



Form Design

FLM version 261

Author: Emily Burfoot
File name: FLM form design
Version: 1
Distribution:

Version History

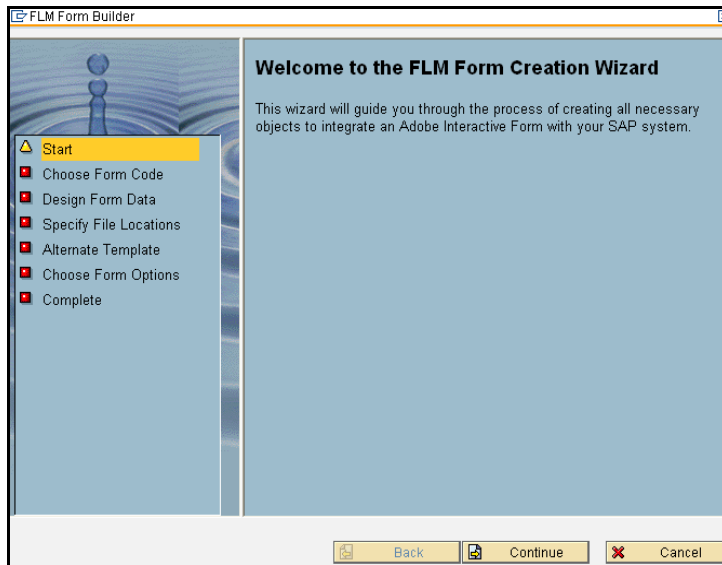
Version	Author(s)	Reason for update
1	Emily Burfoot	Initial release of the document

Table of Contents

1	FLM Form Wizard	3
1.1	Specify Form Type Details	3
1.1.1	Form type code.....	4
1.1.2	Version.....	4
1.1.3	Language	4
1.1.4	Form Description.....	4
1.2	Form Data Structure	5
1.2.1	Subforms	5
1.2.2	Fields.....	6
1.2.3	Field Types	6
1.2.4	Read and Post Routines	6
1.2.5	Read Routines	6
1.2.6	Post Routines	7
1.2.7	Field Editing	7
1.3	Specify Location for XML Data Definition File	7
1.4	Alternate Adobe Designer Templates	8
1.5	Form Options.....	8
1.6	Summary.....	9
2	Adobe Designer	10
2.1	Accessing the FLM form	10
2.2	Data hierarchy	12
2.3	Subform definition and binding	14
2.3.1	Binding for non-repeating subforms.....	14
2.3.2	Binding for repeating subforms.	14
2.3.3	Binding for nested subforms.....	16
2.4	Subform look and feel hints and tips.....	16
3	To import an existing Adobe form into FLM	16

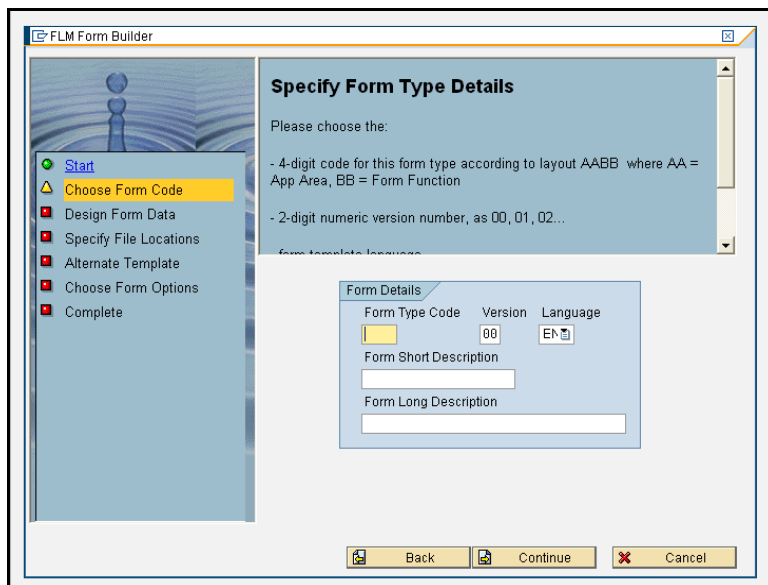
1 FLM Form Wizard

Here you create the logical definition of a new interactive form, including field names, field types and form templates. This section will take you through an example form creation to explain all the processes involved.



On the welcome page, click 'Continue' to begin creating your form.

1.1 Specify Form Type Details



This section of the form wizard defines how the form will be identified within SAP. Each form is defined by a four-character code, version number and language. You can enter a short description and long description of the form here.

1.1.1 Form type code

Invent a form type code here: the first two letters define the application area while the second two denote the form's function. For example, a sales order form may have the code SAOR. It is a good idea to decide and agree on a universal form naming system so that each form code accurately describes the form's application area and function. This will be helpful, for example when managing form routings.

1.1.2 Version

One of the features of FLM is the support of concurrent form versions. So it is possible to create a form with the same function and 4-character code, but with a different version number. This facilitates forms management, because the 4-digit character code can be fixed for a specified form type and application, and does not need to be altered with successive versions of the form.

