

# LiveCycle ES Update 1 – FormGuides

by Anthony Rumsey

## Table of contents

Introduction.....	1
Getting Started.....	1
Form Design.....	2
Form Scripting .....	2
Guide Builder.....	3
Customizing Appearance.....	3
Extensions .....	3
Flex Class .....	4
Event Listener.....	4
Flex Property .....	4
Data Model Property.....	4
Debugging .....	5
Accessibility .....	5
About the Author.....	5

## Introduction

Form guides are Flash Player compatible wizard-like panels that help guide people through a data capture experience. Form guides can dynamically change and adapt based on input data to ensure that only relevant and accurate data is captured. These rich and engaging experiences help reduce transaction abandonment rates that are prevalent with more complex data collection interfaces.

The creation of a form guide begins with the Guide Builder tool provided by LiveCycle Designer. Guide Builder allows form designers to create an alternate “guided” data experience for a paginated PDF form. With this capability organizations can meet regulatory document, archive and signature requirements with the PDF form view while providing a compelling and easy to use guided data capture experience.

This technical guide is intended to provide form developers, Flex programmers and project planners a set of best practices to plan, develop, customize and maintain world class data capture experiences with form guides. It should be used as a planning tool for project teams as they begin to develop form guides and use them in their LiveCycle Enterprise Suite applications. Proper planning can reduce the development effort, ease the application maintenance burden and improve performance. The advice and best practices contained in this guide cover LiveCycle ES Update 1 (8.2). For a detailed overview of how Form Guides work, refer to <http://www.avoka.com/blog/?p=12> and <http://www.avoka.com/blog/?p=38> blog entries created by Avoka.

This document is not intended to replace or duplicate product documentation.

## Getting Started

Whether you are creating a new form or have an existing document-based form it is worthwhile to spend some time planning before starting the creation of a form guide.

## Directory Structure

Both the source form design and its associated form guide definition can reference a variety of external resources. Since the locations of resources stored in the guide definition are paths relative to the source XDP or PDF the form design should also use relative paths. Well structured directories will assist making this content easier to find and reduce the possibility of errors occurring.

A typical form design may reference external images, data and schemas. A form guide will also include references to one or more flex libraries (SWC).

While placing all resources being referenced in the same directory is one solution this will become harder to maintain for all but the simplest cases. Instead, directories under the source form design should be created to hold the various types of resources being referenced (images, data, flex libraries, fragments). Examples of possible directory names include: assets, images, data, bin, frag. For more complex applications where resources may be shared across forms it is also possible to place items in parent directories.

---

*You should be familiar with the principles and concepts contained in the “Getting Started with Form Guides” document.*  
[http://www.adobe.com/go/learn\\_lc\\_fgGetStart\\_82](http://www.adobe.com/go/learn_lc_fgGetStart_82)

---

## Form Design

While any form design can be made into a form guide the process of developing the guide can be made easier by following these guidelines:

1. Group form sections into subforms.
2. Do not use nested subforms in repeating subforms. Form guides assume all repeating form objects are immediate children of the repeating subform.
3. Avoid using form fields on master pages.
4. If possible, use field captions instead of static text. This will avoid having to link static text to panel items in the guide definition if you want captions to appear in the form guide.
5. When executing a web service call in a Form Guide, refer to the Form Nation blog, [http://blogs.adobe.com/formnation/2008/11/executing\\_web\\_service\\_calls\\_in.html](http://blogs.adobe.com/formnation/2008/11/executing_web_service_calls_in.html)

## Form Scripting

Form scripting is a common area where issues may occur when trying to develop a form guide from an existing form design. The reason has to do with the level of scripting support provided by form guides and behaviour differences between the Flash runtime and Acrobat/Reader.

Form guides only support a subset of the full XFA scripting model. Any scripting that falls outside of the subset will fail to compile. Referring to the form guide scripting support documentation is imperative in order to ensure the scripts on your form design remain compliant.

Only use JavaScript. When your script is set to run on the client form guides only support JavaScript and not FormCalc. Server side scripts can use either script language.

When developing script on a form design that will be executed in a form guide be sure to follow these guidelines:

1. Scripts that assign a value must be expressed as:

```
this.rawValue = price.rawValue * quantity.rawValue;
```

The script `price.rawValue * quantity.rawValue` will compile since it's within the XFA subset but will not work in a form guide as XFA automatically assigns the result of the calculation to the value of the field.

2. Nested functions in JavaScript are not supported.
3. Maintain a library of common form guide functions in a script object that can be reused.
4. The `formReady` event only fires once each when the PDF and form guide load.
5. The `calculate` event will fire when referenced field values change AND when switching between the guide and PDF view.
6. Only use `xfa.host.messageBox`. Message boxes will appear as default Flex Alert boxes in a form guide. Custom CSS could be added to a form guide's style to match the Alert box style to the rest of the application.
7. Form Guides are unable to wait for user response (eg. Yes/No) from a message box. Such behaviour is possible but requires Flex custom components to be build (advanced usage).
8. When checking for empty field values look for null and zero-length.
9. When scripting the value of Date fields use `field.dataNode.value` to ensure consistent behaviour in both environments. In form guides accessing a date field's `rawValue` returns its formatted value.
10. Do not use `xfa.event.change` since it is not supported by form guides.

