CUSTOMIZING THE SOLUTION TEMPLATE FOR THE MANAGED REVIEW & APPROVAL SOLUTION ACCELERATOR 9.5
Contents

Chapter 1: Using the GlobalCorp solution template
Before you begin .................................................................................................................. 1
About the solution template ............................................................................................... 2
Understanding the solution template customizations ....................................................... 8

Chapter 2: Setting up your development environment
Retrieve files to use the solution template ........................................................................ 9
Extract the solution template source code ....................................................................... 10
Setup to build the solution template EAR file .................................................................. 11

Chapter 3: Building and deploying the solution template
Build the solution template .............................................................................................. 17
Deploy the solution template ......................................................................................... 18
Building and deploying a solution template for debugging ........................................... 19

Chapter 4: Changing the theme
Changing the images in the Java Server Pages ............................................................ 21
Modifying the theme in the solution template ............................................................... 23

Chapter 5: Customizing the flow of the initiate review screens
Removing a screen from the review initiation Flex application ..................................... 30
Chapter 1: Using the GlobalCorp solution template

The GlobalCorp solution template provided with the Managed Review & Approval Solution Accelerator 9.5 is available when you install the Adobe® LiveCycle® Enterprise Suite 2.5 (ES2.5) software.

The solution template provides common functionality to meet cross-industry requirements for regulated and structured review processes. Since the source code is provided, you can:

- Use the solution template as a starting point to build your solution.
- Use portions of the solution template in a separate application that you built.
- Use the solution template as provided and perform simple theme changes.

It is important to recognize that the purpose of the solution template is to accelerate development time. Developers are responsible to build an understanding of the source code.

The solution template uses Flex and Java APIs from the Review, Commenting, and Approval building block. (See Review, Commenting, and Approval Building Block 9.5 Technical Guide.) Knowledge of the building block APIs helps you to better understand the solution template source code. As you become more proficient with the source code, you can choose to write your own code using the building block APIs. The building block APIs are available at the following locations:

- The Flex API is available at ActionScript 3.0 Reference for the Adobe Flash Platform.
- The Java API is available at Solution Accelerators API Reference

This Help provides a sample of solution template customizations intended for developers familiar with Flex and web development. Use this document to identify the best practices and become familiar with ways to customize and use the solution template for your solutions. Each topic describes a customization that is sometimes divided into subtasks. Each subsequent subtask builds on the code provided in a previous subtask. Bolded courier font appears in the examples to indicate the new code that is incrementally added or modified.

Before you begin

It is beneficial if you have an understanding of how the provided solution template works. Consider completing the walkthrough for the GlobalCorp solution template. (See Using the solution template in the Managed Review & Approval 9.5 Solution Guide.)

Ensure that you have the following requirements to use the information this Help:
LiveCycle server requirements
You require access to the LiveCycle ES2.5 server and software as follows:

- Access to an application server with LiveCycle ES2.5 software installed on it. You require access to log on to copy files from server.

  For information on how to obtain the LiveCycle ES2.5, contact your administrator or http://www.adobe.com/cfusion/tdrc/index.cfm?product=livecycle for download details. It is also necessary that sample users for the Managed Review & Approval Solution Accelerator are installed on the LiveCycle ES2.5 server to have a testing environment.

- The LiveCycle ES2.5 DVD to install the LiveCycle ES2 version of the Flex SDK or access to an installation of LiveCycle® Workbench 9.5.

Development environment prerequisites
You are expected to have the following installed and configured in your development environment before you begin.

- An installation of Adobe® Flash® Builder™ 4.x Standalone. When you use the Adobe® Flex Builder™ 3.x, Adobe® Flash Builder™ 4.x plug-in for Eclipse, or Adobe® Flex Builder™ 3.x plug-in for Eclipse, the steps in this document do not correspond exactly because the menu commands differ.

- An Integrated Development Environment for Java and Java Server Pages, such as the Eclipse® Integrated Development Environment.

- Java Development Kit (JDK) 1.5.11 or later. Ensure that you set the JDK version as in your JAVA_HOME environment variable.

- Apache Ant 1.7.1 or later with the ant-contrib 1.0b3. (See Installing Apache Ant.)
  
  - Apache Ant is available at http://ant.apache.org/bindownload.cgi and add the bin path to your PATH variable in Windows.
  
  - Ant-contrib 1.0b3 is available at http://sourceforge.net/projects/ant-contrib/. After you extract the ZIP file, copy the ant-contrib-1.0b3.jar file to the lib folder of your Ant installation.

Before you begin to customize the solution template or build a solution, it is necessary to complete these tasks:

- Copy various files to your development environment and configure your computer to build the solution template. (See “Setting up your development environment” on page 9.)

- After you have completed the configuration your environment, test that you can build and deploy the solution template. (See “Building and deploying the solution template” on page 17.)

About the solution template
The GlobalCorp solution template provides the user interface for users to view a dashboard of the reviews. In addition to viewing reviews in the dashboard, users can search on the reviews. Users can use the solution template to initiate reviews and build review templates.

The solution template is built using a combination of web and Flex projects. The following assets are available in the web and Flex projects:

- HTML pages

- Java Server Pages (JSP)

Last updated 3/3/2011
It is necessary to modify one or more assets depending on the area of customization. After you modify the assets, you compile and package the assets into an EAR file. Then you deploy the EAR file to the LiveCycle server and test the solution template.

About the Flex projects in the solution template

There are three Flex projects that provide specific user interface and functions for the solution template. A user can use the Flex applications based on the roles assigned. For example, if a user is assigned the RCA Review Initiator role, they can use the Flex application to initiate reviews. However, the user cannot use the Flex application to create templates. (See Review and approval personas and user roles in Managed Review and Approval Solution Accelerator 9.5 Solution Guide.)

The Flex projects provide the following functions:

- **Dashboard**: The review_dashboard Flex project provides the user interface to view the status of reviews and to search for reviews.

  Dashboard Flex application in solution template.

- **Review Initiation**: The review_initiation Flex project provides the user interface and functionality to start a review using a review template.
Review Template Creation: The review_template_creation Flex project provides the user interface to create review templates.

Characteristics of a Flex project in the solution template
Each Flex project has the following characteristics:

- Controls the flow of the user interface. Core functionality is provided in the rca project, where reusable Flex components, styles, and resource bundles are externalized into separate projects to facilitate modularity.
Contents of each Flex projects

The following lists the projects in the solution template:

- **review_template_creation**: A Flex project that contains the source code to create a review template.
- **review_initiation**: A Flex project contains the source code to use predefined templates to initialize an active review.
- **review_dashboard**: A Flex project that contains the source code to track of reviews and search reviews.
- **common/logging_console**: A Flex project to view generic logging info as well as all service layer communication for debugging purposes.
- **common/rca_rb**: A generic project that contains the language bundles that are used across all Flex projects. You can customize the strings meet terminology requirements for your organization.
- **common/rca_styles**: A Flex Library project that contains the colors, fonts, CSS, and assets used for all across all Flex projects. When it is necessary to include the images in the compilation, there is a config.xml file (located in the rca_styles project). You edit the config.xml to add the path of your new images.
- **common/rca**: A Flex Library project that contains the views, events, model, and services that are used to construct each of the individual applications.

