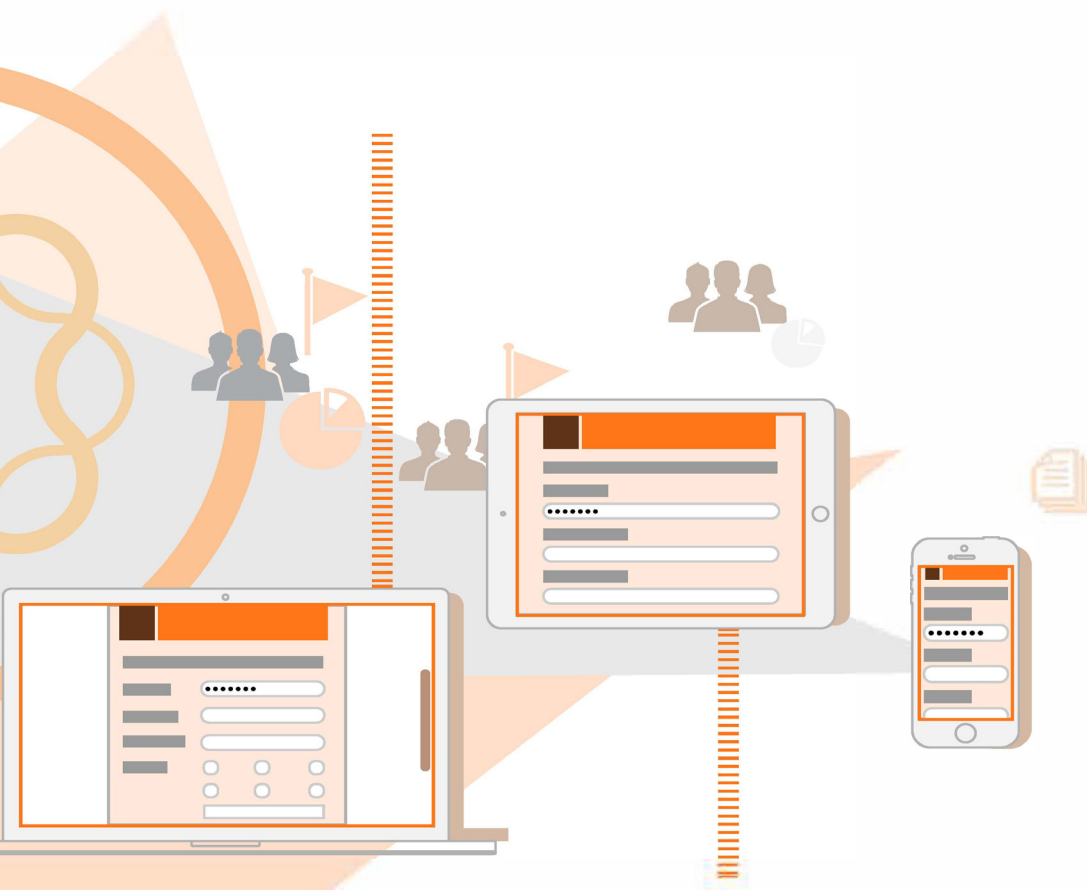# Designer Samples

AEM 6.5 Forms

Legal notices

For legal notices, see http://help.adobe.com/en_US/legalnotices/index.html.

Contents

Sample Form Snippets and Forms

Sample Form Snippets

Designer includes a selection of complete sample form snippets for you to use or refer to when creating your own forms.

The form snippets are installed in the EN\Samples\Form Snippets folder under the Designer installation folder.

The sample form snippets are listed in this table.

NOTE: *This document uses terms Adobe Experience Manager Forms, AEM Forms, AEM Forms on JEE, and LiveCycle interchangeably.*

Snippet sample	Files
Calculate the Field Sum Demonstrates how to calculate sums of fields that are at different levels in the form hierarchy.	CalculateSum.xdp
Change the Fill Color of a Field Demonstrates how to change the fill color of a field at run time.	ChangeFillColor.xdp
Using Conditional Breaks Based on Data Demonstrates the application of conditional breaks to repeating subforms.	ConditionalPageBreak.xdp ConditionalPageBreak.xml
Using Data Nominated Subforms Demonstrates controlling the display of subforms from a choice subform set.	DataNominatedSubform.xdp DataNominatedSubform.xml
Using Expandable Fields Demonstrates how a field expands and flows to accommodate data.	ExpandableFields.xdp
Import and Export Data Demonstrates importing and exporting data in the form.	ImportExportData.xdp ImportExportData.xml
Opening an Application Demonstrates opening an application using scripting.	LaunchApp.xdp

Snippet sample	Files
Using a Paper Forms Barcode Demonstrates the data capacity of a paper forms barcode. It also shows how data capacity changes when mixed case and special characters are included in the data.	PaperFormBarcode.xdp
Validate a Data Demonstrates how to validate a date value using scripts.	RangeValidation.xdp

Calculating sums of fields

The Calculate the Field Sum snippet demonstrates how to calculate sums of fields that are at different levels in the form hierarchy when the form is opened.

To see the following example and others, visit the AEM forms DeveloperCenter.

In this example, the sum of the repeating fields is calculated.

To calculate the sum of repeating fields in a form

1)   Add a `calculate` event to the Sum field:

```
var fields = xfa.resolveNodes("NumericField1[*]");

var total = 0;
for (var i=0; i <= fields.length-1; i++) {
total = total + fields.item(i).rawValue;
```

```
    }

this.rawValue = total;
```

In this example, the sum of the fields nested within a repeating field is calculated.