---

Full details of form guide scripting are available online. [http://www.adobe.com/go/learn\\_lc\\_fgScriptSupport\\_82](http://www.adobe.com/go/learn_lc_fgScriptSupport_82)

---

### Conditional Script Execution

When adding script to your form it may be appropriate to limit execution to only form guides (Flash) or provide alternate execution paths depending on the host environment. This can be achieved by querying the `xfa.host.name` property.

---

```
if (xfa.host.name == "Flash")
{
//script will execute in form guide
} else {
//script will execute in Acrobat/Reader
}
```

---

### Guide Builder

Guide Builder is a tool that is launched from LiveCycle Designer under the tools menu. This tool provides the ability to build a guide definition, set properties, add display rules, define styles and import custom Flex libraries.

The following suggestions may be useful when using Guide Builder to build your guide definition:

1. You can copy/paste from the Designer canvas into the guide tree. Sometimes this will be much easier especially on large forms with complex hierarchies.
2. When building your guide hierarchy you can also click **Hide Properties** to reduce the Guide Builder window size as well as set it to remain always on top through the context menu.
3. The refresh form objects button needs to be clicked whenever there are changes made in Designer.
4. Dragging a subform onto the guide tree will only add the immediate child form objects.
5. Use the **View** drop down list to limit the amount of form objects that appear.

### Customizing Appearance

Guide Builder provides the ability to define and create style sheets that can be used in one or multiple form guides. Using Guide Builder to define your styles ensure that the correct style names are applied that correspond to style name used in the default guide and panel layouts. These same style names can also be used in custom layouts in order to take advantage of the styling support provided by Guide Builder.

By default Guide Builder will save your style sheet as a SWC file. Saving as a SWC allows for the CSS plus any referenced images to be included as one file. This is required since form guides only allow the style to be specified as one file.

Use these guidelines when determining to save a customized appearance as a CSS or SWC:

1. When in doubt always save your style as a SWC
2. If you have any images referenced in your style you **MUST** save as SWC.
3. You can import a CSS that has image references into Guide Builder in order to save it all as a SWC. Whenever you make changes to your CSS however you will need to re-import the CSS and save as SWC again.
4. If you have any custom style definitions that are edited outside of Guide Builder and do not reference any images you can import your CSS and not worry about saving as a SWC. Using this method means you can make changes to your CSS outside of Guide Builder and they will get picked up automatically at runtime without having to update the guide definition.

By default Guide Builder saves your style (SWC or CSS) in the same directory as the source XDP. While this location is fine you are advised to change the location to a relative location just for style definitions. This will become particularly important once you start sharing your style definitions across multiple guides.

---

*LiveCycle Designer help has an entire section devoted to using Guide Builder. [http://www.adobe.com/go/learn\\_lc\\_designer\\_82?context=Adobe\\_LiveCycle\\_Designer\\_Help&topic=000943](http://www.adobe.com/go/learn_lc_designer_82?context=Adobe_LiveCycle_Designer_Help&topic=000943)*

---

---

Instructions on how to customize form guides in Flex Builder can be found in the following document. [http://www.adobe.com/go/learn\\_lc\\_fgCustomizing\\_82](http://www.adobe.com/go/learn_lc_fgCustomizing_82)

---

---

The guide source projects can be found under your LiveCycle ES installation. `LiveCycle_ES_SDK/samples/FormGuides/GuideSource`

---

## Extensions

Extensions to a form guide are any external libraries that are referenced using Guide Builder. These libraries could contain new guide or panel layouts that you have created in Flex Builder or they could even be a third party library that includes new UI controls.

The following guidelines can be followed when developing any type of extension to a form guide:

1. Set up your Flex project according to the instructions in the customizing form guide documentation.
2. After adding both form guide SWCs to your project set their link type to External. This will keep your SWC file size to a minimum.
3. Import all the guide source projects into Flex builder.
4. Try to use the source of one of the existing guide samples as a starting point by copying and pasting into a new MXML file.

A form guide consists of both guide and panel layouts.

A guide layout defines the visual layout and structure of a form guide that remains constant throughout a form-filling session. A panel layout defines the visual layout and presentation of objects on a panel in the form guide hierarchy.

While it is advisable to make use of the various layouts provided by default in many cases a need will arise for a customization to be made. Customizations could be as simple as moving existing controls to far more complex layouts and behaviours. Making use of Customize Appearance in Guide Builder can dramatically change the look of a layout and may reduce the need to perform further customizations in Flex.