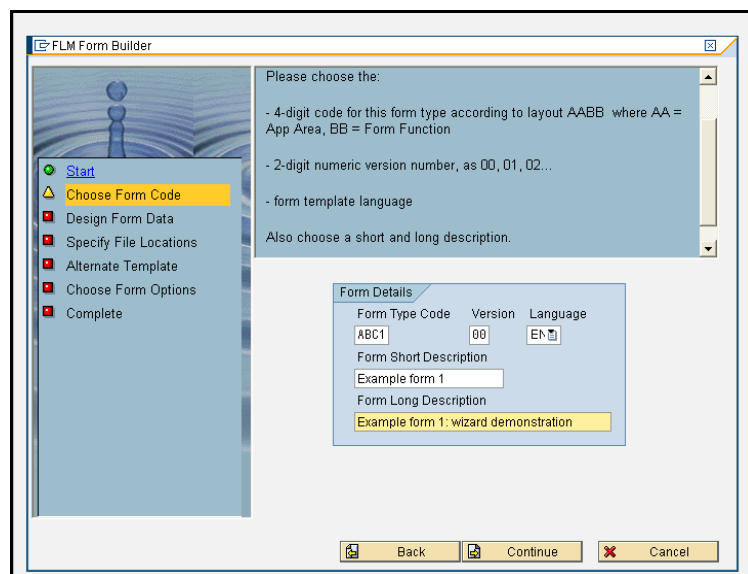
Use this field to enter the version number of the form.

1.1.3 Language

Each form must be generated in English before a copy can be made in another language. Once a form has been designed in English though, any language can be selected for a successive version of the form.

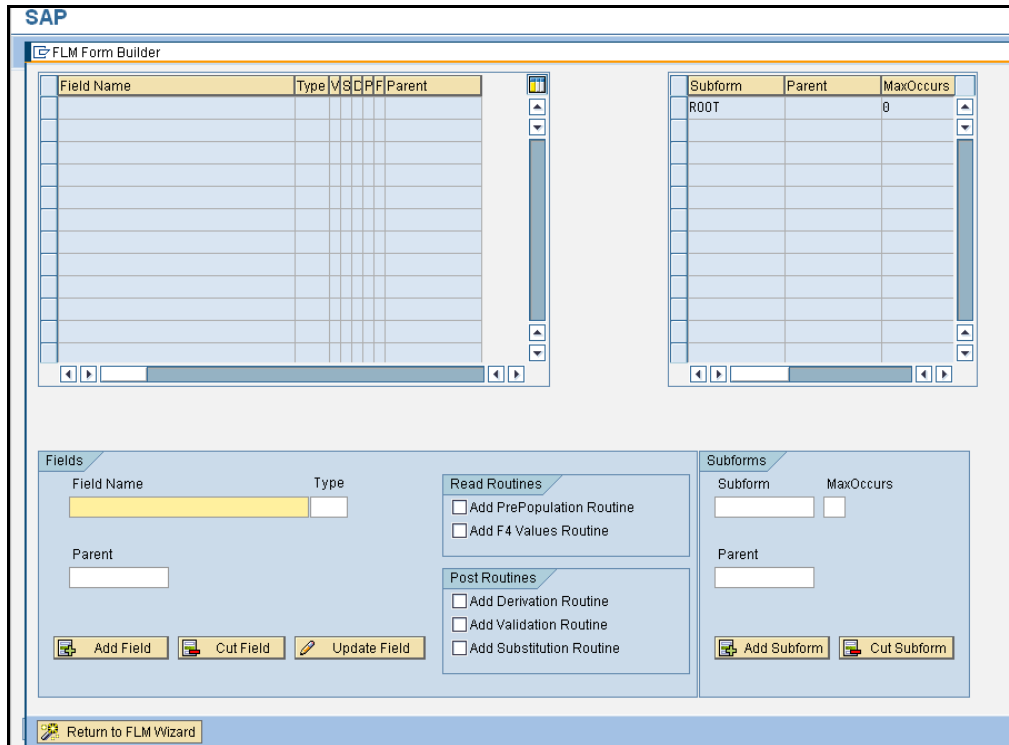
1.1.4 Form Description

Enter a short and long description of the form to easily identify it in the forms list.



1.2 Form Data Structure

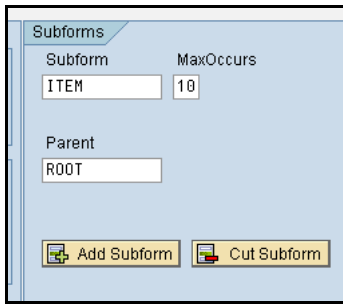
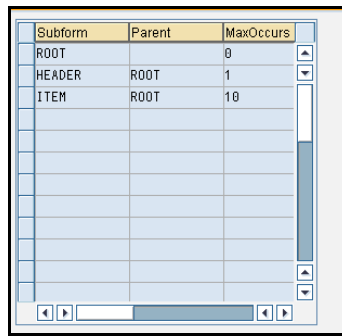
Launch form data designer at this stage of the wizard to input the required fields and subforms.



1.2.1 Subforms

Begin the new form creation by creating a Subform. This is a logical or physical grouping of fields that defines their characteristics on the form. It is a feature of Adobe interactive forms that some fields may be copied according to how many times it is required. For example, on a sales order form, an item description/quantity field may be copied according to the number of discrete items to be ordered.

However it would be undesirable for some fields, for example 'name' and 'address' to be allowed to appear more than once on the form. MaxOccurs allows you to set the maximum number of times a subform field may occur on the form. Following the same example, it would be advisable to set Header fields as MaxOccurs: 1, and Item fields as MaxOccurs: 10. When these fields are eventually created, they can then be assigned to the relevant subform group to maximize the functionality of the fields according to requirements.

