

LiveCycle for Service Orchestration

Introduction

This lab will introduce you to the service orchestration features of LiveCycle ES.

It is built around a fictional use case where travel assistant and travel supervisor staff of a travel agency have to process an e-ticket itinerary and indicate whether the traveller has or has been exposed to swine flu.

The labs are increasing in complexity in terms of service orchestration. In the first lab we will create a simple orchestration that allows a dynamic interactive form to be processed by a travel assistant and then to be rendered as a static final PDF written to the file system. It will illustrate simple user involvement as well as the use of document centric and foundation services in an orchestration.

In the second lab we will extend what we have built in Lab 1 by involving traveler supervisor staff based on what the travel assistant staff has provided as input. Furthermore we will conditionally use database integration by adding records to a database table.

In the third lab we will add even more operations to the orchestration. We will build web service integration, document management integration, exception handling and even incorporate our own Java-built Twitter functionality as a service in LiveCycle.

All the integrations are kept simple just to clearly illustrate the concepts. In between the lab instructions you will get explanations on the functionality and features you are implementing.

The labs follow a clear format.

- Normal paragraphs introduce concepts and explain in detail the rationale behind the instructions.
- List numbered paragraphs are instructions you need to follow.
- **Bold** indicate UI artefacts (menu selections, field labels, dialog titles, etc.).
- Monospace indicate text you have to type in.
- Screenshots are used extensively to introduce new concepts, dialogs, interactions, etc. but are not repeated.

All programs used in the labs are available from the Windows taskbar.


Table of Contents

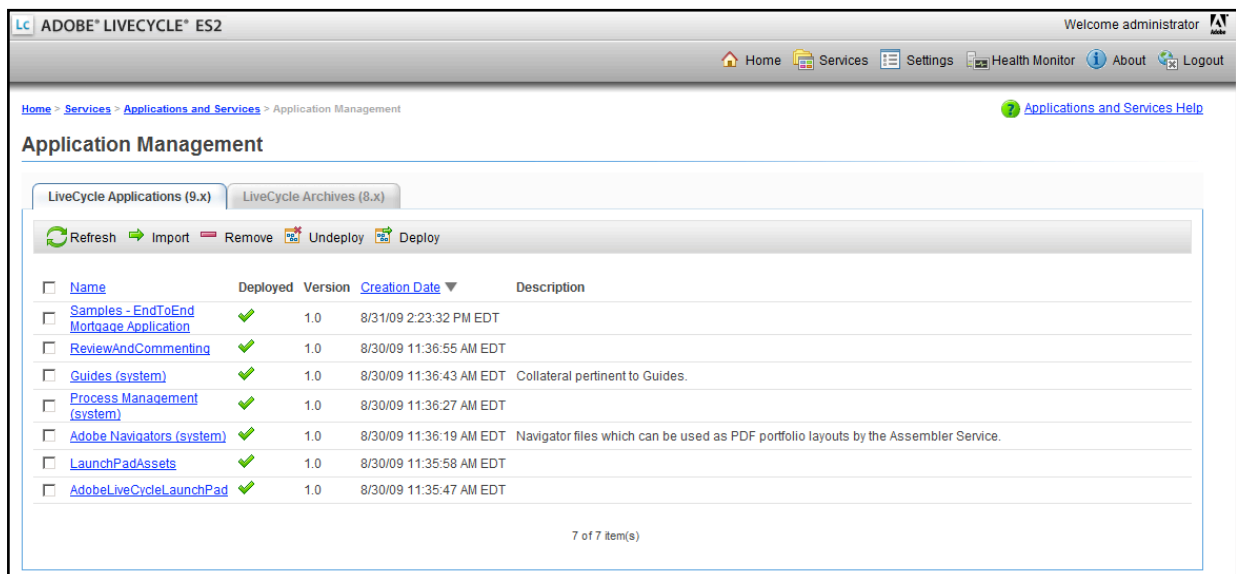
Preparation	5
Lab 1	6
Login to Workbench	6
Create a new application	7
Explore the form	7
Add assets to application	7
Create an orchestration	8
Define Variables	8
Involve travel assistants	9
Generate PDF	11
Connect operations	11
Save PDF	12
Validate	12
Deploy application	13
Invoke application	13
Act as travel assistant	15
Lab 2	17
Map to XML	17
Involve Travel Supervisors	18
Database integration	18
Connect operations	21
Add routing condition	21
Specify evaluation order	22
Test application	23
Lab 3	24
Integrate Content Services as a web service	24
Integrate Content Services natively	26
Add logging	27

Integrate Twitter	28
Handle Exceptions	33
Connect operations	33
Test application	34
Appendix	36
Remove current application	36
Import application	36

Preparation

We first will check whether our environment is up and running correctly before we start our labs.

1. Open **Internet Explorer** by clicking on the Internet Explorer icon  in the taskbar.
2. In **Internet Explorer**, click on **AdminUI** in the Links toolbar
3. Log in to the **LiveCycle Administration Console** with **User ID** administrator and **Password** password.
4. Click on **Services** in the **Administration Console Home** screen
5. Click on **Application and Services** in the **Services** screen
6. Click on **Application Management** in the **Application and Services** screen.
7. You will see the following screen.




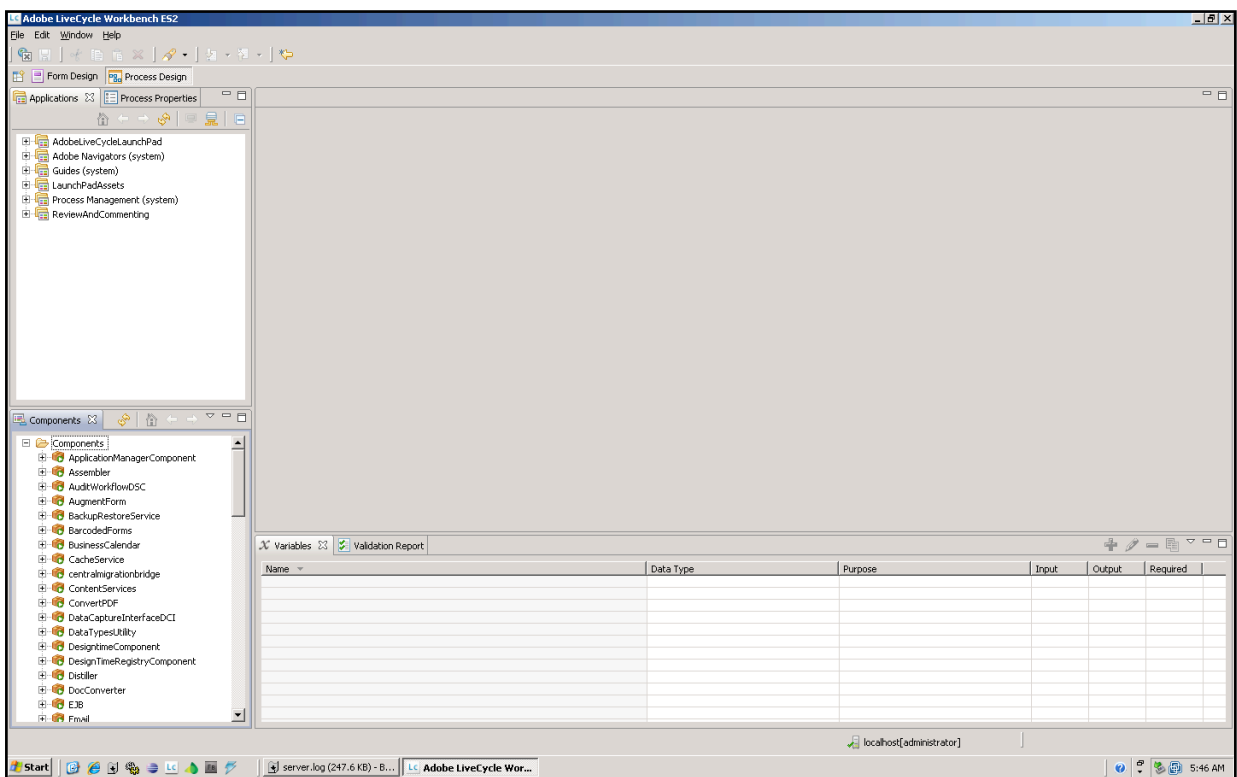
If you have come this far, all is fine. If not contact instructor or lab assistant.

Lab 1

In this lab we are going to create a simple orchestration with user involvement and which will use a document service and foundation service.

Login to Workbench

1. Open **Workbench** by clicking on the Adobe LiveCycle Workbench ES2 icon  in the taskbar.
2. When **Workbench** has opened, click on any of the **Click here to login** links to log in.
3. In the login dialog, provide password as the **Password** and administrator as **User ID**.
4. Click **OK**. **Workbench** will retrieve information from the LiveCycle ES server and display the following window.



In the upper left panel you will see all active applications in the **Applications** panel. There is also a **Process Properties** tab which we will use when building our service orchestrations.

On the lower left side you will see the **Components** panel which we will use at the end of this lab. For now remove the panel by clicking on the **X** next to the **Components** name in the tab.



The biggest part of the Workbench UI is taken up by an empty canvas, but this will change during our lab.

The bigger bottom part of the UI is for defining variables in our orchestrations and validating the orchestrations.

Create a new application

We start by creating a new application in Workbench.

5. Select **File | New > Application** from the **Workbench** menu. In the **New Application** dialog, specify **E-TicketMedical** as the **Application name**, and (optionally) provide a description (e.g. Application for medical approval of e-tickets).
6. Click **Finish** to create the new **E-TicketMedical** application.
7. In the **Application** panel, **E-TicketMedical** should now be available as an application; click on the **+** icon to unfold its structure. You will see **E-TicketMedical/1.0** indicating this is the first version of your application.

Explore the form

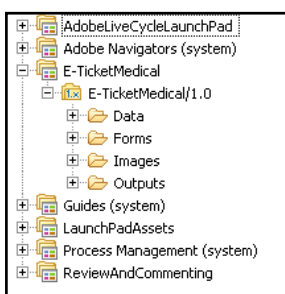
In our lab we are using an interactive dynamic form, based on the E-Ticket.pdf form distributed as part of the LiveCycle ES Designer samples. We will not go into details on this form as it is beyond the scope of this lab, however it is worthwhile to know we extended the form with a swineflu radio button and description field in the Medical Information section. Also, we have set all fields to read-only except for the fields in the Medical Information section.

8. Go to your lab folder in **Windows Explorer**. The lab folder is located at **C:\MAX09L34**. There is a convenient shortcut available on the desktop to get you straight to the content.
9. Open the **LCES\Forms** folder and double click **E-TicketMedical.xdp**. After a while **LiveCycle Designer ES** will open presenting you with the form template we are going to use in this lab.
10. Briefly examine the form. Note that only the Medical Information section is interactive; i.e. the fields are enabled. All other fields are made read-only and hence will not be interactive.
11. Close the form and **LiveCycle Designer ES**. When prompted to save; select **No**. The form is fine as it was when we opened it. And we don't need LiveCycle Designer ES in our labs.

Add assets to application

We are now going to add the form and other assets to the application. These assets are in the **LCES** folder in our lab folder. The **Data** folder contains a sample XML file we will use as input when invoking our orchestration. The **Forms** folder contains our **E-TicketMedical.xdp** form. The **Images** folder contains images used by the E-TicketMedical form. The **Output** folder will contain output from our orchestration.

