Best Practices Document for Designing Forms

Version 0.4

Adobe XFA Training Materials

Table of Contents

Document History:

Version Number	Date Released	Comments
0.1	August 16, 2005	First release
0.2	September 26, 2005	Added examples of how to design accessible tables and incorporated suggestions from various colleagues and SAP personnel.
0.3	November 18, 2005	Added additional examples and made minor updates
0.4	December 5, 2005	Updated with comments from SAP

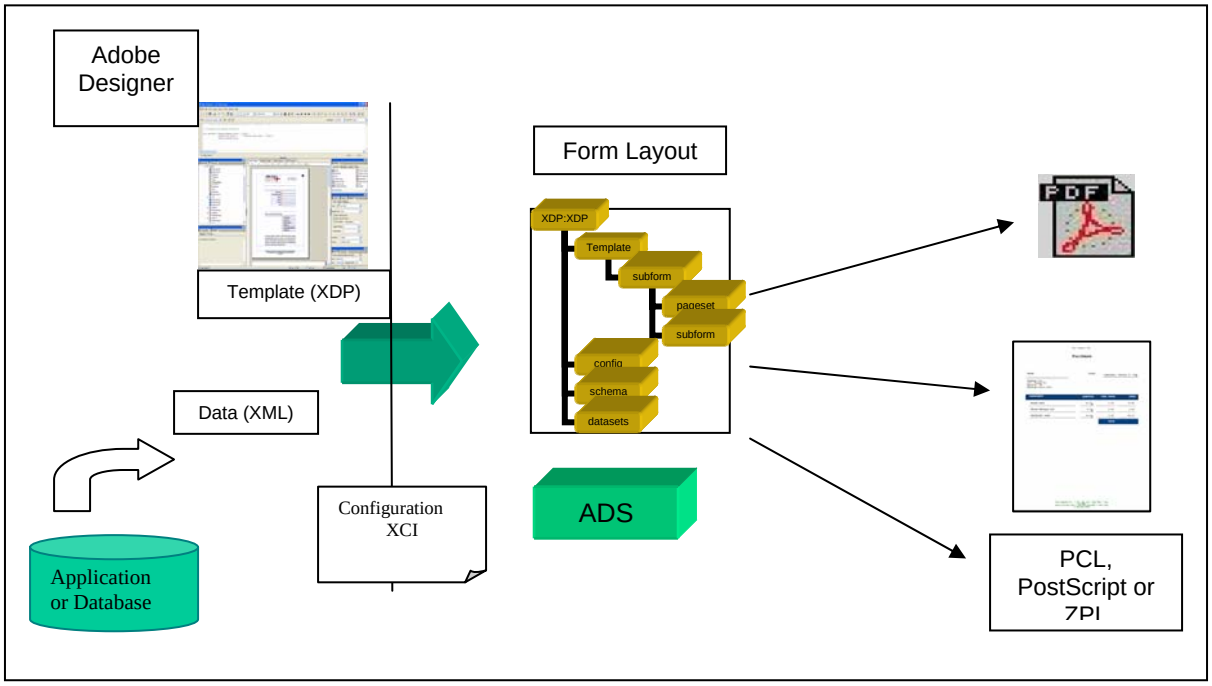
The latest version of this document can be accessed from the Adobe ERoom. The URL for this document is:

https://eroom.adobe.com/eRoom/fid302/SAPAdobe/0_7b81
The ERoom location is:
SAP/Adobe Alliance -> Trollinger/Package 2B -> Trollinger Package 2B Documents of Record -> Document Library

Overview

The purpose of this document is to outline the best practices to follow when designing forms using Adobe LiveCycle Forms Technology. An explanation of the software components that are used to create and render a form is the best way to start to understand the practices outlined in this document.

As you can see from the diagram below, a form consists of a set of XML files that are combined by the Adobe Document Services. In the same way that it is not necessary to understand the mechanics of an automobile to drive one, it is not necessary to view or edit XML files in order to use the Adobe LiveCycle Products. However, possessing a good understanding of the operations taking place behind the scenes is essential if you wish to become a good form designer.



Software Components

Name	Description
Adobe Designer	Adobe LiveCycle Designer – a Graphical User Interface application for creating template (XDP) files
ADS	Adobe LiveCycle Document Server – takes a template, XML data and configuration settings and renders the form to PDF, PCL, PostScript or ZPL.
Application or Database	An application or database that exposes data via XML

Steps to render a form:

Step	Description
Define XML Data	Decide on what data structure will be exposed by the application or database and used by the template
Design template	Drag and drop fields in Adobe LiveCycle Designer and design the template. The template (an XML file typically referred to as an XDP file) is stored on a server and used in the rendering process.
Render the form	Submit the template (XDP), data (XML) and configuration settings (XCI) to the Adobe Document Services (ADS). ADS will build an output independent version of the form and then convert this form layout to PDF, PostScript (PS) or PCL as required.

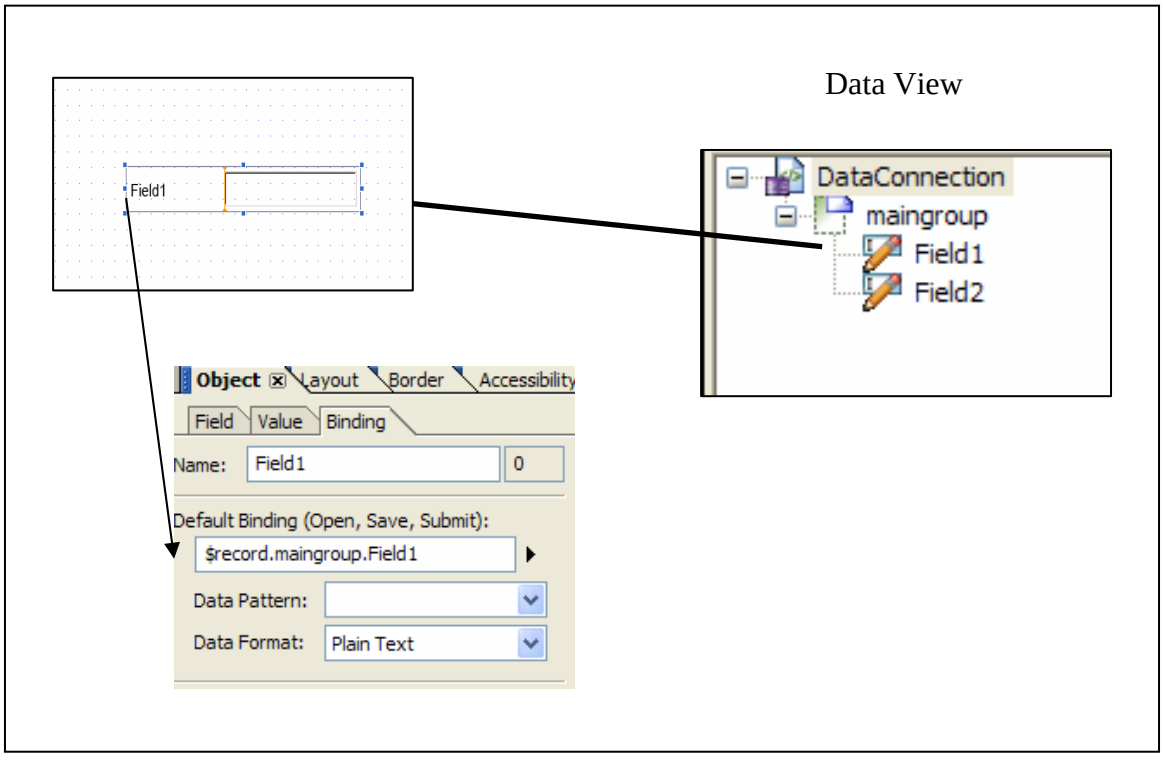
Best Practice #1: Use Relative Binding Whenever Possible

Relative versus Full Binding

In order for data to appear in a form, there needs to be some kind of binding. There are two ways to bind data to a template: relative binding and full binding. Note: Relative binding was referred to as "implicit" binding and full binding was referred to as "explicit" binding in the past.

Full Binding

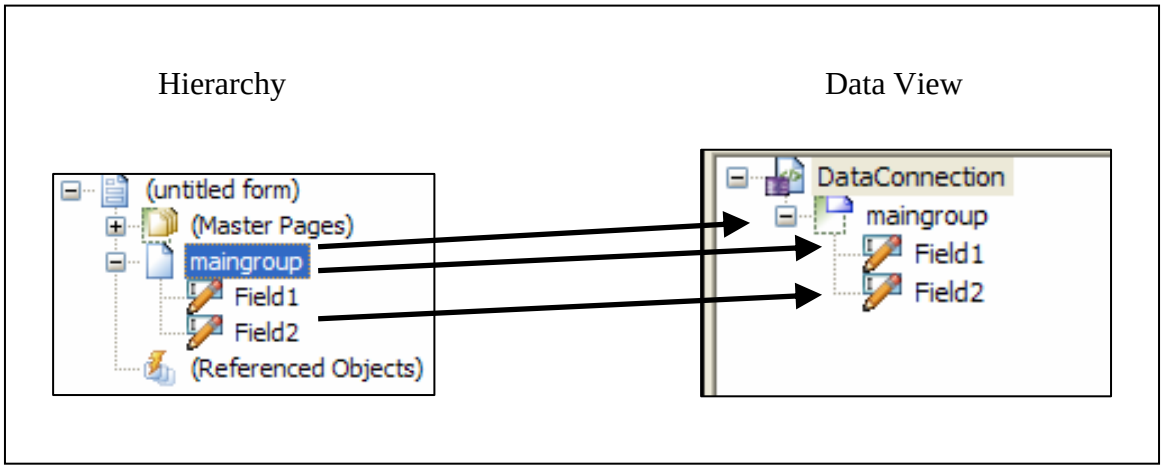
When you drag an individual field from the Data View onto the template, it creates a full binding to the field. For example, if you drag "Field1" from the Data View shown below, the field on the template will have a binding of:"$record.maingroup.Field1". This is called full binding.



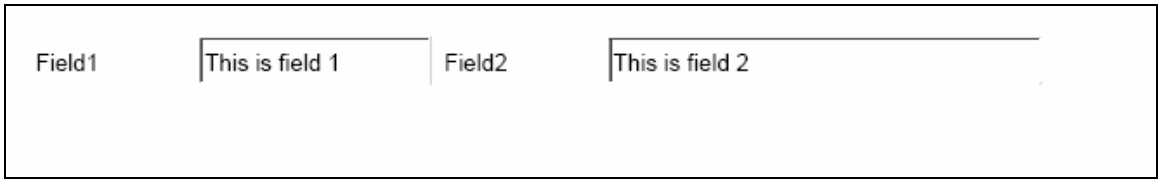
Relative Binding

Relative binding matches the names of the elements in the XML data file with the corresponding field names in the hierarchy of the template.

For example, if a template has the following hierarchy and the XML data source has the following structure:



Note that the fields in the hierarchy map to the same elements in the data view. The rendered form would then contain the field values as follows:



The binding value for the "maingroup" field is: "$record.maingroup". All other fields automatically bind to the corresponding XML elements.

To create fields with relative binding, drag a group of fields onto a form.

The reason relative binding is better than full binding is due to the amount of processing that has to be done by the ADS when merging the data onto the form. When a multiple level expression such as "$record.group.field" has to be processed by ADS, it takes time to traverse the DOM (Document Object Model) in order to find the right node. The fewer number of full binding statements on a template, the less time it will take to merge the XML data into the Form DOM.

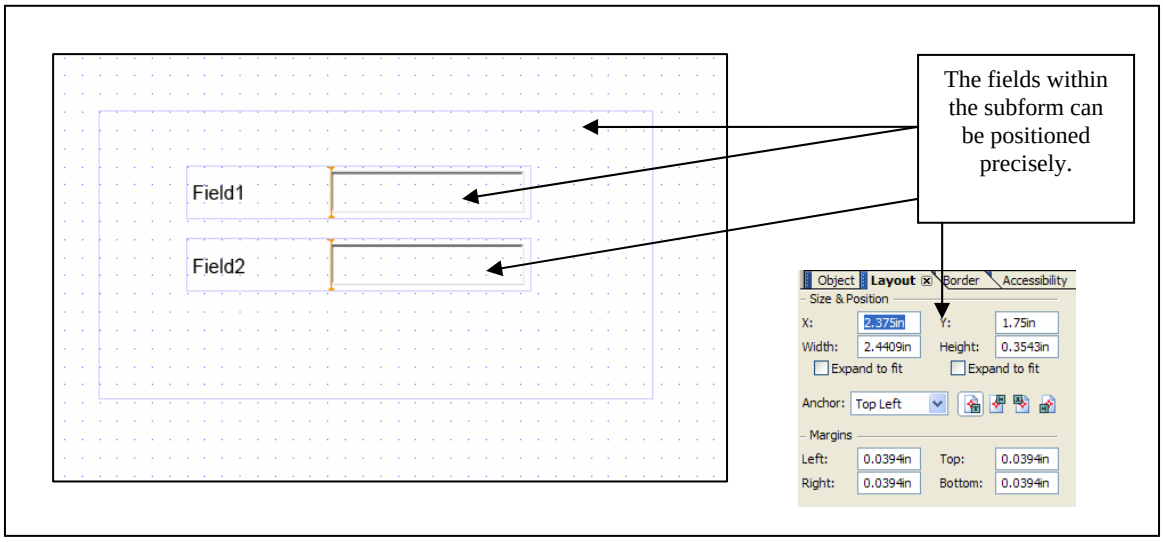
NOTES:

1. Using a lot of subforms can degrade performance, so it is best to avoid this where possible.
2. It is not possible to mix both relative and full binding on the same form.

