Adobe

Customizing Form Guides Using Flex® Builder™

Adobe® LiveCycle® ES

July 2008

Update 1

Contents

About This Document

Welcome to Customizing Form Guides Using Flex® Builder™. This document provides information about creating custom form guide layouts, panel layouts, and controls using Adobe® Flex Builder.

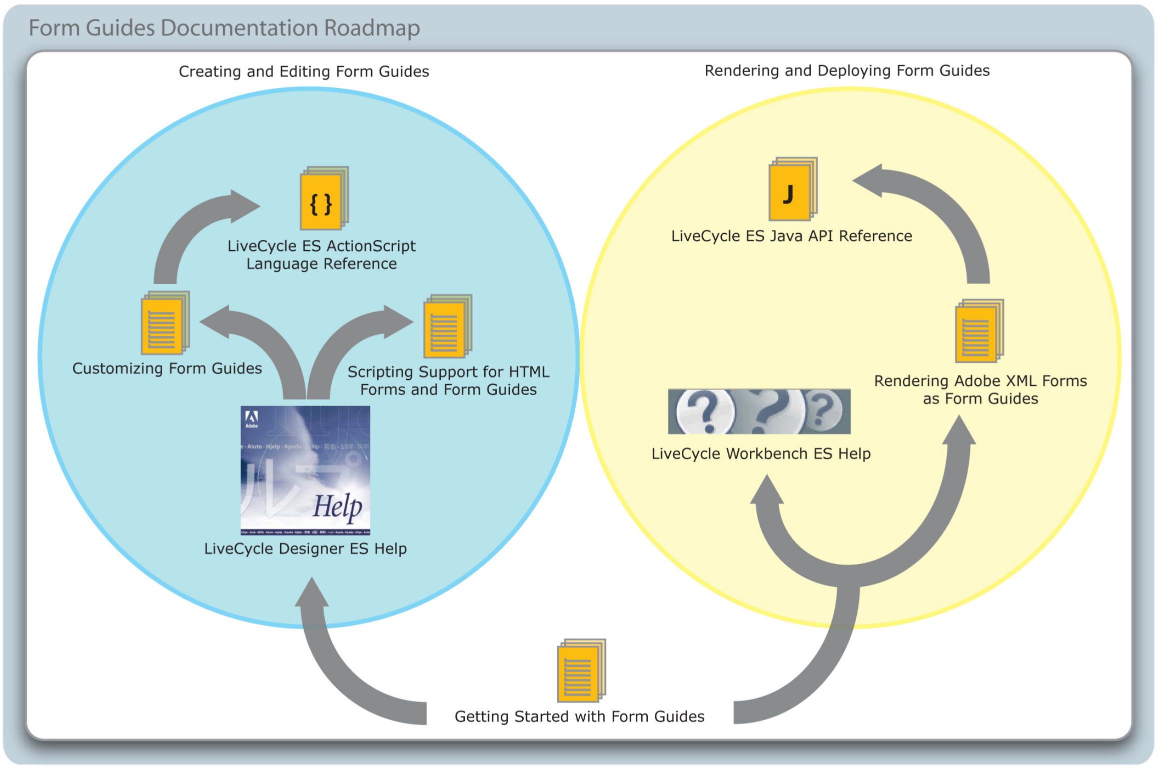
Who should read this document?

This document is intended for Flex developers who are interested in learning how to create custom form guide components to extend the form guide components shipped with Guide Builder, or new form guide components to meet specific needs.

A knowledge of form guides, Guide Builder, and Flex Builder is assumed.

Additional information

Adobe has a variety of resources about form guides focused at different audiences. The following illustration and table outline the resources available.


Form Guides Documentation Roadmap
Creating and Editing Form Guides
Rendering and Deploying Form Guides
LiveCycle ES ActionScript Language Reference
Customizing Form Guides
Scripting Support for HTML Forms and Form Guides
LiveCycle Designer ES Help
LiveCycle ES Java API Reference
Rendering Adobe XML Forms as Form Guides
LiveCycle Workbench ES Help
Getting Started with Form Guides

To view these resources, go to the location specified in the See column in the following table.

For information about	**See**
Creating and editing form guides using Guide Builder.	*LiveCycle Designer ES Help*, also available within Designer ES
The starting point for learning about form guides. This guide includes a walkthrough of creating a form guide from an existing form design and an example of how to render and deploy a form guide to Adobe Workspace ES by using processes created in Workbench ES.	*Getting Started with Form Guides*
The ActionScript™ classes and properties included with LiveCycle ES (Enterprise Suite) Update 1.	*LiveCycle ES ActionScript Language*
Rendering and deploying form guides using processes created in Workbench ES.	*LiveCycle Workbench ES Help*, also available within Workbench ES
Rendering a form guide using the Adobe LiveCycle Forms ES API	*Programming with LiveCycle ES*
The classes and methods included with LiveCycle ES.	*LiveCycle ES Java API Reference*
LiveCycle ES terminology	*LiveCycle ES Glossary*
Other services and products that integrate with LiveCycle ES	www.adobe.com
Patch updates, technical notes, and additional information on this product version	LiveCycle Technical Support

1 About Customizing Form Guides Using Flex Builder

Using Flex Builder, you can go beyond the form guide customizing options available in Guide Builder to create custom form guide components designed to suit your specific needs. For example, a Flex developer can create a new custom control that displays form guide sections and panels in a tree structure.

The process for creating new custom form guide components, whether they are form guide layouts, panel layouts, or form guide controls, follows the same general steps:

1. Create a new Flex Library project in Flex Builder. (See Creating Flex Library Projects for Custom Form Guides.)
2. Create new customized form guide components as part of the Flex Library project. See one of the following sections:
   - Creating Form Guide Layouts
   - Creating Panel Layouts
   - Creating Controls
3. Compile the Flex library project to a SWC file. (See Building your custom Flex library project.)
4. Import the SWC file into Guide Builder. (See *LiveCycle Designer ES Help*.)
5. Apply the new form guide components to a form guide.

2 Creating Flex Library Projects for Custom Form Guides

This chapter outlines how to configure your Flex development environment and how to create Flex library projects for custom form guide components.

Setting up your development environment

Before you can create custom form guide layouts, panel layouts, and components, you must first set up your development environment. To create custom form guides, you must have the following software installed on your computer.

Minimum required software

- Flex 3 SDK
- Access to the LiveCycle Designer ES file set

You can download the *Flex 3 SDK* trial.

Recommended software

- Flex Builder 3 Standard, Flex Builder 3 Professional, or Flex Builder 3 Plug-in for Eclipse
- Designer ES 8.2

See the system requirements for *Flex Builder* and for *Designer ES*.

Note: The term Flex Builder in this document refers to all three versions of Flex Builder.

Creating a new Flex library project

The first step in creating custom form guide components is to create a new Flex library project in Flex Builder. When completed, the Flex library projects you create must be compiled into a SWC file for importing into Guide Builder. A *SWC file* is an archive file for Flex components and other assets.

Each Flex library project you create must include the following elements:

- A reference to SWC files for a locale-specific form guide.
- A reference to SWC resource bundles for a locale-specific form guide.
- A folder structure with specific subfolders for each type of form guide component. For more information about creating the correct folder structure, see The folder structure for the Flex library project.

In addition, you can optionally include additional SWC files that contain MXML components or ActionScript classes that you want to leverage, or you can include other assets such as image files or videos.

➤ **To create a Flex library project for custom form guides:**

1. Start Flex Builder.
2. Select **File** > **New** > **Flex Library Project**.
3. Type a project name, assign a workspace, set the Flex SDK version to the default Flex 3 version of the SDK, and then click **Next**.
4. Click **Library Path**.
5. Click **Add SWC**.
6. Go to the \plugins\GuideBuilder folder where Adobe LiveCycle Designer ES is installed. By default, the path is C:\Program Files\Adobe\LiveCycle Designer ES\8.2\plugins\GuideBuilder. Add the following SWC files to the project:
   - GAClientRuntime.swc
   - XFAModel.swc
7. Click **Finish**.

After you create the Flex library project, you must create a folder structure to store your custom form guide components. Guide Builder requires that each type of custom form guide component be in a specifically named project folder.

The folder structure for the Flex library project

Each custom Flex library project must include form guide components stored in a specific folder structure. When you compile the project to a SWC file and import the SWC file into Guide Builder, form guide components are loaded from the project folders and made available through the Guide Builder interface.

The folder structure for the custom Flex library must contain the following elements:

- A top-level folder that has a unique name. The top-level folder in the basic folder structure example below is named *custom*.

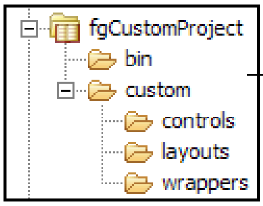
  Note: The folder cannot be named *ga*.
- At least one nested subfolder that has one of the following names:

  controls (form guide controls, such as panel navigation)

  layouts (panel layouts)

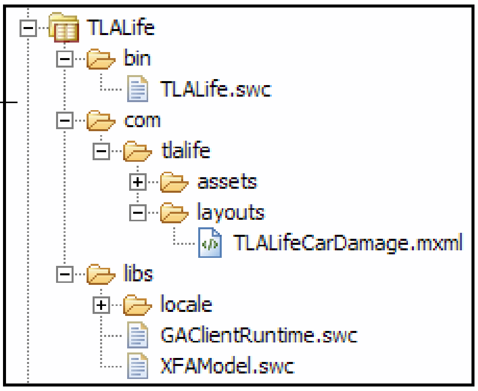
  wrappers (form guide layouts)

For example, the images below illustrate valid folder structures.



A basic folder structure that includes subfolders for all form guide components.

A more common project folder structure that includes one custom panel layout and additional assets, such as images, in the assets folder.



Importing sample Flex library projects

Included with the LiveCycle ES SDK are preconfigured Flex Builder projects that contain the form guide ActionScript source. You can import these projects to help you start creating custom components more quickly.

Caution: Rather than editing the sample files directly, you should copy content from the sample Flex Builder projects to help you get started creating new custom form guide components.

The LiveCycle ES SDK contains the following projects:

GuideControls: Contains copies of the form guide controls that are included with the Guide Builder SWC files.

GuideLayouts: Contains copies of the form guide layouts that are included with Guide Builder.