12. Go to **Windows Explorer** and open your lab folder (**C:\MAX09L34**).
13. Open the **LCES** folder and within this folder open the **Outputs** folder. Delete the two PDF files; they are remnants of an earlier setup.
14. Go back up one level so you see the **Data**, **Forms**, **Images**, **LCA** and **Outputs** directories in Windows Explorer. Select all folders, except the LCA folder and drag the selection over the **E-TicketMedical/1.0** folder in Workbench and drop the selection. The folders will be added as subfolders to the **E-TicketMedical/1.0** folder.



Create an orchestration

Our next step is to create our orchestration for this application.

15. Right click the **E-TicketMedical/1.0** folder, select **New > Process** from the menu.
16. In the **New Process** dialog, enter **MedicalCheck** in the **Name** field and select **Next>**.
17. In the **Configure a Start Point** dialog ensure **Add start points later** is selected and click **Next>**. You will see a **New process Configuration Summary** dialog to confirm your input.



18. Click **Finish** to finish the configuration of your orchestration. Workbench will now display a **MedicalCheck** process tab, displaying the process as it is built/configured sofar. This will be our working canvas for most of this lab.

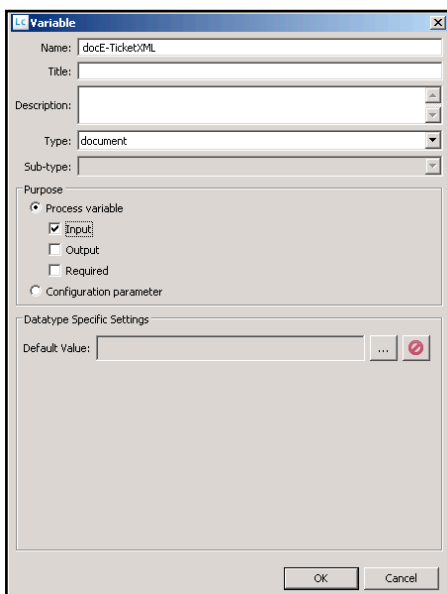
Define Variables

We will have to define some variables we are going to use in our orchestration.

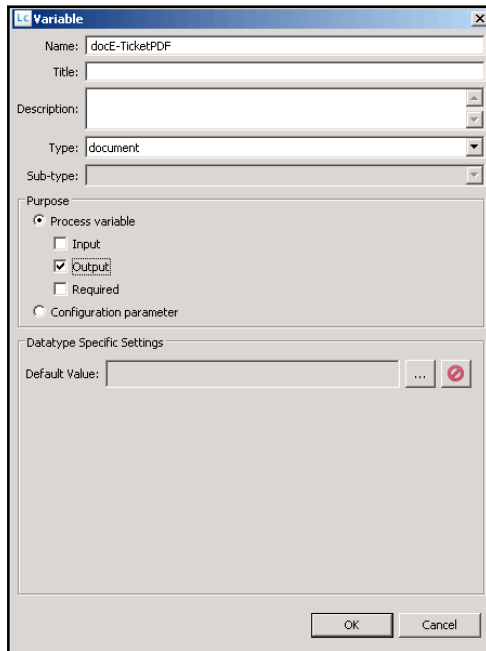
19. In the **Variables** panel, click on the big green + symbol (Create a new variable).



20. In the **Variable** dialog, enter **docE-TicketXML** for the **Name**, select **document** as the **Type**, select **Process Variable** and ensure **Input** is checked.




21. Click **OK**.
22. Create another variable, name it **docE-TicketPDF**, select **document** as the **Type**, select **Process Variable** and ensure **Output** is checked.



23. Click **OK**.

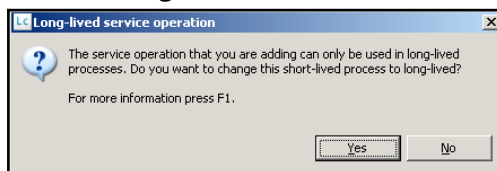
Involve travel assistants

The first thing we want to do in our orchestration is to render the E-TicketMedical form to an end-user so he/she can provide some input. To do this, we will add an Assign Task to the orchestration.

24. Drag the **Assign Task** icon  from the toolbar at the top of the process design canvas and drop it somewhere on the canvas.

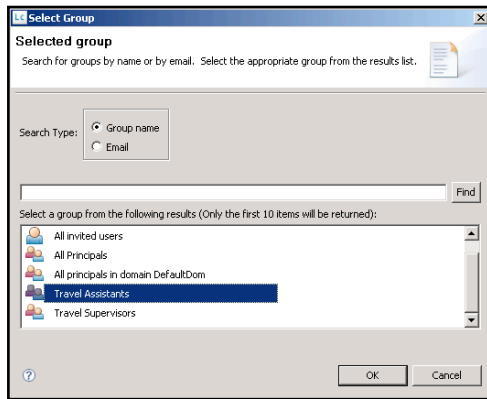


25. You will see a dialog, prompting you whether you want to change the process from short-lived to long-lived.

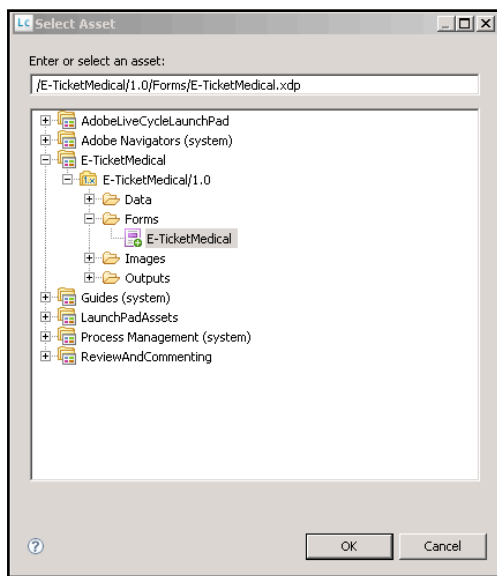


26. Click **Yes** to confirm.
27. The **Assign Task** operation is now automatically connected with the **Default startpoint** and the **Process Properties** panel has become active to further specify the details of the operation.
28. Enter **Medical Check** as **Name** in the **General** properties.
29. Unfold **Initial User Selection**, select **Assign to Group**, ensure **Assign to Group queue** is selected and click **Browse...** In the **Select Group** dialog, set **Search Type** to **Group name**

and select **Travel Assistants** in the list with groups.



30. Click **OK**.
31. (Optionally) Provide some task instructions; they will appear as instructions for the end-user. For example: Please check medical information for this E-Ticket and provide additional details where appropriate... To do this unfold the **Task Instructions** part of the **Process Properties** panel and type the text in the text area.
32. Unfold the **Presentation & Data** part of the **Process Properties** panel. Ensure **use an application asset** is selected and click on the elipsis ... button next to the **Asset** field in the **application asset** box. This will open the **Select Asset** dialog.
33. Within this dialog select the **E-TicketMedical** (the form template) from the **Forms** subfolder of the **E-TicketMedical/1.0** folder.



34. Click **OK**. **E-TicketMedical.xdp** is now defined as the **Asset** and the **Application Profile** is set to **Default**.

In order to render the template form with data, we need to specify initial task data. To do this:


35. Select **docE-TicketXML** from the **Initial Task Data - Variable** dropdown listbox.

There is one more part we need to configure for this operation. We want to capture any changes made by the end-user in the form. To do this:

36. Unfold the **Output** properties of the **Process properties** panel, and select **docE-TicketXML** from the **Output data** dropdown listbox. This will ensure any changes made in the form by the end-user are reflected in an update to the XML we used as input for the form.

Generate PDF

After the user involvement in the process we want to generate a static PDF from the form for later reference. We will use the Generate PDF Output operation of the Output service for that. To do this:

37. Drag the Activity Picker  on the process design canvas.
38. The **Define Activity** dialog will appear. In this dialog, select **Output 1.1 | - generatePDFOutput** from the **<Recently Used>** list, or select **Output - 1.1 | generatePDFOutput** from the **Output** service.

Service	Operation	Description
<Recently Used>		
Convert PDF - 1.1	toPS	Converts a PDF document to a PostScript document.
DocConverter - 1.0	Convert to PDF/A	Converts a PDF document to PDF/A.
DocConverter - 1.0	Validate PDF/A	Validates whether a PDF is a PDF/A.
Document Management - 1.0	createSpace	The operation allows the user to create a space in LiveCyc...
Document Management - 1.0	storeContent	The operation provides a way for storing document/conte...
Encryption - 1.1	Password Encrypt PDF	"Applies password encryption security to an unencrypted ...
Execute Script - 1.0	executeScript	The Script action provides a script editor where you can ty...
File Utilities - 1.0	Write Document	Writes a document to the file-system.
JDBC - 1.0	Execute SQL Statement	Executes SQL statement and saves the number of rows af...
Output - 1.1	generatePDFOutput	Merges a form design with data and generates a PDF doc...
Twitter - 1.2	UpdateTwitter	Doesnt return anything just yet...
User Lookup - 1.0	Find Group	Find a Group given the group search filter.
Variable Logger - 1.0	log	Logs the process variables.
Web Service - 1.0	Invoke Web Service	Invokes the Web Service.

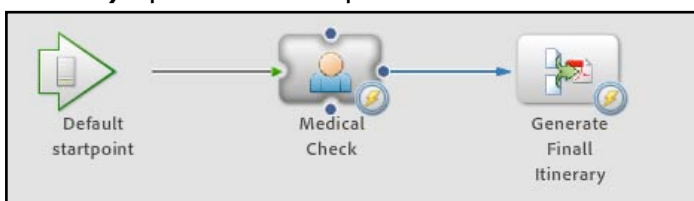
39. Click **OK**.
40. Specify **Generate Final Itinerary** as the name for the operation.
41. Unfold the **Input** part of the **Process Properties** panel and select the elipsis ... button next to the **Form** field.
42. In the **Select Asset**, select **E-TicketMedical** from the **Forms** subfolder of the **E-TicketMedical/1.0** folder. The Form field will now contain URI = Forms/E-TicketMedical.xdp, Created: 10/5/09. (Note that the date might be different)
43. Select **docE-TicketXML** from the **Input Data** dropdown list box.
44. Select **PDF** from the **Transformation Format** dropdown listbox.
45. Unfold the **Output** part of the **Process Properties** panel and select **docE-TicketPDF** from the **PDF Output** dropdown listbox.

We have now specified how to render a template with data into a static PDF using the LiveCycle Output service.

Connect operations

We have setup our operations for our orchestrations, now we we need to link them together. To do this

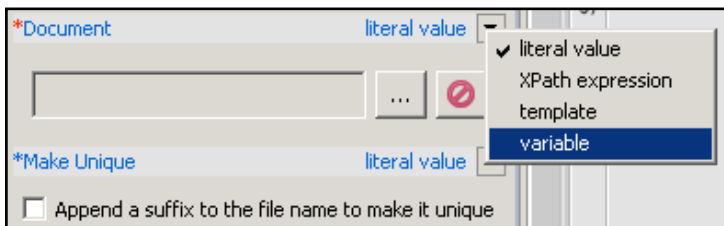
46. Connect the **Medical Check** and **Generate Final Itinerary** operation by selecting the **Medical Check** operation and dragging from its right anchor point to **Generate Final Itinerary** operation. Your process should look like.



Save PDF

The last operation we require for this part of the lab is a Foundation operation. Generating the static PDF itself we have taken care of, but if we want to see the final output we have to make it available somehow. One way of doing this is, is writing the PDF to the local file system. For that we can use the Write Document operation of the Foundation Service.