## Guide Layouts

When creating a custom guide layout the following guidelines should be considered:

1. Override the `createChildren()` function to set up any event listeners
2. Listening for `GAEvent.INITIALIZED` is useful to ensure the guide and its model has been fully initialized.
3. Retain as many style class names as possible. The style names used by the default layouts refer to styles that are defined by Guide Builder in Customize Appearance. Retaining these style names in your own layout means that its style can be easily customized as well.

## Panel Layouts

When creating a custom panel layout the following guidelines should be considered:

1. Override the `documentDescriptor` property (`get/set`) in order for the slot replacement algorithm to work correctly.
2. Retain as many style class names as possible.
3. Place `PanelItem` components inside a hidden container to access their data only. This technique is useful if you have a complex control that binds to multiple data items since in form guides each panel item can only bind to one data node.

## Navigation Controls

Navigation controls are interactive objects that allow a user to navigate through a guide. Form guides provide a number a navigation controls that are provided in the default wrappers.

## Field Controls

A field control is a UI component that is used to display the data contained in a form design object. Guide Builder offers some suggestions by default but in some situations, using a different Flex object may make for a user experience that is more engaging. A custom field control can either extend an existing Flex UI component or it can even be made completely from scratch if desired.

When creating custom field controls it is good practice to understand how the form guide runtime responds to field changes and how it updates the data model. The following table lists what events the form guide runtime listens to and how it maps data from the Flex control to the XFA data model. The first matching flex class that is found will be used.

---

Refer to the form guide `ActionScript` reference when creating custom components. [http://www.adobe.com/go/learn\\_lc\\_fgActionScript\\_82](http://www.adobe.com/go/learn_lc_fgActionScript_82)

---

Flex Class	Event Listener	Flex Property	Data Model Property
Repeater	valueCommit	selected	selectedItems
DateField	valueCommit	text	formattedValue
CheckBox	valueCommit	selected	booleanValue
Radiobutton	valueCommit	selected	booleanValue
Button	valueCommit	label	rawValue
ComboBase	valueCommit	selectedItem	selectedItem
ComboBase (editable)	valueCommit change	text	rawValue
ListBase	change	selectedItems	selectedItems
Default *	valueCommit	text	formattedValue
Default *	valueCommit	selected	booleanValue
Default *	change	value	rawValue

\* By default if a Flex control is not one of the types listed the form guide runtime will determine if the control has a text, selected or value property and act accordingly.

### Debugging

Setting up a debug environment for when run time errors occur in a guide or if it behaves incorrectly will be very beneficial. A single debug environment can be used for all your form guide development. The customizing form guide documentation contains detailed instructions on how to accomplish this.

Even if you are not using any custom Flex libraries the debugger may help determine with your form script once it is executed in the context of a guide.

A debugger version of the Flash Player will assist with debugging the Flex project. If you do not have a debug version of the Flash Player, you can also use the debug log.

### Accessibility

In order to make an application created with Flex (and by extension Form Guides) accessible, you need to enable accessibility for that application. Doing so imports the accessibility object for each component used in the application, which increases the size of the SWF file slightly. For Form Guides, the Guide Accessible option on the renderFormGuide service operation must be set to true.

Furthermore, the following guidelines should be followed when designing for accessibility:

1. The Cobalt standard wrapper is the best choice for accessibility.
2. Limit the number of different panels that are used in the guide to prevent confusing a visually impaired user with different locations of help instructions.
3. When using custom components consider the 28 Flex components that have accessibility already built in ([http://www.adobe.com/accessibility/products/flex/best\\_practices.html#components](http://www.adobe.com/accessibility/products/flex/best_practices.html#components)).

### About the Author

As a Computer Scientist at Adobe for the past ten years, Anthony Rumsey is currently using the next generation of LiveCycle to ensure it is capable of meeting the future demands of enterprise application development. Previously, Anthony has been actively involved in developing a number of data capture solutions at Adobe such as the Correspondence Management solution accelerator and the initial release of form guides. He also was a developer on the original form server technology that is now included in LiveCycle Forms ES. Visit Anthony's blog: <http://blogs.adobe.com/formnation>.

---

For more information on Flex accessibility in general visit: <http://www.adobe.com/accessibility/products/flex/>

---



**Adobe**  
**Adobe Systems Incorporated**  
 345 Park Avenue  
 San Jose, CA 95110-2704  
 USA  
[www.adobe.com](http://www.adobe.com)

Adobe, the Adobe logo, Acrobat, Flex, LiveCycle, Macromedia, and Reader are trademarks or registered trademarks of Adobe Systems Incorporated in the United States, other countries or both. Microsoft, Windows, SharePoint, Excel, PowerPoint, and Outlook are trademarks or registered trademarks of Microsoft Corporation in the United States, other countries or both. Java, all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries or both. Other company, product and service names may be trademarks or service marks of others. All statements regarding Adobe future direction or intent are subject to change or withdrawal without notice and represent goals and objectives only.

© 2009 Adobe Systems Incorporated. All rights reserved. Produced in the USA.  
 05/09