A screenshot of a 'Subforms' configuration dialog box. It has a light blue background. At the top, there's a title bar 'Subforms'. Below it, there are two input fields: 'Subform' with the text 'ITEM' and 'MaxOccurs' with the value '10'. Below these is a 'Parent' field with the text 'ROOT'. At the bottom, there are two buttons: 'Add Subform' with a plus icon and 'Cut Subform' with a minus icon.A screenshot of a table showing subform hierarchy. The table has three columns: 'Subform', 'Parent', and 'MaxOccurs'. The data is as follows:

Subform	Parent	MaxOccurs
ROOT		0
HEADER	ROOT	1
ITEM	ROOT	10

1.2.2 Fields

To create a field, begin by entering the field name in the 'Field' section of the page. The Field Type can be assigned as character, numeric, date etc. The field can then be assigned to one of the subforms you have created under 'Parent'. Please note that the field name must consist of alphanumeric characters only; underscores are allowed but spaces are not. Though the field names should resemble their content, they need not be the label of the field as it will appear on the form. This is done via form design at a later stage.

1.2.3 Field Types

CHAR - Character field; will allow free text to be input to the field

NUMC - Numeric field; only numeric characters are allowed (drop down numbers? Validation according to numeric characters?)

DATE - A date field will automatically bring up a calendar date selector in the field on the form

TIME - Only allows times to be input (can be prepopulated with time submitted etc?)

BOOL - Checkbox; will produce a labelled checkbox which can either be ticked or unticked.

1.2.4 Read and Post Routines

Read and Post routines can be assigned to each field:

1.2.5 Read Routines

Read Routines will apply to the field in any reading instance of the form.

A Prepopulation Routine will cause the field to be prepopulated and will not be amendable by the form user. This could be used, for example, for the Form ID field.

An F4 Values Routine limits the field input to options selected from a drop-down menu. This could be used in the case of there being a limited number of field options to select from, such as Country of Residence.

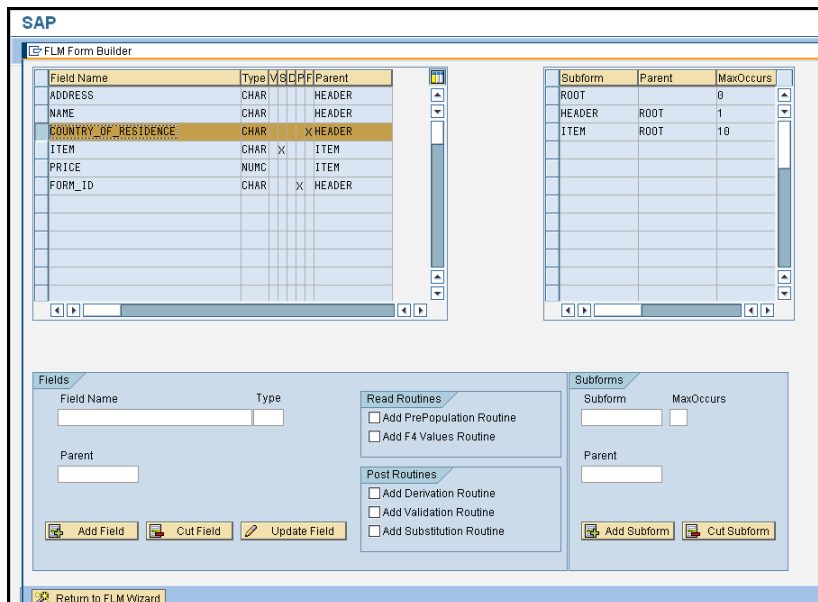
1.2.6 Post Routines

Post routines apply to the transfer of data back into SAP via the Posting Engine.

- A Derivation Routine allows the data on the form to be used create a new field before the data is input to SAP.
- A Validation Routine checks the data in the field for validity before the form can be submitted. For example, a postcode and address may be checked against a data bank of addresses and postcodes to confirm that the submitted data is valid.
- A Substitution Routine substitutes the data input to the field for another value. For example, an item option featured as full text in the form might be substituted for an item code as the data is input to SAP.

Further details of the above processes are available in the Development Guide.

1.2.7 Field Editing



To edit or cut a field after it has been entered, first select the action you wish to perform (e.g. update field), then select its row in the Field Name table by clicking on the blue square to the left of it. Then re-click on the desired action to make changes.

Once all the fields have been entered, click on 'Return to FLM Form Wizard' and proceed to the next step.

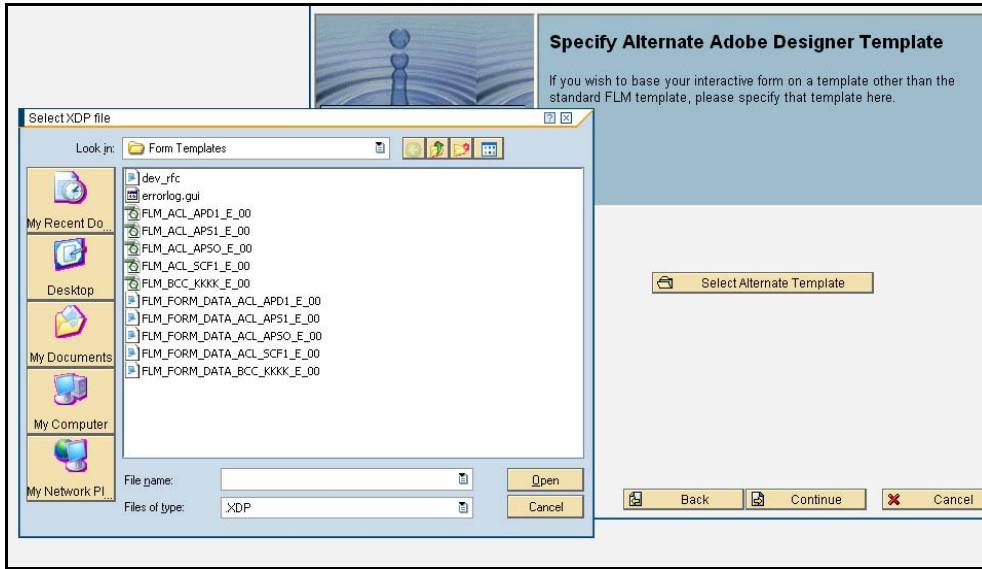
1.3 Specify Location for XML Data Definition File