47. Drag once more the **Activity Picker** from the toolbar on the canvas and from the **Define Activity** dialog, select **File Utilities - 1.0 | Write Document** from the <Recently Used> list of services. Click **OK**.
48. In the Input part of the **Process Properties** panel for this service, specify `C:\MAX09L34\LCES\Outputs\eticket.pdf` as the **Pathname Pattern**.
49. Modify **literal value** to **Variable** for **Document** by selecting this from the dropdown menu



50. Select **docE-TicketPDF** from the **Document** dropdown listbox.
51. Ensure **Make Unique - Append a suffix to the filename to make it unique** is checked and **Over Write - Overwrite the existing file** is unchecked.
52. Connect the **Generate Final Itinerary** operation to the **Write Document** operation. The process should now look like.



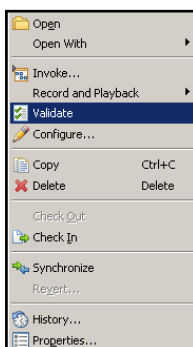
53. Save your orchestration by clicking on the **Save** button.



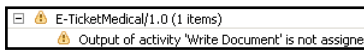
Validate

LiveCycle ES does have a validation feature so you can verify whether your orchestration is valid (e.g. you didn't forget required fields/values). To ensure we have a valid orchestration

54. Select the **Applications** panel, and right-click the **MedicalCheck** process in the **E-TicketMedical/1.0** folder. From the menu select **Validate**.



- You will see various errors/warning listed in the Validation Report tab at the bottom of Workbench, but there should be only one warning listed for our process: **E-TicketMedical/1.0 | Output of Activity 'Write Document' is not assigned.**



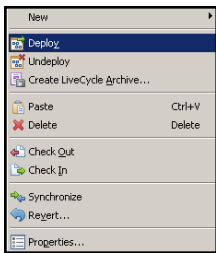
To fix this we need to capture the result of the Write Document operation in a variable.

- Go to the **Process Properties** panel.
- Select the **Write Document** operation on the canvas.
- In the **Output** part of the **Process Properties** panel, click on the large green + button. This will allow you to define a variable on the fly.
- In the **Variable** dialog enter `strWriteResult` as the **Name** and leave all other fields unchanged.
- Save the orchestration and once more validate the orchestration. Go to the **Application** panel, right-click **MedicalCheck** and select **Validate** from the menu. There should be no errors/warnings anymore for E-TicketMedical/1.0.

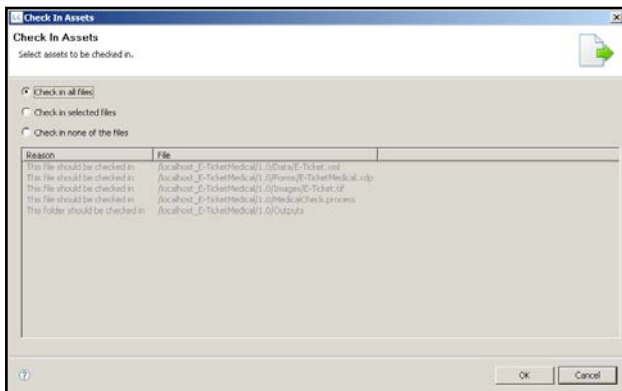
Deploy application

Sofar, all we have done is build a local version of our application. But we want to deploy the application to the LiveCycle ES server. To do this:

- Select **E-TicketMedical/1.0**, right-click and from the menu select **Deploy**.



- You will see a **Check In Assets** dialog. Ensure **Check in all files** is selected.



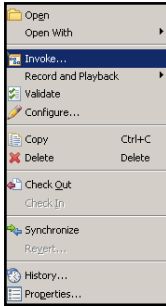
- Click **OK**.

Progress dialogs and indicators at the right bottom of Workbench will let you know of the progress of the deployment. It should take a couple of seconds to do the initial deployment.

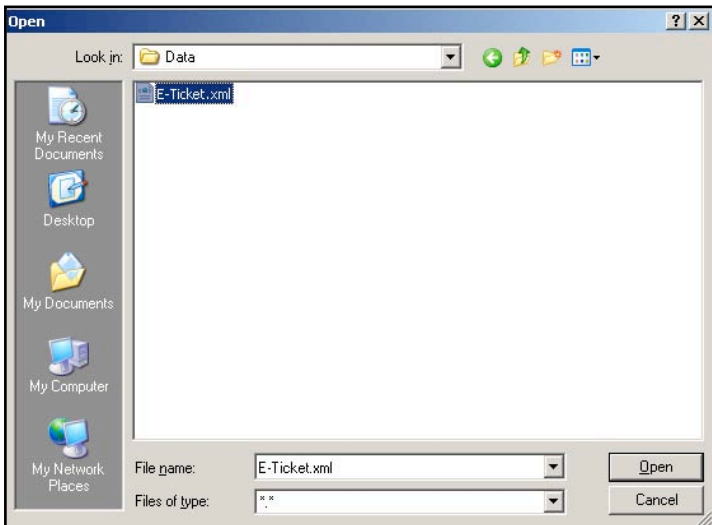
Invoke application

Once the deployment has finished, we can test our application by invoking it. To do this:

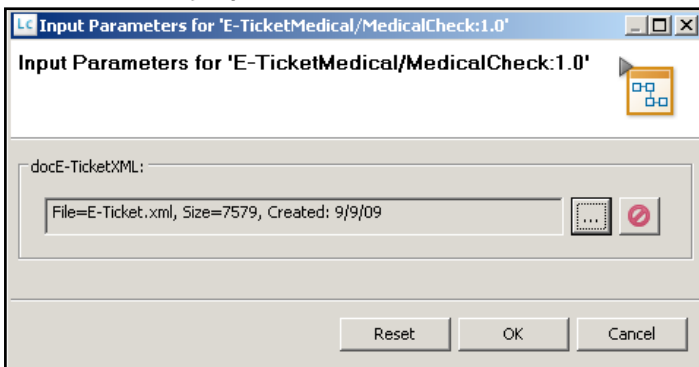
64. Select **MedicalCheck** from the **E-TicketMedical/1.0** subfolder in the **Applications** panel and click **Invoke**.



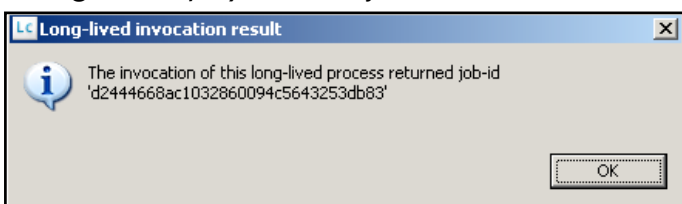
65. In the **Input Parameters** for '**E-TicketMedical/MedicalCheck:1.0**' dialog click on the elipsis ... button for the **docE-TicketXML** field.
66. In the **Open** dialog browse to the **C:\MAX09L34\LCES\Data** folder and select **E-Ticket.xml**.



67. When you click **OK**, a value similar to **File=E-Ticket.xml, Size=7579, Created: 9/12/09** is displayed in the **docE-TicketXML** field.



68. Click **OK** and you will see a progress indicator. After a while a **Long-lived invocation result** dialog will display with the job-id returned from the process invocation.

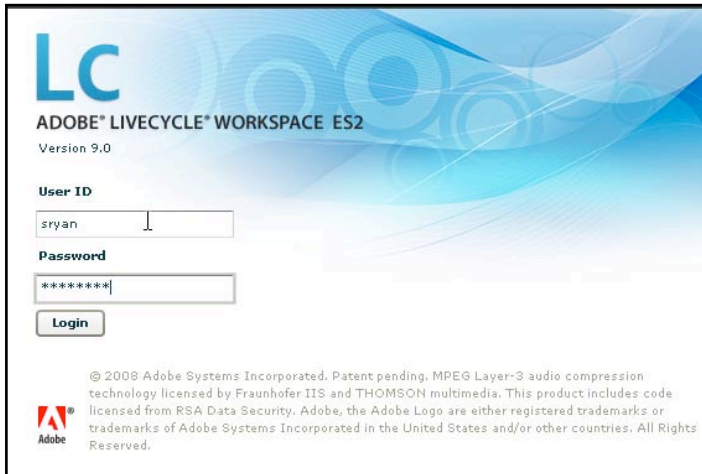


Act as travel assistant

We will check whether the process works correctly by acting as a travel assistant user. We will have to log in to Workspace, the default interface for LiveCycle ES, to see whether any tasks are awaiting action.

69. Open **Internet Explorer** and click on the **Workspace** button in the **Links** toolbar. This will show the **LiveCycle Workspace ES login** page.

70. Login with **User ID** sryan and **Password** password. Click **OK**.



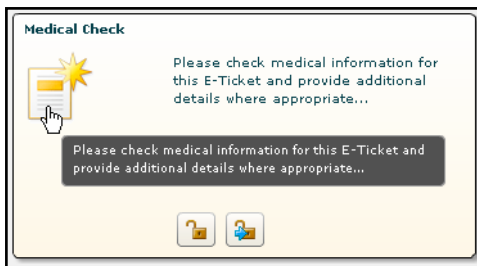
71. At the following page, click the **To Do** button. The next page will show you the waiting tasks for Shelly Ryan.

Shelly is member of the Travel Assistants group. She does not have a task waiting for her right now, but there should be one task at the Travel Assistants group queue.

72. Click on **Travel Assistants**



73. You will see a card named **Medical Check**.



74. Click on the icon in the card and you will see the **Task Details** of the selected task.

75. Now click the **Claim & Open** button at the bottom of the screen.

This will claim the task from the group queue, add it to the personal queue of Shelly Ryan and open the task's form. After a while, the E-TicketMedical form will render within the Workspace user interface.

76. Browse through the form and you can see that most of the form is unchangeable. Only the medical information can be changed and we will use this later in our orchestration.

Medical Check

Task Details Form Audit

Please fill out the following form.

Highlight Fields

Medical Information

	Yes	No	
Asthma	<input type="radio"/>	<input checked="" type="radio"/>	
Diabetes	<input type="radio"/>	<input checked="" type="radio"/>	
Seizure Disorder	<input type="radio"/>	<input checked="" type="radio"/>	
Heart Disease	<input type="radio"/>	<input checked="" type="radio"/>	
Hypertension	<input checked="" type="radio"/>	<input type="radio"/>	Medication required.
Pregnancy	<input type="radio"/>	<input type="radio"/>	
Swine Flu	<input type="radio"/>	<input checked="" type="radio"/>	Please provide details...

Insurance Information

Insurance Company Any Insurance

Address 274 Any Street

Any City, Any State, 33572

Save Offline Complete

77. Click **Complete** to complete the form.

Our second and third operation in the orchestration should have taken care of generating a PDF from the submitted form as well as writing the PDF to the file system. To verify whether that worked:

78. In **Windows Explorer**, go to the **LCES\Outputs** folder in the **C:\MAX09L34**. You should see an **etickets.pdf**. You can open it to check the content.

Lab 2

We are going to amend our orchestration with conditional routing, more user involvement as well as database integration.

We need to route our process in different directions based on what the travel assistant has filled in in the form. When the travel assistant has indicated the person for the itinerary has (had) swine flu, a travel supervisor has to inspect the form as well. Otherwise it can follow the same process we used in our first lab.



In order to route according to this specification we need to capture the XML from the template, and inspect it. This lab will focus on implementing this functionality,

Map to XML

First we need to get our hand on the XML that is part of the submitted form. We cannot inspect this XML straight from the document variable docE-TicketXML, so we need to map the document variable to an XML variable. First we do some prep activities.

