# MAX 2009

## LiveCycle Developer Lab: *Advanced Process Design using Adobe LiveCycle Workbench*


# Advanced Process Design using Adobe LiveCycle Workbench

**Jasmin Charbonneau**

**Senior Enterprise Solution Specialist**

**Adobe Systems, Inc.**

# TABLE OF CONTENTS

**EXERCISE OVERVIEW ..... 3**

**EXERCISE 1: CREATE A MULTI-STEP APPROVAL PROCESS USING A FLEX FORM ..... 4**

OBJECTIVES: ..... 4
ASSETS PROVIDED: ..... 4
*Task 1: Create a new LiveCycle application structure ..... 5*
*Task 2: Create a new LiveCycle process ..... 6*
*Task 3: Populate the Flex form with Workspace user information ..... 8*
*Task 4: Populate the Flex form with information from a LiveCycle process ..... 10*
*Task 5: Send the Flex form to multiple approvals ..... 12*
*Task 6: Completing the approval process ..... 14*

**EXERCISE 2: UNDERSTAND AND USE EVENTS ..... 18**

OBJECTIVES: ..... 18
ASSETS PROVIDED: ..... 18
ADDITIONAL RESOURCES ..... 18
*Task 1: Create a process that captures the TaskCreated event ..... 19*

**EXERCISE 3: UNDERSTAND AND USE EXCEPTIONS ..... 21**

OBJECTIVES: ..... 21
ASSETS PROVIDED: ..... 21
*Task 1: Configure a fault route on an operation ..... 22*
*Task 2: Make use of the exception event in a process ..... 23*

**EXERCISE 4: UNDERSTAND AND USE CUSTOM COMPONENTS ..... 25**

OBJECTIVES: ..... 25
ASSETS PROVIDED: ..... 25
ADDITIONAL RESOURCES ..... 25
*Task 1: Create the custom component ..... 26*
*Task 2: Install and use the custom component in Workbench ..... 28*
*Task 3: Inspect the TaskUtils custom component ..... 29*

## Exercise Overview

| Exercise | Summary | Estimated Time |
|---|---|---|
| 1 | Create a multi-step approval process using a Flex form | 35 mins |
| 2 | Understand and use events | 15 mins |
| 3 | Understand and use exceptions | 15 mins |
| 4 | Understand and use custom components | 10 mins |

Each exercise has the following three sections:

1. *Objective*: States the objective of the exercise.
2. *Assets Provided*: Lists any assets that are provided to you for use in development.
3. *Tasks:* List of specific instructions for participants to follow.

## Exercise 1: Create a multi-step approval process using a Flex form

In this exercise you will create a multiple-step approval process using a Flex form and a PDF form. You will also populate the Flex form with information from the Workspace container and from a database. You will then use the new User 2.0 service to send the form to multiple users in parallel.

### *Objectives:*

After completing this exercise, you should be able to:

☐ Create an approval workflow using a Flex form.
☐ Populate the Flex form with information.
☐ Use the Assign Multiple Tasks operation.
☐ Use a custom workspace container to approve the request.
☐ Call a subprocess to create an audit file.
☐ Call a subprocess to archive the document.

### *Assets Provided:*

1. Database table
    a. max_userinfo table in the Adobe dataspace
    b. Test data to prefill the form.
2. HR_Request.swf
    a. MAX_Advanced_Process_Design _Assets /1.0/Forms
    b. Flex form used in the approval process..
3. EmployeeInputForm.pdf
    a. /MAX_Advanced_Process_Design _Assets/1.0/Forms/
    b. PDF form used in the approval process.
4. PrepopulateFlexForm
    a. /MAX_Advanced_Process_Design_Assets /1.0/Processes/
    b. Process used to prefill the form.
5. ArchiveDocument
    a. /MAX_Advanced_Process_Design_Assets /1.0/Processes/
    b. Process used to archive the pdf.

## Task 1: Create a new LiveCycle application structure

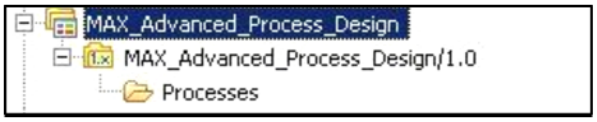
In this task, you will create a new LiveCycle application and the appropriate structure.