The form data definitions you have just created will be stored as an XML file during the generation of the FLM interface. Here you can specify the location in which to store that XML file. You must store it to an accessible location such that it can be imported into the Adobe Livecycle Designer Tool. Click 'Set Directory' and select a new location if the one displayed is not suitable.

1.4 Alternate Adobe Designer Templates

FLM uses a standard template on which to base its forms. To use this standard template, simply click 'continue' at the bottom of the page.

To choose an alternative Adobe Designer template (an .xdp or .pdf file only), click 'Select Alternate Template' and import a previously designed form template.



1.5 Form Options

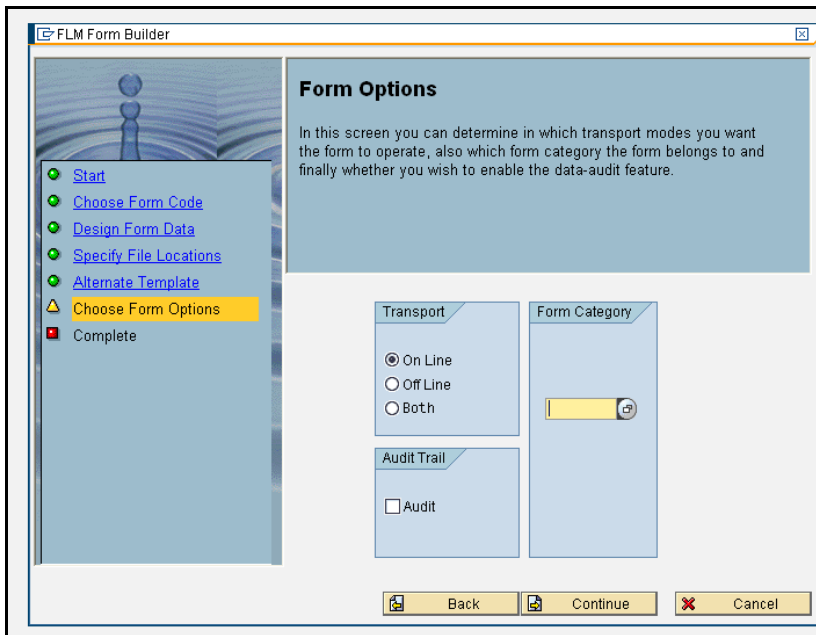
Under Form Options, you can specify the required form category, transport options, and whether an audit trail should be created.

With Offline Transport the form is sent via email as an attachment.

With Online Transport the form is only available via the online form portal.

Selecting 'Both' will allow both online and offline scenarios.

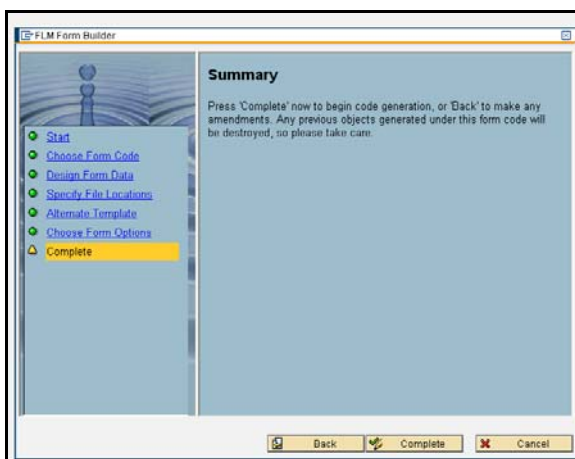
An Audit Trail (Variant Trail) can be created as the form is modified through its routing. Before selecting this option however, it is worth noting that an audit trail generates a significantly larger file size attached to each form, which may not be desirable if the form is to be transported offline.



1.6 Summary

You can now click on 'complete' to generate your form, or 'back' to make any amendments. On clicking 'complete' the components of the form will automatically be saved in the system.

Once the wizard has been closed, you can still make amendments by initiating the form wizard for a form of the same code, and if necessary updating the version number of the form.