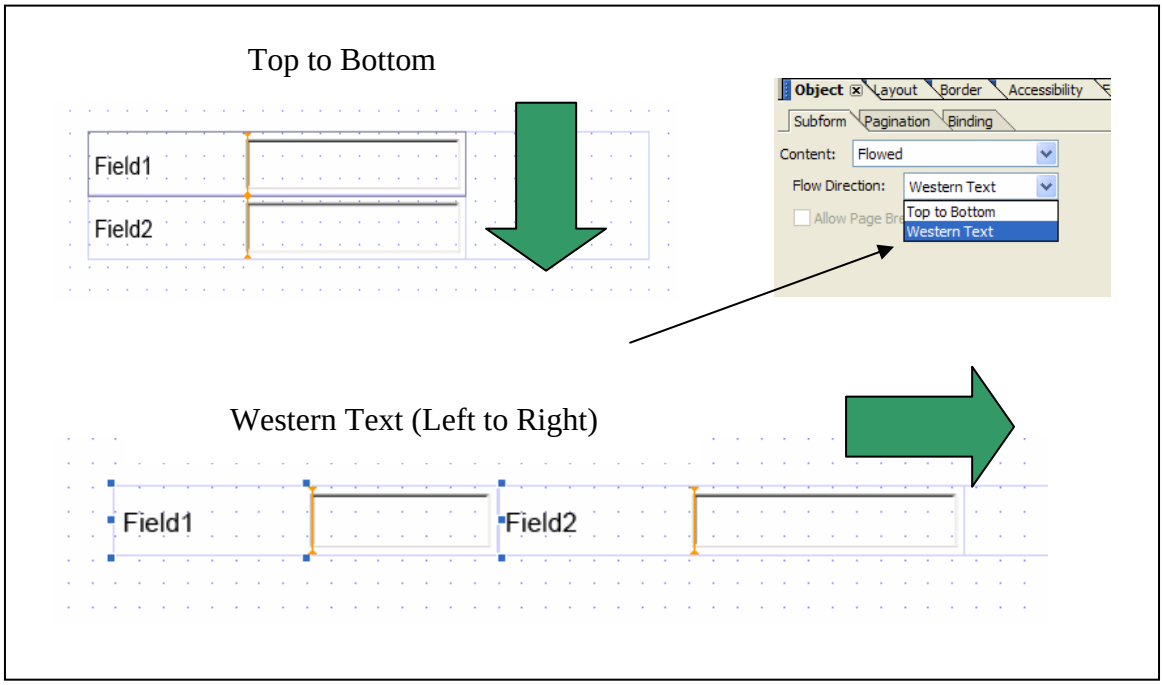
Best Practice #2: Use Flow Content Subforms whenever possible

Subforms are used to contain other objects such as fields. They can either be configured to permit exact positioning of the contained elements ("Positional Content") or to automatically arrange the contained elements ("Flowed Content") in one of two directions ("Top to Bottom" or "Western Text").

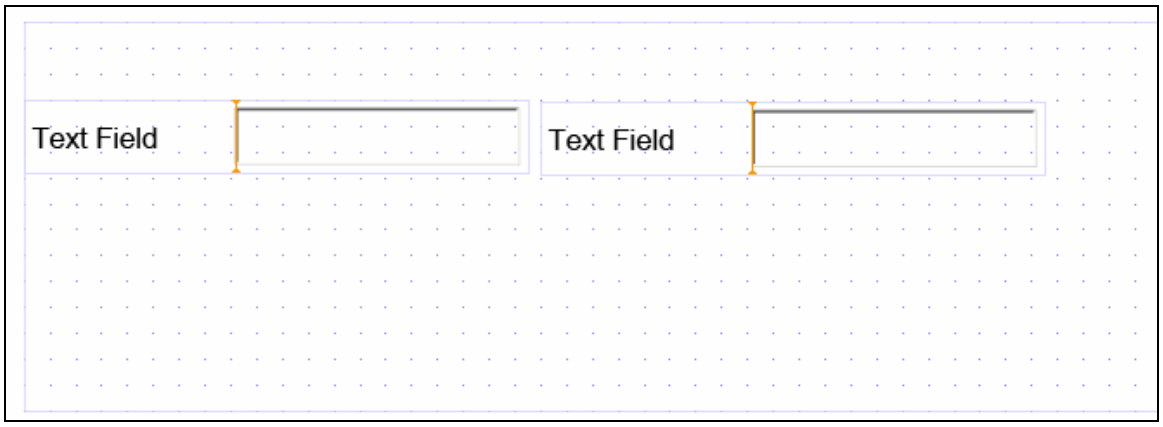
Positional Content Subforms



Flow Content Subforms

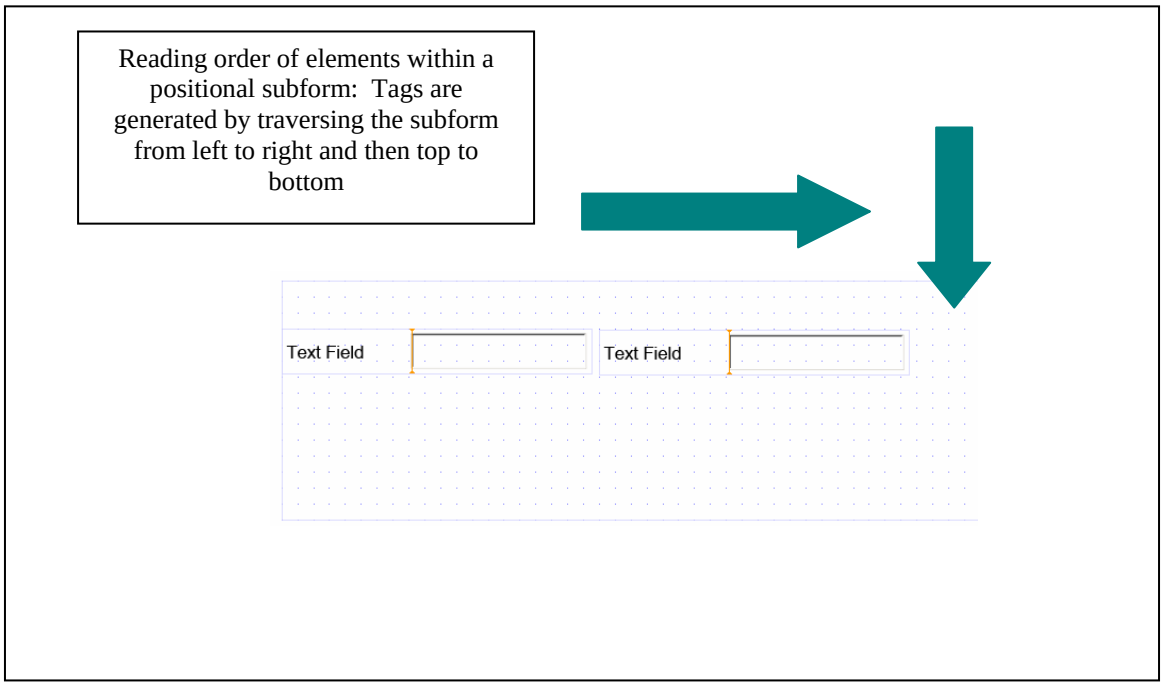


When designing forms, it is best to arrange fields inside subforms and use the "flow content" property to arrange the individual fields. By using this property, you are allowing the layout engine in the ADS code to take care of the exact positioning. This avoids problems when adjacent elements have different co-ordinates and results in accessibility or navigation problems when the form is tested. For example, consider the template below:



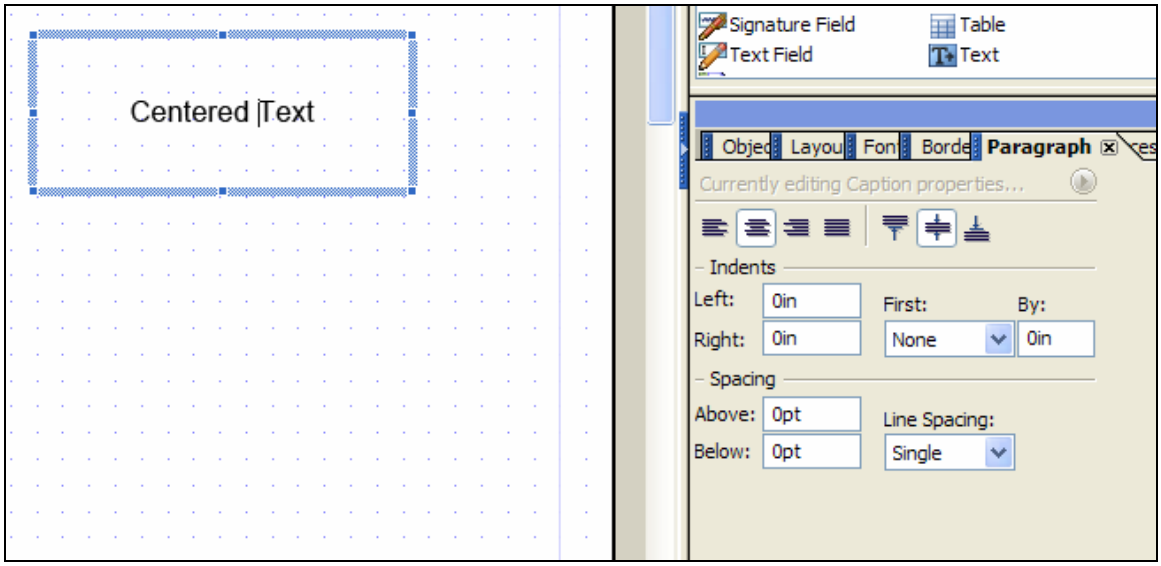
These text fields are inside a "positional" subform and appear to be aligned vertically. But they are not. The vertical heights differ by 1/10 of an inch. Since accessibility tags are generated in positional subforms based on "left to right, top to bottom", the text field

on the right will be read by the screen reader before the left text field, which is not what the requirement is in this case.



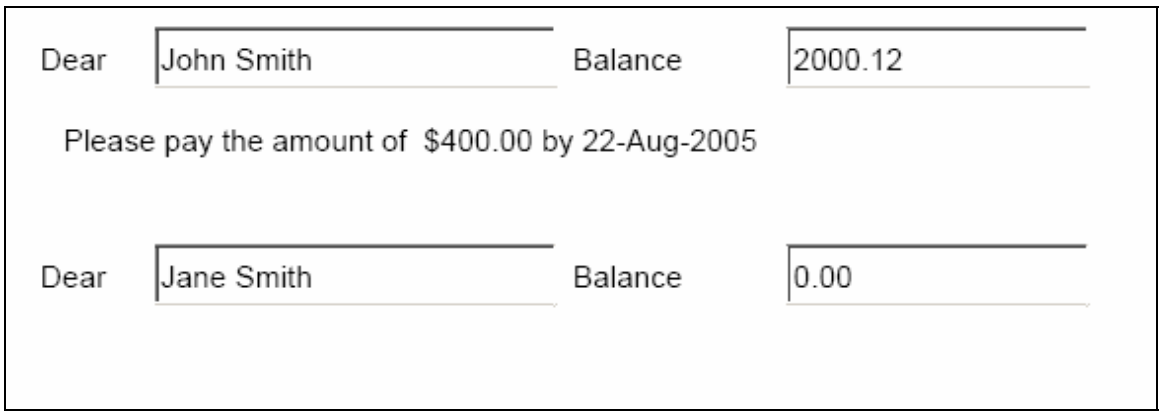
The better design is to enclose these two text fields in a “flow content” subform. If space is required between the fields, then use the margin values in the “Layout” palette.

To position captions and text, you can use the paragraph property sheet for a text field to align text as centered, left justified, etc.



Best Practice #3: Show or Hide Fields without Scripting

Consider the following form design problem:



If the customer does not owe any amount, then the line that begins with "Please pay..." should not appear.