PanelLayouts: Contains copies of the panel layouts that are included with Guide Builder.

TLALife: A sample of a custom Flex library project. The TLALife project is used to create the TLA Life sample form guide that is included with Designer ES, and is used as an example throughout this document.

You can use these Flex Builder projects as a reference when creating your own custom Flex library projects.

➤ To import the sample form guide Flex Builder projects:

1. Start Flex Builder.
2. Click **File** > **Import** > **Flex Project**.
3. Select **Project Folder** and then click **Browse**. Go to the LiveCycle ES SDK\samples\FormGuides folder located where you installed LiveCycle ES (by default, C:\Adobe\LiveCycle8). Select one of the project subfolders, and then click **OK**.

4. (Optional) Choose the workspace where you want to import the new project.

5. Click **Finish**.

After you import the project, you can compile it and add the custom library to your form guide using Guide Builder. Repeat the steps to import the other sample projects included with LiveCycle ES.

Building your custom Flex library project

After you create custom form guide components, make them available for use in form guides. You must compile the Flex library project to a SWC file and then import the SWC file into Guide Builder.

➤ **To compile your custom Flex library project:**

1. Start Flex Builder.

2. If your Flex library project is set to compile automatically, proceed to step 3. Otherwise, in the Navigation view, select your project and click **Project** > **Build Project**.

3. Make the compiled project SWC file available to Guide Builder in one of the following ways:
   - If you are compiling form guides as part of a LiveCycle ES solution and you want to distribute your SWC files using the LiveCycle ES repository, open Workbench ES and drag your SWC file to a folder in the repository.
   - Make the SWC file available on a shared network location.

After the SWC file is available, you must add the custom library in Guide Builder. (See *LiveCycle Designer ES Help*.)

Updating older versions of project resource files

If you want to replace a resource file (an image, form fragment, or SWC file) referenced by a form design or form guide with an older version of the same resource file, you must force a change to the timestamp of the older resource file. The modification is required to force the cached version of the resource file to update. Failing to update the timestamp of the resource file causes the currently cached version of the file to be used at run time.

Renaming project components

If you rename MXML components or ActionScript classes, you must manually include the renamed files in your Flex library.

➤ **To include renamed MXML components or ActionScript classes in a Flex library project:**

1. Ensure that your Flex project is open.

2. Go to the MXML component or ActionScript class that you renamed. Right-click the file and select **Include Class in Library**.

3. If you did not configure Flex Builder to build automatically, you must manually rebuild your Flex library project.

What’s next?

After you import the sample form guide projects, you can start creating your own form guide layouts. (See Creating Form Guide Layouts.)

3 Debugging Custom Form Guide Components

While developing custom form guide components, you can take advantage of the advanced debugging capabilities of Flex Builder to help isolate and eliminate issues with your projects. Each form guide you create in Guide Builder is essentially a Flex application. This means that the MXML definition of a form guide can be executed as a stand-alone Flex application in Flex Builder provided that the Flex project includes the necessary form guide libraries and supporting files.

Tip: While useful for debugging purposes, the method of running a form guide using Flex Builder, instead of using Guide Builder, is also an efficient development process to use while creating custom form guide components in general. This process reduces the amount of repetition when rebuilding your Flex library project and then updating the reference to the compiled SWC file each time you make a change.

To debug your custom form guide components using Flex Builder, you must perform the following general tasks:

- Create a Flex application project. This project acts as a container for the MXML source of your form guide. You run this project to view form guide output in Flex Builder. (See Creating a Flex application for debugging form guides.)
- Generate the form guide MXML definition using Guide Builder. Guide Builder creates the MXML definition of a form guide each time you preview the form guide output. This MXML definition is the source that you copy into the container Flex application project. (See Generating a form guide MXML definition using Guide Builder.)

Note: You cannot include the PDF version of the form design with your form guide output when you run the Flex application using Flex Builder. To view form guide output that includes the PDF version of your form, you must preview the form guide using Guide Builder.

Creating a Flex application for debugging form guides

The first task when you want to debug your custom form guide components using Flex Builder is to create a generic Flex application project. When compiled, this project displays the form guide as an MXML application.

➤ **To create a Flex application for debugging form guides:**

1. Start Flex Builder.
2. Click **File** > **New** > **Flex Project**.
3. Assign the project a unique name, set a project location folder, set the application type to **Web application**, and then click **Next**.
4. Set the output folder name, and then click **Next**.
5. Click **Library Path**.
6. Click **Add SWC**.

7. Browse to the \plugins\GuideBuilder folder where Adobe Designer ES is installed. By default, the path is C:\Program Files\Adobe\LiveCycle Designer ES\8.2\plugins\GuideBuilder. Add the following SWC files to the project:
   - GAClientRuntime.swc
   - XFAModel.swc
8. You must add the Flex Library Project that contains your custom form guide components so that references to your custom objects can be successfully resolved by the compiler.

   Click **Add Project**, select your Flex Library Project, and then click **OK**.
9. Click **Finish**.
10. Right-click the new Flex project and select **Properties**.
11. Click **Flex Compiler**, ensure that the **Enable strict type checking** is not selected, and then click **OK**.

After you create your Flex application, you must replace the MXML for the application with the MXML definition of your form guide, which you can obtain from Guide Builder.

Generating a form guide MXML definition using Guide Builder

When you want to debug your custom form guide components using Flex Builder, after you create a Flex application project, you must generate the form guide MXML definition using Guide Builder.

➤ **To create the MXML source for your form guide:**

1. Start Designer ES.
2. Start Guide Builder by clicking **Tools** > **Create or Edit Form Guide**.
3. Ensure that your custom SWC file is referenced in the **Add custom library** drop-down list, and then click **Preview**.
4. Ensure that the **Quick preview** option is not selected, and then click **Preview**.
5. After the form guide preview displays in your web browser, copy the MXML source from the Guide Content region of the Preview tab.
6. Paste the MXML source into the application file for the Flex application you already created. Ensure that you overwrite any existing MXML source that exists in the Flex project application file.
7. Save your Flex project and then click **Run**.

To view the latest version of the form guide during debugging, you must repeat this procedure each time the form guide hierarchy changes, or if there are changes to the original form design that affect objects or the values of objects on the form guide.

4 Creating Form Guide Layouts

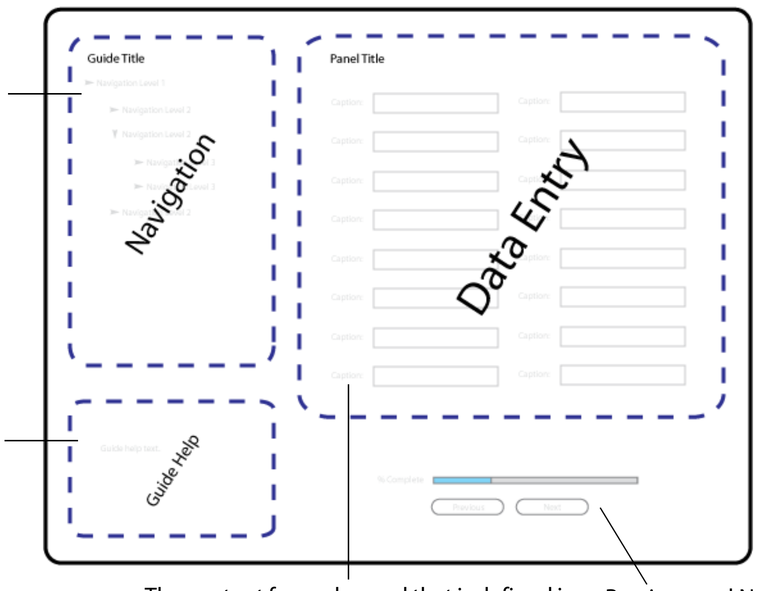
A *form guide layout* defines the visual layout and structure of a form guide that remains constant throughout a form-filling session. Guide Builder includes default form guide layouts designed to help you quickly create form guides that are structured in visually appealing and meaningful ways. However, using Flex Builder, you can create new form guide layouts to structure the data capture experience of your end users to meet your specific needs.

Overview of form guide layouts

In general, a form guide layout consists of a number of components that divide the rendered form guide into distinct areas:

- Form guide help
- Panel content
- Navigators
- Navigation controls
- PDF toolbar

The following image illustrates one example of a form guide layout structure.



Navigators display the form guide navigation control.

Form guide help text entered by a form author in Guide Builder.

The content for each panel that is defined in the Guide Builder form guide hierarchy. Each data entry panel defines its own individual layout.

Previous and Next buttons for navigating the form guide, and a progress bar to indicate the percentage of mandatory fields into which the user entered data.

Getting started creating form guide layouts

This section provides the following information to get you started creating form guide layouts:

- Creating a simple form guide layout

  Walks through creating a basic form guide layout using MXML.
- Creating an icon for a custom form guide layout

  Learn how to add custom icon images to distinguish your custom form guide layouts in Guide Builder.
- Referencing your Flex library project in Guide Builder

  Add your Flex library project to your form guide in Guide Builder to apply your custom panel layout.

Creating a simple form guide layout

Creating a simple form guide layout will familiarize you with the basic concepts, including the structure and MXML definition of a form guide layout.

➤ **To create a simple form guide layout:**

1. Start Flex Builder.
2. Create a new Flex library project and configure it using the procedures in Creating Flex Library Projects for Custom Form Guides. Ensure that you create the required folder structure in your Flex library project. For custom form guide layouts to display in the list of form guide layouts in Guide Builder, you must create all new form guide layouts in the `wrappers` folder in your Flex library project.
3. Right-click the `wrappers` folder and select **New** > **MXML Component**.
4. Type a unique file name. By default, Guide Builder adds a space immediately before each capital letter in the name of your component. For example, an MXML component named TabNav.mxml appears as Tab Nav in Guide Builder.
5. In the **Based On** list, select **Wrapper**.
6. (Optional) Set values for **Width** and **Height**.
7. Click **Finish**.