NumericField1   4

NumericField1   4

NumericField1   4
_____

Sum   12

To calculate the sum of fields nested within a repeating subform

1) Add a `calculate` event to the Sum field:

```
var fields = xfa.resolveNodes("detail[*].NumericField1");

var total = 0;
for (var i=0; i <= fields.length-1; i++) {
total = total + fields.item(i).rawValue;
}

this.rawValue = total;
```

In this example, the sum of the fields on the first page is calculated.
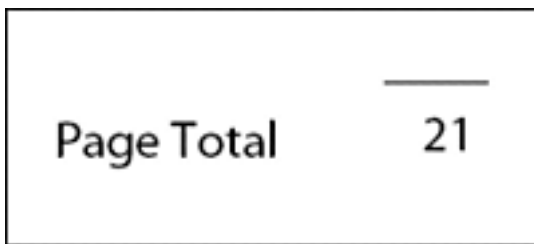
To calculate the sum of the fields on the first page

1) Add a `calculate` event to the Sum field:

```
var fields = xfa.layout.pageContent(0 , "field", 0);

var total = 0;
for (var i=0; i <= fields.length-1; i++) {
if (fields.item(i).name == "NumericField1") {
total = total + fields.item(i).rawValue;
}
}

this.rawValue = total;
```

Changing the fill color of a field

The Fill Color of a Field snippet demonstrates how to change the fill color of a field at run time.

In this snippet, when the Apply Fill Color button is clicked, the text field color changes to gray.

To change the fill color of a field

1)   Add the following script to the Apply Fill Color button.

```
TextField1.fillColor = "200,200,200";
```

To see similar examples, visit the AEM forms DeveloperCenter.

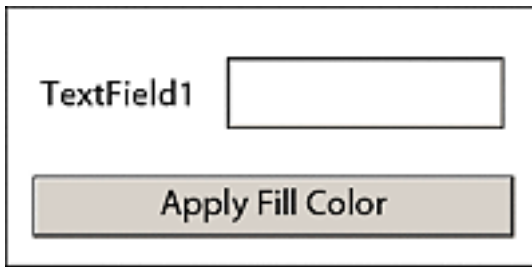## Using conditional breaks in repeating subforms

The Using Conditional Breaks Based on Data snippet demonstrates the application of conditional breaks to repeating subforms. As opposed to paginating in response to data overflow, conditional breaks allow you to manually control how a subform breaks in a form based on a series of checks called *conditional statements*. In this example, three content areas are used to organize the data into three columns when the form is rendered. When the category changes, a break is inserted and the new category starts at the top of the next column.

The conditional breaks are specified in the Edit Conditional Breaks dialog box. To display this dialog box, select the Movies subform object and click Edit on the Pagination tab of the Object palette.
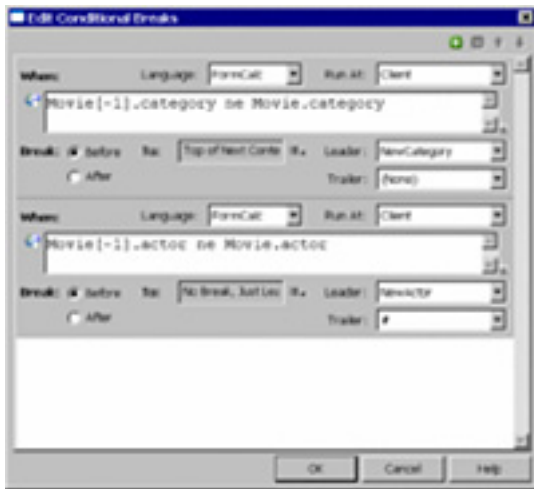
To see similar examples, visit the AEM forms DeveloperCenter.

Controlling the display of subforms based on a conditional statement

This Data Nominated Subforms snippet demonstrates how one of several alternative subforms can be displayed on the form based on a conditional statement. You use data binding to set the condition that determines which of the subforms appears on the form.

The subforms are wrapped in a subform set. In Designer, a subform set can control the display of subforms based on the flow of data. In this example, the Comedy subform is displayed each time the category in the data equals comedy, the Action subform is displayed each time the category equals action, and the Drama subform is displayed each time the category in the data equals drama.
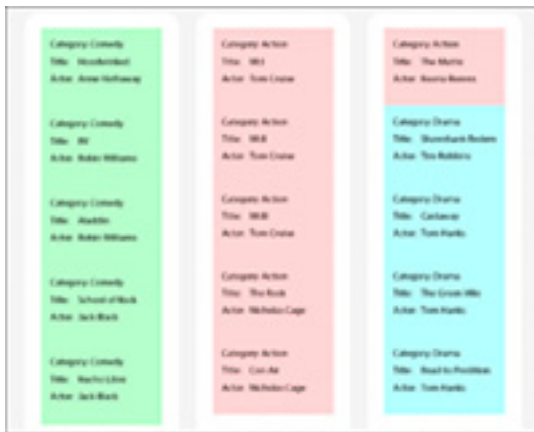
The alternative subforms are specified in the Edit Data Nominated Subforms dialog box. To display this dialog box, select the Movie subform set object and click Edit Alternatives on the Subform Set tab of the Object palette.

To see similar examples, visit the AEM forms DeveloperCenter.

You must merge the form with data when generating the PDF file. If not, the data will be unavailable for the script that decides which subform to instantiate. Users who open the PDF file in Adobe® Acrobat® will get a warning message. To avoid the warning, you can add script to verify the existence of a data value before executing the script.

Allowing multiple lines of text in text fields

The Expandable Fields Snippet demonstrates how a text field expands to accommodate multiple lines of data. In this example, five question fields must adjust to accommodate more than one line of text.

For each Text Field object, you specify that the text fields can accept one or more lines of wrapping text by selecting the Allow Multiple Lines option in the Field tab of the Object palette.
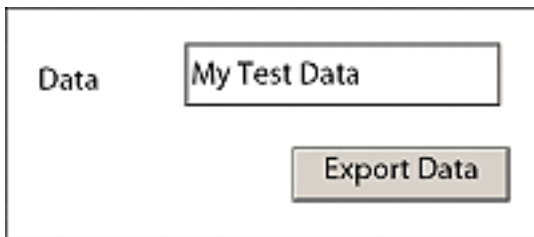
Also, to properly display text that doesn't fit on the same page, select the Allow Page Breaks within Content option for the Questions subform.