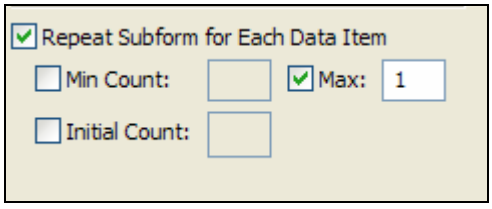
This can be designed using a line of script that will check with balance and selectively hide or show the "Please pay…" field. This script would look as follows:

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

However, if the form is processing a large number of records, this is not the most efficient solution.

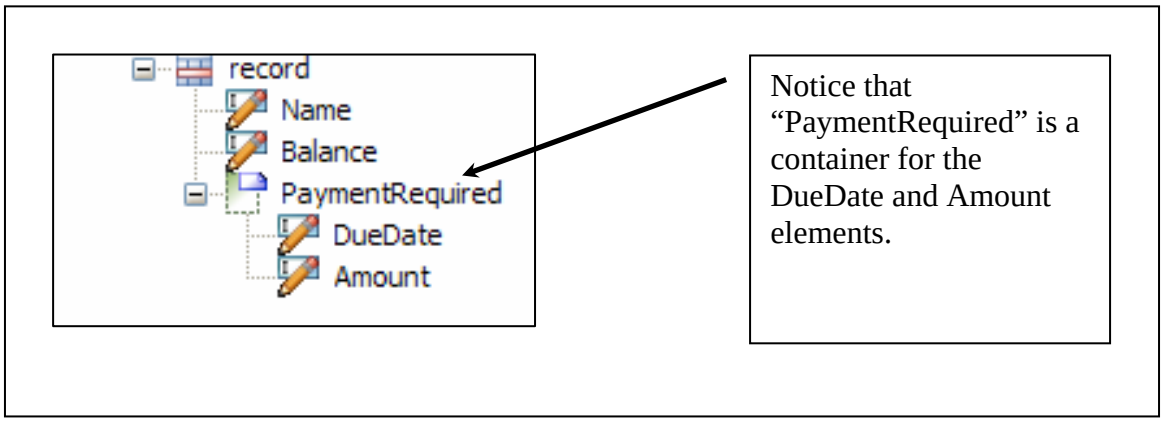
Repeatable Subforms

A better approach is to structure the XML data so that the fields which may not be present are contained within a parent element. These elements can then be mapped to a subform whose binding property page will look like the following (this is referred to as a repeatable subform):

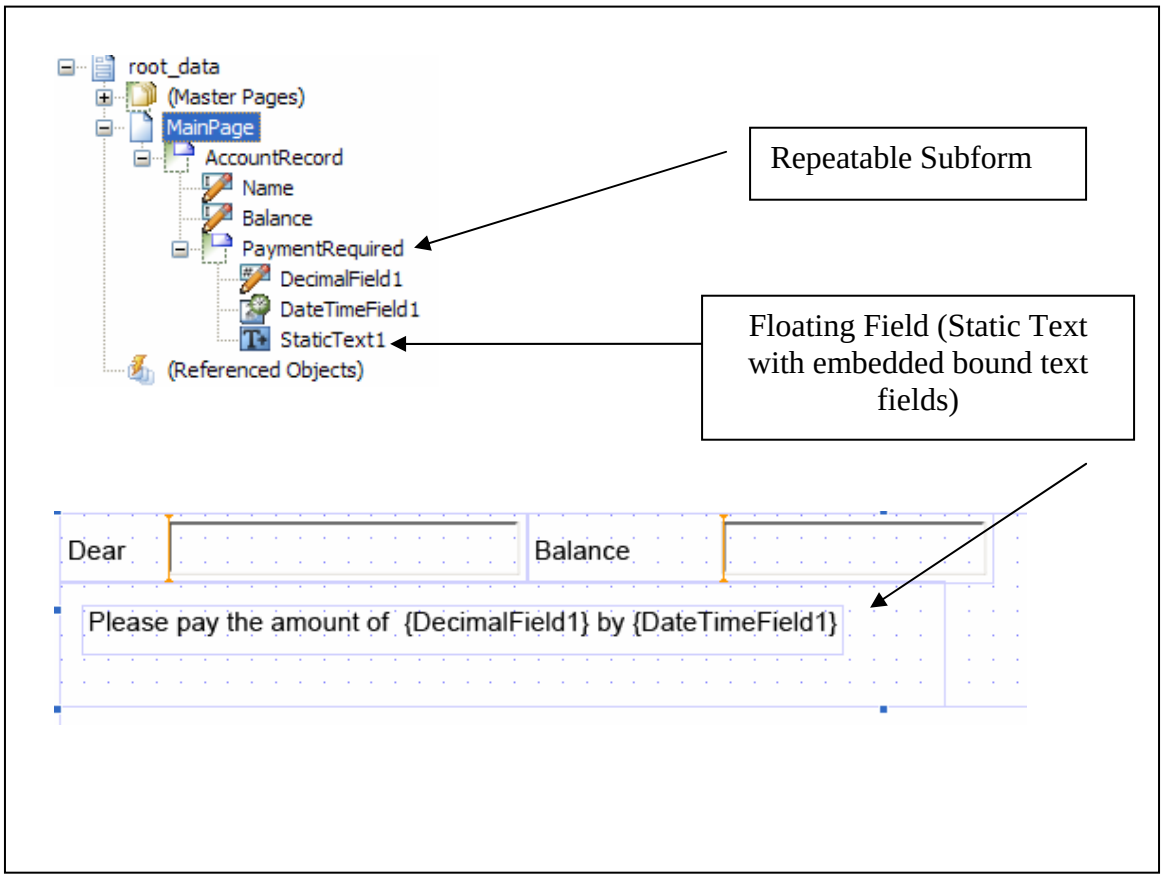


This will indicate that the subform will repeat a minimum of zero times and a maximum of 1. If there is no XML node in the data stream, the subform and all of the contained fields will be hidden.

If the XML data for this form can be structured as follows:



Then the template can be designed as follows:



Best Practice #4: Use Subforms to Structure Form Content

Subforms are used to organize the content of a form. According to the Adobe LiveCycle Online Help:

"A subform provides anchoring, layout, and geometry management for objects. The objects in a subform can be arranged in rows, columns, or some other kind of balanced arrangement."

In Adobe LiveCycle Designer, tables are "flow content" subforms which contain a repeatable subform with data (e.g. "a body row"), ensure that the heights and widths of adjacent subforms remain consistent and also export accessibility information (see Best Practice #7: Ensure Forms are Accessible).

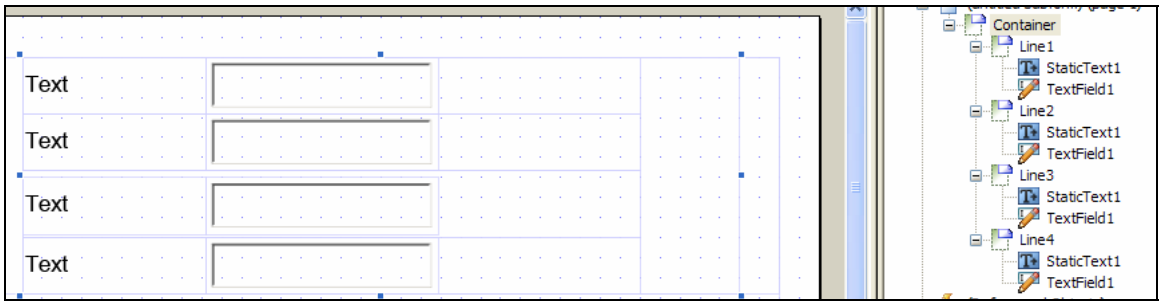
Groups of subforms can be used to structure content as well, but tables are the best approach, particularly if the content originates from a relational database and contains multiple record types that are repeated or if there is nested data.

If tables are used, the screen reader software will read out the structure as a table (e.g. "table with 4 columns, etc.") which may not be desirable. It is possible to create a form that resembles a table in appearance, but is simply a set of nested subforms. To see details on how this done, look at the example in Best Practice #11 - Designing an Accessible Multi-line Table. If you do not set the "Accessibility Role" for the subforms in this case, you can structure your form content without the screen reader announcing the form content as a table.

For example, consider the following 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

This can be designed as a set of subforms, using a hierarchy as follows:



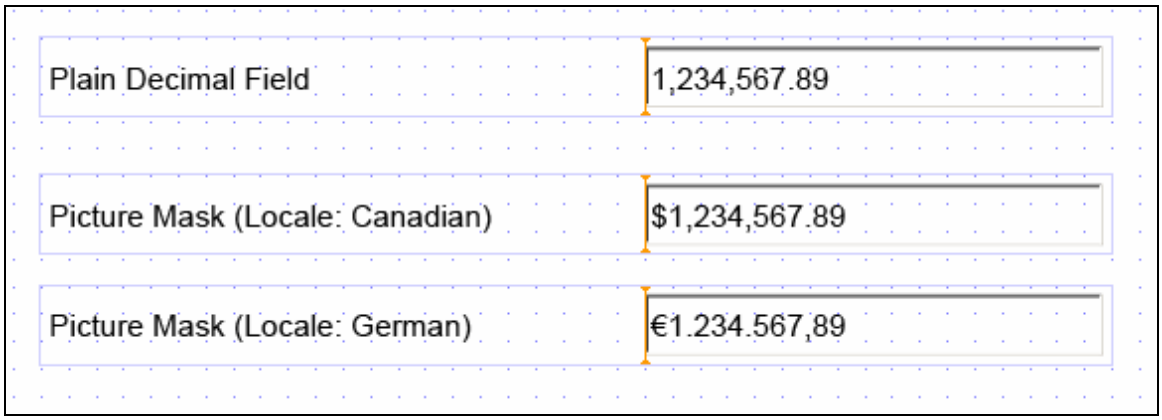
This structures the content and guarantees the order that will be read by the screen reader.

Best Practice #5: Use Picture Masks for Universal Formatting

Picture Masks

Picture masks are strings of characters that can be used to format the display of data, for editing a field in an interactive form or when XML data is being bound to a field. Picture masks are only appropriate as long as the result is correct with regard to all locales involved.

For example:



All three fields contain the same value “1234567.89”.

The second and third fields in the example above use a picture mask of “$zzz,zzz,zz9.99”. The “$” sign is for the currency symbol, using the locale of the object. Notice that the “$” symbol is mapped to the Euro symbol if the locale of the field is German.

For more details on what can be contained in a picture mask, see “Patterns” in the Adobe LiveCycle online help file.

BEST PRACTICES NOTE:
Picture clause contents are locale independent. For example, if the decimal separator character (“.”) is used in a picture mask (e.g. “99.99”), the rendered character will depend on the locale of the form. This means if the locale is German, then the comma (“,”) will be used as the decimal separator character.

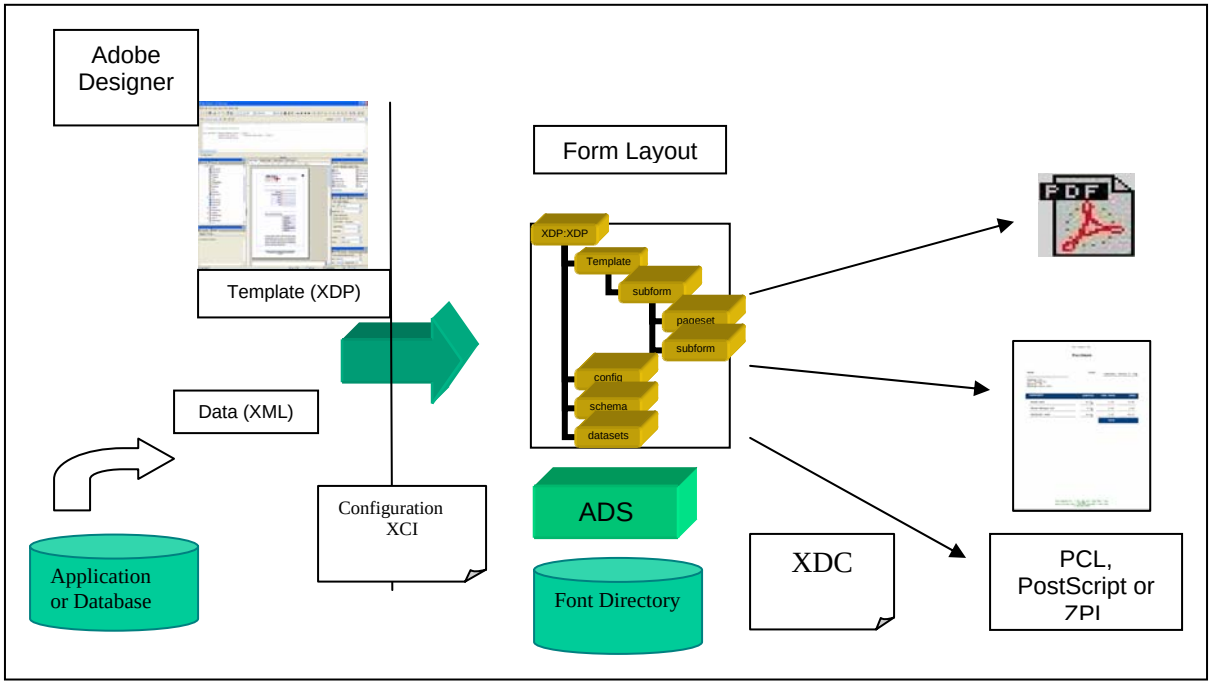
Determining the locale of the form

In the above example, the locale of the field is determined by the following:

A. Locale property of the field (if set)
B. Locale property of the form (if set)
C. The ambient locale (the locale of the machine that is used to render the form).
   NOTE: The ambient locale is also known as the viewer locale.