The MXML source for your new form guide layout should look like the following:

```
<?xml version="1.0" encoding="utf-8"?>
<Wrapper xmlns="ga.controls.*"
  xmlns:mx="http://www.adobe.com/2006/mxml" >

</Wrapper>
```

Note that the `<Wrapper>` element includes the namespace attribute `xmlns="ga.controls.*"`. It is considered good practice to provide namespaces when you reference objects from the form guide API, both to reduce the amount of MXML code, and to increase code readability. Updating the MXML source for the blank panel layout, the panel layout source should look like the following:

```
<?xml version="1.0" encoding="utf-8"?>
<gc:Wrapper xmlns:mx="http://www.adobe.com/2006/mxml"
  xmlns:gc="ga.controls.*" >

</gc:Wrapper>
```

Adding content to your form guide layout

After you create the shell of the new guide layout, you add Flex Builder components using the Flex Builder Source view based on the behavior you are trying to achieve. In addition, you can include other form guide and Flex components to suit your specific needs.

In this example, the blank panel layout created in the topic Creating a simple form guide layout is extended to include an area for displaying panel content, as well as some navigation buttons to move between panels and submit the form guide data.

Add the following to the blank panel layout:

- `<mx:VBox width="100%" height="100%">`

  A standard Flex component for organizing objects into a vertical list.
- `<gc:PanelContent width="100%" height="100%" />`

  The region of the form guide layout for displaying the data entry panel and the associated layout.
- `<mx:HBox>`

  A standard Flex component for organizing objects into a horizontal list.
- `<gc:PreviousPanelButton label="Back" />`

  A button object that goes to the previous panel in the form guide. This object is inactive if the current panel is the first panel in the form guide. The `label` attribute controls the button caption text.
- `<gc:NextPanelButton label="Forward" />`

  A button object that goes to the next panel in the form guide. This object is inactive if the current panel is the last panel in the form guide. The `label` attribute controls the button caption text.
- `<gc:SubmitButton label="Submit Data" />`

  A button object that submits the form guide data. This object is inactive if the current panel is the first panel in the form guide. The `label` attribute controls the button caption text.

  Note: Specifying a value for the `label` attribute overwrites the default SubmitButton labels that are added based on the submission option selected in Guide Builder.

  The SubmitButton component behaves differently depending on the submission options the form author sets for the form guide in Guide Builder. The following table outlines the behavior of the SubmitButton component for each submission option:

Guide Builder submit option	SubmitButton behavior	Default SubmitButton label
PDF	Clicking the submit button opens the PDF form from which the form filler submits the data by clicking the appropriate submit button. If the PDF rendition of the form is not included with the form guide, the submit button forces Forms ES to render the PDF to the user. The form filler verifies the form, and then clicks the submit button on the form to perform the data submission.	Submit from PDF
Guide	Users must submit the data by clicking the submit button on the form guide.	Submit
Printed Form	The submit button displays the label `Print form` and, when clicked, opens the PDF form that the form filler must manually print.	Print form
Hosted Application	The SubmitButton component does not appear on the form guide. The hosted application, such as Workspace ES, must extract the data from the form guide and perform the data submission.	N/A

Your MXML source should look like the following snippet.

```
<?xml version="1.0" encoding="utf-8"?>
<gc:Wrapper width="100%" height="100%"
  xmlns:mx="http://www.adobe.com/2006/mxml"
  xmlns:gc="ga.controls.*" >

  <mx:VBox width="100%" height="100%">
    <gc:PanelContent width="100%" height="100%" />
    <mx:HBox>
      <gc:PreviousPanelButton label="Back" />
      <gc:NextPanelButton label="Forward" />
      <gc:SubmitButton label="Submit Data" />
    </mx:HBox>
  </mx:VBox>
</gc:Wrapper>
```

At this point, your Flex library project should build with no warnings or errors. You can now either move on to Creating an icon for a custom form guide layout or Referencing your Flex library project in Guide Builder.

Creating an icon for a custom form guide layout

When you import your Flex library project into Guide Builder, by default, Guide Builder assigns generic icons to your custom panel layouts. To distinguish your panel layouts from each other, and from the default panel layouts included with Guide Builder, you can create custom icons and include them in your Flex library project.

➤ **To set up custom form guide layout icons:**

1. Create a new PNG image that is 90 pixels wide by 60 pixels tall. Name the new icon using the full folder structure and component naming you use for your custom panel layout. For example, a panel layout named MyPanel.mxml located in the custom > layouts folder of your Flex library project would have a corresponding icon image file named *custom.layouts.MyWrapper.png*.
2. Create a new folder named thumbnails at the root of your Flex library project. Add the new thumbnails image file to the thumbnails folder in the Flex library project.
3. Right-click your Flex library project and click **Properties**.
4. Click **Flex Library Build Path** > **Assets**. To add the thumbnails image to your compiled SWC file, select your thumbnails image file and then click **OK**.
5. Build your Flex library project.

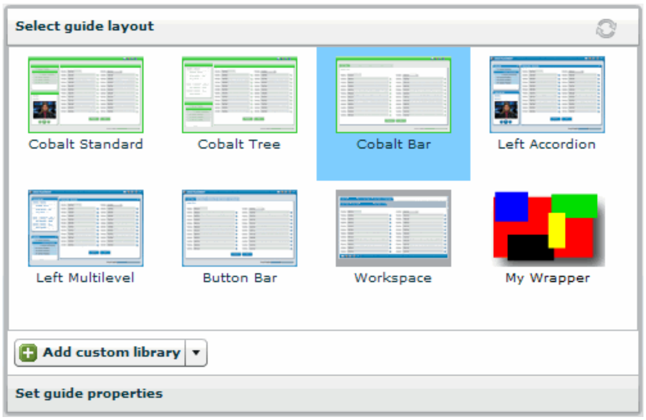
You should now reference the Flex library project for your new custom panel layout from Guide Builder.

Referencing your Flex library project in Guide Builder

After you build your Flex library project in Flex Builder, you must reference the compiled SWC file in Guide Builder to make your custom panel layout available.

➤ **To reference your Flex library project in Guide Builder:**

1. Start Designer ES.
2. Open the form design that contains the form guide into which you want to incorporate the custom form guide layout.
3. Click **Tools** > **Create or Edit Form Guide**.
4. Click **Add custom library**.
5. Go to the SWC file for your Flex library project, and then click **Open**.
6. Select the guide in the form guide hierarchy. Your custom form guide layout and icon image should appear in the **Select guide layout** menu.



Select your custom form guide layout to apply it to the panel.

7. Click **Preview** to render the form guide.

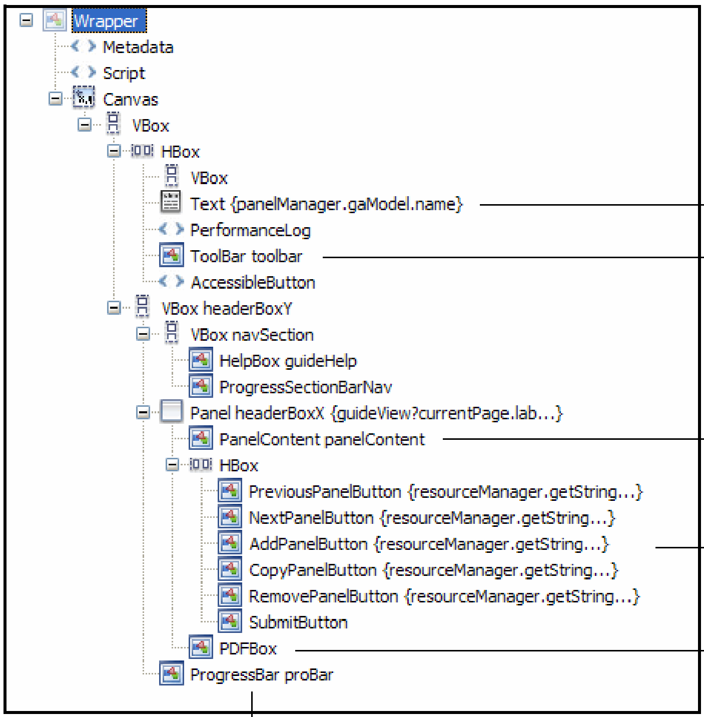
What's next?

Try creating one of your own custom form guide layouts, either by starting with a new blank layout, or by using the MXML source for one of the form guide layouts included with Guide Builder to get started. To learn more about custom form guide layouts by walking through a larger example, see the section Button Bar form guide layout.

Button Bar form guide layout

To better understand how to structure form guide layouts, this document will use the Button Bar form guide layout included with Guide Builder as an example. Specifically, this section looks at how the Button Bar form guide layout is used with the TLALife form guide that is included with LiveCycle ES. The TLALife example Flex project is located in the LiveCycle ES SDK\samples\FormGuides folder where you installed LiveCycle ES (by default, C:\Adobe\LiveCycle8).

The following image illustrates the Outline view of the Button Bar form guide layout for the TLALife example.



```
Wrapper
  Metadata
  Script
  Canvas
    VBox
      HBox
        VBox
        Text {panelManager.gaModel.name}
        PerformanceLog
        ToolBar toolbar
        AccessibleButton
      VBox headerBoxY
        VBox navSection
          HelpBox guideHelp
          ProgressSectionBarNav
        Panel headerBoxX {guideView?currentPage.lab...}
          PanelContent panelContent
          HBox
            PreviousPanelButton {resourceManager.getString...}
            NextPanelButton {resourceManager.getString...}
            AddPanelButton {resourceManager.getString...}
            CopyPanelButton {resourceManager.getString...}
            RemovePanelButton {resourceManager.getString...}
            SubmitButton
          PDFBox
        ProgressBar proBar
```

Standard Flex `Text` object that displays the form guide title.

The `Toolbar` component displays the PDF toolbar that allows end users to Save PDF, Print PDF, Email PDF, or Show/Hide PDF.

The `PanelContent` object defines the region that displays each panel of the form guide.

Navigation controls for moving through the set of panels, as well as adding and removing repeating panels. `SubmitButton` controls form guide data submission.

`PDFBox` defines the region that will display the PDF rendition of the form.

The `ProgressBar` component indicates the number of completed mandatory fields. If there are no mandatory fields, the progress bar is hidden.