1. Select the connection between **Medical Check** and **Generate Final Itinerary** and delete it by pressing the Del key. You will most likely get a prompt asking whether you want to check out this asset. Click **Yes**.
2. Move both **Generate Final Itinerary** and **Write Document** to the right of the canvas. Leave them connected.

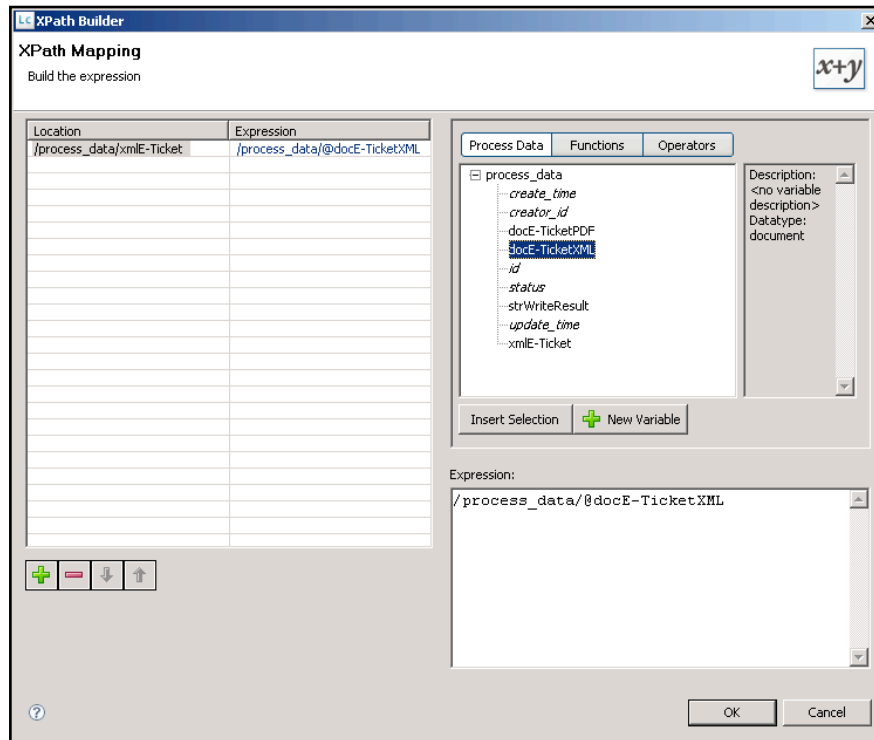
Now we will define the necessary mapping.

3. Insert a **Set Value** operation by dragging its icon  from the toolbar on the canvas to the right of the **Medical Check** operation.
4. In the **General** part of the **Process Properties** panel, give it the name Get Medical Status.
5. In the **Mapping** part of the **Set Value** operation we need to set up proper mappings to inspect the XML from the submitted form. Click on the **Edit the mapping data** button ; this will open the **XPath Builder** dialog.

We will start creating a new variable that will contain the raw XML from the submitted form.

6. In the **XPath Builder** dialog click the + **New Variable** button. As you can see this is one more place in Workbench where you can create a process variable on the fly.
7. In the **Variable** dialog specify xmlE-Ticket as **Name** and select xml as **Type**. Leave all other fields unchanged and click **OK**.
8. Click the first **[location]** cell of the **Location Expression** table and then double-click **xmlE-Ticket** in the **Process Data** listbox. This will insert /process_data/xmlE-Ticket in the **Location** column as well as in the **Location** text area.
9. Select the **[expression]** cell in the **Location Expression** table and then double-click **docE-TicketXML**. This will insert /process_data/@docE-TicketXML in the **Expression** column as

well as in the **Expression** text area. The **XPath Builder** dialog should look like



We have mapped the content of a document into a variable of type xml. This way we can easily inspect the XML which would not be possible if we were using a variable of type document.

10. Press **OK** to close the **XPath Builder** dialog. You should see the mapping, you just defined, in the **Process Properties** panel.
11. Connect the **Medical Check** operation to the **Get Medical Status** operation.

Involve Travel Supervisors


We need to add a new Assign Task to the orchestration in order to involve the Travel Supervisors. Rather than creating an Assign Task from scratch we can copy the existing Assign Task and modify some of its parameters. To do this:

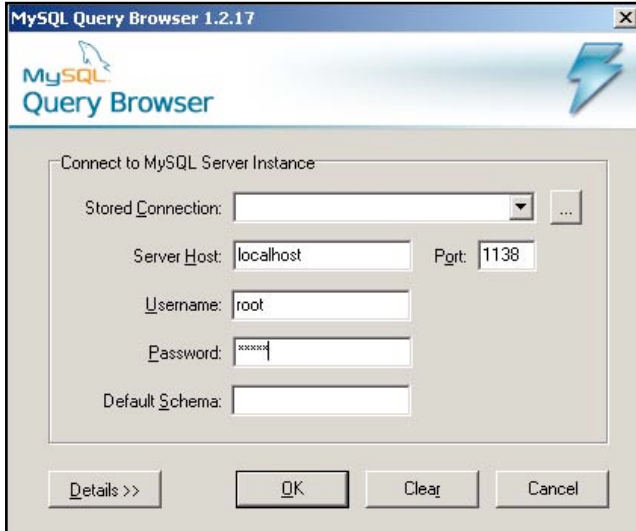
12. Select the **Medical Check** operation and click the **Copy** button (or use Ctrl-C) from the toolbar below the **Workbench** menu.
13. Unselect any operation by clicking on an empty spot on the canvas and click the **Paste** button (or use Ctrl-V).
14. Move the **Copy(1) of Medical Check** operation, somewhat below the current operations and to the right of the **Get Medical Status** operation.
15. Rename the operation to **Final Medical Check**.
16. In the **Initial User Selection** click the **Browse...** button as part of the **Assign to Group**.
17. From the **Select Group**, select **Travel Supervisors**.
18. Optionally change the **Task Instructions**. All other values for the various parameters can be left unchanged.

Database integration

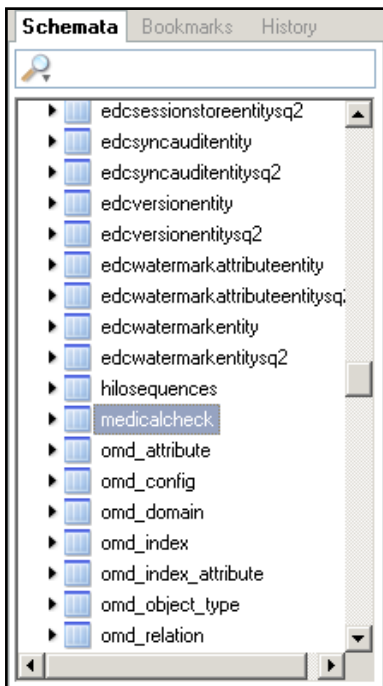
When the Travel Assistant has indicated that the traveller(s) has (had) swine flu, we want not only to route the process to travel supervisors but also add a record to a database table. The setup for this record insertion is quite simple for this lab, but illustrates how you can integrate a


LiveCycle orchestration with databases using the Execute SQL statement operation from the JDBC service. There are other database operations available within LiveCycle ES as well.

19. For a view of the specific table, open **MySQL Query Brower** by clicking on the icon  in the taskbar.
20. Log in with **Username** root and **Password** adobe.

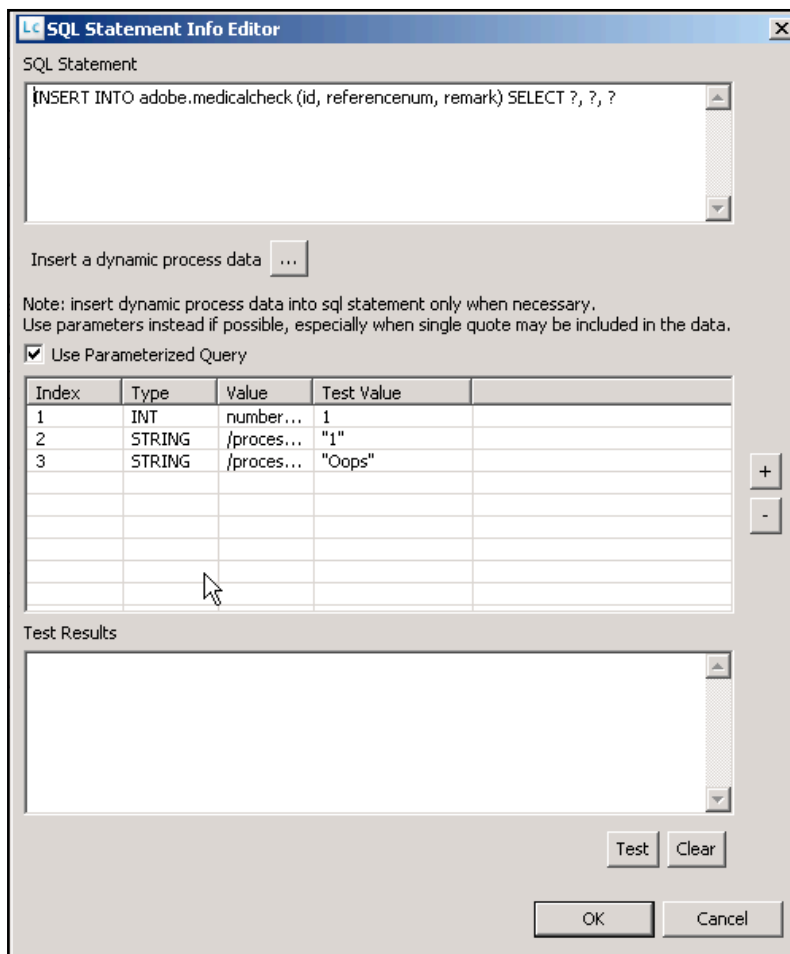


21. Unfold the **adobe** schema (in the **Schemadata** panel) and double click the **medicalcheck** table.



22. This will insert an `SELECT * FROM adobe.medicalcheck m;` into the **ResultSet1** tab.
23. Click on the green arrow button  to execute the SQL statement and you will see no data is currently in the table.
24. Return back to **Workbench**. To add a record to the database table through our orchestration, drag the **Activity Picker** to an empty spot on the canvas and select **JDBC - 1.0 | Execute SQL statement** from the **<Recent Used>** list.
25. In the **Name** field in the **General** area of the **Process Properties** panel enter Insert Record.

26. Click the elipsis ... button next to the **SQL Statement** field; this will open the **SQL Statement Info Editor**.
27. In the **SQL Statement** text area type `INSERT INTO adobe.medicalcheck (id, referencenum, remark) SELECT ?, ?, ?`.
28. The question marks in the statement indicate we will use a parameterized query, so select the **Use Parameterized Query** checkbox.
29. Add a row in the table by clicking the + button. In the first row, select **INT** for the **Type** column. In the **Value** column click the small button. This will open the **XPath Builder** dialog.
30. Double-click **xmlE-Ticket** and edit the expression to read `number(/process_data/xmlE-Ticket//SSN)`.
31. Set **Test Value** to 1.
Normally we would have defined a schema for the XML variable and would be able to traverse to the appropriate element in the **XPath Builder** dialog. For now, we make it ourselves easy by using `//`.
32. Insert a second row; set **Type** to **STRING**, set **Value** to `/process_data/xmlE-Ticket//referenceNum` (once more using the **XPath Builder**) and set **Test Value** to "1" (include quotes).
33. Insert a third row: set **Type** to **STRING**, set **Value** to `/process_data/xmlE-Ticket//swinefluDetails` (once more using the **XPath Builder**) and set **Test Value** to "Oops" (include quotes).