To see similar examples, visit the AEM forms DeveloperCenter.

Prompting for a file name for exported and imported data

The Import and Export Data snippet demonstrates how to prompt the user to provide a file name for exported or imported XML data.
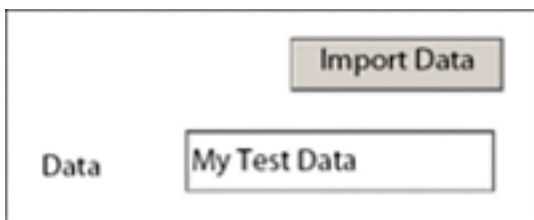
In this example, the script for the Export Data button does not supply a file name; therefore, the form prompts the user to specify a name for the exported data file.

To add script to the Export Data button

```
xfa.host.exportData("",0);
```

Similarly, the script for the Import Data button does not supply a file name; therefore, the form prompts the user to specify a name for the imported data file.

To add script to the Import Data button

```
xfa.host.importData("",0);
```

To see similar examples, visit the AEM forms DeveloperCenter.

Opening an application from a form

The Opening an Application snippet demonstrates opening a browser to a user-specified URL and a file in Acrobat or Adobe® Reader® from a form.

In this example, when the Open button is clicked, the user-specified URL opens in a separate browser container.

To add script to open a browser to a user-specified URL

1) Add the following script to the Open button:

```
app.launchURL(TextField1.rawValue, true);
```

In this example, when the Open button is clicked, the user-specified URL opens in the current container.
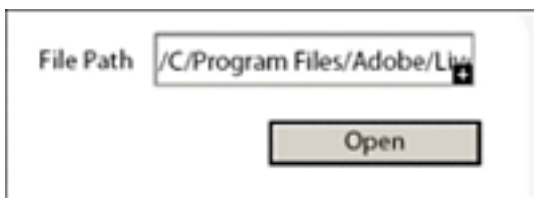
To add script to open a browser to a user-specified URL

1) Add the following script to the Open button:

```
xfa.host.gotoURL(TextField2.rawValue, 0);
```

In this example, when the Open button is clicked, the user-specified PDF file opens in Acrobat or Adobe Reader.

To add script to open a user-specified PDF file

1) Add the following script to the Open button:

```
app.openDoc(FilePath.rawValue);
```

To see similar examples, visit the AEM forms DeveloperCenter.

Viewing the data of a paper forms barcode

The Using a Paper Forms Barcode snippet demonstrates the capacity of a paper forms barcode. It also shows how data capacity changes when mixed case and special characters are included in the data.

In this example, entering a number of characters less than 255 and then clicking the Generate Data button changes the barcode data capacity. Also, deselecting the Use UpperCase Characters Only check box demonstrates how the data capacity of the barcode is reduced when mixed case is used.

Number of characters to generate ☐ 255

☒ Use uppercase characters only

Generate Data

Generated Data
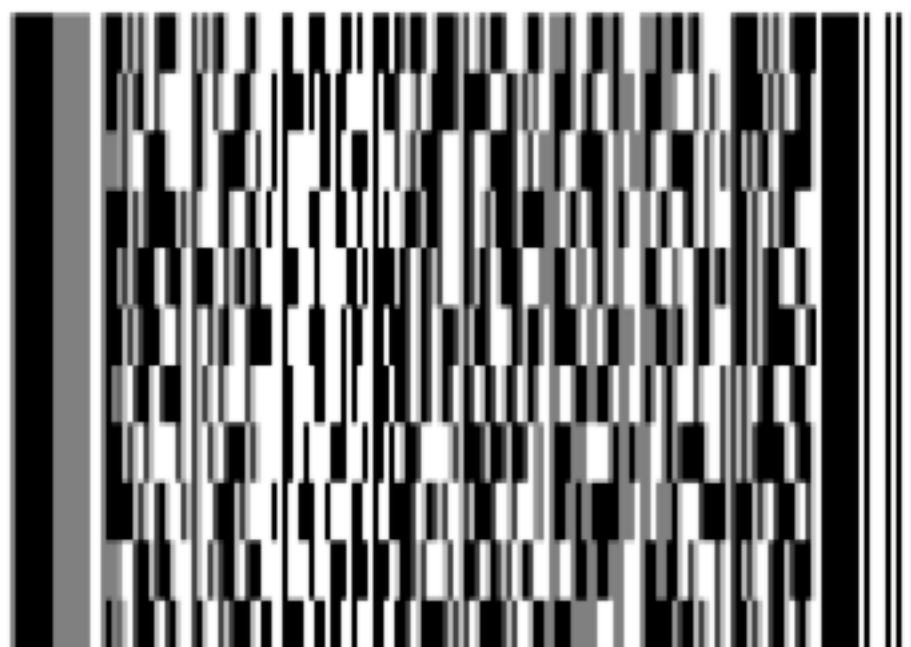
To add script for the Number of Characters to Generate text box

```
form1.#subform[0].CharstoGen::validate - (JavaScript, client)
// The maximum number of characters to generate is 255.
if (this.rawValue > 255)
{
this.rawValue = 255;
false;
}
```

To add script for the Use Uppercase Characters Only check box

```
form1.#subform[0].UpperCaseOnly::change - (JavaScript, client)
if (this.rawValue == 1)
{
CharacterBase.value = " ABC DEF GHI JKL MNO PQR STU VWX YZ,";
}
else
{
CharacterBase.value = " ABC DEF GHI JKL MNO PQR STU VWX YZa bcd efg hij klm nop
qrs tuv wxy z01 234 567 890 <xml>,";
xfa.host.messageBox("When data is encoded into a PDF417 barcode, all UPPER CASE
data requires less space in a barcode.  When switching back and forth from upper
case to lower case, special characters must be embedded into the barcode to let
the decoder or scanner know that a case change has taken place.  As you will be
able to see from the example, the data capacity of the barcode is reduced when
mixed cases are used.", "Upper Case" ,3, 1);
}

if (CharstoGen.rawValue >= 255)
CharstoGen.rawValue = 255;
```