In addition to the structure, the Button Bar form guide layout contains the following custom ActionScript variables that define the physical location of the regions of the form guide for easy referencing:

- `TOOLBAR_TAB:int` (the default value is 100)
- `HELP_CENTER_TAB:int` (the default value is 200)

- `BUTTONS_TAB:int` (the default value is 9000)

In addition, the Button Bar form guide layout contains the following custom ActionScript functions:

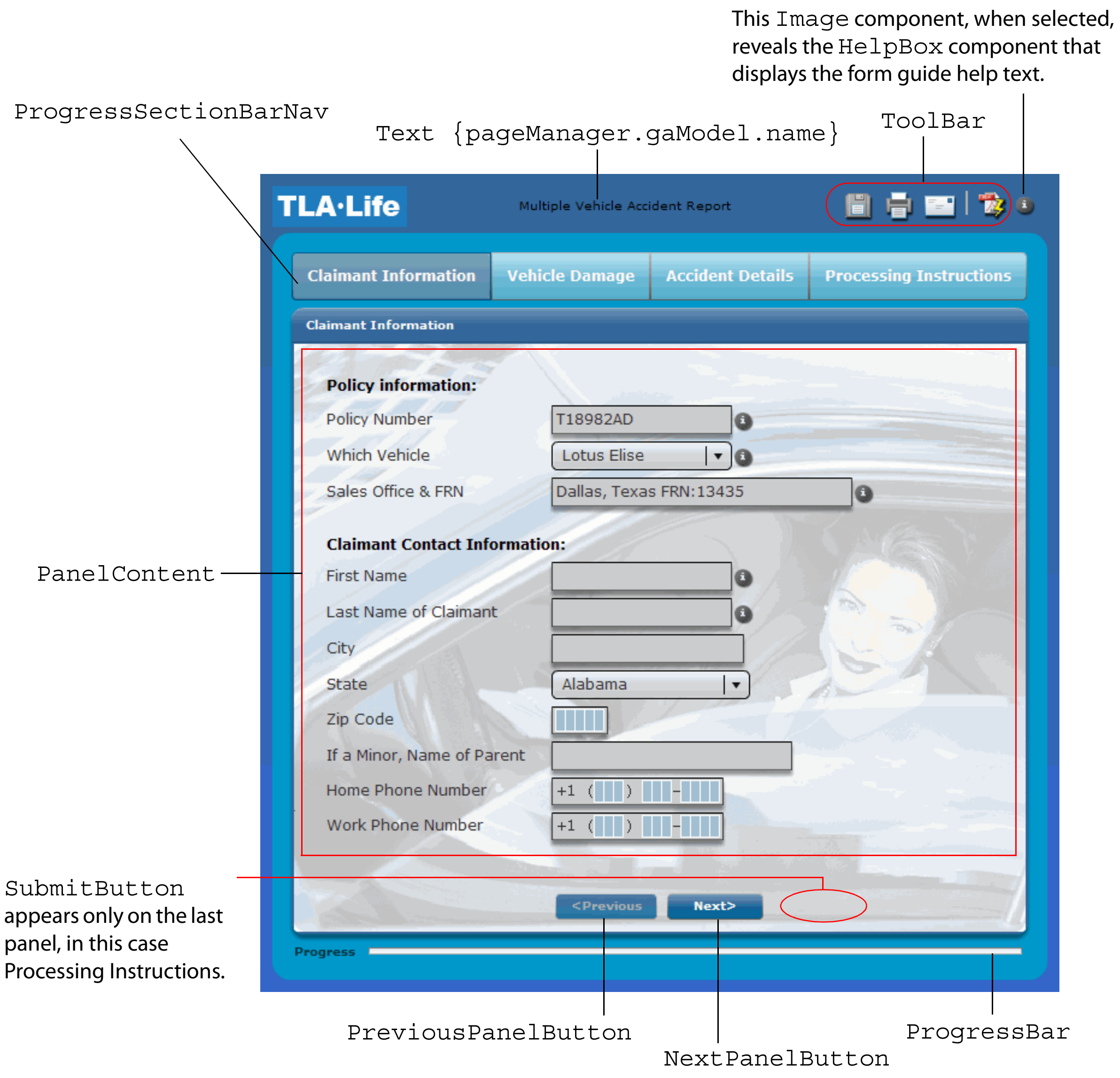
- `createChildren():void`

  Adds a new PAGE_SELECTION_CHANGE form guide event listener to the form guide layout.

- `pageChange(event:GAEvent):void`

  Dispatched when a user navigates to the next section of the form guide. The Button Bar form guide layout uses a navigation control that segments the form guide according to the sections you define in the form guide hierarchy. In the TLA LIfe example, each section contains only one panel, which is the recommended structure when using the Button Bar layout.

The following image illustrates the output of the TLALife sample form guide, which uses the Button Bar form guide layout.


This `Image` component, when selected, reveals the `HelpBox` component that displays the form guide help text.
ProgressSectionBarNav
Text {pageManager.gaModel.name}
ToolBar
TLA·Life
Multiple Vehicle Accident Report
Claimant Information
Vehicle Damage
Accident Details
Processing Instructions
Claimant Information
PanelContent
SubmitButton appears only on the last panel, in this case Processing Instructions.
<Previous
Next>
Progress
PreviousPanelButton
NextPanelButton
ProgressBar

Policy information:

Policy Number	T18982AD
Which Vehicle	Lotus Elise
Sales Office & FRN	Dallas, Texas FRN:13435

Claimant Contact Information:

First Name	
Last Name of Claimant	
City	
State	Alabama
Zip Code	
If a Minor, Name of Parent	
Home Phone Number	+1 ( ) -
Work Phone Number	+1 ( ) -

What's next?

Create your own custom form guide layouts, beginning with either the simple form guide layout created in the section Creating a simple form guide layout, or by copying and modifying one of the default Guide Builder panel layouts. For more information about the form guides ActionScript API, see *LiveCycle ES ActionScript Language Reference*.

To start learning about creating custom panel layouts, see Creating Panel Layouts, or for custom form guide controls, see Creating Controls.

5 Creating Panel Layouts

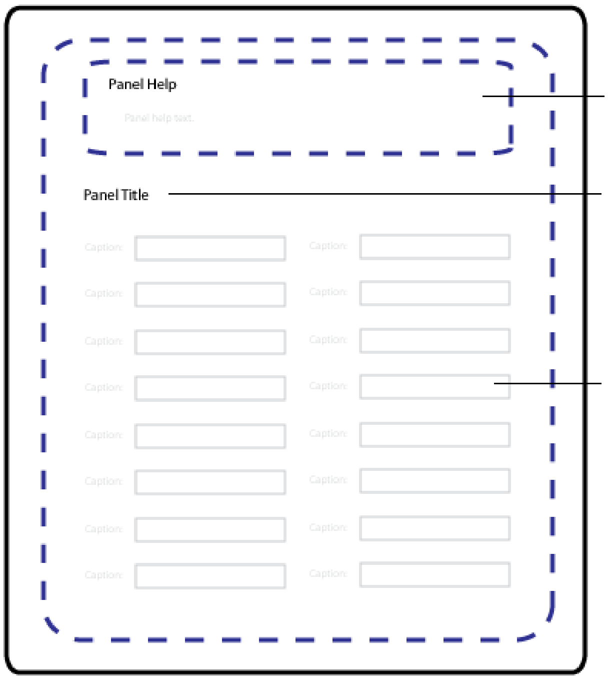
A *panel layout* defines the visual layout and presentation of objects on a panel in the form guide hierarchy. Guide Builder includes default panel layouts designed to help you quickly create form guides with panels that are structured in visually appealing and meaningful ways. Using Flex Builder, you can create new panel layouts to structure panel content in new ways or to include Flex objects that take advantage of the capabilities of Flex.

Overview of panel layouts

In general, a panel layout consists of components that act as placeholders for text, form objects, or questions specified in Guide Builder. The following are the most common form guide placeholder components that appear on panel layouts:

- **PanelItem:** A placeholder object that displays an item, or a repeating sequence of items, from the form guide hierarchy; either a form field, form text object, guide text object, or next area object.
- **PanelText:** Corresponds to the Guide Text utility object in Guide Builder. This component acts as a text object that appears only in a form guide, and not on the PDF version of the form. Each `PanelText` object is contained within a `PanelItem`.
- **PanelBreak:** Corresponds to the Next Area utility object available in Guide Builder. This component acts like a column break in panel layouts that contain multiple columns, causing the objects that follow it to flow to the next column when the form guide is rendered. Each `PanelBreak` object is contained within a `PanelItem`.
- **QuestionItem:** A placeholder object that displays question text to a form filler. Question text is specified in Guide Builder.
- **PanelTitle:** A placeholder object that displays the name of the current panel. The panel title is specified in Guide Builder.
- **HelpPanel:** Displays panel help text to a form filler. The panel help text is specified in Guide Builder.

The following image illustrates one example of how a panel layout structure reflects the common placeholder components.



Panel help text (`HelpPanel`) that a form author enters using Guide Builder. Depending on the form guide layout, the panel help may appear as part of the panel, or with the form guide help text in the Help Center area of the form.

Panel title text (`PanelTitle`) for the panel that a form author enters using Guide Builder.

Panel content that consists of form guide objects (`PanelItem`) as well as Flex Builder components.

Although not represented visually, questions (`QuestionItem`) associated with a particular panel can occur anywhere on a panel layout. The default panel layouts included with Guide Builder display questions immediately after the user navigates away from the current panel.

The MXML source code for the panel layouts included with Guide Builder are available in the \LiveCycle_ES_SDK\samples folder where LiveCycle ES is installed. By default, the folder is C:\Adobe\LiveCycle8.2\LiveCycle_ES_SDK\samples\FormGuides\GuideSource\GuideLayouts\PanelLayouts\samples\layouts.

