE
          FORCURAM
          FORCURKEY
```

The objective is to create a hierarchy as follows:



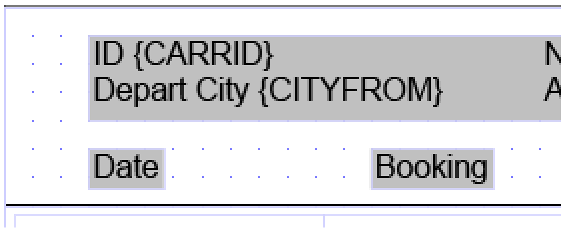
```
Page1
BODY_PAGE
  TABLE
    HEADER
      CARRID
      CITYFROM
      CITYTO
      CONNID
      RecordHeader
      Date
      Booking
      CustNo
      BPCust
      Amount
      Currency
    DATA
      FLDATE
      BOOKID
      CUSTOMID
      CUSTTYPE
      FORCURAM
      FORCURKEY
```

The “RecordHeader” static text field is a floating field containing the text fields (CARRID, CITYFROM, CITYTO and CONNID).

The steps to create this are as follows:

1. Name the main subform in the body page “BODY_PAGE” and set the “Content:” attribute in the Object palette to “Flowed”/”Top to Bottom”.
2. Create a subform called “TABLE” and set the “Content:” attribute in the Object palette to “Flowed”/”Top to Bottom”.
3. Create a child subform of “TABLE” called “HEADER”. Make sure that the “Content:” attribute in the Object palette is set to “Positioned”.
4. Create a child subform of “TABLE” called “DATA and set the “Content:” attribute in the Object palette to “Flowed”/”Western Text”.
5. Add the static text fields shown in the picture above to the HEADER subform. The first field is a floating text field called “RecordHeader”, which will contain the text fields CARRID, CITYFROM, CITYTO and

CONNID. The floating text field should be positioned so that it is directly above and aligned with the next field (as shown below).



6. Add the data fields shown in the picture above to the DATA subform. Note that there are 6 fields in the data subform and 7 in the header subform.

Notes:

A. You cannot convert the nested subforms to a table. The floating text field in the first cell in the HEADER subform will occupy a significant amount of space and there is no corresponding data

B. For screen reader software JAWS 7.1 (as of July 23, 2006), this solution will work properly. This is because JAWS examines the screen contents in addition to counting the elements in the hierarchy when determining the number of columns in the table. In this case, JAWS will report that this is a table with 6 columns, which is the intention of this design. However, this solution is dependent on the fact that future versions of JAWS will continue to behave this way, which is not certain at this time (July 23, 2006).

Preserving decimal digits when calculating

This example explains the steps that need to be done to preserve the number of decimal digits when performing calculations

Note: See the attached files for the complete solution used in this example:

preserve_decimal_places.xdp and preserve_decimal_places.pdf

Background:

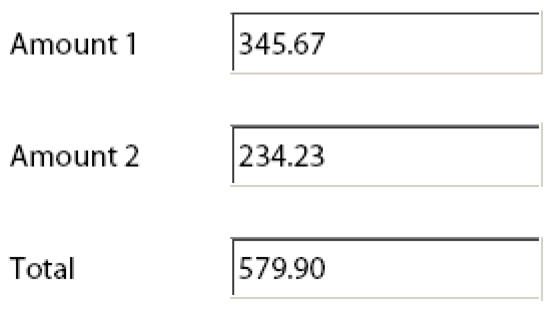
In order to format numeric fields, you can set a display pattern (such as "zz9.99") in the "Field" tab of the "Object" palette. This will ensure that a value such as "42.0" would be shown as "42.00"

This display pattern is also applied if the value is rendered in the "calculate" event.

However, if the number of decimal digits that need to be preserved is dynamic and only determined at run-time, then a script is required to format the data depending on the number of decimal digits are involved in the source value(s).

A typical use case is when it is necessary to calculate the totals for different types of currencies on the same form. For US dollars, two decimal digits are used. For Japanese Yen, there are no decimal digits.

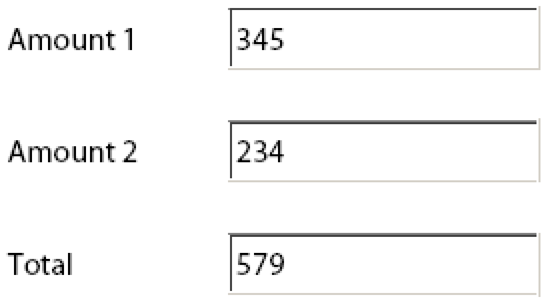
For example, in the example below, the total amount needs to use 2 decimal places.



Amount 1	345.67
Amount 2	234.23
Total	579.90

Without proper formatting, the total would appear as "579.9"

However, if the values had no decimal digits at all, then the value of the “Total” field needs to reflect this.



Amount 1	345
Amount 2	234
Total	579

In order to set the number of decimal digits, a script in the “calculate” event field is needed.

This script will perform the following steps:

1. Determine what the decimal separator character is for the form (e.g. “.” for US, “,” for DE, etc.)
2. Determine the number of decimal digits for one of the fields being used in the calculation.
3. If necessary, add trailing zeros to the target field.

The script for the “calculate” event for the Total field will look as follows:

This assumes that the fields being summed are an array with the name “fld”.

NOTE: The Calculate script needs a value to be the result of the script, so the resulting variable is the final line of the script.