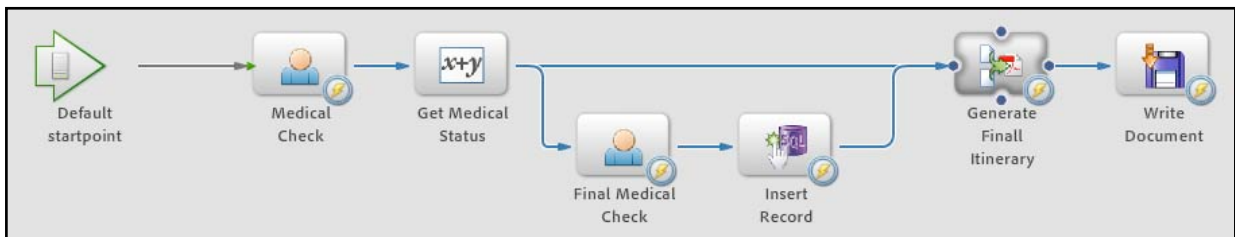
34. Click the **Test** button and if all is correct you will see **Numbers of Rows affected: 1** in the **Test Results** area.

35. If you're not sure yet, switch to **MySQL Query Browser** and click the green arrow button. You will see the row in the table.
36. Switch back to **Workbench** and in the **SQL Statement Info Editor** click **Test** button once more. You will now get an exception which we will deal with in lab 3. For now, click **OK** to close the dialog
37. Back in the **Process Properties** panel, click the + button in the **Output** area to add a variable for the **Numbers of Rows Affected**.
38. In the **Variable** dialog enter `intNrOfRowsAffected` as the **Name** and ensure **int** is selected as the **Type**.
39. Leave all other fields unchanged and click **OK**.

Connect operations

We have to make all the necessary connections for our orchestration. To do this:

40. Connect **Medical Check** to **Get Medical Status**.
41. Connect **Get Medical Status** to **Generate Final Delivery** and also connect **Get Medical Status** to **Final Medical Check**.
42. Connect **Final Medical Check** to **Insert Record**.
43. Connect **Insert Record** to **Generate Final Delivery**. Your orchestration should look like

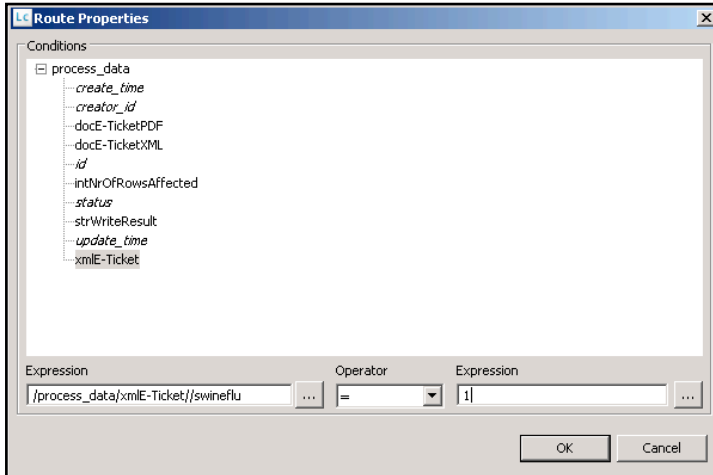


Add routing condition

We have two routes coming out of **Get Medical Status**; we should add at least some condition to let the orchestration select the appropriate route. To do this:

44. Select the route from **Get Medical Status** to **Final Medical Check**.
45. In the **General** part of the **Process Properties** panel set **Name** to `Not OK`.
46. In the **Conditions** table, add a row by clicking on the green + button. This will open the **Route Properties** dialog.
47. Click within the left **Expression** text field, then double-click **xmlE-Ticket**. This will insert `/process_data/xmlE-Ticket` in the left **Expression** text field. Add `//swineflu` so it will now contain `/process_data/xmlE-Ticket//swineflu`.

48. Type 1 in the right **Expression** text field. Leave **Operator** unchanged. Your Route Properties should look like



49. Click **OK** to close the dialog.

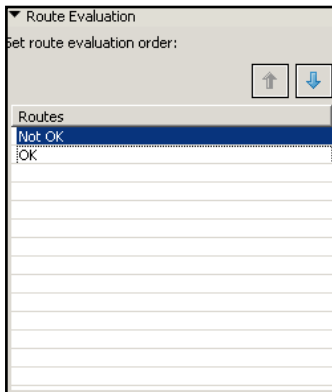
Back on the canvas you can see the edited route is now dashed and has a label.

50. For readability purposes, select the route from **Get Medical Status** to **Generate Final Delivery** and set its **Name** to OK.

Specify evaluation order

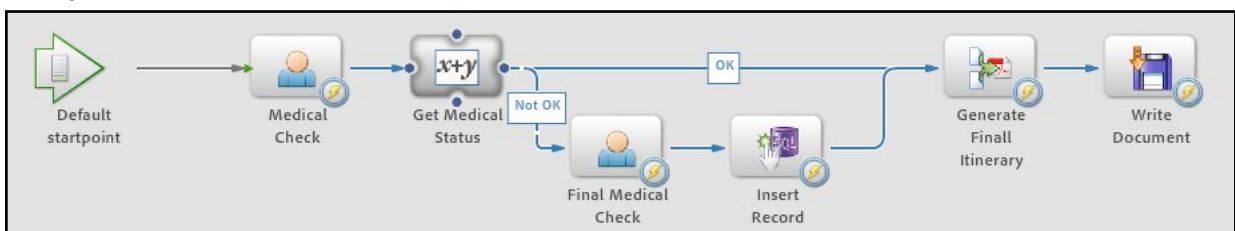
We need to specify in what order our orchestration will evaluate these conditions. To do this:

51. Select the **Get Medical Status** operation.
 52. Unfold the **Route Evaluation** area of the **Process Properties** panel.
 53. Ensure **Not OK** is listed first and **OK** second. To change the order select an item and use the up and down buttons. The Route Evaluation area should finally look like



With this evaluation definition we always want to ensure we first check the **Not OK** route as this has a condition.

54. Verify the final orchestration for lab 2 looks like



Test application

It's time to test the extended orchestration in our application..

55. Save your work and then right click **ETicketMedical/1.0**.
56. Select **Deploy** from the menu.
57. You will see a **Check In Assets** dialog. Ensure **Check in all files** is selected and click **OK**.
58. Right click **MedicalCheck** and select **Invoke** from the menu. Provide the same XML you did earlier as part of lab 1 (you do not have to reselect the XML file using the elipsis button).
59. Log in to **Workspace** with **User ID** sryan and **Password** password. You should see again an item in the **Travel Assistants** group.
60. **Claim & Open** the task as you did before but before completing the form, set the **Swine Flue** radio button in the **Medical Information** section to **Yes** and type some description in the **Description** field behind the **Swine Flu** label and radio buttons, as illustrated below.

	Yes	No	
Asthma	<input type="radio"/>	<input checked="" type="radio"/>	
Diabetes	<input type="radio"/>	<input checked="" type="radio"/>	
Seizure Disorder	<input type="radio"/>	<input checked="" type="radio"/>	
Heart Disease	<input type="radio"/>	<input checked="" type="radio"/>	
Hypertension	<input checked="" type="radio"/>	<input type="radio"/>	Medication required.
Pregnancy	<input type="radio"/>	<input type="radio"/>	
Swine Flu	<input checked="" type="radio"/>	<input type="radio"/>	Three days extremely high temperature


61. Then click **Complete** to submit the form.
Due to our routing in the orchestration, the form should end up in the **Travel Supervisors** queue. To check this:
62. In **Workspace** log out as user Shelly Ryan by clicking on **Logout**, and log in as user John Steward with **User Id** jsteward and **Password** password.
63. You should see a task waiting in the **Travel Supervisors** group.
64. Open the group's list and **Claim & Open** the task like you did before.
65. Ensure the information supplied by Shelly Ryan is in the form (**Swine Flue** set to **Yes** and her description text is shown).
66. Submit the form without making any changes by clicking the **Complete** button.
Our orchestration should have inserted a record in the database after travel supervisor John Steward submitted the form. To check this:
67. Open **MySQL Query Browser** (if not already open).
68. Double-click the **medicalcheck** table in the **adobe** schema. This will insert a select statement in the **ResultSet1** tab window.
69. Execute the statement by clicking the green arrow button at the toolbar. If all is correct you should see a record with data from the form that just has been submitted by the travel supervisor.
70. You should also check your **Outputs** folder in **c:\MAX09L34\LCES\Outputs**; it should contain an additional **e-ticketxxx.pdf**. You can open it to verify the content.

Lab 3

In our third part of the lab we are going to extend the orchestration even further. We will add web service and content management integration, exception handling as well as integrate a custom component.

Integrate Content Services as a web service

We will start with integrating a web service.

1. In **Workbench**, drag the **Invoke Web Service** from the toolbar  on the canvas, below of what is currently designed as orchestration.
2. If you get prompted whether you want to check out this asset, click **Yes**.

In this lab we are going to invoke a very simple web service but LiveCycle ES is perfectly capable of invoking more complex web services.

As we can not rely on any connectivity in our lab room, we are going to use a LiveCycle service itself. Almost any LiveCycle ES service is exposed as a web service and so are the services provided by the LiveCycle ES Content Services component. In our lab we are going to use the DocumentService web service for illustration purpose.

3. Select the **Invoke Web Service** operation you just dragged on the canvas.
4. In the **General** part of the **Process Properties** panel change the **Name** to Call Web Service for CMS.
5. Open the **Web Service Options** part of the **Process Properties** panel and click on the ellipsis ... button next to the **Options** text field. This will open the **Web Service Settings** dialog.
6. In the **Web Service Settings** dialog, ensure the **Settings** tab is active. In the **WSDL URL** field type the following: `http://localhost:8080/soap/services/DocumentManagementService?wsdl&lc_version=9.0.0&version=1.0`
7. Press the **Load** button. This will query the **DocumentManagementService** web service for its definition, operations, etc. and automatically fill in some of the other fields in the dialog.
8. Set **User Name** to administrator and **Password** to password.
9. Select **createSpace** from the **Operation** dropdown listbox. Leave all other fields unchanged.