For example, examine the MXML source for the One Column panel layout:

```
<?xml version="1.0" encoding="utf-8"?>
<ga:LayoutTemplate width="100%" height="100%"
   xmlns:mx="http://www.adobe.com/2006/mxml"
   xmlns:ga="ga.model.*"
   xmlns:gc="ga.controls.*">

   <mx:Metadata>
      [IconFile("assets/GuideComponentDisabled.png")]
   </mx:Metadata>

   <mx:Script>
      <![CDATA[
         import mx.core.UIComponentDescriptor;
         import ga.controls.Wrapper;

         override public function get documentDescriptor(
):UIComponentDescriptor { return Object(this)._documentDescriptor_; }
         override public function set documentDescriptor(
oDescriptor:UIComponentDescriptor ):void { Object(this)._documentDescriptor_
= oDescriptor; }
      ]]>
   </mx:Script>

   <mx:VBox width="100%" height="100%" styleName="layoutobjects">
```

```
     <gc:HelpPanel id="helpPanel" styleName="panelhelp" />
     <ga:PanelItem  itemInstancesPerCycle="-1" repeatItemLimit="-1"
width="100%"/>
     <ga:QuestionItem itemInstancesPerCycle="-1" repeatItemLimit="-1"
width="100%"/>
  </mx:VBox>

</ga:LayoutTemplate>
```

Later in this chapter, you walk through creating MXML source in more detail. At this point, however, it is important to notice that the MXML code is distributed into the following sections:

- Initial namespace definitions. Namespaces are a convenient way to define shortcuts to ActionScript packages to simplify your code. Although not required, they are recommended.
- `<mx:Metadata>`

  This MXML block defines an icon image to display in the Components view of Flex Builder.
- `<mx:Script>`

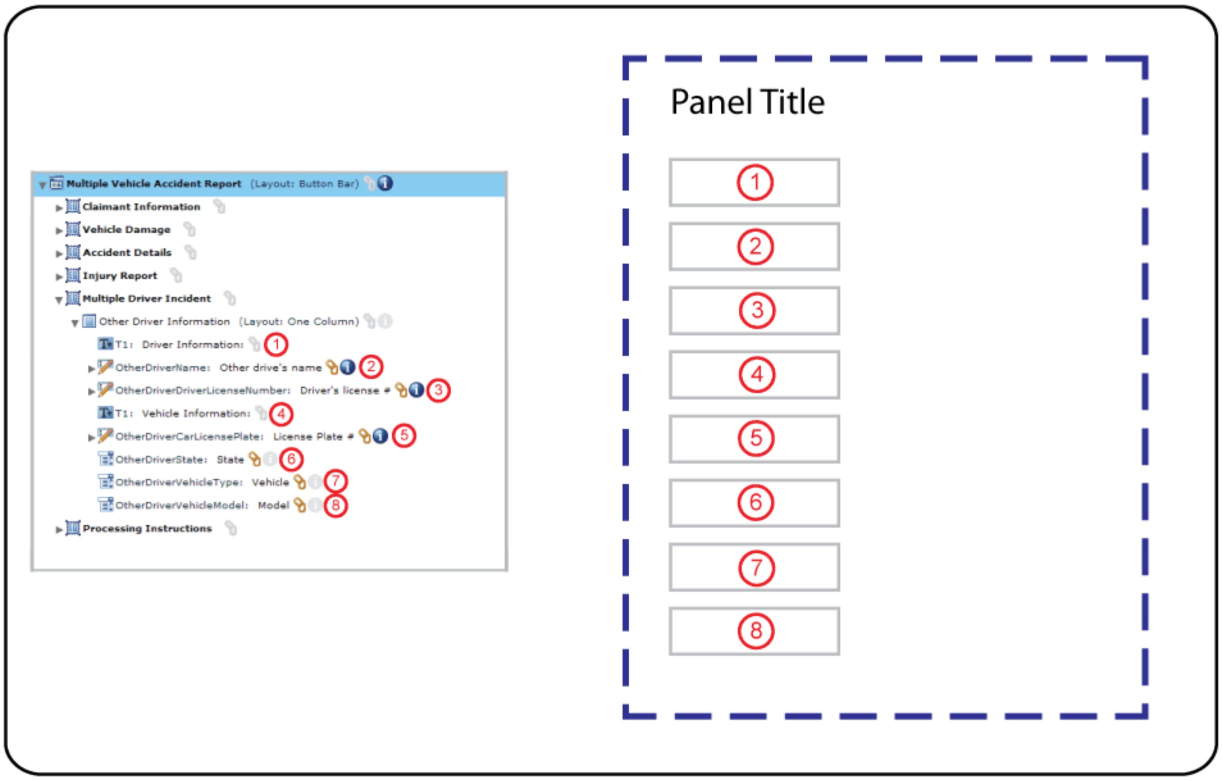
  The One Column panel layout is an MXML component, and it is created declaratively in MXML. Panel layouts created declaratively require this `<mx:Script>` block to display form design objects.
- `<gc:HelpPanel>`, `<ga:PanelItem>`, and `<ga:QuestionItem>`

  Placeholder components for the panel help, form design and form guide objects, and form guide questions respectively. Each of these components is discussed in more detail later in this chapter, but you must understand `PanelItem` to create custom panel layouts.

Understanding the PanelItem class

Each panel on the form guide hierarchy in Guide Builder, where you add form design objects and form guide utility objects, can be thought of as a type of playlist. The objects on a panel are ordered in a sequence that you determine, and each object occupies exactly one space in the sequence. The `PanelItem` class behaves like a column in a table, where each object from the playlist occupies one slot, or cell in the table. You can control how many cells you want the column to display. By having more than one instance of `PanelItem`, you can create a multi-column experience with objects from the playlist distributed across columns.

The following image illustrates how form guide hierarchy items map to `PanelItem` slots in a One Column panel layout configuration.


Multiple Vehicle Accident Report (Layout: Button Bar)
Claimant Information
Vehicle Damage
Accident Details
Injury Report
Multiple Driver Incident
Other Driver Information (Layout: One Column)
T1: Driver Information:
1
OtherDriverName: Other drive's name
2
OtherDriverDriverLicenseNumber: Driver's license #
3
T1: Vehicle Information:
4
OtherDriverCarLicensePlate: License Plate #
5
OtherDriverState: State
6
OtherDriverVehicleType: Vehicle
7
OtherDriverVehicleModel: Model
8
Processing Instructions
Panel Title
1
2
3
4
5
6
7
8

You will see an MXML example of using the `PanelItem` class in the section Extending the blank panel layout, but understanding the concept of the `PanelItem` class is central to understanding panel layouts, and form guides as a whole.

For examples of using the `PanelItem` class, view the MXML source for the panel layouts included with Guide Builder, which are available in the \LiveCycle_ES_SDK\samples\FormGuides\GuideSource\GuideLayouts\PanelLayouts\samples\layouts folder where LiveCycle ES is installed.

What's next?

In the next section, we create a simple, blank panel layout to get familiar with the process of creating panel layouts as new MXML components.

Getting started creating panel layouts

Simple panel layouts do not require a lot of MXML and ActionScript, and are a good introduction to the basic principles of working with the form guide ActionScript packages and classes.

This section provides the following information to get you started creating panel layouts:

- Creating a blank panel layout

  Walks through creating a blank panel layout using MXML.

- Extending the blank panel layout

  Walks through creating a One Column panel layout beginning with the blank panel layout example.
- Creating an icon for a custom panel layout

  Learn how to add custom thumbnail images to distinguish your custom form guide layouts and panel layouts in Guide Builder.
- Referencing your Flex library project in Guide Builder

  Add your Flex library project to your form guide in Guide Builder to apply your custom panel layout.

Creating a blank panel layout

Creating a simple panel layout will familiarize you with the basic concepts, including the structure and MXML definition of a panel layout.

➤ To create a blank panel layout:

1. Start Flex Builder.
2. Create a new Flex library project and configure it using the procedures in Creating Flex Library Projects for Custom Form Guides. Ensure that you create the required folder structure in your Flex Library Project. For custom panel layouts to display in the list of panel layouts in Guide Builder, you must create all new panel layouts in the `layouts` folder of your Flex library project.
3. Right-click the `layouts` folder and select **New** > **MXML Component**.
4. Type a unique file name. When displaying the name of your custom panel layout, Guide Builder adds a space immediately before each upper case character and number in the name of your component for readability. For example, an MXML component named MyPanel.mxml will appear as My Panel in Guide Builder.
5. In the **Based On** list, select **LayoutTemplate**.
6. (Optional) Set the values for **Width** and **Height** to `100%`.
7. Click **Finish**.

The MXML source for your new panel layout should look like the following:

```
<?xml version="1.0" encoding="utf-8"?>
<LayoutTemplate width="100%" height="100%"
   xmlns:mx="http://www.adobe.com/2006/mxml"
   xmlns="ga.model.*" >

</LayoutTemplate>
```

Note that the `<LayoutTemplate>` element includes the namespace attribute `xmlns="ga.model.*"`. It is considered good practice to provide namespaces when you reference objects from the form guide API, both to reduce the amount of MXML code, and to increase code readability. Updating the MXML source for the blank panel layout, the panel layout source should look like the following:

```
<?xml version="1.0" encoding="utf-8"?>
<ga:LayoutTemplate width="100%" height="100%"
  xmlns:mx="http://www.adobe.com/2006/mxml"
  xmlns:ga="ga.model.*" >

</ga:LayoutTemplate>
```

At this point, your panel layout project should compile with no warnings or errors. However, this panel layout in its current state does not display any objects or content. In the next section you extend the blank panel layout to display form design objects and Flex components.

Extending the blank panel layout

After you create the shell of the new panel layout, you add Flex Builder components using the Flex Builder Source view based on the behavior you are trying to achieve. In addition, you can include other form guide and Flex components to suit your specific needs. However, panel layouts are not required to contain any specific form guide components. For example, a panel may contain hardcoded text objects, such as legal text or instructions, that must be read prior to filling the form guide.

In this example, the blank panel layout created in the section Creating a blank panel layout is extended to mimic the functionality of the One Column panel layout that is included with Guide Builder.

Add the following to the blank panel layout:

- `xmlns:gc="ga.controls.*"`

  Defines the namespace for the `HelpPanel` class.

- `<mx:VBox width="100%" height="100%" styleName="layoutobjects" />`

  A standard Flex component for organizing objects into a vertical list.

- `<gc:HelpPanel id="helpPanel" styleName="panelhelp" />`

  A region of the panel layout for displaying panel help text. The `id` attribute must be set to `helpPanel`, and the `styleName` attribute must be set to the class name for the corresponding panel help CSS style.

- `<ga:PanelItem itemInstancesPerCycle="-1" repeatItemLimit="-1" width="100%" />`

  By default, an instance of `PanelItem` requires two attributes: `itemInstancesPerCycle` and `repeatItemLimit`.

  The `itemInstancesPerCycle` attribute indicates the number of form guide hierarchy items that can fill a specified column. Therefore, setting `itemInstancesPerCycle` to the default value of `1` indicates that only one form guide hierarchy item appears in the output. The layout then moves on to the next instance of `PanelItem` to continue laying out form guide hierarchy items. Setting `itemInstancesPerCycle` to a value of `-1` indicates that the layout should continue adding instances of the current `PanelItem`, or column, until the `repeatItemLimit` value is met.