1. In Workbench, select File->New-> Application to display the wizard selection screen.
2. Set the Application Name to **MAX_Advanced_Process_Design** and click **Finish**.

> **Note:** This will create a new application in the Application view of the Process Design perspective.

3. Expand the **MAX_Advanced_Process_Design** application, right-click on **MAX_Advanced_Process_Design /1.0** and select **New -> Folder**.
4. Add a folder called **Processes**.

   At this point we have a basic application structure that contains a folder in which to place some assets.



## Task 2: Create a new LiveCycle process

In this task, you will create a new LiveCycle process that uses a Flex form using the new process wizard. You will also be able to initiate the process from the Workspace interface.

1. Right-click on the **Processes** folder and select **New -> Processes...** to display the wizard selection screen.
2. Set the name to **Approval_Process** and click **Next**.
3. On the **Configure a Start Point** screen select **When a user submits a task in Workspace** and click **Next**.

> **Note:** This allows us to initiate the approval process using Workspace.

4. On the **Workspace Start point Configuration,** select **Use and Existing Form.** Select **HR_Request** from **MAX_ Advanced_Process_Design_Assets/1.0/Forms**. Click **Next**.
5. In the **Workspace Category** section, select **Create a New Workspace Category** and enter **New Hire**.
6. Set the **Workspace Process Name** to **Initiate a New Hire Request**.
7. In the **Description** section enter the following text: Initiate a new Request.

> **Note:** The description and the Workspace process name will show up in Workspace for initiating a new process.

8. Click the **Next** button, verify the configuration summary, and click **Finish**.
   This wizard will create a new LiveCycle process with a Workspace endpoint already configured.

9. In the Application tab, right-click on **MAX_Advanced_Process_Design /1.0 and** select **Deploy.** Select **Check in all files** and click **Ok.**

> **Note:** This will check in and deploy the application to the server. At this point we can open the form from Workspace and submit it to LiveCycle.


**Try It!**

10. Go to the Workspace interface at http://localhost:8080/workspace and log in as **kbowman/password**.
11. Select **Start Process** and click on the **New Hire** category. This is the category that was created by the wizard on step 5.
12. Click on the card. This will load the Flex form that was configured on the Workspace endpoint.
13. Enter some information in the comment section and select **Complete**.

    This will submit the form back to LiveCycle for further processing.

## Task 3: Populate the Flex form with Workspace user information

In this task, you will populate the Flex form with information from the user that's currently logged into Workspace.

You will also look at the structure of the Flex form to understand which object is responsible for enabling the interaction with the Workspace container.


1. Start **Adobe Flash Builder** using the shortcut on the desktop.
2. Expand the project called **MAX_ HR_Request**.
3. Expand the **src** folder and open the file called **HR_Request.mxml**.
4. Go to line 146 and check the **lc:SwfConnector** object.

   <lc:SwfConnector id="connector" formSubmitDataRequest="submitFlexApp(event)" setWorkspaceData="returnWorkspaceSession(event)"/>

> **Note:** This object is responsible for the communication between the Flex form and Workspace.
>
> Before the form loads, the **setWorkspaceData** event is dispatched, which calls the **returnWorkspaceSession** function.
>
> When the user clicks on the **Complete** button, the **formSubmitDataRequest** event is dispatched, which in turn calls the **submitFlexApp** function.

5. Go to line 50 and check the **returnWorkspaceSession** function.

   This function will populate the Flex form.
6. Go to line 61. The **connector.getAuthenticatedUser()** function returns a User object that contains information about the user that's currently logged to Workspace.
7. Uncomment lines 64 to 70.
8. Save the file.
9. In the **MAX_HR_Request/bin-debug** folder, right-click on **HR_Request.swf** and select **Copy**.
10. In Workbench, right-click on **MAX_Advanced_Process_Design _Assets /1.0/Forms** and select **Paste** to overwrite the form.
11. Select **Yes** to overwrite the form and **Yes** to check out the asset.

12. Right-click on **MAX_Advanced_Process_Design_Assets / 1.0** and select **Deploy** to deploy the new changes to the server.

**Try It!**

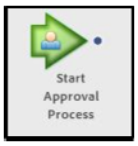
14. Go to the Workspace interface at http://localhost:8080/workspace and log in as **kbowman/password**.
15. Select **Start Process**, click on the **New Hire** category and open the form.
16. Go to the **LiveCycle Related Information** tab on the Flex form to check the Workspace user information.
17. Enter some information in the comment section and select **Complete**.