Best Practice #6: Use Fonts that ensure accurate output and minimal output size

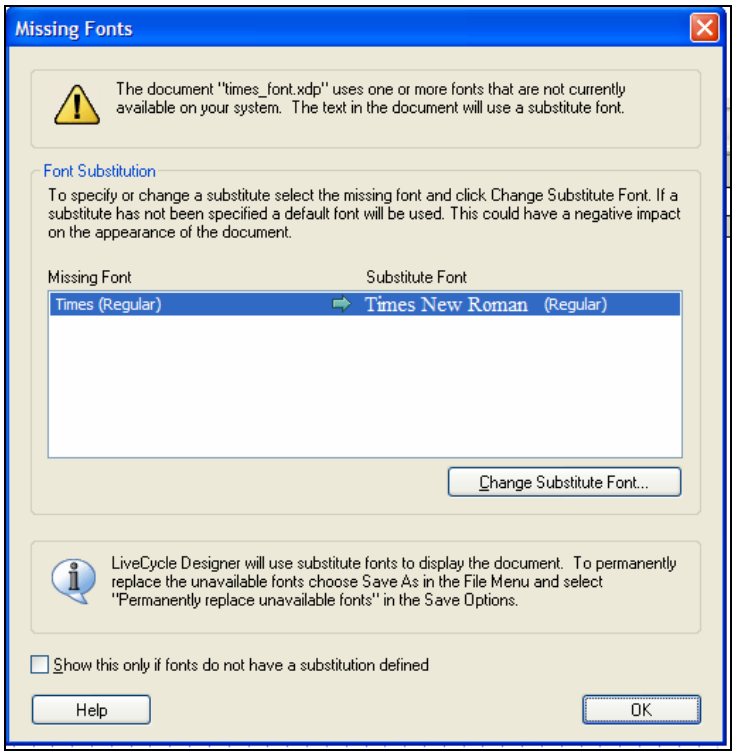
Overview of Font Selection



The following is a list of steps that ADS uses when a particular font is used in a template and rendered in a form:

Step	Description
Font selection by the Form Designer	Adobe LiveCycle Designer displays the list of fonts that are installed on the desktop machine of the person designing the form. All of the fonts shown are guaranteed to look the same on both the screen and when printed.
Template is rendered by ADS	For a particular output target (e.g. PDF, PCL or PS), ADS uses an XML file called XDC to determine what fonts are supported for a particular output target. The Configuration file (XCI) also indicates how to map fonts if required.
Fonts are embedded in the output stream as required by ADS	If a particular output target does not support a particular font, then ADS can embed all or part of the font as part of the output stream. The ability to embed a font is controlled by a setting in the configuration (XCI) file.

The form designer can encounter the following dialog when opening a form:



This indicates that the template (XDP) contains the name of a font (in this case: “Times”) that is not recognized by Adobe LiveCycle Designer. For this particular form, the form designer has chosen “Times” because they plan to render a form to a PostScript printer which contains the “Times” font (e.g. “Times” is a printer resident font). In some cases, a form has to use this font for legal purposes.
Since Adobe LiveCycle Designer cannot accurately show the “Times” font, it will display this dialog and ask the user to select a suitable font to be used as a substitute. Note that the template is not modified in this case.

Best Fonts to Ensure Output Target Support

In order to guarantee that the selected output target will be able to render the form using the font that the form designer intended, here is a list of fonts that ADS supports that are also supported by the various output targets (PDF, PCL and PostScript):

PDF	PCL	PostScript
Courier	Courier	Courier
Helvetica	Helvetica	Helvetica
Times	Times	Times
Arial	Arial	Arial
Times New Roman	Times New Roman	Times New Roman

Embedded Fonts and Output Stream Size

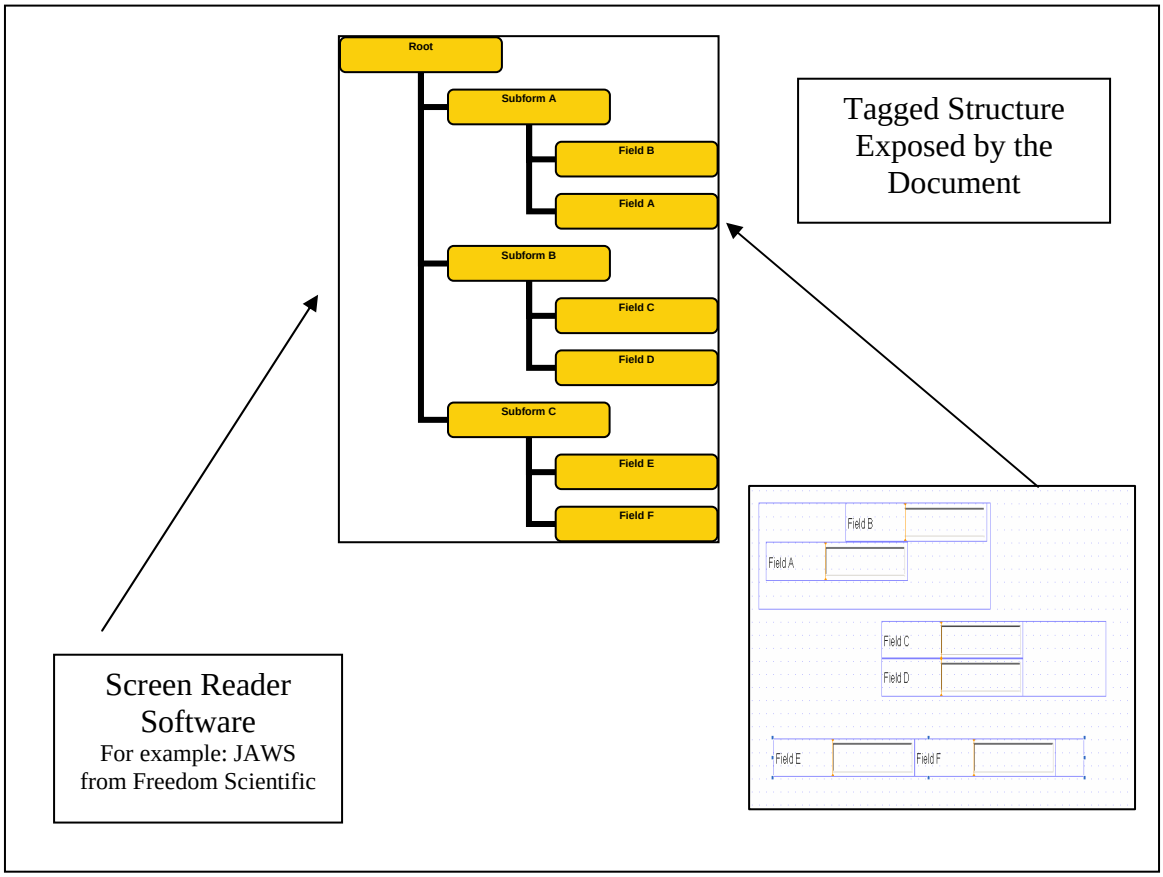
If an output target does not support a particular font, then ADS can embed the font as part of the output stream. This will obviously enlarge the size of the output stream. In order to minimize the size of the output stream, ADS will only embed a subset of the font used by the rendered form. These settings are controlled within the ADS configuration (XCI) file.

Best Practice #7: Ensure Forms are Accessible

Overview of Accessibility

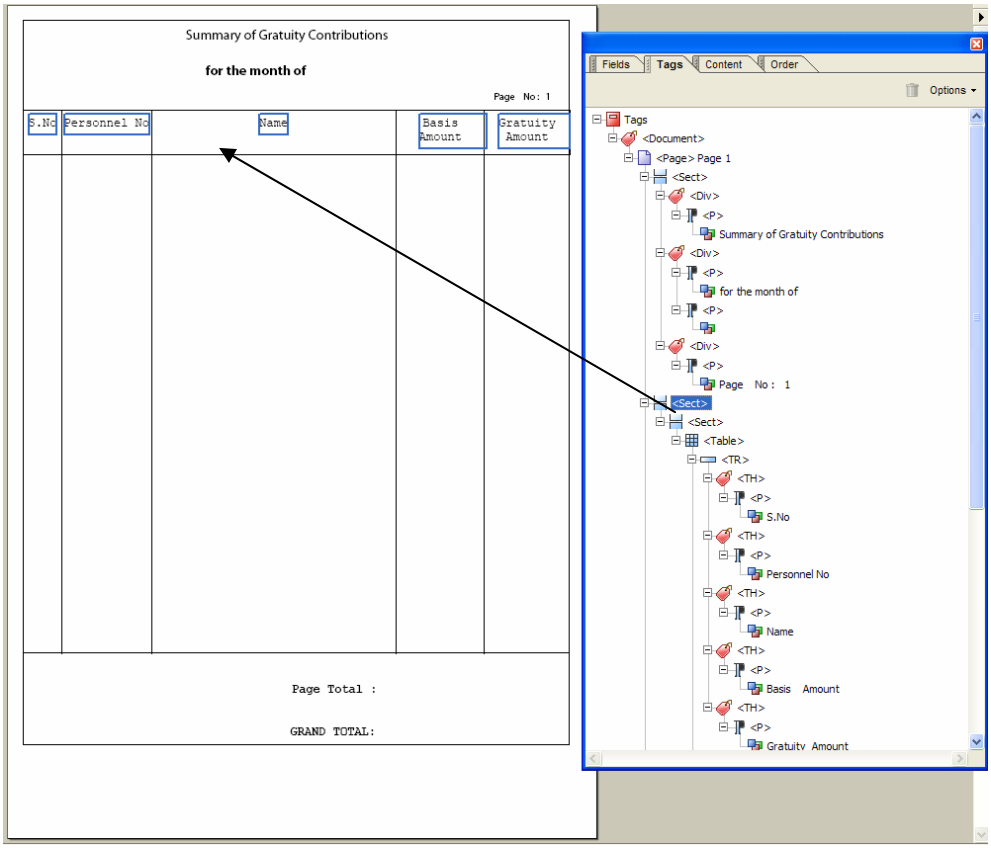
Many government agencies around the world now require that electronic forms be accessible to people with limited vision capabilities. Adobe PDF can be made accessible when they are rendered by specifying that a tagging structure be included with the rendered PDF.

These tagging structures, whose elements resemble those of HTML tags, are read aloud the third party screen reader software.



The accessibility tags can be viewed for a particular PDF using Adobe Acrobat Professional. Select “View/Navigation Tabs/Tags” in Version 7.0 to see the tags. Select “Options/Highlight Content” in the “Tags” window in order to view the corresponding fields for a particular tag.

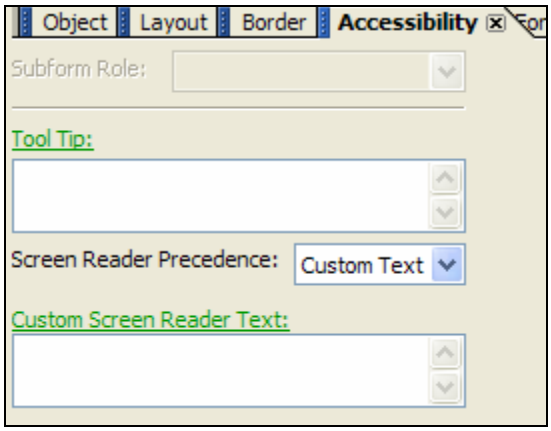
Accessibility Tags visible in Adobe Acrobat Professional



Designing Accessible Forms

In order to ensure that forms are accessible, each element should include text that will be read by the screen reader. This is done in the Accessibility property sheet.

NOTE: For text fields, it is not recommended that the tooltip or custom screen reader text be used for visible elements such as text fields. The visible caption of the text field will be read by the screen reader.



In order to ensure that a particular form is accessible, it is recommended that:

Recommendation	Explanation
Use Tables as much as possible	Tables will help structure the content and also provide the user with feedback with respect to current location within a table by reading the column heading and row location using the accessibility tags.
Avoid positional subforms	Accessibility tags are generated left to right and top to bottom for positional subforms. A form designed using positional subforms may not generate the expected accessibility tags. Use flow content subforms to ensure that consistency.