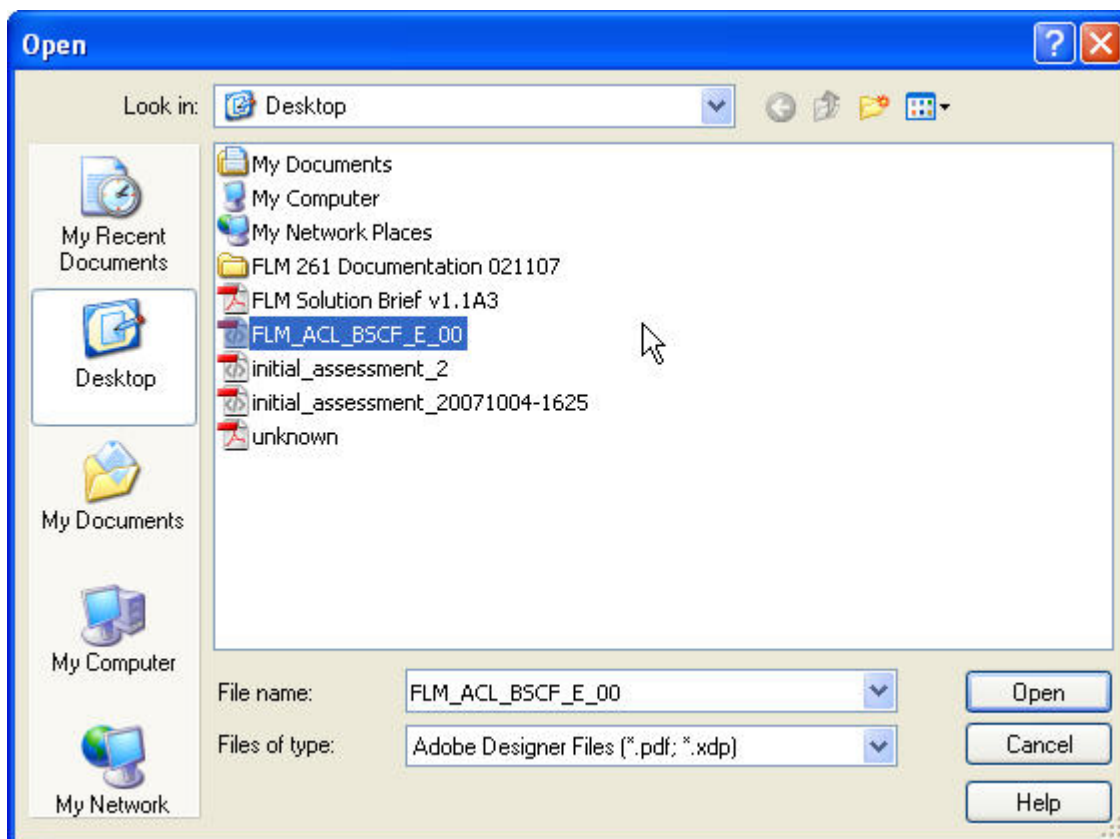
2 Adobe Designer

Once the logical form definition has been created using the form wizard, the aesthetic aspects of form design are done using the WYSIWYG Adobe Designer tool. The minimum that needs to be done in Adobe designer is to bind the fields created using the form wizard to the fields as they appear on the form, but a wide selection of look-and-feel customising options are available. This guide will take you through the basics of Adobe Designer for the purposes of an FLM form; for more information please see the product help.

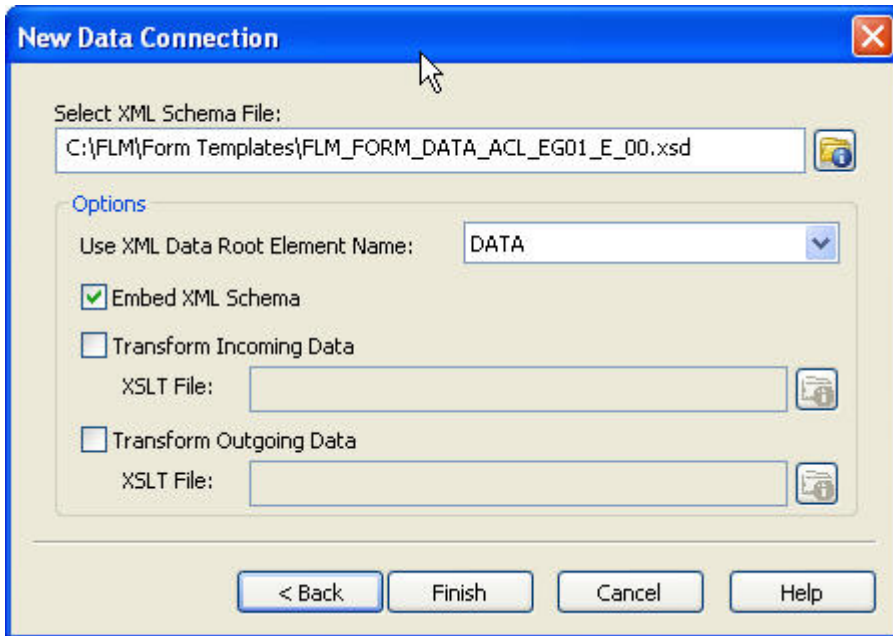
2.1 Accessing and binding the FLM form in Adobe Designer

This step details the minimum actions required to setup a working FLM form. The following steps provide a more detailed explanation of the workings of Adobe Designer to fully configure the form to your needs.