  The `repeatItemLimit` attribute indicates the maximum number of form guide hierarchy items to add to the instance of `PanelItem`. In general, if you want all the items you add to the form guide hierarchy for a particular panel to appear in the output, this attribute should be set to a value of `-1`.

- `<ga:QuestionItem />`

  Not visible in the physical layout, this component displays any questions specific to a panel using this layout that a form author enters in Guide Builder.

In addition, because this example defines `PanelItem` declaratively; that is, it defines a `PanelItem` instance in MXML, the following `<mx:Script>` ActionScript is required for form guide hierarchy objects to correctly display.

```
   <mx:Script>
      <![CDATA[
         import mx.core.UIComponentDescriptor;
         import ga.controls.Wrapper;

         override public function get documentDescriptor(
):UIComponentDescriptor { return Object(this)._documentDescriptor_; }
         override public function set documentDescriptor(
oDescriptor:UIComponentDescriptor ):void { Object(this)._documentDescriptor_
= oDescriptor; }
      ]]>
   </mx:Script>
```

Your MXML source should look like the following snippet.

Example: *Extended blank panel layout*

```
<?xml version="1.0" encoding="utf-8"?>
<ga:LayoutTemplate xmlns:mx="http://www.adobe.com/2006/mxml"
   xmlns:ga="ga.model.*"
   xmlns:gc="ga.controls.*" >

   <mx:Script>
      <![CDATA[
         import mx.core.UIComponentDescriptor;
         import ga.controls.Wrapper;

         override public function get documentDescriptor(
):UIComponentDescriptor { return Object(this)._documentDescriptor_; }
         override public function set documentDescriptor(
oDescriptor:UIComponentDescriptor ):void { Object(this)._documentDescriptor_
= oDescriptor; }
      ]]>
   </mx:Script>

   <mx:VBox width="100%" height="100%" styleName="layoutobjects">
      <gc:HelpPanel id="helpPanel" styleName="panelhelp" />
      <ga:PanelItem itemInstancesPerCycle="-1" repeatItemLimit="-1"
width="100%" />
      <ga:QuestionItem />
   </mx:VBox>

</ga:LayoutTemplate>
```

At this point, your Flex library project should build with no warnings or errors. You can now either move on to Creating an icon for a custom panel layout or Referencing your Flex library project in Guide Builder.

Creating an icon for a custom panel layout

When you import your Flex library project into Guide Builder, by default, Guide Builder assigns generic icons to your custom panel layouts. To distinguish your panel layouts from each other, and from the default panel layouts included with Guide Builder, you can create custom icons and include them in your Flex library project.

➤ **To set up custom panel layout icons:**

1. Create a new PNG image that is 90 pixels wide by 60 pixels tall. Name the new icon using the full folder structure and component naming you use for your custom panel layout. For example, a panel layout named MyPanel.mxml located in the custom > layouts folder of your Flex library project would have a corresponding icon image file named as follows:

   custom.layouts.MyPanel.png

2. Create a new folder named `thumbnails` at the root of your Flex library project. Add the new icon file to the `thumbnails` folder in the Flex library project.
3. Right-click your Flex library project and click **Properties**.
4. Click **Flex Library Build Path** > **Assets**. To add the icon to your compiled SWC, select your icon file and then click **OK**.
5. Build your Flex library project.

You should now reference your new custom panel layout Flex library project from Guide Builder.

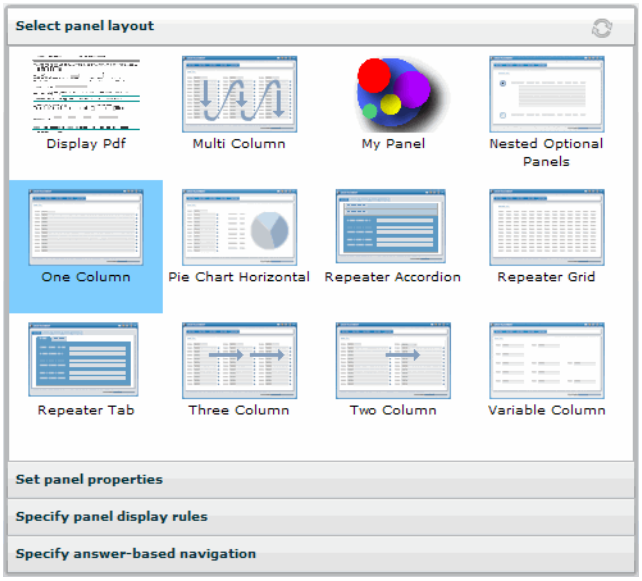
Referencing your Flex library project in Guide Builder

After you build your Flex library project in Flex Builder, you must reference the compiled SWC file in Guide Builder to make your custom panel layout available

➤ **To reference your Flex library project in Guide Builder:**

1. Start LiveCycle Designer ES.
2. Open the form design that contains the form guide into which you want to incorporate the custom panel layout.
3. Click **Tools** > **Create or Edit Form Guide**.
4. Click **Add custom library**.
5. Go to the SWC file for your Flex library project, and then click **Open**.

6. Select a panel in the form guide hierarchy. Your custom panel layout and icon should appear in the **Select panel layout** menu.



Select your custom panel layout to apply it to the panel.

7. Click **Preview** to render the form guide.

What's next?

Try creating one of your own custom panel layouts, either by starting with a new blank layout, or by using the MXML source for one of the panel layouts included with Guide Builder to get started. To learn more about custom panel layouts by walking through the TLA Life form guide, see the section TLA Life panel layout example.

TLA Life panel layout example

To view a more complex example of a custom panel layout, you can use the TLALife project that is included with Designer ES and LiveCycle ES. The TLALife collateral is divided into two separate locations:

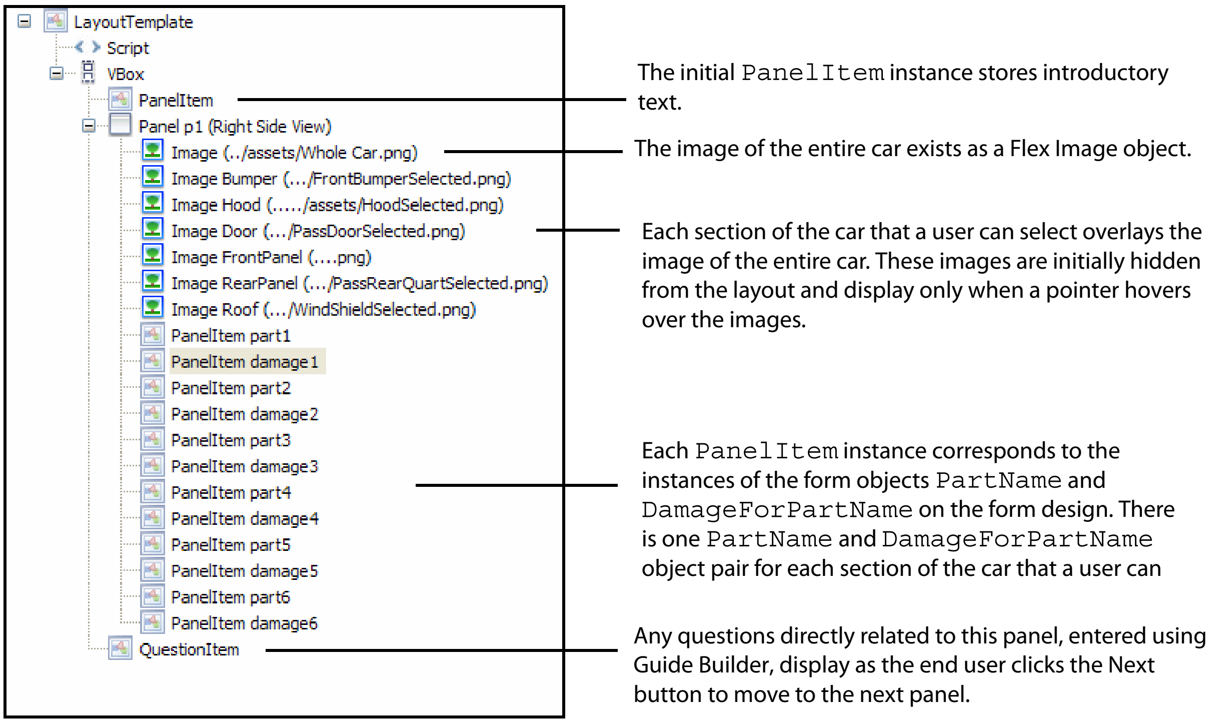
- The TLALife form design that contains the form guide definition, which can be viewed in Guide Builder. The form design is located in the \EN\Samples\Forms\FormGuide\TLALife folder where Designer ES is installed. By default, the folder is C:\Program Files\Adobe\LiveCycle Designer ES\8.2\EN\Samples\Forms\FormGuide\TLALife).
- The TLALife Flex project that contains the custom panel layout MXML file and associated assets. The Flex project is located in the \LiveCycle_ES_SDK\samples\FormGuides\GuideSource\TLALife folder where LiveCycle ES is installed. By default, the folder is C:\Adobe\LiveCycle8.2\LiveCycle_ES_SDK\samples\FormGuides\GuideSource\TLALife.

You can open the TLALife form design in Designer ES to preview the form guide definition in Guide Builder. You can import the TLALife Flex library project into Flex Builder to view the MXML source for the vehicle damage panel layout.

➤ To import the TLALife Flex library project into Flex Builder:

1. Start Flex Builder.
2. Click **File** > **Import** > **Flex Project**.
3. Select **Project folder**, and then click **Browse**.
4. Go to the \LiveCycle_ES_SDK\samples\FormGuides\GuideSource\TLALife folder where LiveCycle ES is installed. By default, the folder is C:\Adobe\LiveCycle8.2\LiveCycle_ES_SDK\samples\FormGuides\GuideSource\TLALife.
5. Click **Finish**.