Each of the user interface Flex projects use the rca Flex project as a library. The review_template, review_initiation, and review_dashboards projects use and extend components from the common Flex library. The Flex Library project contains the following items:

- **Components**: The components package contains generic components used by other Flex projects. The components provide building block functions for other components. All components dispatch common events and customized using a common theme file.
- **Events**: The events facilitate navigation container applications and user interfaces screens. Many of the events are used to between user interface screens and passing data between classes.
- **Model**: The components provide classes to store data for the solution template. For example, the information in the review template, the stages in a review, documents, and participants in a review.
- **Services**: Provides components to enable communication between classes, such as the LiveCycle Remoting endpoints and HTTP connections. The components provide mappings to Workspace API components and components from the Review, Commenting, and Approval building block. All Flex projects that use the Flex Library require a ServiceConfig.xml file that configures where services are located and provides the credentials associated to each endpoint.
- **Utilities**: Components provide generic utility classes or helper classes to facilitate code reuse.

All Flex projects provided in the solution template use common libraries as shown in the following illustration:
Organization of Flex projects in the solution template.

About the solution template web projects

The web projects provide an HTML container to display Flex applications. It also provides the functionality to handle the login functionality. Many of the theme changes related to branding are customized in the web projects.

The following web projects are available:

- **mra_reviewportal**: Contains the main source code for the GlobalCorp solution template.
- **login.jsp**: The Java Server Page (JSP) that handles authentication. The web page contains the content that a user sees when they access http://[server name]:[server port]/reviewportal. The JSP page provides single sign-on support.
CUSTOMIZING THE SOLUTION TEMPLATE FOR MANAGED REVIEW & APPROVAL SOLUTION ACCELERATOR 9.5

Using the GlobalCorp solution template

- index.jsp: Contains a JSP that hosts the Flex applications and external user registration web pages. Each tab in the page provides functions using either a Flex application or another JSP. The tabs that are visible depend on the roles of the user that is logged in to the solution template. The roles for each user are assigned in Adobe User Management. (See Review and approval personas and user roles in Managed Review and Approval Solution Accelerator 9.5 Solution Guide.)
CUSTOMIZING THE SOLUTION TEMPLATE FOR MANAGED REVIEW & APPROVAL SOLUTION ACCELERATOR 9.5

Using the GlobalCorp solution template

- **mra_external_users_webapp**: Contains the web pages that can be accessed only after a user authenticates with the LiveCycle server. Users must be configured as users in User Management. The following pages are provided:
  - `showResult.jsp`: A web page that displays that status of the `inviteUsers.jsp` action.
  - `inviteUsers.jsp`: A web page where a user with the RCA Administrator role can submit requests to invite external users to a review.
  - `registerUser.jsp`: A web page that allows users external users invited to a review to register themselves in User Management.
  - `registrationComplete`: A web page that displays the result of the `registerUser.jsp` action.
  - `error.jsp`: A web page that is used to catch exceptions if the users’ session has expired or is invalid.
  - `genericError.jsp`: A web page that displays errors.
  - `login.jsp`: The login page that is displayed after a session expires.
- **mra_enterprise_application**: Contains the files to build the EAR file to deploy the web application and SWC and SWF files built in Flex. It is not common to modify these files.

Each web project contains cascading style sheets (CSS) that can be customized to suite your requirements. Images are also stored in each project. The names of the JSP pages cannot be renamed. Renaming the pages breaks the Single Sign-on functionality implemented by the solution template.

**Understanding the solution template customizations**

If you choose to use the solution template as provided, you can perform simple customizations to change the theme. Generally, theme customizations change the appearance of an application.

**Theme changes**

All styles related to the solution template are available in a separate theme file. Often, you can modify the cascading style sheet (CSS) files to reskin and change the appearance of the solution template.

Depending on the theme changes that you require, you modify the web project files, Flex projects, or both. For example, if you wanted to make branding changes to the header in the solution template, you modify the web project. However, if you wanted to reskin the buttons in the review initiation user interface, you modify the Flex projects. (See “Changing the theme” on page 21.)

**Flow changes**

You can also choose to change the flow of how the solution template functions. These customizations often require that you modify the solution template in the Flex projects. The solution template is written using Model-View-Controller methodologies. Familiarity with the programming model helps you to better understand the source code to achieve the flow changes that you require.

The source code uses APIs provided by the Review, Commenting, and Approval building block. Depending on the complexity of your customizations and the area of the solution template can require that you understand the Java or Flex API. (See “Customizing the flow of the initiate review screens” on page 30.)
Chapter 2: Setting up your development environment

It is necessary that you configure your development environment to build the solution template. Before you step through the setup for your development environment, ensure that you have the prerequisites installed on your development environment. (See “Development environment prerequisites” on page 2.)

To set up your development environment, perform the following steps:

1. Retrieve files to build the solution template. (See “Retrieve files to use the solution template” on page 9.)
2. Extract the solution source code to your computer. (See “Extract the solution template source code” on page 10.)
3. Import the source code to Flash Builder and a Java integrated development environment (IDE) and configure your development environment to build the solution template. Optionally, you can configure Flash Builder and your IDE to build the Flex projects and web projects separate. (See “Setup to build the solution template EAR file” on page 11.)

Retrieve files to use the solution template

To customize the solution template, retrieve and copy the solution template source code and the related files. To retrieve the files, you require access to a LiveCycle server and LiveCycle ES2.5 DVD.

As part of setting up your development, copy the following files to your computer:

- LiveCycle ES2.5 SDK, which contains the necessary files to compile the solution template. (See “Copy the LiveCycle SDK, J2EE JAR files, and Solution Accelerator APIs” on page 9.)
- Solution Accelerator APIs, which contain the binaries require to build the solution template. (See “Copy the LiveCycle SDK, J2EE JAR files, and Solution Accelerator APIs” on page 9.)
- Connection mode files, which are J2EE JAR files that are specific for the application server for which LiveCycle is installed and for accessing the servlet APIs. (See “Copy the LiveCycle SDK, J2EE JAR files, and Solution Accelerator APIs” on page 9.)
- LiveCycle version of the Flex SDK, which is required to compile the Flex applications provided as part of the solution template. (See “Copy the LiveCycle version of the Flex API” on page 10.)

Copy the LiveCycle SDK, J2EE JAR files, and Solution Accelerator APIs

   For example, on the LiveCycle server, navigate to C:/Adobe/LiveCycle ES2.