```
var Total = sum(fld[*])

// Extract the character that is used to indicate the decimal place
// For example: For US locale, this is the period (".")
//              For German locale, this is the comma (",")
// This is done by extracting the character from a formatted
// preset value (9.9)
var chardec = substr(format("9.9",1.2),2,1)

// Compute the # of decimal places in the source field
var SrcValue = fld[0].formattedValue
//  now add in the decimal digits into the total
if (At(SrcValue , chardec) ne 0) then
    var DigitCount = Len(SrcValue ) - At(SrcValue ,chardec)
    //  format the total using the same # of decimal places
    Total = Ltrim(Str(Total,len(Total)+DigitCount+1,DigitCount))
endif

Total
```

Note that the script is written in FormCalc.

The JavaScript version is as follows:

```
// amt1 and amt2 are the fields involved
// amt1 will be the source field
var Total = Number(amt1.rawValue) + Number(amt2.rawValue);

// determine the number of decimal digits by subtracting the number of
// characters in the whole number version of the value from the number of
// characters in the original numeric value
// substract one to account for the decimal separator
var NumDigits = String(amt1.rawValue).length - String(Math.floor(amt1.raw-
Value)).length - 1;

Total.toFixed(NumDigits);
```

In the attached template (preserve_decimal_places.xdp/pdf), the “calculate” event of the “Total” field contains the script. The attached sample is an interactive form, so you can test out various input scenarios

Both the JavaScript and FormCalc versions are included on the attached form.

Fitting a Large Graphic

This example explains the steps that need to be done if you wish to fit a large graphic onto a page. If the graphic is too large for the page, then make the graphic as large as possible and auto-fit the graphic to fit on the page. Otherwise, render the graphic using the original dimensions.

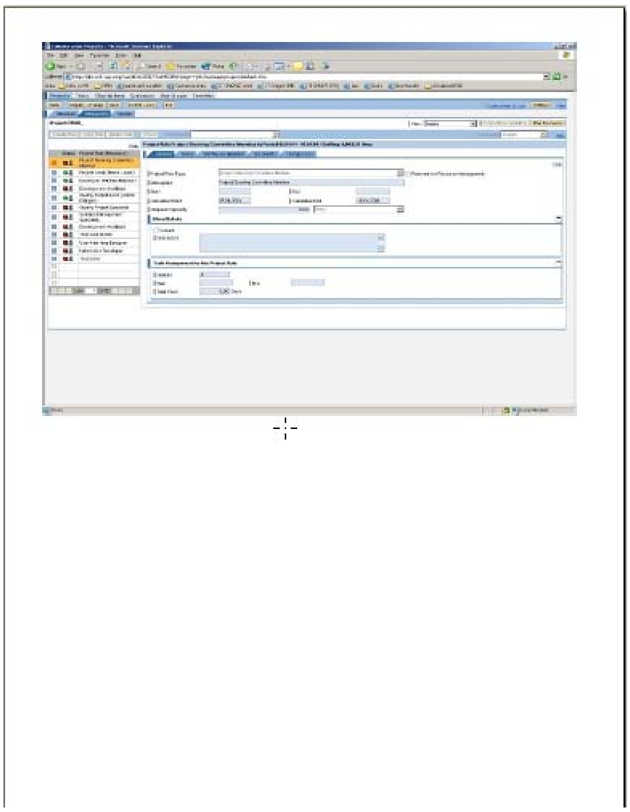
Note: See the attached files for the complete solution used in this example:

fit_large_graphic.xdp, fit_large_graphic.xml and fit_large_graphic.pdf

Background:

Auto sizing of the image fields is done by setting the "Sizing" drop down list to "Scale Image Proportionally" in the "Field" tab of the "Object" palette.

For example, in the example below, the image is set to "Scale Image Proportionally".



If the source of the image is XML data, then it will not be known in advance if the image is larger than the form.   If it is desirable to preserve the image size if it is smaller than the form, then a script will be needed to compare the image size with that of the form to determine if scaling needs to be done.

Solution:

In order to set the scaling on the image field, do the following:

1. Create two image fields, one that will remain invisible and contain the original image in the original size (GRAPHIC_HIDDEN) and one that will be visible (GRAPHIC_PRINT).
2. Make sure the parent subform of the image fields is a “positioned content” subform. This is the “Content:” drop down list on the “Subform” tab of the “Object” palette.
3. Place the following JavaScript to “run at client” in the layout_ready event of the GRAPHIC_PRINT image field:

```
// NOTES:
// 1.  The parent subform has to be a positional content subform in order
//     for this code to work.
// 2.  This routine has to be executed in the layout_ready event

// Stores the dimensions of the image fields
var lw1 = xfa.layout.w(GRAPHIC_HIDDEN, "mm");
var w2 = GRAPHIC_PRINT.w;
var lh1 = xfa.layout.h(GRAPHIC_HIDDEN, "mm");
var h2 = GRAPHIC_PRINT.h;

var lw2 = parseFloat(w2.substring(0, w2.length-2));
var lh2 = parseFloat(h2.substring(0, h2.length-2));

// Compares the dimensions of the dummy field with the print field
if ((lw1>lw2)||(lh1>lh2)) {
    // If greater, then change Sizing to "Scale to fit"
    GRAPHIC_PRINT.value.image.aspect = "fit";
} else {
    // If lower, then change Sizing to "Use Image Size"
    GRAPHIC_PRINT.value.image.aspect = "actual";
}