When examining the custom panel layout for vehicle damage (TLALifeCarDamage.mxml), the following image of the Outline perspective in Flex Builder illustrates the general structure.



```
LayoutTemplate
    Script
    VBox
        PanelItem
        Panel p1 (Right Side View)
            Image (../assets/Whole Car.png)
            Image Bumper (.../FrontBumperSelected.png)
            Image Hood (...../assets/HoodSelected.png)
            Image Door (.../PassDoorSelected.png)
            Image FrontPanel (....png)
            Image RearPanel (.../PassRearQuartSelected.png)
            Image Roof (.../WindShieldSelected.png)
            PanelItem part1
            PanelItem damage1
            PanelItem part2
            PanelItem damage2
            PanelItem part3
            PanelItem damage3
            PanelItem part4
            PanelItem damage4
            PanelItem part5
            PanelItem damage5
            PanelItem part6
            PanelItem damage6
        QuestionItem
```

The initial `PanelItem` instance stores introductory text.

The image of the entire car exists as a Flex Image object.

Each section of the car that a user can select overlays the image of the entire car. These images are initially hidden from the layout and display only when a pointer hovers over the images.

Each `PanelItem` instance corresponds to the instances of the form objects `PartName` and `DamageForPartName` on the form design. There is one `PartName` and `DamageForPartName` object pair for each section of the car that a user can

Any questions directly related to this panel, entered using Guide Builder, display as the end user clicks the Next button to move to the next panel.

In addition to the structure, the TLALife panel layout contains the following custom ActionScript functions:

- `damageClick(event:Event, sPart: String):void`

  When a user selects a section of the car, the alpha level of the overlay image changes to provide a visual cue, and the damage selection drop-down list displays. The user can deselect the section of the car by selecting it a second time.

- `findUiPair(sPart:String):Array`

  This function creates a new array object consisting of a part `PanelItem` object paired with a corresponding damage `PanelItem` object.

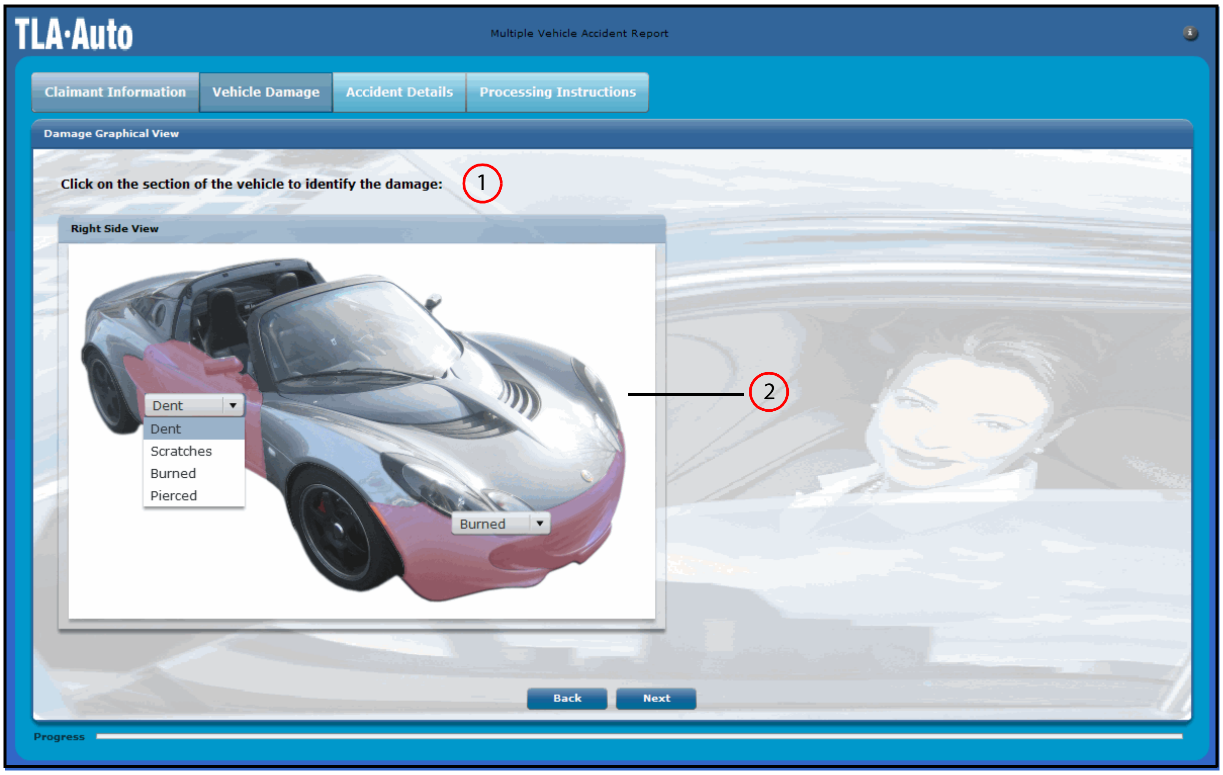
- `refresh(oEvent:Event):void`

  This function is not used by the TLALife example.

- `setAlphaRollOver(sPart:String, bOver:Boolean):void`

  This function controls the alpha level of the overlay images for each section of the car that users can select. When the pointer hovers over a section of the car, this method changes the alpha value to display a visual cue.

The following image illustrates the output of the Vehicle Damage panel in the TLALife sample form guide and calls out the various form guide components within the panel.



1 - Initial instance of `PanelItem`.

2 - The additional instances of `PanelItem` used in conjunction with the image objects to create the experience of selecting a section of the car and then selecting the type of damage from a menu.

What's next?

Create your own custom panel layouts, beginning with either the blank panel layout example, or by copying and modifying one of the default Guide Builder panel layouts. For more information about the form guides ActionScript API, see *LiveCycle ES ActionScript Language Reference*.

To start learning about creating custom form guide layouts, see Creating Form Guide Layouts, or for custom form guide controls, see Creating Controls.

6 Creating Controls

Controls are form guide components that perform specific actions. For example, the Next and Previous buttons that display on a form guide are navigation controls that let users move forwards and backwards through a form guide. In general, a control performs only a single action. However, that action can be as simple or as complex as you require for your specific solution.

Overview of form guide controls

The different types of controls can be grouped according to their intended purpose:

Navigation controls: Guide Builder, in conjunction with LiveCycle Designer ES, provides a tremendous amount of functionality. However, in some situations you may want to create specific functionality based on your solution requirements. For example, you may want to create navigation controls that allow your users to quickly move to the first or last panel in a large form guide.

Field controls: By default, form guides map supported form design objects to their logical equivalents. For example, a Numeric Field maps to a Flex TextInput control. In some situations, using a different Flex object may make for a user experience that is more engaging. For example, if the original Numeric Field object is used to store a percentage value, mapping the Numeric Field to an HSlider control creates a more intuitive data entry experience for a user.

To take full advantage of custom controls in a form guide, you should be familiar with the form guide ActionScript packages. (See *LiveCycle ES ActionScript Reference*.)

Creating a new form guide control

You can create new form guide controls in either MXML or ActionScript.

➤ **To create a custom form guide control in MXML:**

1. Start Flex Builder.
2. Select **File** > **New** > **MXML Component**.
3. Type a unique file name.
4. In the **Based On** list, select a component that is most similar in behavior to the component you want to create. This lets you take advantage of existing behaviors through inheritance.
5. (Optional) Set values for **Width** and **Height**.
6. Click **Finish**.

➤ **To create a custom form guide control in ActionScript:**

1. Start Flex Builder.
2. Select **File** > **New** > **ActionScript Class**.

3. Type a unique class name.
4. In the **Superclass** field, specify the ActionScript class that is most similar in behavior to the component you want to create. This lets you take advantage of existing behaviors through inheritance.
5. Click **Finish**.

After you create the shell of the new control, you add the desired functionality using the Source view in Flex Builder. For more information about the form guide ActionScript API that you can take advantage of while adding functionality, see *LiveCycle ES ActionScript Reference*.

What's next?

Learn to create custom controls by studying several examples:

- Example: FirstPanelButton
- Example: LastPanelButton
- Example: CustomHSlider

Creating navigation controls

Navigation controls are custom components that affect how a user navigates through the sections and panels of a form guide. The Next and Previous buttons, controlled by the `NextPanelButton` and `PreviousPanelButton` ActionScript classes respectively, are examples of navigation controls.

In this section, we explore how to create navigation controls to help users browse form guides that contain many sections and panels.

Example: FirstPanelButton

Description

This example creates a new button object that, when clicked by a user at run time, displays the first panel in the form guide. Adding this control to a form guide layout should create the following experience:

- When the form guide renders, a button labelled First should be dimmed. The user cannot interact with the button immediately.
- After the user navigates to another panel in the form guide, using the Next button or through the navigation supplied by the form guide layout, the First button should display normally and be enabled to the user. Clicking the button returns the user to the first panel in the form guide.

Source

```
package custom.controls
{
   import mx.controls.Button;
   import flash.events.Event;
   import flash.events.MouseEvent;
   import ga.model.GAEvent;
   import ga.model.PanelManager;

   public class FirstPanelButton extends Button
   {
        // The PanelManager class controls the organization and behavior
        // of panel instances at run time. The class contains
        // convenience properties and methods that are useful for
        // manipulating panels within a form guide.
        private var _pm:PanelManager;

        protected override function createChildren():void
        {
            super.createChildren();
            this.label = "First";
            _pm = PanelManager.instance;

            // Adds event listeners for the three events that can cause a
            // change in the state of this button, and reevaluates if the button
            // should display as enabled.
            _pm.addEventListener(GAEvent.PAGE_SELECTION_CHANGE, pageChange);
            _pm.addEventListener(GAEvent.PAGE_REMOVE, pageChange);
            _pm.addEventListener(GAEvent.PAGE_ADD, pageChange);

            // Sets the default behavior when the form guide renders. In this
            // case, on the initial panel, the button should be dimmed.
            super.enabled = false;
        }

        // When the button is clicked by a user, the current panel displayed
        // is set to be the first panel in the form guide.
        override protected function clickHandler(event:MouseEvent):void
        {
            if (super.enabled)
            {
                _pm.currentPage = _pm.firstPage;
            }
        }

        // Conditionally sets the button’s enabled property dependent on
        // whether the current panel is the first panel in the form guide.
        private function pageChange(event:Event):void
        {
            super.enabled = _pm.previousPage!=null;
        }
    }
}
```

For information about adding this control to a form guide layout, see Adding custom navigation controls to panel layouts and form guide layouts.