2. Copy the LiveCycle_ES_SDK folder to your computer.
   For example, copy C:/Adobe/Adobe LiveCycle ES2/LiveCycle_ES_SDK from the server to C:/Adobe/Adobe LiveCycle ES2/LiveCycle_ES_SDK on your computer.

   Note: When you have Workbench installed, you can copy the LiveCycle ES2.5 SDK from C:/Program Files/Adobe/Adobe LiveCycle Workbench ES2/LiveCycle_ES_SDK.

3. Copy the sa_resources folder to your computer.
For example, copy the C:/Adobe/Adobe LiveCycle ES2/sa_resources from the LiveCycle server to C:/Adobe/Adobe LiveCycle ES2/sa_resources on your computer.

4 Copy the following the connection mode files from the LiveCycle server to your computer.
For example, when LiveCycle is installed on a JBoss application server, copy the jboss/client/jbossall-client.jar and jboss/client/servlet-api.jar files to C:/Adobe/Adobe LiveCycle ES2/LiveCycle_ES_SDK on your computer.

The connection mode files you require are dependent on application server that LiveCycle is running on. (See Connection mode and J2EE application JAR files in Programming with LiveCycle ES2.5 Help.)

**Copy the LiveCycle version of the Flex API**
The solution template is created using a specific version of the Flex API that is compatible with LiveCycle.

1 Copy the LiveCycle version of the Flex SDK to your computer and install it to Flash Builder. The LiveCycle version of the Flex SDK is available from LiveCycle ES2.5 DVD.
   • On the LiveCycle ES2 DVD, navigate to the additional\flex_sdk folder and copy the flex_sdk_3.zip to your computer.
   
   **Note:** When you have Adobe® LiveCycle® Workbench 9.5 installed, you can copy the LiveCycle version of the Flex SDK from C:/Program Files/Adobe/Adobe LiveCycle Workbench ES2/Flex SDK.

2 Navigate to the sdks folder located in the root folder of your installation of Flash Builder. For example, navigate to the C:/Program Files/Adobe/Adobe Flash Builder 4/sdks folder.

3 Create a folder, such as 3.4.1.lc

4 Using an archival and extraction tool, extract the flex_sdk_3.zip file to the folder created in the previous step.

**Extract the solution template source code**
The source code is located in the sa_resource folder that you copied to your computer. (See “Copy the LiveCycle SDK, J2EE JAR files, and Solution Accelerator APIs” on page 9.)

1 Navigate to the location you copied the sa_resources folder on your computer.

2 Navigate to SA_SDK_9.5/ManagedReviewAndApproval folder.

3 Extract the GlobalCorpReviewAndApproval.zip file to your computer. The location you extract it to is called [MRA_ROOT] throughout this Help.

   For example, if you extract it a folder named C:\MRA, then your [MRA_ROOT] is called C:\MRA.

**About the solution template folder structure**
After you extract the GlobalCorpReviewAndApproval.zip file, there a number of folders that are created that you should understand.

**build:** Contains ANT scripts to compile the Flex and web projects and build the managedreviewandapproval.ear file. The EAR file is deployed to the LiveCycle server.

**web:** Location of the web projects for the solution template.

**flex:** Location of the Flex source code for the solution template.
Setting up your development environment

**deploy**: Location where the managedreviewandapproval.ear file is created. You access this folder after a successful build to access the EAR file.

The following is the extracted folder structure of the source code:

_Note: The folder flex/common/rca/dist is created after a folder is built._

```
- GlobalCorpReviewAndApproval
  - build
  - deploy
  - flex
    - common
      - logging_console
    - rca
      - .settings
      - src
    - rca_rb
    - rca_styles
  - review_dashboard
  - review_initiation
  - review_template_creation
  - lca
  - thirdparty
  - web
```

Folder structure of the extracted GlobalCorpReviewAndApproval.zip file.

---

**Setup to build the solution template EAR file**

An Ant script is provided to build the solution template. The script compiles the Flex project and web projects and packages the Web ARchive (WAR) and Enterprise ARchive (EAR) files. Optionally, you can configure Flash Builder and Eclipse to build your projects separately, which is useful when you want to modify Flex or Web projects.

1. Navigate to the location you extracted the GlobalCorpReviewAndApproval.zip file and navigate to the build folder. For example, $MRA_ROOT$/GlobalCorpReviewAndApproval/build. (See “Extract the solution template source code” on page 10.)

2. Edit the variables under the User-defined properties file in the mra.build.properties file. Replace the default values for the following properties as they correspond to your development environment:

   - **lcsdk.dir**: Set the value you copied the LiveCycle_ES_SDK folder to on your computer. (See “Copy the LiveCycle SDK, J2EE JAR files, and Solution Accelerator APIs” on page 9).
     For example, set the value to $C:/Adobe/Adobe LiveCycle ES2/LiveCycle_ES_SDK$.

   - **FLEX_HOME**: Set the value to the location you extracted the LiveCycle ES2 version of the Flex SDK. (See “Copy the LiveCycle version of the Flex API” on page 10).
     For example, set the value to $C:/Program Files/Adobe/Adobe Flash Builder 4/sdks/3.4.1.lc$.

   - **j2ee.jar**: Set the value to location and name of application-specific J2EE JAR files and servlet-api.jar file you copied to your computer. (See “Copy the LiveCycle SDK, J2EE JAR files, and Solution Accelerator APIs” on page 9.)
For example, set the value to `C:/Adobe/Adobe LiveCycle ES2/LiveCycle_ES_SDK/jbossall-client.jar;C:/Adobe/Adobe LiveCycle ES2/LiveCycle_ES_SDK/servlet-api.jar`.

- `rca.core.client.dir`: Set the value to client libraries for the Review, Commenting, and Approval building block. The folder is located in the `sa_resources` folder that you copied to your computer. (See “Copy the LiveCycle SDK, J2EE JAR files, and Solution Accelerator APIs” on page 9.)

  For example, set the value to `C:/Adobe/Adobe LiveCycle ES2/sa_resources/SA_SDK_9.5/client-libs/common`.

3  Save the file.

After you complete the changes, the `mra.build.properties` looks similar to the following file:

```
#
#
# User defined properties
# ................................................................. # .................................................................
# lcsdk.dir=C:/Adobe/Adobe LiveCycle ES2/LiveCycle_ES_SDK
# FLEX_HOME=C:/Program Files/Adobe/Adobe Flash Builder 4/sdks/3.4.1.lc
#J2EE jar path (path should include the jar name)
#j2ee.jar= C:/Adobe/Adobe LiveCycle ES2/LiveCycle_ES_SDK/jbossall-client.jar;C:/Adobe/Adobe LiveCycle ES2/LiveCycle_ES_SDK/servlet-api.jar
#Review Commenting And Approval client directory path
#rca.core.client.dir = C:/Adobe/Adobe LiveCycle ES2/sa_resources/SA_SDK_9.5/client-libs/common
#
```

### Set up build the solution template with Flex debug option

Optionally, you can configure the build scripts allow you to run Flash Player debugger on the Flex applications in the solution template. When you build the Flex projects with the debug option, you can run the Flash Builder debugger to step through and trace the code. (See “Building and deploying a solution template for debugging” on page 19.)