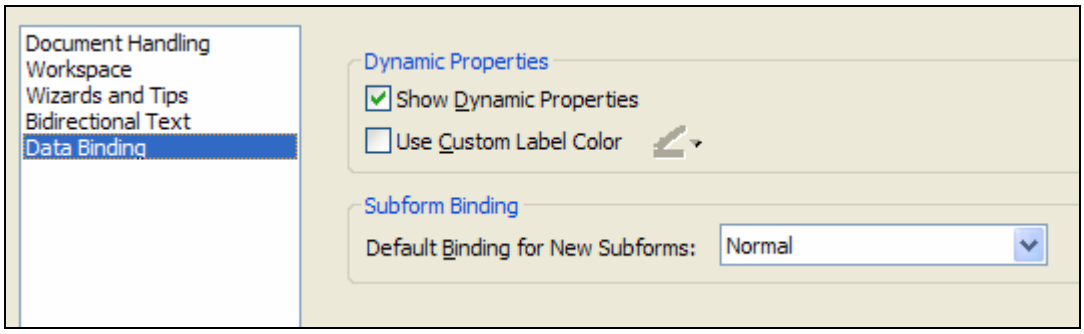
Checkboxes and Radio Buttons

For check boxes and radio buttons, it is recommended that the caption be set and that no custom text is required.

Best Practice #8: Use Dynamic Properties to Bind Data to a List

When a list of objects in an XML data stream needs to be placed into a list (for example, a drop down list), then the Dynamic Properties feature is the best approach.

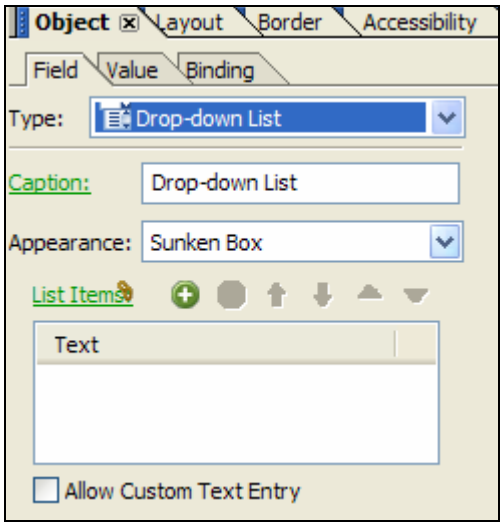
To enable Dynamic Properties, select Tools/Options/Data Binding in Adobe LiveCycle Designer:



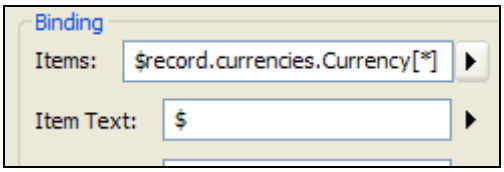
To bind the following XML data to a drop down list:

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

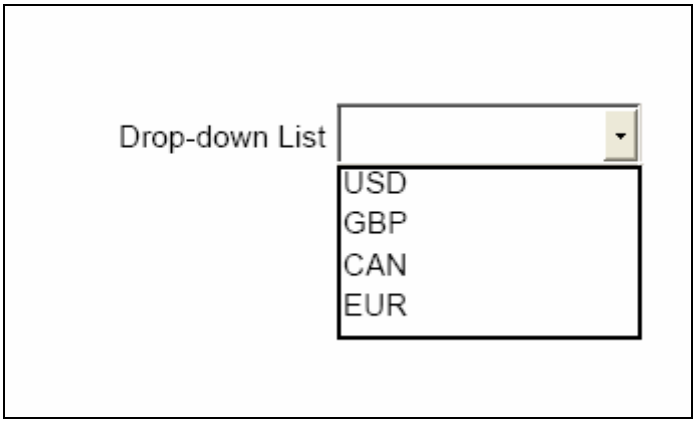
Simply click on the drop down list in the template and select “List Items” from the Object property sheet:



Select the following binding in the Dynamic Properties Dialog box:



The resulting interactive form will now contain the list of items in the drop down list:



Dynamic properties are much faster at populating drop down lists than using a script-based solution.

When designing forms that will be using drop down lists, it will be more efficient to have your XML data structured so that there is a list of child nodes below a parent node. For example:

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

That way, it is easy to map the XML data to a drop down list.

Best Practice #9: Useful Online References

Here is a list of useful references that can be helpful when designing forms using Adobe LiveCycle technology:

Adobe "eRoom"

For employees of Adobe and SAP, the Adobe "eRoom" is a web site where various documents and specifications related to the Adobe LiveCycle Forms Technology can be downloaded.

The "eRoom" can be accessed from:
https://eroom.adobe.com/eRoom

In the eRoom, you can download the XFA specification from the following room:
https://eroom.adobe.com/eRoom/fid302/SAPAdobe/0_6449

Adobe Web Site

The XFA specification can also be downloaded from the Adobe Web site:

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

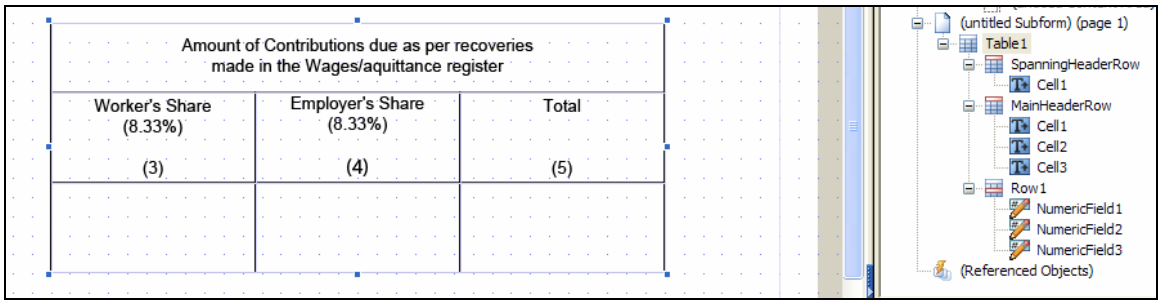
Adobe LiveCycle Online Help

The Adobe LiveCycle Designer has an online help file which contains detailed reference information as well as recommendations to follow when designing online forms.

Best Practice #10 – Use the Hierarchy View to Select Form Content

Although Adobe LiveCycle Designer does an excellent job of showing a form designer what a form will look like, some objects on a form are not readily seen without inspecting the hierarchy. As well, when using tables or nested subforms, adjacent objects have very little discernible space between them. This makes it more difficult to select a specific object. Instead, use the hierarchy to make your selections more absolute. As a result, operations such as resizing or copying are much easier.

For example, when editing the following table:



If you wish to add another header row after the "SpanningHeaderRow", it will be easier to select the row in the hierarchy and then right click to insert the row rather than select the object in the template view.

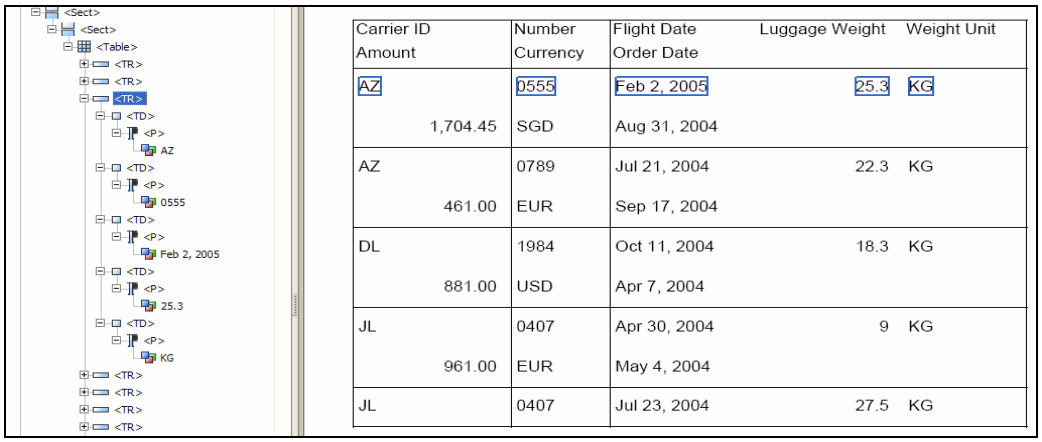
Best Practice #11 - Designing an Accessible Multi-line Table

The objective is to design a multi-line table such as the following:

NOTE: See the attached files (ml_output.pdf, multiline_table.xdp, multiline_table.xml) for the complete solution to this exercise:

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

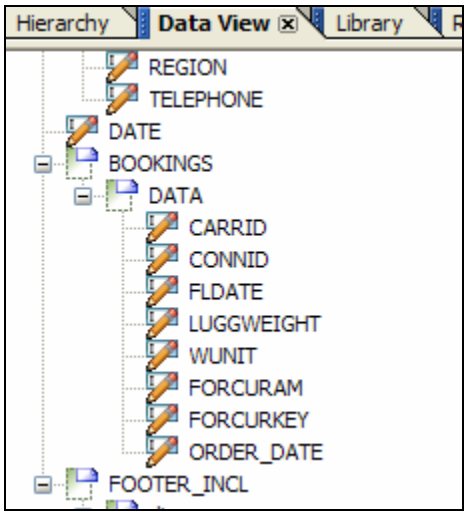
If this form is constructed using the Table Designer, then it will not be accessible. This is because if multiple header and body rows are used in the table, then it will generate multiple lines of accessibility tags (see below).
NOTE: This is true even if the individual body rows are grouped as part of a table section.



Carrier ID Amount	Number Currency	Flight Date Order Date	Luggage Weight	Weight Unit
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

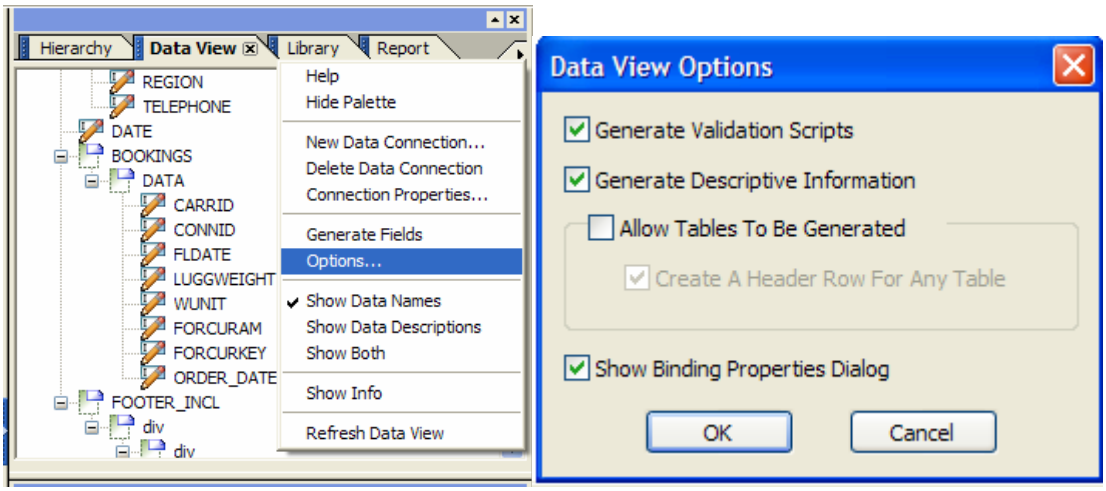
Instead of using the table designer, this solution can be built with a set of nested subforms and then set the accessibility role of the subforms to emulate a table. Here are the required steps:

This solution assumes that you have a data schema already set up for the template. For SAP form designers, the data schema is automatically generated from the context when the form is opened in Designer. You should have a set of fields in your data view that you wish to place in the table. See the example below:



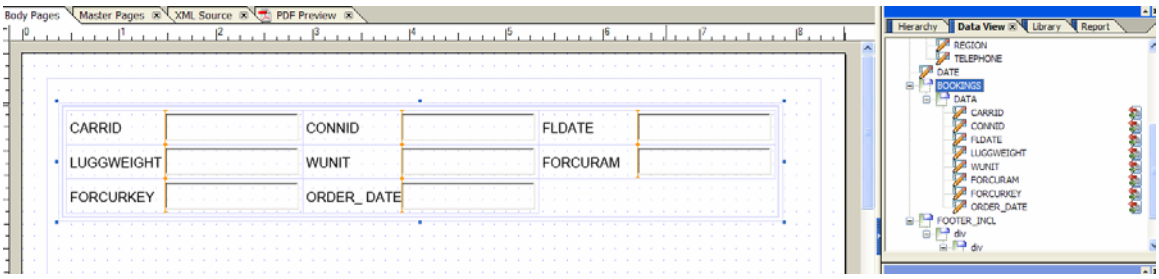
Step 1: Turn off table generation option

We will be dragging and dropping data fields onto the form. We do not wish to create a table in this case. Tables do not support wrapped header and body rows, which is what we need in order to make this solution accessible. Click on the Data View palette menu in the upper right hand corner and then make sure that the "Allow Tables To Be Generated" option is turned off (see screenshots below);