To add script to the Generate Data button

```
form1.#subform[0].Button1::click - (JavaScript, client)
// Display random data in Paper Form barcode.
var RandomLetters = "";
var sequence = "";
var FieldFillSize = 0;
var nIndex = 0;

RandomLetters = CharacterBase.value;
FieldFillSize = CharstoGen.rawValue;

for (var i = 1; i <= FieldFillSize; i++)
{
nIndex = Math.floor(Math.random() * RandomLetters.length);
sequence = sequence + RandomLetters.charAt(nIndex);
}

BarcodeData.rawValue = sequence;
```

To add script to the barcode

```
BarcodeData.rawValue = sequence;
form1.#subform[0].Barcode::calculate - (JavaScript, client)
//{{Start Generated Content//
var is705ViewerRequired = false;
//End Generated Content}}//

function createBarcodeContent()
{
   // TODO: Encode your barcode content here
   var barcodeContent = " ";
   barcodeContent = BarcodeData.rawValue;
   return barcodeContent;
}

// Assign content to the barcode.
// Note: Do not assign a null value to
// the barcode. Otherwise, the barcode is
// displayed as blank.
if(createBarcodeContent() != null)
{
   this.rawValue = createBarcodeContent();
}
else
{
   this.rawValue = " ";
}
// Final sanity check, located here not for
// efficiency but for consistency and correctness
if (is705ViewerRequired && xfa.host.version < 7.05)
{
   this.rawValue = " ";
}
```

To see similar examples, visit the AEM forms DeveloperCenter.

Validating a date by using a script

The Validate a Date snippet demonstrates validating a date based on a date range that the user enters.

In this example, the user selects a date from a calendar and then specifies the start and end dates to validate the specified date against. The form compares the specified date against the date range and reports whether the specified year, month, and day are valid.

To add script to the Validate button

```
form1.#subform[0].Button1::click - (JavaScript, client)
Status.clearItems(); //Clear the status listbox.

// Create a date() for parsing information.
var sDate = Date.rawValue;
var oDate = util.scand("yyyy-mm-dd", sDate);
if(oDate == null)
{
xfa.host.messageBox("Pleae enter a valid date.");
exit;
}

// Store date values.
var nYear = oDate.getFullYear();
var nMonth = oDate.getMonth() + 1;  // 0 based
var nDay = oDate.getDate();

// Validation flags.
var bStartYear = false;
var bEndYear = false;
var bStartMonth = false;
var bEndMonth = false;
var bStartDay = false;
var bEndDay = false;

// Validate the year range.
if((StartYear.rawValue == null) || (StartYear.rawValue <= nYear))
bStartYear = true;
if((EndYear.rawValue == null) || (EndYear.rawValue >= nYear))
bEndYear = true;
if(bStartYear && bEndYear)
Status.addItem("valid year");
else
Status.addItem("invalid year");
```

```
// Validate the month range.
if((StartMonth.rawValue == null) || (StartMonth.rawValue <= nMonth))
bStartMonth = true;
if((EndMonth.rawValue == null) || (EndMonth.rawValue >= nMonth))
bEndMonth = true;
if(bStartMonth && bEndMonth)
Status.addItem("valid month");
else
Status.addItem("invalid month");

// Validate the day range.
if((StartDay.rawValue == null) || (StartDay.rawValue <= nDay))
bStartDay = true;
if((EndDay.rawValue == null) || (EndDay.rawValue >= nDay))
bEndDay = true;
if(bStartDay && bEndDay)
Status.addItem("valid day");
else
Status.addItem("invalid day");
```

To see similar examples, visit the AEM forms DeveloperCenter.

Sample Forms

Designer includes a selection of complete sample forms. Each one includes a form design and the final version of the form. Some also include sample data and/or a schema. The samples illustrate both simple and complex form design techniques. The sample forms are installed in the EN\Samples\Forms folder under the Designer installation folder.

NOTE: Some sample form designs, such as Purchase Order Dynamic.xdp, are meant to be merged with data. Saving these forms in PDF format may generate warnings when you open them in Acrobat or Adobe Reader.

The following table lists all the samples and their location in the Designer installation folder.

Form Sample	Location
Dunning Notice Demonstrates the behavior of a dynamic PDF form.	\EN\Samples\Forms\Dunning Notice
E-Ticket Demonstrates the behavior of a dynamic PDF form. *Designer Help* explains the numbered notes that appear in the E-Ticket.pdf sample file.	\EN\Samples\Forms\E-Ticket
Grant Application Demonstrates the behavior of an interactive PDF form.	\EN\Samples\Forms\Grant Application
Purchase Order Demonstrates the behavior of dynamic, interactive, and dynamic interactive PDF forms.	\EN\Samples\Forms\Purchase Order
Scripting Helps you to experience scripting with the FormCalc and JavaScript™ languages. You can enter a script into a box and run it by clicking a button.	\EN\Samples\Forms\Scripting
Subform Set Demonstrates the behavior of a dynamic PDF form. The sample consists of four sub-samples that demonstrate the instantiation behavior of the Subform Set object according to the varying relationships of the subform type and occurrence values.	\EN\Samples\Forms\SubformSet
Tax Receipt Demonstrates the behavior of a dynamic PDF form.	\EN\Samples\Forms\Tax Receipt

Dunning Notice Sample

The Dunning Notice sample demonstrates the behavior of a dynamic PDF form.

The sample has three available Dunning levels. The information appropriate for each level should be printed along with a list of the unpaid documents.