    This will complete the process.

## Task 4: Populate the Flex form with information from a LiveCycle process

In this task, you will populate the Flex form with information coming from a LiveCycle process. This process will get the data from a database.

1. Select the **Workspace** start point in **Workbench**.




2. On the **Process Properties** view, select **Prepop** in the Action Profile drop-down under **Presentation & Data.**
3. Click on the **Manage Action Profile** button  to check the settings of the **Prepop** action profile.
4. Select **Prepop** from the **Action Profiles** list.
5. The process being called for this action profile is specified in the **Prepare Data Process** and is called **PrepopulateFlexForm**.
6. Select **Cancel** and open the **PrepopulateFlexForm** process from **/MAX_Advanced_Process_Design_Assets /1.0/Processes/**

   This process is a simple process that makes a database call and returns an XML file following this structure:

```
<form1>
  <subEmp_and_Header>
    <fldSSN>11111111111</fldSSN>
    <rdoGender>M</rdoGender>
    <fldFName>Tony</fldFName>
    <fldLName>Blue</fldLName>
    <fldDOB>11/09/1980</fldDOB>
  </subEmp_and_Header>
  <subContactInfo>
    <fldAddress1>343 Main Street</fldAddress1>
    <fldAddress2>Ottawa</fldAddress2>
    <fldAddress3>12345</fldAddress3>
    <fldCPhone>8888888888</fldCPhone>
```

```
        <fldHPhone>9999999999</fldHPhone>
    </subContactInfo>
</form1>
```

> **Note:** This XML data will be available as part of the **setWorkspaceData** event, which calls the **returnWorkspaceSession**. You can extract the XML using the following code:
>
> ```
> var xmldata:XML = new XML(event.data);
> ```

7. Go back to the **Workspace** start point. Check the **Permit adding attachments** checkbox in the **Attachment Options** section.
8. Save and deploy.

**Try It!**

9. Go to the **Workspace** interface at http://localhost:8080/workspace and log in as **kbowman/password**.
10. Select **Start Process** and click on the **New Hire** category.
11. Click on the card.

    The information on the **Employee Information** and **Employee Contact Information** tabs comes from the **PrepopulateFlexForm** process.

    The information on the **LiveCycle Related Information** tab comes from the user **connector.getAuthenticatedUser()** object within the Flex form.
12. Enter some information in the comment section and select **Complete**.

    This will complete the process.

## Task 5: Send the Flex form to multiple approvals

In this task, you will expand the approval process and use the new Assign Multiple Task operation to send the form to multiple users in parallel. You will also use a new Workspace container to display the Flex form.

1. In Workbench, add a multiple-user step by dragging the **Assign Multiple Tasks** icon  from the toolbar to the **Approval_Process** process.
2. Set the name to **Send to Multiple Users**.
3. In the **Participant** section in the **Process Properties** view, click the + button, select **User** and add the following users who will receive the form to review:
    - Akira Tanaka
    - Heather Douglas
4. In the **Task Instruction** section, enter the following text: "Please review this form".

   The instruction will appear on the card in the **To Do** section of **Workspace**.
5. In the **Workspace User Interface**, select **Approval Container.**

> **Note:** This new feature will allow you to use a different interface within Workspace instead of the default interface. In the approval container, the form will open full screen, and a comment section is added at the bottom of the container.

6. In the **Presentation & Data** section, select **use an application asset** and select **/MAX_Advanced_Process_Design _Assets/1.0/Forms/HR_Request** as the asset.
7. Set the **Initial Task Data** to **formdata**.

   This will allow the data from the initiator to be passed to the users in this step.
8. In the **Output** section, create a new **TaskResultCollection** variable called **TaskResultColl** using the  button.

**Note:** Assign Multiple Tasks operations create several tasks. Therefore, the Task Result value from each task is saved in a special collection called Task Result Collection. The Task Result value that is submitted with each task is appended to the Task Result Collection value.

The Workspace ES2 approval container displays information that is stored in the Task Result Collection. Any comments and user action information that is stored in the Task Result Collection is displayed in the approval container.

The task information from the single user step can also be appended to the Task Result Collection.

**Try It!**

1. Go to the **Workspace** interface at http://localhost:8080/workspace and log in as **kbowman/password**.
2. Select **Start Process** and click on the **New Hire** category.
3. Click on the card to open the form.
4. Enter some information in the form and select **Complete**.