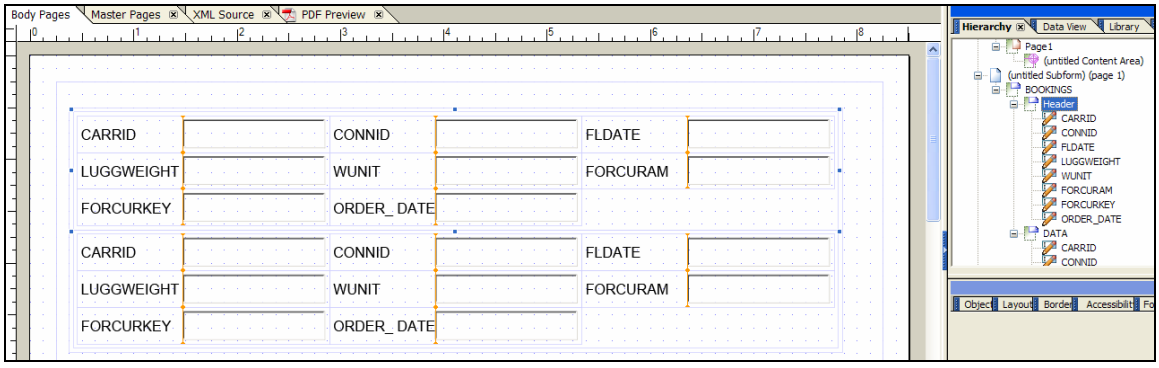
Step 2: Create the data bindings

Drag the BOOKINGS data node from the data view to the far left side of the form (see screenshot below):



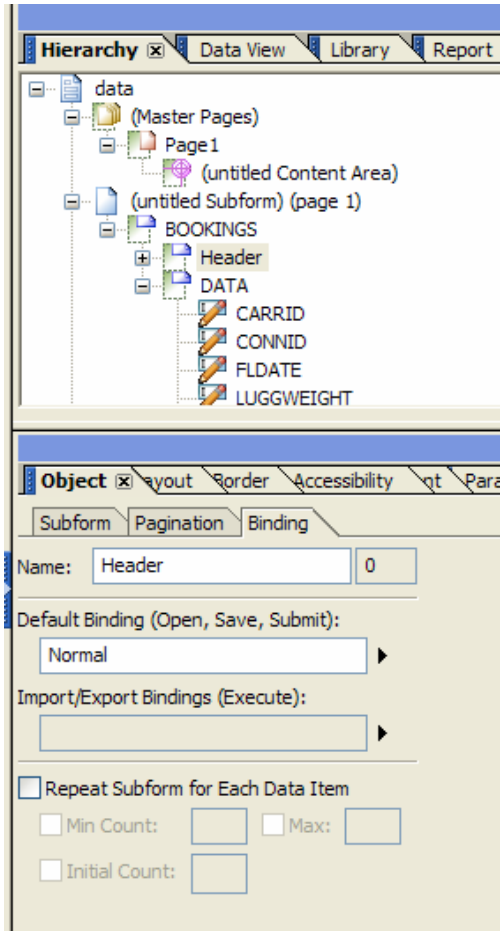
Step 3: Create the Header Subform

In the hierarchy view, select the "DATA" subform and select "Edit/Duplicate"(keyboard shortcut CTRL-D) to create the header subform. We are duplicating this subform so that it will be easier to create the headings from the DATA subform text field captions. You will now have two subforms named DATA (DATA[0] and DATA[1]). Rename the first DATA subform as "Header". Your template should now look as follows:



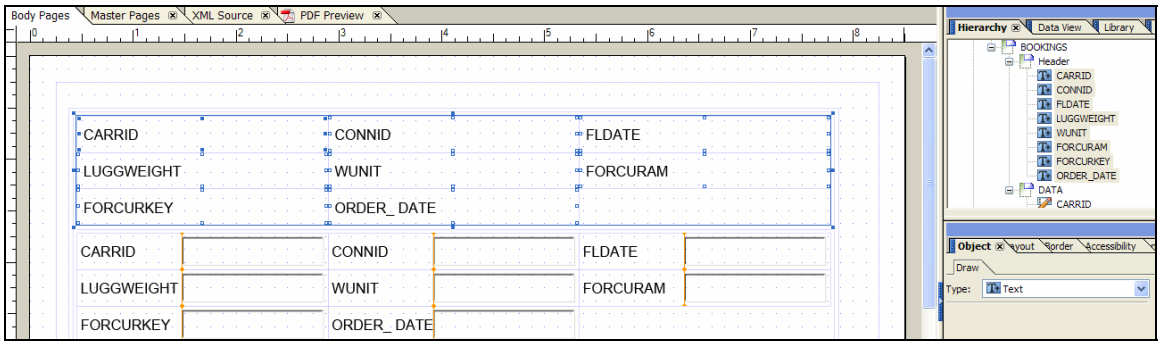
Important Note: When the DATA subform was copied, the Header subform inherited two properties that are not required.

The "Repeat Subform for each data item" option in the Binding tab of the Object palette should be cleared. Also, the binding of the Header subform should be "Normal". See the screenshot below for details:



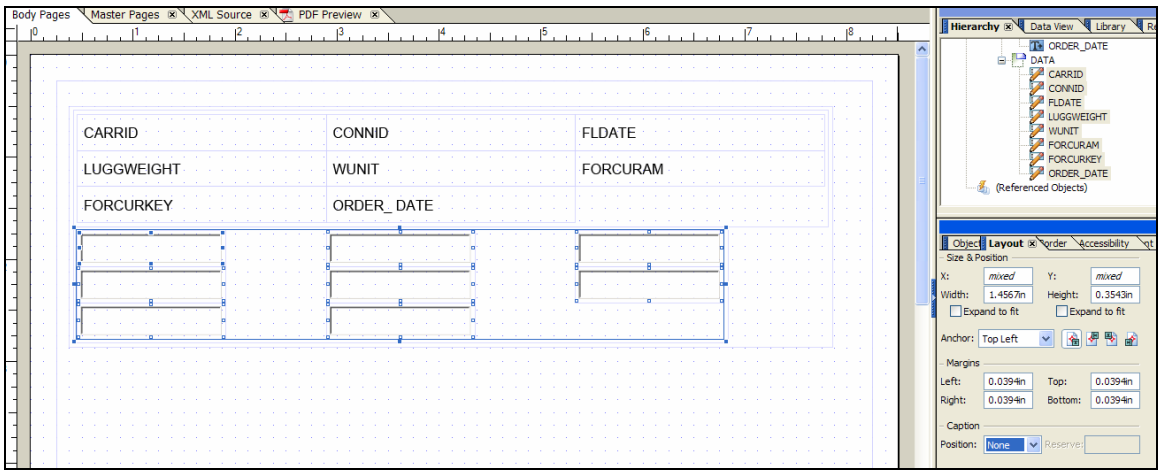
Step 4: Create Captions for Header Subform

Select all of the text fields in the "Header" subform. To do this in the hierarchy view, click on the first field, hold down the <SHIFT> key and select the last field in the list. Select the "Type:" drop down list in the Object Palette and change the text fields to field type "Text", This converts all of the text fields to static texts and copies the text field captions at the same time. See the screenshot below for details.



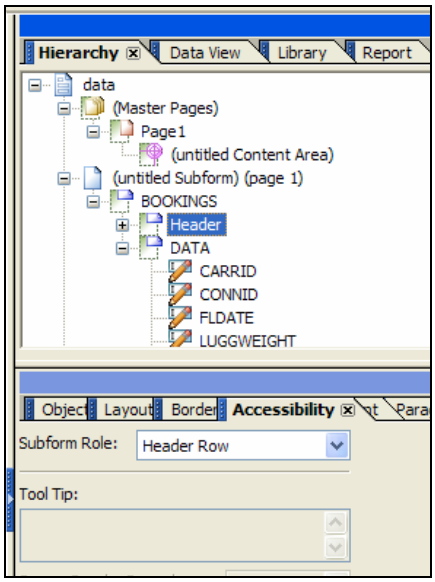
Step 5: Remove the captions from the DATA subform

Select all of the text fields in the "DATA" subform. To do this in the hierarchy view, click on the first field, hold down the <SHIFT> key and select the last field in the list. Select the "none" option for the Caption/Position drop down list in the Layout palette. See the screenshot below for details.



Step 6: Set the Accessibility Role for the Subforms

In order for the subforms to be recognized as a table by the Screen Reader software, we must ensure that <Table>, <TR>, <TH> and <TD> tags are generated as part of the PDF file. To do this, we must set the Subform Role field in the Accessibility Palette for the "BOOKINGS", "Header" and "DATA" subforms.

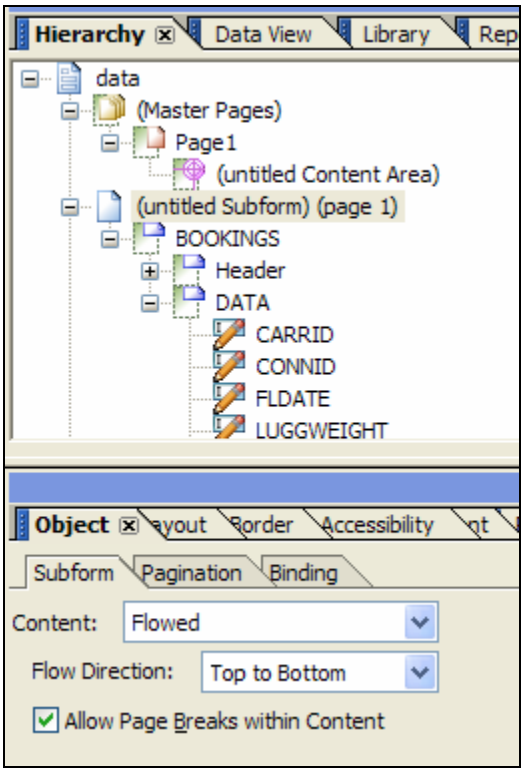


The role values should be set as follows:

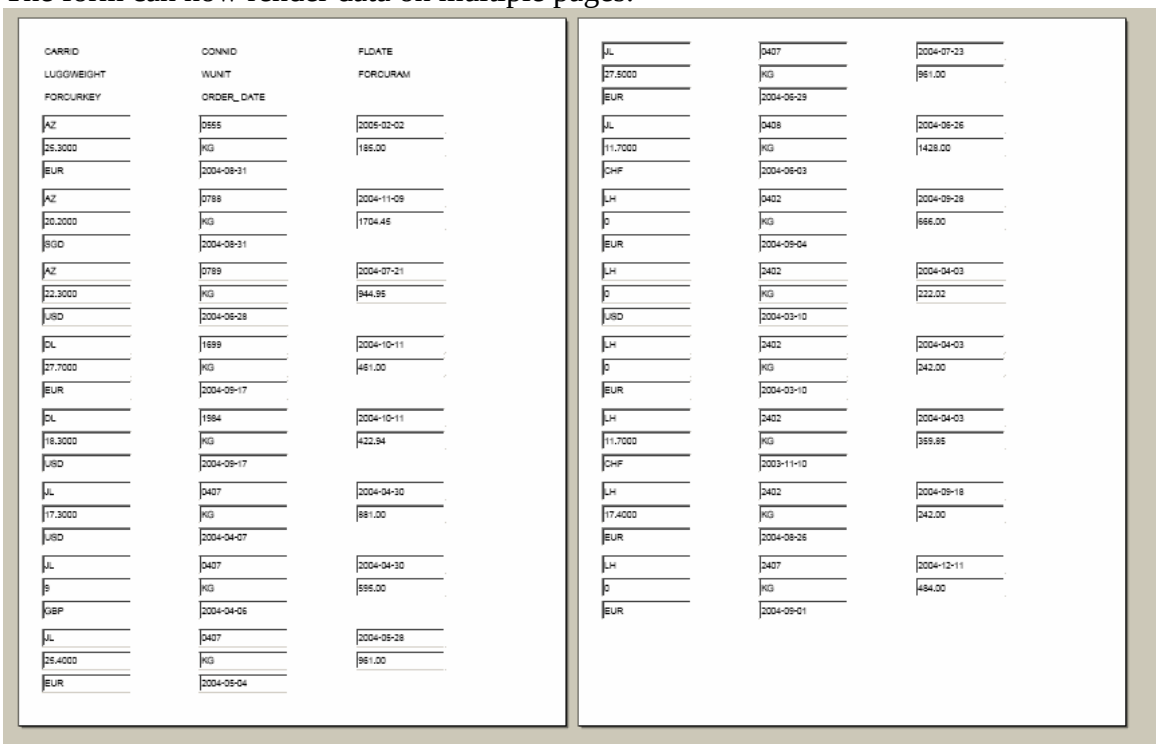
BOOKINGS	"Table"
Header	"Header Row"
DATA	"Body Row"

Step 7: Allow the table to span multiple pages

The "Content" property of the parent subform on the body page is set to "Positioned". This means that any data in the table will be confined to the first page when rendered with XML data. To change this, the parent subform should have the "Content" property set to "Flowed" and the "Flow Direction" property set to "Top to Bottom".