WebService Settings

Settings | HTTP Settings | Request | Attachment | Test

WSDL URL: Load

User Name: XPath

Password: XPath

Port:

Target URL: XPath

Operation: XPath

Time Out (in secs):

WSDL Content:

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://adobe.com/ldp/sei"
<!--Generated For Adobe LiveCycle 9.0.0-->
<wsdl:types>
<schema elementFormDefault="qualified" targetNamespace=
<import namespace="http://xml.apache.org/xml-soap"/>
<import namespace="http://ws-i.org/profiles/basic/1.1/XMLSchema"
<element name="storeContentAPI">
<complexType>
<sequence>
<element name="storeName" type="xsd:string"/>
<element name="spacePath" type="xsd:string"/>
<element name="nodeName" type="xsd:string"/>
<element name="nodeType" type="xsd:string"/>
<element name="nodeContent" type="impl:LOB"/>
<element name="nodeEncoding" type="xsd:string"/>
<element maxOccurs="1" minOccurs="0" name="update/e
<element maxOccurs="1" minOccurs="0" name="aspects"
<element maxOccurs="1" minOccurs="0" name="attributef
</sequence>

```

Embed WSDL

OK Cancel

10. Move to the **Request** tab of the **Web Service Settings** dialog.
11. Click on the **Generate** button next to the **SOAP Request** text area and also click on the **Generate** button next to the **SOAP Request for Test** text area. Both button clicks will generate a skeleton web service request.
12. Remove the `<ser:StoreName?></ser:StoreName>` element and replace `?` in the the `<ser:spacePath?></ser:spacePath>` element with `/Company Home/Medical` in both the **SOAP Request** and **SOAP Request for Test** text areas, so each looks like


```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ser="http://adobe.com/idp/services">
  <soapenv:Header/>
  <soapenv:Body>
    <ser:createSpace>
      <ser:spacePath>/Company Home/Medical</ser:spacePath>
    </ser:createSpace>
  </soapenv:Body>
</soapenv:Envelope>
```

The **Request** tab should look like



13. To test out our web service, move over to the **Test** tab and click the **Test** button. This will invoke our **SOAP Request for Test** request and if all is good you should see


```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <createSpaceResponse xmlns="http://adobe.com/idp/services">
      <createdSpaceID>1910bfc9-df50-4bd1-991e-8c48a6424783</
  createdSpaceID>
    </createSpaceResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

Note that the `createSpaceID` can be different in your environment.

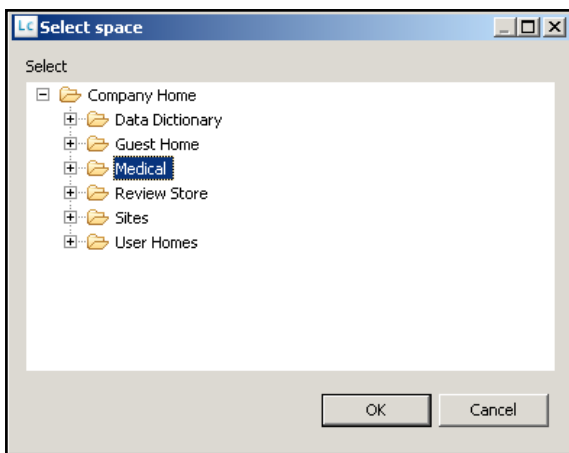
It is important to remember that the createSpace operation is also available as a native LiveCycle ES operation. In the **Define Activity** dialog that pops up when you drop the **Activity Picker** on the canvas, you can find it in the **Content Services** list.

14. Click **OK** to close the **WebService Settings** dialog.
15. To finish off our web service operation, unfold the **Web Service Response**.
16. Click the green + button and in the **Variable** dialog enter strWSResult as the **Name**. Ensure **string** is selected as the **Type**.
17. Select **Unlimited Length** in the **Datatype Definition** area. This is important: we want to ensure the web service response fits in the size we allocate for this string variable. The default of 100 characters most likely is not sufficient.
18. Leave all other fields unchanged and click **OK**.

Integrate Content Services natively

We will now add another operation that natively uses the LiveCycle ES Content Services. To do this:

19. Drop the **Activity Picker** button from the toolbar on the canvas right to your latest **Call Web Service for DMS** operation.
20. In the **Define Activity** dialog, from the **Content Services** list, select the **Document Management - 1.0 | storeContent** operation.
21. Click **OK**.
22. With the **storeContent** operation selected, in the **General** area of the **Process Properties** panel, change its name to Store CMS.
23. In the **Input** area click the **Browse** button next to the **Space Path** text field and select the **Medical** subfolder from **Company Home**. Remember, by testing our web service earlier, we effectively created the space already.



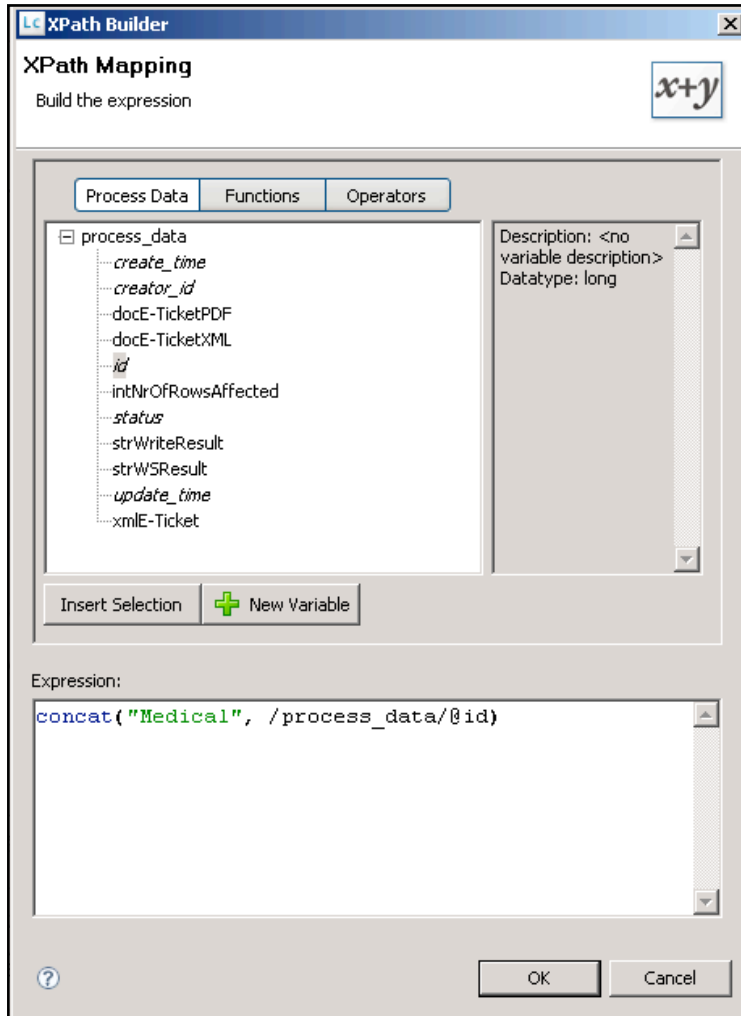
We want to name each document we store in LiveCycle ES Content Services uniquely by appending the process id number to the name. To do this

24. Select **XPath expression** from the dropdown menu next to **Node Name**.



25. Click the ellipsis ... button and in the **XPath Builder** dialog select the **Functions** button.
26. Unfold **String functions** and double-click **concat(string, string, string*)**; this will insert `concat(string, string, string*)` in the **Expression** text area.

27. Double-click the first string in the **Expression** text area and type “Medical Record” (include the quotes).
28. Double-click the second string in the **Expression** text area, click the **Process Data** button and double-click **id**; this will replace string with /process_data/@id.
29. Remove the third string* from the expression. The final expression should now read like `concat("Medical Record", /process_data/@id)`.



30. Click **OK** to close the **XPath Builder** dialog.
31. Select **content** from the **Node Type** dropdown listbox.
32. For **Node Content**, select **Variable** from the dropdown menu button and select **docE-TicketPDF** from the dropdown listbox.

Add logging

To add some logging to our orchestration we are going to add a log operation. To do this:

33. Drag the **Activity Picker** to the canvas and select **Variable Logger - 1.0 | log** from the **<Recently Used>** list.
34. In the **General** area of the **Process Properties** panel enter **Log Variables** as the **Name**.
35. Unfold **System Logging** and check both **Log to system logger** and **Log to standard out** checkboxes.
36. Select **Debug** from the **Logging Level** dropdown listbox.


Integrate Twitter

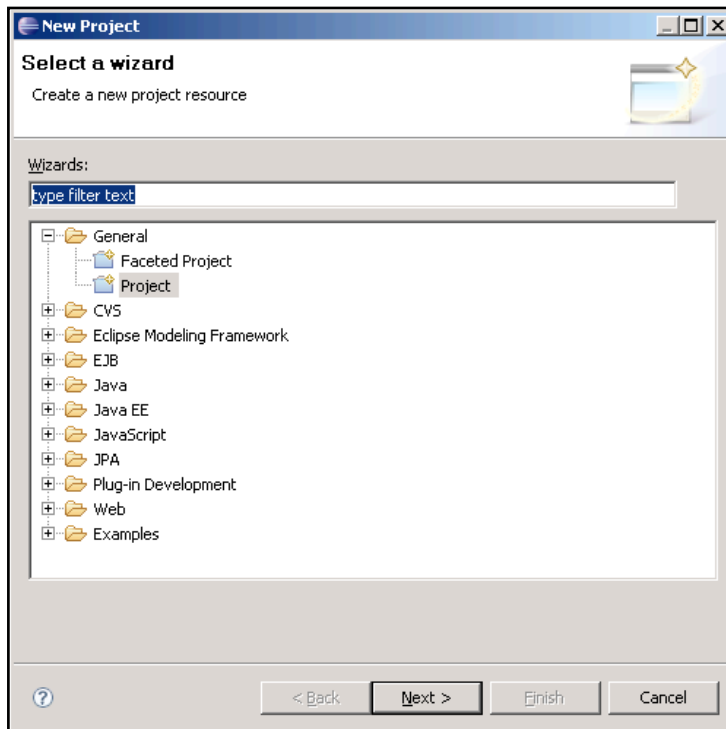
Finally we are going to integrate a custom component into our LiveCycle orchestration. This is just to illustrate how you can add any plain old Java functionality as a service with operations into LiveCycle. Our service is a very simple Twitter service written once by Mark Szulc (<http://www.markszulc.com/>) and somewhat adopted for this purpose. It questionable whether Twitter integration does make sense in this orchestration from a business perspective, but for illustrating how to integrate your own Java code in LiveCycle ES, it serves a purpose.

First we want to remove the already existing service from our LiveCycle ES system. To do this

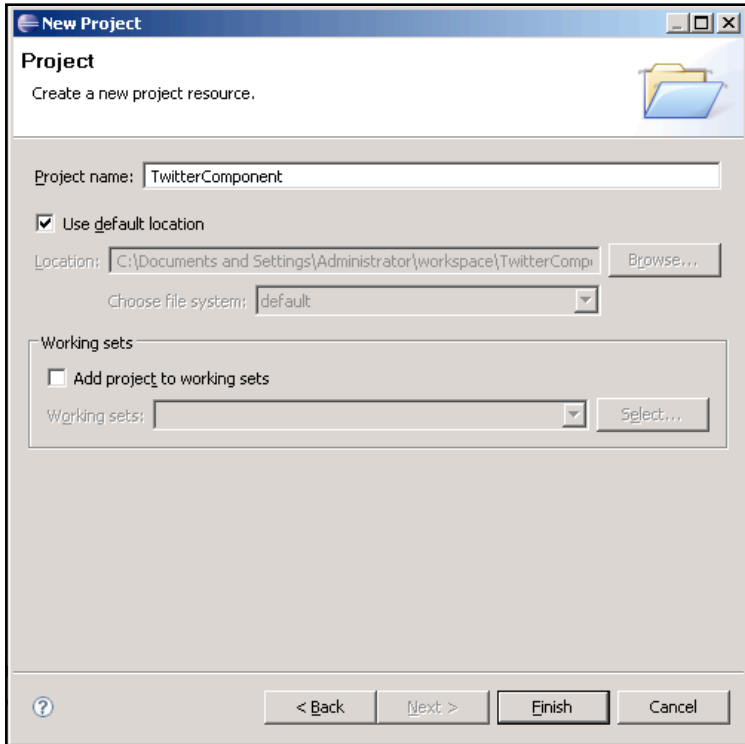
37. Bring back the **Components** panel in **Workbench** by selecting **Windows | Show View | Components** from the menu.
38. In the **Components** panel, unfold **Components**, right-click **TwitterComponent** and select **Stop Component** from the menu.
39. Once more right-click **TwitterComponent** and select **Uninstall Component**. When prompted whether you're sure, click **Yes**.