The sample includes these files:

File	Description
Dunning Notice.xdp	Designer form file located in the Forms folder.
Dunning Notice Level1.xml Dunning Notice Level2.xml Dunning Notice Level3.xml	XML data files located in the Data folder. The data file for the Level 3 Dunning notice contains enough invoices to overflow onto a second page.
Dunning Notice.tif	Image file located in the Images folder.
Dunning Notice Level1.pdf Dunning Notice Level2.pdf Dunning Notice Level3.pdf	Rendered forms with merged data located in the Outputs folder.

To run the Dunning Notice sample

1) Open the Dunning Notice.xdp file, located in \EN\Samples\Forms\Dunning Notice\Forms, in Designer.

2) To specify the preview options, select File > Form Properties.

3) Click the Preview tab and do the following tasks:

   - In the Preview Type list, select Print Form.

   - In the Data File box, browse to the Data folder and select one of the sample data files (for example, Dunning Notice Level1.xml).

   - Click OK.

4) To preview the form, click the Preview PDF tab.

The output sample PDF files in the Outputs folder show the rendered form merged with each of the three sets of data. Use these files to compare to the results of the previewed form.

About the Dunning Notice form

The form demonstrates several features.

Form hierarchy structure

> The Dunning Notice form is based on the data structure in order to take advantage of the implicit data binding process.

Master page

> Two master pages are required. The first one displays the company logo, form title, and static text. This page is the first page to print and appears on an odd-numbered printed page. The second master page is used for every subsequent page. The same static text is displayed as well as the page numbering.

Page numbering

> The page numbers are calculated values obtained by inserting run-time properties into a Text object.

Flowed content

> When data is merged with the form, the subforms are placed one below the other by setting the subforms' parent (dunningNotice) Content option to Flowed. The Content option is on the Subform tab of the Object palette. The Flow Direction list, also on the Subform tab of the Object palette, is set to Top to Bottom. The subform margins, specified in the Layout palette, add the extra spacing required between two subforms.

> The header, level1, level2, level3, and closing subforms also have flowed content. The field margins add the extra spacing required between two objects.

Subform occurrence

> The form includes a number of subforms that work together to properly accommodate the data. The subforms themselves are a mixture of repeating and non-repeating subforms. The minimum count values of all subforms, except the detail header and closing subforms, are set to 0 to specify that only the required subform will print. The maximum value of each non-repeating subform is set to 1. The repeating subforms do not have a maximum occurrence value because the number varies for each set of data. The minimum count value of the closing subform is set to 1 because no data is available to instantiate the subform.

Detail header

> The detail header displays the column headings and should print before the first detail line. This detail header information should repeat at the top of the page when the detail lines can no longer fit on the current page and flow on to a new page. Setting the minimum count value of the detailHeader subform to 1 ensures that the header is always printed at least once.

Overflow leader

The detailHeader subform is associated with the document subform as its overflow leader. This feature forces the overflow leader subform of the document subform to print every time the document subform flows onto a new page. Because there are no fields in the detailHeader subform, the Data Binding option, on the Binding tab of the Object palette, is set to No Data Binding.

Global fields

The currency value is provided once in the data file. By setting the default binding of the currency field to Global and by using this field in multiple locations, the value is replicated in each occurrence of the field.

Image field

Because the company logo may vary, the logo file is embedded in the data file. An Image Field object is used to display the image.

Maximum number of characters for each field

The maximum characters value for each field is specified as per data specifications. The Max Chars option is on the Field tab of the Object palette.

Expandable fields and anchor position

To accommodate data values of varying length, the Expand to Fit option in the Layout palette is selected for the billToAddress, Salutation, and Closing fields. The anchor position of these fields is set to allow the field to expand in the proper direction.

Variables

The closing of every paragraph is the same. For easier maintenance, its value is defined as a variable, and a script is used to reference this value and display it. Because the field's value is calculated, the Data Binding option is set to No Data Binding.

Scripting

The address values should be displayed as a block. A JavaScript script is used to concatenate the data values. Because the billToAddress field's value is calculated, the Data Binding option is set to No Data Binding.

Multiple line field

The billToAddress field's Allow Multiple Lines property, on the Field tab of the Object palette, is selected so that the calculated value can print on several lines.

Floating fields

The level3 subform contains both text and fields. Using floating fields results in a natural flow of the information. A display pattern can be specified directly on the field and displayed in the paragraph. Use the Patterns option on the Field tab of the Object palette to specify a display

pattern. The floating field is a hidden field inserted into the Text object. The reference to the floating field is represented by the field name between curly braces; for example, {fieldName}.

Picture patterns

Some values are easier to read by applying a display pattern. For example, the level3 subform's deadline value is formatted with a medium-length date for data pattern value on the Binding tab of the Object palette.

E-Ticket Sample

The E-Ticket sample demonstrates and explains the behavior of a dynamic PDF form. The sample form contains numbered notes represented as black circles with white numbers. The notes are explained in Aboutthe E-Ticket output.

This form prints a customized Travel Package that contains an airline itinerary, Customs form, medical declaration, flight transfer slips, accommodation information, and boarding passes. The sections of the package are expected to print using different paper orientations.

This sample includes these files:

File	Description
E-Ticket.xdp	Designer form file located in the Forms folder.
E-Ticket.xml	XML data files located in the Data folder.
E-Ticket.tif	Image file located in the Images folder.
E-Ticket.pdf	Rendered forms with merged data located in the Outputs folder.

To run the E-Ticket sample

1) Open the E-Ticket.xdp file, located in \EN\Samples\Forms\E-Ticket\Forms, in Designer.

2) To specify the preview options, select File > Form Properties.

3) Click the Preview tab and do the following tasks:

   - In the Preview Type list, select Print Form (Two-Sided).

   - In the Data File box, browse to the Data folder and select the E-Ticket.xml sample data file.