   Now the form will go to the **Send to Multiple Users** step.
5. Go to the **Workspace** and log in as **atanaka/password**.
6. Click on **To Do** to open the item.
7. Click on **Approve**.
8. Go to the **Workspace** and log in as **hdouglas/password**.
9. Click on **To Do** to open the item.
10. Click on **Approve**.

## Task 6: Completing the approval process

In this task, you are going to complete the approval process by sending the form to an archival process if all the users approve the form. Otherwise the form will go to an HR representative in a PDF format.

1. Add a single-user step by dragging the **Assign Task** icon  from the toolbar.
2. In the **General** section, set the name to **HR Rep.**
3. In the **Initial User Selection** section, select **Assign to specific user** and select **Sarah Rose**.
4. In the **Task Instruction** section, enter the following text: "Please verify the following request."
5. In the **Presentation & Data** section, select **use an application asset** and select **/MAX_Advanced_Process_Design _Assets/1.0/Forms/Employee_Input_Form** as the asset.
6. Set the **Initial Task Data** to **formdata**.
7. Check the **Submit via Reader As : XDP** in the **Reader submit** section.

> **Note:** This will allow the PDF to be submitted from Workspace without the need to add a submit button on the form. This will only work with Adobe Reader 9.1.

8. In the **Output** section, set the **TaskResultCollection** to **TaskResultColl**
9. Add a new subprocess by dragging the subprocess icon  from the toolbar. Select the **ArchiveDocument** process from the **MAX_Advanced_Process_Design _Assets** application within the **Deployed Process** category.

> **Tip:** You can type "Archive" in the Find section to narrow down the search.
> This process will create a PDF and store it in Content Space.

10. Create a new variable of type document called **generatedPDF**.
11. In the **Process Properties** tab of the **ArchiveDocument** operation, set the **Form Data** property to **formdata**.
12. In the **Output** section, set the **renderedForm** property to **generatedPDF**.
13. Add a subprocess by dragging the subprocess icon  from the toolbar. Select the **AuditTaskInfo** process from the **MAX_Advanced_Process_Design _Assets** application within the **Deployed Process** category.

> **Tip:** You can type "Audit" in the Find section to narrow down the search.
> This process will loop though all the Task Result values in the TaskResultCollection variable to get the name, the action and the comments of all the users involved in the process.

14. Set the name to **Create Audit Log.**
15. Set **objTaskCollection** to variable -> **TaskResultColl**.
16. Create two routes out of the **Send to Multiple Users** step with the following settings:

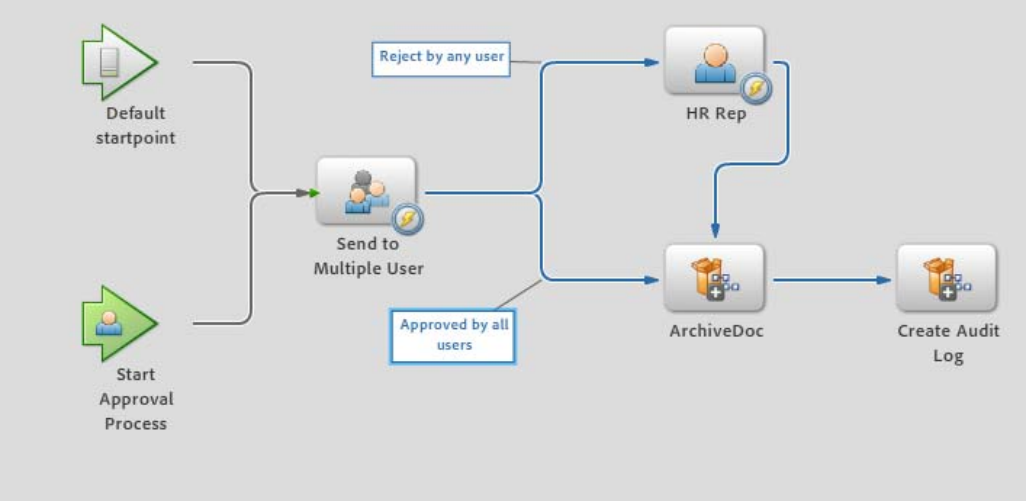
| Name | Destination |
|---|---|
| Reject | HR Rep |
| Approve | ArchiveDocument |

17. Create a route from the **ArchiveDocument** step to the **Create Audit Log** step.
18. Create a route from the **HR Rep** step to the **ArchiveDocument** step.
19. On the **Send to Multiple Users** step, in the **User Action** section, add two custom action names: **Approve** and R**eject.**

    Those are the actions the user will see in Workspace for submitting the form.