1  Navigate to the location you extracted the `GlobalCorpReviewAndApproval.zip` file and navigate to the `flex` folder. For example, `[MRA_ROOT]/GlobalCorpReviewAndApproval/flex`. (See “Extract the solution template source code” on page 10.)

2  Open the `[MRA_ROOT]/GlobalCorpReviewAndApproval/flex/common/logging_console/build.xml` and search for the `mxmlc` element file.

3  Add the `debug="true"` option after each `mxmlc` you find. Ensure that you search the entire `build.xml` file.

   For example, add modify each line as follows:

```xml
<mxmlc debug="true" compiler.accessible="true"
```

4  Save the changes.

5  Repeat steps 2 – 4 for the following files.

   - `[MRA_ROOT]/GlobalCorpReviewAndApproval/flex/review_dashboard/build.xml`
   - `[MRA_ROOT]/GlobalCorpReviewAndApproval/flex/review_initiation/build.xml`
   - `[MRA_ROOT]/GlobalCorpReviewAndApproval/flex/review_template_creation/build.xml`
Set up Flash Builder to build the solution template code

You can use Flash Builder to customize the Flex portions of the solution template. If you choose not to use Flash Builder, you can edit the source code using a text editor of your choice. You can build the Flex portions of the solution template using the provided Ant script.

To import the Flex projects and configure Flash Builder to build the Flex projects, complete the following procedures:

**Configure Flash Builder to use the LiveCycle version of the Flex SDK**

To customize the solution template Flex projects or applications with Flex using the building block Flex APIs, configure Flash Builder with LiveCycle version of the Flex SDK. It is recommended that you set the LiveCycle version of the Flex SDK as the default.

1. In Flash Builder, select Project > Build Automatically to deselect the option to build automatically.
2. Select Window > Preferences.
3. In the Preferences dialog box, select Flash Builder > Installed Flex SDKs and click Add.
4. In the Add Flex SDK dialog box, complete the following steps:
   - Click Browse.
   - In the Browse For Folder dialog box, select the folder you created in step 3 of the task, “Copy the LiveCycle version of the Flex API” on page 10, and click OK. For example, C:\Program Files\Adobe\Adobe Flash Builder 4\sdks\3.4.1.lc.
   - In the Flex SDK name box, type a name to identify the LiveCycle ES2 version of the Flex SDK. For example, Flex 3.4.1 - LCES2.5.
   - Click OK.
5. Select the check box beside the Flex SDK version that you added in the previous step, click Apply, and OK.

**Import the source code to Flash Builder**

You can import the source code to Flash Builder to build the portions of the solution template developed with Flex. After you import the source code, it is necessary for you to add environment variables. The following steps presume that you have extracted the source code to a folder on your computer.

1. In Flash Builder, navigate to File > Import > Other.
2. In the Import dialog box, select General > Exist Projects into Workspace, and click Next.
3. Click Browse beside the Select root directory option and navigate
   
   [MRA_ROOT]/GlobalCorpReviewAndApproval/flex.

   [MRA_ROOT] is the location you extracted the solution template source code. (See “Extract the solution template source code” on page 10.)
4. Ensure that the review_template_creation, review_dashboard, review_initiation, rca, rca_styles, rca_rb are selected and click Finish.

**Configure Flash Builder to build the solution template**

You do not require Flash Builder to build the solution template because the Ant script compiles the Flex projects as part of building the EAR file. (See “Setup to build the solution template EAR file” on page 11.) However, the ability to build the source is useful when you want to debug issues or ensure that your syntax is correct.
CUSTOMIZING THE SOLUTION TEMPLATE FOR MANAGED REVIEW & APPROVAL SOLUTION ACCELERATOR 9.5

Setting up your development environment

Note: There are a number of warnings when you build the provided source code. You can ignore the warnings related to the provided source code.

1. In Flash Builder, select Window > Preferences
2. In the Preferences dialog box, select General > Workspace > Linked Resources.
3. Click New.
4. In the New Variable dialog box, type LC_SDK in the Name box, and type the location to which you copied the LiveCycle SDK. For example, C:\Adobe\LiveCycle ES2\LiveCycle_ES_SDK.
5. Click OK.
6. Repeat steps 3- 5 but create the following environment variables:
   - Name: ESUB_HOME Location: [MRA_ROOT]/GlobalCorpReviewAndApproval/flex
   - Name: AS3CORELIB_DIR: [MRA_ROOT]/GlobalCorpReviewAndApproval/thirdparty/as3corelib_tp/as3corelib
   [MRA_ROOT] is the location you extracted the solution template source code. (See “Extract the solution template source code” on page 10.)
7. Click OK.

Set up Eclipse to build the web project

You can import the source code to an integrated development environment (IDE) for editing the Java source and Java Server Page code. An integrated development environment, such as Eclipse can be used to build a Web ARchive (WAR) file to deploy to a web server. After you import the source code, it is necessary for you to configure environment variables and configure the projects to include JAR files.

To import the web projects and configure Eclipse to build a WAR file, complete the following procedures:

Import the Java code into Eclipse
1. In Eclipse, select Project > Build Automatically to deselect the option.
2. Select File > Import.
3. In the Import dialog box, select General > Existing Projects into Workspace, and click Next.
4. Click the Browse button beside the Select Root Directory box.
5. In the Browse For Folder dialog box, navigate to the location where you [MRA_ROOT]/GlobalCorpReviewAndApproval/web folder, and click OK.
   [MRA_ROOT] is the location you extracted the solution template source code. (See “Extract the solution template source code” on page 10.)
6. In the Import dialog box, ensure all the projects are selected in the Projects pane and click Finish.
7. In the Project Explorer, close the mra_enterprise_application project. There is no requirement to modify it as it is used to build the EAR file.

Configure Eclipse to build the web project

After you import the web projects into Eclipse, configure it to build the WAR file. You can also use Eclipse to ensure that the syntax you provide is correct. You can deploy the created WAR file to a web server to test the Java Server Pages (JSP).
Note: Because Eclipse is strict with HTML rules, after you complete configuration, HTML errors appear. You can safely ignore the errors.