Tip: When creating custom navigation controls, you should be aware of the methods and properties of the `PanelManager` class. (See *LiveCycle ES ActionScript Reference*.)

Example: LastPanelButton

Description

This example creates a new button object that, when clicked by an user at run time, displays the last panel in the form guide. Adding this control to a form guide layout should create the following experience:

- When the form guide renders, a button labelled Last should display to the user. Clicking the button sends the user to the last panel in the form guide.
- If the user is on the last panel of the form guide, this button should appear dimmed and be disabled to the user.

Source

```
package custom.controls
{
  import mx.controls.Button;
  import flash.events.Event;
  import flash.events.MouseEvent;
  import ga.model.GAEvent;
  import ga.model.PanelManager;

  public class LastPanelButton extends Button
  {
    // The PanelManager class controls the organization and behavior
    // of panel instances at run time. The class contains
    // convenience properties and methods that are useful for
    // manipulating panels within a form guide.
    private var _pm:PanelManager;

    protected override function createChildren():void
    {
      super.createChildren();
      this.label = "Last";
      _pm = PanelManager.instance;

      // Adds event listeners for the three events that can cause a
      // change in the state of this button, and reevaluates whether
      // the button should display as enabled.
      _pm.addEventListener(GAEvent.PAGE_SELECTION_CHANGE, pageChange);
      _pm.addEventListener(GAEvent.PAGE_REMOVE, pageChange);
      _pm.addEventListener(GAEvent.PAGE_ADD, pageChange);

      // Sets the default behavior when the form guide renders. In this
      // case, on the initial panel, the button should be disabled.
      super.enabled = true;
    }

    // When the button is clicked by a user, the current panel displayed
    // is set to be the last panel in the form guide.
    override protected function clickHandler(event:MouseEvent):void
    {
      if (super.enabled)
      {
        _pm.currentPage = _pm.lastPage;
```

```
            }
        }

        // Conditionally sets the button's enabled property dependent on
        // whether the current panel is the last panel in the form guide.
        private function pageChange(event:Event):void
        {
            super.enabled = _pm.nextPage!=null;
        }
    }
}
```

For information about adding this control to a form guide layout, see Adding custom navigation controls to panel layouts and form guide layouts.

Adding custom navigation controls to panel layouts and form guide layouts

To implement controls that you want to display as part of a form guide layout or panel layout, you must add the control to the MXML definition of the form guide layout or panel layout.

As an example, the following MXML source illustrates how to modify the form guide layout created in the Creating a simple form guide layout section with the First and Last buttons created in the Example: FirstPanelButton and Example: LastPanelButton sections.

```
<?xml version="1.0" encoding="utf-8"?>
<gc:Wrapper width="100%" height="100%"
    xmlns:mx="http://www.adobe.com/2006/mxml"
    xmlns:gc="ga.controls.*"
    xmlns:cc="custom.controls.*" >

    <mx:VBox width="100%" height="100%">
        <gc:PanelContent width="100%" height="100%" />
        <mx:HBox>
            <cc:FirstPanelButton />
            <gc:PreviousPanelButton label="Back" />
            <gc:NextPanelButton label="Forward" />
            <cc:LastPanelButton />
            <gc:SubmitButton label="Submit Data" />
        </mx:HBox>
    </mx:VBox>
</gc:Wrapper>
```

In this example, a new namespace `custom.controls.*` is added to simplify the references to the new controls. In addition, the bold text represents the references to the custom navigation controls.

Creating custom field controls

Field controls are custom components that replace standard form guide objects to provide an enhanced user experience. Field controls exist only on the form guide, and do not replace the original form design object on the PDF version of the form.

In this section, we explore how to create field controls to provide a more intuitive data entry experience for users.

Example: CustomHSlider

Description

This example creates an HSlider control that you can use for mapping form design objects. This custom field control assumes that the original form design object is used to specify integer values between 0 and 100 that indicate a percentage. Mapping the form design object to this control creates the following experience:

- When the form guide renders, an HSlider control appears in place of the original form design object. The HSlider should have the same caption value as the original form design object, and the user can drag the slider to set a value between 0 and 100.
- When switching to the PDF view of the form, the value the user sets using the slider should appear in the corresponding field. If the user changes the value on the PDF and then returns to the form guide, the slider value should reflect the updated value.

Source

```
package custom.controls
{
   import mx.controls.HSlider;

   public class CustomHSlider extends HSlider
   {

      protected override function createChildren():void
      {
         super.createChildren();

         // Sets the minimum, maximum, and initial values for the range.
         this.minimum = 0;
         this.maximum = 100;
         this.value = 0;

         // The increment range for the slider, and the increments for the
         // increment label.
         this.snapInterval = 1;
         this.tickInterval = 10;

         // The label for the range represented by the slider.
         this.labels=['0%', '100%'];

         // Sets interaction properties allowing the slider to update in
         //real-time in response to user interaction.
         this.allowTrackClick = true;
         this.liveDragging = true;
      }
   }
}
```

For information about adding this control to a form guide layout, see Adding custom navigation controls to panel layouts and form guide layouts.

Mapping form design objects to custom field controls

To use your custom field control on a form guide, you must associate an object in the form guide hierarchy with your custom field control. You must perform this mapping for each form guide object you want to associate with the new custom field control. You do not need to add the custom field control to a form guide layout or panel layout.

➤ **To map form design objects to custom field controls:**

1. Start Designer ES, open a form design that contains a form guide, and start Guide Builder.
2. Select the form design object that you want to map in the form guide hierarchy.
3. On the **Edit field properties** menu, click **Display field as a**, and select the name of your custom control. For example, using the custom control created in the Example: CustomHSlider section, select **Custom H Slider**.
4. Save the form guide.

Preview your form guide to view the new custom control.

What's next?

Create your own custom controls, beginning with any of the examples in this chapter, or by starting from a new MXML component or ActionScript class. For more information about the form guides ActionScript API, see *LiveCycle ES ActionScript Language Reference*.

To start learning about creating custom panel layouts, see Creating Panel Layouts or, for custom form guide layouts, see Creating Form Guide Layouts.

7 Custom Component Lists

When you create a custom Flex library project, you must include the form guide SWC files. Including the SWC files adds a collection of custom form guide components in a Form Guide folder in your Flex Builder Components view. The custom form guide components include everything you need to create custom form guide layouts and panel layouts.

The custom components that you can add to layouts are listed alphabetically in the Flex Builder Components view but conceptually fall into the following categories:

- Button components
- Help components
- Label components
- Navigation components
- Output components

Button components

Button components provide the most common form guide actions.

AddPanelButton: Adds a new panel to a list of repeating panels. This button is available only when the following statements are true:

- The current panel can repeat.
- Adding a new panel does not conflict with the maximum occurrence value of the associated subform object on the form.

CopyPanelButton: Creates a copy of the currently selected panel and adds it to the list of repeating panels. This button is available only when the following statements are true:

- The current panel can repeat.
- Adding a copy of the current panel does not conflict with the maximum occurrence value of the associated subform object on the form.

NextPanelButton: Displays the next panel in the form guide hierarchy. If the current panel is the last panel in the hierarchy, this button is not available.

PreviousPanelButton: Displays the previous panel in the form guide hierarchy. If the current panel is the first panel in the hierarchy, this button is not available.

RemovePanelButton: Removes the current panel from the list of repeating panels. This button is available only when the following statements are true:

- The current panel can repeat.
- Removing the current panel does not conflict with the minimum occurrence value of the associated subform object on the form.

SubmitButton: Displays a submit button, but only when the current panel is the last panel in the form guide hierarchy. When clicked, this button performs one of the following actions, depending on the values selected in the Submit From list in Guide Builder:

PDF: Submits the form guide data from the PDF form. Clicking the submit button in the form guide opens the PDF form, from which the form filler can submit the data. If the PDF form is not available, clicking the submit button in the form guide instructs LiveCycle Forms ES to create the PDF form and return it to the browser so that the form filler can verify it and then click the Submit button in it.

Printed Form: Clicking the submit button opens the PDF form so that the form filler can print the form.

Guide: Clicking the submit button submits the data from the form guide.

Hosted Application: Specifies that the hosted application, such as Workspace ES, controls the data submission. In this case, the form guide does not have a submit button. The hosted application extracts the data from the form guide and performs the data submission.

Help components

Help components let you display help to end users in text, image, and video format:

HelpBox: Displays form guide help.

HelpCenter: Displays a centralized region within a form guide layout to display form guide help and panel help.

HelpPanel: Displays panel help.

HelpVideo: Displays the help video control.

Label components

Label components provide objects that display section and panel titles:

PanelTitle: A label that displays the name of the currently selected panel.

SectionTitle: A label that displays the name of the currently selected section.

Navigation components

Navigation components provide the system for navigating the sections and panels of a form guide:

AccordionNav: An accordion menu composed of sections that each contain a list of panels. The default form guide layout in Guide Builder, named *Left Accordion*, is an example of the AccordionNav component.

MxTreeNav: A tree structure that lists multiple section and panel levels. The default Guide Builder form guide layout named *Cobalt Tree* is an example of the MxTreeNav component.

Note: The MxTreeNav component is the only navigation component that displays nested section and panel levels.

ProgressSectionBarNav: A horizontal list of buttons that represents each section in the form guide hierarchy. The default Guide Builder form guide layout named *Cobalt Bar* is an example of the ProgressSectionBarNav component.

Note: This navigator is useful when each form guide section includes only one panel.

StepNav: An accordion menu that lists section names where each section contains a list of panels. The default Guide Builder form guide layout named *Cobalt Standard* is an example of the usage of the StepNav component.

TabTabNav: A navigation system that consists of two corresponding levels of tab menus. The top-level tabs list the sections in the form guide hierarchy, and the bottom-level tabs list the panels for the currently selected top-level tab. The default Guide Builder form guide layout named *Workspace* is an example of the TabTabNav component.

Output components

Output components provide objects that display content or functionality to users at run time.

PanelContent: Displays the content of form guide panels.

ProgressBar: Indicates the percentage of mandatory fields into which an end user entered data. This control is not available if the form guide does not contain mandatory fields.

ToolBar: Displays the form guide toolbar, which includes the Save PDF, Print PDF, Email PDF, and Show/Hide PDF controls.

Index