1. Open the xdp template you used in the FLM wizard:



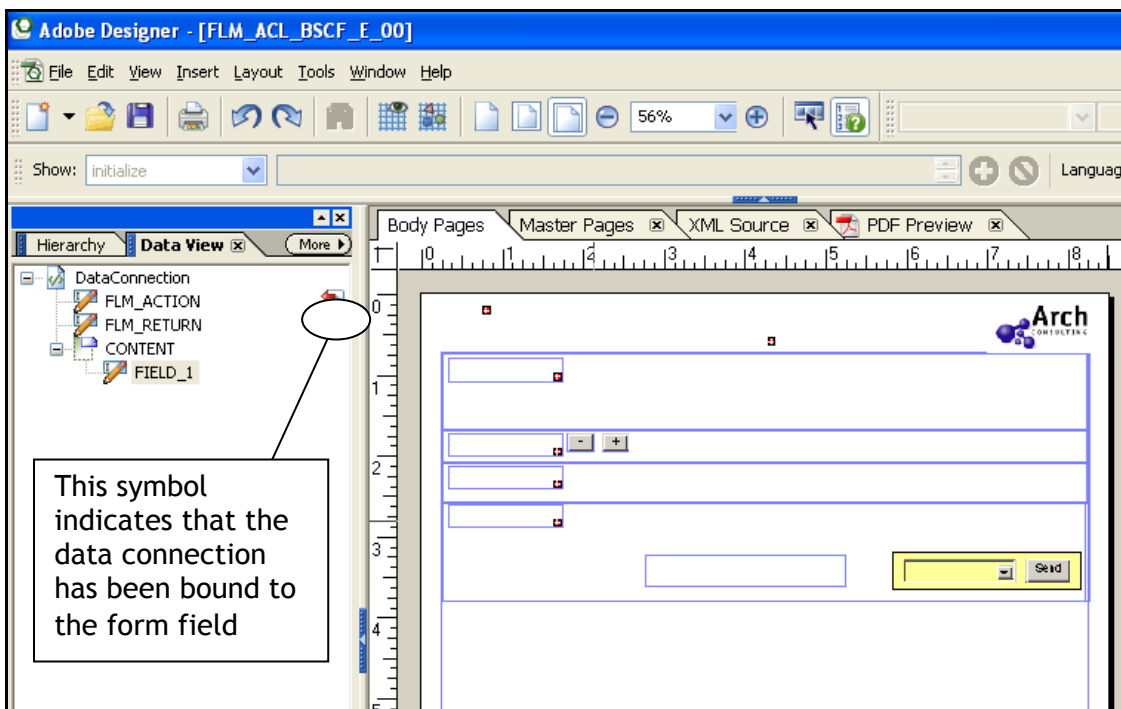
2. With this form open, go to File>New Data Connection and select 'get description from XML schema'. Then open the xsd file created by the wizard, which should be in the 'FLM> Form Templates' folder. The following window will appear:



3. Select the option to embed the XML schema and click Finish.

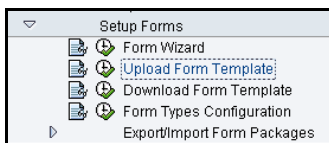
Now you will see in the 'Data View' tab a list of all the subforms and fields you created in the wizard, including two automatic options: 'FLM_ACTION' and 'FLM_RETURN'.

4. You then need to bind the data connection to the field hierarchy in the Designer template. To do this, drag and drop the fields from the 'data view' onto the corresponding field of the template

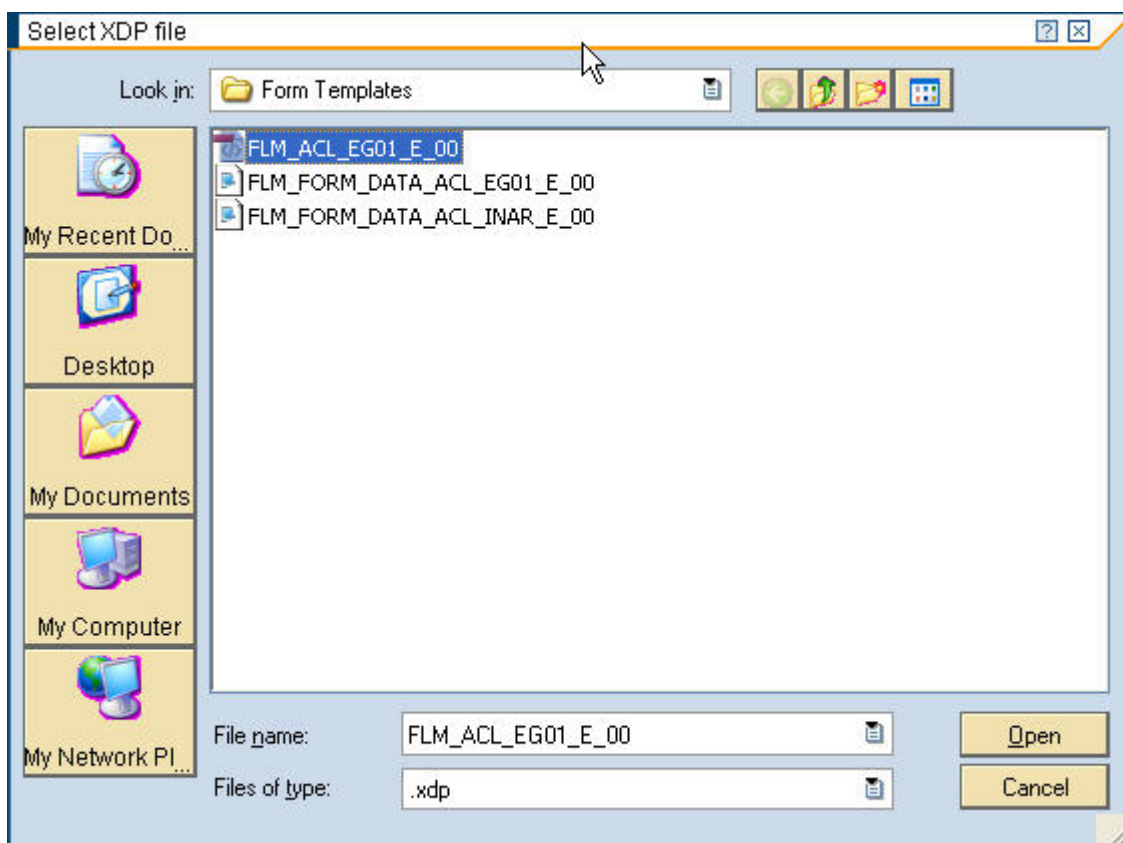


The details of the bindings are shown in the ‘bindings’ tab in the Object Window.