We will take a closer look at our very basic Java code, which uses a Twitter API library, to update Twitter.

40. Open **Eclipse** by clicking on the Eclipse  button in the Windows taskbar.
41. If you're getting a **Usage Data Upload** dialog in Eclipse, select **Don't Upload Now** and select **Finish**.
42. In **Eclipse**, select **File | New > Project...** from the menu.
43. In the **New Project | Select a Wizard** dialog, select **Project from General**. Click **Next**.



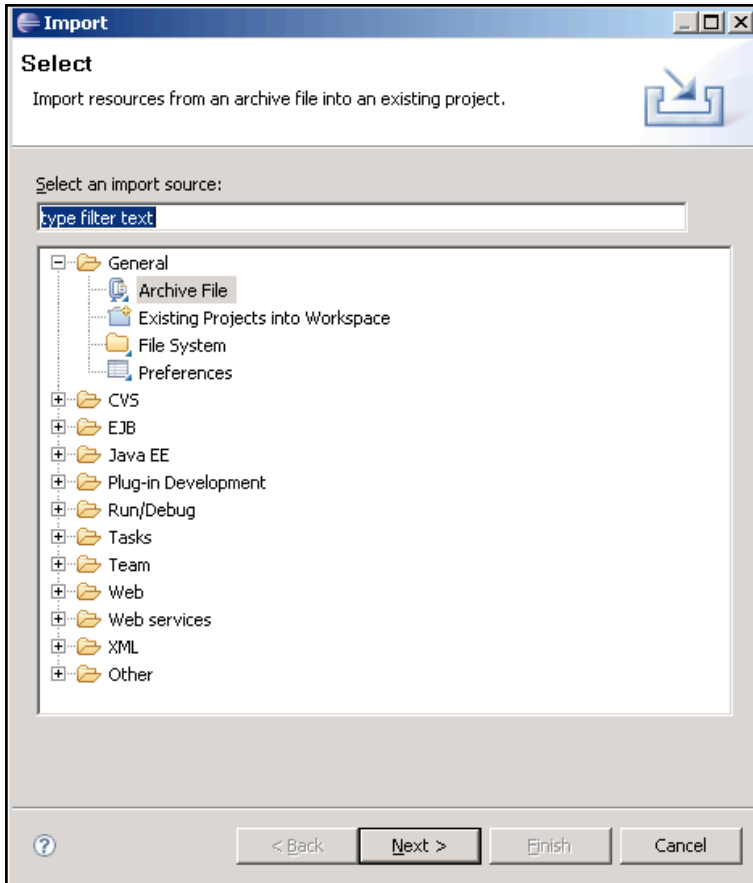
44. Set the **Project Name** for the project to `TwitterComponent`; leave all other fields unchanged.



45. Click **Finish**.

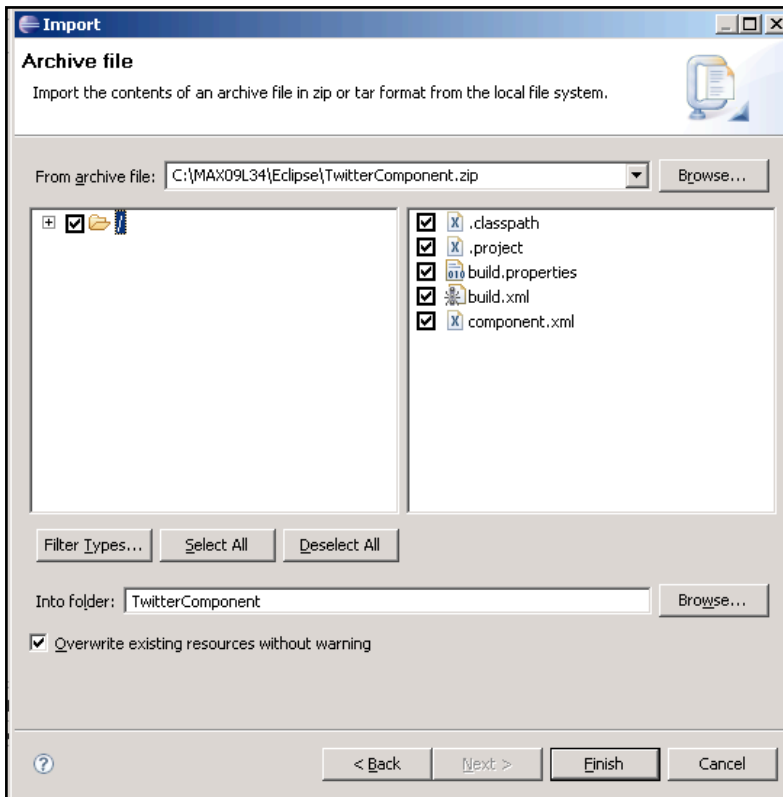
46. You should see your project in the **Project Explorer**. Right-click on it and select **Import...** from the menu.

47. In the **Import Select** dialog, select **Archive File** from **General**.

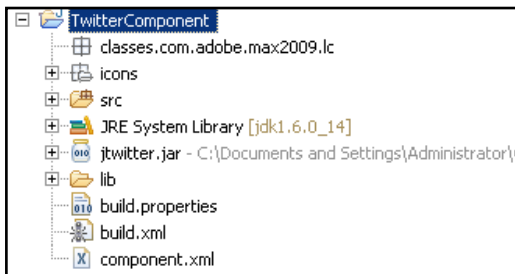


48. Click **Next**.

49. In the **Import Archive File** dialog, click the **Browse..** button. You should see the **Eclipse** subdirectory from **c:\MAX09L34** listing **TwitterComponent.zip** and **TwitterComponentComplete.zip**. If not, browse to that directory.
50. Select **TwitterComponent.zip** and click **Open**. You will see all components of the project appearing in the dialog.



51. Don't change any of the fields, simply click **Finish**.
52. In the **Project Explorer**, unfold **TwitterComponent**. Your **Project Explorer** should look like



53. Unfold **src**, unfold **com.adobe.max2009.lc** and double-click **TwitterComponent.java**. Eclipse will show the very simple code to call the Twitter API to update a Twitter account with a tweet. This **UpdateTwitter** method needs to become an operation we can use from within LiveCycle ES. In order to do this we need to tell LiveCycle ES how to treat this Java functionality properly. This is done by configuring the **component.xml** file with the appropriate information. To do this:

54. Double-click **component.xml** in the **Project Explorer** in **Eclipse**.

Some of the information is already provided for you, but the important services information is missing. To fix this

55. Using **Windows Explorer**, browse to **C:\MAX09L34\Eclipse**.
56. Right click the **TwitterComponentComplete.zip** file. From the menu select **Winzip | Extract to here...**

57. When the files are extracted, right click the **component.xml** file in **Windows Explorer** and select **Open With | Wordpad**

58. Select the following XML snippet from within the `<services></services>` element in **Wordpad** and paste it into the **component.xml** file that is currently open in **Eclipse**, at the exact same location (in between the `<services></services>` element).

```
<service name="Twitter" orchestrateable="true" title="Twitter">
  <hint>Provide Account ID and Password</hint>
  <description>Allows status updates to be written to Twitter</description>
  <small-icon>icons/twitter.png</small-icon>
  <large-icon>icons/twitter.png</large-icon>
  <auto-deploy major-version="1" minor-version="2" category-
id="OnlineServices"/>
  <implementation-class>com.adobe.max2009.lc.TwitterComponent</
implementation-class>
  <operations>
    <operation name="UpdateTwitter" method="UpdateTwitter">
      <hint>Doesnt return anything just yet...</hint>
      <input-parameter name="accountid" title="Twitter Account ID"
type="java.lang.String" required="true">
        <description>Twitter Account ID</description>
        <hint>Enter your Twitter Account ID that you wish to
update</hint>
        <supported-expr-types>Literal,XPath,Template,Variable</
supported-expr-types>
      </input-parameter>
      <input-parameter name="password" title="Twitter Account
password" type="java.lang.String" required="true">
        <property-editor editor-
id="com.adobe.idp.dsc.propertyeditor.system.PasswordPropertyEditorComponent"/>
        <description>Twitter Account Password</description>
        <hint>Enter your Twitter Account password that you wish
to update</hint>
        <supported-expr-types>Literal,XPath,Template,Variable</
supported-expr-types>
      </input-parameter>
      <input-parameter name="tweet" title="Tweet (Message)"
type="java.lang.String" required="false">
        <description>Tweet (Message)</description>
        <hint>Enter your Tweet (Message) you'd like to send</
hint>
        <supported-expr-types>Literal,XPath,Variable</supported-
expr-types>
      </input-parameter>
    </operation>
  </operations>
</service>
```

59. In Eclipse, save the project by clicking on the **Save** button.

Let us briefly visit the XML we just inserted.

With the service element we specify the service. For that service we can specify details like icons, a description, etc. but most importantly we can define which class implements the service.

For the service we also define the operations, in our case it is only one operation

UpdateTwitter, with its input and output parameters.,

Our service implementation only uses three inputs (accountid, password, tweet) and for each of these input paramters we can define title, type, description (which will be the label in the Process Properties panel), hints, which expression types are supported (literal, xpath, variable), etc.

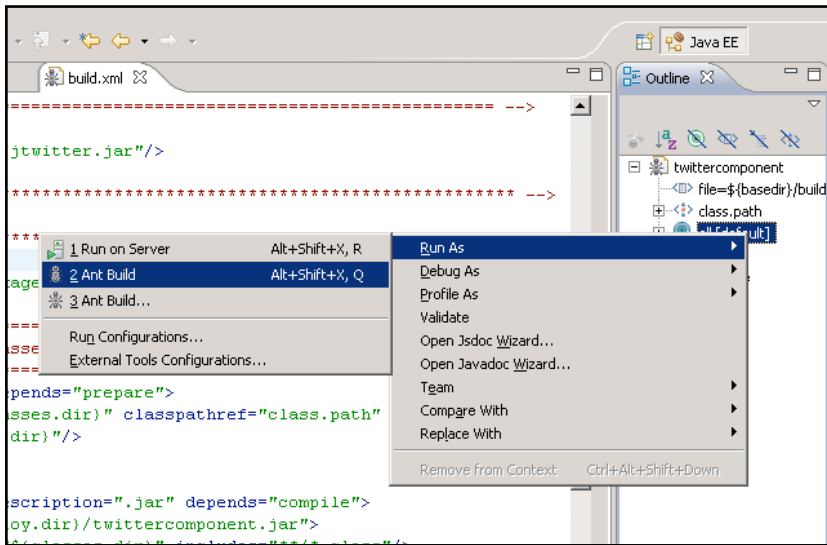
As you can see, ultimately, the **component.xml** defines how your plain old Java code is represented as a service with operations in LiveCycle. ES

We will now build the component as a jar file. All setup to do this already in place.