1. In Eclipse, select Window > Preferences.
2. In the Preferences dialog box, select Java > Installed JREs and complete the following steps:
   • Click Add.
   • In the Add JRE dialog box, select Standard VM, and click Next.
   • Click Directory.
   • In the Browse For Folder dialog box, select the location of your JDK (version 1.5.11 and higher), and click OK.
   • Click Finish.
   • In the Installed JREs, select the name of the JDK you previously selected.
3. Select Java > Build Path > Classpath Variables and click Add.
4. Click New.
5. In the New Variable dialog box, type LC_SDK in the Name box, and type the location to which you copied the LiveCycle SDK. For example, C:/Adobe/Adobe LiveCycle ES2/LiveCycle_ES_SDK.
6. Click OK.
7. Repeat steps 4 - 6 to create the following environment variables and then click OK:

<table>
<thead>
<tr>
<th>Name</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>J2EE_HOME</td>
<td>C:/Adobe/Adobe LiveCycle ES2/LiveCycle_ES_SDK</td>
</tr>
<tr>
<td>TOOLS_DIR</td>
<td>[MRA_ROOT]/GlobalCorpReviewAndApproval/thirdparty</td>
</tr>
</tbody>
</table>

Note: The location for each variables depends on the location you copied the files to your computer and extracted the solution template source code. (See “Retrieve files to use the solution template” on page 9 and “Extract the solution template source code” on page 10.)

Note: The J2EE_HOME variable is the same as the LC_SDK variable because the projects are configured to different variables in the build scripts.

8. In the Project Explorer, right-click the mra_reviewportal project, select Properties, and complete the following steps:
   • Select Java Build Path and select the Libraries tab.
   • Select and Remove the following entries:
     • adobe-reviewcommentingapproval-internal-lib.jar
     • IDBU_TOOLS_TP_LIB/j2ee_tp/j2ee-1.3.1/j2ee.jar
     • LC_CORE_DIR/external/sdk/common/adobe-livecycle-client.jar
     • LC_CORE_DIR/external/sdk/common/adobe-usermanager-client.jar
   • Click Add External JARS and add the following JARs to the project:
     • C:/Adobe/LiveCycle ES2/LiveCycle_ES_SDK/jbossall-client.jar (or the JAR specific for your application Server.)

Last updated 3/3/2011
CUSTOMIZING THE SOLUTION TEMPLATE FOR MANAGED REVIEW & APPROVAL SOLUTION ACCELERATOR 9.5

Setting up your development environment

- C:/Adobe/LiveCycle ES2/LiveCycle_ES_SDK/servlet-api.jar
- C:/Adobe/LiveCycle ES2/LiveCycle_ES_SDK/client-libs/common/adobe-user-management-client.jar
- C:/Adobe/LiveCycle ES2/LiveCycle_ES_SDK/client-libs/common/adobe-livecycle-client.jar
- [MRA_ROOT]/GlobalCorpReviewAndApproval/thirdparty/standard_tp/standard-1.0.6/standard.jar
  - Select the JRE System Library [jdk1.6.0](unbound) and click Edit.
  - In the Edit Library, select Workspace default JRE (jdk1.5.11) or your default JDK version.

9 In the Project Explorer, right-click the mra_external_user_webapp project, and select Properties.
  - In the Properties For mra_external_users_webapp dialog box, select Java Build Path, and then the Libraries tab.
  - Select and Remove the following entries:
    - adobe-reviewcommentingandapproval-client.jar
    - adobe-reviewcommentingapproval-internal-lib.jar
    - IDBU_TOOLS_TP_LIB/commons-lang_tp/commons-lang-2.4/commons-lang.jar
    - IDBU_TOOLS_TP_LIB/j2ee_tp/j2ee-1.3.1/j2ee.jar
    - IDBU_TOOLS_TP_LIB/jstl_tp/jstl-1.0.6/jstl.jar
    - IDBU_TOOLS_TP_LIB/standard_tp/standard-1.0.6/standard.jar
    - LC_CORE_DIR/external/sdk/common/adobe-livecycle-client.jar
    - LC_CORE_DIR/external/sdk/common/adobe-usermanager-client.jar
  - Click Add External JARS and add the following JARs to the project:
    - C:/Adobe/LiveCycle ES2/LiveCycle_ES_SDK/jboss-client.jar (or the JAR specific for your application Server.)
    - C:/Adobe/LiveCycle ES2/LiveCycle_ES_SDK/servlet-api.jar
    - C:/Adobe/LiveCycle ES2/LiveCycle_ES_SDK/client-libs/common/user-management-client.jar
    - C:/Adobe/LiveCycle ES2/LiveCycle_ES_SDK/client-libs/common/adobe-livecycle-client.jar
    - [MRA_ROOT]/GlobalCorpReviewAndApproval/thirdparty/commons-lang_tp/commons-lang-2.4/commons-lang.jar
    - [MRA_ROOT]/GlobalCorpReviewAndApproval/thirdparty/jstl_tp/jstl-1.0.6/jstl.jar
    - [MRA_ROOT]/GlobalCorpReviewAndApproval/thirdparty/standard_tp/standard-1.0.6/standard.jar
  - Select the JRE System Library [jdk1.6.0](unbound) and click Edit.
  - In the Edit Library, select Workspace default JRE (jdk1.5.11) or your default JDK version.

*Note: [MRA_ROOT] is the location you extracted the solution template source code. (See “Extract the solution template source code” on page 10.*)
Chapter 3: Building and deploying the solution template

When you customize the solution template, it is necessary to compile the source code and create an EAR file. The EAR file is deployed on the application server where LiveCycle is installed. You use an Ant script to compile the solution template source code and create an EAR file.

The solution template is deployed as an EAR file, named adobe-mra-managedreviewandapproval.ear. Before you deploy your custom EAR file, create a backup of the original adobe-mra-managedreviewandapproval.ear file that is located on the LiveCycle server.

Build the solution template

You can build and compile the solution template using an Ant command at command prompt. After the build completes, an EAR file is provided in the [MRA_ROOT]/GlobalCorpReviewAndApproval/deploy folder as shown in the following illustration:

<table>
<thead>
<tr>
<th>Folders</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlobalCorpReviewAndApproval</td>
<td>adobe-mra-managedreviewandapproval.adobe</td>
</tr>
<tr>
<td>deploy</td>
<td>adobe-mra-managedreviewandapproval.ear</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Create EAR file in folder structure.

Note: [MRA_ROOT] is the location you extracted the solution template source code. (See “Extract the solution template source code” on page 10.)

You can deploy the EAR file to an application server for testing.

Optionally, you can configure Flash Builder to compile the solution template source code and your integrated development environment to compile the web projects for the solution template. (See “Build the Flex projects in the solution template” on page 18.)

Even if you compile the code in the Flash Builder or your IDE, you are required to run the Ant script to create the EAR file.

1. In a command prompt window, navigate to the location you extracted the GlobalCorpReviewAndApproval.zip file and navigate to the build folder.
   
   For example, [MRA_ROOT]/GlobalCorpReviewAndApproval/build.

2. Type ant.
Build the Flex projects in the solution template

Optionally, after you import the source code, you can build the Flex project in Flash Builder simplify the editing and debugging experience. There are a number of warnings that you can ignore. You can build the provided Flex code in Flash Builder if you have configured your development environment. (See “Set up Flash Builder to build the solution template code” on page 13.)

❖ In Flash Builder, select Project > Build All.