   - Click OK.

4) To preview the form, click the Preview PDF tab.

The output sample PDF file in the Outputs folder shows the rendered form merged with data. Use this file to compare to the results of the previewed form.

About the E-Ticket output

These numbered notes correspond to the numbered circle icons in the sample form file.

1) The Package cover page should print once, and it should use a portrait orientation.

    - The coverPage subform is invoked by the presence of the data group cover page.

    - The coverPage subform is placed on the portrait-oriented master page because it is the first page area defined in the Page Set.

    - The coverPage subform has a minimum count value of 0 and a maximum count value of 1. The Min Count and Max Count options are on the Binding tab of the Object palette.

    - The image file is linked to the form. The link is provided as the URL of an Image object.

    - In the "Items in this travel package" section, a JavaScript script sets the field caption text to plural when appropriate.

    NOTE: The data values that must print in several locations in the Travel Package are created as global fields; for example, the lastName, firstName, and initial fields.

2) The Airline Itinerary header information should print before the listing of the flight itineraries, and it should use a landscape orientation.

    - The subform and subformSet occurrence and subformSet relationship type is the required combination to invoke the flightHeader subform.

    - The itinerary subformSet is instantiated only if a flight subform is instantiated. This is possible by setting the subformSet Min Count value to 0. After the subformSet is instantiated, the flightHeader subform is invoked because it has a Min Count value of 1. This is as a result of setting the subform set Type value, on the Subform Set tab of the Object palette, to Use All Subforms in Order.

    - The flightHeader subform is associated with the landscape master page. The flightHeader subform will be placed On Page "landscape".

    - The flightHeader subform is associated with the flight subform as its overflow leader. The flightHeader subform should print at the top of every page in the Airline itinerary section. The Data Binding field on the Binding tab of the Object palette is set to Use Global Data because the information must repeat.

3) The flights are specific to the Travel Package and may vary between itineraries.

    - The flight subform is invoked by the presence of the data group flight. The flight subform has a Min Count value of 0 and an unlimited maximum occurrence value. This means that it will be instantiated only if data is available for it and will repeat as many times as necessary to print the available data. The maximum value is unlimited when the Repeat Subform for Each Data Item option is selected and the Max option is deselected. The Repeat Subform for Each Data Item option is on the Binding tab of the Object palette.

- The flight subform will be placed following the previous subform.

4) A single Customs declaration is required for each family. The declaration should print using a portrait orientation.

- The customs subform is invoked by the presence of the data group customs.

- The customs subform has a Min Countvalueof 0 and Max value of 1.

- The customs subform is associated with the portrait-oriented master page. The customs subform will be placed at the top of the portrait-oriented page, thereby forcing a new page.

5) A single Medical Declaration should print per page, and it should use a portrait orientation.

- The medical subform is invoked by the presence of the data group medical.

- The medical subform has a Min Count value of 0 and an unlimited maximum occurrence value.

- The medical subform is associated to the portrait-oriented master page, and it will be placed at the top of the portrait-oriented page, thereby forcing a new page.

- A JavaScript script will concatenate the passenger's home address information. The homeAddress field's Allow Multiple Lines option is enabled to properly display the block of information. The Allow Multiple Lines option is on the Field tab of the Object palette.

6) The Accommodation header information should print before the listing of the hotels, and it should use a landscape orientation.

- The subform and subformSet occurrence and subformSet relationship type is the required combination to invoke the hotelHeader subform.

- The accommodation subformSet is instantiated only if a hotel subform is instantiated. This is possible by setting the subformSet Min Count value to 0. After the subformSet is instantiated, the hotelHeader subform is invoked because it has a Min Count value of 1. This is the result of setting the subform set Type to Use All Subforms in Order.

- The hotelHeader subform is associated with the landscape master page. The hotelHeader subform will be placed on the landscape oriented page

- The hotelHeader subform is associated with the hotel subform as its overflow leader. The hotelHeader subform should print at the top of every page in the Hotel accommodation section. The fields are set to global because the information is required to repeat.

- A JavaScript script is used to concatenate the lastName and firstName data values.

- The guest address values should be displayed as a block. A JavaScript script is used to concatenate the data values.

7) The hotel listing is specific to the Travel Package and may vary between itineraries.

- The hotel subform is invoked by the presence of the data group hotel. The hotel subform has a Min Count value of 0 and an unlimited maximum occurrence value. It will be

instantiated only if data is available for it, and it will repeat as many times as necessary to print the available data.

- The hotel subform will be placed after the previous subform.

- The room type values should be displayed as a block. A JavaScript script is used to concatenate the data values. Because the roomName fields value is calculated, the Data Binding option is set to No Data Binding.

8) A single boarding pass should print per page, and it should use a landscape orientation.

- The boarding subform is invoked by the presence of the data group boarding.

- The boarding subform has a Min Count value of 0 and an unlimited maximum occurrence value. This means that it will be instantiated only if data is available for it, and it will repeat as many times as necessary to print the available data.

- The boarding subform is associated with the landscape master page. The boarding subform will be placed on top of the landscape-oriented page, thereby forcing a new page.

Grant Application Sample

The Grant Application sample demonstrates the behavior of an interactive PDF form.

The sample includes a grant application to be filled for the institution or foundation that issued the form.

The sample includes the file named Grant Application.pdf. Both the form design and the form that the user will interact with are in the Forms folder.

To run the Grant Application sample

1)   Open the Grant Application.pdf form, located in \EN\Samples\Forms\Grant Application\Forms, in either Acrobat or Adobe Reader.

2)   Click in the Title of Project field and start filling the form.

About the Grant Application form

The form demonstrates several features.

Page numbering

The page numbers are calculated values obtained by inserting run-time properties into a Text object.

Maximum number of characters for each field

The Max Chars option, on the Field tab of the Object palette, for each field is specified as per data specifications.