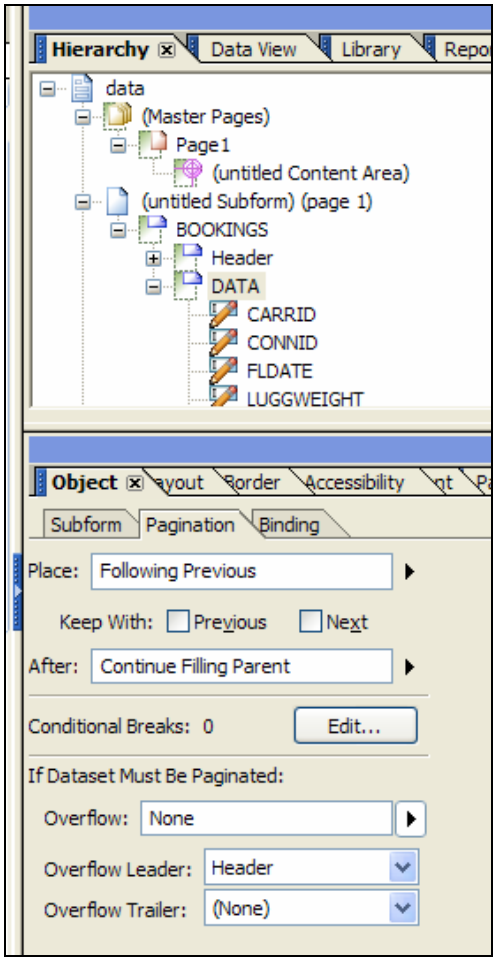
The form can now render data on multiple pages.



However, the Header will only appear on the first page. This can be corrected with the next step:

Step 8: Set the Header to appear on each page

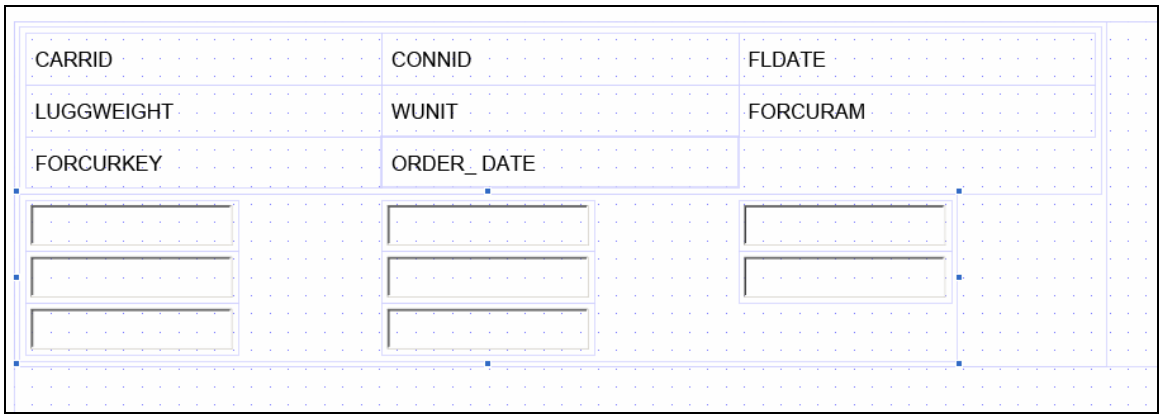
In order to ensure that the Header subform appears on the top of each page, set the "Overflow Leader" drop down list to "Header" for the "DATA" subform. This drop down list is available in the "Pagination" tab of the "Object" palette. See the screenshot below for details.



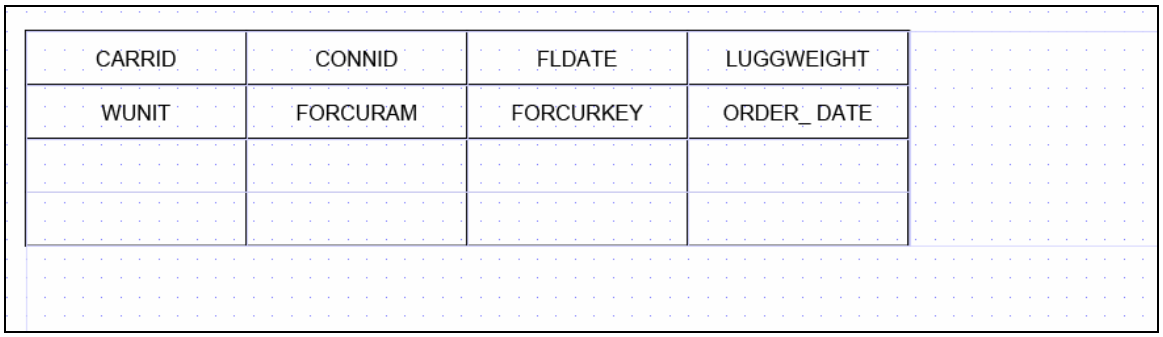
Important Note:
The "Content:" attribute of the subform must be set to "Positional" if it is to be used as an "Overflow Leader" subform.

Step 9: Arrange the position and size of the fields

In this step, the subforms will be configured so that the rendered output will more closely resemble a table. The layout of the form will be transformed from this:



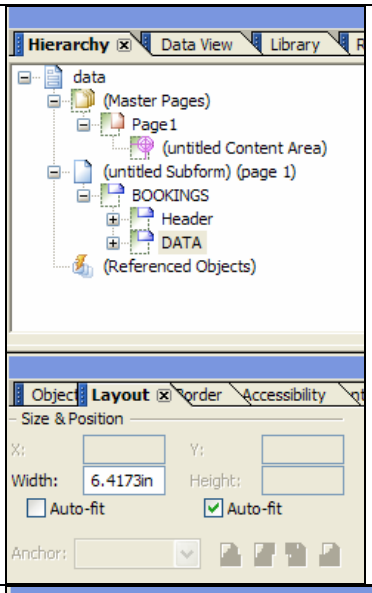
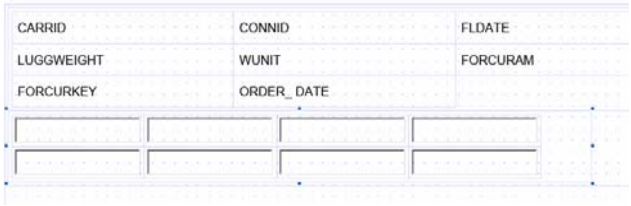
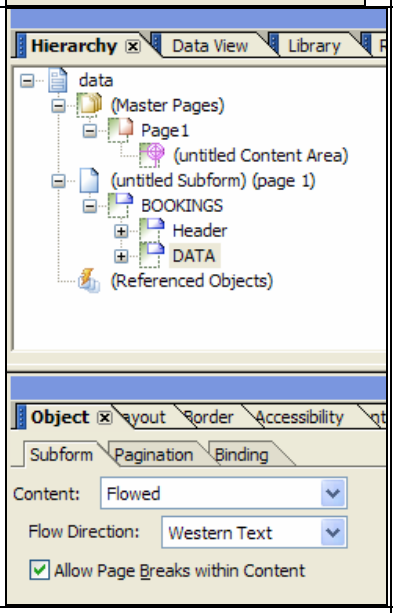
To this:

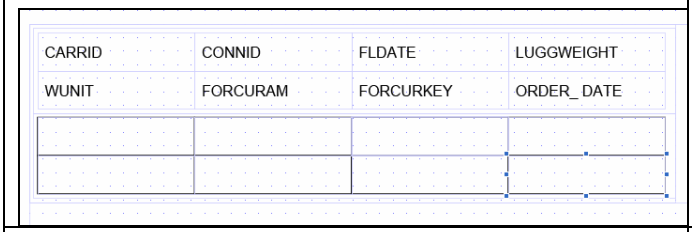


Align the DATA subform fields

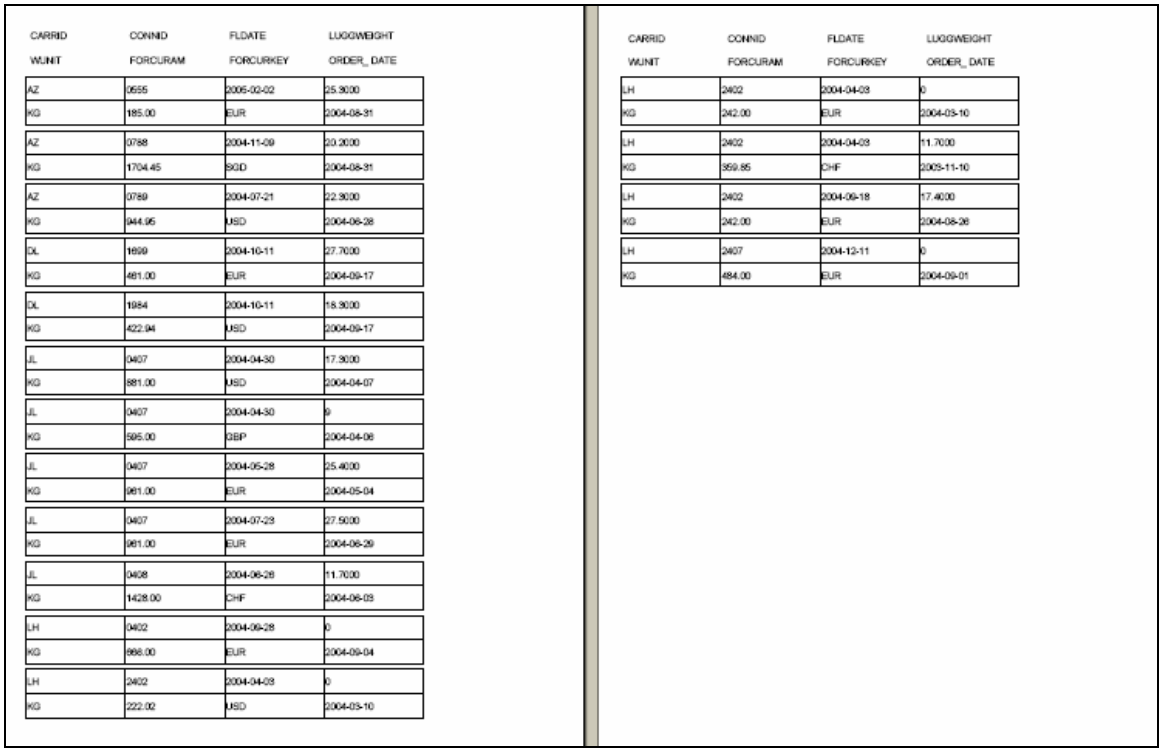
In order to arrange the fields as a body row in a table, we will set the flow content of the subform to “Western Text”, which means the fields will flow from left to right, top to bottom within the subform. Since changing the flow content of the subform will initially set the direction to “Top to Bottom”, we will first ensure that the width of the subform will be fixed and will not automatically adjust. If we fail to do this, when we set the flow content of the subform, it will not change the width of the subform to a size that exceeds the content area.

The steps are:

Turn off the “Auto-Fit” checkbox in the Layout palette of the DATA subform.	
Set the “Content:” drop down list in the Object palette of the DATA subform to “Flowed”. This will resize the subform so that all fields are vertically arranged. Do not be concerned. Set the “Flow Direction” to “Western Text”. The subform now arranges the fields in two rows of 4 fields each. 	
In the hierarchy view, select all of the fields in the DATA subform and set the “Appearance” drop down list in the Field tab of the Object palette to “Solid”. While still selecting all fields, set the margins to “0” in the Layout palette.	
Repeat the same process for the “Header” subform. Turn off the “Auto-Fit” checkbox in the Layout palette of the “Header” subform. Then set the “Content:” drop down list in the Object palette of the “Header” subform to “Flowed”. This will resize the subform so that all fields are vertically arranged. Do not be concerned. Set the “Flow Direction” to “Western Text”. The subform now arranges the fields in two rows of 4 fields each.	

The purpose for this is to arrange the fields next to each other without manually resizing.	
We will now set the size of the “Header” subform fields to match those in the DATA subform. In the hierarchy, first select the first field in the DATA subform and then while holding down the <CTRL> key select the first field in the “Header” subform. Then select the menu item Layout/Make Same Size/Width. This will resize the “Header” field to match the DATA subform field.	
Resize the “Header” and DATA subforms so that they are the same width as the 4 fields in the first row of the DATA subform.	
Now set the all of the “Header” fields so that they are the same size as the first field. Your screen should now look like this: 	
Set the “Content:” drop down list in the Object palette of the “Header” subform to “Positional”. This is necessary if the Header subform is to be used as an Overflow leader to show up on each page.	
NOTE: If you wish to have different column widths, you will have to do the following: 1. Set the “Content:” property of the DATA subform to “Positional” 2. Adjust the width of a Header field and then use the Layout/Make Same Size/Width menu command to set the corresponding DATA field. 3. It is not necessary to restore the “Content:” property of the DATA subform to “Flowed”. This helps to arrange the fields, but it is not mandatory.	
Set the margins of the DATA subform in the Layout palette to “0”.	
The form is now complete. If you wish, you can adjust the alignment of columns to suit the type of data being presented.	