GRAPHIC_PRINT.rawValue = GRAPHIC_HIDDEN.rawValue;
```

The script compares the dimensions of the hidden graphic image field with visible image field. Since the visible image field is expanded to fit the width of the form and the parent subform is a positional subform, comparing these values will determine if the image is larger than the parent subform. If this is the case, the sizing of the image field is set to “scale to fit”.

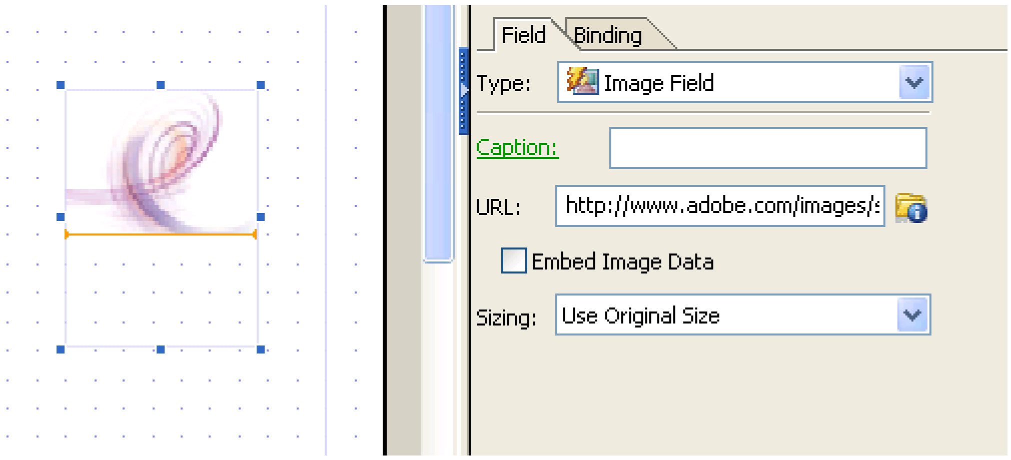
Creating a URL based read-only graphic

This example explains the steps that need to be done if you wish to place a graphic on the form whose source data is from a URL. Additionally, the sample explains how to disable the browsing dialog for the image if the form was to be rendered as interactive.

Note: The solution for this example is URL_readonly_graphic.xdp and URL_readonly_graphic.pdf

Solution:

A graphic whose source is a URL can be created by using an Image Field on the template. For example, see below:



Type: Image Field
Caption:
URL: http://www.adobe.com/images/:
Embed Image Data
Sizing: Use Original Size

For this example, a URL on the web that contains an image will be used. This URL could change in the future, but the steps in this example will still apply.

In order to create the image field, do the following:

1. Create an image field and place the URL of the graphic in the URL: field in the “Field” tab of the “Object” palette.
2. Place the following JavaScript to “run at client” in the form_ready event of the image field:

```
// Disables the image browsing dialog for an interactive form:
this.access = "readOnly";
```

The script will ensure that the field cannot be clicked on in the event that it is used as an interactive form.

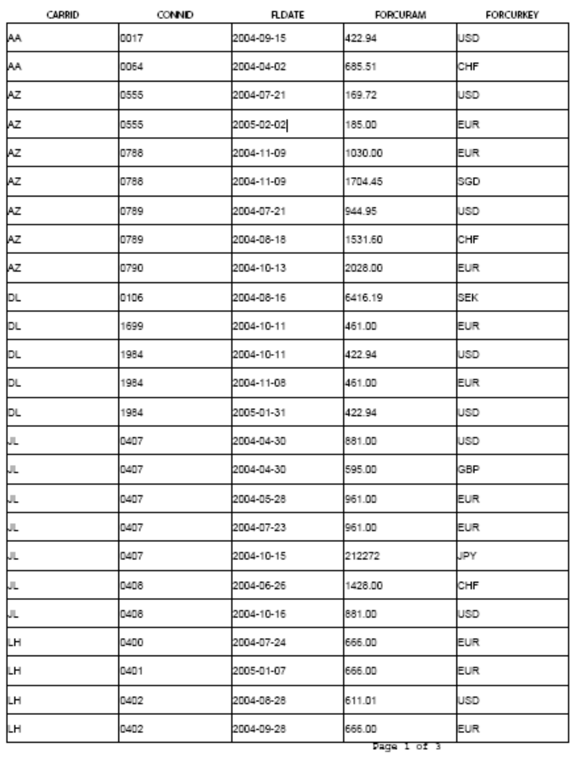
Setting Overflow Leaders/Trailers

This example explains how to create an accessible table that contains an overflow leader and trailer.

Note: See the attached files for the complete solution used in this example:

overflow_lead_trail.pdf, overflow_lead_trail.xdp, and overflow_lead_trail.xml.

The objective is to produce an output as follows:

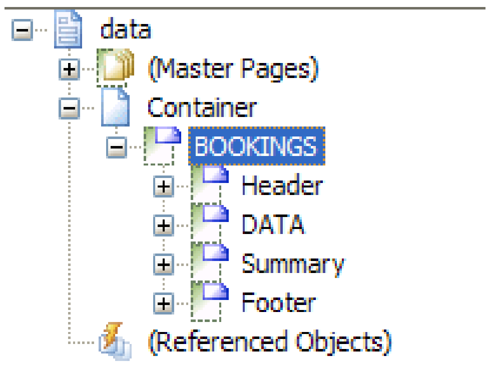


CARRID	CONNID	FLDATE	FORCURAM	FORCURKEY
AA	0017	2004-09-15	422.94	USD
AA	0064	2004-04-02	685.51	CHF
AZ	0555	2004-07-21	169.72	USD
AZ	0555	2005-02-02	185.00	EUR
AZ	0788	2004-11-09	1030.00	EUR
AZ	0788	2004-11-09	1704.45	SGD
AZ	0789	2004-07-21	944.95	USD
AZ	0789	2004-08-18	1531.60	CHF
AZ	0790	2004-10-13	2028.00	EUR
DL	0106	2004-08-16	6416.19	SEK
DL	1699	2004-10-11	461.00	EUR
DL	1984	2004-10-11	422.94	USD
DL	1984	2004-11-08	461.00	EUR
DL	1984	2005-01-31	422.94	USD
JL	0407	2004-04-30	881.00	USD
JL	0407	2004-04-30	595.00	GBP
JL	0407	2004-05-28	961.00	EUR
JL	0407	2004-07-23	961.00	EUR
JL	0407	2004-10-15	212272	JPY
JL	0408	2004-06-26	1428.00	CHF
JL	0408	2004-10-16	881.00	USD
LH	0400	2004-07-24	666.00	EUR
LH	0401	2005-01-07	666.00	EUR
LH	0402	2004-08-28	611.01	USD
LH	0402	2004-09-28	666.00	EUR

Page 1 of 3

Creating Overflow leaders/trailers

The assumption for this solution is that a form hierarchy exists as illustrated below (the attached solution files can be used in this case):



The "Summary" subform will appear at the end of the rendered form. The "Footer" subform will appear at the bottom of each page. The "Header" subform will appear at the top of each page.

The steps to create this are as follows:

1. Place the required fields in the Header, DATA, Summary and Footer subforms. The attached template solution already contains these subforms with the required fields. In this case, the BOOKINGS subform is the subform that represents the table. The "Container" subform is the parent of BOOKINGS.
2. Make sure that the "Container" and "BOOKINGS" subforms are "Flow Content" subforms, with the flow direction set to "Top to Bottom"