Note: If you are building the Flex projects for the first time, it is often necessary to clean the files. Select Project > Clean to clean all the projects.

Note: There are a number of warnings and errors when you build the provided source code. You can ignore the warnings related to the source code.

Build the web projects in the solution template

Optionally, you can create a WAR file using an integrated development environment (IDE), such as Eclipse. You can build the source in Eclipse after you configure your development environment and import the source code. (See "Set up Eclipse to build the web project" on page 14.)

1 In Eclipse, close the mra_enterprise_application project.
2 Select Project > Build All.
3 Select File > Export > WAR file.
4 In the WAR Creation dialog box, select the location to export the file to.

Note: If you are building the Flex projects for the first time, it is often necessary to clean the files. Select Project > Clean to clean all the projects.

Note: Because Eclipse is strict with HTML rules, after you complete the configuration, HTML errors appear. You can ignore all warnings and the errors that pertain to missing start tags, such as No start tag (<PRE>.

Deploy the solution template

You deploy an EAR file to a LiveCycle ES2 server to deploy a custom version of the solution template. The EAR file is created after you successfully run the Ant script. (See “Build the solution template” on page 17.)

The provided steps use a JBoss Turnkey installation of LiveCycle. There are minor variations to deploying the EAR file depending on the application server that the LiveCycle server is installed on.

Note: After you complete this procedure, the original adobe-mra-managedreviewandapproval.ear file is overwritten. Ensure you backup the EAR file before you complete this procedure.

1 In a command prompt window, navigate to the location you extracted the GlobalCorpReviewAndApproval.zip file and navigate to the deploy folder.
   
   For example, [MRA_ROOT]/GlobalCorpReviewAndApproval/deploy.

   [MRA_ROOT] is the location you extracted the solution template source code. (See “Extract the solution template source code” on page 10.)

2 Copy the adobe-mra-managedreviewandapproval.ear file to the LiveCycle server. For example, copy the EAR file to C:/Adobe/LiveCycle ES2/server/lc_turnkey/deploy folder for a development server.
Building and deploying the solution template

**Note:** In a production environment, copy to the [location LiveCycle is installed]/Adobe/LiveCycle ES2/deploy folder and run the LiveCycle Configuration Manager to deploy the EAR file.

3 Restart the application server if necessary. Some application servers require a restart after an EAR file is deployed.

4 Test the solution template in a web browser at http://[server_name]:[server port]/reviewportal. For example, if you have a server named mra-server on a JBoss installation, type http://mra-server:8080/reviewportal in the web browser.

**Building and deploying a solution template for debugging**

Optionally, you build your solution template with the debug option to allow you to run a debugger on it. You can choose to debug the entire solution template or individual Flex projects. (See “Set up build the solution template with Flex debug option” on page 12.)

**Note:** To use debugging, install the debug version of Flash Player. (See Adobe Flash Player Support Center.)

**Debugging a specific Flex project in the solution template**

You can

1 Configure the build.xml files to build the solution template with the debug option. (See “Set up build the solution template with Flex debug option” on page 12.)

2 Build and deploy the EAR file. (See “Build the solution template” on page 17 and “Deploy the solution template” on page 18.)

3 In Flash Builder, select the project you want to debug, such as review_initiation Flex project.

4 Select Run > Debug > Other.

5 (Optional) Place breakpoints in the source files for the project.

6 In the Debug Configurations dialog box, select Web Application and click the New button.

7 In the Main tab, deselect the Use Default check box.

8 In the box below the Use Default check box, type the URL for the Flex project you want to debug as follows:

<table>
<thead>
<tr>
<th>Flex Project</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>review_dashboard</td>
<td>http://[server name]:[server port]/reviewdashboard/ReviewDashboard_en_US.html?debug=true</td>
</tr>
<tr>
<td>Review_initiation</td>
<td>http://[server name]:[server port]/reviewinitiation/ReviewInitiation_en_US.html?debug=true</td>
</tr>
<tr>
<td>Review_template_creation</td>
<td>http://[server name]:[server port]/reviewtemplatecreation/ReviewTemplateCreation_en_US.html?debug=true</td>
</tr>
</tbody>
</table>

9 Click Apply and then Close.

10 Ensure that the Flex application you want to debug is selected in the Package Explorer view.

11 Switch to the Flash Debug perspective.

12 Click the Debug button in the toolbar.

13 In the web browser, enter a user ID and password to log in to the solution template.
After you complete the steps, the Flex application is visible in the web browser. In addition, you can see tracing messages in the Console view and execution stop at the line where you placed a breakpoint.

**Debugging the solution template**

Set the output folder to the location the WAR file is deployed on the LiveCycle server to debug multiple Flex projects in the solution template. You require access to the location that EAR file is expanded on the LiveCycle server. The location differs for each application server.

*Note: You can complete these steps only when you have access to the file system on the LiveCycle server.*

The location of the WAR file can change after you deploy.

1. Configure the build.xml files to build the solution template with the debug option. (See “Set up build the solution template with Flex debug option” on page 12.)

2. Build and deploy the EAR file. (See “Build the solution template” on page 17 and “Deploy the solution template” on page 18.)

3. In Flash Builder, select the project you want to debug, such as Review_initiation.

4. (Optional) Place breakpoints in the source files for the project.

5. Select Run > Debug > Other.

6. In the Debug Configurations dialog box, select Web Application and click the New button.

7. In the Main tab, deselect the Use Default check box.

8. In the box below the Use Default check box, type the location where the solution template is extracted on your application server.

   In a LiveCycle Turnkey installation, the folder would be C:\Adobe\Adobe LiveCycle ES2\jboss\server\lc_turnkey\tmp\deploy\tmp5353856867532484677adobe-mra-managedreviewandapproval.ear-contents\adobe-managedreviewandapproval-reviewportal-exp.war.

   For example, if the name of your LiveCycle server was lcserver, then type \lcserver\C:\Adobe\Adobe LiveCycle ES2\jboss\server\lc_turnkey\tmp\deploy\tmp5353856867532484677adobe-mra-managedreviewandapproval.ear-contents\adobe-managedreviewandapproval-reviewportal-exp.war.

9. Click Apply and then Close.

10. In a web browser, log in to the solution template. For example, use the user ID apink and password of password.

11. In Flash Builder, add breakpoints as required. For example, in the review_initiation Flex project, navigate to src > com.adobe.solutions.rca > view > workflow. Open the ReviewWorkflow.mxml file and set a breakpoint at the following line under the viewReviewDefinition function at approximately line 77:

   workflowViewStack.selectedChild = reviewDefinition;

12. In the Flash Debug perspective, click the Debug icon.

13. Minimize the Adobe Flash Player window when it appears.

14. Access one of the areas that has a Flex application in the solution template. For example, click the Initiate Review tab and then double-click a review template.