60. Double-click the **build.xml** file in **Eclipse's Project Explorer**.

61. The structure of the ant build file appears in the **Outline** panel.

62. Select **all [default]**, right-click and from the menu select **Run As > Ant Build**.

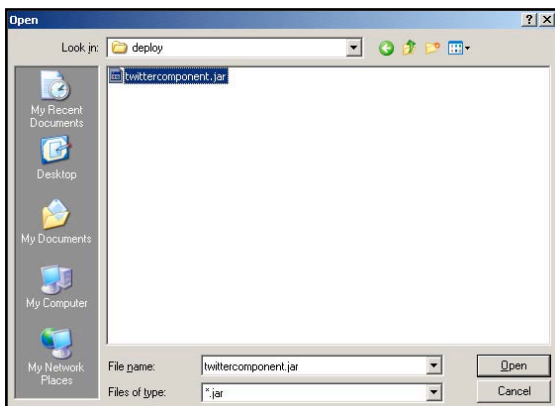


63. If all is ok, you should see the build process progress in the **Console** panel in **Eclipse**, ending with the line **BUILD SUCCESSFUL**.

64. Right-click **TwitterComponent** in **Project Explorer** and select **Refresh** from the menu. Your Project Explorer will now show a **deploy** subfolder with **twittercomponent.jar** in it.

We are going to add our Twitter component to LiveCycle. To do this:

65. Switch over to **Workbench**, and in the **Components** panel, right click on **Components** and select **Install Component**. In the **Open** dialog, select the **twittercomponent.jar** file from the **c:\Documents and Settings\Administrator\workspace\TwitterComponent\deploy** directory.



66. After the installation is finished, select **TwitterComponent** in the **Components** panel. It should have a red icon, indicating it is not started yet.
67. Right-click the **TwitterComponent** and select **Start Component** from the menu. Now the component is available for use in any orchestration.
68. To add the Twitter service to your orchestration, drag **Activity Picker** from the toolbar to the canvas and select **Twitter - 1.2 | UpdateTwitter** from the **<Recently Used>** or **OnlineServices** list in the **Define Activity** dialog.
69. In the **General** area enter Update Twitter as the **Name**.
70. In the **Input** area of the **Process Properties** panel, set **Twitter Account Id** to your twitter account id if you have one. Otherwise, use the dedicated Twitter account we have setup for this lab: max09134.
71. Set the **Password** to your password.. If you want to use the dedicated Twitter account for this lab, the password is max2009.
72. Set **Tweet** to any message you like.

Handle Exceptions

We have one more operation to add. We need to handle the exception of a duplicate insertion into the database.

We are going to use the Script service in LiveCycle. With the Script service you can add on the fly Java code to your orchestration. This is not really recommended as it will make your orchestration difficult to debug, but it might be useful in certain scenarios and we use it here to illustrate exception handling in general and another Foundation service of LiveCycle ES more specifically.

73. Drag the Activity Picker from the toolbar and select **Execute Script - 1.0 | executeScript** from **<Recently Used>** list in the **Define Activity** dialog.
74. In the **General** area in the **Process Properties** panel, name the operation **Log Exception** and click on the ellipsis ... button next to the **Script** text field in the **Input** area.
75. In the **TextArea** dialog type `System.out.println("SQL Exception...");` (Note the trailing semicolon!).

Connect operations

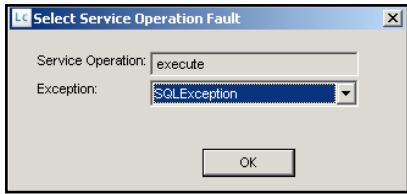
The final step is to connect all operations in our process.

76. Connect **Write Document** to **Call Web Service to DMS**.
77. Connect **Call Web Service to DMS** to **Store CMS**.
78. Connect **Store CMS** to **Log Variables**.
79. Connect **Log Variables** to **Update Twitter**.

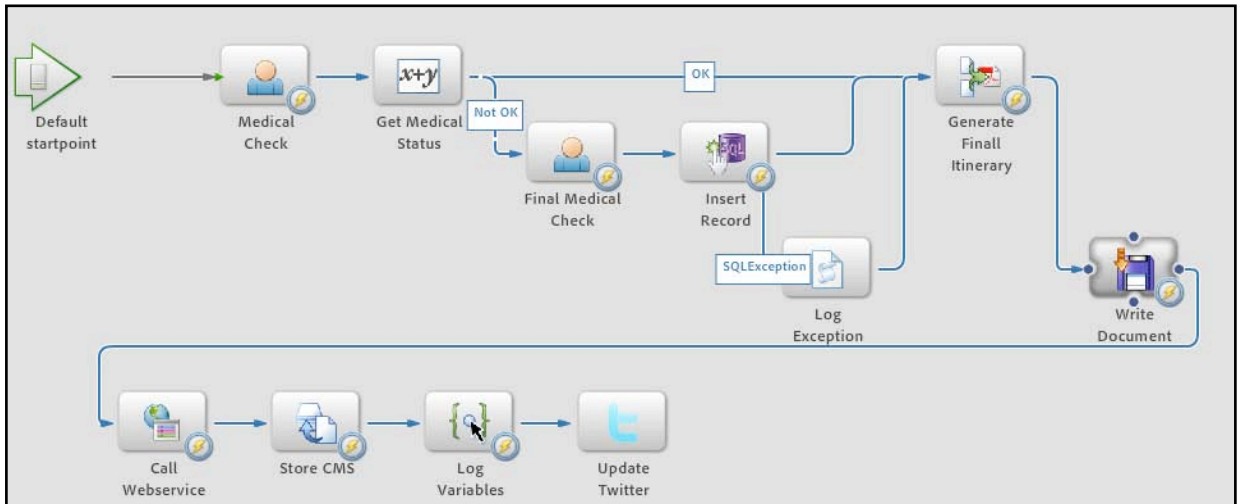
We will also setup the required connection for our exception handling. To do this:

80. Connect from the exception handler icon  in the **Insert Record** operation to **Log Exception**.

81. In the **Select Service Operation Fault** dialog, select **SQLException** from the **Exception** dialog. This will ensure that when an SQL exception occurs the process will not be stalled.




82. Connect the **Log Exception** to **Generate Final Delivery**. Your orchestration should now look like

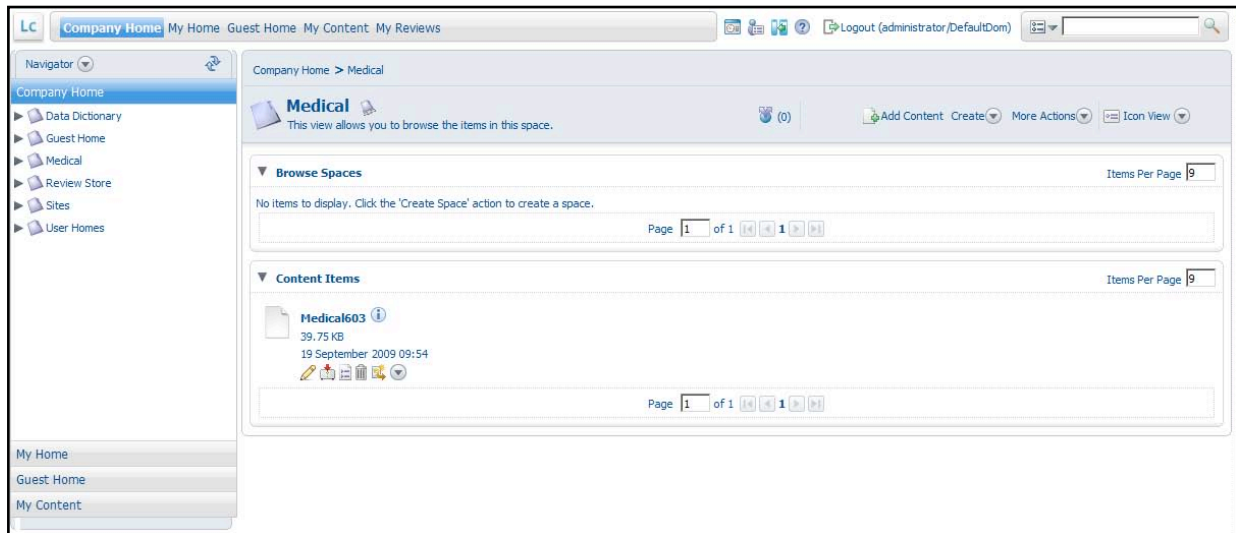


Test application

The proof of the pudding is in eating it. So, we will do our final test of the application and underlying process.

83. Save the orchestration, deploy it by going to the **Application** panel, right-click **E-TicketMedical/1.0** and click **Deploy**.
84. Invoke the process by selecting **MedicalTest**, right click and select **Invoke...** from the menu. Select the same XML file you did earlier.
85. Use **Workspace** to logon as user Shelly Ryan with **User ID** sryan (as you did before) and you should see a task waiting in the **Travel Assistants** queue.
86. **Claim & Open** the task like you did before. Do not change anything and **Complete** the form. Your process should continue and finish.
87. Use **BareTail** from the Windows taskbar  to monitor the process execution. If all is ok, you should see your Twitter account being updated at the end of the log. It might be that the process is failing at this last step due to connectivity issues with the outside internet.
88. Goto to LiveCycle ContentSpace ES by clicking on the **ContentSpace** button in the **Links** toolbar in **Internet Explorer**.
89. Login to **ContentSpace** with **User Name** administrator with Password password.

90. You should see your content in the **Company Home | Medical**.



91. (Optionally) Repeat the invocation step with the same XML as input, but now select **Yes** for **Swine Flue** in the form.
92. The form should end up with John Steward, the travel supervisor (as you've seen before) and when you complete the form as this user, the process should continue and finish. Again use **BareTail** and you should see the exception handling message we introduced as part of this final lab.

Appendix

In case you're lost or didn't make it to the final end of any of the labs, we have provided intermediate applications you can load into your LiveCycle ES environment.

The files can be found in **C:\MAX09L34\LCES\LCA** and are named

- **E-TicketMedicalPart1.lca** for final application at the end of lab 1
- **E-TicketMedicalPart2.lca** for final application at the end of lab 2
- **E-TicketMedicalComplete.lca** for final application at the end of lab 3

To import any of these application at the appropriate stage do the following:

Remove current application

1. In **Workbench**, right-click your current **E-TicketMedical/1.0** application and select **Undeploy** from the menu. Click **Yes** to confirm.
2. Right-click the parent **E-TicketMedial** folder and click **Delete** from the menu.
3. When prompted check the **Also delete application from the server** and click **Yes**. After a while the application should disappear from your **Applications** panel.

Import application

4. Switch to **Internet Explorer** and login to the **Admin Console** by clicking on the **AdminUI** button. **User ID** is administrator, **Password** is password.
5. Click on **Services** in the **Home** page.
6. Click on **Application and Services** in the **Services** page.
7. Click on **Application Management** in the **Application and Services** page.
8. In the **Application Management** page, click the **Import** button.
9. In the **Application/Archive Import** page, click the **Browse** button and browse to **C:\MAX09L34\LCES\LCA**. Select one of the files listed above depending on which solution you want to import and click **Open**.
10. Back in the **Application/Archive Import** page, click **Preview**.
11. On the **Preview Page**, select **Deploy assets to runtime when import is complete** and click **Import**. The application will be imported into LiveCycle ES. You will return to the **Application Management** page and will see the **E-TicketMedical** application in the list.
12. Switch over to **Workbench**. From the menu select **Get Application**.
13. Select **E-TicketMedical/1.0** from the **E-TicketMedical** folder in the **Get Application** dialog and click **OK**. The application will appear in the **Applications** panel.
14. Unfold **E-TicketMedical/1.0** in the **Applications** panel and double-click **MedicalCheck**. You will see the orchestration appear in **Workbench**.