20. In the bottom section of the **User Action** section check the **Use Completion Policies** checkbox and add the following completion policies using the + button:
    - Go to HR Rep when at least 1 user(s) pick the Reject action.
    - Go to ArchiveDocument when exactly 100 % pick the Approve action.

21. The process should look like the following image:




22. Save and deploy.

**Try It!**

23. Go to the **Workspace** interface at http://localhost:8080/workspace and log in as **kbowman/password**.
24. Select **Start Process** and click on the **New Hire** category.
25. Click on the card to open the form.
26. Enter some information in the form and select **Complete**.
    Now the form will go to the **Send to Multiple Users** step.
27. Go to the **Workspace** and log in as **atanaka/password**.
28. Click on **To Do** to open the item.
29. Click on the **Enter Comments** button  to enter some comments.
30. Click on **Approve**.
31. Go to the **Workspace** and log in as **hdouglas/password**.
32. Click on **To Do** to open the item.
33. Click on the **Enter Comments** button  to enter some comments.
34. Click on **Approve**.

Since the two users approved the form, it will call the ArchiveDocument sub process which will create a PDF and store it in Content Space. It will also call the Create Audit Log process which will create an audit log.

35. Go to content space at http://localhost:8080/contentspace/faces/jsp/login.jsp.
36. Login as **Administrator/password.**

> **Note:** If you go http://localhost:8080/contentspace instead of the login page, content space will automatically log the user as **Guest** without asking for any credentials.

37. Go under **Company Home/Guest Home** and verify that the PDF has been created.
38. Verify the audit file at **C:\MAX2009\Advanced_Process_Design\Audit.txt**
39. Initiate another form, but have **hdouglas** reject the form.
40. Go to **Workspace** and log in as **srose/password**.
41. Click on the card to open the form.

> **Note:** The form is a PDF form instead of a Flex form. The PDF form doesn't contain a submit button, but can still be submitted. The new version of Reader takes care of submitting the PDF without the need of an actual submit button on the form. This was enabled by the **Submit via Reader** option on the user step.

42. Click on **Complete** to submit the form.

## Exercise 2: Understand and use events

In this exercise, you will modify an existing process so that it only starts when a new task is assigned by the Assign Multiple Tasks operation.

### *Objectives:*

After completing this exercise, you should be able to:

☐ Handle an event thrown by the User service.

### *Assets Provided:*

1. SendEmailNotification
    a. /MAX_Advanced_Process_Design_Assets /1.0/Processes/
    b. Process that sends email notification to all users of an Assign Multiple Task operation.

2. Akira Tanaka's mailbox
    a. Thunderbird mail client
    b. Mail account to receive email notifications.

### *Additional Resources*

☐ http://livedocs.adobe.com/livecycle/8.2/wb_help/000926.html

## Task 1: Create a process that captures the TaskCreated event

In this task, you will modify an existing process so that it start when a TaskCreated event occurs. You will also configure a filter so that it reacts only to events generated by a specific step. Finally you will configure the event to populate process variables.

1. Open the process called **SendEmailNotification** from **/MAX_Advanced_Process_Design_Assets /1.0/Processes/**.

2. Add a **TaskCreated** event type by dragging the event picker from the toolbar  , select the **TaskCreated** type from the **Asynchronous** category and click OK.

> **Tip:** You can type "Task" in the Find section to narrow down the search.

3. Select **Start Point** for the event behavior.

   The **Filter** tab allows the event start point to react to only specific events that match particular filters.

   The **Callback – Process Data Map** tab allows the mapping of event related information into local process variables.

4. In the **Filter** tab, set the following data mapping:

| Event Data Name | Event Data Value |
|---|---|
| /MapContent/StepName | Send to Multiple Users<br>**Note**: This should match the name defined on the Assign Multiple Task operation (Exercise 1, Task 5, step 2). |

   This will only react to the tasks that are created at the **Send to Multiple User** step of the **Approval_Process**.