After you complete the steps, you can see tracing messages in the Console view and execution stop at the line where you placed a breakpoint.
Chapter 4: Changing the theme

You can change the style to reskin the solution template. Reskinning is a frequently used term to describe changing the theme, styling an application, or changing the look and feel (appearance), without impacting the applications functions.

There are two areas that you can customize depending on your customization requirements.

- Java Server Pages (JSP): To customize the images that appear in the header that are HTML-based, you customize the Java Server pages in the solution template. You can customize CSS files or the JSP pages to change the layout or what appears on the web page.
- Flex applications: To customize the appearance of the review initiation or review template user interface, you customize a CSS file that creates a theme file. The other solution template Flex applications use the theme file to control their appearance.

A good understanding of modifying themes in Flex, HTML, Cascading Style Sheets (CSS), and JSP pages is required to achieve the theme customizations you require.

A common sample of customizations has been described to help you to understand how to customize the solution template. Since the source code for the solution template is provided, consider familiarizing yourself with the source code to understand how to best optimize your theme customizations.

Changing the images in the Java Server Pages

All resources used in the solution template are located under [MRA_ROOT]/GlobalCorpReviewAndApproval/web/review_portal/WebContent/images. For example, the banner or header in the solution template is commonly changed to match an organization’s logo or branding requirements.

When you replace an image, it is a good idea to use an image with the same dimensions. Following this practice helps to ensure that your image appears as you intended.

In some cases, it is necessary to modify the HTML to achieve the appearance that you want. For example, when you use an image with a different color from the original image, modify the HTML to use a matching color.

Change the GlobalCorp header in the solution template

The GlobalCorp header is simply an image displayed in a JSP page in the mra_reviewportal. The name of the image is Banner.png. There are two files that you customize named login.jsp and index.jsp. They have separate implementations so different steps are used to customize each file.

- Banner.png: The current image is the GlobalCorp logo.
- BannerBuffer.png: The current image is a blue background that matches the GlobalCorp logo background.
- Logo.gif: A transparent image of the GlobalCorp logo. When you customize this image, create a transparent image.
- header_bg.jpg: An image that is used as background. The transparent Logo.gif is displayed on top of it.
Change the files to modify the header
The following images are used to replace the existing images.

- In place of the Banner.png, SampleHeader.png is used.
- In place of BannerBuffer.png, SampleBannerBuffer.png is used.
- In place Logo.gif, SampleHeaderTransparent.png is used.
- In place of header_bg.jpg, SampleBackground.png is used.

Complete the following steps to customize the header in the solution template.

1. In Eclipse, in the Project Explorer, navigate to mra_reviewportal > WebContent > Images and complete the following steps:
   - From Explorer, drag your images to the mra_reviewportal > WebContent > images folder.
     For example, drag SampleHeader.png and SampleBannerBuffer.png to the folder.
   - Edit the WebContent/index.jsp page, change the name of the Banner.png and BannerBuffer.png to the names of your images, and save the file.
     For example, Banner.png to SampleHeader.png and BannerBuffer.png to SampleBannerBuffer.png.
   - Change the alt property to match your organizations name.

2. From Explorer, drag your images to the mra_reviewportal > WebContent > images > org folder. For example, drag SampleTransparentHeader.png and SampleBannerBuffer.png to the folder.

3. In the Project Explorer, edit WebContent/login.jsp page and change the image to image in the header to use the image your imported. For example, type SampleTransparentHeader.png image for the src attribute in the <img> tag.

4. From Explorer, edit the WebContent > css > style.css file, and configure your image for the header background in the background attribute of the header class.
23

CUSTOMIZING THE SOLUTION TEMPLATE FOR MANAGED REVIEW & APPROVAL SOLUTION ACCELERATOR 9.5

Changing the theme

5. Save the files.

6. (Optional) You can build the web project in the solution template to help debug changes you make. The web project is also compiled in the following step. (See “Build the web projects in the solution template” on page 18.)

7. Build and deploy the EAR file. (See “Building and deploying the solution template” on page 17.)

After you complete the steps, the header in the login screen looks like the following illustration:

Modifying the theme in the solution template

Each Flex project uses a common skinning SWC in the solution template. Images for branding are also included in the rca_styles project.

Last updated 3/3/2011
You can modify the following cascading style sheets (CSS) files to reskin the Flex applications:

- styles.css: Contains the common styles shared by all Flex applications and more than one Flex component. For example, background colors, images controls, and fonts.
- components.css: Contains component-specific styles.

For example, you can change the color of the navigation bar or the color of the text in the Flex applications.

To determine the CSS elements to modify, navigate through the source code to understand how the elements are used. The more you familiarize yourself with the source code, the more proficient you become to determine what to change to achieve theme customizations you require.

**Modify the navigation bar in the review dashboard Flex application**

The navigation bar provides search and shows the step the user is on when they create a review template or initiate a review. You can change the background color or customize the fonts. By changing the `navigationHeaderLabel` and `leftNav` CSS classes in the `styles.css` file.

1. In Flash Builder, in the rca_styles project, navigate to the src > (default package > styles folder.
2. Edit the `styles.css` file.
3. Change the `background-color` attribute for the `leftNav` class. For example, change the color to `#ffebfb`.
   ```css
   .leftNav
   {
      background-color: #ffebfb;
      padding-left:5;
   }
   ```
4. Change the `font-color` attribute for the `navigationHeaderLabel`. For example, change the color to `#ff0000`.
   ```css
   .navigationHeaderLabel
   {
      font-size:12;
      font-weight:bold;
      color: #ff0000;
   }
   ```
5. Save the file.
6. (Optional) You can build the Flex project in the solution template to help debug changes you make. The Flex project is also compiled in the following step. (See “Build the Flex projects in the solution template” on page 18.)
7. Build and deploy the EAR file. (See “Building and deploying the solution template” on page 17.)

After you complete the steps, the dashboard appears with different fonts and a colored background as shown in the following illustration:
A pink background in the dashboard.

Modify the background for the review template creation Flex application

The review template creation uses the same CSS element for background colors. You can create a copy of the existing style and modify the source to use your new style.

1 In the review_template_creation project, edit src > default > ReviewTemplateCreation.mxml and change the styleName attribute in <mx:Application> tag to another name. For example, change application to applicationSampleCompany:

```xml
<mx:Application
    layout="absolute"
    xmlns:mx="http://www.adobe.com/2006/mxml"
    xmlns:common="com.adobe.solutions.rca.components.common.*"
    xmlns:template="com.adobe.solutions.rca.view.template.*"
    xmlns:definition="com.adobe.solutions.rca.view.definition.*"
    preinitialize="preinit()"
    styleName="applicationSampleCompany">
```

2 In the rca_styles project, edit the src > default > styles.css and create a copy the CSS class you changed in step 1 and then change the attributes to meet your styling requirements.
For example, change the `background-color` attribute for the review template creation Flex application.

```css
.application
{
  font-family: Arial;
  font-size: 12pt;
  color: #434343;
  padding-bottom: 0;
  padding-left: 0;
  padding-right: 0;
  padding-top: 0;
  background-color: #ffffff;
  backgroundGradientAlphas: 1.0, 1.0;
  backgroundGradientColors: #ffffff, #ffffff;
}

.applicationSampleCompany
{
  font-family: Arial;
  font-size: 12pt;
  color: #434343;
  padding-bottom: 0;
  padding-left: 0;
  padding-right: 0;
  padding-top: 0;
  background-color: #ffebfb;
  backgroundGradientAlphas: 1.0, 1.0;
  backgroundGradientColors: #ffebfb, #ffebfb;
}
```

3  Save the files.

4  Build and deploy the EAR file. (See “Building and deploying the solution template” on page 17.)

After you complete the steps, the background for the Review Template Creation screen looks like the following illustration:
Changing the theme

Customized gradient in the Review Template Creation user interface.