3. For the "BOOKINGS" subform, in the "Pagination" tab in the "Object" palette, set the "Overflow Leader" drop down list to "Header". Set the "Overflow Trailer" to "Footer". NOTE: the overflow leader and trailer have to be set in the parent subform of the leader/trailer, not in the affected subform (which in this case is the "DATA" subform).
4. For the "Header" subform, ensure that the "Content:" drop down list in the "Subform" tab of the "Object" palette is set to "Positioned". NOTE: This is an important step. If the header subform is not "Positioned", then the fields on subsequent pages are overlaid and not positioned properly.
5. For the "Header" subform, ensure that the "Allow Page Breaks within Content" checkbox in the "Subform" tab of the "Object" palette is turned off. NOTE: This is an important step. For simple one line headers, this may not have impact. However, if the header subform contains other subforms which may overflow onto another page, it is important to turn off the "Allow Page Breaks within Content" setting to avoid creating duplicate headers.

6. Add the data fields shown in the picture above to the DATA subform. Note that there are 6 fields in the data subform and 7 in the header subform.

7. For the “Summary” subform (the subform that will appear at the end of the rendered output), make sure that it appears in the hierarchy right after the main body of the table (the “DATA” subform). The reason for doing this is to ensure that the subform will appear at the end (after the multiple instances of the “DATA” subform have been rendered).

8. For the “Footer” subform, ensure that the “Repeat Subform for Each Data Item/Min Count” is set to ‘1”. This field is in the “Binding” tab of the “Object” palette. Also make sure that the “Content:” drop down list in the “Subform” tab of the “Object” palette is set to “Positioned”.

The rendered form will now contain the “Header” and “Footer” subforms at the top and bottom of each page. The “Summary” subform will appear on the last page.

Important Notes:

A. The overflow leader and trailer have to be set in the parent subform of the leader/trailer, not in the affected subform. Although the drop down list for overflow leader and trailer is available and will work in some situations, it is strongly recommended to set the overflow leader/trailer using the parent subform.