5. In the **Callback – Process Data Map** tab set the following data mapping:

| Process Data | Event Content |
|---|---|
| /process_data/@actionid | /MapContent/ActionID |
| /process_data/@processinstanceid | /MapContent/ProcessInstanceID |
| /process_data/@assigneduser | /MapContent/AssignedUser |

**Note:** The start point automatically connects to the start activity, which in this case is a custom component that returns the list of all members that are part of the multiple user step in the Approval_Process.

The source code for that custom component is located in Eclipse under the TaskUtils project.

6. Click **OK**.
7. Save and deploy.
8. Go to the **Workspace** interface at http://localhost:8080/workspace and log in as **kbowman/password**.
9. Select **Start Process** and click on the **New Hire** category.
10. Click on the card to open the form.
11. Enter some information in the form and select **Complete.**

    This will send the form to the **Send to Multiple Users** step and generate a **TaskCreated** event for each member configured on that step.
12. Open **Thunderbird** mail client using the icon on the desktop.
13. Check Akira Tanaka's mailbox to see if the **SendEmailNotification** was executed properly.

    You might need to click on the **Get Mail** button to refresh the Inbox.

## Exercise 3: Understand and use exceptions

In this exercise, you will modify the approval process to handle specific exceptions. You will also modify an existing process to react to all exceptions thrown by any LiveCycle processes.

### *Objectives:*

After completing this exercise, you should be able to:

☐ Catch an exception using the exception handler of a particular service.
☐ Handle an exception using the exception event.

### *Assets Provided:*

1. ExceptionHandling
    a. /MAX_Advanced_Process_Design_Assets /1.0/Processes
    b. Process to handle exceptions.

## Task 1: Configure a fault route on an operation

In this task, you will configure a fault route from an operation to handle a specific exception.

> **Note:** Some operations in LiveCycle have exception handling built-in, and catch an exception throw event. You can identify operations that implement this by the lightning bolt in a circle placed at the bottom right corner of the operation. 
>
> To make use of the exception handling built into the operation, you click and drag from the event catch (lightning bolt) to another operation on the process.

1. In the Approval Process, add an **Email with Document** operation by dragging the appropriate icon from the toolbar  .
2. Set the **To** in the **To Addresses** section to **atanaka@aquo.com.**
3. Set the **From** to administrator@aquo.com.
4. Set the **Subject** in the **Contents** section to "Invalid Principal Error".
5. Click and drag from the **Multiple User Tasks** event (lightning bolt) to the **Email With Document** event.
6. Select **InvalidPrincipal** for the Exception type.

**Generate an exception!**

Let's generate an exception that can be caught by the exception handler.

7. Select the **Multiple User Tasks** step in the main process.
8. In the Participants section, click the + button to add a new user and select **xPath Expression**.
9. Enter a value that won't resolve to a valid user (ex. 'Test User').
10. Save and deploy the process.
11. Initiate the process using the Workspace interface. The **Multiple User Tasks** operation should generate an InvalidPrincipal exception for the invalid user and follow the route from the lightning bolt.
12. Verify by using Record and Playback if the right route is being followed. An email to atanaka@aquo.com should also be sent.

## Task 2: Make use of the exception event in a process

In this task, you will modify an existing process to handle all exceptions generated by any LiveCycle process.

> **Note:** LiveCycle provide an exception event handler that can handle all exceptions that are generated by a LiveCycle process. By implementing the exception event as a start point in its own process, you can create a generic exception handler without adding multiple steps to many different processes.

1. Open the ExceptionHandling process from **/MAX_Advanced_Process_Design_Assets /1.0/Processes.**
2. Add an **Exception** event type by dragging the event picker from the toolbar  and select the **Exception** operation from the **Exception** category and click OK.

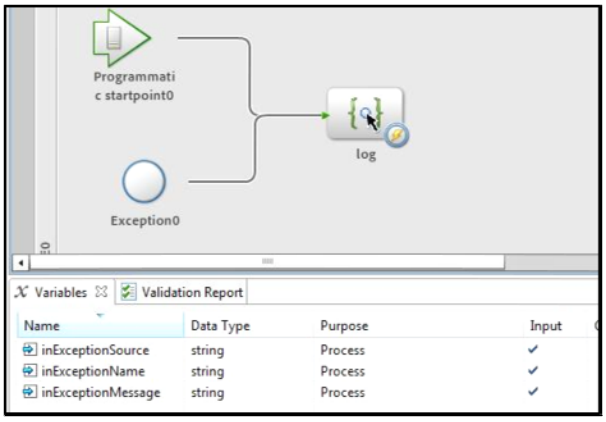
> **Tip:** You can type "Exception" in the Find section to narrow down the search.