5. Once all the components of the data hierarchy, including ‘FLM_ACTION’ and ‘FLM_RETURN’ have been bound to their corresponding fields on the form, it is ready for use. Click ‘save’ to store the .xdp file with the .xsd data schema bound to it.
6. The next step is to upload the form template back into SAP. Use the ‘upload form template’ activity in Setup Forms:



In the activity, select the name of the XDP file and click ‘open’

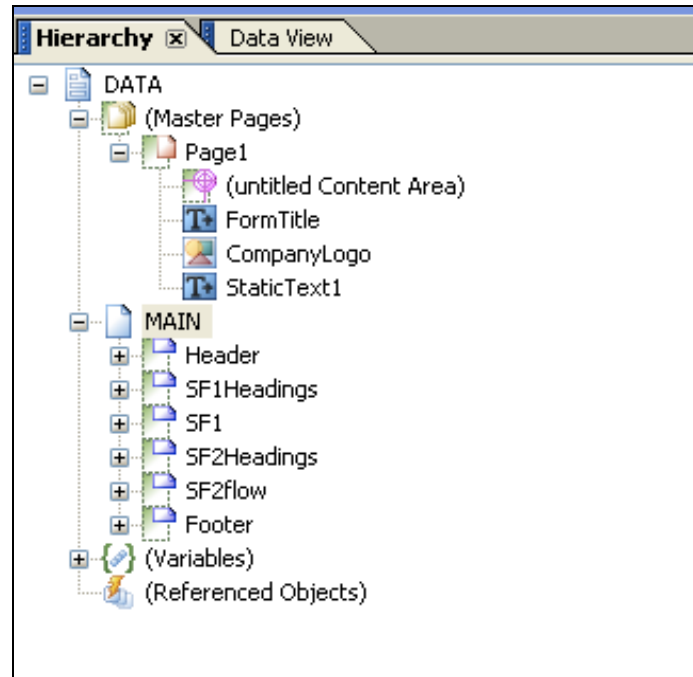


You will be prompted to confirm the form upload, then the form will be copied into SAP for access and use.

2.2 Data hierarchy

The top-level node in the data hierarchy represents the form. There are no settings to configure on this node. It is suggested that this node is called the same name as the 4-digit 'Form Type' in FLM.

A. Data hierarchy in Adobe Designer



Beneath the Data hierarchy are three types of node:

- **Master pages**
At least one master page must be defined. Typically the form header/footer/logos - anything that should be repeated on each page, should be put inside the master page. Also a Content Area must be defined, which is basically a box defining the boundaries for all the other form fields. This ensures that form fields do not overlap with headers and footers on the master. The Content area by default does not have a name it should be named and by default this should be called 'content'. This is critical when coding as un-named objects or subforms make referencing in script very difficult page.
- **Subforms**
There are two types of subform; 'Flowed' and 'Positioned'.
The 'Flowed' subforms generally follow the hierarchy represented by the xsd data schema. The 'Positioned' subforms represent physical groupings of fields at the same logical level within the xsd data schema. A flowed approach will allow the form to work in a dynamic nature. For example, for an invoice you would define one header one detail line and a footer. The data would then be bound and for each line on the invoice a new detail line will be created, and the Reader will then flow all trailing subforms correctly.
- **Variables**
There are two types of variable; 'Variables' and 'Script Objects'.
Variables are used for storing any data string. You can use Javascript to read or write to these variables. 'Script Objects' are (groups of) Javascript sub-routines that can be called at any point using Javascript behind form/field events.

- **Referenced Objects**
These are objects that are not naturally occurring in the form, but may appear based upon certain events. The classic example of this is a Footer subform that should only appear if a page break occurs on another subform.

2.3 Subform definition and binding

The top-level subform is always FLOWED.

As you work your way logically through the subform then each subform is FLOWED until the final subform in which data fields sit, which is POSITIONED. Fields must always sit inside POSITIONED subforms if they are to be displayed, as it is not possible to physically position a field inside a FLOWED subform.

It is essential to have this hierarchy of flowed subforms since the option to set a subform to be repeating is only available if the parent subform (the next level up) is FLOWED. However, it is possible for either a FLOWED or a POSITIONED subform to be repeating as long as the parent subform is FLOWED.

It is not necessary to have a 1:1 match between the subforms in the form and the subforms designed within FLM: The form will often require additional subforms in order to handle the field positions as well as the data flow.

It is not necessary to bind each subform in the data hierarchy, unless the subform has repeating rows or is nested within another subform that has repeating rows. In other cases the binding of the field is sufficient.

2.3.1 Binding for non-repeating subforms.

There is no need to bind non-repeating subforms as all the fields within those subforms are effectively at root level, be they header fields, item headings, footer fields etc.

The binding of the fields inside non-repeating subforms takes the following form:

```
$record.Header.form_status
```

In this example 'Header' is the data node in the xsd data schema and 'form_status' is the field name under the 'Header' node in the xsd data schema. If there were more sub-nodes in the data schema then they would all appear in the binding path.