Multiple line field

The Description field Allow Multiple Lines option, on the Field tab of the Object palette, is selected so that users can enter several description lines.

Picture patterns

The user is expected to enter some values in a specific format. A validation message, defined on the Value tab of the Object palette, will prompt the user to enter values in the proper format if the user has not done so. For example, a script will validate the BudgetDirectCosts field's user-entered value.

Custom library objects

The U.S. States Custom Library Object was used to create the State and OrgState drop-down lists.

Tabbing order

The default tabbing order has been modified to allow a column flow. Display the tabbing order by selecting the Tab Order command from the View menu.

Purchase Order Samples

The four Purchase Order samples demonstrate the behavior of dynamic, interactive, and dynamic interactive PDF forms.

The Dynamic Purchase Order sample includes these files:

File	Description
PurchaseOrder.xdp	Designer form file located in the Forms folder.
Purchase Order.xml	XML data files located in the Data folder.
Purchase Order.tif	Image file located in the Images folder.
Purchase Order.pdf	Rendered form with merged data located in the Outputs folder.

The Dynamic Interactive Purchase Order sample includes these files:

File	Description
Purchase Order.pdf	Designer form file located in the Forms folder.
Purchase Order.tif	Image file located in the Images folder.

The Form Fragments Purchase Order sample includes these files:

File	Description
Purchase Order Dynamic Interactive.pdf Purchase Order Dynamic.xdp Purchase Order Interactive.pdf	Designer form file located in the Forms folder.
Purchase Order Canada.xml Purchase Order US.xml	XML data files located in the Data folder.
Purchase Order.tif	Image file located in the Images folder.

File	Description
CountryScript.xdp DeliverToAddress.xdp Footer.xdp FooterCanada.xdp FooterUS.xdp Header.xdp OrderedByAddress.xdp PartNoScript.xdp	Fragments located in Fragments folder.
Purchase Order Canada.pdf Purchase Order US.pdf	Rendered form with merged data located in the Outputs folder.

The Interactive Purchase Order sample includes these files:

File	Description
Purchase Order.pdf	Both the form design and the form that the user will interact with, located in the Forms folder.
Purchase Order.tif	Image file located in the Images folder.

The Schema Purchase Order sample includes these files:

File	Description
Purchase Order.xdp	Designer form file located in the Forms folder.
Purchase Order.xml	XML data file located in the Data folder.
Purchase Order.xsd	Schema file located in the Schema folder.
Purchase Order.tif	Image file located in the Images folder.
Purchase Order.pdf	Rendered form with merged data located in the Outputs folder.

To run the E-Ticket sample

1) Open the E-Ticket.xdp file, located in \EN\Samples\Forms\E-Ticket\Forms, in Designer.

2) To specify the preview options, select File > Form Properties.

3) Click the Preview tab and do the following tasks:

- In the Preview Type list, select Print Form (Two-Sided).

- In the Data File box, browse to the Data folder and select the E-Ticket.xml sample data file.

- Click OK.

4) To preview the form, click the Preview PDF tab.

The output sample PDF file in the Outputs folder shows the rendered form merged with data. Use this file to compare to the results of the previewed form.

To run the Dynamic and Schema Purchase Order

1) In Designer, open the appropriate form:

- Purchase Order.xdp file located in \EN\Samples\Forms\Purchase Order\Dynamic\Forms

- Purchase Order.xdp file located in \EN\Samples\Forms\Purchase Order\Schema\Forms

2) To specify the preview options, select File > Form Properties.

3) Click the Preview tab and do the following tasks:

- In the Preview Type list, select Print Form (Two-sided).

- In the Data File box, browse to the Data folder and select the Purchase Order.xml sample data file.

- Click OK.

4) To preview the form, click the Preview PDF tab.

The output sample PDF file in the Outputs folder shows the rendered form merged with data. Use this file to compare to the results of the previewed form.

To run the Interactive and Dynamic Interactive Purchase Order

1) In Acrobat or Adobe Reader, open the appropriate form:

- Purchase Order.xdp file located in \EN\Samples\Forms\Purchase Order\Interactive\Forms

- Purchase Order.xdp file located in \EN\Samples\Forms\Purchase Order\Dynamic Interactive\Forms

2) Click in the P.O. Number field and start filling the form.

The output sample PDF file in the Outputs folder shows the rendered form merged with data. Use this file to compare to the results of the previewed form.

Scripting Sample

The Scripting sample helps you to experience scripting with the FormCalc and JavaScript languages. You can enter a script into a box and run it by clicking a button. The sample includes a variety of field types that the script can use.

The sample includes the file named Scripting.pdf. Both the form design and the form that the user will interact with are located in the Forms folder.

To run the Scripting sample

1) Open the Scripting.pdf form, located in \EN\Samples\Forms\Scripting\Forms, in either Acrobat or Adobe Reader.

2) Enter a script in the Enter Your Script Here field.

3) Enter any input that is expected by the script.

4) Click the appropriate button to run the script. Any output appears in the Result field. If you prefer, you can assign the results to another field.

5) After the script runs, click Clear Fields in order to clear the fields to rerun or alter the script.

   To try a simple script, use one of these samples.

JavaScript

```
if (NF1.rawValue < NF2.rawValue)
NF3.rawValue = 0
else
   NF3.rawValue = NF1.rawValue - NF2.rawValue
```

FormCalc

```
if (NF1 < NF2) then
NF3 = 0
else
NF3 = NF1 - NF2
endif
```

Both scripts use the numerical fields $NF1$, $NF2$, and $NF3$. The user can type numbers into the $NF1$ and $NF2$ fields, and the script subtracts the numbers ($NF1-NF2$) and displays the difference in the $NF3$ field. If the result is negative, the $NF3$ value is set to 0.