3. Select **Start Point** so it initiates the new process.
4. Set the name to **Catch All Exceptions**.
5. Leave the **Filter** tab blank to catch all exceptions.
6. On the **Callback – Process Data Map** tab, set the following data mapping:

| Process Data | Event Content |
|---|---|
| /process_data/@ inExceptionName | /Exception/FaultName |
| /process_data/@ inExceptionSource | /Exception/FaultSource |
| /process_data/@ inExceptionMessage | /Exception/FaultMessage |

This will put the content of the exception in local process variables.

7. Your process should be similar to the following image:




**Now, let's catch the exception using the generic exception handler**

8. Delete the route that goes from the lightning bolt to the **Send Email** step.

> **Note:** The route needs to be deleted to prevent the exception to be dealt with. If the exception is not dealt with then an exception event is thrown.

9. Save and deploy both processes.
10. Make sure to record the **ExceptionHandler** process.

> **Tip:** You will see a little red spot on the process icon if the process is being recorded.

11. Initiate the Approval process using **Workspace**.
12. Play the recording from the **ExceptionHandler** process and inspect the content of the input variables.

## Exercise 4: Understand and use custom components

In this exercise, you will modify an existing Eclipse project to create a new custom component. You will also compile and deploy the new component to LiveCycle. You will also use that new custom component in an existing process.

### *Objectives:*

After completing this exercise, you should be able to:

☐ Understand the basic structure of a component.
☐ Modify and compile an existing custom component.
☐ Install and use a custom component.

### *Assets Provided:*

1. MAX_CustomComponent
    a. Eclipse project
    b. A sample custom component.

### *Additional Resources*

☐ http://livedocs.adobe.com/livecycle/8.2/programLC/programmer/help/000934.html

## Task 1: Create the custom component

In this task, you will modify an existing Eclipse project to create a new custom component. You will also compile the code to create a new LiveCycle component.

1. Start **Eclipse** using the shortcut on the desktop. 

2. Open the project called **MAX_CustomComponent**.
3. Open the **component.xml** file. This is where the metadata information about the component is defined.
4. On line 21, change the category-id to **MAX**.

   This is the category the service will be deployed under Workbench.
5. Open the File_IO.java file from the **src/max/demo** folder.

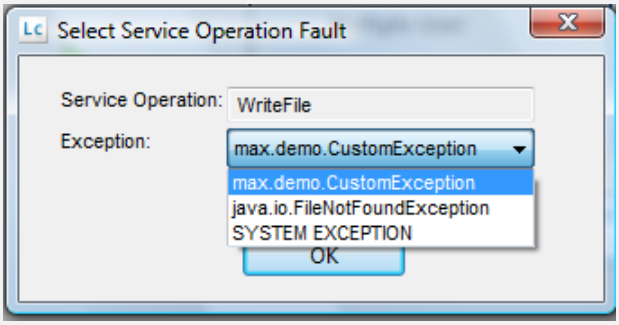
   This is an interface that defines the different methods implemented by the custom component.
6. Open the File_IO_Impl.java file from the **src/max/demo** folder.

> **Notes:** This is where the source of the component is defined.
>
> This component contains two functions: writeFile and ReadFile. These functions are going to be available when using the component in Workbench.
>
> This component can also throw exceptions. The lightning bolt at the bottom right corner of the service icon gets populated from the list of exceptions from the method's signature

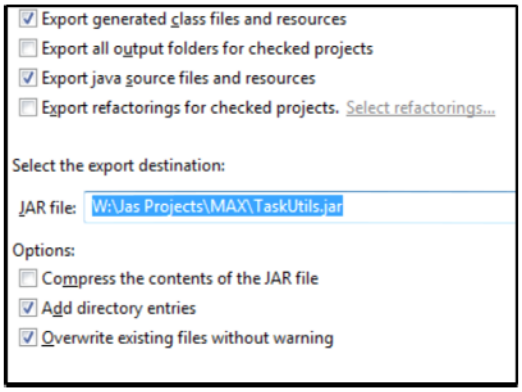
7. The writeFile method can throw a CustomException or FileNotFoundException exceptions.

   public void WriteFile (Document doc,String docName)throws CustomException, FileNotFoundException

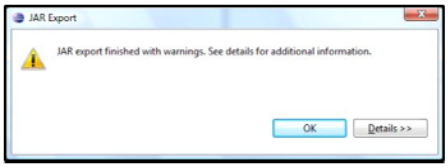
**Note:** These two exceptions will show up in the lightning bolt for that operation along with the more generic SYSTEM_EXCEPTION.