Reskin the buttons in all Flex applications

You can customize and reskin the buttons used in the solution template. When you add new images, it is necessary to specify the location of the files in the config.xml file.

1. Copy the new button images to the src > assets > skins folder in the rca_styles project.

   You require an image for each of the button states. For example, copy the following images:
   - LCButton_disabled.png
   - LCButton_downSkin.png
   - LCButton_overSkin.png
   - LCButton_selectedSkin.png
   - LCButton_upSkin.png

2. In the rca_styles project, edit the config.xml file and add names and path to the images that you copied to rca_styles project in step 1. The config.xml file uses the following XML element structure:

   `<flex-config>`<include-file><name> [name of the asset] </name><path>[path to the asset in the project]</path></include-file></flex-config>

   For example, add the bolded entries to represent the files you added in step 1.

Last updated 3/3/2011
3 Edit the src > default > components.css file and change Button attributes to use your custom skins.

```css
.Button {
    disabledSkin: Embed(source="assets/skins/LCButton_disabled.png", scaleGridRight="48", scaleGridLeft="4", scaleGridTop="1", scaleGridBottom="20");
    downSkin: Embed(source="assets/skins/LCButton_downSkin.png", scaleGridRight="48", scaleGridLeft="4", scaleGridTop="1", scaleGridBottom="20");
    overSkin: Embed(source="assets/skins/LCButton_overSkin.png", scaleGridRight="48", scaleGridLeft="4", scaleGridTop="1", scaleGridBottom="20");
    upSkin: Embed(source="assets/skins/LCButton_upSkin.png", scaleGridRight="48", scaleGridLeft="4", scaleGridTop="1", scaleGridBottom="20");
    color: #434343;
    font-size: 11;
    font-weight: normal;
    padding-Top: 5;
    padding-Left: 4;
    padding-Right: 4;
    disabled: Embed(source="assets/skins/LCButton_disabled.png");
}
```

4 Save the files.
5 Complete the following steps to permit Flash Builder to resolve the files you added.
   • In the Package Explorer view, right-click the rca_styles and select Properties.
   • Select Flex Library Build Path.
   • Select the Assets Tab.
   • Click Select All and then click OK.

6 Build and deploy the EAR file. (See “Building and deploying the solution template” on page 17.)

After you complete the steps, the buttons look like the following illustration:

Customized buttons in the solution template. (Small LiveCycle icon in the upper-left corner of the button.)
Chapter 5: Customizing the flow of the initiate review screens

You can change the screens that are used to initiate a review in the GlobalCorp solution template. As part of the solution template, a Flex application provides the Definition, Stages, Supporting Documents, and Documents functionality.

You can alter the order, add, or remove screens to change the flow of the screens displayed to a user. The user interface screens are defined in the rca project in separate MXML files under src > com.adobe.solutions.rca > components > initiation folder. Add additional MXML or AS files to add a new user interface to the review initiation Flex application.

For example, your organization initiates reviews and never adds supporting documents to a review. You can modify the source code to exclude the Add Supporting Documents screen:

**Note:** Deploy the SWF file that is created from review_initiation project using Workbench to change the Review Initiation Flex application that is displayed in LiveCycle Workspace 9. The SWF file to replace is found in the ManagedReviewAndApproval application under the resources > swf folder. (See Working with assets in LiveCycle Workbench 9.5 Help.)

Removing a screen from the review initiation Flex application

You can add, remove, and change the flow of screens in the ReviewWorkflow.mxml file located in the review_initiation project. In the steps below, you modify the WorkflowNavigation.mxml file to prevent the user from accessing the Add Supporting Documents screen.
Despite removing the Add Supporting Documents screen from the flow, the Add Supporting Documents label still appears in the leftmost navigation pane. Therefore, it is also necessary to remove the label from the navigation pane.

1. In Flash Builder, in the review_initiation project, navigate to src > com.adobe.solutions.rca > view > workflow, and open the ReviewWorkflow.mxml file for editing.

2. Change the `<initiation:StageTimelines>nextStep` attribute to register for the events from the `initiation:ReviewDocumentsEvent` component:

   `<initiation:StageTimelines
     id="timelines"
     reviewContext="{ initiationController.currentReview }">
     previousStep="{ StaticEventDispatcher.dispatch( new ReviewDefinitionEvent() ) ">
     addParticipants="{ StaticEventDispatcher.dispatch( new StageParticipantsEvent() ) }"
     nextStep="{ StaticEventDispatcher.dispatch( new InitiationReviewDocumentsEvent() ) }"
     cancel="cancel()" />

3. Change the `<initiation:ReviewDocuments>previousStep` attribute to register for the `StageInitiationEvent` component events:

   `<initiation:ReviewDocuments
     id="reviewDocuments"
     review="{ initiationController.currentReview }">
     previousStep="{ StaticEventDispatcher.dispatch( new StageInitiationEvent() ) }"
     cancel="cancel()" />

4. Open the WorkflowNavigation.mxml file for editing.

5. Locate the `workflow:WorkflowStep component that has the key, navigation.reviewSupportingDocuments and comment it out:

   `<workflow:WorkflowStep
     id="participants"
     stepName="@Resource(bundle='resources',
     key='navigation.reviewParticipants')"
     eventNameToTrigger="{ StageParticipantsEvent.VIEW }"
     width="100%"
     stepLabelWidth="159"/>

   <!--
   <workflow:WorkflowStep
     stepName="@Resource(bundle='resources',
     key='navigation.reviewSupportingDocuments')"
     eventNameToTrigger="{ InitiationSupportingDocumentsEvent.VIEW }" />
   -->

   `<workflow:WorkflowStep
     stepName="@Resource(bundle='resources',
     key='navigation.reviewDocuments')"
     eventNameToTrigger="{ InitiationReviewDocumentsEvent.VIEW }" />

6. Save your changes, build, and deploy the EAR file. (See “Building and deploying the solution template” on page 17.)

After you complete the steps, your Review Initiation Flex application looks like the following illustration:
Custom Initiate Review Flex application without the Add Supporting Documents screen.