Subform Set Sample

The Subform Set sample demonstrates the behavior of a dynamic PDF form. Four samples demonstrate the instantiation behavior of the Subform Set object according to the varying relationships of the subform type and occurrence values. The sample form contains numbered notes represented as black circles with white numbers. The notes are explained in Aboutthe Subform Set output.

The sample includes these files:

File	Description
SubformSet1.xdp SubformSet2.xdp SubformSet3.xdp SubformSet4.xdp	Designer form file located in the Forms folder
DataA.xml DataB.xml	XML data files located in the Data folder
SubformSet1 DataA.pdf SubformSet1 DataB.pdf SubformSet2 DataA.pdf SubformSet2 DataB.pdf SubformSet3 DataA.pdf SubformSet3 DataB.pdf SubformSet4 DataA.pdf SubformSet4 DataB.pdf	Rendered forms with merged data located in the Outputs folder

To run the Subform Set sample

1) Open one of the forms in the Forms folder, located in \EN\Samples\Forms\SubformSet, in Designer.

2) To specify the preview options, select File > Form Properties.

3) Click the Preview tab and do the following tasks:

- In the Preview Type list, select Print Form (Two-sided).

- In the Data File box, browse to the Data folder and select one of the sample data files (for example DataA.xml).

- Click OK.

4) To preview the form, click the Preview PDF tab.

The output sample PDF files in the Outputs folder show the rendered forms merged with each of the two sets of data. Use these files to compare to the results of the previewed form.

About the Subform Set output

These numbered comments correspond to the numbered black-circle icons on the Subform Set sample form.

1. Data

> Two sample data files are available for this sample. This section provides the name of the data file merged with the form along with the content of the data file. This information is useful for comparing the data displayed in section 3, "Subform Relationship".

2. Subform Occurrence

> Each form contains three subforms. This section provides the minimum count and maximum occurrence values of the subforms. The occurrence values are specified in the Binding tab of the Object palette.

Subform specifications per form

Form	subformA	subformB	subformC
SubformSet1.xdp	Min Count = 0 Max =-1*	Min Count = 0 Max = -1*	Min Count = 0 Max = -1*
SubformSet2.xdp	Min Count = 0 Max = 1	Min Count = 0 Max = 1	Min Count = 0 Max = 1
SubformSet3.xdp	Min Count = 1 Max = 1	Min Count = 1 Max = 1	Min Count = 1 Max = 1
SubformSet4.xdp	Min Count = 2 Max = 2	Min Count = 1 Max = 1	Min Count = 3 Max = 3

* The Max value is unlimited when the Repeat Subform for Each Data Item option in the Binding tab of the Object palette is selected.

3. Subform Relationship

> The third section contains three columns used to compare the layout of the subforms inside different relationship types:

Column	Type	Comments
1	Use All Subforms in Order	The subforms are instantiated in the order in which they are declared in the form. This has the effect of potentially reordering the content to satisfy the form order.
2	Select One Subform from Alternatives	The subforms are exclusive of each other, and only one subform may be instantiated. The determination of which subform to instantiate is based on the data.

Tax Receipt Sample

The Tax Receipt sample demonstrates the behavior of a dynamic PDF form.

This form prints three copies of a tax receipt on a single page: the foundation's copy, the chapter's copy, and the income tax copy.

The sample includes these files:

File	Description
Tax Receipt.xdp	Designer form file located in the Forms folder
Tax Receipt.xml	XML data files located in the Data folder
Signature.tif Tax Receipt.tif	Image files located in the Images folder
Tax Receipt.pdf	Rendered forms with merged data located in the Outputs folder

To run the Tax Receipt sample

1) Open the Tax Receipt.xdp form in Designer.

2) To specify the preview options, select select File > Form Properties.

3) Click the Preview tab and do the following tasks:

4) In the Preview Type list, select Print Form (Two-sided).

- In the Data File box, browse to the Data folder and select the Tax Receipt.xml sample data file.

- Click OK.

5) To preview the form, click the Preview PDF tab.

The output sample PDF file in the Outputs folder shows the rendered form merged with data. Use this file to compare to the results of the previewed form.

About the Tax Receipt form

The form demonstrates several features.

Form hierarchy structure

The Tax Receipt form is based on the data structure in order to take advantage of the implicit data binding process.

Master page

Because the footer text needs to be displayed at the bottom of each page, it is defined on the Master Page.

Flowed content

The instances of the receipt subform are placed one below the other, which is made possible by setting the taxReceipt subform Content option to Flowed. The Content option is in the Subform tab of the Object palette. The Flow Direction, also in the Subform tab, is set to Top to Bottom. The subform margin settings are defined in the Layout palette and add the extra spacing required between two subforms.

Subform occurrence

The receipt subform contains the tax receipt information, including fields and static elements. Because the receipt is expected to print three times on a single page, the maximum value of the receipt subform is set to 3. The maximum value is defined in the Binding tab of the Object palette.

Global fields

The donor information is provided once in the data file. Making the donor fields global enables these values to print for every tax receipt copy.

Image field

Because the treasurer's signature may vary, the signature image link is provided in the data file. An image field is used to display the signature image.

Maximum number of characters for each field

The maximum characters value for each field is defined in the Field tab of the Object palette and is specified as per data specifications.

Expandable fields and anchor position

To accommodate data values that vary in length, the Expand to Fit option in the Layout palette is selected for the receiptSerial and copyLabel fields. The anchor position of these fields is set to allow the field to expand in the proper direction.

Scripting

The address values should be displayed as a block. A JavaScript script is used to concatenate the data values. Because the donorAddress field value is calculated, the Data Binding option is set to No Data Binding. The default binding is defined in the Binding tab of the Object palette.

Multiple line field

The donorAddress field Allow Multiple Lines option in the Field tab of the Object palette is selected so that it can print on several lines.

Picture patterns

Some values are easier to read by using the Patterns option in the Field tab of the Object palette to apply a display pattern. For example, the date value is formatted with a long style date format.