2.3.2 Binding for repeating subforms.

It is essential to bind subforms that can repeat.

The binding of the subform takes the following form:

```
$record.EmployeeUtil.ResUtilisation[*]
```

In this example, 'EmployeeUtil' is the data node in the xsd data schema that sits underneath the root ('DATA') node. 'ResUtilisation' is the data node in the xsd schema that contains repeating fields. The notation of the [*] means for all nodes that match this Xpath query.

If the repeating subform was defined with a parent of 'ROOT' then the binding would take the following form, as no path of the node hierarchy would be necessary:

\$record.ITEM[*]

In this example, of course, the node is called 'ITEM' and its parent is the root 'DATA' node.

The binding of a repeating subform is different depending on whether there are any nested subforms within the subform to be bound.

If there are no nested subforms then the POSITIONED subform is set to have repeating rows, and is bound to the data node.

If there are further nested subforms within the subform that contain data to be bound, then the FLOWED subform is set to have repeating rows and is bound to the data node.

This often means that there is a POSITIONED subform without binding (ie 'Normal' binding) that sits in-between the bound FLOWED subform and the fields within that subform. This in turn means that the binding for those fields is takes a different form.

Note that there is an option to define a data schema with separate nodes for the repeating subform and for all the fields at the level of repeating subform:

Workorder	FLOWED, REPEATING
Workorder_info	POSITIONED
<fields>	
Workorder_details	FLOWED
Employee_details	POSITIONED, REPEATING
<fields>	
Plant_details	POSITIONED, REPEATING
<fields>	

In this scenario the subforms 'Workorder', 'Employee_details' and 'Plant_details' must be bound, but it is not necessary to bind the subform 'Workorder_info'. (While it is not needed it is best practice to always bind subforms to the corresponding node.) However, if there is a corresponding data node(ie. There are no data fields are defined under the 'workorder' node, just nodes for info, employees and plants) then the Workorder_info subform should be bound, and this means that the binding is the same as for a normal POSITIONED subform regardless that there are nested subforms.

In the scenario where the POSITIONED subform is bound then the binding for fields within a repeating subform takes the form:

EBELP

ie. The immediate parent of the field defines the full data path so there is no need to define it again. Binding always work on a relative path unless they start with \$record in which cae they become absoulte paths.

In the scenario where the FLOWED subform is bound but not the child POSITIONED subform, then the binding for fields within a repeating subform takes the form:

```
$record.ITEM[*].EBELP
```

This is because the immediate parent of the field (the POSITIONED subform) has no binding, so the full data path is required.

Do not attempt to bind both the FLOWED and POSITIONED subforms to the same data node.

2.3.3 Binding for nested subforms.

The binding for nested subforms follows exactly the same pattern as above. This means that if there are further nested subforms then the FLOWED subform must be bound, otherwise the POSITIONED subform should be bound.

Since nested subforms always sit inside FLOWED subforms that are bound, then the binding of the nested subform is not fully declared, but instead it is just the subform name:

```
plantdata[*]
```

The binding of the fields is the same as described in the previous section.

Note that in all cases the bound subform is the subform that is set to have repeating rows.

2.4 Subform look and feel hints and tips.

- All subforms in the data hierarchy should be set to 'Allow page breaks within content' except for the bottom-level positioned subforms where it may be desirable to keep the form fields together. The allow Page break will allow a set of flowing subforms to natural flow over a page and then you can define a trailer (footer) and header subform to be placed on the next page. These can include Referenced Subforms. If you have a group of subforms that you want to bind together then set the Page Break option of and then use the 'Keep With' Flag. An example of this is an invoice line with special details section which could be long text that could span one or more lines. Therefore truning of the Page Break option here and setting the keep with options will ensure if the detail line fits but the special instrcutions does not it will place both on a new page.

3 To import an existing Adobe form into FLM

1. Save Form A (the template of the form you wish to import)
2. Open in designer

3. Print
4. Open a form that was created in FLM to copy data
5. Create FLM variables important: this must be done in order for the form to be successfully imported into the FLM wizard
 - a. Go into existing FLM form
 - b. Select 'variable' and open XML schema
 - c. Copy block of script after the words "End of Do Not Modify"
 - d. Return to Form A and go File> Form Properties> Variables> Create
 - e. Create variable with recognisable name and enter a value (e.g. 1)
 - f. Go into XML schema and locate 'variable' section
 - g. Replace variable text with variable code copied from FLM form
 - h. Save
6. Tidy up subform hierarchy by removing excess subforms and renaming them and the fields contained within them
7. Print finalised hierarchy and annotate fields with types and Userexit requirements
8. Go into FLM form wizard and create subform/field hierarchy in the data designer to match the hierarchy in designer
9. Import 'Form A' as the designer template in the wizard
10. Save the new form in FLM
11. Go back to the designer file> File> new data connection> XML schema
12. Select newly created xml file and check the option to embed the xml schema
13. Drag and drop the new fields from the hierarchy onto the corresponding fields on the page
14. Go back to the FLM-created form and select the final drop-down and submit buttons. Drag-and-drop into the standard library buttons menu on the top right
15. Create a new subform at the end of Form A to contain the FLM return/action buttons
16. Insert the buttons on the last page of Form A and copy the relevant script associated with them into the XML schema of Form A. FLM Return> Event> Form Ready> Copy to New
17. Upload form template into SAPGUI
18. Configure Userexits
19. Check form in portal