8. Right-click on the project and select **Export...**
9. Select **Jar File** in the **Java** category.
10. Make sure to select the following options:




11. Select **Finish**.
12. Click Ok on the following warning.



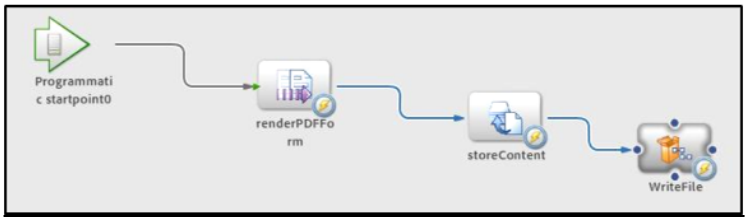
## Task 2: Install and use the custom component in Workbench

In this task, you will install the new custom component in Workbench and use it as part of an existing orchestration.

1. In **Workbench**, make sure the **Components** view is available (Windows>Show View>Components).
2. In the **Components** view, right-click on the **Components** category and select **Install Components...** .
3. Right-click on the **MAX Custom Component** component and select **Start Component**. The component is now ready to use.
4. Open the **ArchiveDocument** process from **/MAX_Advanced_Process_Design_Assets /1.0/Processes/**.
5. Add the custom component by dragging the activity picker from the toolbar  , selecting the **WriteFile** operation from the **File_IO** service within the **MAX** category, and clicking OK.

> **Tip:** You can type "Write" in the Find section to narrow down the search.

6. Add a route from the **Store in Content Space** step to the new step.
7. Set the **Document to write** to **renderedForm** in the **Input** section.
8. Set the **File name** to **output.pdf**.
9. The process should look like the following:




10. Save and deploy.

**Note:** Task 3 is an optional task. If you have completed the previous tasks, and time permits, work through the following task.

## Task 3: Inspect the TaskUtils custom component

In this task, you will inspect the TaskUtils custom component that uses the LiveCycle API to get all the users defined for a particular XXX

1. Start **Eclipse**, using the shortcut on the desktop. 
2. Open the project called MAX_LCTaskUtils.
3. Open the **component.xml** file. This is where the metadata information about the component is defined.
4. Since this component interacts with LiveCycle, we need to make sure we have a reference to the proper client JAR files.

   On line 8 the import-packages section defines which client JAR files will be needed to execute the code.

```
<import-packages>
    <package version="1.0">com.adobe.idp.taskmanager.dsc.client.task</package>
    <package version="1.0">com.adobe.idp.taskmanager.dsc.client.query</package>
    <package version="1.0">com.adobe.idp.taskmanager.dsc.client.queuemanager</package>
    <package version="1.0">com.adobe.idp.taskmanager.dsc.client.endpoint</package>
    <package version="1.0">com.adobe.idp.taskmanager.dsc.client</package>
    <package version="1.0">com.adobe.idp.taskmanager.dsc.client.events</package>
</import-packages>
```

5. The getUserInfo method is defined on line 33-48. The method takes two input parameters and returns one output parameter.
6. Open the file TaskUtilsServiceImpl.java from the src/com/adobe/etech/lc8/service/components folder.
7. This is where the code is implemented.
8. Since we need to query information about a task, we need to create a TaskManagerQueryService object using the following code:

```
_queryManager = TaskManagerClientFactory.getQueryManager(ServiceClientFactory.createInstance());
```

**Note:** The ServiceClientFactory.createInstance() will return the actual security context under which the component is being run within the LiveCycle process.

9. The rest of the code just implements a task filer and perform a task search.
10. The function can throw a Get_User_Exception or Exception exception.

    public List getUserList(String processId, String actionId)throws Get_User_Exception,Exception

**Note:** These two exceptions will show up in the lightning bolt for the operation.