B. The header subform must always be set up as a “Positioned” content subform and the “Allow Page Breaks” option should be off. If these settings are not correct, the rendered output will contain errors such as duplicate headers or overlaid fields.

Printing Duplex Forms

This chapter explains how to generate duplex (i.e. two sided) forms.

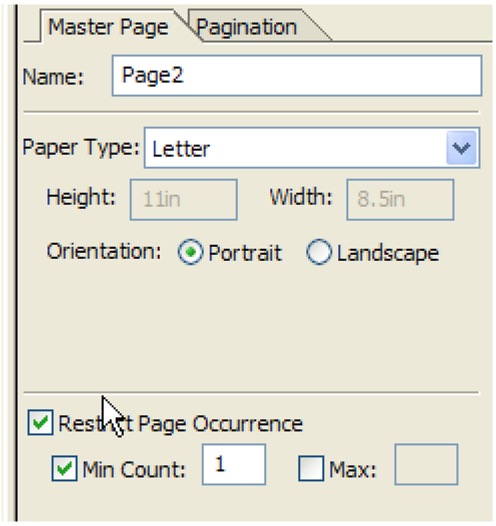
Note: See the attached files for the complete solution used in this example:

Note: This feature is only available in Designer 8/Adobe Reader 8.

Creating a duplex form

You can render an existing form by setting the <pagination> element on the XFA.XCI file on the server.

Traditionally, the only option for designing master pages was to use ordered occurrences. This meant that master pages were laid out in sequence. Repetition was controlled by setting the min/max value of the pages.



Name: Page2

Paper Type: Letter

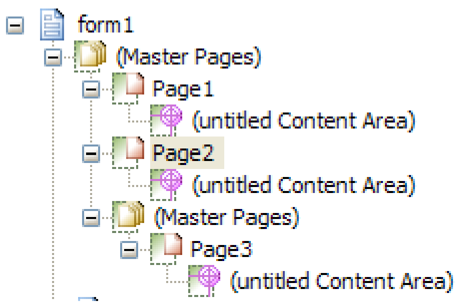
Height: 11in Width: 8.5in

Orientation: [✓ Portrait] Landscape

✓ Restrict Page Occurrence

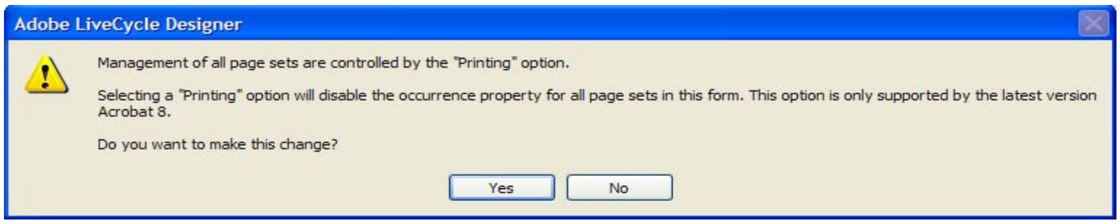
✓ Min Count: 1 ☐ Max:

A sequence of pages (AAA, AABBAA, etc.) could be created using this property. Pages could be grouped in sets which can be combined in ahierarchy.



In order to create duplex forms, a new method of designing master pages must be used. This new method is called "explicit pagination".

To begin using "explicit pagination", select the "(Master Pages)" root element in the hierarchy and then select the "Page Set" tab in the "Object" palette. Change the "Pagination:" drop down list to "Print on Both Sides". When you do, the following dialog will appear:


Adobe LiveCycle Designer

Management of all page sets are controlled by the "Printing" option.

Selecting a "Printing" option will disable the occurrence property for all page sets in this form. This option is only supported by the latest version Acrobat 8.

Do you want to make this change?

Once you click on "Yes", the template cannot revert back to using ordered occurrences for the master pages.

To define a set of master pages within a page set, set the attributes for odd/even and placement. These options give you the choice of defining the first page, last page, remaining page(s) as well as which master page is used on the front (odd) and back (even).

Note: If you have more than one page set, you must explicitly move to a new master page using either a page break or conditional page break.