The final solution is as follows:



CARRID	CONNID	FLDATE	LUGGWEIGHT
WUNIT	FORCURAM	FORCURKEY	ORDER_ DATE
AZ	0555	2005-02-02	25.9000
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	22.8000
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	18.9000
KG	422.94	USD	2004-09-17
JL	0407	2004-04-30	17.9000
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	2004-06-28	11.7000
KG	1428.00	CHF	2004-06-03
LH	0402	2004-09-28	0
KG	868.00	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

Best Practice #12 – Designing an Accessible Multi-line Heterogeneous Header Table

An example of a multi-line heterogeneous header table would be as follows:

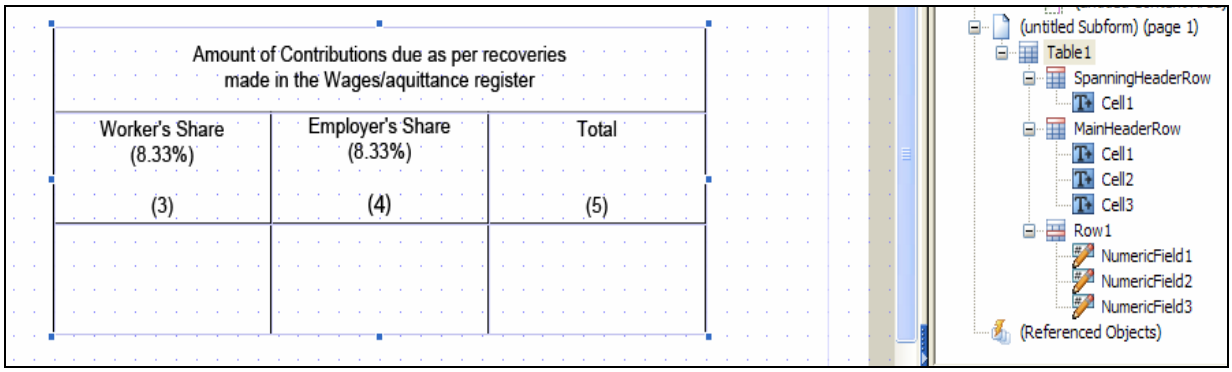
Amount of Contributions due as per recoveries made in the Wages/aquittance register		
Worker's Share (8.33%) (3)	Employer's Share (8.33%) (4)	Total (5)
	1.666,00	1.666,00

In order to design this table, you can use the Table Designer and then use column spanning on the first header row.

The steps to create this kind of table are as follows:

1. Create a table using the “Insert Table…” menu command in Adobe Designer using 3 columns, 1 header row and 1 body row
2. Insert a second header row by right clicking on the first header row in the hierarchy and then selecting the “Insert/Row Below” menu command.
3. Select all three cells in the first header row and then right click and select the “Merge Cells” menu command.

The design of this template would look like the following:



The screen reader software (JAWS) is expected to 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	H2 3
Data 1	Data 2	Data 3

If the user was in the following location, JAWS would read out the position 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”

Another more complex example:

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

If the user was in the following location, JAWS would read out the position 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”

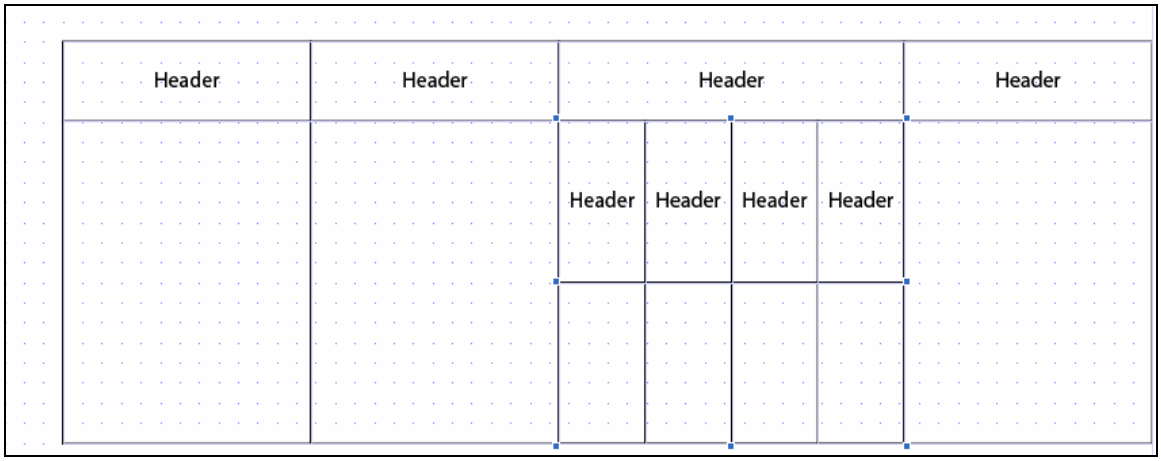
Important Note:
As of the writing of this document (September 24, 2005), the only supported software solution that is accessible is:

JAWS 7
Adobe Reader 7.07

It is recommended that forms requiring multi-line heterogeneous header tables use this kind of design. In order for this solution to comply with accessibility requirements, fixes for the JAWS software may be required.

Best Practice #13 – Designing Accessible Nested Tables

It is possible to design nested tables using the Adobe Table feature. See the screenshot below for an example of a nested table. This is created by selecting a cell on the outer table and then inserting a second table using the "Insert Table…" command.



As of JAWS 7.00 release, nested tables are read as layout tables by JAWS. Layout tables are used to arrange objects on the screen, but no hierarchical relationship is implied. Although the Adobe Reader will provide a hierarchy of table structure to JAWS, it will not be possible to have JAWS read out the inner table to the user.

Best Practice #14 – Designing Accessible Horizontal Tables

In order for data to flow horizontally across each page, the Adobe Table Designer cannot be used. Instead, a set of nested subforms must be used. An example is as follows:

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

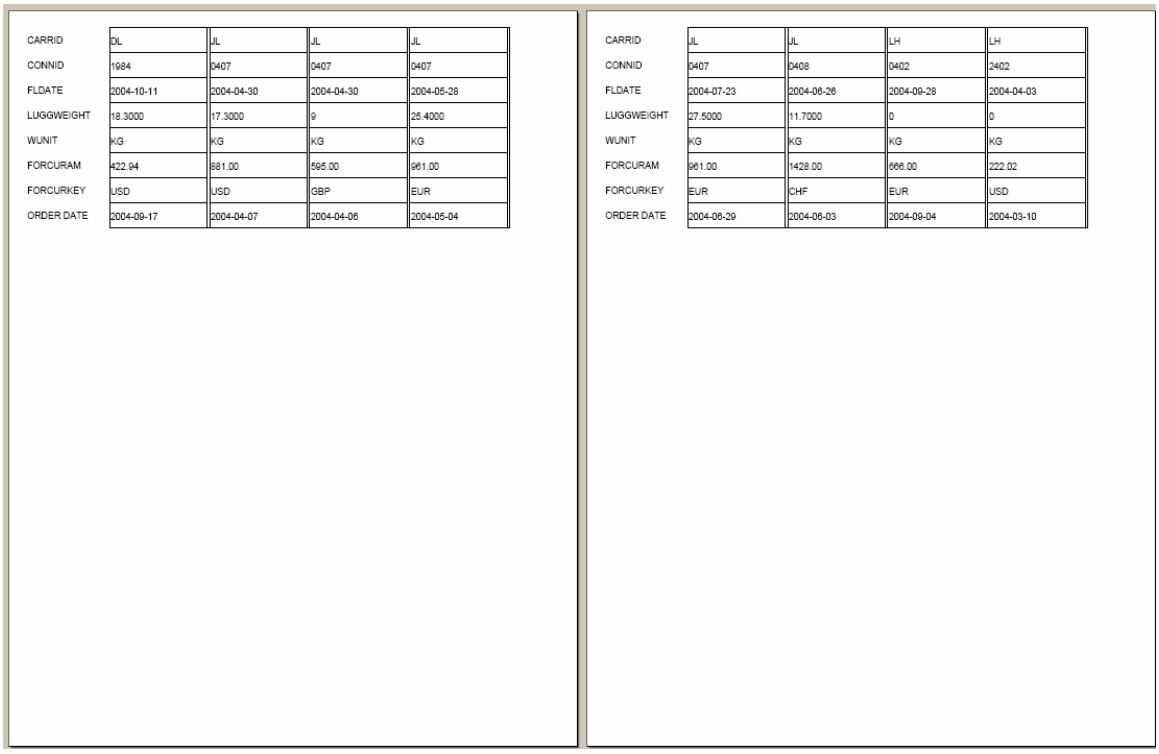
For details, please see the following instructions: Best Practice #11 - Designing an Accessible Multi-line Table.

For this particular form, you must make the following changes:

The header subform must have a “Content” property set to “Positional”
The DATA subform must be Positional as well
Turn off “Allow Page Breaks within Content” for the DATA subform

NOTE: At the time of writing of this document, JAWS 7 is not reading the accessibility tags as intended for this solution. JAWS 7 uses the physical location of the cells on the page to help it analyze table structure. Therefore, **this is not an officially recommended solution at this time** for creating accessible horizontal tables. This document will be updated when a solution for this problem is found.

The form will render 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

Best Practice #15 – Designing Accessible Tables with “Out of place” Rows

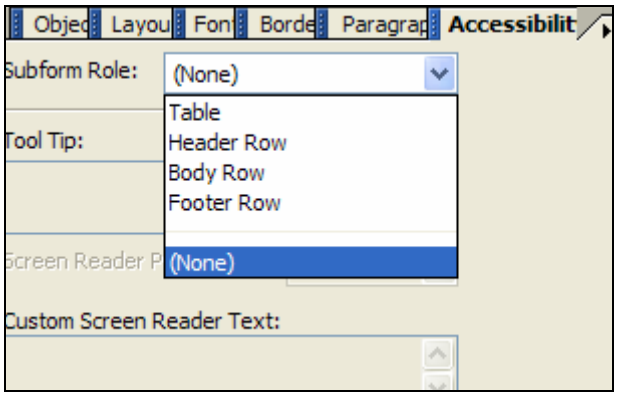
An example of a table with an “out of place” row is shown below:

Beleg-Nummer	Beleg-Datum	Vorgang	GsBer	Währung	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

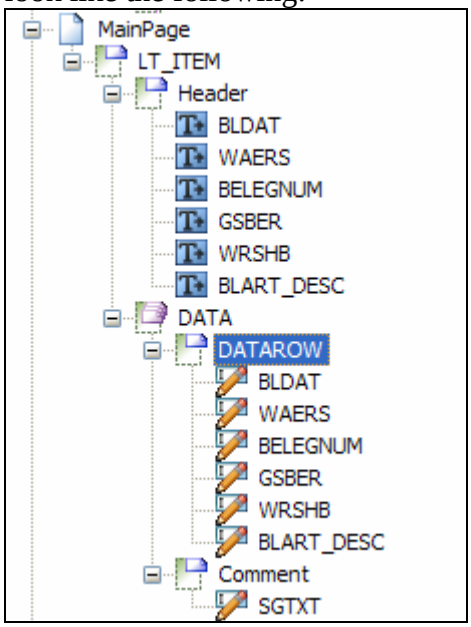
In this example, there are 6 columns in the table. The seventh data field is a comment text that appears in some of the body rows of the table.

One possible solution is to create this form as a wrapped, multi-line table. However, this solution means that the table will be read out as a 7 column table including the headers. For an example of how to create a set of nested subforms see the example in Best Practice #11 - Designing an Accessible Multi-line Table.

A better solution to the problem would be to design it the form using a set of nested subforms and then add an extra subform for the comment row. Then set the accessibility role of the extra subform to “None”. This is done in the Accessibility palette for the subform as shown in the following screen shot:



The hierarchy view would look like the following:



See the attached template, data and PDF file (BP15.*) for details.

Best Practice #16 – Designing Accessible Tables with a large number of data fields

If you need to place a large number (e.g. more than 10) of data fields in a table, then you should follow Best Practice #11 - Designing an Accessible Multi-line Table. Since a large number of data fields are involved in this case, it means that the header and body subforms will contain a significant number of rows.

You should avoid designing such forms if the number of fields is large. Below is a form containing 57 data fields that illustrates that such a design may not create a very usable form.